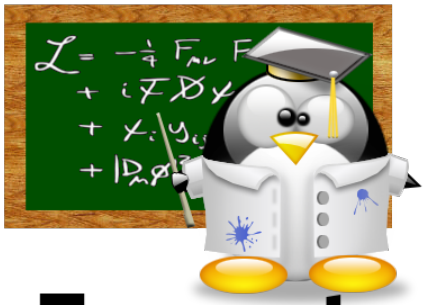


Formatux

Sécurisation SELinux

Table des matières

1. Généralités 2
1.1. Le contexte SELinux 2
Le contexte SELinux des processus standards 4
Le contexte SELinux des processus importants 4
2. Gestion 6
2.1. Administrer les objets de type booléens 6
3. Mode de fonctionnement 8
3.1. Le fichier /etc/sysconfig/selinux 9
4. Les jeux de règles (Policy Type) : 10
5. Contexte 11
5.1. Aller plus loin avec SELinux 12
Exemple de configuration 13

Avec l'arrivée du noyau en version 2.6, un nouveau système de sécurité a été introduit pour fournir un mécanisme de sécurité supportant les stratégies de sécurité de contrôle d'accès.

Ce système s'appelle SELinux (Security Enhanced Linux) et a été créé par la NSA (National Security Administration) pour implémenter dans les sous-systèmes du noyau Linux une architecture robuste de type Mandatory Access Control (MAC).

Si, tout au long de votre carrière, vous avez soit désactivé ou ignoré SELinux, ce chapitre sera pour vous une bonne introduction à ce système qui travaille dans l'ombre de Linux pour limiter les privilèges ou supprimer les risques liés à la compromission d'un programme ou d'un démon.

Avant de débuter, sachez que SELinux est essentiellement à destination des distributions RHEL, bien qu'il soit possible de le mettre en œuvre sur d'autres distributions comme Debian (mais bon courage !). Les distributions de la famille Debian intègrent généralement le système AppArmor, qui pour sa part, fonctionne relativement différemment de SELinux.

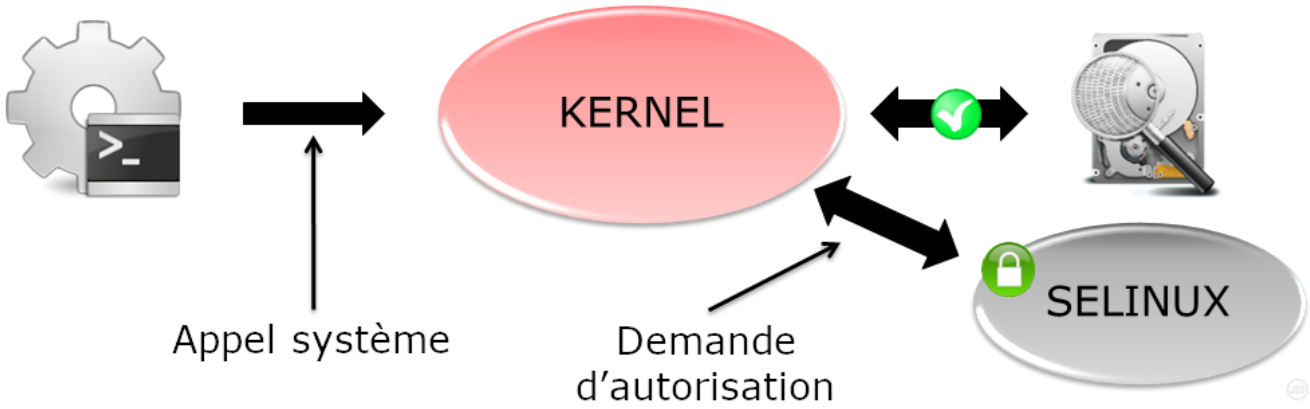
Chapitre 1. Généralités

SELinux (Security Enhanced Linux ou Amélioration de la Sécurité Linux) est un système de contrôle d'accès obligatoire (Mandatory Access Control).

Avant l'apparition des systèmes MAC, la sécurité standard de gestion d'accès reposait sur des systèmes DAC (Discretionary Access Control). Une application, ou un démon, fonctionnait avec des droits UID ou SUID (Set Owner User Id), ce qui permettait d'évaluer les permissions (sur les fichiers, les sockets, et autres processus...) en fonction de cet utilisateur. Ce fonctionnement ne permet pas de limiter suffisamment les droits d'un programme qui est corrompu, ce qui lui permet potentiellement d'accéder aux sous-systèmes du système d'exploitation.

