GALLAND Cyprien

VATIN Clément

TP 4 : Utilisateurs, groupes et permissions

Exercice 1 : Gestion des utilisateurs et des groupes

Linux est un système conçu dès l'origine comme étant multi utilisateurs : Il est donc possible de définir plusieurs utilisateurs sur un même appareil, et de les activer simultanément. Ces utilisateurs sont répartis dans des groupes : au sein de ces groupes, on définit des droits pour les différents utilisateurs. Chaque utilisateur possède un groupe primaire par défaut, et peut être ajouté à plusieurs groupes secondaires.

Dans ce TP, toutes les commandes (ou presque) nécessitent sudo, même si non écrit explicitement dans le compte rendu.

1) On commence par creer deux groupes, nommés groupe1 et groupe2 :

```
sudo addgroup groupe1

sudo addgroup groupe2
```

Pour vérifier la bonne création des groupes, on utilise `cat /etc/group | tail -n 2`

2) On crée ensuite 4 utilisateurs u1, u2, u3, u4, en demandant la création de leur dossier personnel et avec bash pour shell :

```
sudo useradd u1 --create-home --shell /bin/bash

sudo useradd u2 --create-home --shell /bin/bash

sudo useradd u3 --create-home --shell /bin/bash

sudo useradd u4 --create-home --shell /bin/bash
```

Pour vérifier : sudo cat /etc/passwd | tail -n 4 affiche les 4 derniers utilisateurs crées. Ils ont un uid allant de 1000 à 1001 (uid : cf questions ultérieures).

Useradd tient en une seule ligne, alors que adduser nécessite plusieurs input => Pas adapté pour la ligne de commande. Ainsi les deux commandes ont des résultats identiques, mais ne s'utilisent pas dans les mêmes conditions.

3) On ajoute les utilisateurs dans les groupes créés selon la répartition suivante : u1, u2 et u4 dans groupe1, u2, u3 et u4 dans groupe2 :

```
usermod -a -G groupe1 u1

usermod -a -G groupe1 u2

usermod -a -G groupe1 u4

usermod -a -G groupe2 u2

usermod -a -G groupe2 u3

usermod -a -G groupe2 u4
```

4) Voici deux moyens d'afficher les membres de groupe2 :

La première consiste à utiliser la commande getent.

```
getent group groupe2
```

La commande `getent` affiche les entrées des bases de données prises en charge par les bibliothèques du Name Service Switch (NSS)

La deuxième consiste quand à elle à utiliser grep pour chercher dans le fichier /etc/group. Ce fichier contient la liste des utilisateurs appartenant aux différents groupes, sous la forme :

nom du groupe / champ special / numero de groupe / liste des membres

En utilisant grep pour rechercher les occurrences de "groupe2" dans ce fichier, on aura donc la liste de tout les utilisateurs membres de groupe2.

```
cat /etc/group | grep "groupe2"
```

5) On souhaite faire de groupe1 le groupe propriétaire de /home/u1 et /home/u2 et de groupe2 le groupe propriétaire de /home/u3 et /home/u4. Pour ce faire on va utiliser l'ensemble de commandes suivant :

```
sudo chgrp groupe1 /home/u1

sudo chgrp groupe1 /home/u2

sudo chgrp groupe2 /home/u3

sudo chgrp groupe2 /home/u4
```

La commande `chgrp` permet de changer le groupe d'utilisateur possédant un fichier ou un dossier.

6 On veut modifier les groupes primaires des utilisateurs comme suit : groupe1 pour u1 et u2, groupe2 pour u3 et u4.

```
usermod --gid groupe1 u1

usermod --gid groupe1 u2

usermod --gid groupe2 u3

usermod --gid groupe2 u4
```

`usermod` modifie les fichiers d'administration des comptes du système selon les modifications qui ont été indiquées sur la ligne de commande. Ici, --gid change le groupe primaire d'un utilisateur.

