PAM Pluggable Authentication Modules

- Objet : Apprendre qu'est-ce que PAM(Pluggable Authentication Modules) et comment le configurer
- Niveau requis :
  avisé
- Commentaires : *La modification de PAM se fait sur des machines de type serveur ou pour des usagers avisés qui tiennent à la sécurité.*
  Les options par défaut de Debian conviennent dans la quasi-totalité des cas à n'importe quel usager lambda.
- Suivi :
  à-tester
  - Création par greenmerlin 26/01/2016
  - Testé par <...> le <...>
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]

Introduction

Pam est un système de bibliothèques qui **garantit** l'utilisation des commandes d'administration en récupérant les **processus d'authentification des services et applications** du système.
Peu connu il n'est pas moins utilisé sur beaucoup de distributions, c'est une couche massive de sécurité.

il est grandement recommandé de faire des essais de configuration PAM sur une machine virtuelle avant de toucher à la machine réelle.

- De Philou92 sur le forum

Découverte

Pam à configurer : Le concept est plutôt simple, un fichier par programme ou daemon.
Bien entendu les développeurs de Jessie nous ont tout bien configuré comme il faut :

```
ls /etc/pam.d/
```

pour un petit aperçu. les plus importants sont :

- /etc/pam.d/common-auth
- /etc/pam.d/common-account
- /etc/pam.d/common-password
- /etc/pam.d/common-session

Pour savoir si un programme peut-être utilisé dans PAM :

```
ldd /bin/vers_le_programme | grep libpam
```

Exemple

common-password

```
...
# here are the per-package modules (the "Primary" block)
password        [success=1 default=ignore]      pam_unix.so obscure
sha512
# here's the fallback if no module succeeds
password        requisite                       pam_deny.so
# prime the stack with a positive return value if there isn't one
already;
# this avoids us returning an error just because nothing sets a success
code
# since the modules above will each just jump around
password        required                        pam_permit.so
# and here are more per-package modules (the "Additional" block)
# end of pam-auth-update config
...
```

Ici la première chose à noter c'est que la syntaxe employée est toujours de type :

NOM	COMMENTAIRES
Module type	Quel module de sécurité à utiliser (qu'est-ce que PAM va gérer) ?
Control	que fait un module en cas d'échec
Module path	le chemin du module
Arguments	les arguments du module

Chronologiquement une action PAM se déroule comme suit :

1. L'utilisateur entre **son nom de connexion** et **mot de passe**.
2. Le programme login traite la demande puis délègue l'authentification à PAM.
3. PAM lit le fichier `/etc/pam.d/login`, exécute les directives et renvoie au programme.

Décortiquons la ligne :

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

- Ici le module est **pam_unix.so**, ce qui signifie que pam va vérifier la concordance avec les fichiers `/etc/passwd` et `/etc/shadow`
- Pour `Success = 1` et `default = ignore` c'est un peu plus compliqué j'aborderai ça plus loin.
- Les Options `obscure` et `sha512` signifient que le module va respectivement tester si votre niveau de complexité de mot de passe est suffisant et que le mot de passe sera crypté avec

l'algorithme **SHA512**.

Par défaut l'option `obscure` teste les points suivants sur le mot de passe entré :

- Palindrome.
- Changement de la casse uniquement (ex :password, PAsSworD).
- Si il est trop similaire au dernier.
- Si il est vraiment trop simple (ex : 0000).
- Si il est retourné (ex : user:root;passwd:toor).

Comprendre PAM

Pour bien comprendre les mécanismes de PAM nous allons reprendre notre tableau, mais avant il est nécessaire d'aborder un autre point concernant les fichiers de configuration :

```
ls /etc/pam.d/*
```

L'empilement des modules

Prenons l'exemple suivant :

```
auth       required     /lib/security/pam_securetty.so
auth       required     /lib/security/pam_env.so
auth       sufficient   /lib/security/pam_rootok.so
auth       required     /lib/security/pam_unix.so
```

Voila comment PAM va les interpréter :

Le module `pam_securetty` va vérifier son fichier de configuration dans `/etc/securetty`, afin de vérifier que le terminal utilisé est listé dans le fichier.
Si cela n'est pas le cas, les connexions en tant qu'utilisateur root ne seront pas permises.

Vu que le statut est à `required`, le module appellera les trois autres modules de la pile mais, même si tous les autres sont ok, la connexion échouera.

Notons que si le module avait le statut `requisite`, l'opération aurait été soldée par un échec et aucun des autres modules n'aurait été invoqué, peu importe leur statut.

- Le module `pam_env` va fixer les variables d'environnement d'après le fichier `/etc/security/pam_env.conf`.
- Le module `pam_rootok` va vérifier si l'utilisateur est root.
  - Si cela échoue, l'opération peut réussir tout de même si le module `pam_unix` authentifie l'utilisateur.
- Si “pam_rootok” authentifie l'utilisateur, `pam_unix` ne sera pas invoqué.
- Le module “pam_unix” essaiera alors d'authentifier l'utilisateur en utilisant `les appels système getpw*`.

1. Si `pam_unix` échoue et que `pam_rootok` avait échoué, l'opération échouera.
2. Si `pam_rootok` échoue mais que `pam_unix`" est correct, l'opération sera exécutée.

On reprend notre tableau pour la suite

NOM	COMMENTAIRES
Module type	quel module de sécurité à utiliser (qu'est-ce que PAM va gérer) ?
Control	que fait un module en cas d'échec ?
Module path	le chemin du module.
Arguments	les arguments du module.

