Laboratoire 4 - Linux

1 Durée prévue : 2h00

2 Permissions d'accès aux fichiers

Tout fichier d'un système Linux appartient à la fois à un utilisateur (son "propriétaire") et à un groupe. Ainsi, pour chaque fichier ses utilisateurs potentiels est scindé en 3 catégories :

1. **u**, l'utilisateur normal, son propriétaire, bien souvent son créateur, qui n'a pas pour autant tous les droits sur le fichier !
2. **g**, son groupe, ensemble d'utilisateurs ayant parfois des "permissions" particulières.
3. **o**, tous les autres (others).

Attention, l'utilisateur propriétaire et le groupe propriétaire du fichier peuvent être indépendants :

Le groupe propriétaire n'est pas forcément le groupe primaire de l'utilisateur propriétaire (voir Labo1 – Linux) et même, le propriétaire n'est pas forcément membre du groupe ! Mais (heureusement) une règle générale simple s'applique à la création de tout nouveau fichier (ou dossier) son propriétaire est l'utilisateur (humain ou système) qui l'a créé et son groupe est le groupe primaire de ce même utilisateur.

2.1 Droits d'accès aux fichiers

Linux permet de spécifier les droits d'action sur un fichier, que peuvent exercer les utilisateurs des 3 catégories précédentes, ou plutôt les permissions d'accès que leurs accordent les fichiers et les dossiers.

Linux a reprend 3 niveaux de protections sur les fichiers et les répertoires. Leur notation symbolique est :

I. **r**, lecture
II. **w**, écriture
III. **x**, exécution

De façon générale, ces permissions sont consultables par la commande : `ls -l`

`ll` étant un alias plus court, pour la commande `ls -l`

Essayez cette commande dans votre home directory :

```
mangon@kali-2020:~$ ll .profile
-rw-r--r-- 1 mangon student 807 Feb 25  2020 .profile
```

On trouve de gauche à droite :

- le 1er caractère indique la nature du fichier (voir point 2.4 Les types de fichiers *ci-dessous*)
- le système de droits est spécifié symboliquement par les 9 attributs suivants, correspondants aux 3 catégories d'utilisateurs du fichier.

```
. . . | . . . | . . .
u     g     o
```

  - La section **u** fixe les droits accordés au propriétaire du fichier.
  - La section **g** fixe les droits accordés aux utilisateurs faisant partie du groupe auquel appartient le fichier.
  - La section **o** fixe les droits des autres utilisateurs.

- le nombre de liens sur le fichier. **1** signifie que le fichier n'a aucun lien qui pointe vers lui, **2** (ou plus) signifiant qu'il existe un lien (ou plus) vers lui.
- le nom de l'utilisateur propriétaire du fichier **mangon**
- le nom du groupe propriétaire du fichier **student**
- la taille du fichier en octets **807**
- la date de la dernière modification du fichier **Feb 25 2020**
- le nom complet du fichier

2.2 Permissions des fichiers normaux

Pour chaque fichier, les utilisateurs sont ainsi séparés en 3 catégories, le propriétaire **u**, les membres du groupe **g** et tous les autres **o**.

Les permissions accordées par le fichier à ces catégories sont complètement indépendantes mais leur signification est la même. Vis à vis de chacune de ces 3 catégories, on trouve dans l'ordre :

- le droit de lecture , afficher son contenu → **r** si permis , **-** si refusé
- le droit d'écriture , modifier son contenu → **w** si permis , **-** si refusé
- le droit d'exécution , pour un fichier script ou binaire → **x** si permis , **-** si refusé

Essayez ces 3 commandes à partir de votre home directory :

```
mangon@kali-2020:~$ ll /etc/init.d/sysstat
-rwxr-xr-x 1 root root 1582 Jul 14 11:39 /etc/init.d/sysstat
```

Le fichier script de démarrage **/etc/init.d/sysstat** possède les droits **rwxr-xr-x**
Tous les utilisateurs ont donc le droit de lire et d'exécuter ce fichier (ce qui est à éviter) ? Seul son propriétaire, le super utilisateur **root** peut le modifier.

```
mangon@kali-2020:~$ ll /etc/fstab
-rw-r--r-- 1 root root 840 Sep 19 01:03 /etc/fstab
```

La table de montage **/etc/fstab** possède les droits **rw-r—r**, elle peut donc être lue par tous, modifiée uniquement par **root**

```
mangon@kali-2020:~$ stat /etc/passwd
  File: /etc/passwd
  Size: 3311            Blocks: 8          IO Block: 4096   regular
file
Device: fe00h/65024d    Inode: 659749      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2020-10-09 16:26:23.511293530 +0200
Modify: 2020-09-24 18:16:14.570427917 +0200
Change: 2020-09-24 18:16:14.570427917 +0200
 Birth: -
```

La commande **stat** permet d'obtenir une information plus poussée sur un fichier.

