TD1 – Modèles de contrôle d'accès

Exercice 1:

1. Compléter la description, vue en cours, du cas du système hiérarchique.
2. Compléter aussi la description des actions et des commandes pour le cas UNIX.

Exercice 2: On considère un système dans lequel on a trois utilisateurs, $U = \{a, b, c\}$ et trois fichiers, $F = \{x, y, z\}$. Les actions possibles sont la lecture et l'écriture d'un fichier par un utilisateur, ainsi que la création, et les droits d'accès sont o (possesion), r et w.

Supposons que

- a possède le fichier x, que b et c peuvent lire,
- b possède le fichier y que c peut lire et écrire, et a peut seulement lire,
- c possède le fichier z, qu'elle peut lire ou écrire et qui n'est accessible par personne d'autre.

1. Donnez la matrice de contrôle d'accès correspondante.
2. Écrivez une commande permettant à c de donner à a le droit de lecture sur z, et une autre commande permettant à b d'enlever à a le droit de lecture sur y.
3. Est-ce que les commandes précédentes peuvent être appliquées pour d'autres combinaisons de sujets/objets ?

Exercice 3: Dans la situation de l'exo précédent, on veut imposer une politique de sécurité dans laquelle a pourrait écrire dans le fichier z seulement s'il n'a pas lu un fichier dans lequel b aurait écrit auparavant. Bien-sûr on suppose que b ne peut pas écrire ni lire directement dans z !

Modéliser cette situation comme système de contrôle d'accès obligatoire, en rajoutant des permissions et en modifiant les commandes de l'exo précédent. Ne pas oublier que les commandes ne s'appliquent pas à des sujets/objets précis – elles ont des paramètres !

Exercice 4: Dans le modèle Take-Grant, montrer que si, dans la configuration initiale, $g \in M(x, y)$ et $g \in M(y, z)$ alors on peut arriver à une configuration dans laquelle $g \in M'(x, z)$.

Exercice 5: On considère un système d'exploitation Linux géré par un système de contrôle d'accès **typé**. Concevoir un système de types, une matrice initiale et des commandes permettant d'exécuter les actions suivantes :

- Tout processus de type `bash` peut lancer en exécution des fichiers exécutables contenant des programmes/applications relatives à un jeu, `game`.
- Les processus de type `game` ont le droit de lire/écrire des fichiers dans le répertoire `game-config` et ses sous-répertoires.
- Les processus de type `bash` n'ont pas le droit de lire/écrire dans ce type de fichiers, mais ont le droit de créer ces fichiers.

Pour résoudre cet exercice il faudrait se rappeler les mécanismes de création de processus et de fichiers en Linux !