Un système MAC renforce la séparation entre les informations sur la confidentialité et l'intégrité du système pour obtenir un système de confinement. Le système de confinement est indépendant du système de droits traditionnels et il n'existe pas de notion de superutilisateur.

À chaque appel système, le noyau interroge SELinux pour savoir s'il autorise l'action a être effectuée.



SELinux utilise pour cela un ensemble de règles (en anglais policy). Un ensemble de deux jeux de règles standards (targeted et strict) est fourni et chaque application fournit généralement ses propres règles.

1.1. Le contexte SELinux

Le fonctionnement de SELinux est totalement différent des droits traditionnels Unix.

Le contexte de sécurité SELinux est défini par le trio **identité+rôle+domaine**.

L'identité d'un utilisateur dépend directement de son compte linux. Une identité se voit attribué un ou plusieurs rôles, mais à chaque rôle correspond un domaine et un seul. C'est en fonction du domaine du contexte de sécurité (et donc du rôle...) que sont évalués les droits d'un utilisateur sur une ressource.

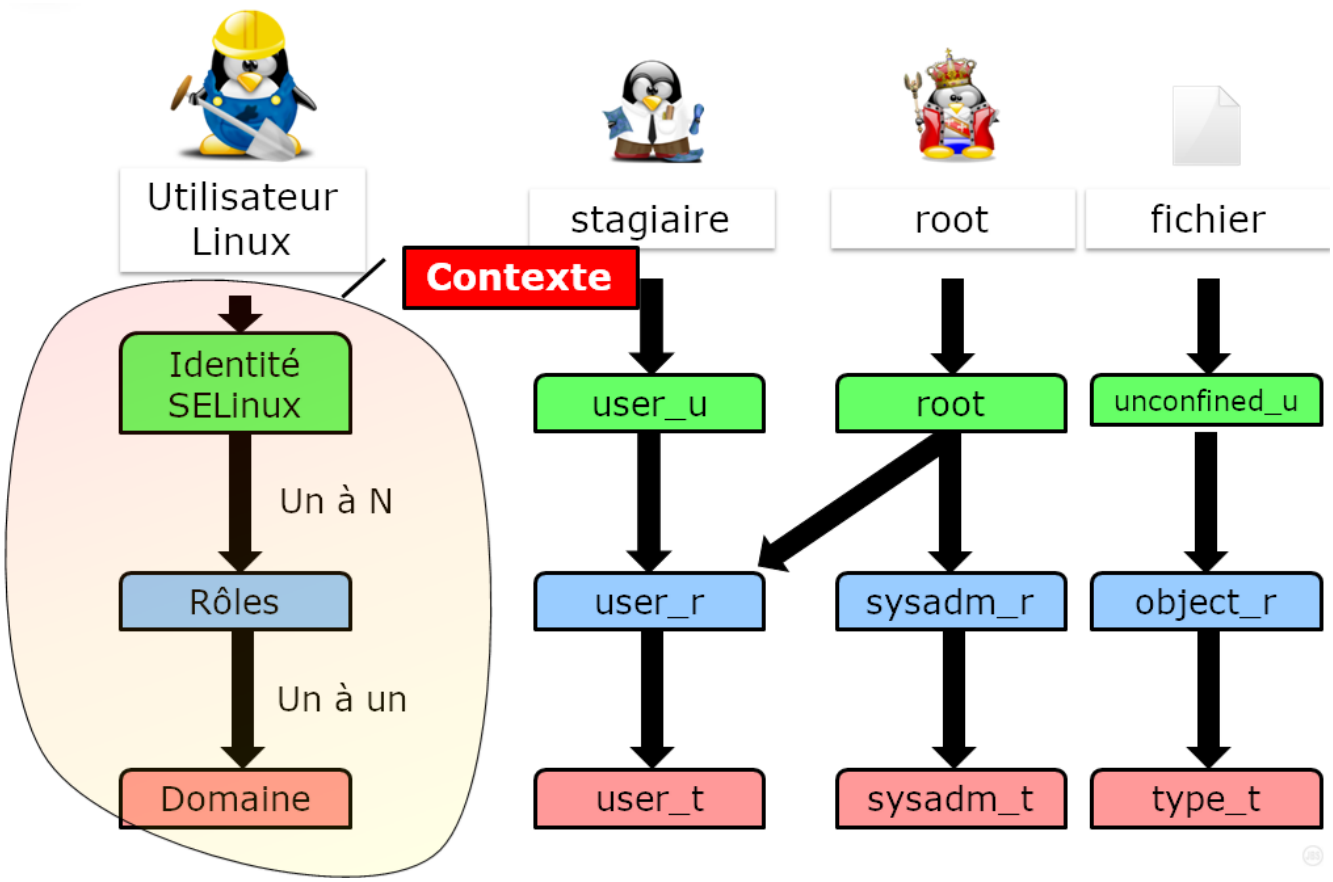


Figure 1. Le contexte SELinux

```
Les termes "domaine" et "type" sont similaires, typiquement "domaine" est utilisé
lorsque l'on se réfère à un processus tandis que "type" réfère à un objet.
La convention de nommage est :  _u user ; _r  rôle ; _t type.
```

Le contexte de sécurité est attribué à un utilisateur au moment de sa connexion, en fonction de ses rôles. Le contexte de sécurité d'un fichier est quant à lui défini par la commande **chcon** (change context) que nous verrons plus tard dans la suite de ce chapitre.

Considérez les pièces suivantes du puzzle SELinux :

- Les sujets
- Les objets
- Les stratégies
- Le mode

Quand un sujet (une application par exemple) tente d'accéder à un objet (un fichier par exemple), la partie SELinux du noyau Linux interroge sa base de données de stratégies. En fonction du mode de fonctionnement, SELinux autorise l'accès à l'objet en cas de succès, sinon il enregistre l'échec dans le fichier /var/log/messages.

Le contexte SELinux des processus standards

Les droits d'un processus dépendent de son contexte de sécurité.

Par défaut, le contexte de sécurité du processus est défini par le contexte de l'utilisateur (identité + rôle + domaine) qui le lance.

Un domaine étant un type (au sens SELinux) spécifique lié à un processus et hérité (normalement) de l'utilisateur qui l'a lancé, ses droits s'expriment en termes d'autorisation ou de refus sur des types (liés à des objets) :

Un processus dont le contexte a le domaine de sécurité D peut accéder aux objets de type T.

