Utilisateurs et groupes

FICHIERS

/etc/passwd
Informations sur les comptes des utilisateurs.
/etc/shadow
Informations sécurisées sur les comptes utilisateurs**.**
/etc/group
Informations sur les groupes.
/etc/default/useradd
Valeurs par défaut pour la création de comptes.
/etc/skel/
Répertoire contenant les fichiers par défaut.
/etc/login.defs
Configuration de la suite des mots de passe cachés « shadow password ».

Gestion des utilisateurs

http://www.commentcamarche.net/contents/646-linux-gestion-des-utilisateurs
http://www.systemx.fr/linux/systeme/gestioncomptes/adminusers.html

1) créez un utilisateur [initiale de votre prenom][nom de famille]
ex : nmace
a) avec les paramètres par défaut

```
# useradd nmace
```

b) observez le résultat, et déduisez en les paramètres par défaut, puis supprimer cet utilisateur

```
# cat /etc/passwd | grep nmace
# userdel nmace
```

c) recréer cet utilisateur avec les paramètres suivant :

- répertoire personnel /home/[login] créé automatiquement et contenant
- Téléchargements
- TPs
- Cours
- date d'expiration : 1er Décembre 2013
- shell : /bin/bash
- commentaire : mon utilisateur
- uid : 1200
- groupe primaire : idem que login
- groupe secondaire : users

```
# cd /etc/skel
# mkdir Telechargement TPs Cours
# useradd -m -e "2013-12-01" -s /bin/bash -c "Ceci est un utilisateur" -u
1200 -G users nmace
```

d) modifiez le mot de passe de cet utilisateur afin qu'il puisse s'authentifier
```
# passwd nmace
```

e) s'authentifier avec ce compte sur une autre console
ctrl + alt + F2

2) créer un utilisateur système "lpic" ne pouvant pas s'authentifier sur le système
rmq : pensez à une double sécurité

```
# useradd -r -s /bin/false lpic
# usermod -L lpic ou # passwd -l lpic
```
(ils le sont déjà par défaut)

3) modifier l'utilisateur "lpic" afin de lui permettre de s'authentifier, avec le mot de passe de votre choix

```
# usermod -s /bin/bash lpic
```
/!\ pas d'option -p car sinon le mot de passe est en clair
```
# passwd lpic
```

4) vérouiller puis dévérouiller l'utilisateur de l'exercice 1

```
usermod srazat -L
controler > less /etc/shadow
unlock > usermod srazat -U
```

5) supprimer ces deux comptes utilisateur
```
# userdel -r nmace
# userdel lpic
```

Bonus 1] étudier et modifier la configuration par défaut de useradd
cf useradd -D et /etc/login.defs (lire le MAN)
$ man useradd
y son présentés : les fichiers de configuration (dont /etc/default/useradd, le fichier de configuration des paramètres par défaut à la création d'un utilisateur), les options de ces fichiers (exemples : CREATE_HOME=yes qui permet d'inverser le comportement par défaut pour la création du répertoire personnel, c'est a dire -m par défaut), ainsi que l'option -D et ses options (exemple : -D -e "date")
$ man login.defs
configuration générale de l'authentification sur notre système
exemple : UID_MIN qui permet d'indiquer l'UID en dessous duquel les comptes seront considérés comme appartenant à des utilisateurs système

Modifier les valeurs par défaut

When invoked with only the **-D** option, **useradd** will display the current default values. When invoked with **-D** plus other options, **useradd** (/etc/default/useradd) will update the default values for the specified options. Valid default-changing options are:

Tapez :
`useradd -D`
> affiche la liste des options configurées
`useradd -D -s /bin/bash` (on modifie la valeur --shell à…)

-b, **--base-dir** REP_BASE
The path prefix for a new user´s home directory. The user´s name will be affixed to the end of BASE_DIR to form the new user´s home directory name, if the **-d** option is not used when creating a new account.

-e, **--expiredate** DATE_FIN_VALIDITE
Date à laquelle le compte utilisateur sera désactivé.

-f, **--inactive** DUREE_INACTIVITE
Nombre de jours après la fin de validité d´un mot de passe avant que le compte ne soit désactivé.

