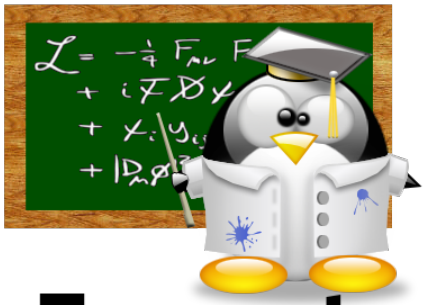


FORMATUX - Support de cours GNU/Linux

Sécurité CentOS 6

1.0, 30/06/2017

Chapitre 1. Préface

GNU/Linux est un **système d'exploitation** libre fonctionnant sur la base d'un **noyau Linux**, également appelé **kernel Linux**.

Linux est une implémentation libre du système **UNIX** et respecte les spécifications **POSIX**.

GNU/Linux est généralement distribué dans un ensemble cohérent de logiciels, assemblés autour du noyau Linux et prêt à être installé. Cet ensemble porte le nom de "**Distribution**".

- La plus ancienne des distributions est la distribution **Slackware**.
- Les plus connues et utilisées sont les distributions **Debian**, **RedHat** et **Arch**, et servent de base pour d'autres distributions comme **Ubuntu**, **CentOS**, **Fedora**, **Mageia** ou **Manjaro**.

Chaque distribution présente des particularités et peut être développée pour répondre à des besoins très précis :

- services d'infrastructure ;
- pare-feu ;
- serveur multimédia ;
- serveur de stockage ;
- etc.

La distribution présentée dans ces pages est la CentOS, qui est le pendant gratuit de la distribution RedHat. La distribution CentOS est particulièrement adaptée pour un usage sur des serveurs d'entreprises.

1.1. Crédits

Ce support de cours a été rédigé par les formateurs :

- Patrick Finet ;
- Antoine Le Morvan ;
- Xavier Sauvignon ;

1.2. Licence

Formatux propose des supports de cours Linux libres de droits à destination des formateurs ou des personnes désireuses d'apprendre à administrer un système Linux en autodidacte.

BY : Paternité. Vous devez citer le nom de l'auteur original.

SA : Partage des Conditions Initiales à l'Identique.

- Licence Creative Commons-BY-SA : https://creativecommons.org/licenses/by-sa/3.0/fr/
- Licence Art Libre : http://artlibre.org/

Les documents de Formatux et leurs sources sont librement téléchargeables sur framagit :

- https://framagit.org/alemorvan/formatux.fr-support/

Vous y trouverez la dernière version de ce document.

A partir des sources, vous pouvez générer votre support de formation personnalisé. Nous vous recommandons le logiciel AsciidocFX téléchargeable ici : http://asciidocfx.com/

1.3. Gestion des versions

Table 1. Historique des versions du document

Version	Date	Observations
1.0	Avril 2017	Version initiale.

Chapitre 2. Elévation des privilèges (su et sudo)

Le compte du super-utilisateur (root) possède de nombreuses contraintes car :

- il a tous les droits ;
- il peut causer des dommages irréparables ;
- il est la cible privilégiée des intrus.

L'administrateur qui travaille sur le serveur, se connecte avec son compte (qui dispose de droits restreints), puis au moment où il doit réaliser une tâche d'administration, endosse l'identité de **root** de façon temporaire, exécute ses actions, puis reprend son identité d'origine.

Il convient donc d'effectuer les opération suivantes :

- interdire le login root ;
- restreindre l'accès à la commande su ;
- privilégier la commande sudo ;
- robustesse du mot de passe :
  - il doit être complexe ;
  - non issu du dictionnaire ;
  - comporter des minuscules, majuscules, lettres, chiffres et caractères spéciaux ;
- changer régulièrement (ne pas être conservé identique indéfiniment) ;
- sécurisation (ne doit pas être inscrit à proximité des postes ou communiqué à des personnes qui n'ont pas à le connaître).

Ces principes sont les premières règles de sécurité.

2.1. Limiter le compte root

Le fichier **/etc/securetty** contient la liste des terminaux accessibles par le login **root**.

Pour interdire un terminal au login **root**, il faut :

- soit mettre la ligne en commentaire ;
- soit supprimer la ligne.

En fonction des distributions, la syntaxe du fichier /etc/securetty peut changer.

Pour une distribution **CentOS**, le fichier est constitué d'une ligne **tty** et d'une ligne **vc**, indispensable pour la connexion d'un utilisateur.

2.2. La commande su

su signifie **Substitute User** ou **Switch User**. Elle permet d'endosser l'identité d'un autre utilisateur sans se déconnecter. Cette commande utilisée sans login permet par défaut de prendre l'identité de **root**.

Syntaxe de la commande su

```
su [-] [-c commande] [login]
```

Exemple :

```
[root]# su - alain
[albert]$ su -c "passwd alain"
```

Option	Description
-	Charge l'environnement complet de l'utilisateur.
-c commande	Exécute la commande sous l'identité de l'utilisateur.

Pour la sécurité, les accès par la commande **su** sont répertoriés dans le fichier journal **/var/log/secure**.

Les utilisateurs non **root** devront taper le mot de passe de la nouvelle identité.

Il y a création de couches successives. Pour passer d'un utilisateur à un autre, il faut d'abord taper la commande **exit** pour reprendre son identité puis la commande **su** pour prendre une autre identité.

Il est conseillé de restreindre l'accès à la commande su à certaines personnes définies explicitement. Pour cela, il faut donner les droits à un groupe particulier pour accéder à la commande. Toutes les personnes appartenant à ce groupe obtiendront les droits réservés de celui-ci.

Le principe de restriction mis en place consiste à autoriser seulement un groupe d'utilisateurs particulier à exécuter la commande su.

2.3. La commande sudo

La commande **sudo** permet d'exécuter une ou plusieurs commandes avec les privilèges de root. Il n'est pas nécessaire de connaître son mot de passe.

Syntaxe de la commande sudo

```
sudo [-options] commande
```

Cet outil représente une évolution de la commande su.

Exemple :

```
[alain]$ sudo /sbin/route
Mot de passe : ********
Table de routage IP du noyau
Destination   Passerelle  Genmask    Indic Metric Ref Use Iface ·····
```

Option	Description
-E	Garde l'environnement du compte appelant.
-u	Désigne le compte qui exécutera la commande.

L'environnement est constitué des paramètres de l'utilisateur comme son répertoire de connexion, ses alias ...

Les accès possibles à la commande et les possibilités offertes aux utilisateurs sont configurables par l'administrateur qui leur affecte des droits explicites.

Vérification de l'existence du paquet :

```
[root] # rpm –qa "sudo*"
sudo-1.8.6p3-12.el6.i686
```

Avantages

Par rapport à l'utilisation des restrictions des groupes :

- Le contrôle d'accès aux commandes est centralisé dans le fichier **/etc/sudoers** ;
- Tous les accès et tentatives à partir de sudo sont journalisés dans le fichier **/var/log/secure**;
- L'utilisateur doit s'identifier à sudo en donnant son mot de passe.

Le fichier **/etc/sudoers** contient pour chacun des utilisateurs :

- son login ;
- sur quel ordinateur il est autorisé ;
- quelles commandes il peut exécuter.

2.4. Commande visudo

La commande visudo permet d'éditer le fichier /etc/sudoers (qui est en lecture seule) afin de configurer efficacement l'accès à la commande sudo pour les utilisateurs.

Syntaxe de la commande visudo

```
visudo [-c] [-f sudoers]
```

Exemple :

```
[root]# visudo
```

Option	Description
-c	Vérifie la syntaxe du fichier.
-f file	Désigne le fichier de configuration de sudo.

