4.11. Fournir des accès sécurisés aux utilisateurs

4.11.1. Authentification utilisateur: PAM

PAM (Pluggable Authentication Modules) permet aux administrateurs système de choisir comment les applications authentifient les utilisateurs. Remarquez que PAM ne peut rien faire tant qu'une application n'a pas été compilée avec la prise en charge pour PAM. La plupart des applications livrées dans Debian ont cette prise en charge intégrée (Debian n'avait pas de prise en charge pour PAM avant la version 2.2). La configuration actuelle par défaut pour tout service activé avec PAM est d'émuler l'authentification UNIX (consultez **`/usr/share/doc/libpam0g/Debian-PAM-MiniPolicy.gz`** pour plus d'informations sur la façon dont les services *devraient* fonctionner dans Debian).

Chaque application avec la prise en charge de PAM fournit un fichier de configuration dans **`/etc/pam.d`** qui peut être utilisé pour modifier son comportement

- quelle fonction de base est utilisée pour l'authentification;
- quelle fonction de base est utilisée pour les sessions;
- comment les vérifications de mots de passe se comportent.

La description qui suit est loin d'être complète, pour plus d'informations vous pouvez lire les http://www.linux-pam.org/Linux-PAM-html/ de référence. Cette documentation est fournie par le paquet **libpam-doc** dans **`/usr/share/doc/libpam-doc/html/`**.

PAM vous offre la possibilité de passer en revue plusieurs étapes d'authentification en une seule fois, à l'insu de l'utilisateur. Vous pouvez vous authentifier à une base de données Berkeley et à un fichier **`passwd`** normal, ainsi l'utilisateur pourra se connecter seulement si l'authentification est correcte des deux côtés. Vous pouvez restreindre beaucoup de choses avec PAM comme vous pouvez laisser libre accès au système. Donc soyez prudent. Une ligne de configuration typique a un champ de contrôle comme deuxième élément. Généralement, il devrait être paramétré sur **`requisite`** qui retourne un échec de connexion si un module échoue.

4.11.2. Sécurité de mot de passe dans PAM

Vérifiez **`/etc/pam.d/common-password`**, englobé dans **`/etc/pam.d/passwd`** [24]. Ce fichier est aussi englobé dans d'autres fichiers de **`/etc/pam.d/`** pour définir le comportement des mots de passe utilisés dans les sous-systèmes qui donnent accès aux services sur la machine, comme la connexion en console (**login**), les gestionnaires de connexion graphiques (comme **gdm** ou **lightdm**) et la connexion à distance (comme **sshd**).

Assurez-vous que le module pam_unix.so utilise l'option « sha512 » pour les mots de passe chiffrés. C'est le cas par défaut dans Debian Squeeze.

La ligne avec la définition du module pam_unix devrait ressembler à :

```
password   [success=1 default=ignore]      pam_unix.so nullok obscure minlen=8 sha512
```

Cette définition :

- renforce le chiffrement des mots de passe conservés, en utilisant la fonction de hachage SHA-512 (option *sha512*) ;
- active les vérifications de complexité des mots de passe (option *obscure*) conformément à la définition de la page de manuel pam_unix(8),
- impose une taille minimale de mot de passe (option *min*) à 8 caractères.

Assurez-vous que les mots de passe chiffrés sont utilisés dans les applications PAM, car cela aide à protéger contre les attaques par dictionnaire. L'utilisation du chiffrement permet aussi d'utiliser des mots de passe plus longs que 8 caractères.

Puisque ce module est aussi utilisé pour définir la façon dont les mots de passe sont changés (il est inclus par **chpasswd**), vous pouvez renforcer la sécurité des mots de passe dans le système en installant **libpam-cracklib** et en introduisant cette définition dans le fichier de configuration **/etc/pam.d/common-password** :

```
# Vérifier que libpam-cracklib soit installé avant sinon vous ne
# pourrez pas vous connecter.
password   required     pam_cracklib.so retry=3 minlen=12 difok=3
password   [success=1 default=ignore]      pam_unix.so obscure minlen=8 sha512 use_authok
```

La première ligne charge le module PAM cracklib, qui fournit la vérification de la sûreté des mots de passe, attend un nouveau mot de passe avec une taille minimale [25] de 12 caractères, une différence d'au moins 3 caractères par rapport à l'ancien et autorise 3 essais. cracklib dépend d'une liste de mots (comme **wenglish**, **wfrench**, **wbritish**, etc.), assurez-vous donc d'en avoir installé une adaptée à votre langue, sinon, cela peut être totalement inutile.

La seconde ligne (utilisant le module pam_unix.so) est la configuration par défaut dans Debian, conformément à la description précédente, hormis pour l'option *use_authok*. L'option **use_authok** est nécessaire si pam_unix.so est empilé après pam_cracklib.so, et est utilisé pour passer le mot de passe du module précédent. Sinon le mot de passe serait demandé deux fois à l'utilisateur.

Pour plus de renseignements sur le réglage de cracklib, consultez la page de manuel pam_cracklib(8) et l'article http://www.deer-run.com/~hal/sysadmin/pam_cracklib.html de Hal Pomeranz.

En activant le module PAM cracklib, vous définissez une règle qui oblige les utilisateurs à utiliser des mots de passe sûrs.

Autrement, les modules PAM peuvent être configurés pour utiliser une authentification à deux facteurs, comme : **libpam-barada**, **libpam-google-authenticator**, **libpam-oath**, **libpam-otpw**, **libpam-poldi**, **libpam-usb** ou **libpam-yubico**. La configuration de ces modules permettrait d'accéder au système en utilisant des mécanismes d'authentification externes comme des cartes à puce, clefs USB externes ou mots de passe uniques générés par des applications externes exécutées, par exemple, sur le téléphone portable de l'utilisateur.

