Protection, domaine et matrice d'accès

Lorsque l'on parle de **protection**, on parle de l'ensemble des mécanismes mis en place pour l'accès, la gestion des ressources systèmes par les processus, etc. Cette protection doit être fournie autant par le système que par les applications.

La **sécurité** en revanche concerne un spectre plus large incluant les virus, les attaques et la cryptographie.

La protection a pour but de prévenir les violations d'accès et d'améliorer la fiabilité (en détectant des erreurs humaines par exemple).

Si une ressource n'est pas protégée, elle peut être mal utilisée par des utilisateur·ice·s autorisé·e·s (ou non).

Une **politique** de protection correspond à la définition de ce qui doit être protégé. Tandis qu'un **mécanisme** de protection indique comment protéger.

Zones de protections

Sur l'ordinateur, il faut pouvoir protéger les processus et les **objets**.

Les objets peuvent être autant matériel (CPU, mémoire, imprimante, disque, etc) que logiciel (fichiers, programmes, sémaphore, etc).

Pour ce qui est des processus, il faut s'assurer que chaque processus ne peut accéder qu'aux ressources auxquelles il a l'autorisation. Il ne doit accéder qu'aux ressources qui sont nécessaires pour terminer sa tâche.

Domaine de protection

Un **domaine** définit un ensemble d'objet et de type d'opération qui peut être exécutée sur les objets (la possibilité d'exécuter une action sur un objet s'appelle un **droit d'accès**).

Un domaine est donc finalement une collection de droits d'accès.

Chaque processus opère à l'intérieur d'un domaine de protection. Les mêmes objets peuvent être référencés dans plusieurs domaines.

Un domaine peut être créé de plusieurs manières (par utilisateur, processus, procédure, etc). Et des opérations pour changer de domaine sont prévues.

L'association entre un processus et un domaine peut être **statique** (ne change pas, les droits d'accès restent donc tout le temps les mêmes), ou **dynamique** (les droits d'accès peuvent changer).

Matrice d'accès

La matrice d'accès est une représentation des domaines et de leurs droits d'accès.

C'est un tableau à deux dimensions où chaque ligne représente un domaine (aussi appelé rôle ou sujet), et chaque colonne représente un objet (aussi appelé asset).

Voici un exemple de matrice d'accès :

	Objets				
Sujets	Fichier 1	Segment 1	Segment 2	Processus 2	Editeur
Processus 1	Lire	Executer	Lire Ecrire		Entrer
Processus 2	Lire Ecrire				Entrer
Processus 3		Lire Ecrire Executer		Entrer	Entrer

La matrice d'accès permet de vérifier les politiques de protection voulues. Il faut donc définir les politiques pour chaque objet, assigner chaque domaine à l'exécution d'un processus.

Voyons maintenant quelques droits spécifiques,

Droit au changement de domaine

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

Pour représenter la permission de changer vers un autre domaine, il suffit de considérer les domaines aussi comme des objets. Ensuite pour permettre à D1 de pouvoir avoir les droits accès de D2, il suffit de mettre **switch** dans l'intersection de D1 et D2.

Droit à la copie de son droit

Le droit **copy** permet à un domaine de copier son droit pour un objet donné à un autre domaine.

Ce droit est représenté par une `*`.

domain \ object	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

Par exemple, ici, D2 peut copier le droit lecture de F2 sur un autre domaine que le sien.

Il existe aussi une variante du droit copy qui est le droit **transfert** (ou copie limitée). Si dans l'exemple précédent, il s'agit d'un droit de transfert, alors lorsque D2 passe son droit de lecture de F2 à un autre domaine, il perd le droit de lecture.

Droit à la modification des droits d'un objet

Le droit **owner** permet à un domaine de modifier n'importe quel droit pour un certain objet.

domain \ object	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

Dans cet exemple, D1 possède un accès total à F1. De cette manière, D1 peut par exemple s'accorder un droit d'écriture sur F1 ou encore accordé un droit de lecture de F1 à D2.

Droit au contrôle des droits d'un domaine

Le droit **control** permet à un domaine de supprimer des droits d'accès à un autre domaine.

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

Dans cet exemple, D2 peut par exemple supprimer le droit d'accès en écriture (F1) de D4.

Sous UNIX (setuid)

Sous UNIX, chaque domaine correspond à un utilisateur·ice, et lorsque qu'un exécutable est lancé, le processus prend le domaine de l'utilisateur·ice qui l'a lancé.

Sauf lorsque le bit `setuid` est mis. Ce bit est représenté par un `s` lorsque l'on regarde les permissions d'un fichier. Lorsque c'est le cas, l'exécutable va se lancer en tant que l'utilisateur·ice qui possède le fichier, à la place de s'exécuter en tant qu'utilisateur·ice qui l'a lancé.

Par exemple, avec le fichier sudo, lorsque l'on fait un `ls -l` dessus, on peut voir les permissions à gauche.

```
-r-s--x--x 1 root root 63432 Jan  6 09:22 /run/wrappers/bin/sudo
```

On peut voir qu'il y a un `s`, cela signifie que lorsque j'exécute ce fichier en tant qu'utilisateur `snowcode`, le fichier est réellement exécuté en tant que `root`.

Implémentation

Utiliser une matrice d'accès dans le système ne serait pas très pratique, car prendrait beaucoup de place et ne pourrait pas être mise en mémoire centrale.

C'est pourquoi, on utilise des **access list** à la place. L'access list est une liste associée à chaque objet. Elle contient des pairs de domaines et de droits. Un objet dont le domaine n'a pas de droit n'est pas présent sur la liste.

Il est également possible d'étendre la liste avec des droits par défaut (dans quel cas, pour vérifier les droits d'une opération, on vérifie d'abord les droits par défaut avant de rechercher la paire correspondante au domaine).

Révocation des droits

Ensuite, chaque système doit aussi prévoir comment révoquer des droits. Par exemple pour définir si c'est immédiat ou décalé, pour tous les utilisateurs ou seulement certains, si cela est temporaire ou permanent, etc.

Chaque système définit les stratégies possibles et laisse le choix à l'utilisateur·ice.

Revision #1
Created 6 January 2024 09:57:21 by SnowCode
Updated 6 January 2024 19:13:15 by SnowCode