Linux Certif

Comprendre SELinux

Cet article introduit les mécanismes de SELinux, son modèle de sécurité et quelques outils indispensable pour son utilisation. Vous apprendrez les bases po administrer un système avec SELinux: les mécanismes de sécurité, le labeling du système de fichier, la personnalisation, etc. L'écriture de nouvelles politiqu sécurités n'est par contre par abordé dans cette introduction.

Sommaire

Introduction

SELinux (pour *Security-Enhanced Linux*) est un modèle de sécurité que l'on peut ajouter au système standard de Linux afin d'augmenter la sécurité face au diverses attaques que subit le système. Avec SELinux, on peut configurer les accès de chaque processus pour les restreindre à un strict minimum. Cela per

de rendre inexploitable certaines failles, et en cas de piratage, de limiter l'étendue des dégâts.

SELinux est parfois perçu comme un système de sécurité obscur et compliqué. Pourtant il propose un formidable modèle pour renforcer les droits classiques Linux, et il serait dommage de s'en passer.

Cet article tente de démystifier SELinux en présentant les principes généraux et les différents composants du système. Le but n'est pas de pouvoir écrire de règles de sécurité pour SELinux, mais de comprendre comment fonctionne ce modèle de sécurité et de découvrir les commandes pour l'utiliser.

SELinux en 4 minutes

Avant de voir la théorie, voyons un exemple de l'intérêt de SELinux. Imaginons que quelqu'un vient de pirater apache, et qu'il peux lancer des commandes arbitraires via ce processus. Pour ne rien arranger, le processus piraté de apache fonctionne en root.

En tant que root, le pirate peut modifier le fichier /etc/shadow pour s'ajouter un compte sur le serveur. Les droits du fichier permettent de le faire (après tout, notre pirate à l'accès root):

```
$ ls -l /etc/shadow
-rw-r----- 1 root shadow 779 2008-08-06 02:13 /etc/shadow
```

Et pourtant le pirate ne pourra pas ouvrir le fichier. Pourquoi? Grâce à la configuration de SELinux.

Il se trouve que apache fonctionne dans un contexte de sécurité particulier de SELinux, il a le type **httpd_t**

```
$ ps -Z
system_u:system_r:httpd_t:s0   14560 ?        Ss     0:00 /usr/sbin/apache2
```

Le fichier *shadow*, possède lui aussi un contexte de sécurité, et son type est **shadow_t**:

```
$ ls -Z /etc/shadow
-rw-r-----  root shadow system_u:object_r:shadow_t:s0    shadow
```

Avec SELinux, pour que apache puisse lire ou écrire le fichier *shadow*, il faut une règle explicite qui permet aux processus de type **httpd_t** de lire ou écrire d un fichier de type **shadow_t**. Il faudrait donc une règle comme celle-ci dans les politiques de sécurité de SELinux:

```
allow httpd_t shadow_t : file { read write }
```

Cette règle dit qu'un processus avec le label **httpd_t** est autorisé (**allow**) à lire et écrire (**{ read write }**) dans un fichier (**file**) possédant le type/label **shadc**

Il n'existe évidemment aucune règle de ce type, apache n'a aucune raison d'aller lire ce fichier, même si il fonctionne avec les droits root. Un pirate est ainsi à ce qu'apache peut faire dans une utilisation normale ce qui l'empêchera de pirater le reste du système.

Installer SELinux

Un bon départ pour comprendre SELinux est de l'installer et de jouer avec les droits.

SELinux est présent dans les noyaux de la plupart des distributions, il suffit donc d'installer les packages proposant les politiques de sécurités (SELinux est parfois activé dès l'installation, vous pouvez le voir avec la commande sestatus). Voyez la page dédiée pour votre distribution:

- Debian
- Ubuntu
- Fedora
- Gentoo

Notez que openSuSE n'utilise pas SELinux mais un autre système avec un modèle de sécurité extrêmement similaire: appArmor. OpenSuSE proposera auss SELinux à partir de la version 11.1.

