Matrice d'accès

Master SEMS, 2013-2014

Pierre Paradinas

October 16, 2013

Le Concept de Matrice d'Accès

- Introduit en 1971 par Butler Lampson

Definition

On note O, l'ensemble des entités *objet* qui sont impliquées dans le système.
On note S, un sous ensemble des objets appelés *sujets* qui sont "actifs".
R un ensemble de *droits*. Un droit $r\,\varepsilon\,[s,o]$.
Les liens entre ces entités sont capturés dans la matrice d'accès A où les objets sont les colonnes et les sujets les lignes.

Le Concept de Matrice d'Accès

- Introduit en 1971 par Butler Lampson

Definition

On note O, l'ensemble des entités *objet* qui sont impliquées dans le système.
On note S, un sous ensemble des objets appelés *sujets* qui sont "actifs".
R un ensemble de *droits*. Un droit $r\,\varepsilon\,[s,o]$.
Les liens entre ces entités sont capturés dans la matrice d'accès A où les objets sont les colonnes et les sujets les lignes.

Le Concept de Matrice d'Accès (suite)

Example

	File 1	File 2	Process 1	Process 2
Process 1	r, w, o	r	r, w, e, o	w
Process 2	append	r, o	r	r, w, e, o

- L'interprétation dépend du système auquel s'applique la matrice
- r : read ; w : write ; o : owner ; e : execute ; etc.

Exercice

Hypothèse

- Soit les sujets : Alice, Bob et Cyndy
- Soit les objets (de type fichier) alicef, bobf et cyndyf appartenant respectivement aux 3 sujets ci-dessus
- De plus :
  - Alice peut lire bobf
  - Bob et Cyndy peuvent lire alicef
  - Cyndy peut lire et écrire bobf
  - Chaque propriétaire peut exécuter les fichiers qui lui appartiennent

Question 1

- Représenter la matrice d'accès

Exercice-Solution

Hypothèse

- Soit les sujets : Alice, Bob et Cyndy
- Soit les objets alicef, bobf et cyndyf appartenant respectivement aux 3 sujets ci-dessus
- De plus :
  - Alice peut lire bobf
  - Bob et Cyndy peuvent lire alicef
  - Cyndy peut lire et écrire bobf
  - Chaque propriétaire peut exécuter les fichiers qui lui appartiennent

Correction 1

	alicef	bobf	cyndyf	Alice	Bob	Cyndy
Alice	o,r,w,e	r		x		
Bob	r	o,r,w,e			x	
Cyndy	r	r,w	o,r,w,e			x

Exercice Suite

Hypothèse

- Soit les sujets : Alice, Bob et Cyndy
- Soit les objets alicef, bobf et cyndyf appartenant respectivement aux 3 sujets ci-dessus
- De plus :
  - Alice peut lire bobf
  - Bob et Cyndy peuvent lire alicef
  - Cyndy peut lire et écrire bobf
  - Chaque propriétaire peut exécuter les fichiers qui lui appartiennent

Question 2

Cyndy donne a Alice le droit de lire cyndyf, Alice retire le droit de lire à Bob sur ses fichiers.
Représenter la nouvelle matrice d'accès

Exercice-Solution

Matrice d'accès avant

	alicef	bobf	cyndyf	Alice	Bob	Cyndy
Alice	o,r,w,e	r		x		
Bob	r	o,r,w,e			x	
Cyndy	r	r,w	o,r,w,e			x

- Cyndy donne a Alice le droit de lire cyndyf, Alice retire le droit de lire à Bob sur ses fichiers

Matrice d'accès après

	alicef	bobf	cyndyf	Alice	Bob	Cyndy
Alice	o,r,w,e	r	***r***	x		
Bob	~~***r***~~	o,r,w,e			x	
Cyndy	r	r,w	o,r,w,e			x

Le Concept de Matrice d'Accès (suite)

