Utilisateurs, groupes, et sécurité

C'est évidemment à l'administrateur que reviennent les tâches de gestion des utilisateurs et des groupes sur la machine – notions fondamentales sur un système Unix, dans la mesure où elles forment la base de ses mécanismes de sécurité.

SOMMAIRE

MOTS-CLEFS

- Utilisateurs et groupes
- Gestion des permissions
- Sécurité

La gestion des utilisateurs et groupes est typiquement le domaine où chaque Unix utilise des outils spécifiques. Leur raison d'être est de faciliter la vie de l'administrateur système, mais ils ont parfois le mauvais goût de la compliquer, en introduisant autant de comportements différents qu'il existe de versions d'Unix.

Heureusement, il est possible de gérer ses utilisateurs en s'en tenant aux outils et méthodes qui relèvent d'un tronc commun. C'est ce que nous allons examiner dans ce chapitre, qui se veut extrêmement générique.

Gérer ses utilisateurs

PIÈGE À C... **Tout faire en tant que root**
Certains administrateurs débutants s'interrogent parfois sur l'intérêt de mettre en place un compte utilisateur normal alors qu'ils peuvent travailler en permanence en tant que root.
La réponse à cette question s'imposera à eux – douloureusement – le jour où ils feront le ménage à la racine du disque en croyant se trouver sous /tmp, détruisant irrémédiablement tout le système.

Contrairement à ce que le titre pourrait laisser penser, nous n'aborderons pas ici l'aspect humain de la gestion des utilisateurs. Non que le sujet soit inintéressant, mais il pourrait faire l'objet d'un ouvrage entier. Nous nous cantonnerons donc à la gestion technique des comptes des utilisateurs. Une des premières tâches de l'administrateur d'un système Unix est de créer des comptes utilisateur. Même sur une machine sans utilisateur local, l'ingénieur système doit se créer un compte personnel.

Sur Unix canal historique

À l'origine, toutes les informations des comptes utilisateur se trouvaient dans le fichier /etc/passwd, y compris les mots de passe. Pour modifier des comptes, on intervenait sur ce fichier avec un éditeur de texte, tel que **`vi`**. Voici un exemple de fichier /etc/passwd :

```
root:2hKTuNumO4sU6:0:0:Charlie Root:/root:/bin/sh
daemon:*:1:31:The devil himself:/:/sbin/nologin
operator:*:2:5:System operator:/usr/guest/operator:/sbin/nologin
bin:*:3:7:Binaries Commands and Source:/:/sbin/nologin
news:*:6:8:Network News:/var/spool/news:/sbin/nologin
uucp:*:66:1:UNIX-to-UNIX Copy:/var/spool/uucppublic:/usr/libexec/uucp/uucico
manu:pCs71RItmHgdc:500:500:Emmanuel Dreyfus:/home/manu:/bin/ksh
```

Chaque ligne définit un compte utilisateur. Elle comprend sept champs séparés par le caractère `:` (deux points).

- L'identifiant de connexion (*login*) de l'utilisateur, limité à huit caractères sur certains systèmes. Même si ce n'est pas le cas de tous, il est conseillé de toujours s'en tenir à huit caractères maximum pour des raisons d'interopérabilité,
- Le mot de passe de l'utilisateur, chiffré avec un procédé à sens unique (*one-way function*) pour produire une valeur de hachage (*hash* en anglais) : c'est le mot de passe chiffré. On peut facilement chiffrer un mot de passe, mais il est difficile de retrouver un mot de passe produisant un hachage donné. Lorsque l'usager tente de se connecter au système, il fournit son mot de passe, qui est chiffré puis comparé à ce champ du fichier /etc/passwd. Si les versions chiffrées coïncident, le candidat est accepté. Dans le cas contraire, la connexion est refusée. Laissé vide, ce champ laisse un accès sans mot de passe. On peut encore y placer un `*` (astérisque) pour interdire toute connexion par mot de passe sur le compte en question.