Veuillez remarquer que ces restrictions s'appliquent à tous les utilisateurs mais *pas* aux changements de mot de passe du superutilisateur. Le superutilisateur pourra définir n'importe quel mot de passe (n'importe quelle taille ou complexité) pour lui-même et les autres, quelque soient les restrictions définies ici.

4.11.3. Contrôle de l'accès utilisateur dans PAM

Afin d'être sûr que le superutilisateur peut se connecter uniquement à partir des terminaux locaux, la ligne suivante doit être activée dans **/etc/pam.d/login**:

```
auth     requisite  pam_securetty.so
```

Puis, vous devez modifier la liste des terminaux sur lesquels la connexion du superutilisateur est autorisée dans le fichier **/etc/securetty** (comme c'est décrit en Section 4.7, « Restreindre les accès aux consoles »). Vous pouvez sinon activer le module **pam_access** et modifier **/etc/security/access.conf** qui permet un contrôle plus général et affiné, mais à qui il manque (malheureusement) des messages de journalisation décents (la journalisation dans PAM n'est pas standard et est un problème particulièrement peu gratifiant à traiter). Nous reviendrons au fichier **access.conf** un peu plus tard.

4.11.4. Limites des utilisateurs dans PAM

La ligne suivante devrait être activée dans **/etc/pam.d/login** pour mettre en place des limites de ressource utilisateur.

```
  session  required   pam_limits.so
```

Cela restreint les ressources du système auxquelles les utilisateurs sont autorisées (consultez ci-après Section 4.11.8, « Restreindre l'utilisation des ressources: le fichier limits.conf »). Par exemple, vous pouvez restreindre le nombre de connexions (d'un groupe d'utilisateurs donné ou tout le système), le nombre de processus, la taille de la mémoire, etc.

4.11.5. Contrôle de su dans PAM

Si vous voulez protéger **su**, pour que seules quelques personnes puissent l'utiliser pour devenir superutilisateur sur le système, vous avez besoin de créer un nouveau groupe «wheel» (c'est la meilleure façon, étant donné qu'aucun fichier n'a ces permissions d'attribuées). Ajoutez root et les autres utilisateurs, qui auront la possibilité d'utiliser **su** pour devenir superutilisateur, à ce groupe. Ensuite, ajoutez la ligne suivante dans **/etc/pam.d/su**:

```
  auth       requisite   pam_wheel.so group=wheel debug
```

Cela permet d'être sûr que seules les personnes du groupe «wheel» pourront utiliser **su** pour devenir superutilisateur. Les autres utilisateurs ne seront pas capables de le devenir. En fait, ils recevront un message de refus s'ils essayent de devenir superutilisateur.

Si vous désirez que seulement certains utilisateurs s'authentifient à un service PAM, il suffit d'utiliser les fichiers où sont stockés les utilisateurs autorisés (ou pas) à se connecter. Imaginons que vous ne vouliez autoriser que l'utilisateur «ref» à se connecter avec **ssh**. Vous le mettez dans **/etc/sshusers-allowed** et écrivez ce qui suit dans **/etc/pam.d/ssh**:

```
  auth       required    pam_listfile.so item=user sense=allow file=/etc/sshusers-allowed
onerr=fail
```

4.11.6. Répertoires temporaires dans PAM

Puisqu'il y eu de nombreuses vulnérabilités dites de fichier temporaire non sécurisé, dont thttpd est un exemple (consultez http://www.debian.org/security/2005/dsa-883), **libpam-tmpdir** est un bon paquet à installer. Tout ce que vous avez à faire est d'ajouter ceci à **/etc/pam.d/common-session** :

```
 session    optional     pam_tmpdir.so
```

Une discussion a eu lieu à propos de l'ajout par défaut dans Debian. Consultez http://lists.debian.org/debian-devel/2005/11/msg00297.html pour obtenir plus de renseignements.

4.11.7. Configuration pour les applications PAM non définies

La dernière étape, mais pas la moindre, est de créer le fichier **/etc/pam.d/other** et d'ajouter les lignes suivantes:

```
auth     required       pam_securetty.so
auth     required       pam_unix_auth.so
auth     required       pam_warn.so
auth     required       pam_deny.so
account  required       pam_unix_acct.so
account  required       pam_warn.so
account  required       pam_deny.so
password required       pam_unix_passwd.so
password required       pam_warn.so
password required       pam_deny.so
session  required       pam_unix_session.so
session  required       pam_warn.so
session  required       pam_deny.so
```

Ces lignes vont fournir une bonne configuration par défaut pour toutes les applications qui gèrent PAM (accès refusé par défaut).

4.11.8. Restreindre l'utilisation des ressources: le fichier limits.conf

Vous devriez vraiment jeter un sérieux coup d'œil à ce fichier. Vous pouvez y définir les limites des ressources par utilisateur. Dans d'anciennes versions, ce fichier de configuration était **/etc/limits.conf**, mais dans les nouvelles versions (avec PAM), le fichier de configuration à utiliser devrait être **/etc/security/limits.conf**.

Si vous ne désirez pas restreindre l'utilisation des ressources, *n'importe quel* utilisateur ayant une invite de commandes valable sur le système (ou même un intrus qui aurait compromis le système par un service ou un démon devenu fou) pourra utiliser autant de CPU, de mémoire, de pile, etc. que le système pourra fournir. Ce problème d'*épuisement de ressources* peut être réglé par l'utilisation de PAM.

Il existe un moyen d'ajouter des limites de ressources pour certains interpréteurs de commandes (par exemple, **bash** a **ulimit**, consultez bash(1)), mais comme ils ne fournissent pas tous les mêmes limites et qu'un utilisateur peut changer d'interpréteur (consultez chsh(1)), il est préférable de placer ces limites dans les modules PAM ainsi elles s'appliqueront quel que soit l'interpréteur de commandes utilisé et également aux modules PAM qui ne sont pas orientés interpréteur.