- Le modèle de la matrice d'accès peut s'appliquer à différents niveaux:
  - Au niveau des services du SE (like in Unix)
  - A bas niveau du SE
  - Au niveau des utilisateurs d'un SGBD
- Le modèle de la matrice d'accès peut au niveau de la manière de réaliser le contrôle aller au delà du oui/non :
  - évaluation booléenne
  - évaluation historique
  - évaluation/calcul plus ou moins complexe

Exercice

Hypothèse

- Fonction de “lock” sur un fichier
- Un seul “écrivain” à la fois
- Plusieurs “lecteurs” possible mais aucun pendant l’action d’écriture

Solution

	lock	inc_lock	dec_lock	read_f	write_f
inc_lock					
dec_lock					
read_f					
write_f					

Exercice-Solution

Hypothèse

- Fonction de “lock” sur un fichier
- Un seul “écrivain” à la fois
- Plusieurs “lecteurs” possible mais aucun pendant l’action d’écriture

Solution

lock $\in \mathbb{Z}$, si *lock* ≥ 0 correspond aux lecteurs; si lock = -1 correspond à l’unique écrivain

	lock	inc_lock	dec_lock	read_f	write_f
inc_lock	inc(lock)	x			
dec_lock	dec(loc)		x		
read_f		yes, if lock ≥ 0		x	
write_f			yes, if lock = 0		x

Exercice-Mailing-List

Contexte

- Les intervenants :
  - Administrateurs des listes (toutes)
  - Créateurs de listes
  - Modérateurs de listes
  - Membres de listes
  - Personnes de l'organisation externe à la liste
  - Personnes extérieures à l'organisation
- Actions sur une liste
  - Création, destruction
  - Envois de courrier sur une liste

Questions

- Définir les règles de fonctionnement du système de "*Mailing List*"
- Préciser ce fonctionnement à l'aide de matrice d'accès

Exercice-Mailing-List

Contexte

- Les intervenants :
  - Administrateurs des listes (toutes)
  - Créateurs de listes
  - Modérateurs de listes
  - Membres de listes
  - Personnes de l'organisation externe à la liste
  - Personnes extérieures à l'organisation
- Actions sur une liste
  - Création, destruction
  - Envois de courrier sur une liste

Questions

- Définir les règles de fonctionnement du système de "*Mailing List*"
- Préciser ce fonctionnement à l'aide de matrice d'accès

Transition des états de protection

Il est intéressant d'étudier les matrices ses évolutions...

- On note $X_0 = (S_0, O_0, X_0)$, l'état initial du système
- Le passage d'un état i à i+1 est noté $X_i \vdash X_{i+1}$
- On notera $X \vdash^* Y$, l'état Y obtenu après une série de transition

Procédures de transition des états de protection

- On note **c** une commande ou préocédure de transition. Elle précisie l'entrée de la matrice A qui est changée et la nature du changement
- L'état initial du système X_0, on a $X_i \vdash c_{i+1}(p_{i+1},\ldots ,p_{i+m})X_{i+1}$
- Les commandes de bases ont été introduites par Harrison, Ruzzo & Ullman, " Protection in Operating Systems ", Communication of the ACM, 19(8), pp. 461-471, 1976 :

1. Création d'un sujet
2. Création d'un objet
3. Supprimer un sujet
4. Supprimer un objet
5. Ajouter un droit
6. Supprimer un droit

Procédures de création d'un sujet

Create subject s

- Préconditions :
  - $s \notin S$
- Post conditions :
  - $S' = S \cup \{s\}$, $O' = O \cup \{s\}$,
  - $(\forall\, y \in O')\ [a'[s, y]] = \emptyset]$, $(\forall\, x \in S')\ [a'[x, s]] = \emptyset]$,
  - $(\forall\, x \in S)\ (\forall\, y \in O)\ [a'[x, y]] = a[x, y]]$

Note

- Le sujet s ne doit pas exister avant dans la matrice en tant qu'objet ou sujet
- Aucun droit est ajouté