Le modèle de sécurité classique

Avant de commencer avec les nouveautés de SELinux, repassons le modèle de sécurité classique d'Unix, et ses limitations.

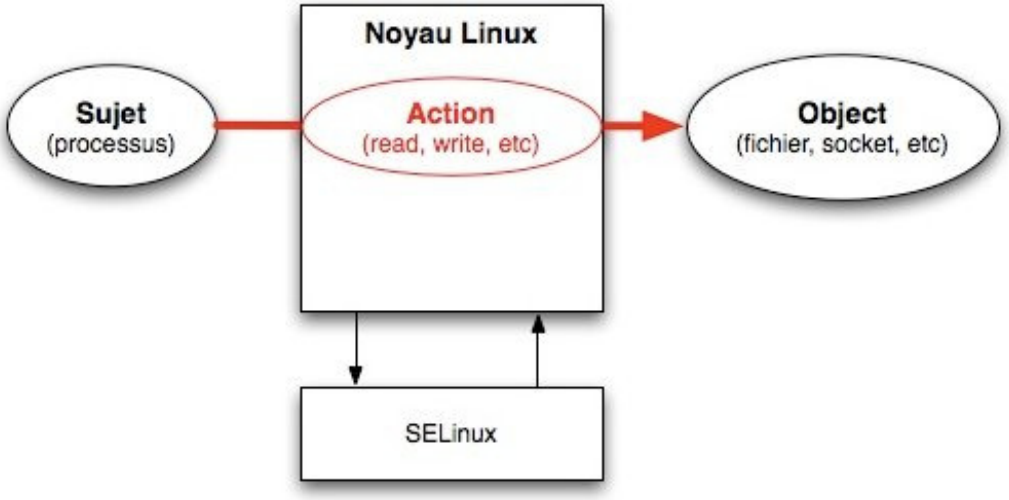
Sur Linux, les fichiers possèdent un propriétaire, un groupe, et une chaîne de bit permettant de savoir qui peut faire quoi. Lorsque qu'un processus veut réa une action sur un fichier, le noyau vérifie que le processus possède les droits nécessaires.

Ce système de sécurité a été conçu pour protéger chaque utilisateur par rapport aux autres utilisateurs, et il réalise parfaitement cette tâche, mais il ne prot pas les programmes d'une utilisation détournée. Par exemple, le système standard ne peut rien faire pour les programmes qui doivent fonctionner en root.

Le type de problèmes de sécurité rencontrés à largement évolué depuis la conception de la sécurité de base de Unix. Les problèmes de sécurité viennent désormais majoritairement de failles dans les applications, et il est devenu nécessaire de se protéger des applications au niveau du système d'expoitation.

Le modèle de sécurité de SELinux

SELinux utilise un système de sécurité extrêmement souple: pour chaque appel système, le noyau vérifie que le processus peut exécuter l'appel sur l'objet manipulé en fonction de leur contexte de sécurité. Le type de vérification réalisé par le noyau est parfois comparé au découpage d'une phrase en français, un découpage "Sujet Verbe Objet", le sujet est un processus, l'objet est un objet système (par exemple un fichier), et le verbe est l'action. SELinux vérifie do que le sujet peut réaliser le verbe sur l'objet.



SELinux vient s'ajouter au modèle de sécurité classique, les droits normaux des fichiers sont toujours de rigueur. Si un accès est refusé par le système class SELinux n'entre pas en jeu. Il n'est donc toujours pas possible pour un utilisateur de lire le compte d'un autre utilisateur, même si la politique de sécurité de SELinux le permet.

Les informations de SELinux

Pour identifier sujet et objet, SELinux ajoute à ceux-ci trois informations: **identité**, **rôle** et le **type** (c'est ce qu'on appelle le label pour les objets). Celles-ci s accessibles via l'option *-Z* que l'on peut ajouter à de nombreuses commande. Les trois informations sont séparé par des double point (comme dans *system_u:system_r:crond_t*).