-g, **--gid** GROUPE
Nom de groupe ou identifiant numérique du groupe initial d´un nouvel utilisateur. Le groupe spécifié doit exister, et un identifiant de groupe numérique doit déjà exister.

-s, **--shell** INTERPRETEUR
The name of a new user´s login shell.

Bonus 2] *tentez* de reproduire ce TP avec adduser
Default configuration file for adduser and addgroup : **/etc/adduser.conf**
`adduser -D` (même option disponibles que useradd soit --s,e,-b,-f,-g,)

Lire : man adduser

Gestion des mots de passe

1) ajouter un utilisateur "lpic", lui définir un mot de passe :
```
useradd lpic
passwd lpic
```
puis établir les paramètres shadow suivant :

- date d'expiration : 30 Décembre 2013
- nombre max de jours d'inactivité : 10
- validité maximum d'un mot de passe (en jours) : 20
- nombre de jours durant lesquels l'utilisateur sera averti de devoir modifier son mot de passe avant désactivation de son compte : 5

```
$chage -E 2013-12-31 -I 10 -M 20 -W 5 lpic
```
Pour vérifier > `#chage -l lpic`

2) modifier la date du dernier changement de mot de passe de cet utilisateur, pour que le système considère que celui-ci n'a pas été changé depuis :

- 17 jours `$chage -l 2013-09-13 <user>`
- le 1er Janvier 2010 `$chage -l 2010-01-01 <user>`

Que ce passe-t-il dans ces deux cas ?

```
$chage -l 2013-09-1 <user>
su lpic
```

> il nous signale que notre mot de passe va expirer dans 3 jours

```
$chage -l 2010-01-01 <user>
su lpic
```

> il nous signale que notre compte est désactivé

3) afficher /etc/shadow, et vérifiez bien connaître le sens de chaque colonne (éventuellement en utilisant plusieurs fois la commande chage pour expérimenter)

Gestion des groupes

1) ajouter le groupe "tps"

```
#groupadd tps
#cat /etc/group | grep tps
```

2) faites en sortes que :
lpic ait "tps" pour groupe primaire

```
#usermod -g tps lpic
# less /etc/passwd
# cat /etc/group | grep 1003
tps:x:1003:
```

votre utilisateur (ex : debian ou nmace) ait "tps" comme groupe secondaire

```
# usermod -a -G tps srazat
# cat /etc/group | grep tps
tps:x:1003:srazat
```

-a étant pour ajouter, sans quoi -G tps remplacera toute la liste des groupes secondaires de “srazat” par "tps". Il ne sera donc plus membre que de tps, en plus de son groupe primaire (et ce n'est pas ce qui était demandé).

pour vérifier :

```
# su srazat
$ groups
```

affiche la liste des groupes dont vous êtes membres, en mettant en premier le groupe principal ou le groupe rejoint via newgrp

3) modifier le mot de passe de ce groupe

```
#gpasswd tps
```

4) tentez une commande newgroup tps en tant que votre utilisateur

```
# su srazat
$ newgrp tps
```
aucune mot de passe n'est demandé car srazat est déjà membre de ce groupe.

5) supprimer votre utilisateur de la liste de membres de ce groupe, et retentez l'opération précédente

```
# gpasswd -d srazat tps
```

on aurait aussi pu faire :
`# gpasswd -M "" tps` qui supprime toute la liste des membres de ce groupe

Contrôles, test :

```
# su srazat
$ groups
```
tps n'apparait pas
```
$ newgrp tps
```
un mot de passe nous est demandé
```
$ groups
```
tps apparait bien en premier
```
$ exit
$ groups
```
tps n'apparait plus

6) ajouter votre utilisateur comme administrateur de ce groupe, puis supprimer son mot de passe avec cet utilisateur, et enfin, tentez de réutiliser la commande newgrp tps

```
# gpasswd -A "srazat" tps
```
attention : **-A** remplace toute la liste des administrateurs. Si vous aviez déjà des administrateurs, il faudra donc écrire toute la liste précédement configurée, avec en plus l'utilisateur. Ex :
```
# gpasswd -A "lpic,srazat" tps
# su srazat
$ gpasswd -r tps
```
j'ai bien le droit car je suis administrateur du groupe
à présent, seuls le champ mot de passe (seconde collonne de /etc/gshadow) est vide. Le mot de passe sera donc toujours considéré comme faux, ce qui implique que seul les membres du groupe pourront utiliser la commande newgrp sur celui-ci. Le résultat est exactement le meme qu'avec -R (restrict), qui remplace le mot de passe par "!".