- L'identifiant unique de l'utilisateur (UID), constamment utilisé par le système pour l'identifier. Ce nombre doit être compris entre 0 et (souvent) 32767, 0 étant réservé à l'administrateur. Certains systèmes permettent des UID supérieurs à 32767, mais pour des raisons d'interopérabilité, il est conseillé ici encore de s'en tenir au dénominateur commun.
- L'identifiant unique du groupe de l'utilisateur (GID), soumis aux mêmes limitations que l'UID. L'utilisateur peut appartenir à plusieurs groupes, comme nous le verrons plus loin. Dans ce cas, le numéro indiqué ici est celui de son groupe principal (on dit aussi « groupe primaire »).
- Le nom complet de l'utilisateur, éventuellement suivi de ses coordonnées : adresse, téléphone, etc.
- Le chemin du répertoire personnel de l'utilisateur. S'il n'existe pas, c'est la racine du système de fichiers qui sera utilisée comme répertoire par défaut au moment de la connexion. L'utilisateur n'y aura pas pour autant des droits particuliers...
- Le shell (interpréteur de commandes) par défaut de l'utilisateur. C'est le programme qui sera invoqué lors de la connexion. On peut ainsi empêcher quelqu'un de se connecter en session interactive en précisant ici un exécutable tel que /sbin/nologin. /sbin/nologin n'est pas un shell. Lorsqu'on l'invoque, il rend immédiatement la main, ce qui aura pour effet de terminer instantanément une tentative d'ouverture de session.

Le fichier /etc/passwd s'édite avec n'importe quel éditeur de texte, à l'exception de son champ de mot de passe, qu'on réserve souvent aux bons soins de la commande **passwd**. Invoquée par root, elle permet de changer le mot de passe de n'importe qui, sans devoir prouver au préalable sa connaissance de l'ancien mot de passe :

```
# passwd manu
Changing local password for manu.
New password:
```

EN PRATIQUE **Quels UID utiliser ?**

Les UID (numéros d'identifiant utilisateur) sont libres, à condition de ne pas entrer en conflit avec les UID réservés par le système. Sur BSD, les UID supérieurs à 200 ne sont pas alloués ; vous pouvez en faire ce qui vous plaît. Sur un certain nombre de systèmes Unix, l'UID maximum est 32767.

PIÈGE À C... **Le shell de root**

On peut trouver rudimentaire le shell par défaut de root (souvent, /bin/sh). Si vous désirez le changer, n'oubliez pas que root peut être amené à se connecter alors que /usr n'est pas encore monté. Préférez donc un shell lié statiquement, ou à défaut, un shell lié avec des bibliothèques présentes sur la partition racine du système de fichiers – le shell lui-même doit évidemment aussi être présent sur la partition racine !

Autre erreur : le séparateur de champs étant le caractère deux points (**:**), tout ce qui traîne en fin de ligne (espace, tabulation, ou autre) sera considéré comme partie intégrante du nom du shell. Ce shell n'existera donc pas, et il sera impossible de se connecter. Ce genre de faute peut être assez difficile à corriger sur certains Unix, où **init** invoque le shell de root et non pas forcément /bin/sh, même en mode mono-utilisateur.

CULTURE **Casser les mots de passe**

Pour casser les mots de passe chiffrés d'Unix, il suffit de tester toutes les combinaisons possibles, de les chiffrer tour à tour, et de comparer les résultats à la chaîne de caractères du fichier /etc/passwd. En y mettant les moyens, on peut désormais trouver en quelques jours un mot de passe chiffré par DES, l'algorithme utilisé historiquement pour les mots de passe d'Unix. On peut restreindre cette méthode de force brute, en rien subtile, à un cas particulier : l'attaque dite « du dictionnaire » se contente de tester les mots et combinaisons simples de mots du dictionnaire dans plusieurs langues.