Par convention, les identifiants possèdent un suffixe: *_u* pour l'identité, *_r* pour le rôle et *_t* pour le type.

Parmi les informations ajoutées par SELinux, le type est l'élément fondamental, c'est par lui que les droits sont accordés. L'identité et le rôle ont un rôle secondaire que nous verrons plus tard.

Voyons comment afficher les informations avec des commandes courantes, le type pour les processus du serveur PostgreSQL est *postgresql_t*:

```
$ ps -Z
system_u:system_r:postgresql_t:s0 1531 ?       S      0:05 /usr/lib/postgresql/8.1/bin/postmaster
system_u:system_r:postgresql_t:s0 1533 ?       S      0:01 postgres: writer process
```

Pour les fichiers de PostgreSQL, le type est postgresql_db_t

```
$ ls -Z
drwx------  postgres postgres user_u:object_r:postgresql_db_t:s0 base
```

```
-rw-------  postgres postgres user_u:object_r:postgresql_db_t:s0 pg_hba.conf
-rw-------  postgres postgres user_u:object_r:postgresql_db_t:s0 pg_ident.conf
```

Un point important concernant les informations utilisées par SELinux est que celles-ci ne sont pas restreintes aux fichiers, comme pour le système de sécurit classique. Les *objects* peuvent ainsi être des sockets, des sémaphores, de la mémoire partagée, etc. Ce système est donc beaucoup plus rigoureux que le système de sécurité classique. Il est par exemple impossible pour un processus d'ouvrir un socket sans avoir le bon label.

Note sur la terminologie

Pour des raisons lié à la théorie derrière SELinux, le vocabulaire est un peu bordélique pour désigner les informations de sécurité. Ces informations sont pa appelées "*type*" lorsque l'on parle de fichier et "*domaine*" lorsque l'ont parle de processus. On parle aussi de "*contexte de sécurité*" (security context).

Quand vous communiquez avec quelqu'un à propos de SELinux, assurez vous bien que vous parlez de la même chose avant d'en venir aux mains.

Les autorisations dans SELinux

Voyons maintenant comment SELinux assure que seules les opérations autorisées sont effectuées.

Par défaut, SELinux n'autorise aucune action, il faut ajouter des règles pour autoriser les actions. Ces règles sont de type:

```
autorisation type_du_sujet type_de_lobjet : type_des_objects opérations_autorisées
```

Par exemple, la règle qui permet au processus PostgreSQL (de type postgresql_t) de lire les fichiers de la base de données (de type postgresql_db_t) est:

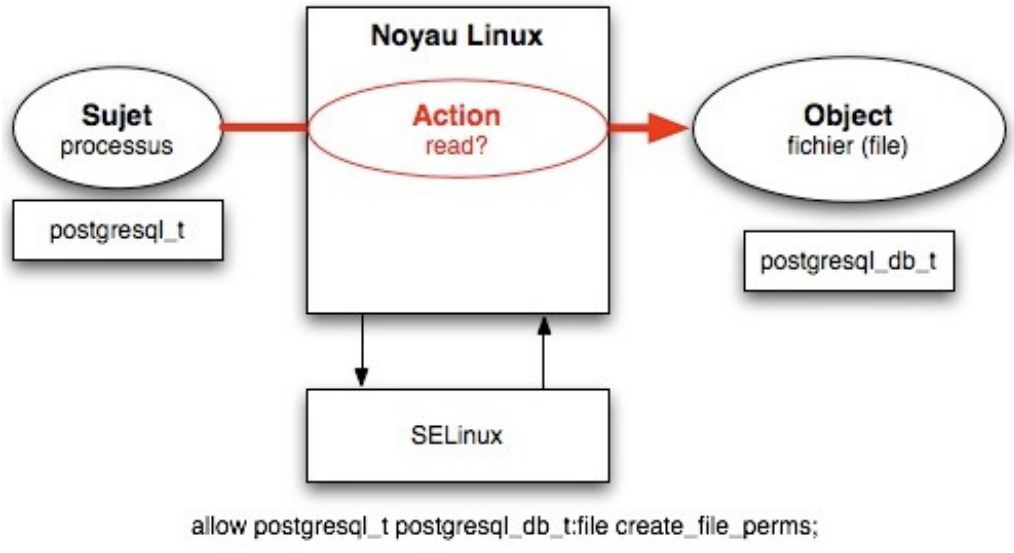
```
allow postgresql_t postgresql_db_t:file create_file_perms;
```