en revanche, je peu être administrateur sans être membre
```
# gpasswd -d srazat tps
# su srazat
$ newgrp tps
```
un mot de passe m'est demandé (même si, en tant qu'administrateur, j'ai très bien pu le modifier juste avant sans connnaitre le précédent)

Utilisateurs, fichiers et droits d'accès

```
#chown [option] [user][:group] [fichier/dossier]
-R > recursive
```

Exercice :

1) dans votre dossier personnel, créer, en tant que votre utilisateur, un dossier test, contenant les fichiers fichier1, fichier2, fichier3 et le dossier dossier1 contenant lui même un fichier fichier4.

```
cd, touch, mkdir
```

ces fichiers et dossiers ont alors pour propriétaire et groupe propriétaire votre utilisateur et son groupe principal.

2) tenter, en tant que votre utilisateur, de rendre root le propriétaire du fichier test/fichier1

```
$ chown root ~/test/fichier1
```
chown: modification du propriétaire de «fichier1»: Opération non permise

3) rééffectuer cette opération en tant que root

```
# chown root test/fichier1
$ ls -l fichier1
-rw-rw-r-- 1 root xxxx 0 oct.   2 10:30 fichier1
```

4) rendez le groupe root propriétaire du fichier test/fichier1

```
# chgrp root test/fichier1
$ ls -l fichier1
-rw-rw-r-- 1 root root 0 oct.   2 10:30 fichier1
```

5) enfin, rendez root et son groupe propriétaires de tout le dossier test et son contenu.

```
# chown -R root:root test
$ ls -l test/
total 4
drwxrwxr-x 2 root root 4096 oct.   2 10:30 dossier1
-rw-rw-r-- 1 root root  0 oct.   2 10:30 fichier1
-rw-rw-r-- 1 root root  0 oct.   2 10:30 fichier2
-rw-rw-r-- 1 root root  0 oct.   2 10:30 fichier3
```

6) à présent, en tant que votre utilisateur, tentez d'afficher le contenu de test, d'y créer un nouveau fichier fichier5, d'éditer fichier1 et de supprimer fichier2. Notez, pour plus tard, le contenu de la première colonne de tout ces fichiers et dossier, ainsi que le résultat des opérations précédentes.

lister le contenu du dossier : ok
(car drwxrwxr-x 3 root root   4096 oct.   2 10:30 test)
```
$ ls -l test/
```

créer un fichier : impossible

```
$ touch test/fichier5
```

touch: impossible de faire un touch «test/fichier5»: Permission non accordée

ouvrier fichier1 en lecture : ok
en écriture : impossible

```
$ vim test/fichier1
```

W10: Alerte : Modification d'un fichier en lecture seule

suppression de fichier2 : impossible

```
$ rm test/fichier2
```

rm : supprimer fichier vide (protégé en écriture) «test/fichier2» ? y
rm: impossible de supprimer «test/fichier2»: Permission non accordée

```
$ ls -l test/
total 4
drwxrwxr-x 2 root root 4096 oct.   2 10:30 dossier1
-rw-rw-r-- 1 root root  0 oct.   2 10:30 fichier1
-rw-rw-r-- 1 root root  0 oct.   2 10:30 fichier2
-rw-rw-r-- 1 root root  0 oct.   2 10:30 fichier3
```

Permissions et Droits d'accès

http://fr.openclassrooms.com/informatique/cours/les-chmod

#chmod

Linux est un système multi-utilisateurs.
Chaque utilisateur appartient au moins à un groupe.

Les fichiers ont donc des permissions visant 3 types d'utilisateurs :

- celle concernant le propriétaire du fichier
- celle concernant le groupe du propriétaire du fichier
- celle concernant les autres utilisateurs.

