Linux Administration

Tome 3

Sécuriser un serveur Linux

Jean-François Bouchaudy

Tsoft EDITEUR

 ISBN : 978-2-212-13462-9

EYROLLES

- *UID, GID*
- *login, su, sudo*
- *wtmp*
- *SUID, SGID, ACL*
- *CAP_CHOWN*

3

La sécurité locale

Objectifs

Ce chapitre présente la sécurité locale. Ses concepts de base sont rappelés : la sécurité d'un système multi-utilisateur, les droits, la sécurité de connexion. On présente également les « capabilities » du noyau et la commande `sudo`.

Contenu

La sécurité multi-utilisateur.

sudo.

La connexion.

Les mots de passe.

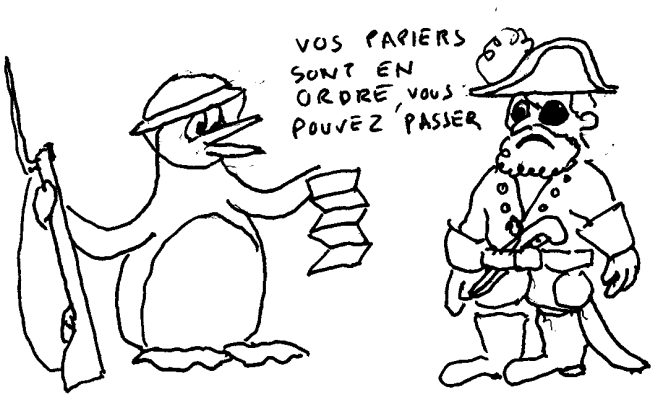
La sécurité pour les utilisateurs.

Les droits.

Les ACL.

Les « capabilities ».

Ateliers.



La sécurité multi-utilisateur

La théorie

Sur un système Linux, une application accède aux fichiers avec des restrictions. Par exemple, un serveur Apache ne peut transmettre une page Web que s'il a accès en lecture aux fichiers correspondants. Cette approche, appelée « sécurité multi-utilisateur » repose sur les concepts suivants :

- L'existence d'une base de comptes utilisateurs et d'une base de comptes groupes d'utilisateurs.
- Le fait qu'un fichier possède des droits précisant les utilisateurs et les groupes qui sont habilités à y accéder.
- Une application en cours d'exécution est associée à un compte utilisateur et à des comptes groupes, ce qui détermine ses droits d'accès aux fichiers.
- Les services (applications activées automatiquement par l'administrateur) sont associés à des comptes grâce à leurs fichiers de configuration.
- La connexion d'un utilisateur, le « login », associe son shell à un compte utilisateur et à un compte groupe. Ceci va déterminer ses droits d'accès, et par héritage, ceux de toutes les applications qu'il activera par la suite.
- L'administrateur (root) a tous les droits sur le système. Il peut créer des comptes et accéder à l'ensemble des fichiers sans restriction. Il peut déléguer une partie de ses prérogatives à certains utilisateurs…

Les caractéristiques d'un compte utilisateur

- Login, c'est le nom de l'utilisateur (ou de l'application).
- Mot de passe, il est utilisé lors de la connexion pour authentifier l'utilisateur.
- UID, ce numéro identifie l'utilisateur (User IDentification).
- GID, ce numéro spécifie le groupe principal de l'utilisateur (Group IDentification)
- Commentaire.
- Répertoire de connexion.
- Shell, ce logiciel, le plus souvent un véritable shell, est activé en début de session en mode texte.

IMPORTANT ! L'UID 0 est réservé. Toute application ayant cet UID a tous les droits sur le système. C'est l'UID de l'administrateur root.

Les caractéristiques d'un compte groupe

- Le nom du groupe.
- GID, ce numéro identifie le groupe.
- Un mot de passe. Cette valeur n'est jamais renseignée.
- La liste des membres en tant que membres secondaires, ce qui exclut les comptes dont c'est le groupe principal.

Le savoir concret

Les fichiers

/etc/nsswitch.conf	Indique dans quels annuaires locaux ou réseaux (NIS, LDAP…) sont recherchés les comptes.
/etc/passwd	Contient la base locale des comptes utilisateurs.
/etc/group	Contient la base locale des comptes groupes.
/etc/default/useradd	Configure la commande `useradd`.

Les commandes

`useradd, usermod, userdel`	Ajout, modification, destruction d'un compte utilisateur.
`groupadd, groupmod, groupdel`	Ajout, modification, destruction d'un compte groupe.
`id`	Affiche les identités d'un compte.
`getent`	Affiche les données d'un annuaire (passwd, group, shadow).
`pwck, grpck`	Vérifie la syntaxe des fichiers passwd et group.
`sudo`	Exécute une commande avec les droits d'un autre utilisateur. L'administrateur peut, en configurant cette commande, déléguer une partie de ses prérogatives ou celles de tout autre compte.

Les particularités des distributions

Debian/Ubuntu

La commande `adduser` crée un compte utilisateur, elle est différente de la commande `useradd`.

Pour en savoir plus

Man

useradd(8), usermod(8), userdel(8), groupadd(8), groupmod(8), groupdel(8), chfn(1), chsh(1), pwck(8), grpck(8), sudo(8)

Howto

Le User-Group HOWTO

Internet

Gestion des comptes utilisateur
http://docs.redhat.com/docs/mr-IN/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/ch-users-groups.html

sudo

La théorie

Grâce à la commande `sudo`, des commandes nécessitant a priori les prérogatives de root, peuvent être exécutées par un compte ordinaire.

Le savoir concret

La commande sudo

La commande `sudo` permet à un utilisateur d'exécuter des commandes qui nécessitent une identité différente de l'utilisateur, par exemple celle de root. Exemple :

```
pierre$ sudo  useradd   paul
```

Le fichier /etc/sudoers

Ce fichier décrit qui peut activer une commande avec `sudo` et sous quelle identité.

Syntaxe d'une ligne d'endossement de privilèges

```
Qui       hôte = (identité) commande [,…]
```

Exemple :

```
pierre  ALL = (root) /usr/bin/useradd, /usr/bin/usermod
```

L'utilisateur pierre peut activer les commandes `useradd` et `usermod` avec les droits de root.

Les alias

Pour simplifier l'écriture du fichier, on définit des alias (il faut spécifier la catégorie de l'alias en 1re colonne). Exemple :

```
User_Alias          ADM_USERS = pierre, paul
Runas_Alias         OPERATOR = root
Cmd_Alias           USER_CMDS = /usr/bin/useradd, /usr/bin/usermod
ADM_USERS           ALL = (OPERATOR) USER_CMDS
```

Mot de passe

Par défaut, la commande `sudo` demande le mot de passe de l'utilisateur. Le mot-clé NOPASSWD peut précéder les commandes. Dans ce cas, le mot de passe n'est pas demandé.

Exemple :

```
%admins ALL = (root) NOPASSWD:  USER_CMDS
```

Remarque
Un groupe est précédé du caractère %.

La commande sudoedit

La commande `sudoedit` édite un fichier avec les privilèges d'autrui.

Pour en savoir plus

Man

sudo(8), sudoers(5), visudo(8), sudoedit(8)

La connexion

La théorie

C'est durant sa connexion qu'un utilisateur est authentifié. Ensuite une application, habituellement un shell, est associée à son compte via l'API `setuid()`. Toutes les commandes ou applications qu'il activera par la suite seront associées à ce compte.

Les applications exécutées sont également associées à des groupes d'utilisateurs. C'est le fait qu'une application soit taguée par un UID et des GID qui lui confère ses droits d'accès aux fichiers.

Durant la connexion d'un utilisateur, on lui demande son nom. Pour l'authentifier, on lui demande également, le plus souvent, un mot de passe. Il est possible d'utiliser d'autres techniques d'authentification, notamment la biométrie.

Pour rendre les applications réalisant la connexion plus indépendantes des méthodes concrètes d'authentification, Linux, comme la plupart des systèmes Unix, a choisi le système PAM. Une application utilisant PAM demande par exemple à un utilisateur de s'authentifier. Les techniques d'authentification choisies sont simplement consignées dans un fichier de configuration.

Le savoir concret

Les principales commandes réalisant l'authentification

`login`	Assure la connexion locale.
`sshd`	Réalise la connexion distante via un tunnel SSH.
`gdm`	Effectue la connexion en mode graphique (suite Gnome).
`kdm`	Idem (suite KDE).
`xdm`	Idem (outil d'origine MIT).
`su`	Réalise une connexion secondaire, on doit être déjà connecté.

Remarque
L'ensemble de ces commandes utilise le système PAM pour effectuer l'authentification.

Autres commandes

`who`	Liste les utilisateurs actuellement connectés.
`last`	Affiche les dernières connexions ayant abouti.
`last -a`	Idem, affiche l'hôte à l'origine de la connexion.
`lastb`	Affiche les dernières connexions ayant échoué.
`lastlog`	Affiche tous les utilisateurs et leur dernière connexion.

Les fichiers

/etc/passwd	Les comptes utilisateurs locaux.
/etc/group	Les comptes groupes locaux.
/etc/pam.d/*	La configuration PAM.
/var/log/wtmp	Fichier binaire contenant l'historique des connexions.

/var/run/utmp	Fichier binaire contenant la liste des utilisateurs actuellement connectés.
/var/log/btmp	Fichier binaire contenant l'historique des connexions ayant échoué.

Pour en savoir plus

Man

login(1), setuid(2), pam(8), sshd(8), gdm(1), su(1), last(l), lastb(1), lastlog(8)

Les mots de passe

La théorie

Il existe plusieurs méthodes d'authentification. L'utilisation de mots de passe est la plus répandue. D'après le SANS, de mauvais mots de passe ou une mauvaise gestion de ceux-ci constitue la deuxième plus importante faille d'un système (la 1re étant l'absence de mise à jour des logiciels).

Un bon mot de passe – ce qu'il ne faut pas faire

- Ne pas utiliser vos nom, prénom ou nom de connexion (login).
- De manière générale ne pas utiliser un mot qui a un rapport avec vous : le nom, le prénom ou la date d'anniversaire d'un membre de votre famille, d'un ami, de votre chef, de votre animal familier, le nom de votre ancienne école, votre département, votre numéro de téléphone, votre numéro de sécurité sociale, le nom de votre ordinateur…
- Des mots du dictionnaire (en français, en anglais, ou en toute autre langue).
- Des noms propres.
- Des suites de lettres composant des motifs sur votre clavier (azerty, qwerty…). Par exemple aqwsefvgy représente un W sur un clavier AZERTY.
- Un mot à l'envers.
- Un mot quelconque précédé ou suivi d'un seul chiffre.
- En bref, tout ce qui peut être deviné.
- Ne pas utiliser le mot de passe fourni par défaut. Il faut immédiatement le changer.
- Ne pas laisser votre compte sans mot de passe. Ajoutez-en un immédiatement.

Un bon mot de passe – ce qu'il faut faire

- Un bon mot de passe doit être long (au moins 8 caractères).
- Un bon mot de passe doit paraître aléatoire, mais néanmoins être facilement mémorisé.
- Utiliser des minuscules et des majuscules.
- Utiliser des chiffres et des signes de ponctuation en plus de lettres.
- Un bon mot de passe doit pouvoir être tapé rapidement (pour éviter l'espionnage).

Un bon mot de passe – comment en générer un

- Utiliser les initiales des mots d'une phrase. Par exemple : `jpLaOS400` créé à partir de la phrase « Je préfère Linux à OS/400 ».
- Mélanger deux mots courts. Par exemple `GlRiUlBo`, mélange des mots GRUB et lilo.
- Une suite de syllabes sans signification, mais facile à retenir. Par exemple : `LoCiDaVuKo`, `ribeloPAMU`.

Les bonnes pratiques

- Il faut changer son mot de passe régulièrement. Par exemple tous les quatre mois.
- Il ne faut pas taper son mot de passe si l'on est observé.
- Il ne faut jamais écrire sur papier son mot de passe. Si l'on est obligé de le faire, il faut enfermer ce papier dans un coffre.
- Il ne faut jamais transmettre son mot de passe à quelqu'un par téléphone ou par e-mail.
- De manière générale, ne pas transmettre à quiconque votre mot de passe.

Remarque
La gestion de la pérennité des mots de passe s'appelle le *password aging*.

La politique de l'administrateur

L'administrateur doit enseigner aux utilisateurs les bonnes pratiques et comment créer un bon mot de passe. Il peut aussi, via la configuration du système, les contraindre à respecter les règles de sécurité.

S'il fait confiance aux utilisateurs pour choisir leur mot de passe, il doit s'assurer qu'ils sont de bonne qualité. La meilleure technique pour ce faire est d'essayer (comme un pirate) de les cracker.

Le savoir concret

Les commandes

`login, sshd…`
: Commandes réalisant la connexion d'un utilisateur. Elles invoquent des modules PAM. Certains de ces modules authentifient l'utilisateur par un mot de passe.