Où create_file_perms équivaut à

```
{ create ioctl read getattr lock write setattr append link unlink rename }
```

Vous l'aurez compris, ces règles sont difficiles à écrire, et leur compréhension nécessite parfois la compréhension de certain concepts propre au noyau Linu Heureusement, il ne faut pratiquement jamais écrire directement ces règles, il faut juste adapter les labels à nos besoins.

L'image suivante montre ce qui se passe lorsqu'un processus de PostgreSQL tente de lire un des fichiers: le noyau doit vérifier que le type du processus (*postgresql_t*) a l'autorisation de lire le fichier avec le type *postgresql_db_t*. Le noyau interroge SELinux qui cherche si une règle des politiques de sécurité autorise cet action.



Une différence importante avec le système classique de sécurité est que dans SELinux, les règles de sécurités ne sont pas écrites statiquement dans le noy elles sont chargés par l'utilisateur à partir de packages de politiques de sécurité. C'est cette différence qui donne tout sa puissance au système.

Transition de type

Vous pourriez vous demander: "d'accord, l'autorisation est donnés à partir des types, mais comment le processus *postgres* a t'il obtenu le type nécessaire: postgresql_t?". La question est intéressante car si n'importe quel processus pouvait prendre n'importe quel type, SELinux ne servirait pas à grand chose.

Le passage d'un type à un autre est strictement définit par les politiques de sécurité. Pour passer d'un type à un autre, il faut que la *transition soit autorisée* le *rôle courant ait accès au type* visé, et il faut lancer un *exécutable qui fait office de point d'entrée pour le type*.

Tout cela peut sembler compliqué mais on va détailler tout ça. Nous parlerons des rôles dans la section suivante.

Pour faire une transition, il faut que celle-ci soit autorisée. Par exemple, il n'est pas question pour PostgreSQL de se faire passer pour apache. Pour Postgre c'est au démarrage qu'il est lancé, il existe donc une autorisation pour une transitions depuis le type *initrc_t*(utilisé au démarrage) vers *postgresql_t*.

Ensuite, il faut que l'exécutable soit un point d'entrée pour le nouveau type. Un point d'entrée signifie que lorsque le fichier est exécuté, le type du processus sera du type fixé par la politique de sécurité.

Pour PostgreSQL, les exécutables de type *postgresql_exec_t* servent de point d'entrée pour le type *postgresql_t*. Les exécutables de PostgreSQL sont donc marqués avec le bon type:

```
$ ls -Z
-rwxr-xr-x  root root system_u:object_r:postgresql_exec_t:s0 postgres
```

Et quand un processus marqué *initrc_t* lance un fichier *postgresql_exec_t*, le processus créé à finalement le type *postgresql_t*.

Je vous passe les commandes SELinux utilisées pour les transitions (pour les curieux, cherchez "file entrypoint" et "process transition"). Ces transitions sont correctement implémentées dans les politiques fournies par les distributions, mais il est intéressant de comprendre leur fonctionnement pour le dépannage.

Par exemple, si un processus n'a pas le bon type, la première chose à vérifier est que le fichier exécutable possède le bon type (pour qu'il puisse servir de p d'entrée).

Si le type du fichier est correct, il faut vérifier que le processus a été lancé par un processus du bon type pour les transitions. Par exemple, pour un script lan au démarrage, la commande run_init permet de relancer un processus avec le type approprié pour la transition:

```
# run_init /etc/init.d/sshd restart
```

Le rôle des rôles

Il peut sembler bizarre que je n'ai pas encore parlé ni d'utilisateur, ni des rôles. Les utilisateurs ont déjà leur droits assurés par le système de sécurité classi d'Unix, ils ont donc un rôle tout à fait secondaire dans SELinux.