Le fichier /etc/sudoers

```
# This file must be edited with the
# 'visudo' command.
#
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root ALL=(ALL) ALL
%wheel ALL=(ALL) ALL
```

Le groupe wheel

Le mot **wheel** fait référence à un groupe système qui dispose de privilèges lui permettant l'exécution de commandes à accès restreint.

Le groupe **wheel** est présent par défaut sur les distributions RHEL/CentOS.

Ne plus utiliser root

1 - Autoriser les utilisateurs du groupe **wheel** à lancer toutes les commandes (via visudo) en supprimant le commentaire de la ligne suivante :

```
%wheel    ALL=(ALL)    ALL
```

2 - Ajouter le compte utilisateur **bob** dans le groupe **wheel** :

```
[root]# usermod -aG wheel bob
```

3 - L'utilisateur **bob** dispose maintenant des pleins pouvoirs :

```
[bob]$ sudo chown root:root /tmp/test
[sudo] password for bob:
```

Il n'existe plus de raisons valables d'utiliser le compte root.

Le mot de passe **root** peut être séquestré (Keepass !) et son accès via ssh verrouillé.

Restreindre le sudo

Il est possible de ne pas donner tous les droits à un utilisateur mais de limiter ses accès et les commandes qu'il peut lancer. Par exemple, n'offrir à un compte de supervision que le droit de relancer le serveur (reboot) et uniquement celui-là.

Les accès à l'utilisation de sudo sont enregistrés dans les logs ce qui offre une traçabilité des actions entreprises.

Les restrictions se gèrent dans les différentes sections du fichier /etc/sudoers par l'intermédiaire de la commande visudo.

Section Host aliases

Définir des groupes de machines ou réseaux.

```
Host_Alias HOSTS-GROUP = host1 [,host2 ...]
```

Exemple :

```
Host_Alias FILESERVERS = 192.168.1.1, SF1, SF2
```

Cette section est utilisée pour créer des groupes de ressources réseaux autorisés à utiliser sudo.

Section User aliases

Définir des alias pour les utilisateurs.

```
User_Alias USER-GROUP = alain [,philippe, ...]
```

Exemple :

```
User_Alias ADMINS = root, AdminSF, adminPrinters
```

Cette section est utilisée essentiellement pour réunir sous un alias unique plusieurs utilisateurs ayant les mêmes besoins de la commande sudo.

Section Command aliases

Définir des alias pour les commandes.

```
Cmnd_Alias CDE-GROUP = cde1 [,cde2, ...]
```

Exemple :

```
Cmnd_Alias SOFTWARE = /bin/rpm, /usr/bin/yum
Cmnd_Alias SERVICES = /sbin/service, /sbin/chkconfig
```

Cette section est utilisée pour réunir plusieurs commandes Linux dans un groupe de commandes sudo.

Il faut alors créer des groupes cohérents.

Section User Specification

Lier les utilisateurs aux commandes.

```
USER-GROUP HOSTS-GROUP = [(cpte-cible)] CDE-GROUP
```

Exemple :

```
root       ALL=(ALL)          ALL
AdminSF    FILESERVERS=       SOFTWARE
```

Cette section définit qui a le droit d'utiliser des commandes particulières à partir de postes particuliers.

Il est possible de préciser qui exécute la commande (compte cible).

Exemple 1 :

Grâce au fichier **/etc/sudoers** ci-dessous, les utilisateurs alain, patrick et philippe peuvent désormais exécuter les commandes **ping** et **route** et effectuer des transferts FTP comme s'ils étaient

root.

```
# Host alias specification
  Host_Alias STA = 192.168.1.1, ma.machine
# User alias specification
  User_Alias CPTUSER = alain, patrick, philippe
# Cmnd alias specification
  Cmnd_Alias NET = /bin/ping, /sbin/route, /usr/bin/ftp
# User privilege specification
  root ALL=(ALL) ALL
  %wheel ALL=(ALL) ALL
  CPTUSER STA=(root) NET
```

Exemple 2 :

```
# Host alias specification
  Host_Alias MACHINE = station1
# User alias specification
  User_Alias ADMIN = adminunix
  User_Alias UTILISAT = alain, philippe
# Cmnd alias specification
  Cmnd_Alias SHUTDOWN = /sbin/shutdown
  Cmnd_Alias NET = /usr/bin/ftp
# User privilege specification
  root ALL=(ALL) ALL
  %wheel ALL=(ALL) ALL
  ADMIN MACHINE=NOPASSWD:ALL
  UTILISAT MACHINE=(root) SHUTDOWN, NET
```

Explications de l'exemple 2 :

- Création d'un alias pour la station :

```
Host_Alias MACHINE=station1
```

- Création de deux alias pour deux types d'utilisateurs (adminunix étant un équivalent « root »).

```
User_Alias ADMIN = adminunix
User_Alias UTILISAT = alain, philippe
```

- Création de deux alias de commandes qui regroupent les commandes exécutables.

```
Cmnd_Alias SHUTDOWN = /sbin/shutdown
Cmnd_Alias NET = /usr/bin/ftp
```

- L'utilisateur « root » peut exécuter toutes les commandes sur toutes les machines

```
root ALL=(ALL) ALL
```

Les utilisateurs qui font partie de l'alias ADMIN peuvent exécuter toutes les commandes, sur toutes les machines faisant partie de l'alias MACHINE et ce sans entrer de mot de passe (NOPASSWD:).

```
ADMIN MACHINE=NOPASSWD:ALL
```

Les utilisateurs qui font partie de l'alias UTILISAT peuvent exécuter la commande /sbin/shutdown sur toutes les machines faisant partie de l'alias MACHINE. Ils doivent entrer leur mot de passe.

```
UTILISAT MACHINE = SHUTDOWN, NET
```

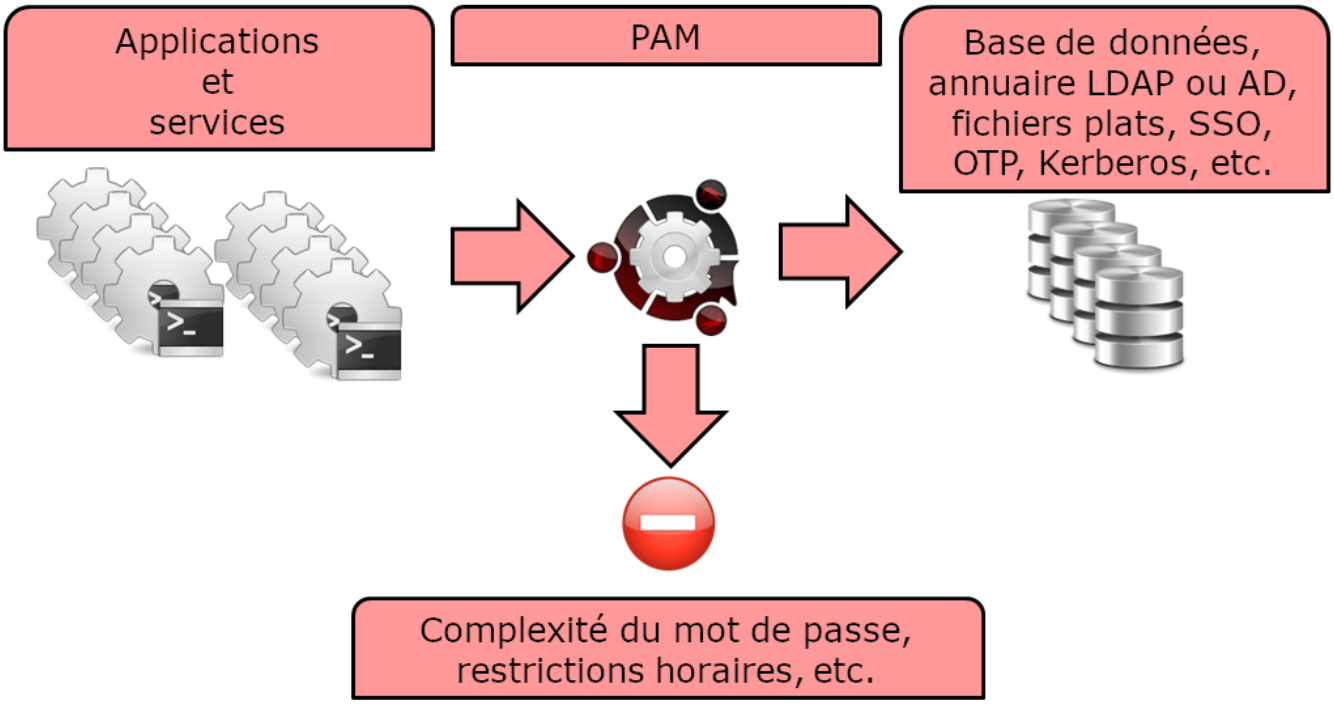
Chapitre 3. Les modules d'authentification PAM

PAM (**Pluggable Authentication Modules**) est le système sous GNU/Linux qui permet à de nombreuses applications ou services d'**authentifier** les utilisateurs de manière **centralisée**.

