ROYAUME DU MAROC

مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle et de la Promotion du Travail

Gestion des utilisateurs et de la sécurité sous Gnu/Linux

www.ofppt.info

OFPPT

DIRECTION **R**ECHERCHE ET **I**NGENIERIE DE **F**ORMATION

SECTEUR NTIC

Sommaire

La règle de sécurité numéro un sous Unix est de ne jamais travailler dans le compte root. En effet, ce compte dispose de tous les droits, et la moindre erreur de la part de l'utilisateur dans ce compte peut endommager non seulement ses propres données, mais également l'ensemble du système d'exploitation. De plus, le fait de ne pas travailler sous le compte root restreint à un seul utilisateur les dégâts que pourraient faire un éventuel virus ou programme défectueux.

L'une des premières étapes dans l'installation d'un système est donc de créer un compte utilisateur normal, qui devra être utilisé pour le travail quotidien. Le compte root ne doit donc être réservé qu'aux tâches d'administration, et toute opération réalisée sous cette identité doit être contrôlée deux fois avant d'être effectivement lancée. Les programmes d'installation des distributions demandent donc toujours un mot de passe pour protéger le compte root et le nom et le mot de passe pour au moins un compte utilisateur standard après une nouvelle installation. Le mot de passe root doit être choisi avec un grand soin, surtout si l'ordinateur est susceptible d'être connecté à Internet. En effet, la moindre erreur de configuration au niveau des services fournis par l'ordinateur, couplée avec un mot de passe faible, risque de laisser votre ordinateur à la merci de pirates mal intentionnés.

Ces mêmes programmes d'installation peuvent être utilisés par la suite pour ajouter de nouveaux utilisateurs dans le système. Il est d'ailleurs recommandé de les utiliser dès qu'une telle opération doit être effectuée. Cela dit, il est bon de connaître la manière dont les utilisateurs sont gérés dans les systèmes Unix, aussi une approche plus bas niveau sera-t-elle adoptée dans cette section.

1. Mécanismes d'authentification des utilisateurs

La sécurité des systèmes Unix repose fondamentalement sur les mécanismes d'authentification des utilisateurs. Ces mécanismes visent à s'assurer que chacun est bien celui qu'il prétend être, afin de donner à chacun les droits d'accès aux différents services du système en fonction de ses privilèges.

L'accès aux services du système repose donc sur deux opérations essentielles : l'*identification* et l'*authentification*. L'opération d'identification consiste à annoncer qui l'on est, afin de permettre au système de déterminer les droits auxquels on a droit, et l'opération d'authentification consiste à « prouver » qu'on est bien celui qu'on prétend être. Le système refuse ses services à tout utilisateur inconnu (c'est-à-dire qui s'est identifié sous un nom inconnu) ou qui n'a pas passé avec succès la phase d'authentification.

En interne, les systèmes Unix identifient les utilisateurs par un numéro qui est propre à chacun, son « UID » (abréviation de l'anglais « User IDentifier »), mais il existe une correspondance entre cet UID et un nom d'utilisateur plus humainement lisible. Ce nom est classiquement appelé le « login », en raison du

fait que c'est la première chose que le système demande lorsqu'on cherche à accéder à ses services, pendant l'opération dite de login.

L'authentification des utilisateurs se fait classiquement par mot de passe, bien que d'autres mécanismes soient possibles en théorie. L'accès au système se passe donc toujours de la manière suivante :

- le système demande à l'utilisateur son nom (c'est-à-dire son login) ;
- il demande ensuite son mot de passe ;
- il vérifie la validité du couple (login / mot de passe) pour déterminer si l'utilisateur a le droit de l'utiliser ;
- et, si l'utilisateur est connu et s'est correctement authentifié, le programme qui a réalisé l'authentification prend l'identité et les privilèges de l'utilisateur, fixe son environnement et ses préférences personnelles, puis lui donne accès au système.