Revoyons les informations de sécurité pour l'utilisateur courant:

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

La première colonne (user_u dans l'exemple) est ce que j'ai appelé l'identité. Parfois l'identité à une relation directe avec l'utilisateur courant, chaque utilisat ayant sa propre identité dans SELinux. Dans la plupart des cas cela n'est pas nécessaire et les utilisateurs ont simplement tous l'identité **user_u**.

La seconde colonne (system_r dans l'exemple) est le rôle. Chaque utilisateur peut avoir plusieurs rôles, et les différents rôles possibles sont configurés dans politiques de sécurité. Le rôle est beaucoup plus important que l'identité, car il lui correspond une liste de type possible. Une transition d'un type à un autre e possible pour un processus uniquement si le type de destination l'est pour le rôle courant.

Pour résumer, aux utilisateurs correspondent des identités, aux identités correspondent des rôles possibles, et aux rôles correspondent des type acceptable

Notons que les fichiers ont aussi des informations d'identité et de rôle:

```
$ ls -Z
-rw-r-----  ikipou ikipou user_u:object_r:user_home_t:s0   pg_hba.conf
-rw-r--r--  ikipou ikipou user_u:object_r:user_home_t:s0   README
-rw-r--r--  ikipou ikipou user_u:object_r:user_home_t:s0   SECURITY
```

Ces informations sont là pour rester cohérentes avec le reste du système de sécurité, mais elles ne servent à rien pour les politiques de sécurités.

Les identités et rôles ne sont pas encore largement utilisés dans les politiques fournies dans les distributions Linux.

Les politiques de sécurités

Pour utiliser les règles de sécurité de SELinux, il faut les charger dans le noyau. Pour ce faire, il y à un format binaire contenant les politiques, les règles doi donc être compilées.

Pour rendre l'administration plus simple, le système de politique de sécurité est modulaire, on peut charger ou décharger des "policy package" afin d'ajouter ensembles de règles au noyau.

Ces packages de politique de sécurité sont administrés à l'aide de la commande semodule. L'option -/ permet de lister les packages actuellement chargés:

```
# semodule -l
dmidecode       1.1.0
gpg     1.1.0
inetd   1.2.0
postfix 1.3.0
postgresql      1.2.0
pythonsupport   0.0.1
sasl    1.3.0
ssh     1.4.0
tcpd    1.1.0
udev    1.4.0
```

Pour ajouter des politiques de sécurité, il suffit d'installer un package dans le noyau. Pour se faire, on utilise encore semodule, et on précise le chemin du package à charger:

```
# semodule -i /usr/share/selinux/refpolicy-targeted/apache.pp
```

Et pour recharger un module après l'avoir modifié:

```
# semodule -R /home/ikipou/rules_linux_certif.pp
```

Labeling du système de fichier

J'ai laissé pour la fin un problème qui se pose lorsqu'on installe SELinux sur un système qui ne l'utilisait pas avant: le "marquage" du système de fichier.

Les informations "*identité:rôle:type*" ne sont pas indiquées par défaut sur le système de fichier, il est nécessaire de les ajouter. SELinux fournit le moyen de donner le "label" aux fichiers.

Pour faire correspondre un label aux fichiers, les packages de politique de sécurité contiennent les informations sur la façon dont les fichiers doivent être marqués en fonction de leur chemin d'accès. Voici par exemple quelques unes des règles utilisées pour le label des fichiers de PostgreSQL:

```
/etc/postgresql(/.*)?                   gen_context(system_u:object_r:postgresql_etc_t,s0)
/usr/bin/initdb                 --      gen_context(system_u:object_r:postgresql_exec_t,s0)
/usr/bin/postgres               --      gen_context(system_u:object_r:postgresql_exec_t,s0)

/usr/lib/pgsql/test/regres(/.*)?        gen_context(system_u:object_r:postgresql_db_t,s0)
```

Dans la plupart des cas, les problèmes que l'on rencontre avec SELinux ne sont pas liés aux règles, mais aux labels donnés aux fichiers. En effet, les politiq de sécurité sont souvent disponible mais certaines configurations particulières nécessitent de changer les labels.