Pour chaque type de personnes visées, il y a trois types de droits :

- r : droit de lecture
- w : droit d'écriture
- x : droit d'exécution.

A ces types de droits on associe un bit, prenant :

- la valeur 0 si la personne n'a pas ce droit
- la valeur 1 si la personne a ce droit.

On a donc des triplets du genre 111 (par exemple) pour chacun des types d'utilisateurs.

Ce qui nous donne 111 100 101 (encore par exemple).

Dans ce cas,

- le propriétaire a les droits 111, c'est-à-dire lecture, écriture et exécution
- le groupe a les droits 100, c'est-à-dire le droit de lecture
- les autres ont les droits 101, c'est-à-dire les droits de lecture et d'exécution.

Le fait d'avoir un nombre contenant des 0 et des 1 est ce qu'on appelle un nombre binaire.
Nous avons pour habitude de compter en *décimal* avec des chiffres de 0 à 9.
Mais on peut compter aussi en *octal* (chiffres de 0 à 7), ou en *hexadécimal* (de 0 à F, A = 10; B = 11; C = 12 ...; F = 15).

Or, les nombres binaires vont de 000 à 111, ce qui fait en octal de 0 à 7. Et nous pouvons ainsi donner à chaque type d'utilisateur son droit sous la forme d'un chiffre entre 0 et 7 (0 = aucun droit : 7 = tous les droits).
Une petite table pour s'y retrouver :

Position binaire	Valeur octale	Les droits	Commentaire
000	0	- - -	Aucun droits
001	1	- - x	Executable
010	2	- w -	Ecriture
011	3	- w x	Ecrire et executer
100	4	r - -	Lire
101	5	r - x	Lire et executer
110	6	r w -	Lire et ecrire
111	7	r w x	Lire ecrire et executer

Exercice :

1) dans l'exercice précédent (linuxtp53), tachez d'expliquer les résultats que vous avez obtenus pour le fichier1

fichier1 lecture mais pas d'écriture pour l'utilisateur nmace :
`-rw-rw-r-- 1 root root  0 oct.   2 10:30 fichier1`
nmace n'est pas propritétaire du fichier et n'appartient pas au groupe root, donc le droit qui s'applique est celui de "others" (3ème groupe). Du coup, il n'a que le droit de lecture.

2) Toujours dans le dossier "test" créé lors de l'exercice 5.3, réalisez les opérations suivantes :
(pour les deux premières questions, il s'agit de ne pas agir sur certains droits, comme l'execution pour la question 2. On ne donnera donc pas les droits exacts, donc ni de = ni d'octal)

- faire en sorte que root soit le seul à avoir droit d'executer "fichier1" (deux réponses possibles)

première réponse : root a de toute manière tout les droits sur tout les fichiers et dossier

```
# chmod -x fichier1
```

seconde réponse (en considérant que root est un utilisateur comme les autres)

```
# chown root ficher1
# chmod u+x,g-x,o-x fichier1
```

cela permet d'anticiper un éventuel futur changement de propriétaire

- faire en sorte que les membres du groupe "users" soient les seuls, en plus de l'utilisateur propriétaire, à pouvoir lire et écrire dans "fichier2"

```
# chgrp users fichier2
# chmod u+rw,g+rw,o-rw fichier2
```

- en octal : définir le droit d'accès à fichier3 comme suit : (votre utilisateur / nmace) lecture + ecriture + execution, (le groupe users) lecture + écriture, (les autres) rien

```
# chown nmace:users fichier3
# chmod 760 fichier3
```

- faites en sorte que tout le monde ai le droit de lecture sur le fichier dossier1/fichier4

```
# chmod +r dossier1/fichier4
```

Bonus 1) un cas concret : je veux que tout le monde puisse lire le fichier1, mais que seuls les membres du groupe "root" puissent y écrire et que seul les membres du groupe "users" puissent l'executer. Est-ce possible ? Pourquoi ?

On peut appliquer les droits sur un fichier de plusieurs façons :

- de manière symbolique
- de manière octale.