7) 8) On crée maintenant deux répertoires /home/groupe1 et /home/groupe2 pour le contenu commun aux groupes, et on met en place en place les permissions permettant aux membres de chaque groupe d'écrire dans le dossier associé :

```
sudo mkdir /home/groupe1

sudo mkdir /home/groupe2

sudo chgrp -R root /home/groupe1

sudo chgrp -R root /home/groupe2

sudo chgrp groupe1 /home/groupe1

sudo chgrp groupe2 /home/groupe2

ls -l /home #Pour afficher le résultat

sudo chmod 1766

ls -l /home #Pour afficher le résultat
```

On obtient drwxrw-rwT. Ce T est un bit particulier appelé sticky bit; il indique que dans ce dossier 1 , seul le propriétaire d'un fichier, le propriétaire du dossier ou root ont le droit de renommer ou supprimer ce fichier. Il est activé par le 1 du chmod.

Ce chmod distribue les droits rwx (lecture écriture et exécution) au propriétaire, aux autres membres du groupe et aux autres utilisateurs.

Le droit r équivaut à 4, w équivaut à 2 et x équivaut à 1. Chacun des 3 derniers chiffre vaut pour une catégorie d'utilisateur (1er chiffre après le 1 = propriétaire : ici, 7 = 4+2+1. De même les deux chiffres suivants correspondent respectivement aux autres membres du groupe et aux autres utilisateurs.

9) On cherche à se connecter en tant que utilisateur u1 :

```
su u1
```

On nous demande un mot de passe. Or ce dernier n'a pas été défini : par conséquent la connexion est impossible : il faut activer l'utilisateur, en lui affectant un mot de passe.

10) On active le compte de l'utilisateur u1 en lui affectant un mot de passe :

```
sudo passwd u1

#On nous demande un mot de passe, on le rentre

#on nous demande de le confirmer on le confirme.
```

On vérifie que la connexion à u1 est désormais possible :

```
su u1

#On tape le mot de passe

whoami
```

On est bien u1 !

11) On cherche l'uid et le gid de u1 (identifiants d'utilisateur et de groupe).

```
id -u u1 #donne leréexecuter uid (1001)

id -g u1 #donne le gid (1001)
```

12) On souhaite connaitre l'utilisateur qui a pour uid 1003.

```
cat /etc/passwd | grep "1003"
```

Cette commande affiche toute la ligne. On vérifie que le 1003 correspond bien au 1er numéro (uid) et non au deuxieme (gid).

L'utilisateur concerné est u3.

13) On cherche l'id du groupe 1 :

```
cat /etc/group | grep "groupe1"
```

ici, c'est 1001.

14) On cherche désormais le groupe qui a pour guid 1002 (Rien n'empêche bien sur d'avoir un groupe dont le nom est 1002) :

```
cat /etc/group | grep "1002"

#on regarde les sorties, on voit laquelle est la bonne

#La bonne est identifiée comme celle ou le numéro après le x: est le bon numéro
```

15) On retire l'utilisateur u3 du groupe groupe2.

```
deluser u3 groupe2
```

Cela est impossible car ce groupe est le groupe primaire de u3. Il faut donc changer le groupe primaire de u3 d'abord, et ensuite exécuter de nouveau la commande (cf question 6).

16) On modifie le compte de u4 de sorte que : il expire au 1 er juin 2020, qu'il faille changer de mot de passe avant 90 jours, attendre 5 jours pour modifier un mot de passe, que l'utilisateur soit averti 14 jours avant l'expiration de son mot de passe et que le compte soit bloqué 30 jours après expiration du mot de passe :

```
sudo usermod -e 2020-06-01 u4

sudo chage -m 90 u4

sudo chage -M 5 u4

sudo chage -W 14 u4

sudo chage -I 30 u4
```

17) On veut savoir quel est l'interpréteur de commandes (Shell) de l'utilisateur root :

```
cat /etc/passwd
```

La fin de chaque ligne contient le shell. On cherche donc l'utilisateur root, et on a la réponse (/bin/bash)

18) L'utilisateur *nobody* correspond à l'utilisateur possédant le moins de droit sur le système. Cela fait que si il est compromis, les dégâts aux systèmes sont minimaux.

