SELinux, AppArmor, etc.

S. Aicardi

CMLS

Journées Mathrice, Poitiers, 18-20 Mars 2008

Plan

Plan

Contrôle d'accès : pourquoi et comment ?

La sécurité dans le monde numérique cherche à garantir les données selon trois axes :

- *confidentialité*,
- *intégrité*,
- *disponibilité*.

La garantie de la confidentialité et de l'intégrité des données passe par un contrôle d'accès.

Dès qu'un système informatique est multi-utilisateur et/ou accessible à distance, la protection physique des données est insuffisante et il faut un contrôle d'accès logiciel.

La politique de sécurité Unix classique

Tout processus est exécuté par un utilisateur qui appartient à un ou des groupes.

Tout fichier possède un propriétaire (a priori son créateur) et un groupe propriétaire. Les droits d'accès donnent le droit de lecture, d'écriture et d'exécution sur le fichier pour le propriétaire, les membres du groupe propriétaire et le reste du monde.

```
-rw-r-r- 1 aicardi info 22494 jan 28 16:31 SELinux.tex
```

Les droits peuvent être changés par le propriétaire ou par root.

Ce type de contrôle d'accès est appelé DAC (*Discretionary Access Control*).

Une gestion des droits plus fine

Il est possible sur certains systèmes de fichiers (ext3, xfs, nfs,...) de mettre en place le support des ACL (*Access Control List*) POSIX. Cela permet de définir des droits individuels par utilisateurs.
Exemple :

```
tex/Mathrice$ getfacl SELinux.tex
# file:  SELinux.tex
# owner:  aicardi
# group:  info
user::rw-
user:hamet:rw-
group::r-
other::r-
```

Là encore, ces droits sont fixés par le propriétaire ou par root, cela reste un DAC.

Critiques du contrôle d'accès discrétionnaire

Il donne tout pouvoir au propriétaire du fichier et à `root`.
$\Longrightarrow$ confiance en l'utilisateur
$\Longrightarrow$ pas d'hiérarchisation des données (confidentiel, secret, etc.)
$\Longrightarrow$ pas d'hiérarchisation des utilisateurs (admin, chef, sous-chef, larbin)
$\Longrightarrow$ pas de cloisonnement entre groupes (marketing, R&D, administration)

Contrôle d'accès obligatoire

Ce contrôle d'accès (en anglais : *Mandatory Access Control* ou *MAC*) respecte une politique définie par l'administrateur. L'utilisateur n'a pas forcément de moyen de partager des fichiers sur lesquels il a des droits.

La mise en œuvre se fait en collant des *labels* sur les processus et les objets (fichiers, sockets réseaux, etc.) et en définissant une hiérarchie sur les labels. Par exemple dans un cadre militaire : $\textit{Confidentiel} \leq \textit{Secret} \leq \textit{Top secret}$.

On pourra par exemple interdire à un objet labellé *Top secret* d'être lu par un utilisateur accrédité *Confidentiel* ou de transiter par le réseau. On peut aussi interdire à un utilisateur accrédité *Top secret* de produire des documents *Confidentiel* pour éviter les fuites.

DAC ou MAC

DAC	MAC
contrôle d'accès fichier par fichier	politique de sécurité globale sur tous les objets
s'applique aux utilisateurs	s'applique aux programmes
contrôle d'accès défini par les utilisateurs	contrôle d'accès imposé par l'administrateur
politique "autorise sauf si"	politique "interdit sauf si"
mise en œuvre aisée	planification précise avant mise en œuvre