Prenons par exemple le cas où vous voudriez que apache utilise le dossier /home/www plutôt que /var/www. Pour fonctionner avec SELinux, le serveur apac nécessite des fichiers avec le label *httpd_sys_content_t*. Il faut donc changer les label de /home/www et tout fonctionne correctement.

Pour manipuler les labels, il existe plusieurs commande. Pour restaurer les labels du système de fichier à partir des informations des modules (ce qui écrase donc les modifications que vous auriez pu faire), la commande la plus utile est *fixfiles*:

```
# fixfiles relabel

    Files in the /tmp directory may be labeled incorrectly, this command
    can remove all files in /tmp.  If you choose to remove files from /tmp,
    a reboot will be required after completion.

    Do you wish to clean out the /tmp directory [N]?
```

Pour restaurer les labels d'un fichier ou d'une partie du système de fichier, la commande *restorecon* est la plus pratique:

```
# restorecon -R /etc/ssh
```

Finalement, pour changer à la main le label d'un fichier ou d'une arborescence, il faut utiliser la commande *chcon*:

```
chcon -t user_home_t /home/ikipou/SECURITY
```

Nous verrons à la section suivante la commande semanage qui permet de changer les labels d'une façon qui, contrairement à chcon, resiste au remarquag système de fichier.

Personnalisation

Vous l'avez compris, écrire des règles est souvent difficile et peut conduire à réduire la sécurité si cela n'est pas fait proprement. Néanmoins, il est rare de d le faire.

Nous avons déjà vu que la plupart des problèmes sont réglés par le changement du label des fichiers afin qu'il soit approprié aux règles. Pour ceux qui ne s pas liés aux fichiers, SELinux propose des méchanismes pour personnaliser les règles.

Les booléens

Pour beaucoup de cas communs, il est possible d'activer des droits particuliers à l'aide de booléens. Les booléens sont des paramètres que l'on peut active désactiver selon l'utilisation particulière qu'on fait des services protégés. Ils peuvent être listés à l'aide de la commande *getsebool -a*:

```
# getsebool -a
allow_cvs_read_shadow --> off
allow_daemons_use_tty --> off
allow_execheap --> off
allow_execmem --> off
allow_execmod --> off
...
```

Voyons un exemple avec un serveur ftp. Par défaut, SELinux ne permet pas aux utilisateurs anonymes de créer des fichiers. Pour activer cela, il suffit d'activ booléen correspondant:

```
# getsebool -a
allow_ftpd_anon_write --> off
# setsebool allow_ftpd_anon_write 1
```

semanage

Pour des changements sur la façon dont les règles sont utilisées, la commande *semanage* permet des modifications sur les informations utilisées par SELinu domaine/contexte de sécurité).

Les possibilités de *semanage* sont trop nombreuse pour les présenter ici (voyez la page de manuel pour d'autres utilisation). Voyons tout de même deux cas d'utilisation courant de la commande: le labeling et l'ajout de ports.

Labeling propre

Comme nous l'avons vu, chaque fichier possède un label qui lui a été donné à partir des règles des modules de sécurités. Ces labels peuvent être modifiés l'aide de la commande chcon. Néanmoins, que se passe t'il si on décide de restaurer les labels avec fixfiles ou restorecon? Tout les labels personnalisés soi perdus.