L'exemple classique de ces opérations est tout simplement l'opération de login sur une console : le programme **getty** de gestion de la console demande le nom de l'utilisateur, puis passe ce nom au programme **login** qui l'authentifie en lui demandant son mot de passe. Si l'authentification a réussi, il prend l'identité de cet utilisateur et lance son shell préféré. La suite des opérations dépend du shell. S'il s'agit de **bash**, le fichier de configuration /etc/profile est exécuté (il s'agit donc du fichier de configuration dans lequel toutes les options communes à tous les utilisateurs pourront être placées par l'administrateur), puis les fichiers de configuration ~/.bash_profile et ~/.bashrc sont exécutés. Ces deux fichiers sont spécifiques à chaque utilisateur et permettent à chacun d'entre eux de spécifier leurs préférences personnelles. Le fichier ~/.bash_profile n'est exécuté que lors d'un nouveau login, alors que le fichier ~/.bashrc est exécuté à chaque nouveau lancement de **bash**.

Bien entendu, ces opérations nécessitent que des informations relatives aux utilisateurs soient stockées dans le système. Historiquement, elles étaient effectivement stockées dans le fichier de configuration /etc/passwd. Cela n'est, en général, plus le cas. En effet, cette technique se révèle peu pratique lorsque plusieurs ordinateurs en réseau sont accédés par des utilisateurs itinérants. Dans ce genre de configuration, il est courant de recourir à un service réseau permettant de récupérer les informations concernant les utilisateurs à partir d'un serveur centralisé. Plusieurs solutions existent actuellement (NIS, LDAP, etc.), mais elles fonctionnent toutes plus ou moins selon le même principe. De plus, même pour des machines isolées, le fichier /etc/passwd ne contient plus que les informations « publiques » sur les utilisateurs. Les informations utilisées pour l'authentification sont désormais stockées dans un autre fichier de configuration, qui n'est lisible que pour l'utilisateur root : le fichier /etc/shadow. La raison première de procéder ainsi est d'éviter que des utilisateurs malicieux puissent « casser » les mots de passe.

Note : Les mots de passe n'ont jamais été stockés en clair dans le fichier passwd. Le mécanisme d'authentification repose en effet sur une fonction à sens unique générant une empreinte. Les fonctions de ce type ne disposent pas de fonction inverse, il n'est donc virtuellement pas possible de retrouver un mot de passe à partir de son empreinte. Lorsqu'un utilisateur saisit son mot de passe,

l'empreinte est recalculée de la même manière que lorsqu'il l'a défini initialement, et c'est cette empreinte qui est comparée avec celle qui se trouve dans le fichier /etc/passwd ou le fichier /etc/shadow. Ainsi, il est nécessaire de connaître le mot de passe en clair pour authentifier l'utilisateur, mais à aucun moment ce mot de passe n'est stocké sur le disque dur.

Cela dit, même si la récupération des mots de passe est quasiment impossible, la connaissance des empreintes des mots de passe peut être d'une aide précieuse. Un intrus potentiel peut essayer de calculer les empreintes de tous les mots de passe possibles et imaginables à l'aide d'un dictionnaire ou de mots de passe probablement choisis par les utilisateurs peu inventifs, et comparer le résultat avec ce qui se trouve dans le fichier de mot de passe. S'il y a une correspondance, l'intrus pourra pénétrer le système et tenter d'utiliser d'autres failles pour acquérir les droits de l'utilisateur root. Cette technique, dite attaque du dictionnnaire, est tout à fait réalisable, d'une part parce que la puissance des machines actuelles permet de calculer un nombre considérable d'empreintes à la seconde, et d'autre part parce que bon nombre de personnes utilisent des mots de passe triviaux (leur date de naissance, le nom de leur chien, etc.) facilement devinables et testables.