Le module-type

Quatre structures de gestion possibles :

- authentification : ceci permet à l'utilisateur de s'authentifier et de définir les droits du compte. Les modules utilisant ce système demandent en général un mot de passe.
- gestion de compte : permet la gestion des comptes utilisateurs.
  - Par exemple, des restrictions peuvent être définies selon les horaires ou la charge du serveur.
- gestion de session : Ce sont les tâches à effectuer en début et fin de chaque session.
  - Par exemple, démarrer un agent ssh, monter des disques cryptés, etc ...
- gestion des mots de passe : mise à jour du jeton d'authentification du compte d'un utilisateur, soit parce qu'il a expiré, soit parce que celui-ci désire le modifier.

Control

Que doit faire la librairie PAM en cas de réussite ou d'échec du module ?
Plusieurs possibilités :

* **required** : si un module `required` retourne un statut qui n'est pas `success`, l'opération échouera mais seulement après que tous les modules en dessous soient invoqués.
La réussite d'au moins un des modules `required` est nécessaire.

- **requisite** : si un module `requisite` échoue, l'application en est tout de suite informée et aucun autre module n'est invoqué.
  La réussite de tous les modules `requisite` est nécessaire.
- **sufficient** : la réussite d'un seul module `sufficient` est suffisant.
  Tous les modules `sufficient` qui suivent ne seront pas invoqués.

Il faut noter, toutefois, que si un module `required` échoue avant un `sufficient`, l'opération échouera ignorant les autres modules `sufficient`.

- **optional** : la réussite d'au moins un des modules `optional` est nécessaire si aucun autre n'a réussi.

Une syntaxe un peu plus complète peut-être employée sous la forme de : `value=action pairs`

Exemple

(Déjà abordé lors de la présentation.)

```
[success=1 default=ignore]
```

- **success=1** : Si le module PAM renvoie `success`, PAM va sauter 1 ligne dans le fichier de config (2 pour 2 lignes,etc...)
- **default=ignore** : Le statut de retour du module, n'influencera pas le reste des règles dans le fichier de config

Cas concret N°1

Pour tester le 1er cas concret `if` faut avoir un serveur SSH fonctionnel

On va faire un exemple ça vaut mieux que les grands discours il paraît.

Modifions le comportement par defaut d'un programme en utilisant d'autres modules

Retrouvez la liste des modules disponibles par défaut sur debian ici -> `/lib/i386-linux-gnu/security`

Ici on va utiliser le module **pam-dbus**, sur le programme `sshd` (OPEN-SSH).

On se logue sous X en graphique (*dbus* fonctionne avec le mode graphique).
On installe le module pam (oui il n'est pas installé par défaut :

```
apt install libpam-dbus
```

On édite le fichier /etc/pam.d/sshd

```
nano /etc/pam.d/sshd
```

Et on remplace la ligne suivante :

```
@include common-auth
```

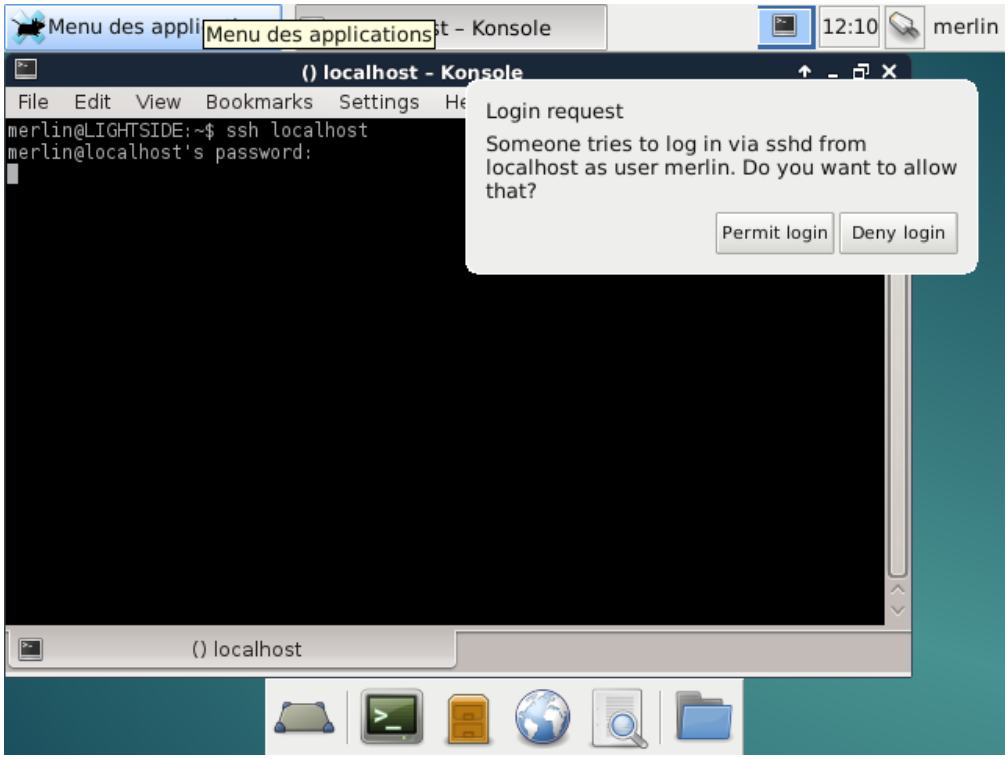
par

```
auth required pam_dbus.so
```

Test

```
ssh localhost
```

Résultat :



Ici le module pam-dbus va toujours envoyer un message “dbus” à tous les utilisateurs connectés localement pour autoriser ou non **la connexion ssh à votre machine**.

Décortiquons maintenant l'option :

```
auth required pam_dbus.so
```

- auth : Utiliser un module de type Authentifier et Autoriser
- required : Le module renvoie une erreur et rend la main à la prochaine option PAM
- pam-dbus : Le nom du module que j'utilise

[1)]

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !