Gestion des profils utilisateurs

Dans le cadre d'un système multi-utilisateur, la capacité à gérer de manière fine et sécurisée chaque profil se pose de façon redondante. La réponse passe par le triptyque utilisateur - droit - application, ou plus exactement par la question : quels sont les droits globaux des utilisateurs et quels sont leurs droits vis-à-vis de chaque application ? On le voit nettement, la réponse dépasse le classique schéma Propriétaire/Groupe/Reste du monde vu au chapitre Gestion des droits utilisateurs.

1. ACL avec Apparmor

Gérer la sécurité des données revient à assurer leur confidentialité, leur intégrité et leur disponibilité au travers d'un contrôle d'accès, le plus souvent logiciel.

Ubuntu comporte une approche, au travers d'un logiciel nommé **Apparmor**, destinée à associer un profil de sécurité à chaque application. Il suit la norme POSIX 1003.1e (ajoutée à celle classique des permissions donnée par POSIX.1) et introduit la gestion des ACL (*Access Control List*) sous Linux.

Il s'agit donc de définir les droits de chaque utilisateur sur une application et donc d'en permettre une restriction. On passe d'une gestion de type DAC (*Discretionary Access Control*) à une gestion de type MAC (*Mandatory Access Control*) définie par l'administrateur système, elle-même englobée dans une approche de contrôle d'accès par rôle (RBAC ou *Role-Based Access Control*).

SELinux ou Apparmor ?

Cette mise en place sur Linux se traduit sur Linux par une bibliothèque de sécurité (framework) constituée de modules (LSM) et par le choix entre deux techniques :

- **SELinux**, développé par la NSA, l'agence de sécurité américaine, et implanté surtout sur les systèmes Red Hat, Fedora.
- **Apparmor**, développé par Novell et intégré sur OpenSuse (racheté par Novell) et les systèmes Debian, Ubuntu.

Le choix entre l'une ou l'autre des techniques mérite un développement plus long hors de propos dans cet ouvrage. Sachez simplement que SELinux sécurise l'ensemble du système avec une interdépendance lourde sur le système de fichiers alors que Apparmor s'intégre plus facilement grâce à son mécanisme de profils. La lourdeur de la première technique s'oppose à la lisibilité de la deuxième, mais Apparmor ne protège que quelques processus. Aussi, on peut dire que le SELinux est plus strict au sens de la sécurité.

Ubuntu depuis Hardy Heron fait le choix d'Apparmor. La documentation complète se situe sur le site de Novell : http://www.novell.com/documentation/apparmor/

2. Administration des profils

Apparmor (licence GPL) se trouve installé et chargé par défaut sur la distribution Ubuntu. Par contre, si certains profils proviennent automatiquement lors de l'installation de certains paquets, d'autres profils se trouvent dans un paquet supplémentaire :

```
aptitude install apparmor-profiles
```

Ces profils se chargent dans le répertoire `/etc/apparmor.d/`. Vous êtes toujours connecté en `root`.

a. Fonctionnement d'Apparmor

La commande `apparmor_status` permet de faire le point sur les profils existants qui viennent d'être installés :

```
root@duncan:/home/max# apparmor_status
apparmor module is loaded.
11 profiles are loaded.
11 profiles are in enforce mode.
   /sbin/dhclient3
   /usr/bin/evince
   /usr/bin/evince-previewer
```

```
   /usr/bin/evince-thumbnailer
   /usr/lib/NetworkManager/nm-dhcp-client.action
   /usr/lib/connman/scripts/dhclient-script
   /usr/lib/cups/backend/cups-pdf
   /usr/sbin/cupsd
   /usr/sbin/mysqld
   /usr/sbin/tcpdump
   /usr/share/gdm/guest-session/Xsession
0 profiles are in complain mode.
2 processes have profiles defined.
2 processes are in enforce mode :
   /usr/sbin/cupsd (1179)
   /usr/sbin/mysqld (996)
0 processes are in complain mode.
0root@duncan:/home/max# cat /sys/kernel/security/apparmor/profiles
/usr/sbin/tcpdump (enforce)
/usr/sbin/mysqld (enforce)
/usr/sbin/cupsd (enforce)
/usr/lib/cups/backend/cups-pdf (enforce)
/usr/bin/evince-thumbnailer (enforce)
/usr/bin/evince-previewer (enforce)
/usr/bin/evince (enforce)
/usr/lib/connman/scripts/dhclient-script (enforce)
/usr/lib/NetworkManager/nm-dhcp-client.action (enforce)
/sbin/dhclient3 (enforce)
/usr/share/gdm/guest-session/Xsession (enforce) processes are
unconfined but have a profile defined.
```

Le résultat de cette commande montre les deux modes de fonctionnement d'**Apparmor** :

- Le mode `complaining/learning` : réservé à l'apprentissage ou aux tests, on autorise la violation de profil avec enregistrement de l'événement.
- Le mode `enforced/confined` : le profil s'applique dans toute sa rigueur, on enregistre toute violation éventuelle.