Les systèmes Unix utilisent deux techniques pour pallier ces problèmes. La première est de ne pas calculer l'empreinte du mot de passe tel qu'il est fourni par l'utilisateur, mais de la calculer avec une donnée générée aléatoirement et stockée dans le fichier de mot de passe. Cette donnée, que l'on appelle classiquement « sel », jour le rôle de vecteur d'initialisation de l'algorithme de génération d'empreinte. Le sel n'est jamais le même pour deux utilisateurs, ce qui fait que deux utilisateurs qui auraient le même mot de passe n'auraient toutefoirs pas la même empreinte. Cela complique la tâche des attaquants qui utilisent la force brute, car ils doivent ainsi recalculer les empreintes pour chaque utilisateur.

La deuxième technique utilisée est tout simplement de ne pas laisser le fichier des empreintes accessible à tout le monde. C'est pour cela que le fichier de configuration /etc/shadow a été introduit. Étant lisible uniquement par l'utilisateur root, un pirate ne peut pas récupérer les empreintes de mots de passe pour les comparer avec des empreintes de mots de passe précalculées. Il doit donc essayer les mots de passe un à un, ce qui peu en décourager plus d'un, surtout si le système se bloque pendant un certain temps après quelques tentatives infructueuses...

La technique d'authentification par mot de passe peut paraître relativement primitive, à l'heure où les cartes à puce sont légions et où l'on commence à voir apparaître des scanners d'empreintes digitales. En fait, c'est une bonne solution, mais qui ne saurait en aucun cas être exhaustive en raison des problèmes mentionnés ci-dessus. L'idéal est donc d'utiliser plusieurs systèmes d'authentification en série, ce qui laisse libre cours à un grand nombre de possibilités.

Il est évident que la technique des mots de passe traditionnellement utilisée sur les systèmes Unix n'évoluera pas aisément vers de nouveaux mécanismes d'authentification. C'est pour cela que les programmes devant réaliser une

opération d'authentification ou de gestion des utilisateurs de manière générale ont été modifiés pour utiliser la bibliothèque PAM (abréviation de l'anglais « Pluggable Authentification Modules »). Cette bibliothèque permet de réaliser les opérations d'identification et d'authentification de manière externe aux programmes qui l'utilisent, et se base pour ces opérations des fichiers de configuration et des modules dynamiquement chargeables. Ainsi, grâce à la bibliothèque PAM, l'administrateur peut définir avec précision les différentes opérations réalisées pour identifier et authentifier les utilisateurs, sans avoir à modifier les programmes qui ont besoin de ces fonctionnalités. De plus, il est possible d'ajouter de nouveaux modules au fur et à mesure que les besoins évoluent, et de les intégrer simplement en modifiant les fichiers de configuration.

De nos jours, la plupart des distributions utilisent la bibliothèque PAM. Bien entendu, il existe des modules qui permettent de réaliser les opérations d'identification et d'authentification Unix classiques, et ce sont ces modules qui sont utilisés par défaut. Nous verrons le format des fichiers de configuration de la bibliothèque PAM plus en détail dans les sections suivantes.

Note : Les mécanismes décrits ici ne sont sûrs que dans le cadre d'une connexion sur un terminal local. Cela dit, il faut bien prendre conscience que la plupart des applications réseau sont *de véritables passoires* ! En effet, ils utilisent des protocoles qui transmettent les mots de passe en clair sur le réseau, ce qui implique que n'importe quel pirate peut les capter en moins de temps qu'il n'en faut pour le dire. Les protocoles applicatifs suivants sont réputés pour être non sûrs et ne devront donc *JAMAIS* être utilisés sur un réseau non sûr (et donc, à plus forte raison, sur Internet) :

- TELNET, qui permet d'effectuer des connexions à distance :
- FTP, qui permet de transférer et de récuperer des fichiers sur une machine distante ;
- POP3, qui permet de consulter son mail ;
- SMTP, qui permet d'envoyer des mails à un serveur de messagerie ;
- X, qui permet aux applications graphiques d'afficher leurs fenêtres sur un terminal X.

  Cette liste n'est pas exhaustive mais regroupe déjà les protocoles réseau des applications les plus utilisées.

La sécurisation de ces protocoles ne peut se faire qu'en les encapsulant dans un autre protocole utilisant un canal de communication chiffré. L'un des outils les plus courant pour cela est sans doute ssh.