Les Unix modernes permettent d'utiliser des algorithmes plus résistants, comme MD5, mais le gain de sécurité est discutable. Certes, il faut beaucoup plus de temps pour casser un mot de passe chiffré en MD5 qu'un mot de passe chiffré en DES, mais le succès de l'attaque n'est toujours qu'une question de temps. La meilleure protection reste encore de ne pas dévoiler les mots de passe chiffrés aux usagers autres que root. Les systèmes Unix modernes proposent des dispositifs pour assurer ce niveau de protection.

Sur UNIX System V

Le fichier /etc/passwd doit être lisible par tous, pour permettre par exemple à la commande **`ls -l`** de traduire les UID des propriétaires de fichiers en noms d'utilisateurs. C'est le point faible du système classique, car cela permet aussi à un utilisateur mal intentionné d'avoir accès aux mots de passe chiffrés – il lui est donc possible de tenter une attaque par dictionnaire ; des programmes spécialisés sont même disponibles à cet effet.

Pour remédier à cela, les UNIX System V (et GNU/Linux), proposent le système des *shadow passwords* (mots de passe cachés) : les mots de passe sont stockés dans un autre fichier, /etc/shadow, et les champs de mot de passe du fichier /etc/passwd ne contiennent alors qu'un astérisque.

Le fichier /etc/shadow ne contenant pas d'information utile à tous, on peut en restreindre l'accès à root. Ainsi, /etc/passwd reste accessible à tous, mais les mots de passe sont mieux gardés.

Voici un exemple de fichier /etc/shadow. Le format est similaire à celui du fichier /etc/passwd. Le *login* et le mot de passe chiffré sont suivis de limites de validité du mot de passe (voir la page **`man`** de shadow pour les détails).

```
root:2hKTuNumO4sU6:11565:0:99999:7:::134550460
daemon:*:10946:0:99999:7:::
operator:*:10946:0:99999:7:::
bin:*:10946:0:99999:7:::
news:*:10946:0:99999:7:::
uucp:*:10946:0:99999:7:::
manu:pCs71RItmHgdc:11395:0:99999:7:::134538036
```

ALTERNATIVE **Autres outils pour la gestion des utilisateurs**
On trouve encore des outils de gestion des utilisateurs plus conviviaux, comme **`sysinstall`** ou **`pw`** sur FreeBSD, ou **`sushi`** sur NetBSD, mais ils sont spécifiques à leurs systèmes respectifs.

Sous UNIX System V, on peut toujours modifier /etc/passwd et /etc/shadow avec un éditeur de texte tel que **`vi`**. On peut aussi utiliser des outils qui permettent d'ajouter ou de supprimer des utilisateurs en ligne de commande. Les pages **`man`** des commandes **`useradd`**, **`userdel`** et **`usermod`** fourniront tous les détails. L'édition directe des fichiers évite d'avoir à se souvenir de la syntaxe de ces commandes, mais laisse le champ libre à toute fausse manœuvre. Une méthode intermédiaire utilisant la commande **`vipw`** sera traitée dans la section suivante.

Sur systèmes BSD

La méthode de protection des mots de passe des UNIX System V a un inconvénient : deux fichiers sont nécessaires à la définition des comptes. Lors d'une intervention manuelle, il faut les traiter tous deux, et il est possible d'aboutir à une situation incohérente ou contradictoire (si par exemple **`useradd`** a planté en cours de route).

Les BSD proposent donc une autre approche : un fichier /etc/master.passwd, lisible uniquement par root, qui contient toute l'information des comptes, mots de passe compris. Le fichier /etc/passwd est généré automatiquement à partir de ce dernier et il est lisible par tous. Bien entendu, les mots de passe chiffrés y sont remplacés par des astérisques. Voici un exemple de fichier /etc/master.passwd.

```
root:2hKTuNumO4sU6:0:0::0:0:Charlie Root:/root:/bin/sh
daemon:*:1:31::0:0:The devil himself:/:/sbin/nologin
operator:*:2:5::0:0:System operator:/usr/guest/operator:/sbin/nologin
bin:*:3:7::0:0:Binaries Commands and Source:/:/sbin/nologin
news:*:6:8::0:0:Network News:/var/spool/news:/sbin/nologin
uucp:*:66:1::0:0:Unix-to-Unix Copy:/var/spool/uucppublic:/usr/libexec/uucp/uucico
manu:pCs71RItmHgdc:500:500::0:0:Emmanuel Dreyfus:/home/manu:/bin/ksh
```