Les limites de ressources sont imposées par le noyau, mais elles doivent être configurées par le fichier **limits.conf** et la configuration PAM des différents services doit charger le module PAM approprié. Vous pouvez vérifier quels services imposent des limites en exécutant:

```
$ find /etc/pam.d/ \! -name "*.dpkg*" | xargs -- grep limits |grep -v ":#"
```

Habituellement, **login**, **ssh** et les gestionnaires de session graphique (**gdm**, **kdm** ou **xdm**) devraient imposer des limites aux utilisateurs, mais vous pouvez vouloir faire cela dans d'autres fichiers de configuration de PAM, comme **cron**, pour empêcher les démons système d'accaparer toutes les ressources système..

Les paramètres de limites spécifiques que vous pouvez vouloir imposer dépendent des ressources du système, c'est l'une des principales raisons pour lesquelles aucune limite n'est imposée dans l'installation par défaut.

Par exemple, l'exemple de configuration ci-dessous impose une limite de 100 processus par utilisateur (pour empêcher les *bombes de fork*) ainsi qu'une limite de 10 Mo de mémoire par processus et une limite de 10 connexions simultanées. Les utilisateurs du groupe **adm** ont des limites supérieures et peuvent créer des fichiers core s'ils le désirent (c'est simplement une limite *soft* (soft)).

```
*               soft    core            0
*               hard    core            0
*               hard    rss             1000
*               hard    memlock         1000
```

```
*               hard    nproc           100
*               -       maxlogins       1
*               hard    data            102400
*               hard    fsize           2048
@adm            hard    core            100000
@adm            hard    rss             100000
@adm            soft    nproc           2000
@adm            hard    nproc           3000
@adm            hard    fsize           100000
@adm            -       maxlogins       10
```

Voici les limites qu'un utilisateur standard (y compris les démons système) aurait:

```
$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) 102400
file size               (blocks, -f) 2048
max locked memory       (kbytes, -l) 10000
max memory size         (kbytes, -m) 10000
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 100
virtual memory          (kbytes, -v) unlimited
```

Et voici les limites d'un utilisateur administratif:

```
$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) 102400
file size               (blocks, -f) 100000
max locked memory       (kbytes, -l) 100000
max memory size         (kbytes, -m) 100000
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 2000
virtual memory          (kbytes, -v) unlimited
```

Pour plus d'informations, consultez:

- guide de référence PAM des modules disponibles
- l'http://www.samag.com/documents/s=1161/sam0009a/0009a.htm ;
- l'article http://seifried.org/security/os/linux/20020324-securing-linux-step-by-step.html pour la section *Limiting users overview* ;
- le http://seifried.org/lasg/users/ pour la section *Limiting and monitoring users*.

4.11.9. Actions de connexion de l'utilisateur: modification de /etc/login.defs

La prochaine étape est d'éditer les configuration et action de base lors de la connexion de l'utilisateur. Notez que ce fichier ne fait pas partie de la configuration PAM, c'est un fichier de configuration qui est pris en compte par les programmes **login** et **su**, il n'est pas logique de l'adapter aux cas pour lesquels ni l'un ni l'autre des programmes n'est appelé au moins indirectement (le programme **getty** qui gère les consoles et offre l'invite de connexion initiale appelle

bien **login**).

```
FAILLOG_ENAB        yes
```

Si vous activez cette variable, les connexions échouées seront enregistrées dans un journal. Il est important d'en garder une trace pour repérer si quelqu'un tente une attaque par force brute.

```
LOG_UNKFAIL_ENAB    no
```

En configurant cette variable à « yes », les noms d'utilisateur seront enregistrés en cas d'échec de connexion. Laisser la configuration à « no » (par défaut) est plus prudent, puisque sinon, les mots de passe d'utilisateurs pourraient être enregistrés par erreur (si un utilisateur fait une faute de frappe et entre le mot de passe à la place de l'identifiant). Si vous configurez à « yes », assurez-vous que les journaux ont les droits adéquats (640 par exemple, avec une configuration de groupe adéquate comme adm).

```
SYSLOG_SU_ENAB      yes
```

Cela va activer l'écriture dans les journaux de **syslog** des tentatives de **su**. Plutôt important sur des machines sérieuses, mais notez que cela peut aussi bien être à la base de problèmes de respect de la vie privée.

```
SYSLOG_SG_ENAB      yes
```

La même chose que SYSLOG_SU_ENAB, mais s'applique au programme **sg**.

```
ENCRYPT_METHOD  SHA512
```

Comme mentionné ci-dessus, les mots de passe chiffrés réduisent considérablement le problème des attaques par dictionnaire étant donné que vous pouvez utiliser des mots de passe plus longs. Cette définition doit être cohérente avec la valeur définie dans **/etc/pam.d/common-password**.

4.11.10. Actions de connexion de l'utilisateur: modification de /etc/pam.d/login

Le fichier de configuration de connexion peut être ajusté pour implémenter une politique plus stricte. Par exemple, la configuration par défaut peut être modifiée pour augmenter le délai entre les invites de connexion. La configuration par défaut définit à 3 secondes le délai :

```
auth       optional   pam_faildelay.so  delay=3000000
```

Augmenter la valeur de *delay* à une valeur suffisamment grande permet de rendre plus difficiles les tentatives de connexion en utilisant la force brute. Si un mauvais mot de passe est fourni, le pirate potentiel (ou le simple utilisateur!) doit attendre plus longtemps avant d'obtenir une nouvelle invite de connexion, ce qui prend pas mal de temps quand vous testez des mots de passe. Par exemple, avec *delay=10000000*, les utilisateurs devront attendre 10 secondes s'ils ont tapé un mauvais mot de passe.

