EPAI, Jérôme Frossard (2017)

AUTHENTIFICATION ET AUTORISATION

Systèmes d'exploitation

Système d'exploitation multi-utilisateur

- Permet l'utilisation d'un ordinateur par plusieurs utilisateurs
  - humains ou non,
  - simultanément ou non,
  - interactivement ou non.
- Doit permettre de :
  - **Authentifier les utilisateurs** : Vérifier que l'utilisateur est celui qu'il prétend être (nom de compte d'utilisateur) en lui demandant de présenter une ou plusieurs preuves de son identité (p. ex. un mot de passe).
  - **Autoriser l'accès aux ressources** : Permettre ou non à un utilisateur d'effectuer une action (lecture, écriture, exécution, etc.) sur une ressource (répertoire, fichier, etc.) en fonction de son identité ou de son appartenance à un groupe.

Authentification (*authentication*)

- Permet au système d'exploitation de s'assurer que l'utilisateur est celui qu'il prétend être.
- L'utilisateur doit amener la **preuve de son identité**.
- Un **facteur d'authentification** est un type de preuve.
- On distingue au moins trois facteurs d'authentification :
  - Quelque chose que l'on **sait** (mot de passe, PIN, etc.).
  - Quelque chose que l'on **possède** (téléphone portable, etc.).
  - Quelque chose qui fait partie de soi (données biométriques).
- Pour plus de sécurité, on peut utiliser une authentification à deux ou trois facteurs, le premier étant généralement un mot de passe (quelque chose que l'on sait).

Authentification par mot de passe

- Repose sur le fait que le mot de passe :
  - **n'est connu que de l'utilisateur et de lui seul**
  - ne peut pas être découvert à son insu
- Règles à observer :
  - Ne **jamais divulguer** son mot de passe.
  - Ne **jamais demander** à quelqu'un de divulguer son mot de passe.
  - Ne **jamais écrire** son mot de passe **de manière permanente,** quel que soit le support.
- Pour respecter ces règles :
  - **On ne stocke jamais un mot de passe**
  - On stocke une **empreinte** du mot de passe
  - Il ne doit pas être difficile de réaliser une table d'empreintes précalculées pour deviner le mot de passe (*rainbow table*)

Empreinte d'un mot de passe

- L'empreinte d'un mot de passe s'obtient avec une fonction de hachage de qualité cryptographique.
- Une fonction de hachage cryptographique doit :
  - Être difficile ou pratiquement impossible à inverser : si l'on a une empreinte, il doit être difficile ou pratiquement impossible de créer une donnée ayant cette empreinte.
  - Avoir un ensemble de valeurs suffisamment grand pour que la probabilité d'une collision soit aussi faible que possible : deux données différentes ne devraient pas avoir la même empreinte.

Compte d'utilisateur (*user account*)

- Les utilisateurs qui ont le droit d'utiliser un système sont répertoriés dans une base de données, par exemple :
  - Ruche (*hive*) **SAM** de la base de registre de Windows.
  - Fichiers **`passwd`** et **`group`** dans le répertoire **`etc`** de Linux.
  - Annuaire LDAP (OpenLDAP, Active Directory, etc.).
- Un compte utilisateur est essentiellement un enregistrement dans l'une de ces bases de données.
- Un compte utilisateur comprend typiquement :
  - Un identificateur (un entier ou un guid)
  - Le nom d'utilisateur (*username*)
  - L'empreinte du mot de passe (*password hash*)
  - Éventuellement d'autres informations comme le nom complet de l'utilisateur, son adresse de courriel, etc.

Compte de groupe (*group account*)

- Représente un groupe d'utilisateurs ayant un même rôle.
- Chaque utilisateur peut appartenir à plusieurs groupes.
- Permet de simplifier la gestion des permissions.
- Un compte de groupe est un enregistrement dans la base de données des utilisateurs.
- Un compte de groupe typique comprend :
  - Un identificateur (un entier ou un guid)
  - Un nom
  - Une liste de membres
  - Éventuellement d'autres informations comme une description

Session et authentification

- Une session est un intervalle de temps durant lequel un utilisateur utilise le système.
- Une session débute par la **connexion** de l'utilisateur (*login*) et se termine par sa **déconnexion** (*logout*).
- L'ouverture d'une session implique en général une demande d'authentification, mais ce sont deux actions bien distinctes.
- Il peut y avoir plusieurs demandes d'authentification au cours d'une même session.
- Il est important que les processus d'ouverture de session et d'authentification ne puissent pas être contournés.

Principe de moindre privilège

- Ce principe stipule qu'une fonctionnalité ne doit posséder que les privilèges strictement nécessaires à son exécution.
- Dans la plupart des cas, une fonctionnalité dispose des mêmes privilèges que l'utilisateur qui en a lancé l'exécution.
- C'est pourquoi il est important de travailler avec des privilèges administrateur que lorsque c'est nécessaire.
- L'UAC de Windows ou la commande **sudo** de Linux sont des exemples de moyens techniques dont le but est de faciliter l'application de ce principe.

Autorisation et permission

- Dans un système multi-utilisateur, on doit pouvoir spécifier pour chaque ressource les **permissions** (*permissions*) ou **droits d'accès** (*access rights*) qui sont accordés à chaque utilisateur ou groupe d'utilisateurs.
- Le système d'exploitation utilise ces information pour **autoriser** ou non une action qu'un utilisateur souhaite effectuer sur une ressource.
- Le **contrôle d'accès** désigne l'utilisation des permissions pour autoriser ou non l'accès à une ressource.

