Sécurité sous Linux

S.berraho@emsi.ma
Berraho.sanae@gmail.com

- Introduction
- ACLs: Liste de contrôle d’accès
- Droits spéciaux: setuid, setgid, sticky bit
- SELinux

Autorisations traditionnelles sous Linux: rappel

- Les systèmes Linux utilisent un modèle de **contrôle d'accès** basé sur des **autorisations de fichiers traditionnelles**.
- Chaque fichier ou répertoire est associé à **un propriétaire** et à **un groupe**, auxquels sont attribuées des autorisations spécifiques, représentées par les trois catégories : **lecture** (r), **écriture** (w), et **exécution** (x)

- Utilisateur propriétaire (**u**)
- Groupe propriétaire (**g**)
- Autres utilisateurs (**o**)

- Lecture (**r**)
- Écriture (**w**)
- Exécution (**x**)

```
[bsanae@192 ~]$ ls -l Fichier.txt
-rw-rw-r--. 1 bsanae bsanae 22 Dec 20 16:53 Fichier.txt
[bsanae@192 ~]$
```

Autorisations traditionnelles sous Linux: limitations

- Seul le **propriétaire** du fichier et les membres du **groupe** associé ont des **autorisations spécifiques**.
- Les **autres utilisateurs** sont limités à des **autorisations génériques**, telles que la lecture seule.
- **Difficulté** à accorder des autorisations spécifiques à un utilisateur **sans le rendre propriétaire du fichier**.
- Les changements d'autorisation s'appliquent **de manière uniforme** à tous les membres du groupe
- Partager un fichier avec **un utilisateur spécifique** tout en maintenant les restrictions pour d'autres utilisateurs peut être **compliqué**.
- **Les autorisations s'appliquent uniformément** à tous les fichiers et sous-répertoires dans un répertoire, ce qui peut être trop large dans certaines situations.

Concept des ACLs

- Une **ACL**, ou ***Access Control List*** (**liste de contrôle d'accès**) est une **liste de permissions** sur un fichier, un répertoire ou une arborescence, **ajoutée aux permissions « classiques »** de ce fichier.
- Ces permissions concernent des **utilisateurs et/ou des groupes** définis.
- Au moyen des ACL, **on peut étendre le nombre d'utilisateurs et de groupes** ayant **des droits sur un même fichier**.
- Avec les **ACL**, on peut (entre autres) **ajouter à un fichier d'autres utilisateurs et groupes** et **définir leurs droits séparément**.
- Les **ACL** sont très utiles (voire **indispensables**) dans des environnements informatiques axés sur **le travail collaboratif et mutualisé**.

Avant les ACLs

- Dans certains cas n'affecter des droits que pour **3 catégories** d'utilisateurs (soi, son groupe et les autres) représente une **limitation importante**.
- Par exemple, on peut vouloir mettre **son devoir en lecture et écriture pour son binôme**, mais **pas pour le reste de la promo.**
- On atteint ce résultat avec **le système de permissions classiques** en **stockant le fichier de son devoir dans un répertoire façade** avec les droits suivants :

```
└── [drwx--x--x]  Binome
    └── [-rw-rw-r--]  devoir_983.tex
```

- Le dossier **Binome** est **accessible à tous**, mais **il est impossible d'en lister son contenu** (pas de droit de lecture). Il suffit alors de **communiquer à son binôme le nom du fichier partagé pour qu'il puisse le modifier.**

ACLs: Commandes

- Il existe deux commandes essentielles : l'une pour **manipuler l'ACL d'un fichier** (**setfacl**) et l'autre pour **la consulter** (**getfacl**).
- Les commandes traditionnelles **chmod** (et **chown**) **ne peuvent accéder aux ACL**
- Les **ACLs** Linux sont **supportées nativement** sur les distributions basées Red Hat.

```
[bsanae@localhost ~]$ yum list installed | grep acl
acl.x86_64                                  2.2.53-1.el8
                          @rhel-8-for-x86_64-baseos-rpms
libacl.x86_64                               2.2.53-1.el8
                          @rhel-8-for-x86_64-baseos-rpms
[bsanae@localhost ~]$
```

Commande setfacl: Modifier les droits

- La commande **setfacl** (***set file's ACL***) permet de **définir ou de modifier les ACL pour un fichier ou un répertoire spécifique**.
- Elle offre une **syntaxe flexible** pour **ajouter, modifier ou supprimer** des entrées d'ACL.
- **Exemple:**

```
# setfacl    -m    u:karim:rw    devoir.txt
```

- Cette commande **modifiera** (**-m**) l'ACL de **devoir.txt** en attribuant à l'utilisateur (**préfixe u:) karim** les droits **rw** et en lui **refusant le droit d'exécution** (qui n'a pas été mentionné dans la commande).