> PAM est un mécanisme permettant d'intégrer différents schémas d'authentification de bas niveau dans une API de haut niveau, permettant de ce fait de rendre indépendants du schéma les logiciels réclamant une authentification.
>
> — Définition de PAM par Wikipedia

3.1. Généralités

L'authentification est la phase durant laquelle il est vérifié que vous êtes bien la personne que vous prétendez être. Il existe d'ailleurs d'autres formes d'authentification que l'utilisation des mots de passe.



La mise en place d'une nouvelle méthode d'authentification ne doit pas nécessiter de modifications dans la configuration ou dans le code source d'un programme ou d'un service.

C'est pourquoi les applications s'appuient sur PAM, qui va leur fournir les primitives nécessaires à l'authentification de leurs utilisateurs.

L'ensemble des applications d'un système peuvent ainsi mettre en œuvre en toute transparence des

fonctionnalités complexes comme le **SSO** (Single Signe On), l'**OTP** (One Time Password) ou **Kerberos** de manière totalement transparente.

L'administrateur d'un système peut choisir exactement sa politique d'authentification pour une seule application (par exemple pour durcir le service SSH) indépendamment de celle-ci.

À chaque application ou service prenant en charge PAM correspondra un fichier de configuration dans le répertoire «/etc/pam.d». Par exemple, le processus «login» attribue le nom «/etc/pam.d/login» à son fichier de configuration.

Une mauvaise configuration de PAM peut compromettre toute la sécurité de votre système.

PAM est un système d'authentification (gestion des mots de passe). Si PAM est vulnérable alors l'ensemble du système est vulnérable.

Syntaxe d'une directive

Une directive permet de paramétrer une application pour PAM.

Syntaxe d'une directive

```
mécanisme [contrôle] chemin-module [argument]
```

Par exemple :

Le fichier /etc/pam.d/sudo

```
#%PAM-1.0
auth            include     system-auth
account         include     system-auth
password        include     system-auth
session         optional    pam_keyinit.so revoke
session         required    pam_limits.so
```

Une **directive** (une ligne complète) est composée d'un **mécanisme** (auth, account, password ou session), d'un **contrôle de réussite** (include, optional, required, ...), du chemin d'accès au module et éventuellement d'arguments (comme revoke par exemple).

L'ordre des modules est très important !

Chaque fichier de configuration PAM comprend un ensemble de directives. Les directives des interfaces de modules peuvent être empilées ou placées les unes sur les autres.

De fait, l'ordre dans lequel les modules sont répertoriés est très important au niveau du processus d'authentification.

3.2. Les mécanismes

Le mécanisme auth - Authentification

Concerne l'authentification du demandeur et établit les droits du compte :

- Authentifie généralement avec un mot de passe en le comparant à une valeur stockée en base de données ou en s'appuyant sur un serveur d'authentification,
- Établit les paramètres du comptes : uid, gid, groupes et limites de ressources.

Le mécanisme account - Gestion de compte

Vérifier que le compte demandé est disponible :

- Concerne la disponibilité du compte pour des raisons autres que l'authentification (par exemple pour des restrictions d'horaire).

Le mécanisme session - Gestion de session

Concerne la mise en place et la terminaison de la session :

- Accomplir les tâches associées à la mise en place d'une session (par exemple enregistrement dans les logs),
- Accomplir les tâches associées à la terminaison d'une session.

Le mécanisme password - Gestion des mots de passe

Utilisé pour modifier le jeton d'authentification associé à un compte (expiration ou changement) :

- Modifie le jeton d'authentification et vérifie éventuellement qu'il est assez robuste ou qu'il n'a pas déjà été utilisé.

3.3. Les indicateurs de contrôle

Les **mécanismes PAM (auth**, **account**, **session** et **password)** indiquent la réussite ou l'échec. Les **indicateurs de contrôle** (**required**, **requisite**, **sufficient**, **optional**) indiquent à PAM comment traiter ce résultat.

L'indicateur de contrôle required

La réussite de tous les modules **required** est nécessaire.

- Si le module réussit :

Le reste de la chaîne est exécuté. La requête est autorisée sauf si d'autres modules échouent.

- Si le module échoue :

Le reste de la chaîne est exécuté. Au final la requête est rejetée.

Le module doit être vérifié avec succès pour que l'authentification puisse se poursuivre. Si la vérification d'un module portant l'indication required échoue, l'utilisateur n'en est pas averti tant que tous les modules associés à cette interface n'ont pas été vérifiés.

L'indicateur de contrôle requisite

La réussite de tous les modules **requisite** est nécessaire.

- Si le module réussit :

Le reste de la chaîne est exécuté. La requête est autorisée sauf si d'autres modules échouent.

- Si le module échoue :

La requête est immédiatement rejetée.

Le module doit être vérifié avec succès pour que l'authentification puisse se poursuivre. Cependant, si la vérification d'un module requisite échoue, l'utilisateur en est averti immédiatement par le biais d'un message lui indiquant l'échec du premier module required ou requisite.

L'indicateur de contrôle sufficient

La réussite d'un seul module **sufficient** est suffisant.

- Si le module réussit :

La requête est immédiatement autorisée si aucun des précédents modules n'a échoué.

- Si le module échoue :

Le module est ignoré. Le reste de la chaîne est exécutée.