On trouve trois champs supplémentaires par rapport à un fichier /etc/passwd traditionnel. Placés entre le GID et le nom complet de l'utilisateur, ils servent à consigner des informations sur la durée de validité du compte ou la classe d'utilisateur telle que définie dans login.conf. Tous les détails sont expliqués dans les pages **man** de master.passwd et login.conf. Ces champs supplémentaires sont absents du fichier /etc/passwd généré à partir de /etc/master.passwd.

Cette approche n'utilise qu'un seul fichier, mais impose l'exécution d'une commande pour reconstruire /etc/passwd à chaque modification de /etc/master.passwd : **pwd_mkdb -p /etc/master.passwd**. C'est peu convivial, mais heureusement, on trouve plus pratique : la commande **vipw**.

vipw invoquera votre éditeur favori, tel que précisé dans la variable d'environnement EDITOR, ou **vi** par défaut, en lui faisant éditer une copie de /etc/master.passwd. À la fin de l'intervention, **vipw** contrôle le format du fichier. S'il est incorrect, il indique l'erreur et propose de ré-éditer le fichier. Si tout va bien, **vipw** remplace /etc/master.passwd par la copie modifiée, puis exécute **pwd_mkdb**. Il s'occupe en outre de verrouiller l'accès à la base de mots de passe pendant l'édition, pour éviter tout accès concurrent.

vipw permet donc d'éditer le fichier de mots de passe en toute sécurité, et il évite d'avoir à se souvenir des détails sordides de la synchronisation des bases de mots de passe sur systèmes BSD. Sur les UNIX System V, dépourvus de fichiers à régénérer, on trouve souvent un programme **vipw** qui permet d'éditer le fichier des mots de passe ; il se contente alors du verrouillage et du contrôle de la syntaxe.

Pour les inconditionnels de la méthode System V, les systèmes BSD proposent des outils de gestion des utilisateurs : **pw** ou **adduser** sous FreeBSD, **user** ou **adduser** sous NetBSD et OpenBSD. Malheureusement, leur syntaxe n'étant pas standardisée d'un système à l'autre, elle est difficile à retenir. On gagne souvent beaucoup de temps à utiliser **vipw**, semblable partout.

PLUS LOIN **Les classes d'utilisateur**

Elles permettent de configurer finement les limites imposées aux utilisateurs. On indique un nom de classe dans le cinquième champ du fichier /etc/master.passwd, et on définit ses attributs dans /etc/login.conf. Il est ainsi possible de définir le nombre maximum de processus autorisés pour un utilisateur, la quantité maximale de mémoire qu'il peut allouer, les priorités de ses processus, des variables d'environnement, etc.
Attention, si le fichier /etc/login.conf existe, tous les utilisateurs doivent appartenir à une classe définie dans ce fichier, sous peine de ne plus pouvoir se connecter au système. Si vous laissez un champ de classe vide dans le /etc/master.passwd, l'utilisateur relèvera de la classe default – qu'il faut définir dans /etc/login.conf.

SUR LES AUTRES UNIX **Commandes BSD sur UNIX System V**

Sur certains UNIX System V, la commande **vipw** se trouve dans le répertoire /usr/ucb, accompagnée d'un certain nombre d'autres utilitaires BSD (notamment un **ps** mode BSD, pour les allergiques au **ps** mode System V).
Le terme ucb signifie « Université de Californie à Berkeley », à l'origine des systèmes BSD.