Commande setfacl: Modifier les droits

- Les principaux paramètres à connaître sont :

Préfixes :

u: droits pour un **utilisateur**, nommé ou désigné par son uid
g: droits pour un **groupe**, nommé ou désigné par son gid
o: droits pour **other**, le reste du monde

Permissions :

⇒ Elles sont codées dans l'ordre r, w et x.
⇒ On les remplace par - pour une interdiction explicite.
⇒ Ne pas mentionner un droit revient aussi à une interdiction

```
# setfacl  -m  u:karim:-w-  devoir.txt
```

=

```
# setfacl  -m  u:karim:w  devoir.txt
```

ACLs: Commandes

Commande setfacl: modifier les droits

- On peut construire des commandes plus complexes en enchaînant les entrées dans l'ACL :

```
# setfacl    -m     u:karim:rw,g:etudiants:r,o:---    devoir.txt
```

- Cette commande définit des permissions dans l'**ACL** de **devoir.txt** pour l'utilisateur **karim**, le groupe **etudiants** et **le reste du monde**.
- La commande fonctionne bien sûr aussi de **manière récursive** (**option -R**) :

```
# setfacl    -Rm     u:karim:rw    Devoirs
```

- Cette commande **modifie l'ACL** de **tous les fichiers situés sous Devoirs/** en attribuant une permission de **lecture et d'écriture** à l'utilisateur **karim**.

ACLs: Commandes

Commande setfacl: Manipulation

1. Créer un fichier **~/devoir** et y écrire quelque chose/
2. Retirer **les droits de lecture et écriture pour tout le monde** (à part vous) avec la méthode classique (**chmod**).
3. Créer un utilisateur avec le login **karim**. Attribuer un mot de passe à ce compte.
4. Ouvrir un autre terminal. Se connecter en tant que **karim**.
5. L'utilisateur **karim** pourrat-il ouvrir et modifier le fichier **devoir**?
6. Revenir à votre terminal. Avec la commande **setfacl**, donner les **droits de lecture et écriture** à l'utilisateur **karim**.
7. Un **ls -l ~/devoir** affiche un **' +'** à la suite des droits montrant que des **ACL ont été ajoutés au fichier**.
8. Donner la permission d'accéder à votre répertoire personnel aux autres . (**chmod +x ~)**
9. Ouvrir un autre terminal. Se connecter en tant que **karim.**
10. L'utilisateur **karim** pourra-t-il ouvrir et modifier le fichier devoir?

Commande setfacl: Retirer les droits

- L'option **-b** de la commande **setfacl** est utilisée pour **supprimer toutes les ACL d'un fichier ou d'un répertoire**. Cela signifie qu'elle réinitialise complètement les autorisations, en supprimant toutes les entrées d'ACL existantes.

```
$ setfacl    -b  fichier.txt
```

- Cette commande **supprime toutes les ACL associées au fichier** **fichier.txt**. Les autorisations par défaut basculent vers les permissions de fichier standard (**propriétaire, groupe, autres**).

```
$ setfacl    -x  u:karim,g:etudiants fichier.txt
```

- Cette commande **retire les permissions propres** à **karim** et au **groupe étudiants.**

Commande setfacl: Retirer les droits: Manipulation

- Retirer les ACL sur **~/devoir** et assurez-vous qu'elles ont disparues :

  ⇒ Avec **ls -l ~/devoir** : il n'y a plus le **' +'**

  ⇒ L'utilisateur **karim** ne peut plus lire le fichier

ACLs: Commandes

Commande getfacl:

- Cette commande suivie d'un nom de fichier affiche l'ACL de ce fichier (**get file's ACL** « récupérer l'ACL du fichier »).