Ce fichier permet aussi de définir le message présenté aux utilisateurs avant de se connecter. C'est désactivé par défaut, comme ci-dessous :

```
# auth       required   pam_issue.so issue=/etc/issue
```

Si la politique de sécurité l'exige, ce fichier peut être utilisé pour montrer un message standard indiquant que l'accès au système est restreint et que l'accès des utilisateurs est journalisé. Ce type de déclaration peut être nécessaire dans certains environnements et juridictions. Pour l'activer, ajoutez simplement les renseignements nécessaires dans le fichier **/etc/issue** [26] et décommentez la ligne activant le module pam_issue.so dans **/etc/pam.d/login**. Dans

ce fichier, vous pouvez aussi activer des fonctionnalités supplémentaires qui pourraient être pertinentes à appliquer aux politiques de sécurité locales comme :

- la définition de règles accordant l'accès à certains utilisateurs suivant l'heure, en activant le module *pam_time.so* et en configurant **/etc/security/time.conf** en conséquence (désactivée par défaut) ;
- la définition de sessions de connexion pour utiliser les limitations aux utilisateurs conformément à la définition de **/etc/security/limits.conf** (activée par défaut) ;
- la présentation à l'utilisateur des renseignements sur la précédente connexion (activée par défaut) ;
- l'affichage d'un message (**/etc/motd** et **/run/motd.dynamic**) aux utilisateurs après la connexion (activée par défaut) ;

4.11.11. Restreindre le FTP: éditer /etc/ftpusers

Ce fichier contient une liste d'utilisateurs qui ne sont pas autorisés à se connecter à l'hôte en utilisant FTP. Utilisez uniquement ce fichier si vous voulez réellement autoriser le FTP (qui n'est, en général, pas recommandé car il utilise des mots de passe en clair). Si le démon gère PAM, cela peut être utilisé pour permettre ou refuser certains services aux utilisateurs.

FIXME (bogue) : Est-ce un bogue que le fichier par défaut **ftpusers** dans Debian ne contienne *pas* tous les utilisateurs d'administration (dans **base-passwd**) ?

Un moyen pratique d'ajouter tous les comptes système à **/etc/ftpusers** est d'exécuter

```
$ awk -F : '{if ($3<1000) print $1}' /etc/passwd > /etc/ftpusers
```

4.11.12. Utilisation de su

Si vous avez réellement besoin que des utilisateurs deviennent superutilisateur sur le système, par exemple pour installer des paquets ou ajouter des utilisateurs, vous pouvez utiliser la commande **su** pour changer d'identité. Vous devriez essayer d'éviter toute connexion en tant que superutilisateur et d'utiliser à la place **su**. En réalité, la meilleure solution est de supprimer **su** et de changer pour le mécanisme **sudo** qui a une logique plus large et plus de fonctionnalités que **su**. Cependant, **su** est plus commun étant donné qu'il est utilisé sur beaucoup d'autres UNIX.

4.11.13. Utilisation de sudo

sudo autorise l'utilisateur à exécuter des commandes définies sous l'identité d'un autre utilisateur, même en tant que superutilisateur. Si l'utilisateur est ajouté à **/etc/sudoers** et est authentifié correctement, il est capable de lancer des commandes qui ont été définies dans **/etc/sudoers**. Les infractions, telles que les mots de passe incorrects ou les tentatives de lancement d'un programme pour lequel vous n'avez pas les permissions, sont logguées et envoyées au superutilisateur.

4.11.14. Désactiver des accès d'administration à distance

Vous devriez également modifier **/etc/security/access.conf** pour désactiver la connexion d'administration à distance. Ainsi, les utilisateurs doivent exécuter **su** (ou **sudo**) pour utiliser des pouvoirs administratifs et ainsi la trace d'audit appropriée sera toujours générée.

Vous devez ajouter la ligne suivante à **/etc/security/access.conf**, le fichier de configuration par défaut Debian contient une ligne d'exemple commentée:

```
    -:wheel:ALL EXCEPT LOCAL
```

Rappelez-vous d'activer le module **pam_access** pour chaque service (ou configuration par défaut) dans **/etc/pam.d/** si vous voulez que vos modifications dans **/etc/security/access.conf** soient prises en compte.

4.11.15. Restriction des utilisateurs

Parfois, vous pensez avoir besoin d'utilisateurs créés dans le système local de façon à fournir un service donné (service courrier POP3 ou FTP). Avant tout, rappelez-vous que l'implémentation PAM dans Debian GNU/Linux vous autorise à valider les utilisateurs avec une grande variété de répertoires de services externes (radius, LDAP, etc.) fournis par les paquets libpam.

Si des utilisateurs doivent être créés et que le système est accessible à distance, prenez en compte que des utilisateurs pourront se connecter au système. Cela peut être corrigé en donnant aux utilisateurs un interpréteur de commandes vide (**/dev/null**) (qui doit être dans **/etc/shells**). Si vous voulez autoriser les utilisateurs à accéder au système mais limiter leurs mouvements, vous pouvez utiliser le fichier **/bin/rbash**, ce qui est équivalent à l'ajout de l'option **-r** dans bash (consultez *INTERPRÉTEUR RESTREINT* dans bash(1)). Veuillez noter que même avec un interpréteur de commandes restreint, un utilisateur ayant accès à un programme interactif (qui peut permettre l'exécution d'un sous-interpréteur) peut être capable de passer outre les limites de l'interpréteur de commandes.