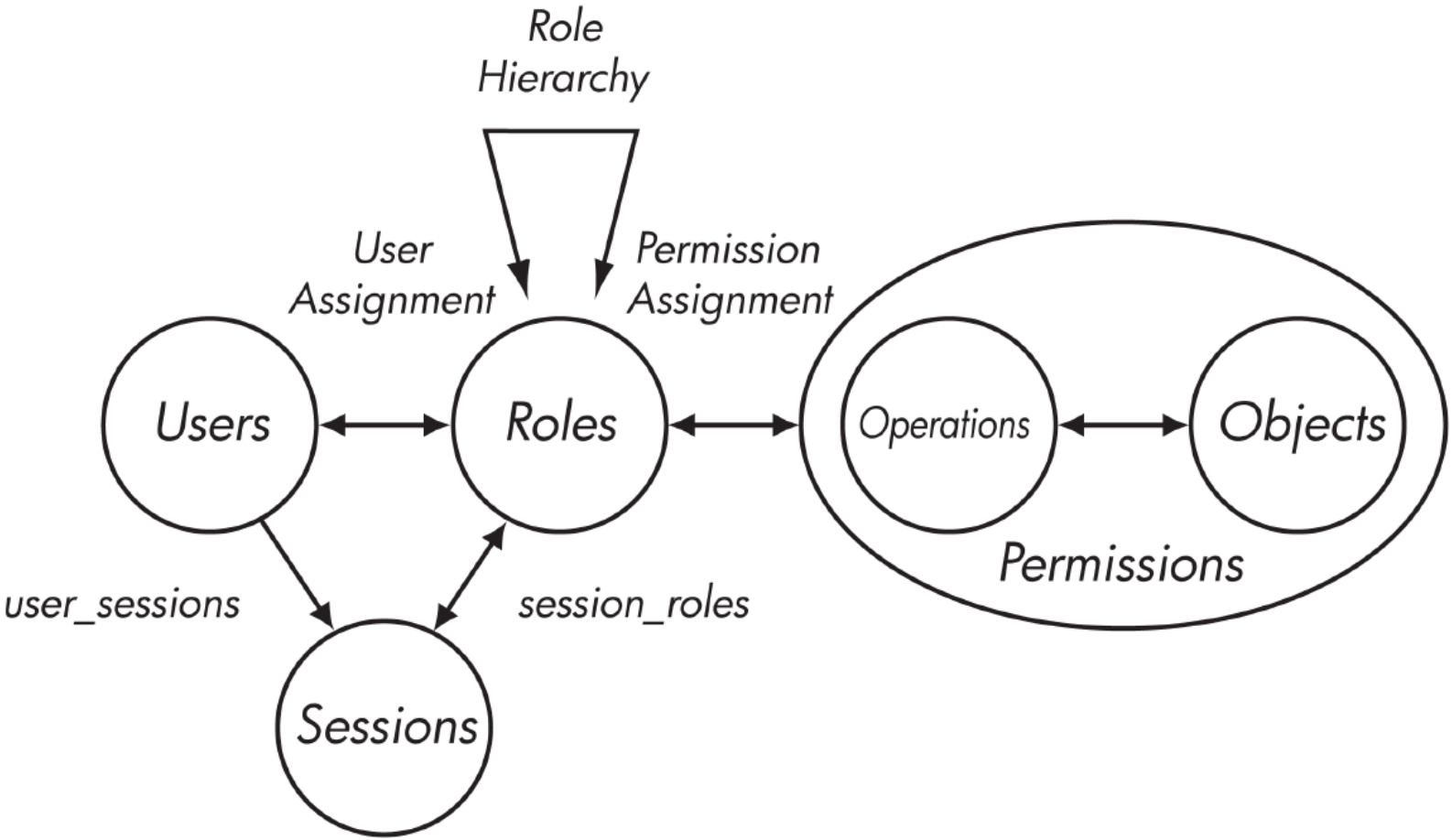
Contrôle d'accès par rôle

Ce contrôle d'accès (en anglais : *Role-Based Access Control* ou *RBAC*) est une nouvelle approche au problème du contrôle d'accès.

On sépare les :
utilisateurs (= compte),
sessions (= processus),
rôles (= fonction dans le SI qui donne un type d'accréditation),
permissions (= droit d'accès dans le SI).

Chaque session correspond à un utilisateur et peut avoir plusieurs rôles (dans la limite des rôles autorisés à l'utilisateur). À chaque rôle est associé un ensemble de permissions.

Contrôle d'accès par rôle



Contrôle d'accès par rôle

Avantage : On peut faire du DAC ou du MAC avec du RBAC.

Le RBAC permet et facilite une mise en œuvre du MAC.

Actuellement utilisé dans SELinux, FreeBSD, Solaris, Oracle, Microsoft Active Directory et bien d'autres.

Type Enforcement

Terminologie utilisée par SELinux pour décrire sa politique de sécurité. Elle reprend du MAC l'idée d'étiqueter processus et objets et d'autoriser suivant des listes de permissions globales.