Manipulation:

1. Créer un fichier projet.txt et utiliser setfacl pour changer ses droits :
   - ⇒ Lecture pour l'utilisateur karim
   - ⇒ Rien pour le groupe et les autres
2. Afficher les droits avec **getfacl**

```
$ getfacl    projet.txt
```

ACLs par défaut

- Les **ACLs par défaut** permettent de **donner des permissions ACL en héritage pour tout sous-répertoire ou fichier créé dans un répertoire**.
- Toutefois, ces **ACLs par défaut** ne **s'appliquent pas aux objets déjà présents dans le répertoire**
- **Exemple:** pour empêcher tous **les autres** en termes de permissions pour tout nouveau fichier ou sous-répertoire créé :

```
$ setfacl    -m  d:o::-   repertoire
```

- Les permissions ACL par défaut d'un répertoire (**d:**) s'annulent par **setfacl -k**.

ACLs par défaut: Manipulation

1. Créer un nouveau répertoire appelé "**acl_demo**" dans votre répertoire personnel
2. Utiliser la commande **getfacl** pour afficher les ACL par défaut de ce répertoire
3. Créer un groupe « **utilisateurs** »
4. Utiliser la commande **setfacl** pour ajouter une ACL par défaut au répertoire. Donner tous les droits au groupe "**utilisateurs**" en tant qu'ACL par défaut.
5. Vérifier que l'ACL par défaut a été ajoutée avec la commande **getfacl**.
6. Créer quelques fichiers dans le répertoire "**acl_demo**"
7. Utiliser **getfacl** pour afficher les ACL des fichiers que vous venez de créer dans le répertoire "**acl_demo**".
8. Remarquer comment les ACL par défaut ont été appliquées aux nouveaux fichiers.
9. Utiliser la commande **setfacl** pour supprimer l'ACL par défaut que vous avez précédemment ajoutée.
10. Vérifier que l'ACL par défaut a été supprimée avec **getfacl**
11. Créer un nouveau répertoire à l'intérieur de "**acl_demo**" pour observer comment les ACL par défaut sont héritées.

Sécurité sous Linux: Droits spéciaux

Introduction

- **Les droits spéciaux** sous Linux sont des **droits supplémentaires** qui peuvent être accordés à un fichier ou à un répertoire.
- Ils sont utilisés pour **modifier le comportement normal** des fichiers et des répertoires.
- Il existe trois droits spéciaux :

  ⇒ **Setuid:** Ce droit permet à **un utilisateur non root** d'exécuter un programme avec les droits du **propriétaire du programme**.

  ⇒ **Setgid:** Ce droit permet à **un utilisateur non root** d'effectuer des opérations sur un répertoire avec **les droits du groupe propriétaire du répertoire**.

  ⇒ **Sticky bit:** Ce droit empêche **les utilisateurs non root** de **supprimer ou de modifier** les fichiers d'un répertoire

Le droit setuid

- Le droit **setuid** est utilisé pour **exécuter des programmes avec les droits du propriétaire du programme**.
- Cela peut être utile pour **exécuter des programmes qui nécessitent des privilèges élevés**, tels que des programmes d'administration système.
- **Par exemple**: le programme **sudo** utilise le droit **setuid** pour **permettre aux utilisateurs non root** d'exécuter des commandes avec les droits du **super-utilisateur**.

```
[bsanae@192 ~]$ ls -l /usr/bin/sudo
---s--x--x. 1 root root 165536 Jan 16  2023 /usr/bin/sudo
[bsanae@192 ~]$
```

Le droit setuid

setuid: Activation du droit

- Pour **activer** le droit **setuid** en mode symbolique:

```
$ chmod    u+s  mon_programme
```

- Vous pouvez utiliser la commande chmod suivie du code de permission **4**xxx, où xxx représente les bits de permission pour le fichier.

```
$ chmod    4xxx  mon_programme
```

