Contrôle d'accès(2)

- Organisationnel (identifier les besoins)
- **Personnel** (clauses de confidentialité...)
- Mécanique (Portes, badges, coffres...)
- Informatique

Sécurité d'un système d'information

- Définir une politique de sécurité.
- Préserver la confidentialité, l'intégrité et la disponibilité des données.
- Détecter les intrusions.

Objectifs du contrôle d'accès

- Préserver la confidentialité, l'intégrité et la disponibilité des données.
- Confidentialité : protection contre divulgation non autorisée intégrité des données : protection contre modification ou destruction non autorisées.
- Disponibilité : fournir les données aux utilisateurs autorisés quand ils en ont besoin.

3 propriétés importantes

- Authentification : assure que l'identité de l'utilisateur est légitime.
- Autorisation : détermine les tâches qu'un utilisateur est autorisé à faire.
- Non-répudiation : assure qu'un utilisateur ne peut pas nier avoir effectué une tâche.

Entités

- Sujets : entités actives du système
  - demandent des droits d'accès correspondant à l'autorisation d'exécuter des actions sur les objets
  - incluent toujours les utilisateurs du système
  - incluent souvent les processus en cours d'exécution pour le compte des utilisateurs
- Objets : entités passives du système
  - contiennent les informations et ressources à protéger
- Actions : moyens permettant aux sujets de manipuler les objets du système

Principe du moindre privilège

Chacun n'a pas plus de permissions que nécessaires pour effectuer sa tâche.

- Différents niveaux de permissions
- Différents niveaux de permissions à des moments différents
- Permissions limitées dans le temps

Entités

- Sujets :
  - Jean et Bob : médecins
  - Tom et Paul : infirmiers
  - Sarah : secrétaire médicale
- Objets :
  - patients
  - dossier : médical ou administratif
- Actions :
  - consulter, modifier le dossier médical d'un patient
  - créer le dossier médical d'un patient

Règles

Un sujet S a $\left.\begin{array}{l}\textit{la permission}\\ \textit{l'interdiction}\\ \textit{l'obligation}\end{array}\right\}$ de réaliser l'action A sur l'objet ou l'ensemble des objets O