PLUS LOIN **Les bases binaires**

En réalité, sur les systèmes BSD, les fichiers /etc/master.passwd et /etc/passwd ne sont pas utilisés par les processus ayant besoin de lire des informations relatives aux comptes des utilisateurs. La commande **pwd_mkdb** utilise aussi /etc/master.passwd pour générer deux bases de données binaires des utilisateurs : /etc/spwd.db et /etc/pwd.db. La première, qui contient les mots de passe, est réservée à root ; la deuxième ne contient pas de mot de passe et elle est accessible à tous.

Ces bases binaires permettent un accès plus rapide à l'information que de simples fichiers texte, mais il faut bien noter qu'elles sont uniquement générées à partir du fichier /etc/master.passwd. Si jamais elles devaient être corrompues, il est facile de les régénérer.
Ces bases binaires ont un format qui change suivant que l'on est sur un système petit boutiste ou gros boutiste. Si vous devez migrer une base d'utilisateurs entre deux architectures différentes, n'oubliez pas d'invoquer **vipw** pour régénérer les bases binaires.

Check-list de la création de compte

Créer une entrée dans la base des utilisateurs ne suffit pas, d'autres étapes doivent suivre. Voici une liste non exhaustive des opérations à effectuer lors de la création d'un compte.

- Créer le répertoire personnel de l'utilisateur, y placer les fichiers de configuration par défaut tels que .profile ou .cshrc, et ne pas oublier la commande **chown -R**, qui lui donnera tous les droits sur ses propres fichiers.
- Mettre en place son quota, le cas échéant. C'est la fonction de la commande **edquota**.
- Placer les alias de messagerie dans le fichier /etc/mail/aliases (qui peut se trouver ailleurs selon le système, par exemple sous /etc/aliases), et exécuter la commande **newaliases** pour que la base binaire des alias de messagerie soit mise à jour.

Les programmes de création de compte à la System V ont au moins le mérite d'automatiser tout cela. Mais rien n'empêche d'utiliser conjointement ces programmes et **vipw** pour tirer parti du meilleur des deux mondes.

RAPPEL **chown**

chown permet de changer le propriétaire d'un fichier ou d'un répertoire.

Comment configurer les droits ?

Gérer les groupes

Sous Unix, les groupes permettent de contrôler l'accès à certaines ressources. Chaque utilisateur, on l'a vu, appartient à un groupe principal. Cela suppose donc qu'il peut appartenir à plusieurs groupes...

Le fichier /etc/group établit la correspondance entre les identifiants de groupes, qu'on trouve dans /etc/passwd, et les noms symboliques des groupes, plus mnémotechniques. On y définit aussi les groupes secondaires des utilisateurs. Exemple de fichier /etc/group :

```
wheel:*:0:root,manu
daemon:*:1:daemon
kmem:*:2:root
sys:*:3:root
tty:*:4:root
operator:*:5:root,manu,dumpy
mail:*:6:
bin:*:7:
news:*:8:
guest:*:31:root
nobody:*:39:
users:*:500:
```

PLUS LOIN **Mot de passe de groupe**

Le deuxième champ du fichier /etc/group est aujourd'hui obsolète : il permettait de mettre en place un mot de passe de groupe pour qu'un utilisateur puisse, par le biais de la commande **newgrp**, s'insérer dans un groupe dont il ne faisait pas partie.

SUR LES AUTRES SYSTÈMES **Les ACL**

Traditionnellement, les droits sur les fichiers sous Unix se limitent au propriétaire, à son groupe, et au reste des utilisateurs.
Certains systèmes connaissent des listes de contrôle d'accès (*Access Control Lists*, ou ACL), d'une bien plus grande finesse et expressivité.
Quelques systèmes d'exploitation proposent les ACL, dont Windows NT/2000/XP, Solaris, AIX, TrustedIRIX, et de façon plus expérimentale encore, GNU/Linux et FreeBSD.
Dans le cas des systèmes Unix, les ACL sont standardisées par la norme POSIX 1e.