2.3 Permissions des répertoires

Pour les répertoires, la signification des attributs est différente de celle d'un fichier normal. Mais elle est toujours identique pour les 3 catégories d'utilisateurs du répertoire. La présence d'un tiret **-** signifie toujours l'absence complète de droits.

- **r** autorise la lecture du contenu, la liste des fichiers (avec **ls** ou **dir**)
- **w** autorise de modifier le contenu : droits de créer et de supprimer des fichiers dans le répertoire (avec **cp**, **mv**, **rm**, etc.)
- **x** autorise l'accès aux fichiers du répertoire et de s'y déplacer (avec **cd**). Si on attribue **w**, il faut attribuer aussi **x** sur le répertoire pour permettre de le parcourir.

Exécutez ces deux commandes et répondez aux questions ci-après :

Passer les commandes **cd /**puis **ls -lLd ***, pour lister les répertoires situés à la racine.

A qui appartiennent les fichiers dans la racine du système de fichiers ?

Un utilisateur quelconque peut-il y créer des sous-répertoire ?

Commentez les 2 cas particuliers **`/root`**, **`/lost+found`** et **`/tmp`**.

Attention ! On voit que le droit **W** est très étendu, et même dangereux quand il est accordé à un groupe, car un membre du groupe peut supprimer des fichiers dont il n'est pas propriétaire et sur lesquels il n'a même pas de droit d'écriture !

Le droit **X** sur un répertoire est un **préalable indispensable** pour qu'un utilisateur de la catégorie (**u**, **g** ou **o**) correspondante au positionnement du **X**, puisse exercer d'éventuels droits sur les fichiers contenus dans le répertoire.

2.4 Les types de fichiers

Sous Linux, tout est fichier, même un répertoire ou des périphériques ! Les droits d'accès déterminent donc la possession d'un fichier ou d'un répertoire ou d'un périphérique à un utilisateur, ou à un groupe d'utilisateurs.
Avec la commande **`ls -l`** et le premier caractère désigne le type d'éléments :

La lettre « **d** » → désigne un directory (ou un dossier)

```
drwxr-xr-x 2 mangon student 4096 Oct  7 13:34 Documents
```

Le trait d'union « **-** » → désigne un fichier normal

```
-rw-r--r-- 1 mangon student   37 Oct  9 16:30 liste.txt
```

La lettre « **l** » → désigne un lien symbolique

```
lrwxrwxrwx 1 root root 7 Sep 19 01:03 /bin -> usr/bin
```

La lettre « **c** » → désigne un périphérique caractères

```
crw--w----  1 root tty       5,   1 Oct  9 16:26 console
```

La lettre « **b** » → désigne un périphérique caractères

```
brw-rw----  1 root disk      8,   0 Oct  9 16:26 sda
```

2.5 Exercice 1 – Les droits d'accès

1. Quels sont les droits sur votre répertoire personnel (pour rappel **`cd /home`** pour se déplacer dans le dossier parent de votre home directory) ?
2. Un utilisateur différent hacker peut-il y pénétrer ou seulement lister vos fichiers ? Expliquez pourquoi.
3. Pour tester vos affirmations, vous pouvez changer d'utilisateur dans votre shell via la commande **`sudo su – hacker`** et essayer d'accéder à vos données. Attention, ne restez pas connecté en tant que hacker sur votre shell, tapez **`exit`** !
4. Et l'utilisateur louis, le pourrait-il ? Sachant qu'il ne fait pas partie du groupe student ? Expliquez pourquoi.

3 Changements des droits

- De façon générale, l'utilisateur qui crée un fichier en devient le propriétaire, et le groupe primaire de celui-ci, au moment de la création, devient le groupe du fichier.
- Les droits accordés au propriétaire, au groupe et aux autres dépendent du masque des droits.
- Et **root** n'est pas soumis à ces restrictions, il a le pouvoir absolu sur le système de fichiers. En contrepartie il peut être considéré comme responsable de tout dysfonctionnement !

3.1 Changer le propriétaire ou le groupe propriétaire

- Changer le propriétaire
  chown [-R] nv-user fichiers
  Commande réservée au propriétaire actuel des fichiers ou des répertoires (et à **root**)
  L'option **-R** (récursif) permet d'agir sur l'ensemble des sous-répertoires.
  Exemple :
  chown -R mangon /home/stage1
- Changer le groupe propriétaire
  chgrp [-R] nv-groupe fichiers
  Ceci doit être effectué par **root** ou le propriétaire, à condition que celui-ci soit membre du nouveau groupe.
  Exemple :
  chgrp -R student /home/stage1
- Changer les 2 en même temps
  chown nv-user:nv-groupe fichiers
  Dans ce cas, en plus, le groupe propriétaire des fichiers est changé.

3.2 Changer les permissions sur les fichiers

Les droits d'accès peuvent être modifiés par le propriétaire des fichiers ou par **root**.

Les paramètres de la commande **chmod** peuvent s'écrire de plusieurs façons équivalentes, sur le modèle **chmod *droits fichiers***. Le paramètre ***droits*** permettant de calculer les nouveaux droits d'accès. Ceux-ci peuvent s'obtenir de façon **relative**, par ajout (symbole **+**) ou retrait (**-**) par rapport aux droits existants, ou bien de façon absolue, en fixant les nouveaux droits qui remplacent les anciens (symbole **=**).

On désigne ainsi le changement de permission sur le modèle « à quelle(s) catégorie(s), quelle action, quel(s) droit(s) » sont alors notés **chmod [u g o a] [+ - =] [r w x] fichiers**

- **u**, **g** et **o** les 3 catégories d'utilisateurs (**user**, **group**, **other**) et de plus par **a** (=all) toutes les catégories d'utilisateurs.
- **+ - =** l'action d'ajouter, de retirer ou de fixer un droit, qui s'applique.
- **r**,**w**,**x** désignant les droits d'accès concernés

par exemple **chmod u+x fichier** signifie « ajouter le droit d'exécution au propriétaire de fichier ».

On peut regrouper les catégories si on veut exercer la même action

Par exemple :

- **`chmod ug+w fichier`** « ajouter le droit d'exécution au propriétaire et au groupe »
- **`chmod go-rwx fichier`** « enlever tous droits d'accès à tous les utilisateurs, sauf au propriétaire »
- **`chmod u=rwx,g=rw,o=r fichiers`** « remplace les permissions précédentes des fichiers, en les fixant à **`-rwxrw-r--`** »
- **Attention : aucun espace dans la liste des droits, pas même autour des éventuelles virgules**
- **`chmod u=rwx,g=r fichiers`** « fixe les permissions à **`-rwxr--???`** en ne changeant pas les permissions précédentes sur la catégorie **other** »
- **`chmod u=rwx,g=r,o= fichiers`** « fixe les permissions à **`-rwxr-----`** »

Le "super-utilisateur" `root` n'est pas soumis aux restrictions des permissions.

3.3 Notation octale des permissions

Il existe une autre façon d'indiquer les permissions de chaque catégorie d'utilisateurs, plus simple en utilisant la numération octale.

Voici la table de correspondance entre les 8 chiffres en numérotation octale (base 8) et les 8 valeurs de droits fichiers. Par convention la présence d'un droit est noté 1, l'absence 0.

```
Binaire ----- Droit ----- Octal
 000 -------- (---) ------- 0
 001 -------- (--x) ------- 1
 010 -------- (-w-) ------- 2
 011 -------- (-wx) ------- 3
 100 -------- (r--) ------- 4
 101 -------- (r-x) ------- 5
 110 -------- (rw-) ------- 6
 111 -------- (rwx) ------- 7
```

Les droits d'accès s'obtiennent alors comme suit (liste non exhaustive):

`rwxrwxrwx` → `chmod 777 fichier`
`rwxrwxrw-` → `chmod 776 fichier`
`rwxrwxr--` → `chmod 774 fichier`
`rwxrwx---` → `chmod 770 fichier`
`rwxrw----` → `chmod 760 fichier`

`rwxr-----` → `chmod 740 fichier`
`rwx------` → `chmod 700 fichier`
`rw-------` → `chmod 600 fichier`
`r--------` → `chmod 400 fichier`
`---------` → `chmod 000 fichier`

3.4 Exercice 2 – manipulations des droits d'accès

Souvenez-vous que l'utilisateur **root** n'a aucune restriction en cas de mauvaise manipulation sur les droits d'accès aux fichiers. Toutefois, changer les droits d'accès sur des dossiers ou fichiers systèmes est dangereux et peut compromettre la sécurité de votre système d'exploitation. Pour cela évitez à tout prix d'utiliser le compte **root**.