`passwd`
: Permet de modifier son mot de passe. Permet à l'administrateur de changer le mot de passe d'un utilisateur quelconque.

`chpasswd`
: Modifie les mots de passe de manière scriptable.

`chage`
: Gère le password aging. Notamment la durée de validité d'un mot de passe.

`htpasswd`
: Cette commande Apache gère un fichier de comptes où les mots de passe sont chiffrés avec la fonction standard crypt().

`john`
: Ce logiciel essaye de cracker les mots de passe.

`pwconv`
: Extrait les mots de passe de /etc/passwd et les met dans /etc/shadow. Cette commande est exécutée lors de l'installation du système.

`pwunconv`
: Fait l'opération inverse de la commande précédente. Elle réintègre les mots de passe dans le fichier /etc/passwd.

`gpasswd`
: Gère les fichiers /etc/group et /etc/gshadow.

Les fichiers

/etc/passwd
: Le fichier qui contient les comptes utilisateur. Antérieurement il contenait également les mots de passe (d'où son nom).

/etc/shadow
: Contient les mots de passe et les données associés à leur pérennité.

/etc/login.defs
: Contient les valeurs par défaut du password aging.

Les modules PAM

`pam_unix`	Authentifie l'utilisateur en lui demandant son mot de passe. C'est le module standard d'authentification.
`pam_cracklib`	Vérifie la qualité d'un mot de passe.
`pam_userdb`	Authentifie l'utilisateur par rapport à un mot de passe stocké dans une base de données de type BDB et chiffré avec la fonction ISO crypt() (basée sur le DES).

Focus : la commande chage

Syntaxe : `chage [options] utilisateur`

-M jours	Fixe la durée de validité d'un mot de passe.
-E epoque	Fixe la date d'expiration du mot de passe. L'époque peut être fournie au format YYYY-MM-DD, par exemple 2007-12-31 pour le 31 décembre 2007.
-m jours	Fixe le nombre minimum de jours avant que l'on puisse modifier son mot de passe.
-w jours	Fixe le nombre de jours pendant lesquels l'utilisateur est prévenu de l'imminence de l'expiration de son mot de passe.
-l	Liste les valeurs des attributs du password aging.

Focus : la commande passwd

-l	Verrouille un compte.
-u	Déverrouille un compte.
-d	Supprime le mot de passe d'un compte.
-S	Affiche des informations sur le mot de passe du compte.

Focus : John The Ripper et l'attaque au dictionnaire

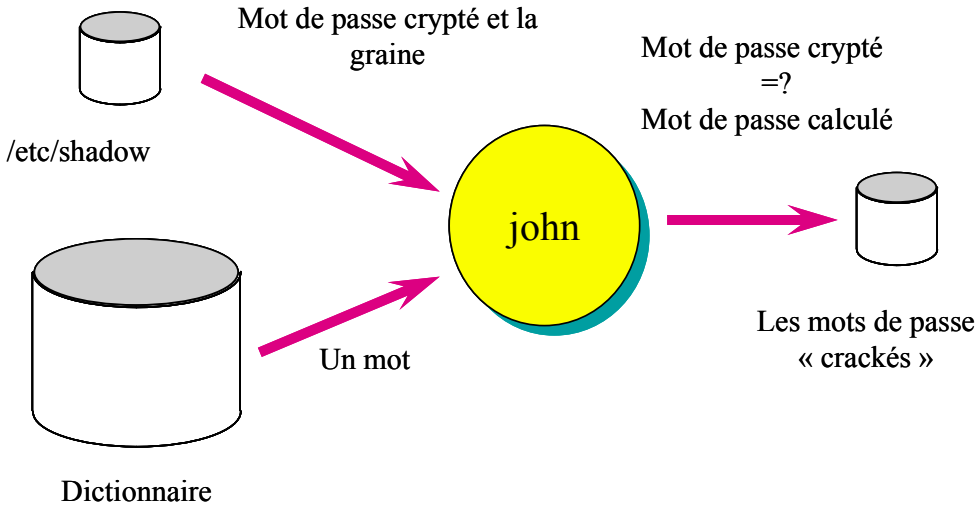


Fig. L'attaque au dictionnaire avec John

Il existe plusieurs logiciels destinés à casser les mots de passe, John The Ripper est le plus connu. L'attaque la plus fructueuse est l'attaque au dictionnaire. Son principe est simple : on

récupère d'abord les mots de passe cryptés (et les graines s'il y en a), ensuite on chiffre chaque mot d'un fichier qualifié de dictionnaire. Si le résultat est identique à un des mots de passe, on a cassé un mot de passe. Le fichier dictionnaire contient une liste de mots qui sont susceptibles d'être choisis comme mots de passe. C'est souvent un fichier qui dérive d'un vrai dictionnaire (ou plutôt de plusieurs) auquel on a fait subir des modifications telles que l'ajout d'un chiffre devant ou derrière le mot. Ainsi, un dictionnaire destiné à casser des mots de passe contient rapidement des millions d'entrées (pour mémoire, le français ne comprend qu'environ 30 à 40 000 mots seulement).

Les commandes

`john`	Casse les mots de passe. Utilise les trois modes d'attaque : attaque triviale, au dictionnaire et exhaustive.
`unshadow`	Crée un fichier contenant les comptes et les mots de passe chiffrés devant être cassés. Ce fichier doit être donné en argument de john.

Les arguments de la commande john

–single	Casse les mots de passe triviaux.
–w:dictionnaire	Casse les mots de passe en utilisant un dictionnaire.
–incremental	Casse les mots de passe de manière exhaustive (temps de calcul infini).
–show	Affiche les mots de passe trouvés.

Les fichiers

john.conf	Le fichier de configuration
password.lst	Le dictionnaire fourni en standard. Il est utilisé implicitement quand john est utilisé sans option.
passwd	Le fichier qui contient les mots de passe à casser. Il a le format de /etc/passwd.

Les particularités des distributions

RedHat

L'option --stdin de la commande `passwd` permet de changer un mot de passe en automatique.

SuSE

Le module PAM `pam_pwcheckIdem` vérifie la qualité d'un mot de passe.

La commande `passwd` permet de changer en automatique les mots de passe via l'option --stdin, mais on peut aussi utiliser la commande `chpasswd`. Dans l'exemple suivant on donne le mot de passe « pass » à l'utilisateur paul:

```
# echo "paul:pass" | chpasswd –c blowfish
```

Le paquetage `john` est disponible, il peut être complété par le paquetage `john-wordlists` qui contient des dictionnaires de cassage de mots de passe.

Debian/ Ubuntu

La commande Debian `passwd` ne connaît pas l'option --stdin. Si l'on veut créer un mot de passe en automatique, on peut utiliser la commande `usermod` avec l'option –p :

```
# usermod –p $(mkpasswd --hash=sha-512 guest)  guest
```

En dehors du paquetage `john` , il existe plusieurs autres logiciels génériques destinés à casser les mots de passe : `crack`, `lcrack` et `medusa`. Cette dernière commande permet même une attaque distribuée.

Il existe aussi des logiciels spécialisés : `aircrack-ng` (pour les mots de passe WEP/WPA), `fcrackzip` (pour les fichiers ZIP), `pdfcrack` (pour les fichiers PDF), `ophcreack` (pour Windows) et `sipcrack` (pour SIP).

Les commandes `makepasswd` et `pwgen` génèrent des mots de passe.

La commande `pwman3` gère les différents mots de passe d'un utilisateur. Elle fonctionne en mode console.

Les logiciels `password-gorilla` et `keepassx` stockent vos mots de passe dans un fichier crypté. Ainsi, il suffit de mémoriser un seul mot de passe. Ces applications graphiques fonctionnent sous plusieurs plate-formes (Linux, Windows...).

Pour en savoir plus

Man

shadow(5), passwd(5), login(1), login.defs(5), crypt(3), chpasswd(8), chage(8), passwd(1), pwconv(8), pwunconv(8), grpconv(8), grpunconv(8)

Howto

Shadow-password Howto

Internet

John The Ripper
http://www.openwall.com/john/

Un dictionnaire (wordlist)
http://www.cs.indiana.edu/classes/a306/word_list.txt

Les 10 meilleurs outils de cassage de mots de passe
http://sectools.org/crackers.html

La sécurité pour les utilisateurs

La théorie

L'administrateur doit enseigner aux utilisateurs un certain nombre de règles de sécurité. Voici les plus importantes.

- Les utilisateurs doivent choisir un bon mot de passe. Il doit être long, facilement mémorisable et difficilement devinable.
- Un utilisateur doit veiller à saisir son mot de passe sans être surveillé.
- Votre répertoire de connexion doit être privé, personne ne doit pouvoir y accéder.
- Il faut définir une valeur de UMASK restrictive, quitte à étendre les droits d'accès de certains de vos fichiers.
- Il ne faut pas abandonner son terminal sans se déconnecter. À défaut, il faut le verrouiller ou provoquer une déconnexion automatique en cas d'inactivité prolongée.
- Il faut prêter attention aux dates de dernières connexions réussies et infructueuses qui sont affichées à chaque connexion.
- Ne pas laisser l'accès, même en lecture, à son fichier .bash_profile.
- Faire attention aux répertoires qui sont nommés dans la variable PATH. Il ne faut pas par exemple mettre le répertoire « . » en première position.

Le savoir concret

Les commandes

`vlock`	Verrouille le terminal courant.
`vlock –a`	Verrouille l'ensemble des terminaux virtuels (sur la console maîtresse).
`umask`	Change ou affiche le UMASK de votre shell. Une valeur 77 est conseillée.

Les fichiers

~	Votre répertoire de connexion. Il doit avoir les droits 700.
~/.bash_profile	Votre fichier d'initialisation de votre session. Il doit avoir les droits 400. Il contient notamment le paramétrage des variables d'environnement et du UMASK.

Les variables d'environnement

TMOUT	Cette variable contient le temps d'inactivité (en secondes) au bout duquel la déconnexion a lieu automatiquement.
PATH	Contient les chemins des répertoires contenant les commandes. Ne doit pas contenir une référence au répertoire courant (« . »).

Les environnements graphiques

Gnome

Pour verrouiller l'écran, on utilise la commande du menu `system/Lock Screen`.

Il est possible d'ajouter un bouton pour verrouiller l'écran et activer l'économiseur. Pour travailler de nouveau, vous devez fournir votre mot de passe.

Pour ajouter un tel bouton : clic droit dans une partie vide du panneau, choisir « Add to Panel » et ensuite « Lock Screen ».

La configuration de l'économiseur d'écran (sreensaver) permet d'indiquer le laps de temps au bout duquel il est activé automatiquement ainsi. Elle précise aussi si le verrouillage de l'écran est effectué simultanément.

Les particularités des distributions

SuSE

La commande `xlockmore` active un économiseur d'écran qui permet de verrouiller la session.

Debian/ Ubuntu

Le logiciel `xtrlock` permet de verrouiller une session graphique.

Pour en savoir plus

Man

vlock(1), bash(1)

Les droits

La théorie

Les catégories d'utilisateurs

Lors de l'accès à un fichier, le noyau Linux considère trois catégories d'utilisateurs :

- Le propriétaire du fichier (user ou u).
- Les membres du groupe (group ou g) auquel est affilié le fichier.
- Les autres utilisateurs (other ou o).

Pour chaque catégorie, il existe trois droits d'accès, dont la signification dépend de la nature du fichier : ordinaire ou répertoire.

Les droits pour un fichier ordinaire

- Le droit de lecture (read ou r) permet de lire les octets du fichier.
- Le droit d'écriture (write ou w) permet d'ajouter, supprimer ou modifier des octets.
- Le droit d'exécution (execute ou x) permet de considérer le fichier comme une commande.

Les droits pour un répertoire

- Le droit de lecture (r) permet de connaître la liste des fichiers du répertoire.
- Le droit d'écriture (w) permet de modifier le répertoire : créer ou supprimer des entrées dans le répertoire.
- Le droit d'accès (x) permet d'accéder aux fichiers du répertoire.

IMPORTANT ! Le dernier droit, le droit d'accès, est le plus important. Sans lui une personne n'a aucun accès aux fichiers présents dans le répertoire, quels que soient leurs droits.

Le sticky bit

Ce droit, réservé à root, s'applique à un répertoire et corrige une bizarrerie du système. Par défaut, un répertoire accessible en écriture à un ensemble d'utilisateurs permet à l'un d'entre eux de détruire les fichiers d'un autre utilisateur. Avec le sticky bit il faut être propriétaire d'un fichier pour avoir le droit de le détruire.

Les droits d'endossement (SUID, SGID) pour un exécutable

La philosophie des droits d'endossement est d'augmenter les privilèges des utilisateurs. Par exemple, le droit Set-UID (SUID) sur un binaire exécutable permet à l'utilisateur de l'application correspondante d'avoir les mêmes droits d'accès que le propriétaire du binaire. Le droit Set-GID (SGID) permet, lui, d'endosser les droits du groupe auquel est affilié le binaire.