Utilité pour sécuriser des serveurs

Un certain nombre de services tournent avec les droits de `root`. L'exploitation d'une faille de sécurité donne a priori tous les droits sur le serveur à l'attaquant.

L'attaquant peut alors télécharger un rootkit/sniffer/bot IRC, le planquer dans un recoin du système de fichiers (par exemple `/var/tmp/ /.  /`) et l'exécuter.

Il peut éventuellement effacer les données des utilisateurs pour installer ensuite un serveur de Warez ou P2P illégal.

Utilité pour sécuriser des serveurs

Une politique de sécurité de type MAC ou RBAC permet d'interdire à un démon réseau même exécuté par `root` (par exemple `cupsd`) :

- d'accéder au réseau sauf sur le port 631,
- d'exécuter des programmes autres qu'une liste précise (les filtres de conversion),
- d'écrire des fichiers sauf dans `/var/spool/cups` et dans `/var/log/cups`,
- de lire des fichiers ailleurs que dans `/etc/cups`.

Plan

Linux security modules

Depuis la version 2.6, un “framework” de sécurité est intégré au noyau linux. Il fournit une API qui intercepte les appels systèmes (par des “hooks”) et en vérifie la conformité avec la politique de sécurité.

Le framework LSM cherche à être à la fois indépendant de l'implémentation de la politique de sécurité et à modifier aussi peu de choses que possible dans le noyau.

Introduit initialement pour SELinux, il est utilisé également par AppArmor, Linux Intrusion Detection System, etc. D'autres systèmes de sécurité pour Linux imposent au contraire de désactiver LSM comme grsecurity et RSBAC.

“Mon module de sécurité va jusqu’à 11”, partie I

Une tendance générale chez les développeurs de logiciels de sécurité est de dire que :

- leur module exécute les hackers dans leur sommeil avant même qu’ils aient eu l’idée d’un exploit,
- celui des autres sont tellement troués qu’un enfant de trois ans les contourne,
- voire que celui des autres ouvre des backdoors qui n’existaient pas avant.

“Mon module de sécurité va jusqu’à 11”, partie I

Critiques de LSM par grsecurity et RSBAC :

- LSM ne s’occupe que de contrôle d’accès,
- LSM est incomplet (manque de hooks pour RSBAC),
- LSM est dangereux car il expose des zones sensibles du noyau aux hackers,
- LSM est sans état.

En filigrane, les critiques contre LSM se résument souvent en “c’est fait pour (par) SELinux”. D’ailleurs, LSM a failli disparaître du noyau en 2006, avec comme argument que seul SELinux dans le noyau officiel s’en servait.

Security-Enhanced Linux

Module de sécurité développé par la NSA (National Security Agency). Historiquement publié comme des patchs noyaux, SELinux a été intégré au noyau 2.6 comme un module utilisant LSM.

SELinux permet de définir des politiques de sécurité de type MAC sur un système Linux classique. Outre la NSA, SELinux est activement développé par RedHat, qui a obtenu une certification EAL4+ pour la RHEL 5 sur matériel IBM.

SELinux : dans les distributions

- Fedora : disponible depuis la core 2,
- RedHat : disponible depuis la RHEL 4,
- Debian : disponible depuis etch (4.0), mais désactivé par défaut,
- Gentoo : disponible dans la version hardened,
- Ubuntu : sera disponible à partir de la version Hardy Heron (8.04) comme alternative à AppArmor

NB : Ce qui suit a été testé sur des Fedora 6 à 8.

SELinux : Activation

1. au démarrage, les noyaux configurés avec l'option `CONFIG_SECURITY_SELINUX_BOOTPARAM=y` permettent d'activer (resp. désactiver) SELinux en passant l'option `selinux=1` (resp. 0) au boot
2. dans le fichier `/etc/selinux/config`, l'option `SELINUX` permet de choisir entre trois possibilités : `disabled`, `permissive` et `enforcing`.
3. si on a booté un noyau avec SELinux activé (se teste avec `selinuxenabled`), on peut passer du mode `permissive` au mode `enforcing` avec la commande `setenforce 1`.
4. Pour savoir où on en est : `sestatus -v`