Procédures de création d'un sujet

Create subject s

- Préconditions :
  - $s \notin S$
- Post conditions :
  - $S' = S \cup \{s\}$, $O' = O \cup \{s\}$,
  - $(\forall\, y \in O')\ [a'[s, y]] = \emptyset]$, $(\forall\, x \in S')\ [a'[x, s]] = \emptyset]$,
  - $(\forall\, x \in S)\ (\forall\, y \in O)\ [a'[x, y]] = a[x, y]]$

Note

- Le sujet s ne doit pas exister avant dans la matrice en tant qu'objet ou sujet
- Aucun droit est ajouté

Procédures de création d'un objet

Create object o

- Préconditions :
  - o $\notin$ O
- Post conditions :
  - S' = S, O' = O U $\{o\}$,
  - $(\forall\, x \in S')$ [a' [x, o]] = $\emptyset$],
  - $(\forall\, x \in S)$ $(\forall\, y \in O)$ [a' [x, y]] = a [x, y]]

Note

- L'objet o ne doit pas exister avant dans la matrice en tant qu'objet
- Aucun droit est ajouté

Procédures de création d'un objet

Create object o

- Préconditions :
  - o $\notin$ O
- Post conditions :
  - S' = S, O' = O U $\{o\}$,
  - $(\forall\, x \in S')$ [a' [x, o]] = $\emptyset$],
  - $(\forall\, x \in S)$ $(\forall\, y \in O)$ [a' [x, y]] = a [x, y]]

Note

- L'objet o ne doit pas exister avant dans la matrice en tant qu'objet
- Aucun droit est ajouté

Procédures de suppression d'un sujet

Delete subject s

- Préconditions :
  - $s \in S$
- Post conditions :
  - $S' = S - \{s\}$, $O' = O - \{s\}$,
  - $(\forall\, y \in O')\ [a'\ [s, y]] = \emptyset]$, $(\forall\, x \in S')\ [a'\ [x, s]] = \emptyset]$,
  - $(\forall\, x \in S')\ (\forall\, y \in O')\ [a'\ [x, y]] = a\ [x, y]]$

Note

- La ligne et la colonne qui correspondent au sujet s sont supprimées

Procédures de suppression d'un sujet

Delete subject s

- Préconditions :
  - $s \in S$
- Post conditions :
  - $S' = S - \{s\}$, $O' = O - \{s\}$,
  - $(\forall\, y \in O')\ [a'\ [s, y]] = \emptyset]$, $(\forall\, x \in S')\ [a'\ [x, s]] = \emptyset]$,
  - $(\forall\, x \in S')\ (\forall\, y \in O')\ [a'\ [x, y]] = a\ [x, y]]$

Note

- La ligne et la colonne qui correspondent au sujet s sont supprimées

Procédures de suppression d'un objet

Delete object o

- Préconditions :
  - o $\in$ O
- Post conditions :
  - S' = S, O' = O - $\{o\}$,
  - $(\forall\, x \in S')$ [a' [x, o]] = $\emptyset$],
  - $(\forall\, x \in S')$ $(\forall\, y \in O')$ [a' [s, y]] = a [s, y]]

Note

- La colonne qui corresond à l'objet o est supprimée

Procédures de suppression d'un objet

Delete object o

- Préconditions :
  - o $\in$ O
- Post conditions :
  - S' = S, O' = O - $\{o\}$,
  - $(\forall\, x \in S')$ [a' [x, o]] = $\emptyset$],
  - $(\forall\, x \in S')$ $(\forall\, y \in O')$ [a' [s, y]] = a [s, y]]

Note

- La colonne qui corresond à l'objet o est supprimée

Procédures création d'un droit

Insert r *into* a[s, o]

- Préconditions :
  - $s \in S, o \in O$
- Post conditions :
  - S' = S , O' = O ,
  - a' [s,o] = a [s,o] U r ,
  - $(\forall x \in S')\ (\forall y \in O')$ [(x,y) $\neq$ (s,o) $\rightarrow$a'[x, y] = a [x,y]]