Exemple : le fichier /etc/shadow n'est en théorie accessible qu'à root. Or, tout utilisateur a accès en écriture à ce fichier lorsqu'il change son mot de passe grâce à la commande /usr/bin/passwd. L'explication réside dans le fait que cette commande, possédée par root, détient le droit SUID et donne de fait à tous les utilisateurs les mêmes droits que root.

ATTENTION ! On vient de le constater, les droits d'endossement sont pratiques. Il n'en demeure pas moins qu'ils sont dangereux. La sécurité dans ce cas ne réside plus que dans le code même de l'application.

La sécurité et les droits d'endossement

Un code malicieux possédé par l'administrateur et ayant le droit SUID constitue une faille majeure. Un utilisateur en exécutant ce code accomplit des actions avec les droits de l'administrateur.

Un audit minimum d'un système Linux recherche les applications possédant ces droits. Moins il y en a, meilleure est la sécurité. Un pirate ne doit pas pouvoir ajouter ou modifier de telles applications.

Le droit SGID pour un répertoire

Lorsque l'on crée un fichier, il est automatiquement affilié à son groupe courant, qui est par défaut son groupe principal. Si l'on crée un fichier dans un répertoire qui possède le droit SGID, son groupe sera identique à celui du répertoire. La conséquence c'est que l'ensemble des fichiers du répertoire appartiendra au même groupe, ce qui est intéressant pour un répertoire accessible à plusieurs personnes.

Le droit de modifier les droits, le « by-pass » de root

Le droit de modifier les droits est un droit inaliénable du propriétaire du fichier. Ce dernier peut également modifier le groupe auquel est affilié le fichier à condition d'être membre du nouveau groupe.

L'administrateur (root) peut également changer les droits ou le groupe de n'importe quel fichier. Il a aussi un accès sans restriction, on dit qu'il a un « by-pass », sur l'ensemble des fichiers, quels que leurs droits.

Un fichier appartient par défaut à celui qui le crée. L'administrateur peut modifier le propriétaire d'un fichier.

Au-delà des droits

Le système des droits que l'on vient de présenter est issu des systèmes Unix. Un système Linux est plus complexe. Il possède d'autres mécanismes qu'il faut considérer pour comprendre les restrictions d'accès aux fichiers :

- Les ACL
- Les attributs Ext2
- Les options de montage des FS
- Les capabilities
- La sécurité SELinux

Le savoir concret

Les commandes de gestion des droits

`ls -l`	Liste les caractéristiques d'un fichier, dont les droits.
`chmod`	Modifie les droits d'un fichier.
`chgrp`	Change le groupe d'un fichier.
`chown`	Change le propriétaire d'un fichier.
`umask`	Fixe les droits retirés automatiquement lors de la création d'un fichier.
`cp -p`	Copie de fichiers avec conservation des attributs.
`find`	Recherche des fichiers selon différents critères. L'option –perm spécifie les droits des fichiers recherchés.

Les commandes de gestion des attributs Ext2

`lsattr`	Liste les attributs Ext2 d'un fichier.
`chattr`	Modifie les attributs Ext2 d'un fichier.

Les options de montage des FS

Voici les options générales de montage (préfixées par –o) qui affectent la sécurité :

`noexec`	Interdit l'exécution de programmes binaires présents sur le FS.
`nodev`	Interdit l'accès aux périphériques présents sur le FS.
`nosuid`	Interdit le comportement SUID des programmes présents sur le FS.

Remarque
Chaque type de FS possède des options spécifiques qui peuvent affecter la sécurité. Par exemple le FS iso9660 (le CD-rom) a les options uid= et gid= qui forcent le propriétaire et le groupe des fichiers du FS.

Les droits en octal

4000	Le droit SUID (u+s).
2000	Le droit SGID (g+s).
1000	Le sticky bit (+t).
0400	Le droit de lecture pour le propriétaire (u+r).
0200	Le droit d'écriture pour le propriétaire (u+w).
0100	Le droit d'exécution pour le propriétaire (u+x).
0040	Le droit de lecture pour le groupe (g+r).
0020	Le droit d'écriture pour le groupe (g+w).
0010	Le droit d'exécution pour le groupe (g+x).
0004	Le droit de lecture pour les autres (o+r).
0002	Le droit d'écriture pour les autres (o+w).
0001	Le droit d'exécution pour les autres (o+x).

Pour en savoir plus

Man

ls(1), chmod(1), chmod(2), chgrp(1), chown(1), umask(1), lsattr(1), chattr(1), find(1)

Les ACL

La théorie

Les droits Linux ordinaires (ISO) sont restreints : il y a les droits qui s'appliquent au propriétaire, ceux qui s'appliquent aux membres du groupe et ceux qui s'appliquent aux autres.

Les ACL Linux, inspirés d'un draft POSIX, vont plus loin : ils permettent de positionner une liste de contrôle d'accès (ACL=Access Control List) associée à un fichier. Chaque élément de cette liste fixant les droits d'un utilisateur ou d'un groupe par rapport au fichier.

Remarque
Les ACL sont prioritaires sur les droits ISO.

Le concept de masque

Le masque ACL fait partie des ACL d'un fichier. Il précise si les ACL doivent être pris en compte en partie (pour r, w ou x), en totalité ou pas du tout.

Les ACL par défaut

La gestion des ACL peut être très lourde si l'on fixe les ACL fichier par fichier.

Les ACL par défaut simplifient les choses : on fixe des ACL par défaut au niveau d'un répertoire ; tout fichier créé ensuite dans ce répertoire héritera par défaut de ces ACL.

Remarque
Les ACL par défaut sont une technique qui permet de simuler l'héritage des droits existant sous Windows.

Implémentation

Les ACL sont gérés par le noyau au niveau du VFS et des pilotes de FS. Les ACL sont pris en charge par les FS Xfs, Ext2/Ext3, ReiserFS et NFS.

Les ACL sont mémorisés sur disque dans les structures de données des FS. Ils sont activés lors du montage du FS. Dans le cas des FS Ext2/Ext3, c'est une option de montage.

Le savoir concret

Les commandes

`getfacl`	Visualise les ACL d'un fichier.
`setfacl`	Crée, modifie, supprime les ACL d'un fichier.
`ls -l`	Liste les attributs d'un fichier, indique la présence d'ACL.
`mount`	Monte un FS. L'option –o acl active l'utilisation des acl.

Focus : la syntaxe de setfacl

```
setfacl  -m  ACL[,…]  fichier…
```

Syntaxe d'une ACL :

```
<type_d_ACL>:[<valeur>]:<droits>
```

Exemples :

user:pierre:rw-	Pierre a les droits de lecture et d'écriture (r et w).

group:g1:r---	Le groupe g1 a le droit de lecture (r).
mask::rwx	Les ACL sont totalement pris en compte.
mask::---	Les ACL sont désactivés.
mask::r--	Seul le droit de lecture est pris en compte dans les ACL.
default:user:paul:rw-	Exemple d'ACL par défaut (pour un répertoire).

Remarque
La commande `setfacl` peut également modifier les droits ordinaires (ISO) avec les pseudo ACL suivants : user::<droits>, group::<droits> et other::<droits>.

Sauvegarde des ACL

DANGER ! Par défaut, les ACL ne sont pas sauvegardés par les outils classiques de sauvegarde. Ainsi, si l'on utilise la commande `tar`, il faut penser à ajouter l'option `--acls`.

Par contre, il est facile de les sauvegarder avec la commande `getfacl -R` et ensuite de les restaurer avec la commande `setfacl --restore`.

Pour en savoir plus

Man

getfacl(1), setfacl(1), ls(1), star(1), tar(1)

Internet

La gestion des ACL
http://lea-linux.org/documentations/index.php/Gestion_des_ACL

Les « capabilities »

La théorie

L'administrateur root a des prérogatives immenses : il peut supprimer n'importe quel fichier, il peut attribuer un fichier à un compte quelconque, il peut modifier l'heure système, etc. Le concept de capability est d'isoler chacune de ces prérogatives pour pouvoir les rendre séparément accessibles (ou non) à un processus utilisateur.

(Extrait de la FAQ)

Un processus a trois ensembles de capabilities : héritables (I), permisses (P) et effectives (E). Chaque capability est positionnée ou non. Chaque ensemble est implémenté sous forme d'un bitmap. Chaque bit représentant une capability.

Quand un processus essaye d'effectuer une opération privilégiée, le noyau vérifie le bit approprié du bitmap effectif. Par exemple, quand un processus essaye de modifier l'horloge, le noyau vérifie si le processus a la capability CAP_SYS_TIME positionnée.

Le bitmap permis indique les capabilities que le processus peut modifier. Les processus peuvent avoir des capabilities positionnées dans l'ensemble permis et qui ne sont pas positionnés dans l'ensemble effectif. Ceci indique que le processus a temporairement désactivé cette capability. Un processus n'est autorisé à positionner une capability que si elle est positionnée dans l'ensemble permis. La distinction entre permis et effectif existe de telle sorte qu'un processus peut mettre entre parenthèses des opérations qui nécessitent des privilèges.

L'ensemble héritable correspond aux capabilities qui seront héritées par un programme exécuté par le programme courant. L'ensemble permis d'un processus est masqué par rapport à l'ensemble héritable durant l'appel système exec(). Rien de spécial n'arrive durant l'appel fork(). le processus fils possède une copie des capabilities de son père.

Remarque : les fichiers exécutables mémorisent les capabilities (à partir du noyau 2.6.24) dans des attributs étendus (xattr). Les processus peuvent ainsi obtenir leurs capabilities directement à partir de la configuration de l'exécutable.

Savoir concret

Quelques capabilities

CAP_CHOWN	Permet de changer le propriétaire et le groupe d'un fichier.
CAP_KILL	Permet d'envoyer un signal à un processus quelconque.
CAP_NET_RAW	Permet d'utiliser les sockets raw et packet.
CAP_SYS_TIME	Permet de manipuler l'horloge système.
CAP_SETPCAP	Peut transférer ou retirer des capabilities à d'autres processus.
CAP_NET_BIND_SERVICE	Permet d'écouter sur un port réseau inférieur à 1024.
CAP_LINUX_IMMUTABLE	Permet de changer le caractère inchangeable d'un fichier (c'est un des attributs Ext2).
CAP_SETUID	Permet de changer l'UID.
CAP_SETGID	Permet de changer le GID.
CAP_DAG_OVERRIDE	Permet d'outrepasser les droits des fichiers.

CAP_DAG_READ_SEARCH	Permet d'outrepasser les droits lecture et exécution des fichiers.
CAP_SYS_CHROOT	Permet d'utiliser l'appel système `chroot()`.
CAP_SYS_MODULE	Permet de charger ou décharger un module du noyau.
CAP_SYS_NICE	Permet d'augmenter la priorité avec nice ou renice.

Commandes

`getpcaps`	Affiche les capabilities des processus donnés en paramètres.
`getcap`	Visualise les capabilities d'un exécutable.
`setcap`	Fixe les capabilities d'un exécutable.

Bibliothèques

pam_cap.so	Fixe les capabilities d'un processus de connexion (login...) en se basant sur celles inscrites dans l'exécutable (fixés par `setcap`). Le module est configuré par le fichier /etc/security/capability.conf qui spécifie quel utilisateur se voit attribué quelle capability.

Fichiers

libcap	La bibliothèque qui implémente les capabilities POSIX 1.e.
/proc/<PID>/status	Les caractéristiques d'un processus. Affiche notamment les capabilities (CapInh...).

Les particularités des distributions

Debian

La commande `lcap` visualise ou enlève des capabilities du noyau pour améliorer la sécurité.