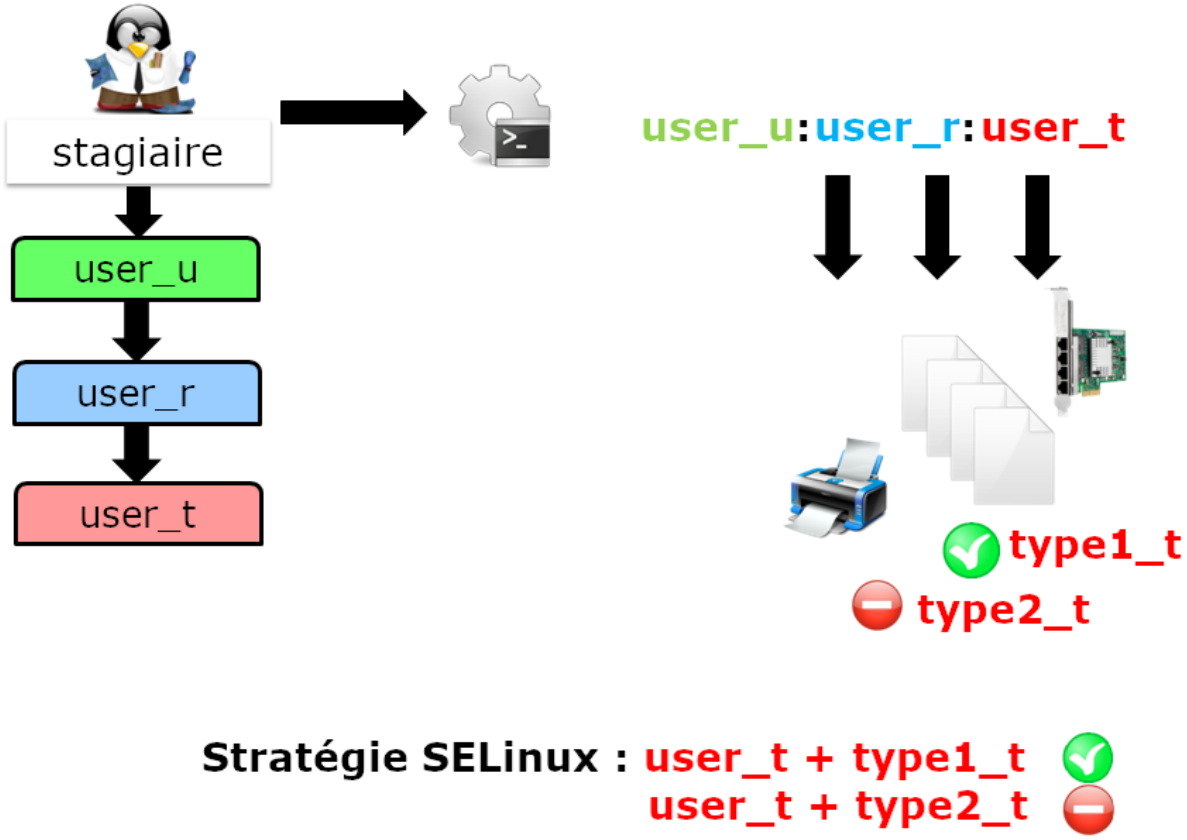


Figure 2. Le contexte SELinux d'un processus standard

Le contexte SELinux des processus importants

La plupart des programmes importants se voient attribuer un domaine dédié.

Chaque exécutable est étiqueté avec un type dédié (ici **sshd_exec_t**) qui fait basculer le processus associé automatiquement dans le contexte **sshd_t** (au lieu de **user_t**).

Ce mécanisme est essentiel puisqu'il permet de restreindre au plus juste les droits d'un processus.