- **Désactiver** le droit **setuid**:

```
$ chmod    u-s  mon_programme
```

ou

```
$ chmod    0xxx  mon_programme
```

setuid: Manipulation

1. En utilisant la commande **which**, Identifier l'emplacement de la commande **passwd** sur votre système.
2. Utiliser **ls -l** pour afficher les droits du fichier de la commande **passwd**
3. Vérifier que **setuid** est activé
4. Se connecter en tant que utilisateur ordinaire. Changer votre propre mot de passe en utilisant la commande **passwd**
5. Supprimer **setuid** pour le fichier de la commande **passwd**
6. Utiliser **ls -l** pour vérifier
7. Essayer de changer de nouveau votre mot de passe. Vous ne devez pas pouvoir le faire.
8. Activer **setuid** pour le fichier de la commande **passwd**
9. Utiliser **ls -l** pour vérifier
10. Vérifier que vous pouvez désormais changer votre propre mot de passe.

Le droit setgid

- Sur les systèmes UNIX, **Setuid** n'a pas d'effet sur les répertoires. L'attribut **setuid** est **principalement conçu pour les fichiers exécutables.**
- Contrairement au bit **setuid**, le bit **setgid** a un effet **à la fois sur les fichiers et les répertoires.**
- le fichier qui a le bit **setgid** activé, lorsqu'il est exécuté, **au lieu de s'exécuter avec les privilèges du groupe de l'utilisateur** qui l'a démarré, **s'exécute avec ceux du groupe propriétaire du fichier.**
- Lorsqu'il est utilisé **sur un répertoire**, le bit **setgid** modifie le comportement standard de sorte que **le groupe de fichiers créés dans ledit répertoire ne sera pas celui de l'utilisateur qui les a créés, mais celui du parent**.
- Ceci est souvent utilisé pour faciliter le partage de fichiers **(les fichiers seront modifiables par tous les utilisateurs faisant partie dudit groupe**).

Le droit setgid

- Tout comme le **setuid**, le bit **setgid** peut facilement être repéré (dans ce cas sur un répertoire **/projet**) :

```
[root@localhost bsanae]# ls -ld /projet
drwxrwsr-x. 2 root collaborateurs 6 Dec 26 17:46 /projet
```

- Cette fois le **s** est présent à **la place du bit exécutable sur le secteur groupe.**

setgid: activation du droit:

```
# chmod  2xxx  /repertoire
```

ou

```
# chmod  g+s  /repertoire
```

setgid: désactivation du droit:

```
# chmod  0xxx  /repertoire
```

ou

```
# chmod  g-s  /repertoire
```

setgid: Manipulation

1. Connecter vous en tant que utilisateur ordinaire
2. Créer un groupe "**collaborateurs**",
3. Ajouter l'utilisateur courant au groupe "**collaborateur**'' .
4. Créer un répertoire appelé "**projet**" et attribuer-lui le groupe "**collaborateurs**".
5. Activer l'attribut **setgid** sur ce répertoire projet
6. Vérifier les permissions et le groupe du répertoire.
7. Créer un fichier à l'intérieur du répertoire "**projet**''
8. Vérifier les propriétés du fichier créé.

Le Sticky Bit

- Ce droit spécial, traduit en "**bit collant**", a surtout un rôle important sur les répertoires.
- Il réglemente le **droit w** sur le répertoire, en **interdisant à un utilisateur quelconque de supprimer un fichier dont il n'est pas le propriétaire.**
- Ce droit **noté symboliquement t** occupe par convention la place du **droit x** sur la catégorie **other** de ce répertoire, mais bien entendu il **ne supprime pas le droit d'accès x (s'il est accordé).**
- Si le **droit x n'est pas accordé** à la **catégorie other,** à la place de **t** c'est la lettre **T** qui apparaitra.
- Pour positionner ce droit :

```
# chmod    1xxx  /repertoire
```

ou

```
# chmod    +t  /repertoire
```

- Pour retirer ce droit:

```
# chmod    0xxx  /repertoire
```

ou

```
# chmod    -t  /repertoire
```

Sticky bit: Manipulation

1. Vérifier les permissions du répertoire **/tmp.** Le **bit sticky** est activé, ce qui signifie que seuls les propriétaires du répertoire et root peuvent supprimer des fichiers dans ce répertoire.
2. Créer deux utilisateurs « **user1** » et « **user2** ».
3. Créer sous **/tmp** un fichier « **fichier1** » en tant que **user1**
4. Se connecter en tant que « **user2** » et supprimer « **fichier1** ». Qu'est-ce que vous remarquez?
5. Supprimer temporairement le sticky bit pour **/tmp**
6. En tant que « **user2** » supprimer « **fichier1** ». Qu'est-ce que vous remarquez?
7. Réactivez le sticky bit pour **/tmp**