Pour en Savoir plus

Man

capabilities(7)[liste les capabilities], cap_get_proc(3), cap_set_proc(3), cap_from_text(3),

Documentation du paquetage

libcap(rpm) (/usr/share/doc/libcap-*/capfaq-*.txt)

Fichiers d'en-tête

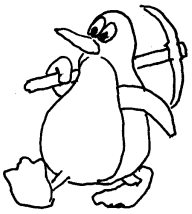
/usr/include/linux/capability.h (kernel-headers)

Internet

Introduction to Linux Capabilities and ACL's
http://www.symantec.com/connect/articles/introduction-linux-capabilities-and-acls

What is a Capability, Anyway?
http://www.eros-os.org/essays/capintro.html

ATELIERS



Tâche 1 :
Casser des mots de passe via un simple script et un dictionnaire maison

1. Créer des comptes utilisateur.

```
# useradd -m francois
# useradd -m firmin
# useradd -m albert
# useradd -m alice
# useradd -m beatrice
# useradd -m gisele
# useradd -m tux
# useradd -d /root -u 0 -o toor
# echo francois |passwd --stdin francois
# echo nimrif |passwd --stdin firmin
# echo bertal |passwd --stdin albert
# echo benedicte |passwd --stdin alice
# echo america7 |passwd --stdin beatrice
# echo belladone |passwd --stdin gisele
# echo shileyon | passwd --stdin toor
# passwd -d tux
Removing password for user tux.
passwd: Success
```

2. Créer un dictionnaire.

```
# vi dico.txt
secret
francois
benedicte
oss117
```

3. Créer le script d'attaque.

```
# vi attakOdico.pl
#!/usr/bin/perl
```

```
$motpasse = shift ;
@shadow = <STDIN> ;

foreach $lig (@shadow) {
        ($nom,$pass,$reste) = split /:/, $lig;
        ($vide,$method,$graine,$passcry) = split /\$/, $pass ;
         $essai = crypt("$motpasse","\$$method\$$graine\$");
         if ($essai eq $pass) {
                print ">>>BINGO!!!: $nom  -> $motpasse\n";
         }
}
```

4. Lancer l'attaque.

```
# cp /etc/shadow shadow
# for nom in $(cat dico.txt);do perl attak0dico.pl $nom < shadow ;done
>>>BINGO!!!: root  -> secret
>>>BINGO!!!: bob  -> secret
>>>BINGO!!!: francois  -> francois
>>>BINGO!!!: alice  -> benedicte
>>>BINGO!!!: hubertbdlb  -> oss117
```

Tâche 2 :
Casser des mots de passe avec John The Ripper

1. Télécharger John à partir de son site officiel (www.openwall.com/john) et l'installer.

Remarque
Le compilateur doit être présent, sinon il faut l'installer :
`# yum install 'Development Tools'`

a) Télécharger le logiciel et le détarer.

```
[root@linux01 ~]# wget 'http://www.openwall.com/john/f/john-1.7.8.tar.gz'
[root@linux01 ~]# tar xzf john-1.7.2.tar.gz
```

b) Patcher le logiciel pour inclure la gestion de sha512.

Remarque : cette mise à jour est complexe, nous renvoyons le lecteur au site suivant pour les détails :
Crack Password with John the Ripper on Ubuntu 9.10
http://pka.engr.ccny.cuny.edu/~jmao/node/26
Le lecteur peut également obtenir John déjà patché sur le site TSOFT.

c) Compiler le logiciel.

```
[root@linux01 ~]# cd john-1.7.2/src
[root@linux01 src]# make
...
linux-x86-sse2           Linux, x86 with SSE2 (best)
linux-x86-mmx            Linux, x86 with MMX
linux-x86-any            Linux, x86
...
[root@linux01 src]# make linux-x86-any
...
rm -f ../run/unique
ln -s john ../run/unique
make[1]: Leaving directory `/root/john-1.7.8/src'
[root@linux01 src]# cd ..
```

2. Créer des comptes utilisateur avec de mauvais mots de passe ou sans mot de passe (cf. tâche précédente).

3. Créer le fichier des mots de passe.

```
# run/unshadow /etc/passwd /etc/shadow > passwd.l
# chmod 600 passwd.l
```

4. Attaque élémentaire, on visualise les résultats.

```
# run/john -single passwd.1
Loaded 7 password hashes with 7 different salts (generic crypt(3) [?/32])
francois         (francois)
nimrif           (firmin)
bertal           (albert)
guesses: 3  time: 0:00:02:35 100%  c/s: 48.67  trying: alice1909 - alice1900
Use the "--show" option to display all of the cracked passwords reliably
# run/john -show passwd.1
francois:francois:500:500::/home/francois:/bin/bash
firmin:nimrif:501:501::/home/firmin:/bin/bash
albert:bertal:502:502::/home/albert:/bin/bash
tux:NO PASSWORD:506:506::/home/tux:/bin/bash

4 password hashes cracked, 4 left
# more run/john.pot
$6$/x4uzg2W$4DWnVH3odWLD3Ot4Rp.ycUE2MHG.OGPoBc12CgsFtiI78ndj85mYbQRBlMUeX/lun5Xd
Nq6qae3odgIEBtLdP0:francois
...
```

5. Attaque au dictionnaire

a) Télécharger un dictionnaire.

```
# wget http://ftp.sunet.se/pub/security/tools/net/Openwall/wordlists/all.gz
# zcat all.gz  | grep -v '^#' > openwall.dico
# head openwall.dico
12345
abc123
password
passwd
123456
newpass
notused
Hockey
internet
asshole
```

b) On se crée son propre dictionnaire.

```
# vi plantes
belladone
cerisier
```

Remarque
On peut utiliser aussi le dictionnaire fourni (run/password.lst) ou celui de Cracklib (cf. tâche suivante).

c) L'attaque.

```
# run/john -w:plantes passwd.1
Loaded 7 password hashes with 7 different salts (generic crypt(3) [?/32])
Remaining 3 password hashes with 3 different salts
```

```
belladone        (gisele)
guesses: 1  time: 0:00:00:00 100%  c/s: 24.00  trying: belladone - cerisier
Use the "--show" option to display all of the cracked passwords reliably


# run/john -w:openwall.dico passwd.1
Loaded 11 password hashes with 11 different salts (generic crypt(3) [?/32])
Remaining 7 password hashes with 7 different salts
secret           (root)
secret           (bob)
america7         (beatrice)
guesses: 3  time: 0:00:01:03 0%  c/s: 48.10  trying: wilson - Fluffy
```

Remarque : on arrête, car il faudrait plusieurs jours de calcul...

6. Attaque exhaustive (temps de recherche infini !).

```
# run/john -i passwd.1
Loaded 3 password hashes with 3 different salts (generic crypt(3) [?/32])
Remaining 1 password hash
```

Remarque
Si l'on appuie sur la barre d'espace, on visualise l'état d'avancement.

```
guesses: 0  time: 0:00:02:21  c/s: 4041  trying: 49390222
shileyon         (toor)
```

Remarque
Si l'on appuie sur Ctrl-C, on interrompt le programme. L'option `-restore` permet de reprendre la recherche.

```
Ctrl-C
guesses: 1  time: 0:00:04:56  c/s: 4059  trying: bag1r
Session aborted
# run/john -restore
Loaded 3 password hashes with 3 different salts (FreeBSD MD5 [32/32])
guesses: 1  time: 0:00:05:00  c/s: 4049  trying: okp
Ctrl-C
```

7. Attaque complète.

Utilise toutes les techniques : simple, au dictionnaire (avec password.lst) et exhaustive.

```
# run/john  passwd.1
Loaded 3 password hashes with 3 different salts (generic crypt(3) [?/32])
Ctrl-C
```

8. Mesurer la rapidité de John.

```
# run/john -test
Benchmarking: Traditional DES [24/32 4K]... DONE
Many salts:     254824 c/s real, 705343 c/s virtual
Only one salt:  244582 c/s real, 671929 c/s virtual

Benchmarking: BSDI DES (x725) [24/32 4K]... DONE
Many salts:     9091 c/s real, 25113 c/s virtual
Only one salt:  8923 c/s real, 24649 c/s virtual

Benchmarking: FreeBSD MD5 [32/32]... DONE
Raw:    5852 c/s real, 16165 c/s virtual

Benchmarking: OpenBSD Blowfish (x32) [32/32]... DONE
```

```
Raw:    351 c/s real, 976 c/s virtual

Benchmarking: Kerberos AFS DES [24/32 4K]... DONE
Short:  238626 c/s real, 656879 c/s virtual
Long:   640819 c/s real, 1770K c/s virtual

Benchmarking: LM DES [32/32 BS]... DONE
Raw:    3993K c/s real, 11114K c/s virtual

Benchmarking: generic crypt(3) [?/32]... DONE
Many salts:     91296 c/s real, 253600 c/s virtual
Only one salt:  92665 c/s real, 256495 c/s virtual

Benchmarking: dummy [N/A]... DONE
Raw:    39733K c/s real, 110149K c/s virtual
# cd
```

Tâche 3 : Le module PAM pam_unix gérant la stratégie password

1. On crée un compte, on lui affecte un mot de passe.

```
[root@linux01 ~]# useradd pierre
[root@linux01 ~]# echo pierre | passwd --stdin pierre
```

2. On essaye sous le compte de l'utilisateur de changer plusieurs fois son mot de passe.

On utilise ab19bc90 comme 1er mot de passe. Ensuite on utilise al33li40 comme 2e mot de passe. Enfin on utilise ab19bc90 de nouveau.

```
[root@linux01 ~]# su - pierre
[pierre@linux01 ~]$ id pierre
uid=513(pierre) gid=514(pierre) groups=514(pierre)
context=root:system_r:unconfined_t:SystemLow-SystemHigh
[pierre@linux01 ~]$ passwd
Changing password for user pierre.
Changing password for pierre
(current) UNIX password: pierre
New UNIX password: ab19bc90
Retype new UNIX password: ab19bc90
passwd: all authentication tokens updated successfully.
[pierre@linux01 ~]$ passwd
Changing password for user pierre.
Changing password for pierre
(current) UNIX password: ab19bc90
New UNIX password: al33li40
Retype new UNIX password: al33li40
passwd: all authentication tokens updated successfully.
[pierre@linux01 ~]$ passwd
Changing password for user pierre.
Changing password for pierre
(current) UNIX password: al33li40
New UNIX password: ab19bc90
Retype new UNIX password: ab19bc90
passwd: all authentication tokens updated successfully.
[pierre@linux01 ~]$ exit
```

3. L'administrateur configure la mémorisation des cinq derniers mots de passe.

```
[root@linux01 ~]# cp /etc/pam.d/system-auth /etc/pam.d/system-auth.000
[root@linux01 ~]# vi /etc/pam.d/system-auth
...
password    requisite     pam_cracklib.so try_first_pass retry=3
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass
use_authtok remember=5
password    required      pam_deny.so
...
```

4. L'utilisateur essaye de nouveau de changer plusieurs fois son mot de passe.

On utilise al33li40 comme 1er mot de passe. Ensuite on utilise az49ux69 comme 2e mot de passe. Enfin on utilise al33li40 de nouveau.

```
[root@linux01 ~]# su - pierre
[pierre@linux01 ~]$ passwd
Changing password for user pierre.
Changing password for pierre
(current) UNIX password: ab19bc90
New UNIX password: al33li40
Retype new UNIX password: al33li40
passwd: all authentication tokens updated successfully.
[pierre@linux01 ~]$ passwd
Changing password for user pierre.
Changing password for pierre
(current) UNIX password: al33li40
New UNIX password: az49ux69
Retype new UNIX password: az49ux69
passwd: all authentication tokens updated successfully.
[pierre@linux01 ~]$ passwd
Changing password for user pierre.
Changing password for pierre
(current) UNIX password: az49ux69
New UNIX password: al33li40
Password has been already used. Choose another.
passwd: Authentication token manipulation error

[pierre@linux01 ~]$ exit
```

5. L'administrateur visualise l'historique des mots de passe (crypté).

```
[root@linux01 ~]# more /etc/security/opasswd
pierre:501:2:$1$vHfoCp4G$SpOem7JlK/YzqMaWnqLFa1,$1$NVgK8uWf$JirJ5yLV9m0gX.YvdefT
61
```

6. On remet la configuration d'origine.

```
[root@linux01 ~]# cp /etc/pam.d/system-auth.000 /etc/pam.d/system-auth
```

Tâche 4 : Le module PAM pam_cracklib

1. Vérifier si Cracklib et le module PAM associé sont installés.

```
[root@linux01 ~]# rpm -q cracklib cracklib-dicts
cracklib-2.8.13-6.1.i686
cracklib-dicts-2.8.13-6.1.i686
[root@linux01 ~]# ls -l /lib/security/pam_cracklib.so
rwxr-xr-x. 1 root root 10160 Jan 12  2010 /lib/security/pam_cracklib.so
[root@linux01 ~]# grep pam_cracklib /etc/pam.d/system-auth
```

```
password    requisite     pam_cracklib.so try_first_pass retry=3
```

2. Visualiser le dictionnaire de Cracklib.

```
[root@linux01 ~]# cd /usr/share/cracklib
[root@linux01 cracklib]# cracklib-unpacker pw_dict |head -15
0
00
000
0000
00000
000000
0000000
00000000
00brucellosis
00faa
00kiribati
00mag
00murree
00whitebait
02an
[root@linux01 cracklib]# cracklib-unpacker pw_dict |grep anne | head
10jeannette
11anneliese
13annemarie
16tanner
18hannelore
1annecke
1anneka
1anneke
1annel
1annelia
[root@linux01 cracklib]# cd
```