SELinux : Comment ça marche : rôles, types, accréditation

- Les descripteurs de fichiers (fichiers, répertoires, devices, points de montages, etc.) sont labellés :

```
$ ls -Z /etc/shadow
-r---- root root system_u:object_r:shadow_t /etc/shadow
```

- Les utilisateurs sont labellés :

```
$ id -Z
user_u:system_r:unconfined_t
```

- Les processus sont labellés :

```
$ ps -Z
 LABEL                          PID     TTY      TIME         CMD
 user_u:system_r:unconfined_t   19710   pts/1    00:00:00     bash
```

SELinux : Comment ça marche : rôles, types, accréditation

Quelques informations étaient restées cachées :

```
# secon -R
user:  root
role:  system_r
type:  unconfined_t
sensitivity:  s0
clearance:  s0:c0.c1023
mls-range:  s0-s0:c0.c1023
```

Elles sont stockées dans l'attribut étendu `security.selinux` de l'inode :

```
# getfattr -n security.selinux .
# file:  .
security.selinux="root:object_r:user_home_dir_t:s0\000"
```

Les sauvegardes doivent conserver les attributs étendus. Par exemple :

```
star -xattr -artype=exustar -c -f toto.tar toto
rsync -X ...
```

SELinux : Comment ça marche : MLS, MCS

SELinux propose 16 niveaux de sécurité (s0,...,s15) et 1024 catégories (c0,...,c1023). Les premiers permettent d'implémenter une sécurité de type militaire (Multi-Level Security). Les deuxièmes permettent de cloisonner l'information selon par exemple les services d'une entreprise (Multi-Category Security).

NB : Les catégories permettent de faire sensiblement la même chose que des groupes et des ACLs POSIX. Les niveaux de sécurité sont beaucoup plus contraignants pour définir une politique globale sur le système (ex : serveur mail).

SELinux : Comment ça marche : TE, RBAC, MCS, DAC

Si un processus (`user_p:role_p:type_p:mcs_p`) doit faire une opération sur un objet (`user_o:role_o:type_o:mcs_o`), on teste successivement :

- les permissions unix standards,
- si le type `type_p` est autorisé à faire cette opération sur un objet de type `type_o`.
- idem pour les niveaux de sécurité/catégories

Un certain nombre d'opérations nécessitent un changement de type (typiquement, l'éxécution d'un démon). La liste des types accessibles est déterminée par le rôle.

SELinux : Comment ça marche : politique

Les politiques sont le cœur de SELinux. Fedora/RedHat propose des politiques toutes faites pour les cas courant :

- *strict* : implémente la politique *strict* de la NSA. Elle vise à protéger tout le système. Adaptée éventuellement à un serveur sans utilisateurs, mais nécessite un long travail de configuration ;
- *targeted* : vise à protéger les démons et utilitaires les plus sensibles. Les autres processes sont du type `unconfined_t` ;
- *mls* : ajoute une déclinaison en 16 niveaux de sécurité (s0,...,s15).

La politique se fixe avec le paramètre `SELINUXTYPE` dans le fichier `/etc/selinux/config`.

SELinux : Comment ça marche : modules

Depuis Fedora Core 5, une politique SELinux n'est plus monolithique. Elle est composée de modules et l'administrateur peut en ajouter de nouveaux à chaud (modulo une éventuelle opération de retypage du filesystem).

La commande `semodule -l` donne la liste des modules installés. Pour ajouter un module : `semodule -i module.pp`.

Attention, le nom du module est important, on peut écraser un module fourni par la distribution, voire toute la politique si le module s'appelle `base`.

SELinux : Comment ça marche : booléens

La politique peut permettre de changer un certain nombre de paramètres à chaud. La liste complète est disponible avec la commande `getsebool -a` ou dans le répertoire `/selinux/booleans`.

On peut changer une valeur avec la commande `setsebool`.

Exemple (utile) :
`setsebool -P allow_execheap=1`

SELinux : Comment ça marche : logs

Si un processus tente de faire une action contraire à la politique, l'action est logguée dans `/var/log/messages` et apparaît dans `dmesg`. Exemple :

```
avc:  denied { execheap } for pid=25845 comm="CALCUL"
scontext=user_u:system_r:unconfined_t:s0
tcontext=user_u:system_r:unconfined_t:s0
tclass=process
```

SELinux : Créér sa politique

Supposons que la politique par défaut est inadaptée pour un logiciel commercial de calcul (il a besoin de `execheap` et de `execstack`). On a trois solutions :

- désactiver SELinux ($\Longrightarrow$ plus aucune protection),
- changer les booléens nécessaires ($\Longrightarrow$ plus de protection contre un code utilisateur malveillant),
- créer son module de politique pour ce logiciel et éventuellement en faire profiter la terre entière !