SELinux: Introduction

- **SELinux** (**Security-Enhanced Linux**) est une architecture de sécurité pour systèmes Linux® qui permet aux administrateurs de mieux contrôler les accès au système. C'est un modèle de sécurité que l'on peut ajouter au système standard de Linux afin **d'augmenter la sécurité** face aux diverses attaques que subit le système.
- Avec **SELinux**, on peut **configurer les accès de chaque processus** pour les **restreindre à un strict minimum**. Cela permet de rendre inexploitable certaines failles, et en cas de piratage, de limiter l'étendue des dégâts.
- Les systèmes Linux et UNIX utilisent généralement le **contrôle d'accès discrétionnaire** (ou **DAC**- **Discretionary Access Control**).
- **SELinux** est un exemple de système de **contrôle d'accès obligatoire** (ou **MAC**: **Mandatory Access Control**) pour Linux.

SELinux: Introduction

- **DAC: Discretionary Access Control**
  - ⇒ Les fichiers et processus ont des **propriétaires.**
  - ⇒ Le propriétaire d'un fichier peut être un **utilisateur, un groupe, ou autre**
  - ⇒ Les utilisateurs ont la possibilité de **modifier les autorisations d'accès à leurs propres fichiers.**
  - ⇒ L'utilisateur root dispose d'un **contrôle d'accès total.**

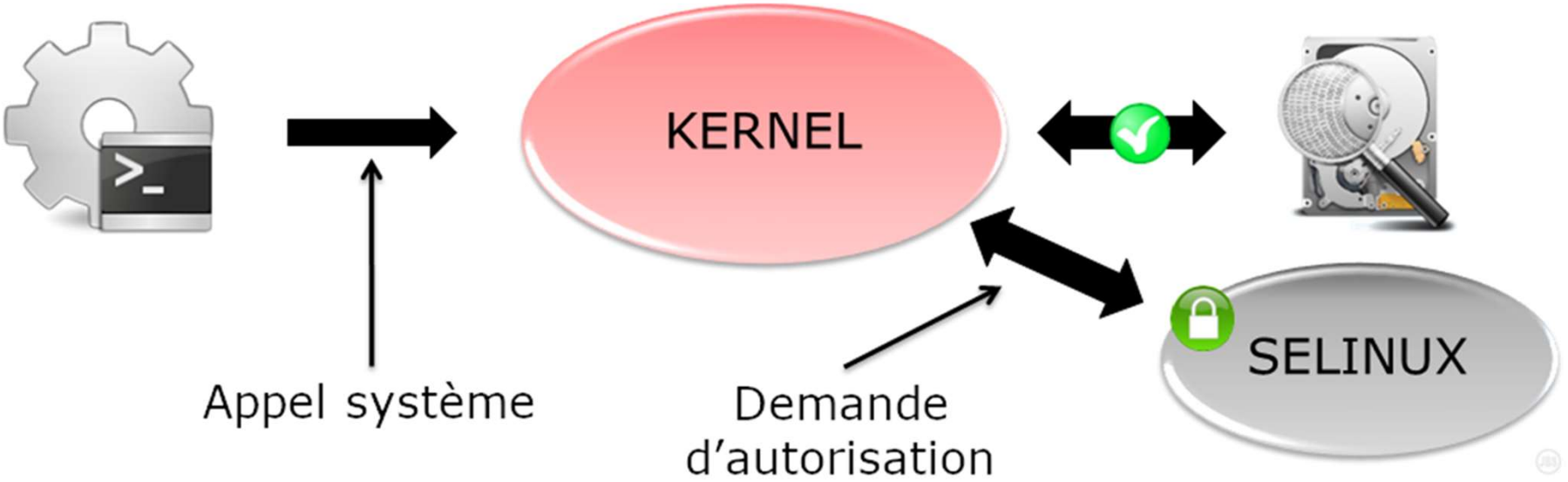
- **MAC: Mandatory Access Control**
  - ⇒ Des politiques d'accès sont **définies par l'administrateur**.
  - ⇒ Même si les paramètres du **système DAC** sont modifiés au niveau de votre répertoire principal, la sécurité du système sera préservée grâce à **une politique SELinux** mise en place pour **empêcher d'autres utilisateurs ou processus** d'accéder au répertoire.

SELinux: Principe

- **SELinux** est une extension du noyau Linux qui permet de **surveiller des processus en cours d'exécution**, en garantissant que ceux-ci respectent certaines règles. **SELinux** repose essentiellement sur **deux fondements** :

  ⇒ Les fichiers et les processus doivent être étiquetés avec un **contexte de sécurité** SELinux approprié.

  ⇒ Les processus surveillés par SELinux doivent respecter un certain nombre de **règles prédéfinies**.



SELinux: intérêt

- Imaginons que quelqu'un vient de pirater **apache**, et qu'il peut lancer des commandes arbitraires via ce processus. Le processus piraté de apache **fonctionne en root**.

```
[root@localhost bsanae]# ps -aux | grep httpd
root       15228  0.1  0.2 258132 11024 ?        Ss   11:03   0:00 /usr/sbin/httpd -DFOREGROUND
apache     15235  0.0  0.2 262832  8396 ?        S    11:03   0:00 /usr/sbin/httpd -DFOREGROUND
apache     15236  0.1  0.4 2696952 18204 ?       Sl   11:03   0:00 /usr/sbin/httpd -DFOREGROUND
apache     15237  0.1  0.3 2500280 12072 ?       Sl   11:03   0:00 /usr/sbin/httpd -DFOREGROUND
apache     15238  0.1  0.4 2500280 16152 ?       Sl   11:03   0:00 /usr/sbin/httpd -DFOREGROUND
```

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

SELinux: intérêt

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

```
[root@localhost bsanae]# ps -eZ | grep httpd
system_u:system_r:httpd_t:s0      15228 ?        00:00:00 httpd
```

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

```
[root@localhost bsanae]# ls -Z /etc/shadow
system_u:object_r:shadow_t:s0 /etc/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 dans 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 }

SELinux: modèle de sécurité

- Le système d'exploitation **contraint** les utilisateurs ou programmes dans leurs **accès aux ressources**:

  ⇒ **Les objets** sont des ressources système : fichiers, répertoires, sockets et autres périphériques,

  ⇒ **Les sujets** sont les process (commandes ou programmes).

  ⇒ **Sujets et objets** se voient attribué un contexte de sécurité.

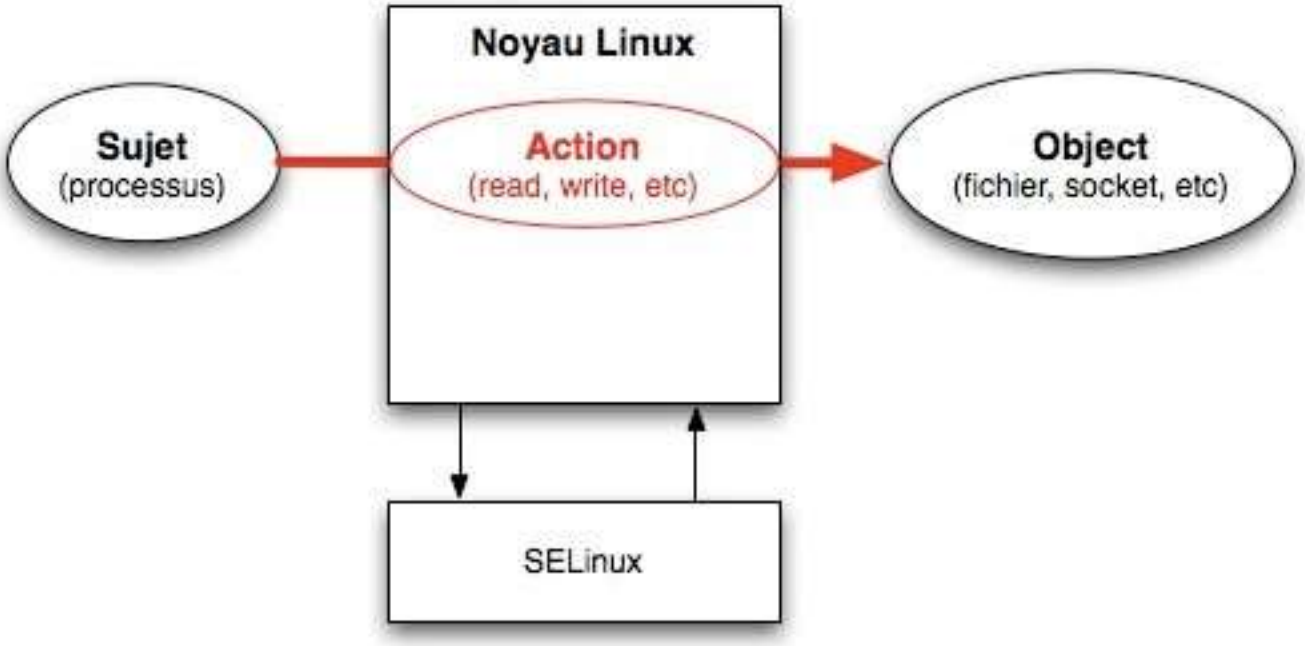
  ⇒ Le système détermine l'accès à un objet par un sujet selon la **politique de sécurité chargée.**

  ⇒ Le sujet ne peut pas modifier le contexte de sécurité d'un objet.

  ⇒ Ainsi un sujet **est confiné.**

SELinux: modèle de sécurité

- **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.



Modes de fonctionnement

- **SELinux** propose **trois modes** différents :
  - ⇒ Dans **le mode strict** (**Enforcing**), les accès sont restreints en fonction des règles SELinux en vigueur sur la machine.
  - ⇒ **Le mode permissif** (**Permissive**) peut être considéré comme un mode de débogage. Les règles **SELinux** sont interrogées, les erreurs d'accès sont enregistrées dans les logs, mais **l'accès ne sera pas bloqué.**
  - ⇒ Lorsque **SELinux** est **désactivé** (Disabled), l'accès n'est pas restreint, et rien n'est enregistré dans les logs.

Modes de fonctionnement: commandes

- La commande **getenforce** vous informe sur le mode en vigueur sur votre machine :

```
[root@localhost bsanae]# getenforce
Enforcing
[root@localhost bsanae]#
```

- La commande **setenforce** permet de basculer temporairement – **jusqu'au prochain redémarrage** – entre les modes strict (**Enforcing)** et permissif (**Permissive**) :

```
[root@localhost bsanae]# setenforce 0
[root@localhost bsanae]# getenforce
Permissive
[root@localhost bsanae]# setenforce 1
[root@localhost bsanae]# getenforce
Enforcing
[root@localhost bsanae]#
```

Modes de fonctionnement: Fichier

- Le **mode SELinux par défaut** est défini dans le fichier :
  /etc/selinux/config:

```
[root@localhost bsanae]# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

- **SELINUX** prendra une des trois valeurs **enforcing, permissive ou disabled**. Quant à SELINUXTYPE, on gardera la valeur par défaut targeted, qui garantit la surveillance des principaux services réseau.

Modes de fonctionnement

Attention !

- Lorsque **SELinux** est exécuté en **mode permissif** ou s'il est désactivé, les nouveaux fichiers crées ne porteront aucune étiquette. Lorsque SELinux sera réactivé, cela pourra poser des problèmes, et il vous faudra donc réétiqueter l'intégralité de votre système:

```
[root@localhost html]# touch /.autorelabel
[root@localhost html]# reboot
```

- Pour connaître l'état général (activé ou désactivé, mode courant, politique chargée, etc...:

```
[root@localhost ~]# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      33
[root@localhost ~]#
```

Consulter le contexte de sécurité

- Consulter les informations de l'actuel contexte SELinux avec l'argument **-Z** (cet argument est utilisé par **ps, ls**, ...:.