Debian fournit actuellement dans la version unstable le module **pam_chroot** (dans le paquet **libpam-chroot**) (et il pourrait être inclus dans les prochaines versions stables). Une alternative à celui-ci est de **chroot**er le service qui fournit la connexion à distance (**ssh**, **telnet**). [27]

Si vous voulez restreindre *quand* les utilisateurs peuvent accéder au système, vous devrez personnaliser **/etc/security/access.conf** en fonction de vos besoins.

Des informations sur la façon de **chroot**er des utilisateurs accédant au système par le service **ssh** sont décrites dans Section B.7, « Environnement de chroot pour **SSH** »

4.11.16. Audit d'utilisateur

Si vous êtes vraiment paranoïaque, vous pourriez configurer l'environnement pour superviser ce que les utilisateurs font sur le système. Cette section présente quelques conseils avec différents utilitaires que vous pouvez utiliser.

4.11.16.1. Audit d'entrée et sortie avec script

Vous pouvez utiliser la commande **script** pour surveiller à la fois ce que les utilisateurs exécutent et les résultats de leurs commandes. Vous ne pouvez pas configurer **script** comme un interpréteur de commandes (même si vous l'ajoutez à **/etc/shells**). Mais vous pouvez faire en sorte que le fichier d'initialisation de l'interpréteur de commandes exécute les commandes suivantes:

```
umask 077
exec script -q -a "/var/log/sessions/$USER"
```

Bien sûr, si vous faites cela pour tout le système, cela veut dire que l'interpréteur ne continuerait pas à lire les fichiers d'initialisation personnels (car l'interpréteur sera écrasé par **script**). Une solution est de le faire dans les fichiers d'initialisation de l'utilisateur (mais l'utilisateur pourrait alors l'enlever, consultez les commentaires sur cela ci-dessous).

Vous devez également configurer les fichiers dans le répertoire d'audit (dans l'exemple **/var/log/sessions/**) pour que les utilisateurs puissent y écrire, mais pas supprimer le fichier. Cela pourrait être fait, par exemple, en créant les fichiers de session d'utilisateur en avance et en positionnant l'option *append-only* («append-only») en utilisant **chattr**.

Une alternative utile pour les administrateurs système, qui inclut des informations de date, serait:

```
umask 077
exec script -q -a "/var/log/sessions/$USER-`date +%Y%m%d`"
```

4.11.16.2. Utiliser le fichier d'historique de l'interpréteur de commandes

Si vous voulez passer en revue ce que les utilisateurs entrent dans l'interpréteur de commandes (mais sans voir le résultat), vous pouvez configurer un **/etc/profile** pour tout le système qui configure l'environnement pour que toutes les commandes soient enregistrées dans le fichier d'historique. La configuration pour tout le système doit être réalisée de telle façon que les utilisateurs ne puissent pas enlever les capacités d'audit de leur interpréteur de commandes. C'est plutôt spécifique à l'interpréteur de commandes, donc assurez-vous que tous les utilisateurs utilisent un interpréteur de commandes qui le permet.

Par exemple, pour bash, le fichier **/etc/profile** pourrait être paramétré ainsi [28]:

```
HISTFILE=~/.bash_history
HISTSIZE=10000
HISTFILESIZE=999999
# Empêcher les utilisateurs d'entrer des commandes qui seraient
# ignorées dans le fichier d'historique
HISTIGNORE=""
HISTCONTROL=""
readonly HISTFILE
readonly HISTSIZE
readonly HISTFILESIZE
readonly HISTIGNORE
readonly HISTCONTROL
export HISTFILE HISTSIZE HISTFILESIZE HISTIGNORE HISTCONTROL
```

Afin que cela fonctionne, l'utilisateur doit être seulement capable d'ajouter des informations au fichier **.bash_history**. Vous devez *aussi* positionner l'attribut *append-only* en utilisant le programme **chattr** sur **.bash_history** pour tous les utilisateurs. [29]

Notez que vous pouvez introduire la configuration ci-dessus dans le fichier utilisateur **.profile**. Mais alors vous devriez configurer les permissions correctement de façon à empêcher à l'utilisateur de modifier ce fichier. Cela inclut: les répertoires personnels de l'utilisateur ne doivent *pas* appartenir à l'utilisateur (sinon, il pourrait supprimer le fichier), mais en même temps lui permettre de lire le fichier de configuration **.profile** et d'écrire dans **.bash_history**. Il serait bien de configurer l'attribut *immuable* (également en utilisant **chattr**) pour le **.profile** aussi si vous procédez ainsi.

4.11.16.3. Audit utilisateur complet avec utilitaires de comptabilité

L'exemple précédent est une manière simple de configurer l'audit utilisateur, mais qui peut ne pas être utile pour des systèmes complexes ou pour ceux dans lesquels les utilisateurs ne peuvent pas exécuter d'interpréteur de commande du tout (ou exclusivement). Si c'est le cas, vous devrez examiner **acct**, les utilitaires de comptabilité. Ces utilitaires archiveront toutes les commandes exécutées par les utilisateurs ou processus du système au détriment de l'espace disque.

Lors de l'activation de la comptabilité, toutes les informations sur les processus et utilisateurs sont conservées dans **/var/account/**, plus spécifiquement dans le fichier **pacct**. Le paquet de comptabilité inclut certains outils (**sa** et **ac**) afin d'analyser ces données.

4.11.16.4. Autres méthodes d'audit utilisateur

Si vous êtes complètement paranoïaque et que vous voulez auditer toutes les commandes des utilisateurs, vous pouvez prendre les codes source de **bash**, les modifier et récupérer dans un fichier toutes les commandes qu'un utilisateur tape. Vous pourriez aussi avoir **ttysnoop** constamment en attente de nouveaux ttys [30] et reverser toutes les sorties dans un fichier. Un autre programme utile est **snoopy** (consultez également

http://sourceforge.net/projects/snoopylogger/) qui est un programme transparent pour l'utilisateur qui se positionne comme une bibliothèque fournissant une encapsulation des appels **execve()**, toute commande exécutée est journalisée par **syslogd** en utilisant la fonctionnalité **authpriv** (généralement stockée dans **/var/log/auth.log**).