La solution est d'ajouter des règles pour les labels directement à la politique noyau courante. On ajoute une règle de marquage avec *semanage*. Par exemp pour utiliser /home/www comme répertoire racine pour apache (comme vu précédemment), on peut utiliser:

```
# semanage fcontext -a -t httpd_sys_content_t /home/www
```

Une telle modification sera toujours effective après un relabel effectué par fixfiles ou restorecon. Cette méthode est donc à préférer à chcon pour tout les changements non triviaux effectué sur les labels.

Ports par défaut

Comme cela a été expliqué, la sécurité de SELinux ne se limite pas qu'aux fichiers, mais à de nombreux objets manipulés par le noyau. En particulier, les soc
sont aussi vérifié par SELinux, et il n'est pas question pour une application d'ouvrir un socket à un port qui n'a pas été autorisé.

Il arrive néanmoins fréquemment que l'on veuille faire fonctionner certains services sur des ports non standard. Pour se faire, il suffit d'ajouter le bon label a
ports nécessaires à l'aide de semanage. Voyons deux exemple:

```
semanage port -a -t httpd_t -p tcp 81
semanage port -a -t openvpn_t -p udp 6865
```

Les commandes

Revoyons les différentes commandes utiles pour gérer SELinux:

ps -Z
: L'option -Z permet d'afficher le contexte de sécurité en plus des informations habituelles affichées avec ps.

ls -Z
: L'option -Z permet d'afficher les droits classique et le contexte de sécurité des fichiers.

id -Z
: Affiche le contexte de sécurité de l'utilisateur courant.

newrole
: Permet de changer de rôle et/ou de type

sestatus
: Affiche les informations d'état de SELinux (actif, mode de fonctionnement, etc)

semodule
: Permet de gérer les modules de sécurité, les "policy package".

fixfiles
: Permet de vérifier et de restaurer les labels du système de fichier

restorecon
: Permet de restaurer les labels d'une partie du système de fichier

chcon
: Permet de spécifier manuellement les informations de sécurité des fichiers

getenforce
: Retourne le mode de fonctionnement courant de SELinux (permissive, enforcing).

setenforce
: Change le mode de fonctionnement de SELinux

semanage
: Permet de configurer les règles actives pour personnaliser certains paramètres de sécurités

getsebool
: Retourne la liste des booléens utilisables pour personnaliser la sécurité des services.

setsebool

Permet de fixer la valeur des booléens utilisé pour personnaliser la sécurité des services

audit2allow

Utiliser les logs des permissions refusées pour écrire des politiques de sécurité appropriées et ne plus générer ces erreurs.

checkmodule et semodule_package

Ces deux commandes sont utilisées pour compiler des règles de sécurité pour faire un "policy package" qui est ensuite chargé avec semodule.

Conclusion

SELinux se révèle assez simple à comprendre et à utiliser. La complexité réside dans l'écriture des règles de sécurité, mais dans la plupart des cas, les règle standards suffisent largement tel quelle.

Le modèle de sécurité proposé par SELinux est suffisamment puissant pour se protéger des nombreuses failles provenant des logiciels, tout en restant relativement simple. Il s'agit d'un bon complément aux systèmes de sécurité basés sur les utilisateurs.

Pour aller plus loin

Il y a de nombreux livres, articles etc sur SELinux mais le mieux est peut être de commencer par l'activer sur votre distribution préférée et de commencer à jo avec.

Un élément intéressant pour l'apprentissage est de lire les politiques de sécurité appliquées sur des démons pas trop complexe. Les sources des politiques sécurité sont souvent dans un package séparé, tel que selinux-policy-refpolicy-src.

Cet article s'est focalisé sur SELinux pour les serveurs et les outils présentés sont les outils en console. Il existe de nombreux outils graphique tel que apol, polgengui, et les outils spécifiques à votre distribution et environnement de bureau.

Références

- Anatomy of Security-Enhanced Linux - Courte présentation de l'implémentation de SELinux dans le noyau
- The UnOfficial SELinux FAQ

Objectifs liés

- Mettre en place la sécurité de l'hôte
- Sécuriser les serveurs FTP
- Sécuriser un serveur DNS
- Mettre en place la sécurité au niveau utilisateur