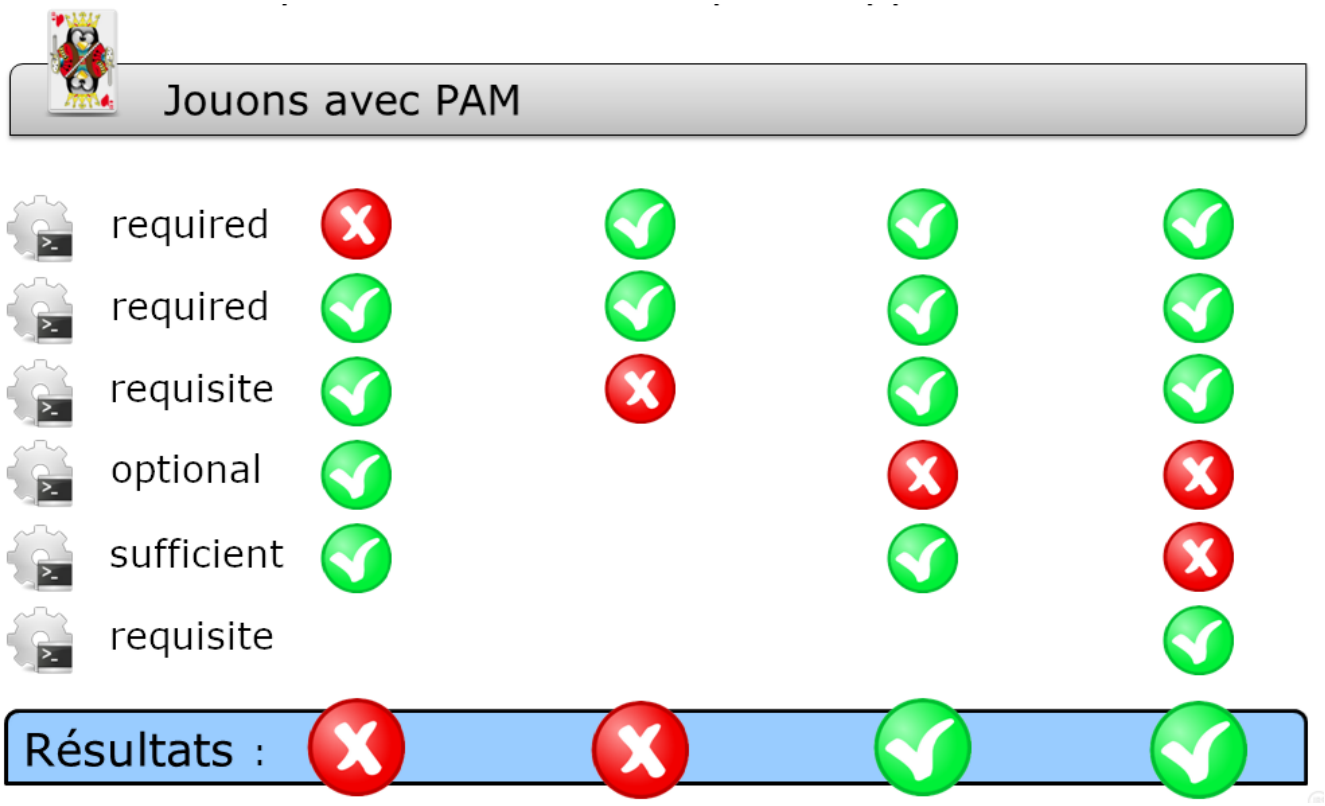
En cas d'échec, les vérifications de modules sont ignorées. Toutefois, si la vérification d'un module portant l'indication sufficient est réussie et qu'aucun module précédent portant l'indicateur required ou requisite n'a échoué, aucun autre module de ce type n'est nécessaire et l'utilisateur sera authentifié auprès du service.

L'indicateur de contrôle optional

Le module est exécuté mais le résultat de la requête est ignoré.

Si tous les modules de la chaînes étaient marqués optional, toutes les requêtes seraient toujours acceptées.

En conclusion



3.4. Les modules de PAM

Il existe de nombreux **modules** pour PAM. Voici les plus fréquents :

- pam_unix
- pam_ldap
- pam_wheel
- pam_cracklib
- pam_console
- pam_tally
- pam_securetty
- pam_nologin
- pam_limits
- pam_time
- pam_access

Le module pam_unix

Le module **pam_unix** permet de gérer la politique globale d'authentification.

Fichier /etc/pam.d/system-auth

```
password sufficient pam_unix.so sha512 nullok
```

Des arguments sont possibles pour ce module :

- **nullok** : dans le mécanisme auth autorise un mot de passe de connexion vide.
- **sha512** : dans le mécanisme password, définit l'algorithme de cryptage.
- **debug** : pour transmettre les informations à "syslog".
- **remember=n** : pour se souvenir des n derniers mots de passe utilisés (fonctionne conjointement avec le fichier "/etc/security/opasswd", qui est à créer par l'administrateur).

Le module pam_cracklib

Le module **pam_cracklib** permet de tester les mots de passe.

Fichier /etc/pam.d/password-auth

```
password sufficient pam_cracklib.so retry=2
```

Ce module utilise la bibliothèque **cracklib** pour vérifier la solidité d'un nouveau mot de passe. Il peut également vérifier que le nouveau mot de passe n'est pas construit à partir de l'ancien. Il ne concerne que le mécanisme password.

Par défaut ce module vérifie les aspects suivants et rejette si tel est le cas :

- le nouveau mot de passe est-il issu du dictionnaire ?
- le nouveau mot de passe est-il un palindrome de l'ancien (ex : azerty <> ytreza) ?
- seule la casse de(s) caractère(s) varie (ex : azerty <>AzErTy) ?

Des arguments possibles pour ce module :

- **retry=n** : impose n demandes (1 par défaut) du nouveau mot de passe.
- **difok=n** : impose au minimum n caractères (10 par défaut) différents de l'ancien mot de passe. De plus si la moitié des caractères du nouveau diffèrent de l'ancien, le nouveau mot de passe est validé.
- **minlen=n** : impose un mot de passe de n+1 caractères minimum non pris en compte en dessous de 6 caractères (module compilé comme tel !).

Autres arguments possibles :

- **dcredit=-n** : impose un mot de passe contenant au moins n chiffres,
- **ucredit=-n** : impose un mot de passe contenant au moins n majuscules,
- **credit=-n** : impose un mot de passe contenant au moins n minuscules,

- **ocredit=-n** : impose un mot de passe contenant au moins n caractères spéciaux.

Le module pam_tally

Le module **pam_tally** permet de verrouiller un compte en fonction d'un nombre de tentatives infructueuses de connexion.

Fichier /etc/pam.d/system-auth

```
auth required /lib/security/pam_tally.so onerr=fail no_magic_root
account required /lib/security/pam_tally.so deny=3 reset no_magic_root
```

- Le mécanisme **account** incrémente le compteur.
- Le mécanisme **auth** accepte ou refuse l'authentification et réinitialise le compteur.

Quelques arguments du module pam_tally sont intéressants à utiliser :

- **onerr=fail** : incrémentation du compteur,
- **deny=n** : une fois le nombre n d'essais infructueux dépassé, le compte est verrouillé,
- **no_magic_root** : inclus ou non les démons gérés par root (éviter le verrouillage de root),
- **reset** : remet le compteur à 0 si l'authentification est validée,
- **lock_time=nsec** : le compte est verrouillé pour n secondes.

Ce module fonctionne conjointement avec le fichier par défaut des essais infructueux /var/log/faillog (qui peut être remplacé par un autre fichier avec l'argument file=xxxx)et la commande associée faillog.

Syntaxe de la commande faillog

```
faillog[-m n] |-u login][-r]
```

Options :

- m : pour définir, dans l'affichage de la commande, le nombre maximum d'essais infructueux,
- u : pour spécifier un utilisateur,
- r : déverrouiller un utilisateur.

Le module pam_time

Le module **pam_time** permet de limiter les horaires d'accès à des services gérés par PAM.

Fichier /etc/pam.d/system-auth

```
account required /lib/security/pam_time.so
```

La configuration se fait via le fichier /etc/security/time.conf.

Fichier /etc/security/time.conf

```
login ; * ; users ;MoTuWeThFr0800-2000
http  ; * ; users ;Al0000-2400
```

La syntaxe d'une directive est la suivante :

```
services ; ttys ; users ; times
```

Dans les définitions suivantes, la liste logique utilise :

- **&** : et logique,
- | : ou logique,
- **!** : négation = « tous sauf »,
- ***** : caractère « joker ».

Les colonnes correspondent à :

- **services** : liste logique de services gérés par PAM qui sont concernés,
- **ttys** : liste logique de périphériques concernés,
- **users** : liste logique d'utilisateurs gérés par la règle,
- **times** : liste logique de détermination de l'horaire (jour/plage) autorisé.

Comment gérer les créneaux horaires :

- **les jours** : Mo Tu We Th Fr Sa Su Wk (du L au V) Wd (S et D) Al (du L au D),
- **la plage** : HHMM- HHMM,
- **une répétition annule l'effet** : WkMo = toutes les jours de la semaine (L-V) moins le lundi (répétition).