Note

- Le sujet s reçoit en plus le droit r sur l'objet o
- Les autres droits ne sont pas modifiés

Procédures création d'un droit

Insert r *into* a[s, o]

- Préconditions :
  - $s \in S, o \in O$
- Post conditions :
  - S' = S , O' = O ,
  - a' [s,o] = a [s,o] U r ,
  - $(\forall x \in S')\ (\forall y \in O')$ [(x,y) $\neq$ (s,o) $\rightarrow$a'[x, y] = a [x,y]]

Note

- Le sujet s reçoit en plus le droit r sur l'objet o
- Les autres droits ne sont pas modifiés

Procédures de suppression d'un droit

Delete r *into* a[s, o]

- Préconditions :
  - $s \in S', o \in O'$
- Post conditions :
  - S' = S , O' = O ,
  - a' [s,o] = a'[s,o] - r ,
  - $(\forall x \in S')\ (\forall y \in O')$ [(x,y) $\neq$ (s,o) $\rightarrow$a'[x, y] = a [x,y]]

Note

- Le droit r sur l'objet o est retiré au sujet
- Les autres droits ne sont pas modifiés

Procédures de suppression d'un droit

Delete r *into* a[s, o]

- Préconditions :
  - $s \in S', o \in O'$
- Post conditions :
  - S' = S , O' = O ,
  - a' [s,o] = a'[s,o] - r ,
  - $(\forall x \in S')\ (\forall y \in O')$ [(x,y) $\neq$ (s,o) $\rightarrow$a'[x, y] = a [x,y]]

Note

- Le droit r sur l'objet o est retiré au sujet
- Les autres droits ne sont pas modifiés

Exercice

Hypothèses

On considère les droits {r, w, e, list, modify et own}

Questions

- Doner avec les notations des procédures les fonctions suivantes :
- Commande *delete_all_rights (p,q,o)* t.q :
  - $(\forall\ q \in S, \forall\ o \in O)$ et $(p = a[q,o] \in A)$
- Commande *delete_part_rights* (s,o) t.q :
  - $(\forall\ p \in S, p =$ modify) et $(\forall\ o \in O)$
- Commande *delete_part_rights* (s,o) t.q :
  - $((\forall\ p \in S, p =$ modify) et $(\forall\ q \in S, q \neq$ own)) et $(\forall\ o \in O)$

Exercice-Solution

Hypothèses

On considère les droits {r, w, e, list, modify et own}

Solutions

- Commande *delete_all_right (q,o)* t.q :
  - ($\forall$ q $\in$ S, $\forall$ o $\in$ O) et (a[q,o] $\in$ A)
- Solution

```
command delete_all_right (p,q,o)
if p in a[q,o]
then
    Delete p into a[q, o]
end
```

Exercice-Solution

Hypothèses

On considère les droits {r, w, e, list, modify et own}

Solutions

- Commande *delete_part_right* (s,o) t.q :
  - ($\forall$ p $\in$ S, p= modify) et ($\forall$ o $\in$ O)
- Solution

```
command delete_all_right (p,q,o)
if p in a[q,o] and p = modify
then
    Delete p into a[q, o]
end
```

Exercice-Solution

Hypothèses

On considère les droits {r, w, e, list, modify et own}

Solutions

- Commande *delete_part_right* (s,o) t.q :
  - ((∀ p ∈ S, p= modify) et (∀ q ∈ S, q ≠ own)) et (∀ o ∈ O)
- Solution

```
command delete_all_right (p,q,o)
if p in a[q,o] and p = modify and q ≠ own
then
    Delete p into a[q, o]
end
```

Droits particuliers : copie, possession et atténuation des droits

Les droits spécifiques de copie et possesion sont spécifiques. Ils sont relatifs à ce que l'on nomme l'atténuation des droits.