SUR LES AUTRES UNIX **Netinfo**

Sur NeXTStep et son descendant MacOS X, les fichiers /etc/passwd et /etc/group ne servent qu'au tout début du démarrage, avant le lancement des services **lookupd** et **netinfod**.
Après cette opération, les programmes ayant besoin de renseignements sur les utilisateurs font appel au service **lookupd**, capable d'interroger diverses sources pour trouver les informations demandées. Dans la configuration par défaut, les données concernant les groupes et les utilisateurs sont stockées dans la base Netinfo.
Cette dernière permet de partager facilement une foule d'informations entre plusieurs machines, mais elle est complètement spécifique à MacOS X. C'est le service **netinfod** qui assure l'accès à la base Netinfo. Il répond le plus souvent aux requêtes de **lookupd**, mais on peut également l'interroger à la ligne de commande en tapant **nidump**, pour récupérer par exemple les groupes définis dans la base Netinfo :

```
$ nidump group .
nobody:*:-2:
nogroup:*:-1:
wheel:*:0:root
daemon:*:1:root
kmem:*:2:root
sys:*:3:root
(...)
```

Le format est encore celui de /etc/passwd. Le premier champ contient le nom du groupe, le troisième champ son GID, et le quatrième la liste des utilisateurs qui appartiennent à ce groupe en plus de leur groupe principal.

Dans l'exemple ci-dessus, il est indiqué entre autres que le groupe de GID 500 a pour nom `users`. Dans le fichier /etc/passwd donné plus haut, l'utilisateur `manu` avait pour GID 500 : il appartient donc au groupe `users`, et il n'est pas utile de préciser cela à nouveau dans le fichier /etc/group. En revanche, cet extrait précise que `manu` a pour groupes secondaires `wheel` et `operator`.

Le fichier /etc/group n'est pas utilisé pour produire de base binaire, et même endommagé, il n'empêchera pas root de se connecter sur la console de la machine. On peut donc le manipuler sans craintes avec un éditeur de texte. Il existe toutefois des commandes spécialisées pour le modifier, mais ici encore leur syntaxe varie beaucoup d'un système à l'autre.

Attributs spéciaux *set-UID* et *set-GID*

Les attributs spéciaux *set-UID* et *set-GID*, qui concernent les exécutables, jouent un grand rôle dans la gestion des droits sur un système Unix : ils modifient les droits avec lesquels s'exécute le programme.

En temps normal, lorsqu'un utilisateur invoque un programme, ce dernier est exécuté avec les droits de cet utilisateur : il porte donc son UID et son GID. Mais un programme *set-UID* et/ou *set-GID* prendra respectivement lors de son exécution l'UID du propriétaire du fichier contenant le programme et/ou le GID de son groupe.

Ce mécanisme permet par exemple aux utilisateurs de modifier leurs mots de passe avec la commande **passwd**. Cette opération requiert en effet des droits en écriture sur le fichier /etc/master.passwd, droits dont seul root dispose. Mais l'exécutable **passwd**, qui appartient à root, est *set-UID* (on parle de programme *set-UID* root) ; il s'exécute donc avec les privilèges de root, ce qui lui permet de remplir sa fonction.

SÉCURITÉ **Attention aux binaires set-UID**

Les programmes *set-UID* sont pratiques, mais dangereux. Un exécutable *set-UID* root doit être très soigneusement programmé, pour éviter toute exploitation abusive par un utilisateur peu scrupuleux (c'est une source classique de failles de sécurité).
Placer un attribut *set-UID* sur un programme quelconque peut être lourd de conséquences. Un /bin/sh *set-UID* donnera par exemple un shell avec les droits de root, permettant ainsi à un utilisateur de tout faire sur le système. Dans le cas général, ne donnez pas l'attribut *set-UID* ou *set-GID* à un programme qui n'a pas été conçu pour cela.