Exemples :

- Bob, peut se connecter via un terminal tous les jours entre 07h00 et 09h00, sauf le mercredi :

```
login ; tty* ; bob ; alth0700-0900
```

- Pas d'ouvertures de sessions, terminal ou distantes, sauf root, tous les jours de la semaine entre 17h30 et 7h45 le lendemain :

```
login ; tty* | pts/* ; !root ; !wk1730-0745
```

Le module pam_nologin

Le module **pam_nologin** permet de désactiver tous les comptes sauf root :

Fichier /etc/pam.d/login :

```
auth required   pam_nologin.so
```

Si le fichier /etc/nologin existe alors seul root pourra se connecter.

Le module pam_wheel

Le module **pam_wheel** permet de limiter l'accès à la commande su aux membres du groupes wheel.

Fichier /etc/pam.d/su

```
auth    required    pam_wheel.so
```

L'argument **group=mon_group** limite l'usage de la commande su aux membres du groupe mon_groupe

Si le groupe mon_groupe est vide, alors la commande su n'est plus disponible sur le système, ce qui force l'utilisation de la commande sudo.

Le module pam_mount

Le module **pam_mount** permet de monter un volume pour une session utilisateur.

Fichier /etc/pam.d/system-auth

```
auth        optional    pam_mount.so
password    optional    pam_mount.so
session optional    pam_mount.so
```

Les points de montage sont configurés dans le fichier /etc/security/pam_mount.conf :

Fichier /etc/security/pam_mount.conf

```
<volume fstype="nfs" server="srv" path="/home/%(USER)" mountpoint="~" />
<volume user="bob" fstype="smbfs" server="filesrv" path="public" mountpoint="/public"
/>
```

Chapitre 4. Sécurisation SELinux

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.

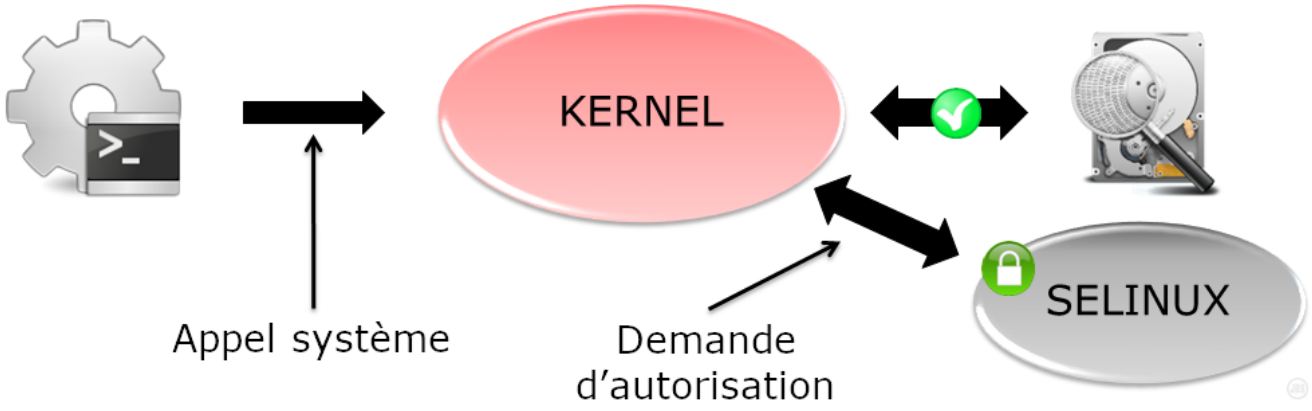
4.1. Généralités

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.

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.