19) Par défaut, la commande sudo conserve notre mot de passe en mémoire pendant 15 minutes. Cette option est modifiable avec la commande

```
Defaults    env_reset,timestamp_timeout=X
```

Ou X est la durée en minute de conservation mémoire voulue.

Il est donc possible de forcer sudo à oublier le mot de passe avec la commande :

```
sudo -k
```

Exercice 2 : gestion des permissions

1) Dans notre $HOME, on crée un dossier test, et dans ce dossier un fichier fichier contenant quelques lignes de texte.

```
cd ~

mkdir test

nano test/fichier

# On tape des lignes de texte dans le fichier
```

Quels sont les droits sur test et fichier ?

```
ls -l

ls -l test/
```

Sur test : proprietaire et membre du même groupe en *rwx*, autres utilisateurs en *rx*

Sur fichier : proprietaire et membre du même groupe en *rw*, autres utilisateurs en *r*

2) On retire tous les droits sur ce fichier, y compris pour nous, puis on passe en root :

```
chmod 000 fichier

ls -l test/ # On voit alors bien que les droits sont nuls pour tous

cat fichier #permission denied ; c'est normal

sudo cat fichier # et la on peut afficher le fichier

sudo vim fichier # et on peut même le modifier
```

On en conclut donc que l'utilisateur root conserve tout les droits, quoi qu'il arrive.

3) On se redonne les droits en écriture et lecture sur fichier (`chmod 600 fichier`), puis on exécute la commande :

`echo "echo Hello" > fichier`.

En affichant le contenu du fichier, on voit que le contenu à bien été remplacé, comme vu lors des TP précédents. C'est donc une commande qui nécessite les droits, ce qui ets normal.

4) En essayant d'exécuter le fichier, on s'aperçoit que ça ne marche pas. En revanche avec sudo c'est possible. Cela est logique car on n'a pas rétabli les droits d'exécution. On peut faire `chmod 700 fichier` pour réparer

cet oubli.

5) On se place dans le répertoire test, et on se retire le droit en lecture pour le fichier fichier, avant d'afficher le contenu du fichier.

```
cd test

chmod 300 fichier

cat fichier
```

Le système refuse de nous afficher le contenu. C'est normal, puisque nous n'avons plus les droits en lecture.

En utilisant `echo >`, on peut toujours écrire dedans.

Cherchons enfin à exécuter le fichier : Cela n'est plus possible.

On en déduit donc que le système, pour exécuter un fichier, doit pouvoir lire son contenu (c'est logique, d'ailleurs).

Ainsi, la commande `chmod 100` n'a aucune utilité. Par contre on peut bien sur disposer seulement du droit en lecture, sans droit d'exécution.

6) On crée dans test un fichier*nouveau*, ainsi qu'un répertoire *sstest*, en leur retirant à tout deux les droits en écriture.

```
touch nouveau

mkdir sstest

chmod 500 nouveau

chmod 500 sstest
```

On va désormais tenter de modifier le fichier*nouveau* avec la commande echo.

Cela n'est pas possible.

Cependant, avec `vim wq! nouveau`, cela devient possible : En effet, en tant que propriétaire du fichier (car créateur), il est possible de récupérer les droits que l'on s'était enlevé, et vim le gères nativement (le *!* demande à vim de récupérer les droits en écriture).

On rétablit les droits sur le répertoire test, et on tente de nouveau de modifier ou supprimer le fichier : Cela est impossible : On en déduit que la modification des droits du répertoire parent n'entraîne pas celle des répertoires/fichiers enfants.

7) On retourne dans notre répertoire personnel `cd home`), puis on se retire le droit en exécution du répertoire test (`chmod 600 test`).