3. On essaye de modifier le mot de passe d'un utilisateur.

On essaye les mots de passe suivants : oh, tseutseu, belladonne, WWII1945, 1945wwii, pierre33, roberttrebor, wwii1945, abc12xyz33. Enfin on essaye le mot lukeskywalker, mais on ne le valide pas.

```
[root@linux01 ~]# useradd pierre
useradd: user pierre exists
[root@linux01 ~]# echo wwii1945 |passwd --stdin pierre
Changing password for user pierre.
passwd: all authentication tokens updated successfully.
[root@linux01 ~]# su - pierre
[pierre@linux01 ~]$ passwd
Changing password for user pierre.
Changing password for pierre
(current) UNIX password: wwii1945
New UNIX password: oh
BAD PASSWORD: it is WAY too short
New UNIX password: tseutseu
BAD PASSWORD: it does not contain enough DIFFERENT characters
New UNIX password: belladone
BAD PASSWORD: it is based on a dictionary word
passwd: Authentication token manipulation error
[pierre@linux01 ~]$ passwd
```

```
Changing password for user pierre.
Changing password for pierre
(current) UNIX password: wwii1945
New UNIX password: WWII1945
BAD PASSWORD: case changes only
New UNIX password: 1945wwii
BAD PASSWORD: is rotated
New UNIX password: pierre33
BAD PASSWORD: it is based on your username
passwd: Authentication token manipulation error
[pierre@linux01 ~]$ passwd
Changing password for user pierre.
Changing password for pierre
(current) UNIX password: wwii1945
New UNIX password: roberttrebor
BAD PASSWORD: is a palindrome
New UNIX password: wwii1945
Password unchanged
New UNIX password: abc12xyz33
BAD PASSWORD: it is too simplistic/systematic
passwd: Authentication token manipulation error
[pierre@linux01 ~]$ passwd
Changing password for user pierre.
Changing password for pierre
(current) UNIX password: wwii1945
New UNIX password: darksidious
Retype new UNIX password:
Sorry, passwords do not match.
New UNIX password:
[pierre@linux01 ~]$ exit
```

4. On modifie le dictionnaire de cracklib.

```
[root@linux01 ~]# vi starwars
darkvador
darksidious
lukeskywalker
leiaorgana
comtedooku
[root@linux01 ~]# cracklib-unpacker /usr/share/cracklib/pw_dict > dico
[root@linux01 ~]# cat dico starwars |sort |uniq > newdico
[root@linux01 ~]# mkdict newdico |packer pw_dict
1692953 1692953
```

5. On teste avec le nouveau dictionnaire.

```
[root@linux01 ~]# su - pierre
[pierre@linux01 ~]$ passwd
Changing password for user pierre.
Changing password for pierre
(current) UNIX password: wwii1945
New UNIX password: darksidious
Ctrl-C
[pierre@linux01 ~]$ exit
```

Tâche 5 :
La sécurité de connexion, le password aging

1. On crée un compte et on lui affecte un mot de passe.

```
[root@linux01 ~]# useradd guest
useradd: user guest exists
[root@linux01 ~]# echo wwii1945 |passwd --stdin guest
Changing password for user guest.
passwd: all authentication tokens updated successfully.
```

2. Visualiser le mot de passe de l'utilisateur.

a) En étant connecté sous un compte utilisateur.

```
[root@linux01 ~]# su - guest
[guest@linux01 ~]$ grep guest /etc/shadow
grep: /etc/shadow: Permission denied
[guest@linux01 ~]$ exit
```

b) En étant connecté sous le compte de l'administrateur. La graine (salt) est entre $.

```
[root@linux01 ~]# grep guest /etc/shadow
guest:$6$bNRERlmh$jzJ3jRM8y8ziipQ.oHvPOkH3DAiT4dKVg2YOUbUDDbKRU6x51TE0N02YHtu
bSKEu3LSIfr9kBQ8Rk8YfmJ.:15218:0:99999:7:::
```

c) L'administrateur essaye de deviner le mot de passe.

```
[root@linux01 ~]# # perl -e 'print crypt("wwii1945","\$6\$bNRERlmh\$") . "\n"'
$6$bNRERlmh$jzJ3jRM8y8ziipQ.oHvPOkH3DAiT4dKVg2YOUbUDDbKRU6x51TE0N02YHtusrfbSKEu3
LSIfr9kBQ8Rk8YfmJ.
```

3. Changer la période de validité du mot de passe (on la limite à 30 jours).

```
[root@linux01 ~]# chage -M 30 guest
[root@linux01 ~]# chage -l guest
Last password change                                    : Sep 01, 2011
Password expires                                        : Oct 01, 2011
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 30
Number of days of warning before password expires       : 7
```

4. Limiter la durée de vie d'un compte puis éliminer cette limite.

```
[root@linux01 ~]# chage -E 2011-12-31 guest
[root@linux01 ~]# chage -l guest
Last password change                                    : Sep 01, 2011
Password expires                                        : Oct 01, 2011
Password inactive                                       : never
Account expires                                         : Dec 31, 2011
Minimum number of days between password change          : 0
Maximum number of days between password change          : 30
Number of days of warning before password expires       : 7
[root@linux01 ~]# chage -E -1 guest
[root@linux01 ~]# chage -l guest |grep 'Account'
Account expires                                         : never
```

5. Empêcher l'utilisateur de modifier son mot de passe.

Limiter la durée de vie du mot de passe à trois mois et empêcher ensuite l'utilisateur de le modifier. Supprimer ces limitations.

```
[root@linux01 ~]# chage -m 90 guest
```

```
[root@linux01 ~]# chage -M 90 guest
[root@linux01 ~]# chage -l guest
Last password change                                    : Sep 01, 2011
Password expires                                        : Nov 30, 2011
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 90
Maximum number of days between password change          : 90
Number of days of warning before password expires       : 7
[root@linux01 ~]# chage -M 99999 guest
[root@linux01 ~]# chage -m 0 guest
[root@linux01 ~]# chage -l guest
Last password change                                    : Sep 01, 2011
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
```

6. Changer le mot de passe d'un utilisateur.

L'utilisateur guest a perdu son mot de passe, il demande à l'administrateur de le lui réinitialiser.

```
[root@linux01 ~]# passwd guest
Changing password for user guest.
New UNIX password: wwii1945
Retype new UNIX password: wwii1945
passwd: all authentication tokens updated successfully.
```

7. Verrouiller un compte, afficher son état, déverrouiller le compte.

```
[root@linux01 ~]# passwd -l guest
Locking password for user guest.
passwd: Success
[root@linux01 ~]# passwd -S guest
guest LK 2011-09-01 0 99999 7 -1 (Password locked.)
[root@linux01 ~]# su - -s /bin/sh bin
-sh-3.1$ su - guest
Password: wwii1945
su: incorrect password
-sh-3.1$ exit
logout
[root@linux01 ~]# passwd -u guest
Unlocking password for user guest.
passwd: Success.
[root@linux01 ~]# passwd -S guest
guest PS 2011-09-01 0 99999 7 -1 (Password set, SHA512 crypt.)
[root@linux01 ~]# su - -s /bin/sh bin
-sh-3.1$ su - guest
Password: wwii1945
Warning: your password will expire in 0 days
[guest@linux01 ~]$ exit
-sh-3.1$ exit
[root@linux01 ~]# chage -l guest | grep 'Password.*expires'
Password expires                                        : never
```

8. Supprimer le mot de passe de l'utilisateur. Essayer de se connecter. Remettre un mot de passe.

```
[root@linux01 ~]# passwd -d guest
Removing password for user guest.
passwd: Success
[root@linux01 ~]# su - -s /bin/sh bin
-sh-3.1$ su - guest
[guest@linux01 ~]$ exit
-sh-3.1$ exit
logout
[root@linux01 ~]# echo wwii1945 |passwd --stdin guest
```

Remarque
On a pu se connecter sans mot de passe avec `su`, par contre, si on essaye de se connecter avec `login` ou `ssh`, cela ne fonctionne pas.

9. Visualiser les dernières connexions qui ont réussi/échoué.

```
[root@linux01 ~]# last |head -5
root     pts/0        192.168.0.254    Thu Sep  1 19:29   still logged in
alice    pts/1        localhost        Thu Sep  1 11:58 - 12:06  (00:08)
bob      pts/1        localhost        Thu Sep  1 11:53 - 11:58  (00:04)
alice    pts/1        localhost        Thu Sep  1 11:35 - 11:53  (00:17)
root     pts/0        192.168.0.254    Thu Sep  1 10:21 - 16:21  (05:59)
[root@linux01 ~]# lastb
alice    tty2                          Thu Sep  1 20:25 - 20:25  (00:00)
bob      tty2                          Thu Sep  1 20:25 - 20:25  (00:00)

btmp begins Thu Sep  1 20:25:04 2011
```

10. Visualiser, pour chaque compte, la dernière connexion.

```
[root@linux01 ~]# lastlog |more
Username         Port     From             Latest
root             pts/0    192.168.0.254    Thu Sep  1 19:29:01 +0200 2011
bin                                        **Never logged in**
daemon                                     **Never logged in**
adm                                        **Never logged in**
...
gisele                                     **Never logged in**
tux                                        **Never logged in**
toor             pts/0    192.168.0.254    Thu Sep  1 19:29:01 +0200 2011
pierre                                     **Never logged in**
```

11. Visualiser les valeurs par défaut du password aging.

```
[root@linux01 ~]# more /etc/login.defs
# *REQUIRED*
#   Directory where mailboxes reside, _or_ name of file, relative to the
#   home directory.  If you _do_ define both, MAIL_DIR takes precedence.
#   QMAIL_DIR is for Qmail
#
#QMAIL_DIR      Maildir
MAIL_DIR        /var/spool/mail
#MAIL_FILE      .mail

# Password aging controls:
#
#       PASS_MAX_DAYS   Maximum number of days a password may be used.
```

```
#       PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#       PASS_MIN_LEN    Minimum acceptable password length.
#       PASS_WARN_AGE   Number of days warning given before a password expires.
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7

#
# Min/max values for automatic uid selection in useradd
#
UID_MIN                  500
UID_MAX                60000

#
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                  500
GID_MAX                60000

#
# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#
#USERDEL_CMD    /usr/sbin/userdel_local

#
# If useradd should create home directories for users by default
# On RH systems, we do. This option is overridden with the -m flag on
# useradd command line.
#
CREATE_HOME     yes

# The permission mask is initialized to this value. If not specified,
# the permission mask will be initialized to 022.
UMASK           077

# This enables userdel to remove user groups if no members exist.
#
USERGROUPS_ENAB yes

# Use SHA512 to encrypt password.
ENCRYPT_METHOD SHA512
```

12. Générer des mots de passe aléatoires.

```
[root@linux01 ~]# head -c 6 /dev/urandom |base64
HjQ6cuPh
[root@linux01 ~]# head -c 6 /dev/urandom |base64
/Hu2pQoi
```

13. Automatiser les changements de mots de passe.

L'administrateur peut décider que les utilisateurs ne doivent pas choisir eux-mêmes leur mot de passe. Dans ce cas il peut automatiser leur changement.

```
[root@linux01 ~]# useradd alice
useradd: user alice exists
[root@linux01 ~]# useradd bob
useradd: user bob exists
[root@linux01 ~]# vi list_users
alice
bob
[root@linux01 ~]# vi change_pass.sh
#!/bin/sh
rm pass.maj
while read user
do
 password=$(head -c 6 /dev/urandom |base64)
 echo $user:$password >> pass.maj
 echo "voici votre nouveau password: $password" | mail -s pass $user
done < list_users
at now  < change_pass_suite.sh
[root@linux01 ~]# vi change_pass_suite.sh
#!/bin/sh
IFS=:
while read user pass
do
 echo $pass | passwd --stdin $user
done < /root/pass.maj
# rm /root/pass.maj
[root@linux01 ~]# sh change_pass.sh
job 13 at 2008-01-02 22:27
[root@linux01 ~]# cat pass.maj
alice:zJWEXihy
bob:l0AcW9zY
[root@linux01 ~]# su - alice
[alice@linux01 ~]$ mail
Heirloom Mail version 12.4 7/29/08.  Type ? for help.
"/var/spool/mail/alice": 1 message 1 new
>N  1 root                  Thu Sep  1 20:34  18/624   "pass"
& 1
Message  1:
From root@linux01.localdomain  Thu Sep  1 20:34:01 2011
Return-Path: <root@linux01.localdomain>
X-Original-To: alice
Delivered-To: alice@linux01.localdomain
Date: Thu, 01 Sep 2011 20:34:00 +0200
To: alice@linux01.localdomain
Subject: pass
User-Agent: Heirloom mailx 12.4 7/29/08
Content-Type: text/plain; charset=us-ascii
From: root@linux01.localdomain (root)
Status: R

voici votre nouveau password: zJWEXihy
```

```
& q
Held 1 message in /var/spool/mail/alice
[alice@linux01 ~]$ exit
[root@linux01 ~]# su - -s /bin/sh bin
-sh-3.1$ su - alice
Password: zJWEXihy
[alice@linux01 ~]$ exit
-sh-3.1$ exit
```

Remarques

1) Pour effectuer les tests, les mots de passe sont modifiés immédiatement. Dans la réalité, il faudrait différer cette opération. Dans le script, la commande `at` pourrait s'écrire comme suit :
```
at now + 5 days <<EOF
```

2) Pour améliorer la sécurité, les mots de passe des utilisateurs pourraient être générés par un programme comme pwgen et être cryptés avec leur clé publique.

3) Le fichier pass.maj qui contient les mots de passe doit soit être détruit après usage soit crypté.