Un profil ne s'applique qu'à un processus en fonctionnement.

Le résultat de la commande `apparmor_status` s'explique donc de la façon suivante :

- 12 profils se trouvent dans le répertoire `/etc/apparmor.d` et sont chargés,
- 11 sont en mode apprentissage et 1 en mode appliqué,
- Seuls deux processus sont en cours (`klogd` et `syslogd`) mais dont les profils sont définis (ils ne sont pas en *learning*) ne s'appliquent pas.

La vérification est facile :

```
ps aZ | grep syslogd
```

La commande d'interrogation sur les processus en cours et plus particulièrement sur le démon `syslogd` retourne :

```
unconfined      4548 tty1  S+       0:00 grep syslogd
```

Il faut bien comprendre qu'un processus peut être soit en mode `complain` contre `enforce` (suivant qu'il est ou non autorisé à suivre le profil), soit en mode `Learning` ou `confined` (suivant qu'un profil est ou non défini).

Une autre façon de suivre les profils consiste à visualiser le fichier `/sys/kernel/security/apparmor/profiles` (essayez par `cat`) :

```
root@duncan:/home/max# cat /sys/kernel/security/apparmor/profiles
/usr/sbin/tcpdump (enforce)
/usr/sbin/mysqld (enforce)
```

```
/usr/sbin/cupsd (enforce) #%PAM-1.0
auth    requisite       pam_nologin.so
auth    required        pam_env.so readenv=1
auth    required        pam_env.so readenv=1 envfile=/etc/default/locale
auth    sufficient      pam_succeed_if.so user ingroup nopasswdlogin
@include common-auth
auth    optional        pam_gnome_keyring.so
@include common-account
session [success=ok ignore=ignore module_unknown=ignore
default=bad] pam_selinux.so close
session required        pam_limits.so
@include common-session
session [success=ok ignore=ignore module_unknown=ignore
default=bad] pam_selinux.so open
session optional        pam_gnome_keyring.so auto_start
@include common-password
/usr/lib/cups/backend/cups-pdf (enforce)
/usr/bin/evince-thumbnailer (enforce)
/usr/bin/evince-previewer (enforce)
/usr/bin/evince (enforce)
/usr/lib/connman/scripts/dhclient-script (enforce)
/usr/lib/NetworkManager/nm-dhcp-client.action (enforce)
/sbin/dhclient3 (enforce)
/usr/share/gdm/guest-session/Xsession (enforce)
```

b. Commandes

L'exemple pris pour ces commandes concerne l'utilitaire `traceroute` (le profil est présent mais pas l'utilitaire) :

```
aptitude install traceroute
```

Pour passer en mode `enforce` le profil :

```
aa-enforce usr.sbin.traceroute
```

Pour revenir en mode `complain` : `aa-complain`.

Pour charger un profil : redémarrez le service `apparmor` ou `aa-logprof`.

Le premier lancement de cette dernière commande fait état d'une demande sur l'activation d'un dépôt de profil externe. Répondez non à cette question, cela positionnera à `no` la variable `enabled` dans le fichier `/etc/apparmor/repository.conf`. À la date d'écriture de ces lignes le dépôt `lucid` n'était pas présent ; seule la version `gutsy` existait (voir le fichier `/etc/apparmor/logprof.conf`. Comme Novell se réfère à OpenSuse, les développements se rapportent à cette distribution ou celles d'obédience Red Hat comme la Fedora. Les profils "en stock" sont visibles à l'adresse : http://apparmor.opensuse.org

Pour l'affichage de la démarche d'application d'un profil, nous allons utiliser la commande bien connue `ping`, dont voici le profil dans `/etc/apparmor.d/bin.ping` :

```
# Last Modified: Thu Aug  2 14:28:48 2007
# $Id: bin.ping 935 2007-08-20 01:28:20Z DominicReynolds_ $
# ----------------------------------------------------------------------
#
#    Copyright (C) 2002-2005 Novell/SUSE
#
#    This program is free software; you can redistribute it and/or
#    modify it under the terms of version 2 of the GNU General Public
#    License published by the Free Software Foundation.
#
# ----------------------------------------------------------------------

#include <tunables/global>
/bin/ping flags=(complain) {
  #include <abstractions/base>
  #include <abstractions/consoles>
  #include <abstractions/nameservice>

  capability net_raw,
  capability setuid,
  network inet raw,

  /bin/ping mixr,
  /etc/modules.conf r,
}
```