On tente alors de créer, supprimer, ou modifier un fichier dans le répertoire test, de nous y déplacer, d'en lister le contenu, etc…

- La commande `ls` renvoie un accès refusé dans chacun des répertoires (et nous les liste ensuite)
- La commande `ls -l` trouve les fichiers, ainsi que les répertoires, mais ne nous trouve pas les droits
- La commande `cat` ne s'execute pas non plus

Il apparaît donc que le droit d'exécution du répertoire est nécessaire pour presque toutes les actions usuelles : aller dans le répertoire avec cd, lire ou écrire un fichier, ...

En fait toutes les commandes executées dans un repertoire sont executées avec les droits de l'utilisateur sur ce répertoire.

8) On rétablit le droit en exécution du répertoire test (`chmod 700 test`). On se positionne alors dans ce répertoire et on retire à nouveau le droit d'exécution (`chmod 600 test`).

On essaye à nouveau de créer, supprimer et modifier un fichier dans le répertoire test, de vous déplacer dans ssrep, de lister son contenu,... Cela est toujours impossible : les droits du répertoire courants en execution sont indispensables pour éxecuter une commande. En revanche, `cd ..` fonctionne et permet de retourner dans le répertoire précédent : Cela permet de ne pas rester bloqué dans le répertoire en ne pouvant rien faire faute de droits suffisants.

9) On rétablit le droit en exécution du répertoire test, puis on attribue au fichier fichier les droits suffisants pour qu'une autre personne de notre groupe puisse y accéder en lecture, mais pas en écriture :

```
chmod 700 test

cd test

chmod 740 fichier
```

10) Un umask (user file creation mode mask) permet de définir les droits par défaut d'un nouveau fichier / dossier. Il s'agit d'une valeur (en octal) qui indique les droits que N'AURA PAS le fichier / dossier. (C'est donc le complément à 777 (pour un dossier, et 666 pour un fichier) du chmod).

On définit un umask très restrictif qui interdit à quiconque à part nous l'accès en lecture ou en écriture, ainsi que la traversée de nos répertoires, puis on teste sur un nouveau fichier et un nouveau répertoire :

```
mkdir umask_restreint

cd umask_restreint

umask 177

touch umask_restreint/test
```

Pour tester les droits on utilise `umask -S`, dans le dossier créé, qui affiche les droits sous forme détaillée (en précisant explicitement la catégorie d'utilisateur et les droits associés)

11) On définit désormais un umask très permissif qui autorise tout le monde à lire nos fichiers et traverser nos répertoires, mais n'autorise que nous à écrire, avant de le tester sur un nouveau fichier et un nouveau répertoire :

```
mkdir umask_permissif

cd umask_permissif

umask 177

touch umask_permissif/test
```

12) Enfin, on définit un umask équilibré qui nous autorise un accès complet et autorise un accès en lecture aux membres de notre groupe, puis on le teste.

```
mkdir umask_equilibre

cd umask_equilibre

umask 177

touch umask_equilibre/test
```

13) On va maintenant faire la transcription entre des commandes en notation octale et des commandes en notation classique :

notation classique	notation octale	droits originaux de fic
chmod u=rx,g=wx,o=r fic	chmod 532	inconnus
chmod u0+w, g-rx	chmod 602	r - -r - x - - - <=> 450
chmod u-x,g+r,o+w	chmod 653	711 <=> r w x - - x - - x
chmod u+x,g=w,o-r	chmod 520	r - - r - x - - - <=> 450

14) On conclut le tp en affichant les droits sur le programme passwd :

```
ls -l /etc/passwd
```

Qui nous affiche `-rw-r--r--`

Ainsi les droits en écriture sont détenus par le seul propriétaire (root), mais les droits en lecture sont commun à tout les utilisateurs.

Le propriétaire en question est root.

Ce fichier est très important pour l'identification : il est donc nécessaire que n'importe qui ne puisse pas le modifier. En revanche, son affichage ne pose pas de failles de sécurité, car les données sensibles sont dans /etc/shadow et non dans /etc/passwd.