PAM

Pluggable Authentication Modules

Loris Renggli

EPFL SB SB-IT

2010-05-27

Introduction

PAM: c'est quoi ?

- "user-authentication API"
- "UNIX authentication framework"
- définit un mécanisme d'authentification "standard"
- Linux-PAM, OpenPAM

Les origines

- Sun, OSF RFC en 1995
- CDE (Common Desktop Environment)
- Red Hat 1996
- puis les autres (AIX FreeBSD NetBSD HP-UX MacOSX ...)

PAM: pourquoi ?

Sans PAM:

- chaque application utilise sa propre authentification (login, ftp, telnet, ...)
- les modifications de l'authentification nécessitent une recompilation

PAM: pourquoi ?

Avec PAM:

- unification (standardisation) de l'authentification
- pas de recompilation
- configuration individuelle des applications possible
- changement de configuration sans redémarrage des services

Authentification

Une authentification, c'est:

- identifier
- authentifier
- autoriser

Authentification PAM

Une authentification PAM, c'est:

- identifier, authentifier: auth
- autoriser: account, session

Et en plus:

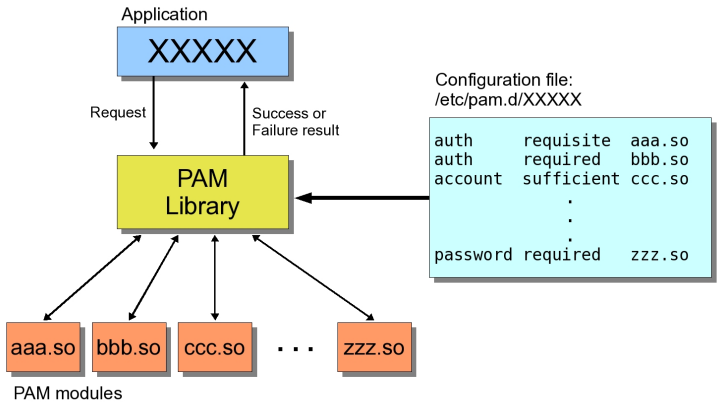
- changement de mot de passe: password

Authentification PAM

PAM définit quatre types de tâches pour contrôler l'authentification.

- identifier, authentifier: auth
  typiquement: nom d'utilisateur et mot de passe
- autoriser: account, session
  la tâche account est utilisé pour contrôler la validité du compte
  (mot de passe expiré? restriction d'horaire? etc)
  la tâche session définit les actions à effectuer en début de session
  (environnement, créer/monter home directory, etc)
- changement de mot de passe: password

Chaque module PAM assure au moins une des tâches `auth`, `account`, `session`, `password`. Certains modules implémentent toutes les tâches (comme `pam_unix`).



Configuration

OpenSuSE 11.1

Configuration dans `/etc/pam.d`, un fichier par application.

```
$ ls /etc/pam.d
atd                 kcheckpass      sshd
chage               login           su
chfn                login.old       su-l
chsh                other           sudo
common-account      passwd          useradd
common-auth         polkit          vmtoolsd
common-password     ppp             vmware-guestd
common-session      rpasswd         xdm
crond               shadow          xdm-np
cups                smtp
```

Par exemple, `sshd` contient:

```
#%PAM-1.0
auth       requisite      pam_nologin.so
auth       include        common-auth
account    include        common-account
password   include        common-password
session    required       pam_loginuid.so
session    include        common-session
```

et `common-auth`:

```
auth    required        pam_env.so
auth    required        pam_unix2.so
```

La pile `auth` est donc:

```
auth    requisite       pam_nologin.so
auth    required        pam_env.so
auth    required        pam_unix2.so
```


Chaque ligne est de la forme:

```
type  control  module-path  module-arguments
```

Les valeurs possibles pour `control` sont:

```
requisite    required    sufficient    optional
```

La suite des directives pour un `type` donné forme une pile (stack). Cette pile est évaluée et son résultat final (réussite ou échec) détermine le résultat de l'authentification.