- **Exemple 1 : liste des contextes pour tous les processus**

```
Max kernel policy version:      33
[root@localhost ~]# ps -ef -Z
LABEL                             UID          PID    PPID  C STIME TTY          TIME CMD
system_u:system_r:init_t:s0       root           1       0  0 00:21 ?        00:00:16 /usr/lib/systemd/systemd --sw
system_u:system_r:kernel_t:s0     root           2       0  0 00:21 ?        00:00:00 [kthreadd]
system_u:system_r:kernel_t:s0     root           3       2  0 00:21 ?        00:00:00 [rcu_gp]
system_u:system_r:kernel_t:s0     root           4       2  0 00:21 ?        00:00:00 [rcu_par_gp]
system_u:system_r:kernel_t:s0     root           5       2  0 00:21 ?        00:00:00 [slub_flushwq]
system_u:system_r:kernel_t:s0     root           7       2  0 00:21 ?        00:00:00 [kworker/0:0H-events_highpri]
system_u:system_r:kernel_t:s0     root          10       2  0 00:21 ?        00:00:00 [mm_percpu_wq]
system_u:system_r:kernel_t:s0     root          11       2  0 00:21 ?        00:00:00 [rcu_tasks_rude_]
system_u:system_r:kernel_t:s0     root          12       2  0 00:21 ?        00:00:00 [rcu_tasks_trace]
system_u:system_r:kernel_t:s0     root          13       2  0 00:21 ?        00:00:07 [ksoftirqd/0]
system_u:system_r:kernel_t:s0     root          14       2  0 00:21 ?        00:00:08 [rcu_sched]
system_u:system_r:kernel_t:s0     root          15       2  0 00:21 ?        00:00:00 [migration/0]
system_u:system_r:kernel_t:s0     root          16       2  0 00:21 ?        00:00:00 [watchdog/0]
system_u:system_r:kernel_t:s0     root          17       2  0 00:21 ?        00:00:00 [cpuhp/0]
system_u:system_r:kernel_t:s0     root          18       2  0 00:21 ?        00:00:00 [cpuhp/1]
system_u:system_r:kernel_t:s0     root          19       2  0 00:21 ?        00:00:00 [watchdog/1]
```

Consulter le contexte de sécurité

- **Exemple 2 : liste des contextes pour le processus httpd**

```
[root@localhost ~]# ps -ef -Z | grep httpd
system_u:system_r:httpd_t:s0    root       15228       1  0 11:51 ?        00:00:01 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache     15235   15228  0 11:51 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache     15236   15228  0 11:51 ?        00:00:10 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache     15237   15228  0 11:51 ?        00:00:07 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0    apache     15238   15228  0 11:51 ?        00:00:07 /usr/sbin/httpd -DFOREGROUND
```

- **Exemple 3: contexte des fichiers du dossier /var/www/**

```
root@localhost ~]#  ls -alZ /var/www/
otal 4
rwxr-xr-x.  4 root root system_u:object_r:httpd_sys_content_t:s0       33 Nov  6 13:25 .
rwxr-xr-x. 22 root root system_u:object_r:var_t:s0                   4096 Nov  6 13:25 ..
rwxr-xr-x.  2 root root system_u:object_r:httpd_sys_script_exec_t:s0    6 Aug 30 16:03 cgi-bin
rwxr-xr-x.  3 root root system_u:object_r:httpd_sys_content_t:s0       41 Jan  1 13:01 html
root@localhost ~]#
```

Comprendre un contexte SELinux

- Un contexte SELinux est présenté de la manière suivante :

Utilisateur : rôle : type : niveau

- Pour le contexte du dossier /var/www/html :

system_u : object_r : httpd_sys_content_t : s0 html

⇒ **Utilisateur (system_u) :** C'est le compte utilisateur SELinux auquel le fichier ou le dossier appartient.

⇒ **Rôle (object_r) :** Le rôle SELinux attribué au fichier ou au dossier. Dans ce cas, il s'agit du rôle "object_r", qui est souvent utilisé pour les objets génériques.

⇒ **Type (httpd_sys_content_t)** : Le type SELinux attribué au fichier ou au dossier. Dans ce cas, le type est "httpd_sys_content_t", ce qui indique que le contenu est destiné au serveur web Apache (httpd).

⇒ **Niveau (s0) :** Le niveau de sécurité SELinux. Dans ce cas, le niveau est "s0", indiquant le niveau par défaut.