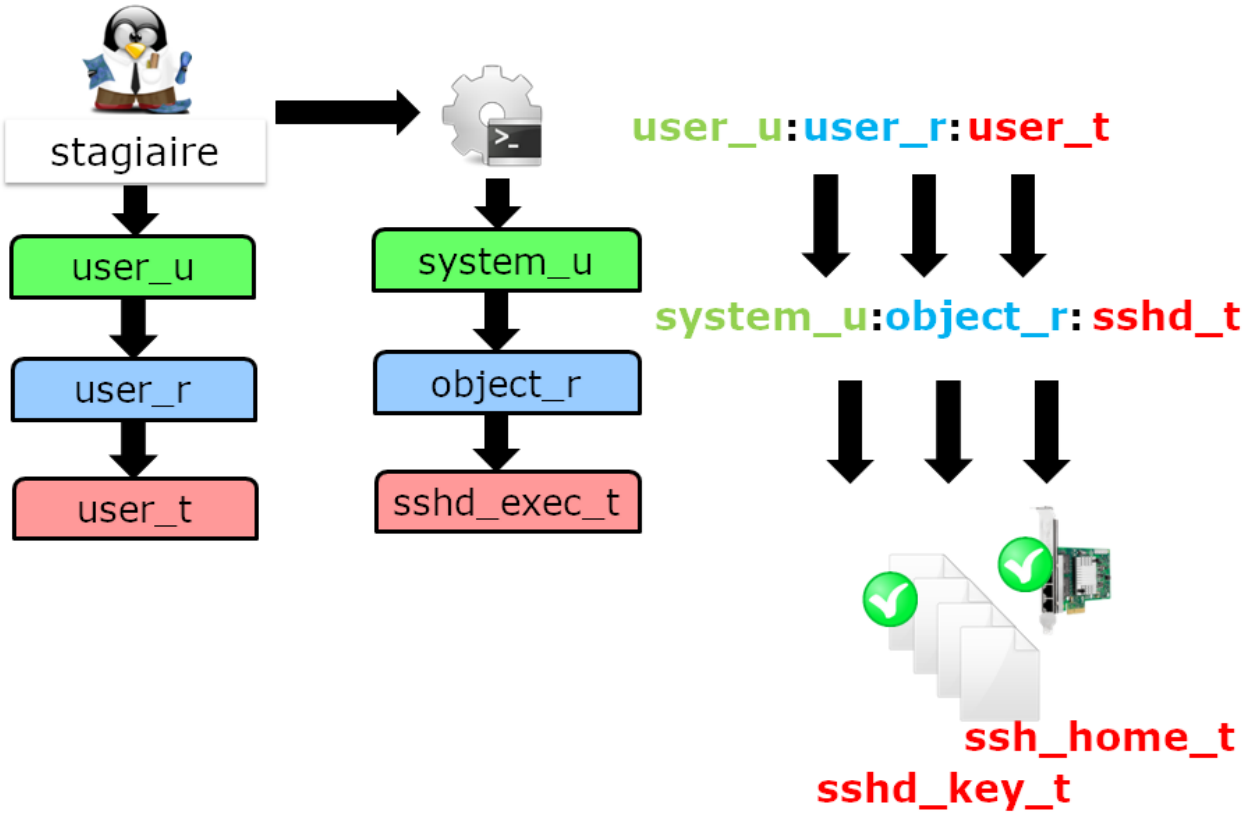


Figure 3. Le contexte SELinux d'un processus important - exemple de sshd

Chapitre 2. Gestion

La commande **semanage** (SE manage)permet d'administrer les règles SELinux.

Syntaxe de la commande semanage

```
semanage [type_d_objet] [options]
```

Exemple :

```
[root]# semanage boolean -l
```

Table 1. Options de la commande semanage

Options	Observations
-a	Ajoute un objet
-d	Supprime un objet
-m	Modifie un objet
-l	Liste les objets

La commande **semanage** n'est pas installée par défaut sous CentOS.

Sans connaître le paquet qui fournit cette commande, il convient de rechercher son nom avec la commande :

```
[root]# yum provides */semanage
```

puis l'installer :

```
[root]# yum install policycoreutils-python
```

2.1. Administrer les objets de type booléens

Les booléens permettent le confinement des processus.

Syntaxe de la commande semanage boolean

```
semanage boolean [options]
```

Pour lister les booléens disponibles :

```
[root]# semanage boolean –l
Booléen SELinux    State Default  Description
...
httpd_can_sendmail (fermé,fermé)  Allow http
  daemon to send mail
...
```

La commande **setsebool** permet de modifier l'état d'un objet de type booléen :

Syntaxe de la commande setsebool

```
setsebool [-PV] boolean on|off
```

Exemple :

```
[root]# setsebool -P httpd_can_sendmail on
```

Table 2. Options de la commande setsebool

Options	Observations
-P	Modifie la valeur par défaut au démarrage (sinon uniquement jusqu'au reboot)
-V	Supprime un objet

La commande semanage permet d'administrer les objets de type port :

Syntaxe de la commande semanage port

```
semanage port [options]
```

Exemple : autoriser le port 81 aux processus du domaine httpd

```