Contrôle

- `requisite` module nécessaire; un échec provoque l'arrêt immédiat du processus d'authentification
- `required` module nécessaire; en cas d'échec les modules restants de la pile sont exécutés malgré tout
- `sufficient` authentification réussie (sauf si échec précédent)
- `optional` succès ou échec pris en compte uniquement si c'est l'unique module de la pile

Exemples

other

Si un service utilisant PAM n'est pas configuré, alors il utilise la configuration par défaut `other`.

```
auth       required       pam_deny.so
account    required       pam_deny.so
password   required       pam_deny.so
session    required       pam_deny.so
```


other

On peut rajouter des messages dans syslog.

```
auth       required       pam_deny.so
auth       required       pam_warn.so
account    required       pam_deny.so
password   required       pam_deny.so
password   required       pam_warn.so
session    required       pam_deny.so
```


sshd

La pile `sshd` (inclusions faites)

```
auth       requisite  pam_nologin.so
auth       required   pam_env.so
auth       required   pam_unix2.so
account    required   pam_unix2.so
password   requisite  pam_pwcheck.so nullok cracklib
password   required   pam_unix2.so use_authtok nullok
session    required   pam_loginuid.so
session    required   pam_limits.so
session    required   pam_unix2.so
session    optional   pam_umask.so
```


sshd restreint

Pour configurer `sshd` de façon à ce que seules certaines connexions distantes soient autorisées, il suffit d'ajouter une ligne de type `account`.

```
account    required   pam_unix2.so
account    required   pam_access.so
```

La configuration se trouve dans le fichier `access.conf` du directory `/etc/security`; par exemple,

```
+ : ALL : 128.178.
+ : renggli : ALL
- : ALL : ALL
```

sshd groupes autorisés

Le module pam_listfile permet d'appliquer divers critères. On peut par exemple n'autoriser que le login pour les utilisateurs d'un groupe donné ainsi:

```
auth required pam_listfile.so item=group sense=allow \
  file=/etc/security/groups.allow onerr=fail
```

Le fichier groups.allow contient la liste des groupes autorisés.

```
root
sb-it
```

Attention: si root n'est pas listé, alors on ne peut plus faire de login root.

sshd groupes non autorisés

Similaire au cas précédent:

```
auth required pam_listfile.so item=group sense=deny \
  file=/etc/security/groups.deny onerr=succeed
```

Le fichier groups.deny contient la liste des groupes non autorisés.

```
www
users
```

On peut combiner autorisation et interdiction.

Modules courants

Il y a une quarantaine de modules dans la distribution, et de nombreux modules indépendants. Voici une sélection des plus courants.

- pam_deny échec inconditionnel
- pam_permit succès inconditionnel
- pam_env initialisation de l'environnement
- pam_lastlog enregistre le login
- pam_limits limite des processus
- pam_loginuid change l'uid du processus de login
- pam_mail avertissement de mail au login

- pam_make lance make (p.ex. NIS)
- pam_nologin login interdit (fichier /etc/nologin)
- pam_pwcheck test du mot de passe (crack)
- pam_rootok utile pour que root soit autorisé
- pam_umask définir un umask du processus
- pam_unix2 authentification unix "standard"
- pam_warn message dans syslog
- pam_xauth transporte clefs xauth (p.ex. su)

Plus complexe: RHEL4

Configuration avec LDAP; service `login`:

```
#%PAM-1.0
auth       required     pam_securetty.so
auth       required     pam_stack.so service=system-auth
auth       required     pam_nologin.so

account    required     pam_stack.so service=system-auth

password   required     pam_stack.so service=system-auth

# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_stack.so service=system-auth
session    required     pam_loginuid.so
session    optional     pam_console.so
# pam_selinux.so open should be the last session rule
session    required     pam_selinux.so open
```


Fichier `system-auth`:

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        sufficient    pam_unix.so likeauth nullok
auth        sufficient    pam_ldap.so use_first_pass
auth        required      pam_deny.so

account     required      pam_unix.so broken_shadow
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 100 quiet
account     [default=bad success=ok user_unknown=ignore] pam_ldap.so
account     required      pam_permit.so

password    requisite     pam_cracklib.so retry=3
password    sufficient    pam_unix.so nullok use_authtok md5 shadow
password    sufficient    pam_ldap.so use_authtok
password    required      pam_deny.so

session     required      pam_limits.so
session     required      pam_unix.so
session     optional      pam_ldap.so
```


Contrôle complexe

A la place d'un mot clé, un contrôle peut prendre la forme:

`[valeur1=action1 valeur2=action2 ...]`

où `valeurN` est le code renvoyé par le module qui peut prendre 32 valeurs différentes (cf. la documentation), et `actionN` peut être:

`ignore, bad, die, ok, done, reset`

Ainsi, `required` est équivalent à

`[success=ok new_authtok_reqd=ok ignore=ignore default=bad]`

et `requisite` est équivalent à

`[success=ok new_authtok_reqd=ok ignore=ignore default=die]`

Donc dans le fichier précédent, le contrôle

`[default=bad success=ok user_unknown=ignore]`

est similaire à `required`, avec `user_unknown=ignore` en plus.