De manière octale

Si je veux changer tous les droits du fichier ou m'assurer qu'ils sont comme je le veux, c'est la meilleure manière.
Prenons un exemple.
J'ai un script *bash* que je veux être le seul à pouvoir modifier, mais que les personnes de mon groupe pourront lire. Et que tous pourront exécuter.
Je devrais mettre les droits suivants :

Utilisateur	U	G	O
Droits d'accès	r w x	r - x	- - x
Position binaire	1 1 1	1 0 1	0 0 1
Octale	7	5	1

On va donc exécuter la commande suivante :
chmod 751 script.sh

De manière symbolique

Si maintenant, je me rends compte que je ne peux pas modifier un fichier texte, ce que pourtant je voudrais.
Avec la méthode octale, il me faudrait tout décomposer pour seulement supprimer un droit.
Mais il existe la méthode symbolique.
Elle est de type : `chmod [ugoa][+-][rwx]`.

C'est l'une de ces lettres **u** (propriétaire du fichier), **g** (groupe), **o** (les autres), **a** (tout le monde = **u** + **g** + **o**), suivie de + ou - pour respectivement ajouter ou supprimer les permissions, et la forme symbolique des permissions est de la forme **r** (*read* : lecture), **w** (*write* : écriture), **x** (exécution).

Par exemple, pour pouvoir modifier ce fichier texte qui nous appartient :
`chmod u+w fich.odt`
on peut mettre plusieurs droits symboliques en les séparant par des virgules :
`chmod u+rw,g+r,o+r,a-x fich.odt`

Dans cet exemple, on ajoute les droits en lecture et en écriture au propriétaire, on ajoute les droits de lecture au groupe, les droits de lecture aux autres, et on enlève les droits d'exécution à tous.

N.B : Seul le propriétaire du fichier ou le super-utilisateur ROOT peut modifier les droits sur les fichiers et répertoires.

Les droits différents entre un répertoire et un fichier

UMASK

http://www.linuxcore.fr/2012/05/umask/

Par défaut, pour les fichiers, le umask est très souvent « 0022 », ce qui correspond à « rw-r--r--».
Pour les répertoires, le umask par défaut est souvent « 0755 »

La base des permissions des répertoires est calculée sur 0777 c'est à dire « rwxrwxrwx » alors que pour les fichiers c'est 0666 soit « rw-rw-rw ».
En effet, l'exécution sur les fichiers ordinaires, non exécutables, n'est pas possible.

Quelle valeur doit-on donner à umask pour créer des répertoires avec les permissions par défaut suivantes "rwxr-x---" ?

Réponse : 0027

Pour un fichier

- **r** : permission de lire le contenu du fichier
- **w** : permission de modifier le contenu du fichier
- **x** : sous Windows, le .exe permet d'exécuter un fichier, mais sous Linux, c'est ce droit qui permet de rendre un fichier exécutable.

Pour un répertoire

- **r** : permission de lister les fichiers
- **w** : permission d'ajouter ou de supprimer des fichiers correspond à la création et suppression de fichiers et dossiers qu'il contient. Cependant, pour pouvoir créer ou supprimer une partie ou la totalité de son contenu, nous avons besoin de lancer des commandes qui font appel à ce contenu. Donc, il s'agit d'executer une opération de création ou suppression sur le contenu du dossier. Cela necessite donc le droit d'execution en plus du droit d'écriture.
- **x** : sur un répertoire, ce droit empêche de rentrer dans le répertoire, et donc de lister les fichiers et répertoires qu'il contient mais on a le droit d'exécuter des opérations sur le contenu du dossier

TP droit dossier :

Modifier les droits sur votre dossier "test" comme indiqué **ci-dessous**, puis tenter, en tant que votre utilisateur (nmace) de :

a) supprimer le fichier "fichier1"
b) de le (re)créer
c) de lister le contenu de "test" puis de "dossier1"
d) d'executer "script.sh" grâce à la commande suivante : $ ./test/script.sh
e) de lancer la commande suivante (en remplaçant X par le numéro de votre essais) : $ echo "test X" >> test/fichier2
f) et enfin de vous rendre (cd) dans "test".