Pour changer d'utilisateur utilisez les commandes :

sudo su – hacker → bascule vers l'utilisateur hacker, exit pour revenir à votre utilisateur.
sudo su - → bascule en root, mais à utiliser avec précaution)

1. Exécutez les commandes pour changer les droits sur votre dossier personnel pour retirer tous droits d'accès à vos données à l'utilisateur **louis**.
2. Donnez les trois manières d'écrire la commande **chmod** pour réaliser l'opération précédente.
3. Sans utiliser le compte **root**, faite de même pour appliquer les mêmes droits sur le home directory de l'utilisateur **hacker**.
4. Comparer les permissions de **/etc/passwd** et **/etc/shadow**. Pourquoi a-t-on nommé ainsi ce dernier fichier ?
5. **hacker** peut-il lire le fichier **/etc/shadow** ?
6. Visualiser sa présence dans le dossier **/etc** ?
7. Examiner son contenu ?
8. A partir de votre compte utilisateur, essayez de faire une copie de **/etc/shadow** sous le nom **~/shadow.bak** ! Vérifiez les droits sur la copie du fichier et concluez !
9. Toujours à partir de votre compte utilisateur, essayez de faire une copie de **/etc/passwd** sous le nom **~/passwd.bak** ! Vérifiez les droits sur la copie du fichier et concluez !
10. Dans le fichier **~/passwd.bak**, supprimez la ligne de l'utilisateur **louis** soit avec l'éditeur **nano** ou **vi** si vous le maitrisez.
11. Essayez de faire la copie inverse **~/passwd.bak** vers **/etc/passwd**. Concluez !
12. En vous connectant sous le compte **root** faite maintenant une copie de **/etc/shadow** dans votre home directory, sous le nom **/home/*[VOTRE_COMPTE]*/shadow.bak**
13. Déplacez-vous dans le dossier **/home/*[VOTRE_COMPTE]*** et accordez-vous la propriété de la copie. Comment réalisez-vous ces opérations ?
14. Revenez à votre compte utilisateur et vérifiez si vous avez l'accès au contenu du fichier **~/shadow.bak** en modifiant quelques lignes de son contenu.
15. Supprimez cette copie **~/shadow.bak**, car celle-ci contient des données sensibles et représente une faille sécurité sur votre système d'exploitation.
16. Avec l'utilisateur **hacker** ou votre propre utilisateur, pouvez-vous créer le répertoire temporaire **/home/temp** ? essayez ! pourquoi ?
17. Effectuez cette création comme **root**.
18. Accordez les permissions maximales sur **/home/temp** et vérifiez.
19. Avec l'utilisateur **hacker**, tout content d'avoir enfin un droit d'écriture dans **/home/temp** essayez de copier les 2 fichiers système **/etc/hosts** et **/etc/passwd** dans **/home/temp** ? Avez-vous les droits suffisants pour le faire ? pourquoi ? Qu'affiche la commande **ll /home/temp** ?

20. L'utilisateur hacker doit vous donner les accès sur ces deux copies dans **/home/temp**. Mais il veut retirer ses propres accès à ces deux fichiers. Comment s'y prend-t-il ? Réalisez l'opération.

4 Le masque de protection umask

Rappelons les règles simples de propriété qui s'appliquent à la création d'un fichier ou d'un répertoire :

- son propriétaire est l'utilisateur qui l'a créé
- son groupe est le groupe primaire de ce même utilisateur

Mais quelles sont les permissions attribuées par défaut à l'utilisateur propriétaire, au groupe propriétaire et à tous les autres ?

- Les permissions maximales accordées par un fichier sont **666 -rw-rw-rw-**
- Les permissions maximales accordées pour un dossier sont **777 -rwxrwxrwx**

On peut restreindre ces permissions par défaut lors de la création de fichiers ou de répertoires. C'est le rôle de la commande **umask** de fixer les permissions masquées, autrement dit les droits non accordés aux fichiers et répertoires lors de leur création.

Exemple de calcul de permissions effectives, affectées lors de la création d'un répertoire, par un utilisateur dont le masque de protection est **027**

```
  777 = 111 111 111 permissions maxi = rwx rwx rwx
- 027 = 000 010 111 masque de protection
= 750 = 111 101 000 permissions effectives = rwx r-x ---
```

La commande **umask** affiche le masque de l'utilisateur actif.

4.1 Exercice 3 – droits d'accès par défaut

1. Donnez le **umask** de votre utilisateur sous forme octal.
2. Que seront les droits d'accès par défaut lors de la création d'un fichier avec ce **umask** ? Expliquez votre réponse.
3. Que seront les droits d'accès par défaut lors de la création d'un dossier avec ce **umask** ? Expliquez votre réponse.