[root]# semanage port -a -t http_port_t -p tcp 81
```

Chapitre 3. Mode de fonctionnement

SELinux propose trois modes de fonctionnement :

- Enforcing (Appliqué)

Mode par défaut pour les Linux RedHat. Les accès seront restreints en fonction des règles en vigueur.

- Permissive (Permissif)

Les règles sont interrogées, les erreurs d'accès sont journalisées, mais l'accès ne sera pas bloqué.

- Disabled (Désactivé)

Rien ne sera restreint, rien ne sera journalisé.

Par défaut, la plupart des systèmes d'exploitation sont configurés avec SELinux en mode Enforcing.

La commande **getenforce** retourne le mode de fonctionnement en cours

Syntaxe de la commande getenforce

```
getenforce
```

Exemple :

```
[root]# getenforce
Enforcing
```

La commande **sestatus** retourne des informations sur SELinux

Syntaxe de la commande sestatus

```
sestatus
```

Exemple :

```
[root]# sestatus
SELinux status:         enabled
SELinuxfs mount:            /selinux
Current mode:           enforcing
Mode from config file : enforcing
Policy version:         24
Policy from config file:    targeted
```

La commande **setenforce** modifie le mode de fonctionnement en cours :

Syntaxe de la commande setenforce

```
setenforce 0|1
```

Passer SELinux en mode permissif :

```
[root]# setenforce 0
```

3.1. Le fichier /etc/sysconfig/selinux

Le fichier /etc/sysconfig/selinux permet de modifier le mode de fonctionnement de SELinux.

Désactiver SELinux se fait à vos risques et périls ! Il est préférable d'apprendre le fonctionnement de SELinux plutôt que de le désactiver systématiquement !

Modifier le fichier /etc/sysconfig/selinux

```
SELINUX=disabled
```

Redémarrer le système :

```
[root]# reboot
```

Attention au changement de mode SELinux !