> *Un sujet ne peut pas accorder un droit (sur un objet) qu'il ne possède pas à un autre sujet.*

Droit de copie ("copy right or grant right")

Seul le *possesseur* d'un droit peut le transmettre.
Le droit de transmettre peut être perdu au bénéfice de celui qui le reçoit ou simplement transmis (il est alors aussi conservé par le possesseur initial).

“Copy flag”

Cas du droit de copie (“copy flag”)

On peut transmettre un droit que l’on possède sur un objet même si cet objet ne vous appartient pas.

Dans le Sytem R (SGBD de IBM)

Sur les tables il y a des droits comme lecture, insertion, mise à jour pour manipuler les données de la table.
Il y a des droits sur la table comme la suppression.
Pour chaque droit il y a un droit de transmission (“grant option”) qui permet de transmettre ou pas ce droit. Il y a donc un “copy flag” par droit dans ce système.

Droit du posseseur

Cas du posseseur d'un objet ("own right")

Souvent le créateur d'un objet est le propriétaire de cet objet.
Le propriétaire d'un objet peut s'allouer plus ou moins de droit sur cet objet.

Atténuation des droits

Un sujet ne peut pas donner de droit sur un sujet qu'il ne possède pas.

Droit du posseseur

Cas du posseseur d'un objet ("own right")

Souvent le créateur d'un objet est le propriétaire de cet objet.
Le propriétaire d'un objet peut s'allouer plus ou moins de droit sur cet objet.

Atténuation des droits

Un sujet ne peut pas donner de droit sur un sujet qu'il ne possède pas.

Atténuation des droits

Exemple

Soit myfile (/home/bob/myfile) appartenant à Bob.
Bob ne possède pas le droit de lecture sur myfile.
Après la commande :
chmod go+r /home/bob/myfile
Bob peut lire myfile.
Si Alice lance la commande :
chmod go+r /home/bob/myfile
Échec de la commande !

Exercice

Hypothèses

On considère les droits {r, w, e, list, modify et own} c un “copy flag”

Questions

- Doner avec les notations des procédures les fonctions suivantes :
- Commande *copy_all_rights (p,q,s)* t.q :
  - $(\forall$ q $\in$ S) et (a'[q,s] = a[p,s] $\in$ A)
- Modifier la commande pour que seulement ceux ayant un copy flag copié
  - $(\forall$ q $\in$ S) et $(\forall p^c \neq \oslash) \longrightarrow$(a'[q,s] = a[p,s] $\in$ A et $q^c = \oslash$)
- Quelles conséquences ?

Exercice

Hypothèses

Le principe d'atténuation des droits

Discussions

- Quelles conséquences si le principe d'atténuation des droits est pas appliqué ?
- Quels droits peut obtenir un sujet (sans coopération, avec coopération) ?
- Si le principe est untilisé sur les droits du type read et write mais pas sur own et grant right ? Quelles conséquences ?

Un peu de théorie

La question

Soit un système informatique, comment déterminer si celui-ci est *sécurisé* ? ou encore existe-t-il un *algorithme* qui puisse répondre à la question ?

Autres questions

Que veut dire “sécurisé” ?
Quelle politique définit le fait d’être sûr !
In english, a *“safety system”* (système sûr au sens de la sécurité) se réfère au modèle abstrait du sytème alors que la notion de “secure” se réfère à l’implémentation du système.

Un peu de théorie

La question

Soit un système informatique, comment déterminer si celui-ci est *sécurisé* ? ou encore existe-t-il un *algorithme* qui puisse répondre à la question ?

Autres questions

Que veut dire “sécurisé” ?
Quelle politique définit le fait d’être sûr !
In english, a *“safety system”* (système sûr au sens de la sécurité) se réfère au modèle abstrait du sytème alors que la notion de “secure” se réfère à l’implémentation du système.

Définitions

Definition

Soit R un ensemble de droit R sur un système S.
La politique de sécurité stipule qu'aucun droit ne peut être ajouté au système ("no leak").
Si un droit r est ajouté à un élément du système alors qu'il n'existait pas avant, on dit que le droit r est "fuité" (leaked).