SELinux : Créér sa politique

Ingrédients d'une politique pour `calcul`™ :

- un fichier `calcul.fc` contenant les exécutables concernés,
- un fichier `calcul.if` contenant les interfaces (??)
- un fichier `calcul.te` contenant la politique de sécurité.

Comment savoir ce qu'il faut y mettre ?

En général, il n'est pas très dur de deviner les exécutables en cause ($\Longrightarrow$ path complet dans `calcul.fc`). Un `calcul.if` vide fait l'affaire pour un programme utilisateur.

SELinux : Créér sa politique

Pour préparer `calcul.te`, le plus simple est de passer SELinux en mode permissif (`setenforce 0`), lancer le programme et d'éxécuter toutes ses fonctionnalités, puis passer la commande suivante : `audit2allow -d -m calcul > calcul.te`

Cette opération donne sûrement des résultats trop laxistes. Il faut relire et corriger le fichier `calcul.te` obtenu.

Pour activer la nouvelle politique :

```
checkmodule -M -m -o calcul.mod calcul.te
semodule_package -o calcul.pp -m calcul.mod -f calcul.fc
semodule -i calcul.pp
```

SELinux : Exemple (extrait !) de fichier `calcul.te`

```
policy_module(CALCUL,1.0.0)
type CALCUL_t;
type CALCUL_exec_t;
domain_type(CALCUL_t)
init_daemon_domain(CALCUL_t, CALCUL_exec_t)

require {
  type file_t;
  type port_t;

  class file { read write getattr execute execute_no_trans rename create unlink link };
  class process { execstack execmem execheap getsched signal signull };
  class tcp_socket { name_bind name_connect send_msg setopt read bind create recv_msg write
getattr connect shutdown getopt node_bind };
}

allow CALCUL_t file_t:file { read write getattr execute execute_no_trans rename create unlink
link };
allow CALCUL_t self:process { execstack execmem execheap getsched signal signull };
allow CALCUL_t port_t:tcp_socket { name_connect send_msg recv_msg };

domain_auto_trans(unconfined_t, CALCUL_exec_t, CALCUL_t)
```

SELinux : Créér sa politique en cliquant

Pour les allergiques à la ligne de commande, RedHat/Fedora a développé une interface graphique pour créer et modifier sa politique SELinux : `system-config-selinux` (dans le paquet `policycoreutils-gui`). On a un accès aux booléens, aux modules, aux utilisateurs, aux types de fichiers, et une assistance pour créer un nouveau module. Cela simplifie la procédure de création d'une politique, mais il faudra toujours éditer le fichier `calcul.te` pour l'adapter à ses besoins.

Un projet concurrent permet également d'éditer une politique (différente de la politique *targeted*) : SEEdit.

SELinux : Problèmes rencontrés et solutions

La plupart des logiciels commerciaux de calcul ne sont pas adaptés à SELinux. Pour arriver à les lancer même dans une politique *targeted*, il faire quelques modifications du type :

```
chcon -t textrel_shlib_t /usr/local/calcul/lib/lib_importante.so
```

Tout changement sérieux de la politique de sécurité impose de retyper tout le filesystem, ce qui peut vraiment prendre du temps !

Certains systèmes de fichiers ne supportent pas les labels SELinux, par exemple NFS. Tout fichier sur un montage NFS est de type `nfs_t`.

SELinux : critiques

- ce sont les distributions qui définissent les politiques,
- donne l'impression de nécessiter un BAC+20 pour modifier les droits,
- marche à peu près en *targeted* sous Fedora si on ne s'écarte pas de l'installation de base,
- difficile à mettre en œuvre et facile à désactiver selon des développeurs d'OpenBSD
- doit-on vraiment en arriver là pour sécuriser quelques services ?