Droits à mettre en place sur "test" (et uniquement "test", pas son contenu) pour les utilisateurs non propriétaires (ie. "other") :

- **Aucun droits (---)**
  rien ne marche

- **Droit de lecture uniquement (r--)**
  Opérations interdites sur les fichiers contenus dans le dossier test
  pas de suppression (a)
  pas de création (b)
  pas d'execution (d)

pas d'écriture (e)
c) on peut bien voir le contenu du dossier, mais pas les attributs des fichiers qu'il contient (on n'a que leur nom)
f) pas de droit d'execution, donc impossible

- **Droit d'écriture uniquement (-w-)**
  nous ne pouvons lancer aucune de ces opération, tout comme si nous n'avions aucun droit

Explication : le droit d'écriture correspond à la création et suppression de fichiers et dossiers qu'il contient. Cependant, pour pouvoir créer ou supprimer une partie ou la totalité de son contenu, nous avons besoin de lancer des commandes qui font appel à ce contenu. Donc, il s'agit d'executer une opération de création ou suppression sur le contenu du dossier. Cela necessite donc le droit d'execution en plus du droit d'écriture.

La seule chose permise avec uniquement ce droit est donc de **supprimer ce dossier si et seulement si il est vide** (rmdir ou rm -r). Cela serait éventuellement utilile dans le cas où un utilisateur peu lancer un rm -r sur un dossier père, mais que vous voulez l'empécher de supprimer certains de ses dossiers fils non vides. On mettra dans ce cas l'accès de ces dossiers fils à 662 par exemple. Ainsi, son rm -r aura bien supprimé certains fichiers, certains dossiers, mais pas ceux que vous avez "protégé".

- **Droit d'execution uniquement (--x)**
  on ne peut que s'y déplacer et le traverser (cd)

on peut également écrire dans un fichier (sur lequel on a le droit), l'afficher et executer un fichier qu'il contient mais on ne peu pas supprimer ni créer à l'intérieur, ou lister son contenu
pour bien s'en rappeller : droit d'**executer** des opérations **sur le contenu** du dossier

- **Droit d'écriture et d'execution (-wx)**
  A présent, on peu bien créer et supprimer du contenu, en plus d'executer des opérations sur celui-ci.

- **Droit de lecture et d'execution (r-x)**
  Contrairement au premier cas (lecture seule), nous obtenons bien les informations sur les fichiers en plus de leur nom lors d'un ls -l.

Explication : récupérer des informations sur ces fichiers et dossiers (droits, propriétaire, date, etc ...) consiste bien à executer une opération sur ceux-ci (le contenu du dossier), alors que leur noms étant simplement "stocké dans le dossier" (pour s'en assurer : ouvrir un dossier avec vim), le droit de lecture (des informations du dossier) suffit à les afficher.

- **Lecture et écritue (rw-)**
  Similaire à r seul (seul le droit de supprimer le dossier si il est vide est ajouté).

Tout les droits (rwx)
Bien entendu, on a la possibilité de tout faire.

D'autres droits

Le Sticky bits :

Il permet :

- lorsqu'on l'applique à un exécutable, de le garder en mémoire lors de sa première exécution
- lorsqu'on l'applique à un répertoire, seul le propriétaire du fichier ou le propriétaire du répertoire parent a le droit d'effacer les fichiers.

On le met ainsi : `chmod u+t fichier`

Le SUID, le SGID :

- Le SUID permet d'avoir accès aux droits du propriétaire à l'intérieur du programme, pour avoir accès aux fichiers de configuration, par exemple. Pour des raisons de sécurité, le SUID ne s'applique qu'aux programmes binaires compilés à l'exception des scripts Perl. **N'a aucune incidence sur les fichiers !**

- Le SGID permet de déterminer le groupe des fichiers créés dans le répertoire.

Suid :

- Appliquer à un fichier, permet d'executer ce dernier en bénéficiant des permissions de son propriétaire.

Exemple : la commande passwd.

Sgid :

- Appliquer à un fichier, permet d'exécuter ce dernier en bénéficiant des permissions du groupe qui en est propriétaire.
- Appliquer à un dossier, fait en sorte que tous les nouveaux fichiers crées appartiennent au groupe propriétaire du dossier.