4.11.17. Inspection des profils utilisateurs

Si vous désirez *voir* ce que font vraiment les utilisateurs, comme l'heure à laquelle ils se connectent, vous pouvez utiliser la base de données **wtmp** qui contient toutes les informations concernant les connexions. Ce fichier peut être employé avec plusieurs utilitaires, parmi eux **sac** peut sortir un profil de chaque utilisateur montrant dans quel créneau horaire il se connecte habituellement au système.

Dans le cas où vous avez la comptabilité activée, vous pouvez également utiliser les outils qu'elle fournit pour déterminer quand les utilisateurs accèdent au système et ce qu'ils exécutent.

4.11.18. Positionner des umasks aux utilisateurs

En fonction de la politique d'utilisateur, vous pourriez modifier la façon dont les renseignements sont partagés entre utilisateurs, c'est-à-dire quels sont les droits de nouveaux fichiers par défaut créés par les utilisateurs.

Le paramètre **umask** par défaut de Debian est *022*, cela signifie que les fichiers (et les répertoires) peuvent être lus et accédés par le groupe de l'utilisateur et par tout autre utilisateur du système. Cette définition est configurée dans le fichier de configuration normalisé **/etc/profile** utilisé par tous les interpréteurs de commandes.

Si la valeur par défaut de Debian est trop permissive pour le système, vous devrez changer ce paramètre umask pour tous les interpréteurs de commandes. Parmi les configurations plus restrictives d'umask, 027 (pas d'accès permis aux nouveaux fichiers pour le groupe *other*, c'est-à-dire aux autres utilisateur du système) ou 077 (pas d'accès permis aux nouveaux fichiers pour les membres du groupe de l'utilisateur) peuvent être utilisés. Debian (par défaut [31]) crée un groupe par utilisateur de telle sorte que seul l'utilisateur soit inclus dans son groupe. Par conséquent, 027 et 077 sont équivalents car le groupe de l'utilisateur ne contient que l'utilisateur lui-même.

Cette modification est configurée en définissant un réglage correct de **umask** pour tous les utilisateurs. Vous pouvez modifier cela en introduisant un appel **umask** dans les fichiers de configuration de l'interpréteur de commandes : **/etc/profile** (source par tous les interpréteurs de commandes compatibles Bourne), **/etc/csh.cshrc**, **/etc/csh.login**, **/etc/zshrc** et probablement d'autres (en fonction des interpréteurs de commandes installés sur le système). Vous pouvez aussi modifier le réglage de **UMASK** dans **/etc/login.defs**. De toutes celles-là, la dernière chargée par l'interpréteur de commandes est prioritaire. L'ordre est : la configuration système par défaut pour l'interpréteur de l'utilisateur (c'est-à-dire **/etc/profile** et les autres fichiers de configuration globaux du système) et ensuite ceux de l'utilisateur (ses **~/.profile**, **~/.bash_profile**, etc.). Certains interpréteurs, cependant, peuvent être exécutés avec une valeur *nologin* avec laquelle certains de ces fichiers pourraient être sautés. Consultez la page de manuel de l'interpréteur pour obtenir de plus amples renseignements.

Pour les connexions qui utilisent **login**, la définition de **UMASK** de **/etc/login.defs** est utilisée avant toutes les autres. Cependant, cette valeur ne s'applique pas aux programmes exécutés par l'utilisateur qui n'utilisent pas **login** comme ceux exécutés à travers **su**, **cron** ou **ssh**.

N'oubliez pas de vérifier et éventuellement modifier les fichiers de configuration utilisateur sous **/etc/skel/** car ce sont ceux qui seront utilisés par défaut quand ils sont créés avec la commande **adduser**. Les fichiers de configuration utilisateur Debian par défaut ne contiennent pas d'appel **umask** mais s'il y en a dans n'importe quel fichier de configuration utilisateur, les utilisateurs nouvellement créés pourraient avoir une valeur différente.

Notez, cependant, que les utilisateurs peuvent modifier leur propre paramètre **umask** s'ils le désirent, le rendant plus permissif ou plus restrictif, en modifiant leurs propres fichiers de configuration utilisateur.

Le paquet **libpam-umask** règle l'**umask** par défaut utilisant PAM. Après l'installation du paquet, ajoutez ceci à **/etc/pam.d/common-session**:

```
session    optional     pam_umask.so umask=077
```

Enfin, vous pourriez envisager de modifier l'umask par défaut du superutilisateur à 022 (tel que défini dans **/root/.bashrc**) à une valeur plus restrictive. Cela évitera à l'administrateur système de laisser fuir par inadvertance des fichiers sensibles lorsqu'il travaille en tant que superutilisateur dans des répertoires lisibles par tous (comme **/tmp**) et en les rendant lisibles aux autres utilisateurs.

4.11.19. Limiter ce que les utilisateurs peuvent voir et accéder

FIXME: Besoin de contenu. Indiquer les conséquences de changement des permissions des paquets lors d'une mise à jour (un administrateur aussi paranoïaque que cela devrait **chroot**er ses utilisateurs au passage) s'il n'utilise pas **dpkg-statoverride**.

Si vous avez besoin d'accorder aux utilisateurs un accès au système avec un interpréteur de commandes, réfléchissez-y très soigneusement. Un utilisateur peut, par défaut à moins d'être dans un environnement extrêmement restreint (comme une prison **chroot**), récupérer un assez grand nombre d'informations concernant le système, y compris :

- certains fichiers de configuration dans **/etc**. Cependant, les permissions par défaut de Debian pour certains fichiers sensibles (qui peuvent, par exemple, contenir des mots de passe) empêcheront l'accès à des informations critiques. Pour voir quels fichiers ne sont accessibles que par le superutilisateur, exécutez par exemple **find /etc -type f -a -perm 600 -a -uid 0** en tant que superutilisateur ;

  ```
  find /etc -type f -a -perm 600 -a -uid 0
  ```

  en tant que superutilisateur

- vos paquets installés, soit en consultant la base de données des paquets, soit dans le répertoire **/usr/share/doc**, soit en devinant en regardant les binaires et bibliothèques installés sur le système ;

- certains fichiers journaux dans **/var/log**. Notez également que quelques fichiers journaux ne sont accessibles qu'au superutilisateur et au groupe **adm** (essayez

  ```
  find /var/log -type f -a -perm 640
  ```

) et certains ne sont même disponibles que pour le superutilisateur (essayez

  ```
  find /var/log -type f -a -perm
      600 -a -uid 0
  ```

).

Que peut voir un utilisateur dans le système? Probablement un assez grand nombre de choses, essayez ceci (prenez une profonde respiration):

```
find / -type f -a -perm +006 2>/dev/null
find / -type d -a -perm +007 2>/dev/null
```

La liste des fichiers qu'un utilisateur peut *voir* et des répertoires auxquels il a accès est affichée.

4.11.19.1. Limiter l'accès aux informations d'autres utilisateurs

Si vous accordez toujours un accès d'interpréteur de commandes aux utilisateurs, vous pouvez vouloir limiter les informations qu'ils peuvent voir des autres utilisateurs. Les utilisateurs ayant un accès d'interpréteur de commandes ont tendance à créer un grand nombre de fichiers dans leur répertoire $HOME: boîtes aux lettres, documents personnels, configuration des applications X/GNOME/KDE, etc.

Sous Debian, chaque utilisateur est créé avec un groupe associé et aucun utilisateur n'appartient au groupe d'un autre utilisateur. Il s'agit du comportement par défaut: quand un compte d'utilisateur est créé, un groupe du même nom est créé et l'utilisateur lui est attribué. Cela évite le concept d'un groupe *users* qui peut rendre plus difficile pour les utilisateurs de cacher des informations aux autres utilisateurs.

Cependant, les répertoires **$HOME** des utilisateurs sont créés avec les permissions 0755 (lisible par le groupe et par tout le monde). Les permissions de groupe ne sont pas un problème car seul l'utilisateur appartient au groupe, cependant les permissions pour les autres peuvent être (ou non) un problème selon vos règles locales.

Vous pouvez changer ce comportement pour que la création d'utilisateur fournisse des permissions sur **$HOME** différentes. Pour changer ce comportement pour les *nouveaux* utilisateurs quand ils seront créés, changez *DIR_MODE* dans le fichier de configuration **/etc/adduser.conf** à 0750 (pas d'accès en lecture pour tout le monde).

Les utilisateurs peuvent toujours partager des informations, mais pas directement dans leur répertoire **$HOME** à moins qu'ils ne changent les permissions de celui-ci.

Notez que désactiver les répertoires utilisateur lisibles par tout le monde empêchera les utilisateurs de créer leurs pages personnelles dans le répertoire **~/public_html** car le serveur web ne pourra pas lire un composant du chemin — leur répertoire **$HOME**. Si vous voulez permettre aux utilisateurs de publier des pages HTML dans leur **~/public_html**, changez *DIR_MODE* en 0751. Cela permettra au serveur web d'accéder à ce répertoire (qui devrait lui-même avoir le mode 0755) et de fournir le contenu publié par les utilisateurs. Bien sûr, nous ne parlons ici que d'une configuration par défaut; les utilisateurs peuvent généralement ajuster les permissions de leurs fichiers comme ils le désirent, ou vous pouvez conserver le contenu destiné au web dans un emplacement séparé qui n'est pas un sous-répertoire du répertoire **$HOME** de chaque utilisateur.

4.11.20. Générer des mots de passe utilisateur

Il y a plusieurs cas dans lesquels un utilisateur a besoin de créer un grand nombre de comptes utilisateur et de fournir des mots de passe pour tous ceux-ci. Bien sûr, l'administrateur peut facilement positionner le mot de passe pour être le même que le nom du compte utilisateur, mais cela n'est pas très conseillé sur le plan de la sécurité. Une meilleure approche est d'utiliser un programme de génération de mots de passe. Debian fournit les paquets **makepasswd**, **apg** et **pwgen** qui contiennent des programmes (dont le nom est le même que celui du paquet) qui peuvent être utilisés dans ce but. **makepasswd** génère des mots de passe vraiment aléatoires avec un accent sur la sécurité plus que la prononçabilité tandis que **pwgen** essaie de créer des mots de passe sans signification, mais prononçables (bien sûr, cela dépend de votre langue maternelle). **apg** dispose d'algorithmes pour les deux (il y a une version client/serveur pour ce programme, mais elle n'est pas incluse dans le paquet Debian).

Passwd ne permet pas une attribution non interactive des mots de passe (car il utilise un accès direct au terminal tty). Si vous désirez changer des mots de passe lors de la création d'un grand nombre d'utilisateurs, vous pouvez les créer en utilisant **adduser** avec l'option **--disabled-login**, puis utiliser **usermod** ou **chpasswd** [32] (tous les deux dans le paquet **passwd**, ils sont donc déjà installés). Si vous voulez utiliser un fichier avec toutes les informations pour créer les utilisateurs comme un processus batch, il sera probablement préférable d'utiliser **newusers**.

4.11.21. Vérifier les mots de passe utilisateur

Les mots de passe des utilisateurs peuvent parfois devenir le *maillon faible* de la sécurité d'un système donné. Cela provient du fait que quelques utilisateurs choisissent des mots de passe faibles pour leur compte (et plus il y a d'utilisateurs, plus grandes sont les chances que cela se produise). Même si vous mettez en place des vérifications avec le module PAM cracklib et les limitations sur les mots de passe comme décrites dans Section 4.11.1, « Authentification utilisateur: PAM », les utilisateurs pourront toujours utiliser des mots de passe faibles. Comme l'accès

utilisateur peut inclure un accès à une invite de commandes à distance (en espérant que ce soit avec **ssh**), il est important de rendre les mots de passe aussi difficile à deviner que possible pour les attaquants à distance, particulièrement s'ils ont pu récupérer des informations importantes comme les noms d'utilisateur ou même les fichiers **passwd** et **shadow** eux-mêmes.

Un administrateur système doit, suivant le nombre d'utilisateurs, vérifier si les mots de passe sont cohérents avec la règle locale de sécurité. Comment vérifier? Essayez de les casser comme le ferait un attaquant s'il avait accès aux mots de passe hachés (le fichier **/etc/shadow**).

Un administrateur peut utiliser **john** ou **crack** (tous deux utilisent la force brute) ensemble avec une liste de mots appropriés pour vérifier les mots de passe utilisateurs et prendre des mesures appropriées si un mot de passe faible est détecté. Vous pouvez rechercher des paquets Debian contenant des listes de mots en utilisant **apt-cache search wordlist** ou vous pouvez également visiter des sites de listes de mots sur Internet classique comme ftp://ftp.ox.ac.uk/pub/wordlists ou ftp://ftp.cerias.purdue.edu/pub/dict.

4.11.22. Déconnecter les utilisateurs inactifs (idle)

L'inactivité des utilisateurs pose habituellement un problème de sécurité, un utilisateur peut être inactif parce qu'il est parti déjeuner ou parce qu'une connexion à distance s'est bloquée et n'a pas été rétablie. Quelqu'en soit la raison, les utilisateurs inactifs peuvent amener à une compromission:

- car la console de l'utilisateur peut être débloquée et peut être accédée par un intrus ;
- car un attaquant peut être capable de se rattacher lui-même à une connexion réseau fermée et envoyer des commandes à l'invite de commandes distante (c'est assez facile si l'invite de commandes distante n'est pas chiffrée comme avec **telnet**).

Certains systèmes à distance ont même été compromis à travers un **screen** inactif (et détaché).

La déconnexion automatique des utilisateurs inactifs est habituellement une partie qui doit être imposée par les règles de sécurité locales. Plusieurs moyens existent pour cela:

- si **bash** est l'interpréteur de commandes de l'utilisateur, un administrateur système peut positionner une valeur **TMOUT** par défaut (consultez bash(1)) qui entraînera la déconnexion automatique des utilisateurs distants inactifs. Notez que cela doit être configuré avec l'option **-o** ou les utilisateurs pourront la changer (ou la désactiver) ;
- installez **timeoutd** et configurez **/etc/timeouts** selon vos règles de sécurité locales. Le démon regardera les utilisateurs inactifs et mettra un terme à leur invite de commandes en fonction ;
- installez **autolog** et configurez-le pour enlever les utilisateurs inactifs.

Les démons **timeoutd** et **autolog** sont les méthodes préférées car, après tout, les utilisateurs peuvent changer d'interpréteur de commandes par défaut ou peuvent, après avoir exécuté leur interpréteur de commandes par défaut, basculer sur un autre interpréteur de commandes (non contrôlé).

---

[24] Dans les anciennes versions de Debian, la configuration des modules était définie directement dans **/etc/pam.d/passwd**.

[25] L'option *minlen* n'est pas tout à fait claire, et n'indique pas exactement le nombre de caractères du mot de passe. Un compromis peut être défini entre la complexité et la taille en ajustant les paramètres « credit » de différentes classes de caractères. Pour plus de renseignements, consultez la page de manuel pam_cracklib(8).

[26] Le contenu par défaut de ce fichier fournit des renseignements sur le système d'exploitation et la version utilisée par le système, que vous pourriez ne pas vouloir dévoiler aux utilisateurs anonymes.

[27] **libpam-chroot** n'a pas encore été testé en profondeur, il fonctionne pour **login**, mais il est possible qu'il ne soit pas facile de mettre en place l'environnement pour d'autres programmes.

[28] Configurer HISTSIZE à une très grande valeur peut poser des problèmes avec certains interpréteur de commandes car l'historique est gardé en mémoire pour la session de chaque utilisateur. Il peut être plus prudent de positionner cela à une valeur assez élevée et de sauvegarder les fichiers d'historique des utilisateurs (si vous avez besoin de tout l'historique de l'utilisateur pour une raison ou une autre).

[29] Sans l'attribut append-only les utilisateurs seraient capables de vider le contenu du fichier des historiques avec **`.bash_history`**.

[30] Les ttys sont créées pour les connexions locales et à distance par SSH et TELNET.

[31] Tel que défini dans **`/etc/adduser.conf`** (USERGROUPS=yes). Vous pouvez modifier ce comportement en configurant cette valeur à « no », bien que ce ne soit pas recommandé.

[32] **`chpasswd`** ne sait pas gérer la génération de mots de passe MD5, il faut donc lui donner le mot de passe sous sa forme chiffrée avant de l'utiliser avec l'option

```
-e
```

.

Précédent 4.10. Monter correctement les partitions
Niveau supérieur
Sommaire
Suivant 4.12. Utilisation de tcpwrappers