moniteur de référence

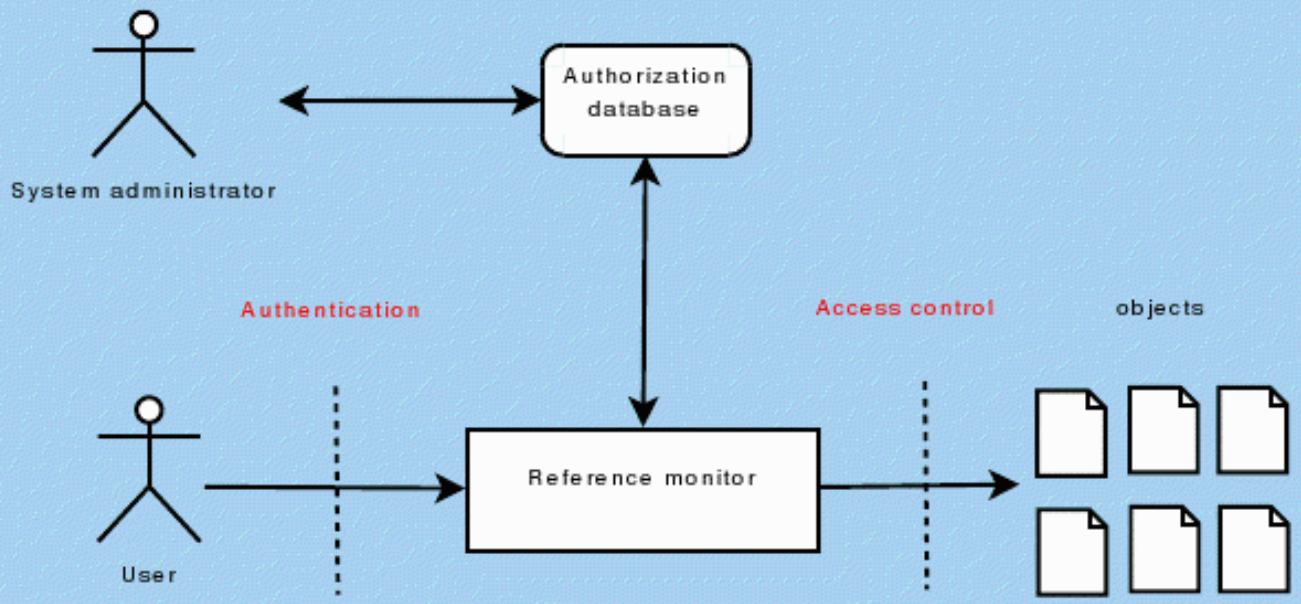
Overview

Terminology
- Overview
- Examples

Access Control Structures

Role-based Access Control

Reading material



Sujet , objets, permissions

- **Sujet** = entité active = utilisateur, processus, etc
- **Objet** = entité passive = fichier, socket, port, etc
- **Permission** = type d'accès = lire, écrire, exécuter, etc

Un **utilisateur lit** un **fichier**.
Un **processus écrit** dans une **socket**.
Alice peut **lire /etc/passwd**

$$\text{Sujet} \cap \text{Objet} \neq \emptyset$$

Matrice de droits d'accès (Lampson)

$$M = (M_{so})_{s \in S, o \in O} \quad \text{avec} \quad M_{so} \in P$$

- S ensemble des sujets
- O ensemble des objets
- P ensemble des permissions

	`bill.doc`	`edit.exe`	`fun.com`
Alice	$\emptyset$	{`execute`}	{`execute`, `read`}
Bob	{`read`, `write`}	{`execute`}	{`execute`, `read`, `write`}

Capability list

	`bill.doc`	`edit.exe`	`fun.com`
Alice	$\emptyset$	{`execute`}	{`execute`, `read`}
Bob	{`read`, `write`}	{`execute`}	{`execute`, `read`, `write`}

Alice : `edit.exe` : `execute` ; `fun.com` : `execute`, `read`

Bob : `bill.doc` : `read`, `write` ; `edit.exe` : `execute` ; `fun.com` : `execute`, `read`, `write`

Access control list

	bill.doc	edit.exe	fun.com
Alice	$\emptyset$	{execute}	{execute, read}
Bob	{read, write}	{execute}	{execute, read, write}

bill.doc : Bob :read, write

edit.exe : Alice :execute ; Bob : execute

fun.com : Alice :execute, read ; Bob : execute, read, write

Canaux cachés

- L crée un fichier f avec ses droits (bas)
- H modifie les permissions sur f en écrivant dedans
- L sait tester si les permission sur f on changé $\rightsquigarrow$ message booléen envoyé de H vers L

Type de contrôle d'accès

Access Control

- Discrétionnaire (DAC). Typiquement : UNIX.
  Un sujet à le droit de changer toutes les permissions d'accès aux objets qu'il contrôle.
- Obligatoire (Mandatory – MAC). Nondiscretionary
  Chaque sujets a un ensemble de permissions fixes, qu'il ne peut pas changer.

Le treillis militaire

Très Secret Défense $>$ Secret Défense $>$ Confidentiel Défense

s accède à o si level(s) $>$ level(o)

Politiques d'accès discrétionnaires (DAC)

Basées sur :

- Identité des utilisateurs
- Règles explicites d'accès Utilisateur autorisé à définir ses propres règlements de sécurité sur les informations dont ils est *propriétaire*.
  Attribue et en retire des droits aux autres utilisateurs
- Typiquement : UNIX
- Limitations : utilisateur mal intentionné ou incompétent

Politiques d'accès obligatoires (MAC, mandatory)

- Affectation aux objets et aux sujets d'*attributs* non modifiables par l'utilisateur
- Autorisation d'accès établie par l'examen de ces attributs de sécurité

Objectifs de sécurité formulés vis à vis des règles obligatoires décrites par la politique de sécurité

Les politiques multi-niveaux

- Basée sur la notion de treillis
- Objets et sujets classés selon différents niveaux de sécurité
- Niveau de sécurité = (niveau de classification, ensemble de catégories)
- Niveau de classification : ensemble totalement ordonné
- Ensemble de catégories : représente le domaine de l'information,
- Relation entre les ensembles : inclusion ensembliste
- Niveaux de sécurité : ensemble partiellement ordonné