2. Création et suppression des utilisateurs

La création d'un nouvel utilisateur est une opération extrêmement facile. Il suffit simplement de lui créer un répertoire personnel dans le répertoire /home/ et de le définir dans le fichier de configuration /etc/passwd. Si votre système utilise les shadow passwords, ce qui est probable, il faut également définir cet utilisateur dans le fichier de configuration /etc/shadow. De plus, il faut ajouter cet utilisateur dans au moins un groupe d'utilisateurs dans le fichier /etc/group.

Le fichier de configuration /etc/passwd est constitué de plusieurs lignes, à raison d'une ligne par utilisateur. Chaque ligne est constituée de plusieurs champs, séparés par deux points (caractère ':'). Ces champs contiennent respectivement le login de l'utilisateur, l'empreinte de son mot de passe, son identifiant numérique, l'identifiant numérique de son groupe principal, son nom complet ou un commentaire, le chemin de son répertoire personnel et le chemin sur son interpréteur de commandes favori. Si le champ du mot de passe contient un astérisque, le compte est désactivé. S'il est vide, le mot de passe est stocké dans le fichier /etc/shadow.

Le fichier de configuration /etc/shadow a une syntaxe similaire à celle de /etc/passwd, mais contient les champs suivants : le login de l'utilisateur, l'empreinte de son mot de passe, le nombre de jours depuis que le mot de passe a été défini (comptés à partir du premier janvier 1970), le nombre de jours après cette date à attendre avant que le mot de passe puisse être changé, le nombre de jours au delà duquel le mot de passe doit obligatoirement être changé, le nombre de jours avant la date d'expiration de son mot de passe pendant lesquels l'utilisateur doit être averti que son mot de passe va expirer, le nombre de jours à attendre avant de désactiver le compte après l'expiration du mot de passe, et le nombre de jours depuis que le compte est désactivé, comptés depuis le premier janvier 1970. Il est possible de supprimer l'obligation pour les utilisateurs de changer régulièrement de mot de passe en donnant un nombre de jours minimum supérieur au nombre de jours maximum avant le changement de mot de passe.

Enfin, le fichier de configuration /etc/group, dans lequel les groupes d'utilisateurs sont définis, ne dispose que des champs suivants : le nom du groupe, son mot de passe (cette fonctionnalité n'est plus utilisée), son identifiant numérique, et la liste des utilisateurs qui y appartiennent, séparés par des virgules.

Bien entendu, tous ces champs ne doivent pas être modifiés à la main. La commande **useradd** permet de définir un nouvel utilisateur simplement. Cette commande suit la syntaxe suivante :

```
useradd [-c commentaire] [-d répertoire] [-e expiration] [-f inactivité] \
        [-g groupe] [-G groupes] [-m [-k modèle]] [-p passe]
           [-s shell] [-u uid [-o]] login
```

Comme vous pouvez le constater, cette commande prend en paramètre le login de l'utilisateur, c'est-à-dire le nom qu'il devra utiliser pour s'identifier sur le système, et un certain nombre d'options complémentaires. Ces options sont récapitulées dans le tableau suivant :

Option	Signification
-c	Permet de définir le champ commentaire du fichier de mot de passe.
-d	Permet de fixer le répertoire personnel de l'utilisateur.
-e	Permet de fixer la date d'expiration du compte. Cette date doit être spécifiée au format AAAA-MM-JJ.