Definition

Si le system ne peut pas laisser un droit r être ajouté (fuité), ce système (y compris l'état initial s_0) est dit sûr. Si le système peut laisser ajouter un droit r (entrer dans un état non autorisé) il est dit non sûr par rapport au droit r.

Définitions

Definition

Soit R un ensemble de droit R sur un système S.
La politique de sécurité stipule qu'aucun droit ne peut être ajouté au système ("no leak").
Si un droit r est ajouté à un élément du système alors qu'il n'existait pas avant, on dit que le droit r est "fuité" (leaked).

Definition

Si le system ne peut pas laisser un droit r être ajouté (fuité), ce système (y compris l'état initial s_0) est dit sûr. Si le système peut laisser ajouter un droit r (entrer dans un état non autorisé) il est dit non sûr par rapport au droit r.

Exemple et théorème

Example

Prenons un système, où l'administarteur réseau qui peut lire les communications mais ne peut pas communiquer.
Ceci semble sûr !
Nénamoins, le système d'exploitation lui permet de créer un fichier et d'y écrire ce qu'il lit; il peut aussi donner à des tiers le droit de lire ce fichier...

Theorem

Il existe un algorithme qui peut déterminer si un système mono-opérationnel de protection avec un état initial s_0 est sûr par rapport à un droit r.

Proof.

"par énumération"

□

Exemple et théorème

Example

Prenons un système, où l'administarteur réseau qui peut lire les communications mais ne peut pas communiquer.
Ceci semble sûr !
Nénamoins, le système d'exploitation lui permet de créer un fichier et d'y écrire ce qu'il lit; il peut aussi donner à des tiers le droit de lire ce fichier...

Theorem

Il existe un algorithme qui peut déterminer si un système mono-opérationnel de protection avec un état initial s_0 est sûr par rapport à un droit r.

Proof.

"par énumération"

□

Exemple et théorème

Example

Prenons un système, où l'administarteur réseau qui peut lire les communications mais ne peut pas communiquer.
Ceci semble sûr !
Nénamoins, le système d'exploitation lui permet de créer un fichier et d'y écrire ce qu'il lit; il peut aussi donner à des tiers le droit de lire ce fichier...

Theorem

Il existe un algorithme qui peut déterminer si un système mono-opérationnel de protection avec un état initial s_0 est sûr par rapport à un droit r.

Proof.

"par énumération"

□

Théorème

Theorem

Il est indécidable de déterminer si un system S est sûr pour un droit générique r.

Proof.

"par équivalence du problème de l'arrêt d'un programme" □

Théorème

Theorem

Il est indécidable de déterminer si un system S est sûr pour un droit générique r.

Proof.

“par équivalence du problème de l’arrêt d’un programme” □

Modèle “Take grant”

Definition

On définit un grahe avec :
Les noeuds sont des sujets ou des objets;
Les arcs sont les droits d'un sujet sur un sujet ou un objet.

$$\bullet \xrightarrow{\quad t \quad} \circ \xrightarrow{\quad \beta \quad} \blacklozenge$$

On a un sujet $\bullet$ et sujet $\circ$; et l'objet $\blacklozenge$. Les droits t et β.

- On peut représenter ainsi la matrice de d'accès.

Exercice

Prendre la matrice d'accès des exemples précédents et représenter la sous forme de graphe.

Modèle “Take grant”

Definition

On définit un grahe avec :
Les noeuds sont des sujets ou des objets;
Les arcs sont les droits d'un sujet sur un sujet ou un objet.

$$\bullet \xrightarrow{\quad t \quad} \circ \xrightarrow{\quad \beta \quad} \blacklozenge$$