4) Sur les autres distributions, le script aurait pu être simplifié avec la commande chpasswd qui fonctionne correctement.

Tâche 6 : Les droits d'endossement

1. On crée un mini-shell.

```
[root@linux01 ~]# vi msh.c
# include <stdio.h>
# include <unistd.h>
main() {
        int status;
        char cmd[256],arg[256];
        for(;;) {
                printf("===> Commande ? ");
                gets(cmd);
                printf("---> Argument ? ");
                gets(arg);
                if ( fork() == 0 ) {
                        execlp( cmd, cmd, arg, 0 );
                } else {
                        wait(&status);
                }
        }
}
[root@linux01 ~]# cc -o msh msh.c
/tmp/ccTcYvve.o: In function `main':
msh.c:(.text+0x2c): warning: the `gets' function is dangerous and should not be
used.
```

2. On donne le droit SUID à ce mini-shell et on le rend accessible aux utilisateurs.

```
[root@linux01 ~]# cp msh /tmp
[root@linux01 ~]# chmod 4555 /tmp/msh
[root@linux01 ~]# ls -l /tmp/msh
-r-sr-xr-x. 1 root root 5085 Sep  1 20:40 /tmp/ms
```

3. Un utilisateur active le mini-shell : il possède les prérogatives de root grâce à lui.

```
[root@linux01 ~]# su - guest
[guest@linux01 ~]$ id
uid=500(guest) gid=500(guest) groups=100(users),500(guest)
context=root:system_r:unconfined_t:SystemLow-SystemHigh
[guest@linux01 ~]$ /tmp/msh
===> Commande ? head
---> Argument ? /etc/shadow
root:$6$6hLdwCPEiesU6f03$MpeuT7BC2mmOYQ6wf/xLx6YB4uR1W.8GpIv1.blD/5lY97pK4atw/zE
ameKkTm3vUAXZx8oSwdPGNPfibCgun0:15189:0:99999:7:::
bin:*:14621:0:99999:7:::
daemon:*:14621:0:99999:7:::
...
===> Commande ? Ctrl-C
[guest@linux01 ~]$ exit
```

4. Rechercher les exécutables possédant des droits d'endossement.

```
[root@linux01 ~]# # find / -type f -perm +111 \( -perm -2000 -o -perm -4000 \) -
exec ls -l {} \; | more
-rwsr-xr-x. 1 root root 30700 Jan 12  2010 /sbin/unix_chkpwd
-rwxr-sr-x. 1 root root 6152 Feb 15  2010 /sbin/netreport
-rwsr-xr-x. 1 root root 7668 Jan 12  2010 /sbin/pam_timestamp_check
...
-rwsr-x---. 1 root dbus 49236 Jan 13  2010 /lib/dbus-1/dbus-daemon-launch-helper
-rwxr-sr-x. 1 root postdrop 174856 Jan  8  2010 /usr/sbin/postdrop
...
```

Tâche 7 : Les options de montage des FS

1. Créer un FS dans un fichier.

```
[root@linux01 ~]# dd if=/dev/zero of=/tmp/GROS_FIC bs=1k count=1024
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB) copied, 0.00983034 seconds, 107 MB/s
[root@linux01 ~]# mkfs -t ext2 -q /tmp/GROS_FIC
/tmp/GROS_FIC is not a block special device.
Proceed anyway? (y,n) y
```

2. Monter le FS avec les options nodev, nosuid, noexec.

```
[root@linux01 ~]# mount -o loop -o nodev,nosuid,noexec /tmp/GROS_FIC /mnt
[root@linux01 ~]# # mount |grep /mnt
/dev/loop0 on /mnt type ext2 (rw,noexec,nosuid,nodev)
```

3. Créer dans le FS un exécutable, un périphérique et un programme SUID.

```
[root@linux01 ~]# cd /mnt
[root@linux01 mnt]# cp /bin/more affiche
[root@linux01 mnt]# ls -l /dev/sda
brw-rw----. 1 root disk 8, 0 Sep  1 10:19 /dev/sda
[root@linux01 mnt]# mknod disque_a b 8 0
[root@linux01 mnt]# cp -a /usr/bin/passwd change_pasword
[root@linux01 mnt]# ls -l
total 75
-rwxr-xr-x. 1 root root 35248 Sep  1 20:58 affiche
-rwsr-xr-x. 1 root root 22632 Jan 12  2010 change_pasword
brw-r--r--. 1 root root  8, 0 Sep  1 20:59 disque_a
```

```
drwx------. 2 root root 12288 Sep  1 20:54 lost+found
```

4. Essayer d'utiliser les fichiers créés précédemment.

Essayer d'exécuter l'application ou d'accéder au périphérique.

```
[root@linux01 mnt]# id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@linux01 mnt]# ./affiche /etc/shadow
-bash: ./affiche: Permission denied
[root@linux01 mnt]# dd if=disque_a of=/dev/null
dd: opening `disque_a': Permission denied
```

5. Remonter le FS en acceptant l'exécution des applications (pour tester les droits SUID).

```
[root@linux01 mnt]# mount -o remount,exec /mnt
[root@linux01 mnt]# mount |grep /mnt
/dev/loop0 on /mnt type ext2 (rw,nosuid,nodev)
[root@linux01 mnt]# ./affiche /etc/issue
Red Hat Enterprise Linux release 6.0 Beta (Santiago)
Kernel \r on an \m

[root@linux01 mnt]# su guest
[guest@linux01 mnt]$ ./change_password
Changing password for user guest.
Changing password for guest
(current) UNIX password: wwii1945
New UNIX password: azertyuiop
Retype new UNIX password: azertyuiop
passwd: Authentication token  manipulation error
[guest@linux01 mnt]$ exit
[root@linux01 mnt]# cd
[root@linux01 ~]# umount /mnt
[root@linux01 ~]# rm /tmp/GROS_FIC
```

Tâche 8 : Les capabilities

1. Vérifier si une application peut utiliser les capabilities.

Remarque
Si vsftpd et bind ne sont pas installées, installez-les.

```
# rpm -q vsftpd
vsftpd-2.2.2-1.el6.i686
# rpm -q bind
bind-9.7.0-1.el6.i686
# whereis vsftpd
vsftpd: /usr/sbin/vsftpd /etc/vsftpd /usr/share/man/man8/vsftpd.8.gz
# ldd /usr/sbin/vsftpd | grep libcap
        libcap.so.2 => /lib/libcap.so.2 (0x00b9a000)
# ldd /usr/sbin/named | grep libcap
        libcap.so.2 => /lib/libcap.so.2 (0x0014f000)
```

2. Visualiser les capabilities d'un processus.

```
# service vsftpd restart
# cat /proc/$(pgrep vsftpd)/status |grep '^Cap'
CapInh: 0000000000000000
```

```
CapPrm: ffffffffffffffff
CapEff: ffffffffffffffff
CapBnd: ffffffffffffffff
# getpcaps $(pgrep vsftpd)
Capabilities for `12927': =ep
```

3. Créer une application qui change les capabilities.

```
# rpm -q libcap-devel
libcap-devel-2.16-5.1.el6.i686
# vi chproprio.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/prctl.h>
#include <sys/capability.h>

int main(int argc, char **argv)
{
        cap_t caps = cap_from_text(argv[1]);

        if (caps) {
                if ( cap_set_proc(caps) )
                        perror("capabilities");
                else {
                        cap_t caps2 = cap_get_proc();
                        char * p = cap_to_text( caps2, NULL);
                        if ( p )
                                printf("caps: %s\n", p);
                }
                cap_free(caps);
        }
        chown(argv[2],0,0);
}
# cc -o chproprio chproprio.c –lcap
```

4. Utiliser l'application précédente pour changer le propriétaire d'un fichier.

a) On crée le fichier sous le compte guest.

```
# rm /tmp/titi
[root@linux01 ~]# su - guest -c "cal > /tmp/titi"
[root@linux01 ~]# ls -l /tmp/titi
-rw-rw-r--. 1 guest guest 145 Sep  1 21:11 /tmp/titi
```

b) On affiche les capabilities du shell courant.

```
# getpcaps $$
Capabilities for `12343': =ep
```

c) On essaye de donner le fichier à root en supprimant toutes les capabilities. L'opération échoue.

```
# ./chproprio all= /tmp/titi
caps: =
# ls -l /tmp/titi
-rw-rw-r--. 1 guest guest 145 Sep  1 21:11 /tmp/titi
```

d) On refait l'opération, mais on garde la capability « CAP_CHOWN ». L'opération réussit : le propriétaire du fichier a changé (il appartient désormais à root).

```
# ./chproprio cap_chown=eip /tmp/titi
caps: = cap_chown+eip
# ls -l /tmp/titi
-rw-rw-r--. 1 root root 145 Sep  1 21:11 /tmp/titi
```

Remarque
La chaîne eip signifie Effectif, Hérité (inheritate), Permissif.

5. Visualiser les appels système d'une application gérant les capabilities.

```
# strace ./chproprio cap_chown=eip /tmp/titi 2> /tmp/err
caps: = cap_chown+eip
# grep cap /tmp/err
execve("./chproprio", ["./chproprio", "cap_chown=eip", "/tmp/titi"], [/* 25 vars
*/]) = 0
open("/lib/libcap.so.2", O_RDONLY)      = 3
capget(0x20080522, 0, NULL)             = -1 EFAULT (Bad address)
capset(0x20080522, 0, {CAP_CHOWN, CAP_CHOWN, CAP_CHOWN}) = 0
capget(0x20080522, 0, NULL)             = -1 EFAULT (Bad address)
capget(0x20080522, 0, {CAP_CHOWN, CAP_CHOWN, CAP_CHOWN}) = 0
write(1, "caps: = cap_chown+eip\n", 22) = 22
```

6. Modifier les capabilities d'un exécutable.

L'application ping manipule des sockets raw qui impliquent la capability CAP_NET_RAW. Par défaut, n'importe quel utilisateur peut utiliser la commande ping, pourquoi ? En fait, via les droits SUID, un utilisateur exécute la commande avec les droits de root. Si l'application comporte un bug, éventuellement l'utilisation de la commande permettrait de faire n'importe quoi sur le système. En utilisant le principe des capabilities, on restreint les dégâts possibles.

a) Supprimer les droits d'endossement de ping.

```
# ls -l /bin/ping
-rwsr-xr-x. 1 root root 41976 Dec  7  2009 /bin/pin
# chmod u-s /bin/ping
# ls -l /bin/ping
-rwxr-xr-x. 1 root root 41976 Dec  7  2009 /bin/ping
```

b) Vérifier que seul root peut maintenant utiliser ping.

```
# su guest ping localhost
bash: ping: Permission denied
# ping -c1 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.676 ms
...
```

c) Modifier les capabilities de la commande ping : mettre la capability CAP_NET_RAW en mode « effectif » et « permissif » (elle sera donc effective).

```
# getcap /bin/ping
# setcap CAP_NET_RAW=ep /bin/ping
# getcap /bin/ping
/bin/ping = cap_net_raw+ep
```

d) Tester en se connectant en tant que simple utilisateur.

```
# ssh -l guest localhost
guest@localhost's password: wwii1945
Last login: Mon Sep  5 11:55:32 2011 from 192.168.0.1
[guest@linux01 ~]$ getpcaps $$
Capabilities for `25221': =
[guest@linux01 ~]$ ping -c1 localhost
```

```
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.745 ms
...
[guest@linux01 ~]$ exit
```

7. Utiliser le module PAM pam_cap.

Maintenant, on veut restreindre l'utilisation de la commande `ping` à quelques utilisateurs seulement. Pour ce faire, on utilise le module pam_cap.so

a) Ajouter le module pam_cap.so à la configuration de la connexion (login) et faire en sorte que guest ait la capability CAP_NET_RAW.

```
# vi /etc/pam.d/login
auth       required     pam_cap.so
...
# vi /etc/security/capability.conf
cap_net_raw  guest
```

b) Supprimer les capabilities et ensuite positionner CAP_NET_RAW, mais en mode « effectif » et « hérité ».

```
# setcap -r /bin/ping
# getcap /bin/ping
# setcap cap_net_raw=ei /bin/ping
# getcap /bin/ping
/bin/ping = cap_net_raw+ei
```

c) Tester en se connectant en tant que simple utilisateur non autorisé.

```
# useradd -m user1
# echo user1 |passwd --stdin user1
```

On se connecte ensuite sous le compte user1 (via la console 2 : CTRL-ALT-F1).

```
[user1@linux01 ~]$ getpcaps $$
Capabilities for `25318': =
[user1@linux01 ~]$ ping -c1 localhost
ping: icmp open socket: Operation not permitted
[user1@linux01 ~]$ exit
```

d) Tester en se connectant sous un compte autorisé : guest.

```
[guest@linux01 ~]$ getpcaps $$
Capabilities for `25081': = cap_net_raw+i
[guest@linux01 ~]$ ping -c1 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.745 ms
...
[guest@linux01 ~]$ exit
```