-f	Permet de définir le nombre de jours avant que le compte ne soit désactivé une fois que le mot de passe est expiré. La valeur -1 permet de ne jamais désactiver le compte.
-g	Permet de définir le groupe principal auquel l'utilisateur appartient. Il s'agit souvent du groupe users.
-G	Permet de donner la liste des autres groupes auxquels l'utilisateur appartient. Cette liste est constituée des noms de chacun des groupes, séparés par des virgules.
-m	Permet de forcer la création du répertoire personnel de l'utilisateur. Les fichiers du modèle de répertoire personnel stockés dans le répertoire /etc/skel/ sont automatiquement copiés dans le nouveau répertoire. Si ces fichiers doivent être copiés à partir d'un autre répertoire, il faut spécifier celui-ci à l'aide de l'option -k
-p	Permet de donner le mot de passe initial du compte. Cette option ne doit jamais être utilisée, car le mot de passe apparaît dans la ligne de commande du processus et peut être lue par un utilisateur mal intentionné. On fixera donc toujours le mot de passe initial de l'utilisateur à l'aide de la commande **passwd**.
-s	Permet de spécifier le shell par défaut utilisé par l'utilisateur.
-u	Permet de spécifier l'UID de l'utilisateur. Cette valeur doit être unique en général, cependant, il est possible de forcer l'utilisation d'un même UID pour plusieurs utilisateurs à l'aide de l'option -o. Cela permet de créer un deuxième compte pour un utilisateur déjà existant.

L'ajout d'un groupe se fait avec la commande **groupadd**, qui suit la syntaxe suivante, beaucoup plus simple que celle de **useradd** :

groupadd [-g GID [-o]] nom

où nom est le nom du groupe et GID son numéro. Il n'est normalement pas possible de définir un groupe avec un identifiant numérique déjà attribué à un autre groupe, sauf si l'on utilise l'option -o.

De la même manière, la suppression d'un utilisateur peut se faire manuellement en effaçant son répertoire personnel et en supprimant les lignes qui le concernent dans les fichiers /etc/passwd, /etc/shadow et /etc/group. Il est également possible d'utiliser la commande **userdel**. Cette commande utiliser la syntaxe suivante :

userdel [-r] login

où login est le nom de l'utilisateur. L'option -r permet de demander à **userdel** d'effacer récursivement le répertoire personnel de l'utilisateur, ce qu'elle ne fait pas par défaut. Il existe également une commande **groupdel** pour supprimer un groupe d'utilisateurs (cette commande supprime le groupe seulement, pas les utilisateurs qui y appartiennent !).

Note : Comme pour la plupart des autres opérations d'administration système, il

est fortement probable que l'outil de configuration fourni avec votre distribution dispose de toutes les fonctionnalités nécessaires à l'ajout et à la suppression des utilisateurs. Il est recommandé d'utiliser cet outil, car il peut effectuer des opérations d'administration complémentaires que les outils standards n'effectuent pas forcément, comme la définition des comptes mail locaux par exemple.

3. Description de la bibliothèque PAM

La bibliothèque PAM permet de centraliser toutes les opérations relatives à l'identification et à l'authentification des utilisateurs. Ces tâches sont en effet déportées dans des modules spécialisés qui peuvent être chargés dynamiquement dans les programmes qui en ont besoin, en fonction de paramètres définis dans des fichiers de configuration. Le comportement des applications peut donc être parfaitement défini simplement en éditant ces fichiers.

La bibliothèque PAM permet une très grande souplesse dans l'administration des programmes ayant trait à la sécurité du système et devant réaliser des opérations privilégiées. Il existe déjà un grand nombre de modules, capables de réaliser une multitude de tâches diverses et variées, et que l'on peut combiner à loisir pour définir le comportement de toutes les applications utilisant la bibliothèque PAM. Il est hors de question de décrire chacun de ces modules ici, ni même de donner la configuration des programmes qui utilisent PAM. Cependant, nous allons voir les principes généraux permettant de comprendre comment les modules de la bibliothèque sont utilisés.

Initialement, toute la configuration de PAM se faisait dans le fichier de configuration /etc/pam.conf. Ce fichier contenait donc la définition du comportement de chaque programme utilisant la bibliothèque PAM, ce qui n'était pas très pratique pour l'administration et pour les mises à jour. Les informations de chaque programme ont donc été séparées en plusieurs fichiers distincts, à raison d'un fichier par application, tous stockés dans le répertoire /etc/pam.d/. Il est fort probable que votre distribution utilise cette solution, aussi le format du fichier /etc/pam.conf ne sera-t-il pas décrit.