Permissions et système de fichiers

- Dans le cas des fichiers et des répertoires, on peut généralement accorder les permissions suivantes :
  - Fichier : lire, écrire, exécuter
  - Répertoire : liste, traverser, créer des fichiers et des répertoires
- Pour que cela soit possible, le **système de fichiers** doit permettre de stocker ces permissions pour chaque fichier et chaque répertoire.

Permissions POSIX : Classes d'utilisateurs

- Dans les systèmes POSIX, on ne spécifie pas les permissions de chaque utilisateur et group d'utilisateurs individuellement, mais les permissions de trois classes d'utilisateurs :
  - Utilisateur propriétaire (*owning* ***u****ser* ou **u**)
  - Utilisateurs du groupe propriétaire (*owning* ***g****roup* ou **g**)
  - Tous les autres (***o****ther* ou **o**)
- On peut utiliser ce type de permissions avec les systèmes de fichiers compatibles POSIX (extFS, HFS+, NTFS, etc.).
- Pour spécifier le propriétaire et le groupe propriétaire d'un fichier ou d'un répertoire, on utilise la commande `chown` :
  - `chown myowninguser:myowninggroup /my/file`
  - `chown myowninguser:myowninggroup /my/directory`

Permissions POSIX : Fichiers

- Pour les fichiers, on peut spécifier les permissions suivantes pour chaque classe d'utilisateurs :
  - Lire (*read* ou **r**)
  - Écrire (*write* ou **w**)
  - Exécuter (*execute* ou **x**)
- Pour spécifier les permissions d'un fichier, on utilise la commande **chmod**. Exemples :
  - **`chmod u+w /my/file`**
    (ajoute la permission w à l'utilisateur propriétaire)
  - **`chmod g-x /my/file`**
    (enlève la permission x au groupe propriétaire)
  - **`chmod 751 /my/file`**
    (donne les permissions **`rwxr-x--x`** avec trois chiffres octaux)

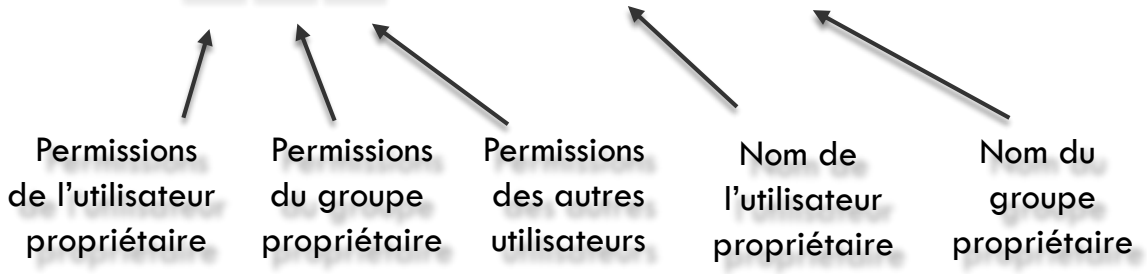
Permissions POSIX : Répertoires

- Pour les répertoires, on peut spécifier les mêmes permissions que pour les fichiers, mais le sens est un peu différent :
  - Lire (*read* ou **r**) : lister le contenu
  - Écrire (*write* ou **w**) : créer des fichiers et des répertoires
  - Exécuter (*execute* ou **x**) : traverser
- Pour spécifier les permissions d'un répertoire, on utilise la commande **chmod**. Par exemple :
  - **`chmod u+w /my/directory`**
    (ajoute la permission w à l'utilisateur propriétaire)
  - **`chmod g-x /my/directory`**
    (enlève la permission x au groupe propriétaire)
  - **`chmod 751 /my/directory`**
    (donne les permissions **`rwxr-x--x`** avec trois chiffres octaux)

Permissions POSIX : Afficher les permissions

- Pour afficher les permissions des éléments d'un répertoire, on utilise la commande : `ls -l`
- Exemple de résultat de la commande :

```
-rw-r--r--@   1 myuser mygroup  811B  4 jan  2017 Gemfile
-rw-r--r--@   1 myuser mygroup  1,1K  4 jan  2017 Gemfile.lock
-rw-r--r--@   1 myuser mygroup  1,2K 13 sep 12:47 _config.yml
drwxr-xr-x@   6 myuser mygroup  192B 14 fév  2017 _drafts/
drwxr-xr-x@  11 myuser mygroup  352B  4 jan  2017 _includes/
drwxr-xr-x@   7 myuser mygroup  224B 21 sep 15:42 _layouts/
drwxr-xr-x@  11 myuser mygroup  352B 24 oct 18:46 _posts/
drwxr-xr-x@   4 myuser mygroup  128B  6 jan  2017 _sass/
drwxr-xr-x@  18 myuser mygroup  576B 21 nov 13:25 _site/
```



Listes de contrôle d'accès (ACL)

- Une liste de contrôle d'accès (ACL) est une liste de longueur variable dont chaque entrée définit les permissions d'un utilisateur ou d'un groupe d'utilisateur et qui est associée à un fichier ou à un répertoire.
- On peut utiliser des ACL avec les systèmes POSIX, mais elles ne sont pas installée par défaut et sont rarement utilisées en pratique.
- Sous Windows, les ACL sont le seul moyen de définir les permissions et on les gère en utilisant l'onglet sécurité des propriétés des fichiers et des dossiers.

Avantages et inconvénients des ACL

- Avantages :
  - Contrôle plus fin des permissions
  - Supporte l'héritage de permissions (les fichiers et répertoires héritent des permissions de leur parent)
  - Permets de refuser explicitement une permission
- Inconvénients :
  - Plus difficile à mettre en œuvre
  - Erreurs parfois difficiles à repérer
  - Les droits effectifs d'un utilisateur sont difficiles à déterminer
  - Peuvent donner un faux sentiment de sécurité