Ces attributs spéciaux se manipulent avec la commande **chmod**, et sont matérialisés par un `s` en lieu et place du `x` dans l'affichage des droits donné par **`ls -l`**. Quelques exemples seront plus parlants :

```
# ls -l toto
-r-sr-sr-x  1 root  daemon    23248 Apr  1 17:57 toto
# chmod ug-s toto ❶
# ls -l toto
-r-xr-xr-x  1 root  daemon    23248 Apr  1 17:57 toto
# chmod u+s toto ❷
# ls -l toto
-r-sr-xr-x  1 root  daemon    23248 Apr  1 17:57 toto
# chmod u-s toto ❸
# ls -l toto
-r-xr-xr-x  1 root  daemon    23248 Apr  1 17:57 toto
# chmod g+s toto ❹
# ls -l toto
-r-xr-sr-x  1 root  daemon    23248 Apr  1 17:57 toto
```

❶ On supprime les deux attributs

❷ On met en place l'attribut *set-UID*

❸ On ôte l'attribut *set-UID*

❹ On met en place l'attribut *set-GID*

PLUS LOIN **Scripts set-UID root**

Le système n'honore pas les attributs *set-UID* et *set-GID* sur les scripts. L'interpréteur correspondant (souvent un shell dans le cas de scripts shell) sera donc invoqué avec les droits habituels.
Cette limitation est volontaire, car les scripts *set-UID* sont dangereux. Un script héritant des variables d'environnement de l'utilisateur, il existe de nombreux moyens de détourner son comportement et compromettre ainsi la sécurité du système. La manière la plus simple est de fournir au script une variable `PATH` lui faisant exécuter des scripts personnels au lieu de commandes classiques. Prenons le cas du script suivant, nommé killxlock.sh, et qui a pour effet de tuer un éventuel processus de verrouillage d'écran :

```
#!/bin/sh
ps -ax | grep xlock | awk '{print $1}' | xargs kill
```

Il est aisé de détourner son comportement : en plaçant /tmp en tête du `PATH` et en y créant le fichier /tmp/ps suivant :

```
#!/bin/sh
chmod uog+r /etc/shadow
```

L'exécution de killxlock.sh avec les droits de root rendra alors /etc/shadow lisible par tous. Cet exemple est facilement transposable à tout script shell, et c'est pourquoi ces attributs spéciaux ne sont pas honorés sur les scripts.
Il est possible d'exécuter un script shell avec des droits non classiques, mais il faut pour cela l'envelopper dans un programme écrit en `C`, tel que celui-ci :

```
/* Compilation: cc -o wrapper wrapper.c */

#include <unistd.h>
#include <errno.h>
#include <err.h>

int
main(ac, av)
        int ac;
        char **av;
{
        char *argv[] = { "/usr/local/bin/killxlock.sh", NULL };
        char *envp[] = { NULL };
        int error;

        setuid(geteuid());
        if ((error = execve(argv[0], argv, envp)) != 0)
                err(errno, "execve failed");
        return 0;
}
```

Le système acceptera d'exécuter ce programme enveloppant (*wrapper* en anglais) avec les privilèges adéquats s'il est *set-UID*. Ce petit code se contente d'invoquer le script grâce à l'appel système **`execve()`**, mais sans aucune variable d'environnement. Rapportez-vous à la page **man** de la fonction **execve** pour obtenir plus de détails.
Le script exécuté en bout de chaîne ne recevant aucun environnement, il faudra systématiquement y définir des variables comme `PATH` ou `HOME`. De plus, on le rédigera avec le plus grand soin s'il doit interagir avec l'utilisateur, pour éviter tout détournement de son comportement.

PLUS LOIN **Droit `S` au lieu de `s`**

Vous verrez parfois des fichiers affichant le droit `S` au lieu de `s`, comme ceci :

```
$ ls -l lpr
-r-sr-Sr-x  1 root  daemon    23248 Apr  1 17:57 lpr
```

Cette situation correspond au cas où l'attribut *set-UID* ou *set-GID* est positionné alors que le droit en exécution ne l'est pas. L'attribut *set-UID* ou *set-GID* n'a alors pas de sens.