Sticky bit :

- Appliquer à un dossier, les utilisateurs ne peuvent plus supprimer ou modifier les fichiers dont ils ne sont pas le propriétaire.

Exemple : le dossier /tmp.

On le met ainsi :

- Pour le SUID : `# chmod u+s prog ou chmod 4755 programme`
- Pour le SGID : `# chmod g+s rep/ ou chmod 2755 repertoire/`

A noter :

Sur un dossier, si vous utilisez chmod avec une permission en octal à 4 chiffres, le 4ème chiffre permet d'ajouter le SUID, le SGID et le Sticky Bit, mais de ne supprimer que le sticky bit. Exemple : si les droits actuels de notre dossier sont à 2775, et que nous faisons un chmod 1777, nous ajoutons le Sticky Bit, mais conservons le SGID. Nous obtenons donc un droit de 3777. Si, ensuite, nous indiquon 0777, nous supprimons le sticky bit, mais conservons le SGID. Nous obtenons donc 2777. Ceci n'est pas applicable pour les fichiers.

Représentation des droits spéciaux

• rwsrwsrwt
lettre minuscule si avec X (droit execution)
majuscule si sans X (droit execution)

• Octal supplémentaire
7777
SUID = 4, SGID = 2, Sticky bit =1
ex : 6744 = rw**s**r-**S**r--

TP Droits spéciaux :

pré-requis : idem que chapitre précédent (0777 root root sur le dossier test en récursif)
Nous travaillerons en tant que notre utilisateur (nmace) et testerons les effets des droits spéciaux sur celui-ci. Toutes nos opérations (à part bien entendu les chmod et chown) se feront en tant que notre utilisateur (nmace).

```
# chmod -R 0777 test
# chmod -R u-s,g-s test
# chown -R root:root test
```

Partie 1 : tout est normal
1) copiez les fichiers /usr/bin/whoami (ou /bin/whoami, a vérifier grâce à la commande whereis) et /sbin/shutdown dans votre dossier test.

```
# whereis whoami
# cp /usr/bin/whoami test/
# whereis shutdown
# cp /sbin/shutdown test/
```

2) Appliquez le droit 0777 sur ceux-ci, et changer leur propriétaire et leur groupe propriétaire en root:root.

cf pré-requis

3) Executez ces fichiers. Pour se faire, il vous suffit de vous rendre dans votre dossier test et d'executer les commandes suivantes :

```
$ ./whoami
```

affiche le nom de votre utilisateur

```
$ ./shutdown -r now
```

shutdown: Nécessite d'être super-utilisateur (root)

Attention : n'oubliez pas le "./" au début, sans quoi vous ferez appel au mauvais fichier (ceux d'origine, dans /usr/bin et /sbin)
Que ce passe-t-il ? Pourquoi ?
(si ces commandes vous sont inconnues, vous pouvez utiliser le man pour vous documenter à leur sujet)

4) Ajoutez le suid sur ces fichier, et réexecutez les. Qu'est ce qui a changé ? Pourquoi ?

```
$ ./whoami
```
root
```
$ ./shutdown -r now
```
redémarre la machine

Partie 2 : sushi ?
1) Qu'est ce qu'une attaque sushi, succinctement ?

suid+droit d'écriture = possibilité de mettre ce qu'on veux dans le fichier, et ainsi de lancer ce que l'on veut en tant qu'un autre utilisateur (comme root)

2) Etablissez le droit 6777 sur votre fichier test/shutdown.
```
# chmod 6777 shutdown
```

3) Nous allons tenter une attaque sushi classique, en copiant bit à bit le contenu du fichier executable /bin/su dans ce fichier test/shutdown. Ainsi, nous devrions avoir un executable nous permettant de changer d'utilisateur à volonté sans demande de mot de passe.
Pourquoi cela devrait-il se passer ainsi ? A votre avis, cela va-t-il fonctionner ? Pourquoi ?

Cela ne fonctionnera pas, étant donné que GNU/Linux empèchera toute modification d'un fichier avec suid.