Tâche 9 : La sécurité pour les utilisateurs

1. Un mauvais PATH.

Remarque
Dans l'exemple, on montre l'inconvénient d'avoir un mauvais PATH dans le cas où on est l'administrateur. Mais le type d'exploit présenté est valable pour tout utilisateur.

a) Le pirate crée l'exploit.

```
[root@linux01 ~]# su - guest
[guest@linux01 ~]$ vi /tmp/ls
#!/bin/sh
```

```
mv ls ~guest
/bin/ls $*
cp /etc/shadow ~guest
chmod 444 ~guest/shadow
[guest@linux01 ~]$ chmod 555 /tmp/ls
[guest@linux01 ~]$ exit
```

b) L'administrateur possède un mauvais PATH, il tombe dans le piège.

```
[root@linux01 ~]# OLDPATH=$PATH
[root@linux01 ~]# PATH=.:$PATH
[root@linux01 ~]# cd /tmp
[root@linux01 tmp]# ls
alice_cle.asc  msg_chif.txt  msg.txt.asc  sbin.md5  titi
GROS_FIC       msg.txt       msh          tata
```

c) Le pirate connaît maintenant le mot de passe (crypté) de root.

```
[root@linux01 tmp]# su - guest
[guest@linux01 ~]$ grep root shadow
root:$6$6hLdwCPEiesU6f03$MpeuT7BC2mmOYQ6wf/xLx6YB4uR1W.8GpIv1.blD/5lY97pK4atw/zE
ameKkTm3vUAXZx8oSwdPGNPfibCgun0:15189:0:99999:7:::
[guest@linux01 ~]$ exit
[root@linux01 tmp]# PATH=$OLDPATH
[root@linux01 tmp]# echo $PATH
/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
```

2. Provoquer une déconnexion automatique en cas d'inactivité.

```
[root@linux01 tmp]# echo "TMOUT=120" >> ~guest/.bash_profile
[root@linux01 tmp]# ssh -l guest localhost
guest@localhost's password: wwii1945
[guest@linux01 ~]$ date '+%Hh%M'
23h47
[guest@linux01 ~]$ timed out waiting for input: auto-logout
Connection to localhost closed.
[root@linux01 tmp]# cd
```

Remarque
Au bout de trois minutes d'inactivité (une minute est toujours ajoutée au délai), la déconnexion a lieu.

3. Verrouiller un terminal.

```
[root@linux01 ~]# rpm -q vlock
vlock-1.3-30.el6.i686
[root@linux01 ~]# ssh -l guest localhost
guest@localhost's password: wwii1945
Last login: Thu Sep  1 22:53:39 2011 from localhost
[guest@linux01 ~]$ vlock
 *** This tty is not a VC (virtual console). ***
 *** It may not be securely locked. ***

This TTY is now locked.
Please enter the password to unlock.
guest's Password: wwii1945
```

4. Vérifier les droits de vos fichiers.

Votre fichier qui initialise de votre session doit avoir les droits 400 ou 600. Votre répertoire de connexion doit avoir les droits 700 et votre UMASK doit avoir la valeur 77.

```
[guest@linux01 ~]$ ls -l ~/.bash_profile
```

```
-rw-r--r--. 1 guest guest 186 Sep  1 22:53 /home/guest/.bash_profile
[guest@linux01 ~]$ ls -ld ~
drwx------. 2 guest guest 4096 Sep  1 22:51 /home/guest
[guest@linux01 ~]$ umask
0002
[guest@linux01 ~]$ echo "umask 77" >> ~/.bash_profile
[guest@linux01 ~]$ . ~/.bash_profile
[guest@linux01 ~]$ umask
0077
[guest@linux01 ~]$ exit
```

Tâche 10 : Les ACL

1. On crée des comptes utilisateurs.

```
[root@linux01 ~]# userdel –r user1
[root@linux01 ~]# userdel –r user2
[root@linux01 ~]# userdel –r user3
[root@linux01 ~]# groupadd paire
[root@linux01 ~]# groupadd impaire
[root@linux01 ~]# useradd -g paire user2
[root@linux01 ~]# useradd -g paire user4
[root@linux01 ~]# useradd -g impaire user1
[root@linux01 ~]# useradd -g impaire user3
```

2. Créer et monter un FS, donner sa racine à user1.

```
[root@linux01 ~]# dd if=/dev/zero of=/tmp/GROS_FIC bs=1k count=1024
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB) copied, 0.00943996 seconds, 111 MB/s
[root@linux01 ~]# mkfs -t ext2 -q -F /tmp/GROS_FIC
[root@linux01 ~]# mount -o acl -o loop /tmp/GROS_FIC /mnt
[root@linux01 ~]# mount |grep loop
/dev/loop0 on /mnt type ext2 (rw,acl)
[root@linux01 ~]# chown user1 /mnt
```

3. Se connecter sous le compte user1 et créer une arborescence dans le FS.

```
[root@linux01 ~]# su - user1
[user1@linux01 ~]$ cd /mnt
[user1@linux01 mnt]$ cal > f1
[user1@linux01 mnt]$ date > f2
[user1@linux01 mnt]$ mkdir rep
[user1@linux01 mnt]$ ls > rep/fic
[user1@linux01 mnt]$ chmod go=- f1 f2 rep/fic
[user1@linux01 mnt]$ ls -l f1 f2 rep/fic
-rw-------. 1 user1 impaire 145 Sep  1 23:09 f1
-rw-------. 1 user1 impaire  30 Sep  1 23:09 f2
-rw-------. 1 user1 impaire  21 Sep  1 23:10 rep/fic
[user1@linux01 mnt]$
```

4. Positionner des ACL sur les fichiers et les visualiser.

```
[user1@linux01 mnt]$ setfacl -m user:user2:r f1
[user1@linux01 mnt]$ setfacl -m user:user2:rwx,group:paire:rw- f2
[user1@linux01 mnt]$ setfacl -m group:paire:r-- rep/fic
[user1@linux01 mnt]$ getfacl f1 f2 rep/fic
# file: f1
```

```
# owner: user1
# group: impaire
user::rw-
user:user2:r--
group::---
mask::r--
other::---

# file: f2
# owner: user1
# group: impaire
user::rw-
user:user2:rwx
group::---
group:paire:rw-
mask::rwx
other::---

# file: rep/fic
# owner: user1
# group: impaire
user::rw-
group::---
group:paire:r--
mask::r--
other::---

[user1@linux01 mnt]$ ls -l f1 f2 rep/fic
-rw-r-----+ 1 user1 impaire 145 Sep  1 23:09 f1
-rw-rwx---+ 1 user1 impaire  30 Sep  1 23:09 f2
-rw-r-----+ 1 user1 impaire  21 Sep  1 23:10 rep/fic
[user1@linux01 mnt]$ exit
```

5. Tester les accès.

a) À partir du compte user2.

```
[root@linux01 ~]# su - user2
[user2@linux01 ~]$ cd /mnt
[user2@linux01 mnt]$ id
uid=506(user2) gid=507(paire) groups=507(paire)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[user2@linux01 mnt]$ ls -l
total 18
-rw-r-----+ 1 user1 impaire   145 Sep  1 23:09 f1
-rw-rwx---+ 1 user1 impaire    30 Sep  1 23:09 f2
drwx------. 2 root  root    12288 Sep  1 23:08 lost+found
drwxr-xr-x. 2 user1 impaire  1024 Sep  1 23:10 rep
[user2@linux01 mnt]$ cat f1
    January 2008
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

```
[user2@linux01 mnt]$ cat rep/fic
f1
f2
lost+found
rep
[user2@linux01 mnt]$ exit
```

Remarques

1) Normalement, sans les ACL, user2 qui est du groupe pair n'aurait pas dû avoir le droit d'accéder à f1 ou rep/fic.

2) La commande `ls -l` ajoute un + après les droits dans le cas où le fichier possède des ACL.

b) À partir du compte user3

```
[root@linux01 ~]# su - user3
[user3@linux01 ~]$ cd /mnt
[user3@linux01 mnt]$ cat f1
cat: f1: Permission denied
[user3@linux01 mnt]$ exit
```

Remarque
Sans les ACL, user3 qui appartient au groupe impair aurait eu accès au fichier f1. Les ACL sont bien prioritaires.

6. Désactiver les ACL en modifiant le masque de f1. L'utilisateur user2 n'a plus l'accès au fichier.

```
[root@linux01 ~]# su - user1
[user1@linux01 ~]$ cd /mnt
[user1@linux01 mnt]$ setfacl -m mask::--- f1
[user1@linux01 mnt]$ getfacl f1
# file: f1
# owner: user1
# group: impaire
user::rw-
user:user2:r--                  #effective:---
group::---
mask::---
other::---

[user1@linux01 mnt]$ exit

[root@linux01 ~]# su - user2
[user2@linux01 ~]$ cd /mnt
[user2@linux01 mnt]$ tail -3 f1
tail: cannot open `f1' for reading: Permission denied
[user2@linux01 mnt]$ exit
```

7. Mémoriser, supprimer et restaurer les ACL.

a) Avec les commandes getfacl/setfacl.

```
[root@linux01 ~]# cd /mnt
[root@linux01 mnt]# getfacl -R . > /tmp/acls
[root@linux01 mnt]# setfacl -R -b .
[root@linux01 mnt]# getfacl f1
# file: f1
# owner: user1
```

```
# group: impaire
user::rw-
group::---
other::---

[root@linux01 mnt]# setfacl --restore=/tmp/acls
[root@linux01 mnt]# getfacl f1
# file: f1
# owner: user1
# group: impaire
user::rw-
user:user2:r--                  #effective:---
group::---
mask::---
other::---
```

b) Avec la commande tar.

```
# cd /
# tar -cz --acls -f /tmp/mnt.tar /mnt
tar: Removing leading `/' from member names
# rm -rf /mnt/*
# tar -xf /tmp/mnt.tar
# ls -l /mnt
total 8
-rw-------+ 1 user1 impaire  145 Sep  1 23:09 f1
...
# getfacl /mnt/f1
getfacl: Removing leading '/' from absolute path names
# file: mnt/f1
# owner: user1
# group: impaire
user::rw-
user:user2:r--                  #effective:---
group::---
mask::---
other::---
```

8. Créer des ACL par défaut pour une arborescence.

```
[root@linux01 /]# su - user1
[user1@linux01 ~]$ cd /mnt
[user1@linux01 mnt]$ setfacl -m default:user2:r rep
[user1@linux01 mnt]$ cal > rep/ficBis
[user1@linux01 mnt]$ getfacl rep/ficBis
# file: rep/ficBis
# owner: user1
# group: impaire
user::rw-
user:user2:r--
group::r-x                      #effective:r--
mask::r--
other::r--

[user1@linux01 mnt]$ exit
[root@linux01 /]# cd
[root@linux01 ~]# umount /mnt
```

```
[root@linux01 ~]# rm /tmp/GROS_FIC
rm: remove regular file `/tmp/GROS_FIC'? y
```

Tâche 11 : Les attributs Ext2

1. Créer un fichier, lister ses attributs.

```
# echo "Bonjour" > fichier
# lsattr fichier
-------------e- fichier
```

Remarque : sur un système de fichier Ext4, normalement les fichiers utilisent les extents pour leur allocation disque. L'attribut e l'indique.

2. Ajouter l'attribut « a » (fichier journal), tester les conséquences.

Un fichier ayant l'attribut a (archive) peut grossir, mais on ne peut en supprimer des données ni le supprimer.

```
# chattr +a fichier
# lsattr fichier
-----a-------e- fichier
# echo "Salut" >> fichier
# cal > fichier
-bash: fichier: Operation not permitted
# rm -f fichier
rm: cannot remove `fichier': Operation not permitted
```

3. Retirer l'attribut précédent, ajouter l'attribut i (fichier immuable).

L'attribut i (immuable) a pour conséquence que l'on ne peut ni modifier le fichier, ni le supprimer, ni le déplacer, ni créer des liens symboliques dessus.

```
# chattr -a fichier
# chattr +i fichier
# lsattr fichier
----i--------e- fichier
# echo "Bye" >> fichier
-bash: fichier: Permission denied
# cal > fichier
-bash: fichier: Permission denied
# rm -f fichier
rm: cannot remove `fichier': Operation not permitted
# ln fichier ficBis
ln: creating hard link `ficBis' => `fichier': Operation not permitted
# mv fichier /tmp
mv: cannot move `fichier' to `/tmp/fichier': Operation not permitted
```

4. Retirer l'attribut précédent, ajouter l'attribut A (noatime).

L'attribut A (no Access time) contrôle la mise à jour de la date de dernier accès. Si l'attribut est positionné, cette mise à jour est désactivée.