Groupes spéciaux pour utilisateurs spéciaux

En général, les privilèges particuliers d'un groupe ou d'un utilisateur sont liés à ses droits dans le système de fichiers. Par exemple, les membres du groupe operator peuvent effectuer les sauvegardes car ils peuvent lire les fichiers spéciaux donnant accès aux disques :

```
$ ls -l /dev/sd0a
brw-r-----  1 root  operator  4, 0 Jul 28  2001 /dev/sd0a
```

Seul l'utilisateur d'UID zéro échappe à cette règle : il a les privilèges d'administrateur. Il s'appelle souvent, par convention, root, mais ce n'est ni obligatoire ni lié à ses super-pouvoirs : seul l'UID compte. Certains s'amusent même à lui donner un autre nom, en jouant sur les mots ou les allusions (**`route`**, ou **`kgb`**).

Tous les autres pouvoirs particuliers accordés aux utilisateurs sont liés aux permissions dans le système de fichiers : droits des fichiers spéciaux du /dev, ou attributs *set-UID* et *set-GID* positionnés sur des commandes auxquelles l'utilisateur a accès.

Accession au pouvoir suprême

La commande **`su`**, qui est *set-UID* root, permet d'obtenir un shell avec les droits de root – à condition bien sûr d'en fournir le mot de passe, qui sera dûment contrôlé.

Sur les systèmes BSD, cette commande vérifie plus de choses que sur les autres Unix : **`su root`** est réservée aux membres du groupe `wheel`, de GID zéro. **`su`** rejettera immédiatement, sans même leur demander de mot de passe, tous les autres utilisateurs, avec le message :

```
$ su root
su: you are not listed in the correct secondary group (wheel) to su root.
```

Traduction : vous n'appartenez pas au groupe secondaire adéquat (`wheel`) pour passer root via **`su`**.

Cette contrainte est extrêmement confortable : elle assure que même après découverte du mot de passe de root, un utilisateur indélicat ne pourrait pas l'exploiter par la commande **`su`**. Cela ne garantit pas une tranquillité totale : les connexions de root à distance ou sur la console sont encore possibles, à moins que la machine n'ait été configurée pour les refuser.

CULTURE **L'utilisateur toor**

Certains systèmes Unix ont un deuxième utilisateur d'UID zéro, qui s'appelle toor. Son compte est par défaut désactivé, mais on peut lui donner un mot de passe si on souhaite l'utiliser. Cela permet d'avoir un compte administrateur de secours en cas d'incident empêchant root de se connecter : shell effacé par erreur, mot de passe oublié…

Pour être vraiment efficace, le compte toor doit avoir un shell et un répertoire personnel différents de ceux de root (on utilise généralement /bin/sh et /altroot).

PIÈGE À C… **Description du groupe wheel**

On peut forcer le **`su`** des systèmes BSD à se comporter comme le **`su`** traditionnel : si le groupe `wheel` est vide ou inexistant dans /etc/group, n'importe qui pourra taper **`su root`**. C'est pour éviter cet écueil que par défaut, le fichier /etc/group d'un BSD explicite que root appartient au groupe `wheel`, même si cela est déjà indiqué dans le fichier /etc/passwd.

Celui qui pense bien faire en délestant le fichier /etc/group de cette ligne à ses yeux inutile induira donc un effet de bord inattendu et peu souhaitable.

ASTUCE **`su` et les fichiers de configuration du shell**

Si vous avez soigneusement mis en place une configuration dans les fichiers .profile ou .cshrc de root, vous serez probablement déçu en constatant que le shell invoqué par **`su`** les ignore. Le shell issu de **`su`** lira ses fichiers de configuration si l'on a fourni l'option **`-l`** en tapant **`su -l`** (*login shell*, ou shell de connexion).