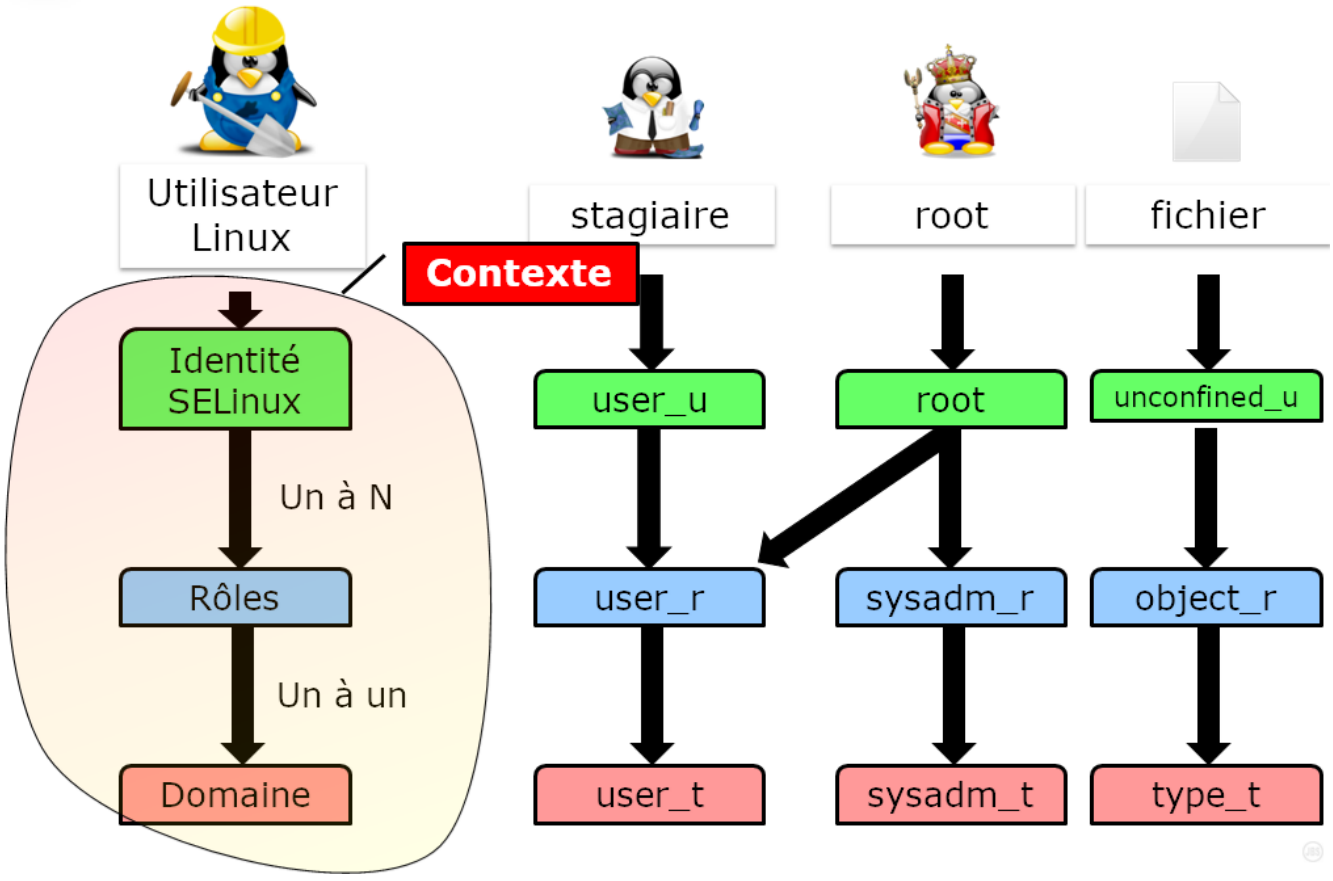


Figure 1. Le contexte SELinux

```
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.

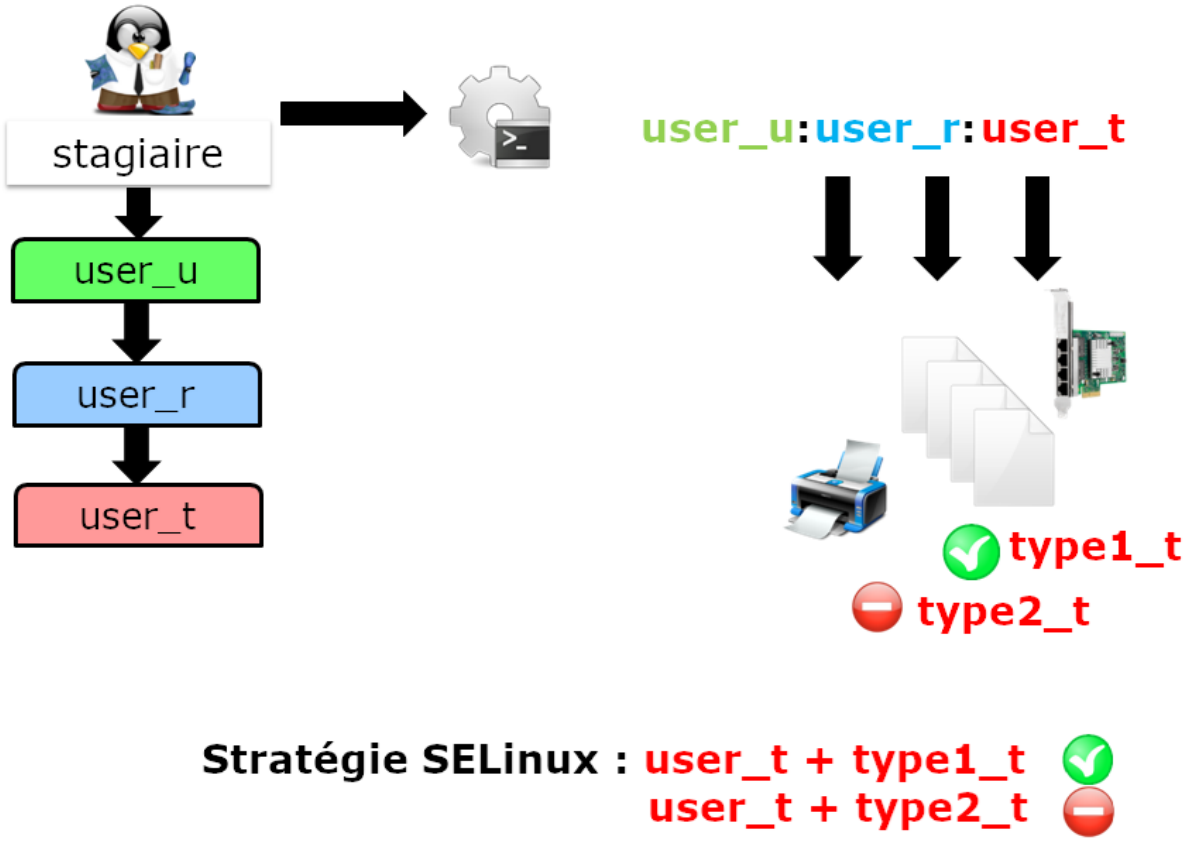


Figure 2. Le contexte SELinux d'un processus standard

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.

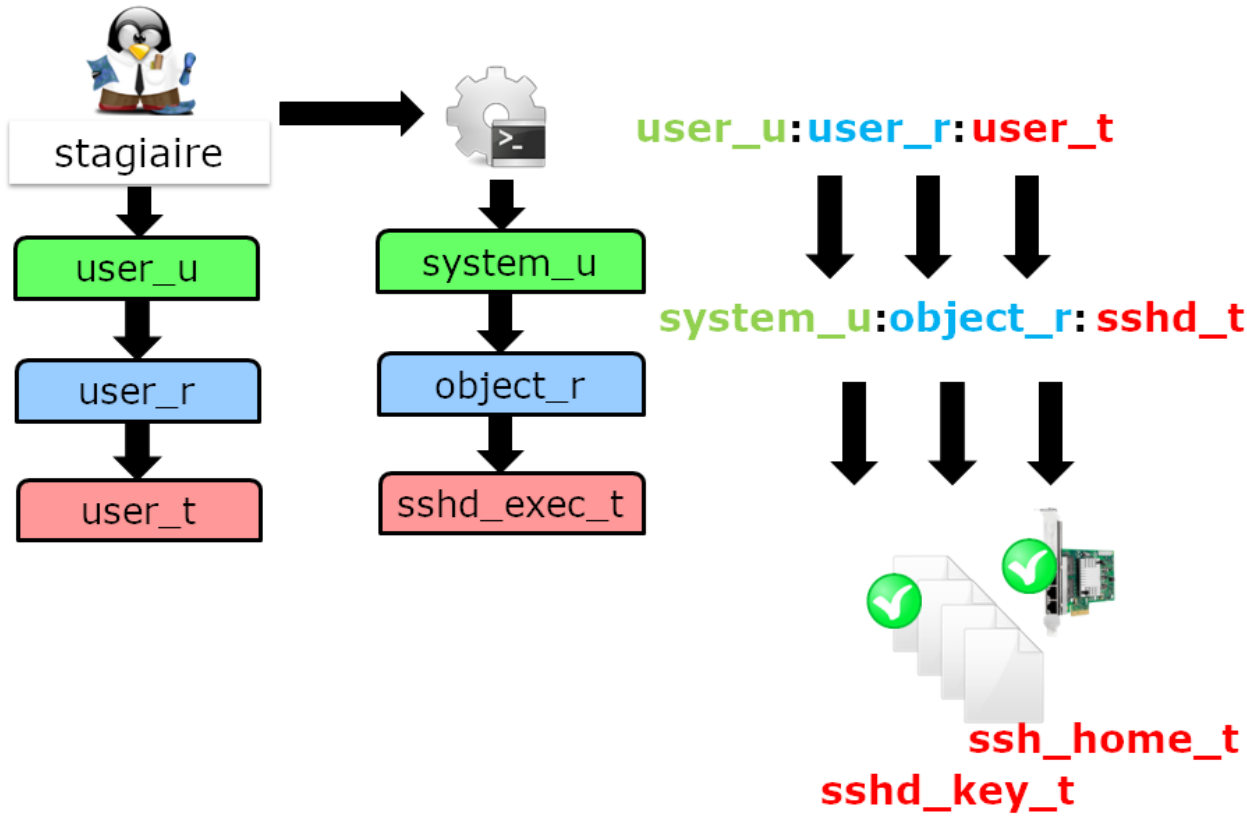


Figure 3. Le contexte SELinux d'un processus important - exemple de sshd

4.2. 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 2. 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
```

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 3. 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
```

4.3. 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
```

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
```

4.4. 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

4.5. 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 4. 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 5. 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 6. 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)

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 7. 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
```

Chapitre 5. IPtables, le pare-feu Linux

Gérer le trafic réseau est certainement la partie la plus difficile du métier d'administrateur système. Il faut impérativement configurer les pare-feu sur l'ensemble des éléments actifs du réseau, serveurs inclus, en prenant en compte les besoins des utilisateurs et des systèmes à la fois pour le trafic entrant et sortant, sans laisser des systèmes vulnérables à des attaques.

C'est le rôle du pare-feu **IPTables**, entièrement administrable en ligne de commandes.

IPtables utilise un jeu de tables qui contiennent des chaînes comprenant les règles du firewall.

Il existe 3 types de tables :

- La table **FILTER**, qui est la table par défaut. Elle est composée de 3 chaînes :
  - La chaîne **INPUT** : les paquets sont destinés à une socket locale.
  - La chaîne **FORWARD** : les paquets sont routés par le système.
  - La chaîne **OUTPUT** : les paquets sont générés en local.
- La table **NAT**, qui est consultée lorsque un paquet tente de créer une nouvelle connexion. Elle est composée de 3 chaînes :
  - **PREROUTING** : utilisée pour modifier des paquets dès leur réception par le système.
  - **OUTPUT** : utilisée pour modifier des paquets générés en local.
  - **POSTROUTING** : utilisée pour modifier des paquets au moment de leur sortie du système.
- La table **MANGLE** : cette table est utilisée pour modifier des paquets. Il y a 5 chaînes :
  - **PREROUTING** : pour modifier des connexions entrantes.
  - **OUTPUT** : pour modifier des paquets générés en local.
  - **INPUT** : pour modifier les paquets entrants.
  - **POSTROUTING** : pour modifier les paquets avant leur départ du système.
  - **FORWARD** : pour modifier les paquets qui sont routés par le système.