Les politiques multi-niveaux

- Niveaux de sécurité : ensemble partiellement ordonné
- Relation de dominance : $\geq$
- Niveau $n_1 = (c_1, C_1)$ et $n_2 = (c_2, C_2)$
  $n_1 \geq n_2$ (n_1 domine n_2) ssi $n_1 \geq n_2$ et $C_2 \subseteq C_1$
- Exemple de politique multi-niveaux :

  le modèle de Bell et LaPadula
  modèle de Biba

Le modèle de Bell et LaPadula

Objectif : assurer la confidentialité

- Interdire toute fuite d'information d'un objet d'un certain niveau de classification vers un objet de niveau de classification inférieur
- Interdire à tout sujet d'une certaine habilitation d'obtenir des informations d'un objet de classification supérieur à cette habilitation

Comme le treillis militaire pour la lecture, mais écriture vers le haut

Le modèle de Bell et LaPadula

No Read up – simple property

Un sujet S peut *lire* un objet O seulement si le niveau de classification du sujet h(S) domine le niveau de classification de l'objet c(O).

$$(S, O, \text{lire}) \rightarrow h(S) \geq c(O)$$

No Write down – *-rule

Un sujet S peut écrire un objet O seulement si le niveau de classification du sujet h(S) est dominé par le niveau de classification de l'objet c(O).

$$(S, O, \text{écrire}) \rightarrow c(O) \geq h(S)$$

Le modèle de Biba

Objectif : assurer l'intégrité

- Interdire toute propagation d'information d'un objet situé à un certain niveau d'intégrité vers un objet de niveau d'intégrité inférieur
- Interdire à tout sujet situé à un certain niveau d'intégrité de modifier un objet possédant niveau d'intégrité supérieur

Le modèle de Biba

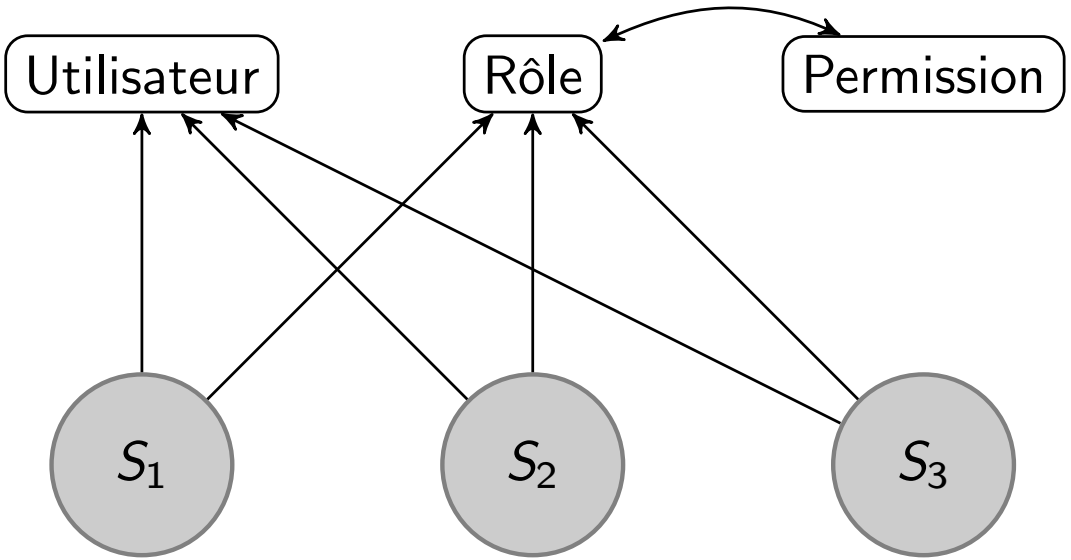
Règle simple

(inverse de Bell-LaPadula) Un sujet S ne peut accéder en lecture à un objet O seulement si le niveau de classification du sujet est dominé le niveau de classification de l'objet c(O).

*-rule

(inverse de Bell-LaPadula) Un sujet S ne peut accéder en écriture à un objet o seulement si le niveau de classification du sujet domine le niveau de classification de l'objet c(o)