4) Tester l'opération décrite dans la question précédente grâce à la commande suivante :
```
$ dd if=/bin/su of=~/test/shutdown
```
Que ce passe-t-il à présent si on execute test/shutdown ? Pourquoi ?

Le suid a été supprimé sur ce fichier. De ce fait, l'utilisateur pourra toujours changer d'utilisateur en executant ce fichier, mais le mot de passe lui sera demandé ... ce qui ne lui sera pas très utile. Même après avoir remis ce droit sur le fichier, celui-ci continu à demander le mot de passe (ie, le suid n'est pas pris en compte).

6) Et si on fait la meme chose avec un script bash ? Mettez le droit 6777 sur le fichier monscript.sh, et, avec vim, remplacez la ligne "echo youpi" par whoami. Enfin, executez ce fichier.

Aucun script bash ne prend en compte le suid, même si vous l'avez activé. Du coup, la protection ici était même en amont de l'édition que vous avez tenté de faire.

Partie 3 : sticky bit et les dossiers
```
# chown -R root:root test
```

Modifier les droits sur votre dossier "test" comme indiqué ci-dessous, puis tenter, en tant que votre utilisateur (nmace) de :

a) de créer un fichier10 et affichez ses droits d'accès.
b) supprimer. le fichier "fichier10"
c) de lancer la commande suivante (en remplaçant X par le numéro de votre essais) :
$ echo "test X " >> test/fichier2

Droits à mettre en place sur "test" (et uniquement "test", pas son contenu) pour les utilisateurs non propriétaires (ie. "other") :

- 0777

Aucun droit spécial. Etant donné que nous avons tout les droits (rwx), aucune restriction ne s'applique.

- 4777

Le SUID sur un dossier n'ayant aucun effet, rien ne change.

- 2777

Le SGID fait en sorte que le groupe propriétaire de tout nouveau fichier créé dans ce dossier, soit celui du dossier en question, au lieu du groupe principal du créateur.

- 2775

Ici, nous n'avons plus le droit d'écriture sur le dossier. Nous ne pouvons donc plus créer de fichier dans celui-ci. Donc le SGID ne nous sert à rien. Il aura en revanche son utilité si le propriétaire n'a pas le groupe propriétaire de ce dossier comme groupe principal.

- 1777

Sticky Bit + droit d'écriture (+ le fait que nous ne soyons pas propriétaire du dossier) = nous pouvons toujours créer des fichiers, supprimer ceux dont nous sommes propriétaires, mais pas ceux dont nous ne sommes pas propriétaire (ce qui est le cas ici avec "fichier2", qui a "root" comme propriétaire, mais nous pouvons bien supprimer "fichier10", qui nous appartient).

- 1774

Pas de droit d'execution, donc on ne peu de toute manière lancer aucune opération sur nos fichiers.

- 1775

Droit de lecture + Sticky bit = pas de droit d'écriture, donc le sticky bit ne sert à rien, vu qu'on n'est interdit (par l'abscence de w) de créer et supprimer des fichiers, même si ils nous appartiennent.

- 1773

droit de création (w) + sticky bit (t) + execution (t minuscule) = droit de créer des fichiers, de lancer des opérations dessus, mais pas de supprimer ceux dont on n'est pas propriétaire.

- 1776

Ici, l'abscence de droit d'execution sur ce dossier nous empèche de lancer quelque opération que ce soit sur son contenu. De ce fait, nous ne pouvons rien y faire.

A noter : sur un dossier, si vous utilisez chmod avec une permission en octal à 4 chiffres, le 4ème chiffre permet d'ajouter le SUID, le SGID et le Sticky Bit, mais de ne supprimer que le sticky bit. Exemple : si les droits actuels de notre dossier sont à 2775, et que nous faisons un chmod 1777, nous ajoutons le Sticky Bit, mais conservons le SGID. Nous obtenons donc un droit de 3777. Si, ensuite, nous indiquon 0777, nous supprimons le sticky bit, mais conservons le SGID. Nous obtenons donc 2777. Ceci n'est pas applicable pour les fichiers.

Pour le sticky bit appliqué sur un dossier : seul le propriétaire du fichier ou le propriétaire du répertoire parent a le droit d'effacer les fichiers.