En mode permissif ou désactivé, les nouveaux fichiers créés ne porteront aucune étiquette.

Pour réactiver SELinux, il faudra repositionner les étiquettes sur l'intégralité de votre système.

Labéliser le système entièrement :

```
[root]# touch /.autorelabel
[root]# reboot
```

Chapitre 4. Les jeux de règles (Policy Type) :

SELinux fournit deux types de règles standards :

- Targeted : seuls les démons réseaux sont protégés (dhcpd, httpd, named, nscd, ntpd, portmap, snmpd, squid et syslogd)
- Strict : tous les démons sont protégés

Chapitre 5. Contexte

L'affichage des contextes de sécurité se fait avec l'option -Z. Elle est associée à de nombreuses commandes :

Exemples :

```
[root]# id -Z   # le contexte de l'utilisateur
[root]# ls -Z   # ceux des fichiers courants
[root]# ps -eZ  # ceux des processus
[root]# netstat –Z # ceux des connexions réseaux
[root]# lsof -Z # ceux des fichiers ouverts
```

La commande **matchpathcon** retourne le contexte d'un répertoire.

Syntaxe de la commande matchpathcon

```
matchpathcon répertoire
```

Exemple :

```
[root]# matchpathcon /root
 /root  system_u:object_r:admin_home_t:s0

[root]# matchpathcon /
 /      system_u:object_r:root_t:s0
```

La commande **chcon** modifie un contexte de sécurité :

Syntaxe de la commande chcon

```
chcon [-vR] [-u USER] [–r ROLE] [-t TYPE] fichier
```

Exemple :

```
[root]# chcon -vR -t httpd_sys_content_t /home/SiteWeb
```

Table 3. Options de la commande chcon

Options	Observations
-v	Passe en mode verbeux
-R	Applique la récursivité
-u,-r,-t	S'applique à un utilisateur, un rôle ou un type

La commande **restorecon** restaure le contexte de sécurité par défaut :

Syntaxe de la commande restorecon

```
restorecon [-vR] répertoire
```

Exemple :

```
[root]# restorecon -vR /home/SiteWeb
```

Table 4. Options de la commande restorecon

Options	Observations
-v	Passe en mode verbeux
-R	Applique la récursivité

La commande **audit2why** indique la cause d'un refus SELinux :

Syntaxe de la commande audit2why

```
audit2why [-vw]
```

Exemple :

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1|audit2why
```

Table 5. Options de la commande audit2why

Options	Observations
-v	Passe en mode verbeux
-w	Traduit la cause d'un rejet par SELinux et propose une solution pour y remédier (option par défaut)

5.1. Aller plus loin avec SELinux

La commande **audit2allow** crée à partir d'une ligne d'un fichier "audit" un module pour autoriser une action SELinux :

Syntaxe de la commande audit2allow

```
audit2allow [-mM]
```

Exemple :

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1|audit2allow -M
MonModule_Local
```

Table 6. Options de la commande audit2allow

Options	Observations
-m	Crée juste le module (*.te)
-M	Crée le module, le compile et le met en paquet (*.pp)

Exemple de configuration

Après l'exécution d'une commande, le système vous rend la main mais le résultat attendu n'est pas visible : aucun message d'erreur à l'écran.

- **Étape 1** : Lire le fichier journal sachant que le message qui nous intéresse est de type AVC (SELinux), refusé (denied) et le plus récent (donc le dernier).

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1
```

Le message est correctement isolé mais ne nous est d'aucune aide.

- **Étape 2** : Lire le message isolé avec la commande audit2why pour obtenir un message plus explicite pouvant contenir la solution de notre problème (typiquement un booléen à positionner).

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1|audit2why
```

Deux cas se présentent : soit nous pouvons placer un contexte ou renseigner un booléen, soit il faut passer à l'étape 3 pour créer notre propre contexte.

- **Étape 3** : Créer son propre module.

```
[root]# less /var/log/audit/audit.log|grep AVC|grep denied|tail -1|audit2allow -M
MonModule_Local
Generating type enforcement: MonModule_Local.te
Compiling policy: checkmodule -M -m -o MonModule_Local.mod MonModule_Local.te
Building package: semodule_package -o MonModule_Local.pp -m MonModule_Local.mod

[root]# semodule -i MonModule_Local.pp
```