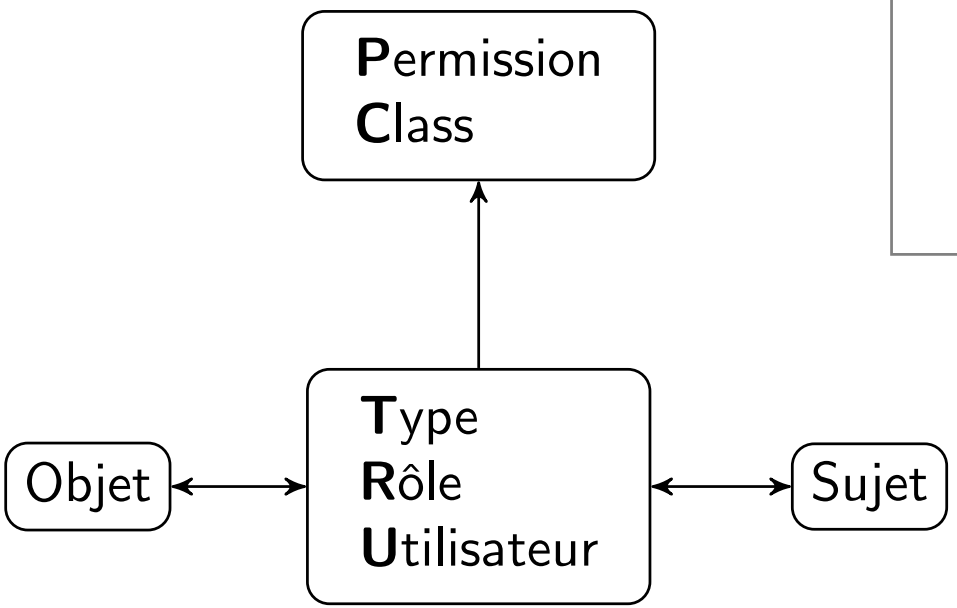
Role based Access control (RBAC)



Exemple : SELinux (NSA, MITRE)

- Module du noyau Linux, appelé à chaque appel système
- MAC + RBAC + Multi-level(optional)
- Permissions : Granularité extrême :
  - ex : écrire un fichier $\neq$ modifier dates de modifications
  - ex : chaque exécutable a ses propres règles de sécurité
  - Règle de sécurité par défaut pour linux : 22500 règles d'accès
  - RBAC = couche d'abstraction pour simplifier
- Comment se convaincre d'un invariant de sécurité (a n'envoie pas d'information à b [sans passer par c]) ?

SELinux – Security context $(T, R, U) \xrightarrow{c,p} (t', r', u')$




```
type t, t', t'';
user u types r, r';
allow t t' : c p
allow r r' : c' p'
allow r r' : c' transition
...
```

Model checking – Modèle

- Analyse des fichiers de règles de SELinux
- Modélisation des flots d'information par une fonction de transition
- États : SELinux Security Contexts $< \textit{type}, \textit{role}, \textit{user} >$
- $(t, r, u) \xrightarrow{c,p} (t', r', u') \rightsquigarrow$ transition $\begin{array}{l} (t, r, u) \longrightarrow (t', r', u') \\ (t, r, u) \longleftarrow (t', r', u') \end{array}$
- Sens de la flèche = sens du flot d'information, dépend de la permission :
  - $c = \texttt{read} : \longleftarrow$
  - $c = \texttt{write} : \longrightarrow$
  - $c = \texttt{exec} : \longleftarrow \ldots$

Model checking – Properties to check

- Ordonnancement
  "*État i sera toujours atteint avant l'état j*"
  Application : compartmented processing ; e-business (livraison après paiement...)
- Événements :
  "*État i sera toujours atteint depuis j en au plus n étapes*"
  Application : Nombre d'essais de login

Logique temporelle (CTL, Computation Tree Logic)

Logique classique + Chronologie gérée par des connecteurs supplémentaires :

- Propositions atomiques : $\Psi, \Phi ::= p|q|...|true|false$
- Connecteurs booléens : $::= \neg\Psi|\Psi \wedge \Phi|\Psi \vee \Phi|\Psi \Rightarrow \Phi|\Psi \Leftrightarrow \Phi.$
- Connecteurs temporels...
  - Prochine fois (Next Time) : $\mathcal{X}P$
    Property P holds in the second state of the path
  - Finalement (Eventually) : $\mathcal{F}P$
    Property P will hold at some state on the path (in the fututre)
  - Toujours (Always) : $\mathcal{G}P$
    Property P holds at every state on the path (globally)
    ...

Logique temporelle (CTL, Computation Tree Logic)

- Jusqu'à (Until) : $P\mathcal{U}Q$
  Property Q holds at some state on the path, and property P holds at all preceding states
- Release : $P\mathcal{R}Q$
  Property Q holds along the path up to and including the first state at which P holds (if it does at all)
- All Executions (quantificateur universel sur les chemins) : $\mathcal{A}P$
  Property P holds for all the paths
- At Least One Execution (Existentiel) : $\mathcal{E}P$
  There exist one path that satisfies property P.