SELinux : critiques

- ce sont les distributions qui définissent les politiques,
- donne l'impression de nécessiter un BAC+20 pour modifier les droits,
- marche à peu près en *targeted* sous Fedora si on ne s'écarte pas de l'installation de base,
- difficile à mettre en œuvre et facile à désactiver selon des développeurs d'OpenBSD
- doit-on vraiment en arriver là pour sécuriser quelques services ?
- **donne trop souvent envie de faire `setenforce 0`**

AppArmor

Initialement développé par Immunix Inc, AppArmor a continué à être développé par Novell après le rachat d'Immunix en 2005.

AppArmor se présente comme une solution simple à mettre en œuvre et qui ne cherche pas à cadenasser hermétiquement tout le système. Elle est intégrée à SUSE Linux Enterprise depuis la version 10 et à Ubuntu depuis Gutsy Gibbon (7.10).

AppArmor : Comment ça marche

AppArmor utilise l'API LSM pour limiter certains exécutables sur plusieurs points :

- accès aux fichiers (`rwx`) selon leur chemin absolu,
- *capabilities* POSIX : un certain nombre de fonctions accessibles à `root` dans un système unix classique (par ex. `CAP_SETUID`, `CAP_NET_BIND_SERVICE`, `CAP_KILL`,...)

AppArmor : Comment ça marche

Un module noyau est chargé au démarrage (via `/etc/init.d/apparmor`). La configuration générale est dans `/etc/apparmor`. Les politiques (profils) sont dans `/etc/apparmor.d` et nommées en remplaçant dans le chemin absolu les `/` par des `..`

Un profil peut exister sous deux états :

- *complain* : le profil produit des erreurs, mais ne bloque rien (idem *permissive* dans SELinux),
- *enforce* : le profil bloque effectivement les actions illégales.

AppArmor : Quelques commandes en ligne

Pour passer un profil en mode enforce :
`aa-enforce /path/vers/executable`.

Pour passer un profil en mode complain :
`aa-complain /path/vers/executable`.

Pour recharger tous les profils :
`/etc/init.d/apparmor reload`.

Pour recharger un profil :
`cat /etc/apparmor.d/nom.du.profil | apparmor_parser -r`.

AppArmor : Comment créer un profil

Lancer la commande `aa-genprof /path/vers/executable`, puis lancer le programme et en explorer toutes les fonctionalités. Ensuite suivre les instructions pour obtenir le profil.

Pour du travail d'envergure, il vaut mieux utiliser `aa-autodep`, puis ajuster le profil grâce à `aa-logprof`.

Les amateurs de souris et de SUSE peuvent utiliser YAST pour faire sensiblement la même chose.

AppArmor : Exemple complet d'un profil

```
/usr/sbin/named flags=(complain) {
  #include <abstractions/base>
  #include <abstractions/nameservice>

  capability net_bind_service,
  capability setgid,
  capability setuid,
  capability sys_chroot,

  /etc/bind/* r,
  /proc/net/if_inet6 r,
  /usr/sbin/named mr,
  /var/cache/bind/* rw,
  /var/run/bind/run/named.pid w,
  # support for resolvconf
  /var/run/bind/named.options r,
}
```

AppArmor contre SELinux

AppArmor ne sécurise qu'un certain nombre de programmes, SELinux vise à sécuriser tout le système (en politique stricte).

Plus grande finesse d'action chez SELinux (granularité des droits, MLS, RBAC).

Profils lisibles et faciles à créer chez AppArmor.

SELinux impose une intervention sur le système de fichiers et les utilitaires de base, AppArmor s'intègre directement en se basant sur les chemins d'accès.

AppArmor permet de créer des sous-profils (par ex. pour sécuriser une page PHP plus spécifiquement que le serveur Apache). SELinux considère Apache comme un seul processus si les pages PHP utilisent `mod_php`.

“Mon module de sécurité va jusqu’à 11”, partie II

Selon diverses personnes anonymes (qui travaillent sur SELinux) :

- AppArmor ne protège que quelques démons, donc ce n’est pas sérieux.
- On ne peut pas faire de sécurité sérieusement en se basant sur les chemins d’accès.
- L’outil de création de politique peut donner des résultats dangereux.
- Il vaut mieux que des pros de la sécurité définissent les politiques plutôt que de laisser ça à un utilitaire automatique