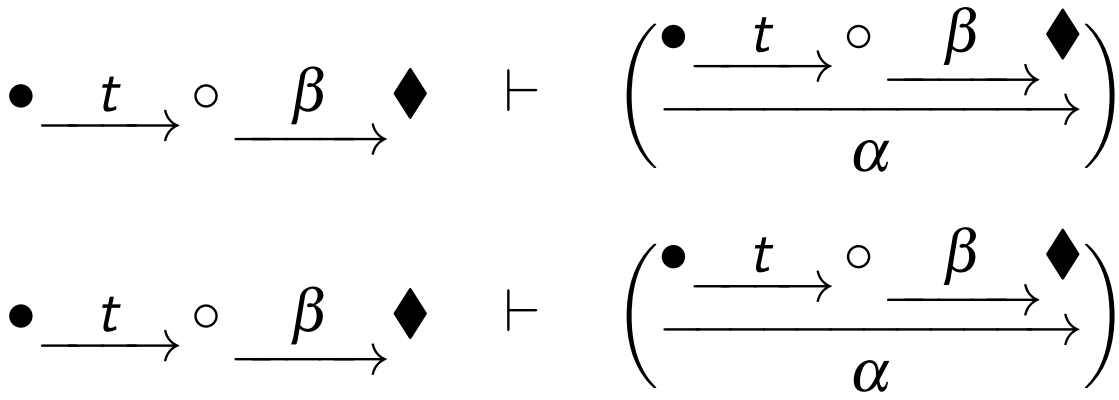
On a un sujet $\bullet$ et sujet $\circ$; et l'objet $\blacklozenge$. Les droits t et β.

- On peut représenter ainsi la matrice de d'accès.

Exercice

Prendre la matrice d'accès des exemples précédents et représenter la sous forme de graphe.

Modèle “Take grant”



Mécanismes de sécurité associés

- General concept of security mechanism

1. Access Hierarchies
2. Authorization list
3. Capability

Hiérarchie

Definition

Dans le cas des modèles hiérachique, les droits d'une entité résultent du fait que le sujet est membre d'un ensemble ou de sous ensemble de droit.

Example

Dans Unix, le mode "super user" fournit les droits sur tout les objets du système.

Le modèle Multics

L'approche hiérarchique est au centre de la protection dans Multics avec la notion d'anneau de protection.
A chaque processus est attaché un mot d'état.
Un mot d'état est un entier [0, r-1].
Chaque anneau définit un domaine A, j est un sous ensemble d'un anneau i, si pour tout $0 \leq i \leq j \leq$ r-1.
The supervisor state r = 2 ($\Longrightarrow$ *state*= 1 ou 0).

Question

Quid du principe du moindre privilèges quand un programme s'exécute en mode 0.
Que se passe t il si il y a une erreur en r= 0.

Le modèle Multics

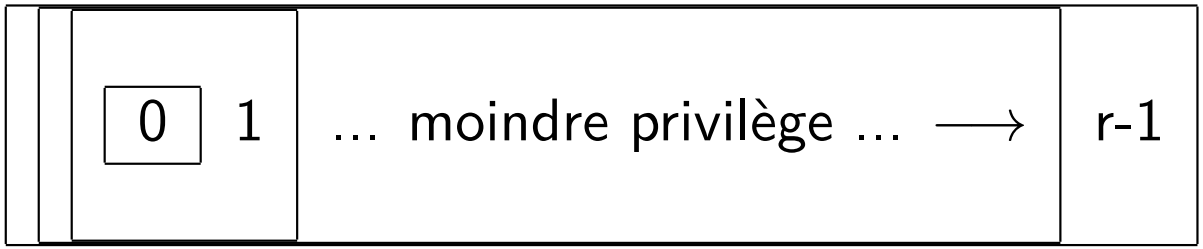
L'approche hiérarchique est au centre de la protection dans Multics avec la notion d'anneau de protection.
A chaque processus est attaché un mot d'état.
Un mot d'état est un entier [0, r-1].
Chaque anneau définit un domaine A, j est un sous ensemble d'un anneau i, si pour tout $0 \leq i \leq j \leq$ r-1.
The supervisor state r = 2 ($\Longrightarrow$ *state*= 1 ou 0).