```
# chattr -i fichier
# chattr +A fichier
# lsattr fichier
-------A-----e- fichier
# ls -lu fichier
-rw-r--r-- 1 root root 14 Jul 21 19:28 fichier
# sleep 60; cat fichier
Bonjour
```

```
Salut
# ls -lu fichier
-rw-r--r-- 1 root root 14 Jul 21 19:28 fichier
# chattr -A fichier
# sleep 60; cat fichier
Bonjour
Salut
# ls -lu fichier
-rw-r--r-- 1 root root 14 Jul 21 19:43 fichier
```

5. Ajouter au fichier les attributs a, i, s, D, S et A au fichier. Les visualiser et les supprimer. Supprimer également le fichier.

```
# chattr +aisDSA fichier
# lsattr fichier
s-S-ia-A-----e- fichier
# chattr -aisDSA fichier
# lsattr fichier
-------------e- fichier
# rm -f fichier
```

Tâche 12 : Utiliser sudo

1. On crée un compte.

```
[root@linux01 ~]# useradd -m pierre
[root@linux01 ~]# echo pierre | passwd --stdin pierre
Changing password for user pierre.
passwd: all authentication tokens updated successfully.
[root@linux01 ~]#
```

2. On modifie la configuration : pierre peut créer des comptes utilisateurs.

```
[root@linux01 ~]# cp /etc/sudoers /etc/sudoers.000
[root@linux01 ~]# visudo
…
pierre  ALL = (root) /usr/sbin/useraddd
```

3. On se connecte sous le compte pierre.

```
[root@linux01 ~]# ssh -l pierre localhost
pierre@localhost's password: pierre
[pierre@linux01 ~]$ sudo /usr/sbin/useradd -m titeuf

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these two things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

Password: pierre
[pierre@linux01 ~]$ id titeuf
uid=515(titeuf) gid=515(titeuf) groups=515(titeuf)
[pierre@linux01 ~]$ exit
```

4. On crée une configuration plus réaliste.

```
[root@linux01 ~]# cp /etc/sudoers /etc/sudoers.`date '+%m%d%H%M'`
[root@linux01 ~]# ls -l /etc/sudoers*
```

```
-r--r-----  1 root root 621 Dec 30 10:30 /etc/sudoers
-r--r-----  1 root root 580 Dec 30 10:14 /etc/sudoers.000
-r--r-----  1 root root 621 Dec 30 10:39 /etc/sudoers.12301039
[root@linux01 ~]# visudo
…
#pierre linux01 = (root) /usr/sbin/useradd

Cmnd_Alias CMDS_USER  =  /usr/sbin/useradd, \
                         /usr/sbin/userdel, \
                         /usr/sbin/usermod, \
                         /usr/sbin/groupadd, \
                         /usr/bin/passwd [A-z]*, ! /usr/bin/passwd root
%admins ALL = (root) NOPASSWD: CMDS_USER
```

5. On crée des administrateurs.

```
[root@linux01 ~]# groupadd admins
[root@linux01 ~]# useradd -m -g admins cathy
[root@linux01 ~]# echo cathy | passwd --stdin cathy
Changing password for user cathy.
passwd: all authentication tokens updated successfully.
[root@linux01 ~]#
```

6. On teste.

```
[root@linux01 ~]# su - cathy
[cathy@linux01 ~]$ sudo -l
Matching Defaults entries for cathy on this host:
    requiretty, env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE INPUTRC
...
User cathy may run the following commands on this host:
    (root) NOPASSWD: /usr/sbin/useradd, /usr/sbin/userdel, /usr/sbin/usermod,
/usr/sbin/groupadd, /usr/bin/passwd [A-z]*, !/usr/bin/passwd root
[cathy@linux01 ~]$ PATH=$PATH:/usr/sbin
[cathy@linux01 ~]$ sudo useradd -m vomito
[cathy@linux01 ~]$ id vomito
uid=506(vomito) gid=506(vomito) groups=506(vomito)
[cathy@linux01 ~]$ exit
```

Tâche 13 : La gestion des utilisateurs, les droits (rappels)

1. Modifier la configuration d'Apache.

On fait en sorte que l'application Apache s'exécute sous le compte utilisateur cochise et sous le compte groupe indien. L'arborescence /reserve contiendra notre site Web.

```
[root@linux01 ~]# rpm -q httpd
httpd-2.2.14-5.el6.i686
[root@linux01 ~]# cd /etc/httpd/conf
[root@linux01 conf]# cp httpd.conf httpd.conf.000
[root@linux01 conf]# vi httpd.conf  # on ajoute en fin de fichier
...
User cochise
Group indien
DocumentRoot "/reserve"
```

2. Créer les comptes.

```
[root@linux01 conf]# groupadd indien
[root@linux01 conf]# useradd -g indien cochise
```

```
[root@linux01 conf]# groupadd tuniques-bleues
[root@linux01 conf]# useradd -g tuniques-bleues carleton
[root@linux01 conf]# useradd -g indien rasmus
[root@linux01 conf]# useradd -g tuniques-bleues -G indien larry
```

3. Créer le site.

```
[root@linux01 conf]# mkdir /reserve
[root@linux01 conf]# chgrp indien /reserve
[root@linux01 conf]# chmod 3775 /reserve
[root@linux01 conf]# ls -ld /reserve
drwxrwsr-t. 2 root indien 4096 Sep  2 00:58 /reserve
[root@linux01 conf]# chcon -R -t httpd_sys_content_t /reserve
```

Remarque : la dernière commande (`chcon`) change le SC (security context) SELinux du site Web. En effet, par défaut, le serveur Apache, quand SELinux est activé, n'a accès qu'à l'arborescence /var/www.

4. Vérifier la syntaxe et activer le serveur.

```
[root@linux01 conf]# service httpd configtest
Syntax OK
[root@linux01 conf]# service httpd restart
Stopping httpd:                                            [FAILED]
Starting httpd:                                            [  OK  ]
[root@linux01 conf]# id cochise
uid=518(cochise) gid=517(indien) groups=517(indien)
[root@linux01 conf]# ps -o pid,uid,gid,cmd -e |grep httpd
 9552     0     0 /usr/sbin/httpd
 9553   518   517 /usr/sbin/httpd
 9554   518   517 /usr/sbin/httpd
 9555   518   517 /usr/sbin/httpd
 9557   518   517 /usr/sbin/httpd
 9558   518   517 /usr/sbin/httpd
 9559   518   517 /usr/sbin/httpd
 9560   518   517 /usr/sbin/httpd
 9561   518   517 /usr/sbin/httpd
 9569     0     0 grep httpd
```

Remarque
Le premier processus httpd s'exécute sous le compte root pour s'associer au port 80. Ses fils s'exécutent sous les comptes cochise/indien.

5. Les concepteurs de pages travaillent (ils créent des pages).

```
[root@linux01 conf]# cd /reserve
[root@linux01 reserve]# su rasmus
[rasmus@linux01 reserve]$ id
uid=520(rasmus) gid=517(indien) groups=517(indien)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[rasmus@linux01 reserve]$ echo "<h1>Vive PHP</h1>" > php.html
[rasmus@linux01 reserve]$ chmod 640 php.html
[rasmus@linux01 reserve]$ exit
exit
[root@linux01 reserve]# su larry
[larry@linux01 reserve]$ id
uid=521(larry) gid=518(tuniques-bleues) groups=517(indien),518(tuniques-bleues)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[larry@linux01 reserve]$ umask 27
```

```
[larry@linux01 reserve]$ echo "<h1>Vive Perl</h1>" > perl.html
[larry@linux01 reserve]$ ls -l
total 16
-rw-r----- 1 larry  indien 19 Jan  3 01:10 perl.html
-rw-r----- 1 rasmus indien 18 Jan  3 01:10 php.html
```

Remarque
Les pages Web font toutes parties du groupe indien grâce au droit SGID du répertoire /reserve. La commande `chmod` ou la commande `umask` protègent les pages Web. Celles-ci sont uniquement accessibles en lecture, exception faite de leur concepteur.

6. Les pages sont protégées contre les modifications non autorisées.

Grâce aux droits sur le répertoire /reserve, notamment le sticky bit, un concepteur ne peut pas détruire la page d'un collègue. Grâce aux droits sur les pages elles-mêmes, il ne pourra également pas les modifier. Le serveur (s'exécutant sous le compte cochise) a les mêmes restrictions. Notons qu'un CGI s'exécute sous le compte du serveur (cochise). Ainsi un CGI compromis n'aura pas d'influence sur les pages.

```
[larry@linux01 reserve]$ echo "PHP:c'est nul" >> php.html
bash: php.html: Permission denied
[larry@linux01 reserve]$ rm -f php.html
rm: cannot remove `php.html': Operation not permitted
[larry@linux01 reserve]$ exit
[root@linux01 reserve]# su cochise
[cochise@linux01 reserve]$ echo "PHP sans Apache c'est rien" >> php.html
bash: php.html: Permission denied
[cochise@linux01 reserve]$ rm -f php.html
rm: cannot remove `php.html': Operation not permitted
[cochise@linux01 reserve]$ exit
```

7. Un concepteur crée un document inaccessible aux autres.

```
[root@linux01 reserve]# su larry
[larry@linux01 reserve]$ echo "Je deteste PHP" > memo_perso.txt
[larry@linux01 reserve]$ chgrp tuniques-bleues memo_perso.txt
[larry@linux01 reserve]$ chmod 600 memo_perso.txt
[larry@linux01 reserve]$ ls -l
total 24
-rw------- 1 larry  tuniques-bleues 15 Jan  3 01:20 memo_perso.txt
-rw-r----- 1 larry  indien          19 Jan  3 01:10 perl.html
-rw-r----- 1 rasmus indien          18 Jan  3 01:10 php.html
[larry@linux01 reserve]$ exit
exit
```

8. Copie avec préservation.

Les options –p ou –a de cp permettent à l'administrateur de copier des fichiers en gardant leurs caractéristiques (droits, propriétaire, groupe, date de dernière modification…).

```
[root@linux01 reserve]# cp -a memo_perso.txt /tmp
[root@linux01 reserve]# ls -l /tmp/memo_perso.txt
-rw------- 1 larry tuniques-bleues 15 Jan  3 01:20 /tmp/memo_perso.txt
```

9. Un non-concepteur n'a pas accès au site Web.

```
[root@linux01 reserve]# su - carleton
[carleton@linux01 ~]$ id
uid=519(carleton) gid=518(tuniques-bleues) groups=518(tuniques-bleues)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[carleton@linux01 ~]$ ls -l /reserve
total 24
```

```
-rw-r--r-- 1 larry  tuniques-bleues 15 Jan  3 01:20 memo_perso.txt
-rw-r----- 1 larry  indien          19 Jan  3 01:10 perl.html
-rw-r----- 1 rasmus indien          18 Jan  3 01:10 php.html
[carleton@linux01 ~]$ cd /reserve
[carleton@linux01 reserve]$ cat perl.html
cat: perl.html: Permission denied
[carleton@linux01 reserve]$ echo "Un bon indien est un indien mort" >> php.html
-bash: php.html: Permission denied
[carleton@linux01 reserve]$ echo "<h1>Je hais les indiens</h1>" > us_army.html
-bash: us_army.html: Permission denied
[carleton@linux01 reserve]$ cd
[carleton@linux01 ~]$ rm -rf /reserve
rm: cannot remove `/reserve/perl.html': Permission denied
rm: cannot remove `/reserve/php.html': Permission denied
rm: cannot remove `/reserve/memo_perso.txt': Permission denied
[carleton@linux01 ~]$ exit
```

L'administrateur supprime les droits other sur le repertoire `/reserve`. Le non-concepteur n'a plus aucun accès au site.

```
[root@linux01 reserve]# ls -ld /reserve
drwxrwsr-t 2 root indien 4096 Jan  3 01:20 /reserve
[root@linux01 reserve]# chmod o-rx /reserve
[root@linux01 reserve]# ls -ld /reserve
drwxrws--T 2 root indien 4096 Jan  3 01:20 /reserve
[root@linux01 reserve]# su - carleton
[carleton@linux01 ~]$ ls -l /reserve
ls: /reserve: Permission denied
[carleton@linux01 ~]$ exit
```

10. Accéder aux pages en client/serveur.

```
[root@linux01 reserve]# rpm -q lynx
lynx-2.8.6-25.el6.i686
[root@linux01 reserve]# lynx -dump 'http://localhost/php.html'
                                Vive PHP
[root@linux01 reserve]# lynx -dump 'http://localhost/perl.html'
                                Vive Perl
[root@linux01 reserve]# lynx -dump 'http://localhost/memo_perso.txt'
                                Forbidden
   You don't have permission to access /memo_perso.txt on this server.
...
[root@linux01 reserve]# cd
```

Remarque
Quand on accède au site en client/serveur, on voit les pages accessibles à cochise/indien.

11. Remettre la configuration d'origine.

```
[root@linux01 ~]# cp /etc/httpd/conf/httpd.conf /etc/httpd/conf/httpd.conf.001
[root@linux01 ~]# cp /etc/httpd/conf/httpd.conf.000 /etc/httpd/conf/httpd.conf
cp: overwrite `/etc/httpd/conf/httpd.conf'? y
[root@linux01 ~]# /etc/init.d/httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
```