5.1. Gestion du pare-feu

Démarrer/arrêter ou redémarrer le parefeu

En fonction de la distribution, si elle utilise les scripts de démarrage SystemD (RHEL 7 et +) ou SysVinit, il faudra utiliser les lignes de commandes suivantes :

Distribution basée sur SystemD

```
[admin]$ sudo systemctl start iptables
[admin]$ sudo systemctl stop iptables
[admin]$ sudo systemctl restart iptables
```

Distribution basée sur SysVinit

```
[admin]$ sudo service iptables start
[admin]$ sudo service iptables stop
[admin]$ sudo service iptables restart
```

Afficher les règles

Pour afficher les règles IPTables :

```
[admin]$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 927K packets, 219M bytes)
pkts bytes target prot opt in out source destination
...

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
...

Chain OUTPUT (policy ACCEPT 127 packets, 18M bytes)
pkts bytes target prot opt in out source destination
...
```

Il est possible de spécifier, avec l'option -t, quelle table afficher :

```
[admin]$ sudo iptables -t input -L -n -v
Chain INPUT (policy ACCEPT 927K packets, 219M bytes)
pkts bytes target prot opt in out source destination
...
```

5.2. Quelques exemples

Voici quelques règles qui mettent en oeuvre les fonctionnalités d'IPTables :

Bloquer des adresses IP spécifiques

Pour bloquer une adresse qui aurait un comportement abusif :

```
iptables -A INPUT -s xxx.xxx.xxx.xxx -j DROP
```

Changer xxx.xxx.xxx.xxx par l'adresse IP à bloquer.

pour la débloquer :

```
iptables -D INPUT -s xxx.xxx.xxx.xxx -j DROP
```

L'option **-D** ou **--delete** supprime la règle dans la chaîne spécifiée.

Bloquer/accepter des ports spécifiques

Pour bloquer un port spécifique en sortie :

```
iptables -A OUTPUT -p tcp --dport xxx -j DROP
```

alors que pour autoriser une connexion entrante :

```
iptables -A INPUT -p tcp --dport xxx -j ACCEPT
```

Pour bloquer le protocole udp, remplacer simplement tcp par udp.

Plusieurs ports peuvent être spécifiés en même temps :

```
iptables -A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
```

Autoriser un réseau

Un réseau complet peut être spécifié, par exemple pour autoriser un accès SSH :

```
iptables -A OUTPUT -p tcp -d 10.10.10.0/24 --dport 22 -j ACCEPT
```

Limiter le nombre de connexion d'une adresse

Pour limiter le nombre de paquets par adresse (protection contre le flooding) :

```
iptables -A INPUT -p tcp --dport 80 -m limit --limit 50/minute --limit-burst 100 -j
ACCEPT
```

Pour limiter le nombre de connexions actives (ici 3 sur le port ssh) :

```
iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 3 -j REJECT
```

Bloquer le protocole ICMP (ping)

Bloquer le protocole ICMP peut être considéré comme une mesure de sécurité.

```
iptables -A INPUT -p icmp -i eth0 -j DROP
```

Accès à la loopback

L'accès à l'interface de loopback doit toujours être possible. Les lignes suivantes doivent impérativement être présentes :

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

Logger les paquets refusés

Iptables peut envoyer à syslog (/var/log/messages) les paquets qu'il refuse :

```
iptables -A INPUT -i eth0 -J LOG --log-prefix "REFUS IPTABLES : "
```

Cette règle est à mettre en toute dernière, juste avant la suppression des paquets.

Gérer les connexions établies

Il est nécessaire d'autoriser les paquets provenant de connexions déjà établies ou en relation avec d'autres connexions.

```
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

Supprimer les paquets invalides

Certains paquets sont marqués invalides à l'arrivée (duplication, dé-séquencement, etc.). Il est possible de les journaliser pour pouvoir éventuellement mener des actions correctrices, puis de les supprimer :

```
iptables -A INPUT -m conntrack --ctstate INVALID -J LOG --log-prefix "REFUS IPTABLES :
iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
```

Bloquer le trafic SMTP sortant

Pour empêcher toute sortie de mails depuis un des ports correspondant aux ports SMTP :

```
iptables -A OUTPUT -p tcp --dports 25,465,587 -j REJECT
```

5.3. Conclusion

Ces exemples permettent de faire le tour des fonctionnalités offertes par le pare-feu iptables. Quelques règles permettent d'offrir un niveau de sécurité plus important.

La mise en place du pare-feu n'est donc pas un élément à négliger.

Chapitre 6. Fail2ban

Fail2ban permet le blocage des adresses IP tentant une intrusion sur votre serveur. Pour cela, il s'appuie sur le pare-feu Netfilter ou sur TCP Wrapper.

Il détecte les tentatives d'intrusion en parcourant les journaux du système.

6.1. Installation

Fail2ban est disponible dans le dépôt EPEL.p

```
yum install fail2ban
```

Le fichier **/etc/fail2ban/jail.conf** est fourni par le paquet **fail2ban**.

Il ne faut le modifier directement, sous peine de voir ses changements perdus lors de la prochaine mise à jour du paquet. Au lieu de cela, il faut créer un fichier **/etc/fail2ban/jail.local**, et y placer sa configuration personnalisée.

```
cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

6.2. Configuration

Options générales (section DEFAULT) :

- **bantime** : le temps en secondes durant lequel l'adresse IP ayant tentée une intrusion sera bannie ;
- **findtime** : la plage de temps en secondes utilisée pour analyser les logs. Plus cette plage est grande, plus l'analyse est longue ;
- **maxretry** : le nombre d'échecs de connexion tolérés avant le bannissement.

Il est primordiale d'ajuster au mieux ces trois valeurs. Un attaquant, avec la configuration par défaut, peut faire 5 tentatives toutes les 10 minutes, sans être banni. Cette valeur peut paraître ridiculement petite, mais elle représente par jour 5x6x24=720 tentatives, et 262 800 par an. Elle est encore à multiplier par le nombre de postes participant à l'attaque.

Même avec un système comme fail2ban, un poste n'est pas à l'abri des attaques. Fail2ban n'empêchera pas un attaquant de prendre possession de votre serveur, mais le retardera. Il est important de respecter les règles de bases de la sécurité informatique : changement fréquent de mot de passe, complexité, etc.

```
[root]# vim /etc/fail2ban/jail.local
[DEFAULT]
bantime  = 3600
findtime = 600
maxretry = 5

[ssh-iptables]
enabled  = true
filter   = sshd
action   = iptables[name=SSH, port=ssh, protocol=tcp] sendmail-whois[name=SSH,
dest=root, sender=fail2ban@formatux.fr]
logpath  = /var/log/secure
maxretry = 5
```

- section ssh-iptables :
  - enabled : active la règle
  - filter : fichier de log à analyser. Un chemin complet ou un raccourci (comme c'est le cas ici)
  - action : que dois faire fail2ban en cas d'échec de connexion ?
    - iptables : activer une règle dans le pare-feu,
    - sendmail-whois : envoyer un mail de rapport.