Question

Quid du principe du moindre privilèges quand un programme s'exécute en mode 0.
Que se passe t il si il y a une erreur en r= 0.

Problème avec l'approche hiérarchique

L'approche hiérarchique :



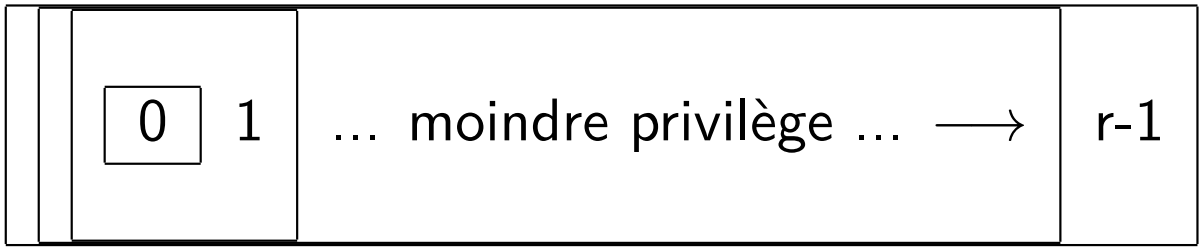
En cas de programme malicieux et/ou d'erreurs, un programme en niveau 0 pourra "modifier" la mémoire et/ou la changer.
Exemple classique sur PDP-11, sous certaines conditions lors d'un appel au mode superviseur en forçant le compteur de programme, le bit du mode privilege peut etre activé; au retour du call le bit reste en position mode privilège...

Néanmoins

- Pour autant ce mode de fonctionnement n'est pas à rejeter.
- Permet de faire de la protection à bas coût.
- Nécessiste des mécanimes de protection complémentaires

Problème avec l'approche hiérarchique

L'approche hiérarchique :



En cas de programme malicieux et/ou d'erreurs, un programme en niveau 0 pourra "modifier" la mémoire et/ou la changer.
Exemple classique sur PDP-11, sous certaines conditions lors d'un appel au mode superviseur en forçant le compteur de programme, le bit du mode privilege peut etre activé; au retour du call le bit reste en position mode privilège...

Néanmoins

- Pour autant ce mode de fonctionnement n'est pas à rejeter.
- Permet de faire de la protection à bas coût.
- Nécessiste des mécanimes de protection complémentaires

Liste d'accès

Les listes d'accès aussi appelée listes d'autorisation en anglais "*ACL*" sont des listes de sujets et des droits attachés à un objet.
Une liste d'accès correspond à une colonne de la matrice d'accès.

Exemple

- ACL of File 1 = ((Process 1; r/w/o), (Process 2; a)).

La révocation

- Si le droit du processus 2 à effcetuer la fonction Append, la liste devient :
  - ACL of File 1 = ((Process 1; r/w/o)).

Liste d'accès

Les listes d'accès aussi appelée listes d'autorisation en anglais "*ACL*" sont des listes de sujets et des droits attachés à un objet.
Une liste d'accès correspond à une colonne de la matrice d'accès.

Exemple

- ACL of File 1 = ((Process 1; r/w/o), (Process 2; a)).

La révocation

- Si le droit du processus 2 à effcetuer la fonction Append, la liste devient :
  - ACL of File 1 = ((Process 1; r/w/o)).

Capacité

Une capacité est un ticket que possède un sujet, ce ticket contient le(s) droit(s) de ce sujet sur cet objet. L'objet contrôle le ticket.
Plusieurs méthodes pour implémanter :

- Chaque mot de la mémoire a un tag. La sémantique du tag définit comment ce mot mémoire peut être uitilisé.
- Mémoire protégée de l'iAPX 432 (Intel).
- Un ticket est un certificat cryptographique qui peut être vérifié par l'objet.

Comparaison

Comparer ACL et capacité:

Discuter avantages et incovénients des deux aproches.

	ACL	Capacité
Implémentation-taille		
Implémentation-vérification		
Implémentation-révocation		