Certains modules utilisent des fichiers de configuration pour déterminer la manière dont ils doivent se comporter. Ces fichiers de configuration sont tous stockés dans le répertoire /etc/security/. Les modules de PAM eux-mêmes sont, quant à eux, stockés dans le répertoire /lib/security/.

Le principe de fonctionnement est le suivant. Lorsqu'un programme désire réaliser une opération relative à l'authentification d'un utilisateur, il s'adresse à la bibliothèque PAM pour effectuer cette opération. La bibliothèque recherche dans le répertoire /etc/pam.d/ le fichier de configuration correspondant à cette application (il porte généralement le nom de l'application elle-même), puis détermine les modules qui doivent être chargés dynamiquement dans l'application. Si le fichier de configuration d'une application ne peut pas être trouvé, le fichier de configuration /etc/pam.d/other est utilisé, et la politique de sécurité par défaut qui y est définie est utilisée. Quel que soit le fichier de

configuration utilisé, chaque module est utilisé en fonction des paramètres qui y sont stockés. Les modules peuvent, s'ils en ont besoin, utiliser leurs propres fichiers de configuration, qui se trouvent dans le répertoire /etc/security/. Ces fichiers portent généralement le nom du module avec l'extension .conf. Par exemple, le fichier de configuration du module limits, qui prend en charge les limites d'utilisation des ressources système pour chaque utilisateur, est le fichier /etc/security/limits.conf.

Les fichiers de configuration des applications sont constitués de lignes définissant les différents modules qui doivent être chargés, le contexte dans lequel ils sont chargés, et comment doit se comporter l'application en fonction du résultat de l'exécution des opérations réalisées par ces modules. L'ordre des lignes est important, puisqu'elles sont analysées les unes après les autres. Chaque ligne est constituée de trois colonnes. La première colonne indique le cadre d'utilisation du module. La deuxième colonne indique le comportement que doit adopter la bibliothèque PAM en fonction du résultat renvoyé par le module après son exécution. Enfin, la troisième colonne donne le chemin d'accès complet au module, éventuellement suivi des options qui doivent lui être communiquées pour son exécution.

Les modules peuvent être utilisés dans l'un des contextes suivants :

- auth, qui est le contexte utilisé par les programmes qui demandent l'authentification de l'identité de utilisateur ;
- account, qui est le contexte utilisé par les programmes qui désirent obtenir des informations sur l'utilisateur (répertoire personnel, shell, etc.)
- password, qui est le contexte utilisé par les applications qui cherchent à revalider l'authentification de l'utilisateur. Les programmes comme **passwd** par exemple, qui demandent le mot de passe de l'utilisateur avant d'en fixer un nouveau, sont susceptibles d'utiliser ce contexte ;
- session, qui est le contexte utilisé par les applications lorsqu'elles effectuent les opérations de gestion d'ouverture et de fermeture de session. Ce contexte peut être utilisé pour réaliser des tâches administratives, comme l'enregistrement de l'utilisateur dans la liste des utilisateurs connectés ou le chargement des préférences personnelles de l'utilisateur par exemple.

Une même application peut définir plusieurs jeux de règles pour plusieurs contextes différents, et certains modules peuvent être utilisés dans plusieurs contextes différents également. Cependant, lorsqu'une application réalise une demande à la bibliothèque PAM, cette demande n'est exécutée que dans le cadre d'un contexte bien défini. Attention cependant, une même application peut effectuer plusieurs opérations successivement dans des contextes différents. Par exemple, le programme **login** peut utiliser le contexte auth pour valider l'identité de l'utilisateur, puis le contexte account pour déterminer le répertoire personnel de l'utilisateur, et enfin le contexte session pour enregistrer l'utilisateur dans le journal des utilisateurs connectés.