Sans entrer dans les détails de configuration (que vous trouverez dans la documentation Novell citée plus haut), sachez que :

- La directive `include` permet l'insertion de données issues d'autres fichiers (ici en commentaire).
- La section `/bin/ping` donne son nom au profil.
- La présence de la directive `flags` indique le mode `complain` (absente en cas de `enforce`).
- La directive `capability` permet l'accessibilité `CAP_NET_RAW Posix`.
- Les options `r`, `m`, `ix` définissent respectivement la permission en lecture, l'autorisation de protocole pour les appels `mmap` et `inherite execute` pour l'exécution.

Vous verrez plus loin que la définition de ces éléments est facilitée par la commande `aa-genprof`.

Les commandes préfixées par `aa-` se révèlent en fait des liens vers d'autres commandes. Gardez cette écriture afin d'éviter des confusions.

Vérification du processus

On va montrer le statut de la commande `ping` dans son fonctionnement :

- Ouvrez une deuxième console.
- Faites un `ping` sur une adresse IP de votre réseau local.
- Sur la deuxième console, tapez la commande :

  ```
  ps aZ | grep ping
  ```

  Le résultat au niveau de la deuxième ligne montre un statut `unconfined`.

  Passage en mode `enforce` :

  ```
  aa-enforce /bin/ping
  ```

Vous pouvez le vérifier par la commande `apparmor_status`.

c. Création d'un profil

Voici un script simple, nommé `test.sh` :

```
#/bin/bash
exec ls -ail
```

- Donnez-lui les droits en exécution :

```
chmod 755 test.sh
```

- Générez le profil :

```
aa-genprof /root/test.sh
```

Le profilage est en cours par une mise sur écoute avec, en retour de messages, l'indication que l'application est démarrée en mode `complain`. À ce stade, deux solutions :

- Soit lancer le programme dans une autre console, revenir dans la première et appuyer sur (`S`) pour `Scan` (**meilleure solution**).

- Soit appuyer directement sur (`F`) pour `Finish`, lancer le programme et ensuite la commande `aa-logprof`.

Une série de questions traite des exécutions et permissions nécessaires au bon déroulement du programme. La plupart du temps, il s'agit de répondre (`I`) pour `Inherit` (héritage du programme principal), (`A`) pour `allow` dans les cas de permissions :

```
Répertoire: /                    -> Allow en mode r pour lecture
Réperoire: /bin/bash             -> Allow en mode r pour lecture
Execute : /bin/ls                -> Inherit en mode x pour exécution
Répertoire : /dev/tty            -> Allow en mode r pour lecture
Fichier : /etc/group             -> Allow en mode r pour lecture
Fichier : /etc/nsswitch.conf     -> Allow en mode r pour lecture
Fichier : /etc/passwd            -> Allow en mode r pour lecture
Répertoire : /root/              -> Allow en mode r pour lecture
Fichier : /root/.aptitude        -> Allow en mode r pour lecture
Fichier : /root/test.sh          -> Allow en mode r pour lecture
Fichier : /root/.viminfo         -> Allow en mode r pour lecture
Fichier : /root/.bash_history    -> Allow en mode r pour lecture
Fichier : /root/.bashrc          -> Allow en mode r pour lecture
Fichier : /root/.profile         -> Allow en mode r pour lecture
```

On voit nettement la succession d'opérations, du lancement du script à l'affichage respectif de la commande `ls` (dans le répertoire `/root` il n'y a que 6 fichiers).

Au final, (`S`) pour sauvegarder le profil :

```
# Last Modified: Fri Jun 20 19:09:25 2008
#include <tunables/global>
/root/test.sh {
  #include <abstractions/base>

  / r,
  /bin/ls ixr,
  /etc/group r,
  /etc/nsswitch.conf r,
  /etc/passwd r,
  /proc/*/mounts r,
  /root/ r,
  /root/.aptitude/ r,
  /root/.bash_history r,
  /root/.bashrc r,
  /root/.profile r,
  /root/.viminfo r,
  /root/test.sh mr,
}
```

Pour faire prendre en compte uniquement le profil :

```
cat /etc/apparmor.d/root.test.sh | apparmor_parser -r
```

Supposons maintenant que l'on change la seule ligne du programme par celle-ci, au combien malveillante :

```
rm *.*
```

L'exécution à nouveau du programme retournera la phase suivante :

```
./test.sh: line 2: /bin/rm: Pemission denied
```

On en voit tout l'intérêt par exemple pour des scripts exécutés à partir d'un serveur.