```
#define PAM_SUCCESS 0           /* Successful function return */
#define PAM_OPEN_ERR 1          /* dlopen() failure when dynamically */
                                /* loading a service module */
#define PAM_SYMBOL_ERR 2        /* Symbol not found */
#define PAM_SERVICE_ERR 3       /* Error in service module */
#define PAM_SYSTEM_ERR 4        /* System error */
#define PAM_BUF_ERR 5           /* Memory buffer error */
#define PAM_PERM_DENIED 6       /* Permission denied */
#define PAM_AUTH_ERR 7          /* Authentication failure */
#define PAM_CRED_INSUFFICIENT 8 /* Can not access authentication data */
                                /* due to insufficient credentials */
#define PAM_AUTHINFO_UNAVAIL 9  /* Underlying authentication service */
                                /* can not retrieve authentication */
                                /* information  */
#define PAM_USER_UNKNOWN 10     /* User not known to the underlying */
                                /* authenticaiton module */
#define PAM_MAXTRIES 11         /* An authentication service has */
                                /* maintained a retry count which has */
                                /* been reached.  No further retries */
                                /* should be attempted */
#define PAM_NEW_AUTHTOK_REQD 12 /* New authentication token required. */
                                /* This is normally returned if the */
                                /* machine security policies require */
                                /* that the password should be changed */
                                /* beccause the password is NULL or it */
                                /* has aged */
#define PAM_ACCT_EXPIRED 13     /* User account has expired */
#define PAM_SESSION_ERR 14      /* Can not make/remove an entry for */
                                /* the specified session */
```

```
#define PAM_CRED_UNAVAIL 15     /* Underlying authentication service */
                                /* can not retrieve user credentials */
                                /* unavailable */
#define PAM_CRED_EXPIRED 16     /* User credentials expired */
#define PAM_CRED_ERR 17         /* Failure setting user credentials */
#define PAM_NO_MODULE_DATA 18   /* No module specific data is present */
#define PAM_CONV_ERR 19         /* Conversation error */
#define PAM_AUTHTOK_ERR 20      /* Authentication token manipulation error */
#define PAM_AUTHTOK_RECOVERY_ERR 21 /* Authentication information */
                                    /* cannot be recovered */
#define PAM_AUTHTOK_LOCK_BUSY 22   /* Authentication token lock busy */
#define PAM_AUTHTOK_DISABLE_AGING 23 /* Authentication token aging disabled */
#define PAM_TRY_AGAIN 24        /* Preliminary check by password service */
#define PAM_IGNORE 25           /* Ignore underlying account module */
                                /* regardless of whether the control */
                                /* flag is required, optional, or sufficient */
#define PAM_ABORT 26            /* Critical error (?module fail now request) */
#define PAM_AUTHTOK_EXPIRED  27 /* user's authentication token has expired */
#define PAM_MODULE_UNKNOWN   28 /* module is not known */

#define PAM_BAD_ITEM         29 /* Bad item passed to pam_*_item() */
#define PAM_CONV_AGAIN       30 /* conversation function is event driven
                                   and data is not available yet */
#define PAM_INCOMPLETE       31 /* please call this function again to
                                   complete authentication stack. Before
                                   calling again, verify that conversation
                                   is completed */
```


Oops...

PAM est le système d'authentification, il faut donc être très précautionneux.

Les choses à ne pas faire:

- tout interdire: `auth required pam_deny.so`
- tout autoriser: `auth sufficient pam_permit.so`
- autoriser beaucoup plus que l'on croit:
  `auth required pam_env.so`
  `auth sufficient pam_unix.so`
  `auth sufficient pam_ldap.so`
  (il manque la dernière ligne avec `required pam_deny.so`)

Références

- La documentation fournie avec PAM (man pages, `/usr/share/doc`)
- `http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html`

Q/R