Le comportement de la bibliothèque PAM en fonction du résultat de l'exécution des modules dépend de ce qui est spécifié dans la deuxième colonne des lignes de ces modules. Les options qui peuvent y être utilisées sont les suivantes :

- required, qui permet d'indiquer que le succès de l'opération effectuée par le module est nécessaire pour que l'opération effectuée dans le contexte spécifié dans la première colonne de la ligne réussisse. Un échec sur cette ligne ne provoque pas l'arrêt de l'analyse du fichier de configuration, ce qui permet d'appeler d'autres modules, par exemple pour générer des traces dans les fichiers de traces du système. Cependant, quel que soit le comportement des modules suivants, l'opération demandée par le programme appelant échouera ;
- requisite, qui permet d'indiquer que le succès de l'opération effectuée par le module est nécessaire, faute de quoi l'opération réalisée par le programme appelant échoue immédiatement. Les autres modules ne sont donc pas chargés en cas d'échec, contrairement à ce qui se passe avec l'option required ;
- sufficient, qui permet d'indiquer que le succès de l'exécution du module garantira le succès de l'opération demandée par le programme appelant. Les modules suivants seront chargés malgré tout après l'appel de ce module, sauf s'il s'agit de modules de type required ;
- optional, qui permet d'indiquer que le résultat du module ne doit être pris en compte que si aucun autre module ne peut déterminer si l'opération est valide ou non. Dans le cas contraire, le module est chargé, mais son résultat est ignoré.

À titre d'exemple, nous pouvons présenter deux implémentations possibles du fichier de configuration par défaut /etc/pam.d/other. Une configuration extrêmement sûre interdira l'accès à toute fonctionnalité fournie par un programme n'ayant pas de fichier de configuration. Dans ce cas de configuration, on utilisera un fichier comme celui-ci :

```
auth       required   /lib/security/pam_warn.so
auth       required   /lib/security/pam_deny.so
account    required   /lib/security/pam_deny.so
password   required   /lib/security/pam_warn.so
password   required   /lib/security/pam_deny.so
session    required   /lib/security/pam_deny.so
```

Nous voyons que toutes les opérations de type authentification ou de revalidation de l'identité sont d'abord tracées dans les fichiers de traces du système, puis déclarées comme interdites. Une telle configuration peut être un peu trop restrictive car, en cas d'erreur dans un fichier de configuration d'une application, l'accès à cette application peut être interdit systématiquement. Une autre configuration, plus permissive, cherchera à utiliser les mécanismes d'identification et d'authentification Unix classiques. Cela se fait avec les modules pam_unix_auth, pam_unix_acct, pam_unix_passwd et pam_unix_session :

```
auth       required   /lib/security/pam_unix_auth.so
account    required   /lib/security/pam_unix_acct.so
password   required   /lib/security/pam_unix_passwd.so
session    required   /lib/security/pam_unix_session.so
```

Les autres fichiers de configuration ne seront pas décrits ici, car ils dépendent de chaque application et de chaque distribution. Vous pouvez consulter ceux qui sont installés sur votre système si vous désirez en savoir plus.

Nous ne décrirons pas non plus la syntaxe des fichiers de configuration des différents modules, car cela dépasserait largement le cadre de ce document. Cela dit, la plupart de ces fichiers sont parfaitement commentés et leur modification ne devrait pas poser de problème particulier. À titre d'exemple, on peut présenter le cas du module de gestion des limites des ressources consommées par les utilisateurs. Si l'on désire restreindre le nombre de processus que les utilisateurs peuvent lancer, on pourra ajouter la ligne suivante dans le fichier /etc/security/limits.conf :

```
@users hard     nproc    256
```

Il faudra également demander le chargement de ce module dans les fichiers de configuration des applications fournissant un accès au système. Pour cela, on ajoutera une ligne telle que celle-ci à la fin de leur fichier de configuration :

```
session    required    /lib/security/pam_limits.so
```

Il est également possible de limiter le nombre de login de chaque utilisateur, la quantité de mémoire qu'il peut consommer, la durée de connexion autorisée, la taille maximum des fichiers qu'il peut manipuler, etc.