6.3. Lancement du service

- Fail2ban s'appuyant sur le firewall netfilter pour bannir les adresses IP tentant une intrusion, il faut s'assurer que celui-ci soit démarré :

```
service iptables status
service ip6tables status
```

- Si le pare-feu n'est pas actif sur le système :

```
chkconfig iptables on
chkconfig ip6tables on
service iptables start
service ip6tables start
```

- Démarrer le service fail2ban :

```
chkconfig fail2ban on
service fail2ban start
```

6.4. Vérification du service

La commande **fail2ban-client status** permet d'obtenir des informations sur les services surveillés ainsi que le nombre de règles iptables mises en place :

```
[root]# fail2ban-client status
Status
|- Number of jail:  2
`- Jail list:       mysqld-iptables, ssh-iptables
```

IPtables doit également renvoyer des informations concernant l'utilisation d'une chaîne fail2ban-SSH :

```
[root]# iptables --list

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
fail2ban-SSH  tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED
...

Chain fail2ban-SSH (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
```

6.5. Interface graphique

Fail2Web est une interface graphique web pour Fail2Ban, qui communique via **Fail2Rest** (service REST).

Attention à la configuration du serveur Fail2Rest, qui par défaut ne propose pas d'authentification, il faudra le faire via l'authentification apache.

Chapitre 7. Sécuriser le serveur SSH

Le serveur openSSH permet l'administration d'un serveur à distance.

7.1. Configuration

La configuration du serveur SSH se fait dans le fichier **/etc/ssh/sshd_config**.

Á chaque modification, il faut relancer le service :

```
service sshd restart
```

7.2. Changer le port d'écoute et la version du protocole

Il est préférable de changer le port par défaut (22) par un port connu de vous seul et de n'utiliser que la dernière version du protocole :

```
Port XXXX
Protocol 2
```

7.3. Utilisation de clefs privées/publiques

Dans la mesure du possible, utilisez un couple de clef privée/publique pour l'accès au serveur, et désactivez les autres possibilités de connexion (authentification par utilisateur + mot de passe) :

```
PasswordAuthentication no
RSAAuthentication yes
PubkeyAuthentication yes
```

7.4. Limiter les accès

Il est possible de limiter les accès directement dans la configuration du service avec la directive AllowUsers :

```
AllowUsers antoine
```

Il est également possible de limiter les accès par adresse IP via TCP Wrapper. Par exemple, refusez tous les accès dans le fichier /etc/hosts.deny :

```
sshd: ALL
```

et n'acceptez dans le fichier /etc/hosts.allow que les connexions depuis des adresses IP validées :

```
sshd: 192.168.1. 221.10.140.10
```

Voir la configuration de TCP Wrapper pour plus d'informations.

7.5. Interdire l'accès à root !!!

La mesure essentielle à prendre est d'interdire l'accès direct à root au serveur ssh :

```
PermitRootLogin no
```

et d'utiliser les possibilités offertes par le fichier sudoers pour permettre aux utilisateurs administrateurs de lancer des commandes d'administration.

7.6. Sécurité par le parefeu

Il est également important de limiter les accès aux services grâce au pare-feu et de bannir les adresses IP tentant des attaques par dictionnaire.

Chapitre 8. Autorité de certification TLS avec easy-rsa

SSL (Secure Socket Layer) est le protocole historique développé par la société Netscape pour sécuriser les échanges entre les clients et les serveurs Web. SSL a été **standardisé par l'IETF** sous le nom de **TLS** (Transport Layer Security). TLS n'est rien d'autre que SSLv3 avec quelques corrections et améliorations.

Une autorité de certification (**CA, Certificate Authority**) agit comme une entité disposant d'une biclé (couple de clé privée/publique) de confiance représentant un "certificat racine". Ce certificat va seulement être employé pour signer d'autres certificats après avoir vérifié l'identité qui y est inscrite. Tout client voulant utiliser un service s'appuyant sur TLS devra disposer du certificat de sa CA pour valider l'authenticité du certificat TLS du serveur.

8.1. Installer easy-rsa :

Easy-rsa est disponible dans le dépôt EPEL.

```
[root]# yum install easy-rsa
```

8.2. Configuration

- Se déplacer dans le répertoire easy-rsa :

```
[root]# cd /usr/share/easy-rsa/2.0/
```

- Configurer les réponses par défaut :

```
[root]# vim vars
...
export KEY_COUNTRY="FR"
export KEY_PROVINCE="fr"
export KEY_CITY="Rennes"
export KEY_ORG="Formatux"
export KEY_EMAIL="admin@formatux.fr"
export KEY_OU="formatux.fr"

# X509 Subject Field
export KEY_NAME="Formatux"
```

8.3. Créer une autorité de certification

- Se déplacer dans le répertoire easy-rsa :

```
[root]# cd /usr/share/easy-rsa/2.0/
[root]# source ./vars
[root]# ./clean-all
[root]# ./build-ca
Generating a 2048 bit RSA private key
..........................+++
.....................................................................+++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [FR]:
State or Province Name (full name) [fr]:
Locality Name (eg, city) [Rennes]:
Organization Name (eg, company) [Formatux]:
Organizational Unit Name (eg, section) [formatux.fr]:
Common Name (eg, your name or your server's hostname) [Formatux CA]:
Name [Formatux]:
Email Address [admin@formatux.fr]:

[root]# ./build-dh
```

8.4. Créer une biclé serveur

```
[root]# cd /usr/share/easy-rsa/2.0/
[root]# source ./vars
[root]# ./build-key-server servername
```

8.5. Installer le certificat de l'autorité de certification

- Installer le package ca-certificates

```
[root]# yum install ca-certificates
```

- Activer la configuration dynamique de l'autorité de certification :

```
[root]# update-ca-trust enable
```

- Ajouter le certificat de l'autorité :

```
[root]# cp foo.crt /etc/pki/ca-trust/source/anchors/
[root]# update-ca-trust extract
```

Chapitre 9. Glossaire

BASH
: Bourne Again SHell

BIOS
: Basic Input Output System

CIDR
: Classless Inter-Domain Routing

Daemon
: Disk And Execution MONitor

DHCP
: Dynamic Host Control Protocol

DNS
: Domain Name Service

FIFO
: First In First Out

FQDN
: Fully Qualified Domain Name

GNU
: Gnu is Not Unix

HTTP
: HyperText Transfer Protocol

ICP
: Internet Cache Protocol

IFS
: Internal Field Separator

LAN
: Local Area Network

LDIF
: LDAP Data Interchange Format

NTP

Network Time Protocol

nsswitch

Name Service Switch

OTP

One Time Password

POSIX

Portable Operating System Interface

POST

Power On Self Test

RDBMS

Relational DataBase Managed System

SGBD-R

Systèmes de Gestion de Base de Données Relationnelles

SHELL

En français "coquille". Á traduire par "interface système".

SMB

Server Message Block

SMTP

Simple Mail Tranfer Protocol

SSH

Secure SHell

SSL

Secure Socket Layer

TLS

Transport Layer Security, un protocole de cryptographie pour sécuriser les communications IP.

TTL

Time To Live

TTY

teletypewriter, qui se traduit téléscripteur. C'est la console physique.

UEFI

Unified Extensible Firmware Interface

Chapitre 10. Index

A

audit2allow, 29
audit2why, 28

C

CA, 41
chcon, 28

D

DAC, 20

E

Enforcing, 25

F

Fail2ban, 36
faillog, 17

G

getenforce, 25

K

Kerberos, 12

M

MAC, 20
matchpathcon, 27

O

OTP, 12

P

PAM, 11
Permissive, 25
pam_cracklib, 16
pam_mount, 19
pam_nologin, 19
pam_tally, 17
pam_time, 17
pam_unix, 15
pam_wheel, 19

R

restorecon, 28

S

SELinux, 20
SSL, 41
SSO, 12
Strict, 27
semanage, 23
sestatus, 26
setenforce, 26
setsebool, 24
su, 3, 4
sudo, 3, 4

T

TLS, 41
Targeted, 27

V

visudo, 6

W

wheel, 6