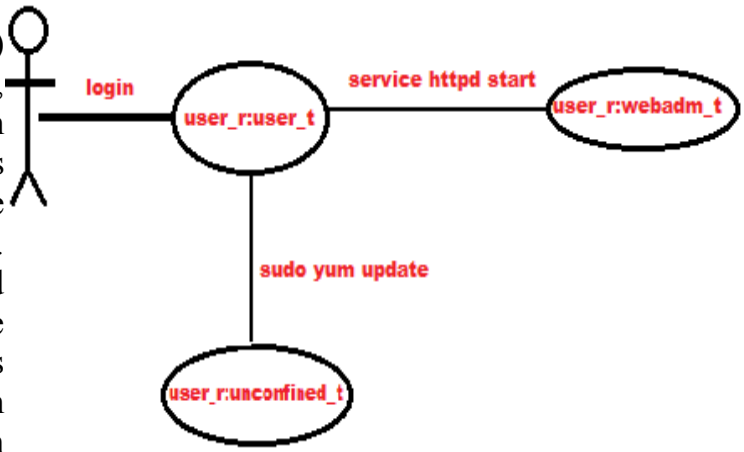
Security Enhanced for Linux
Travail de Bachelor
Session 2013

Professeur Responsable : LITZISTORF Gérald
Diplômant : CARLI Kevin
Filière ITI
Genève, le 10 Juillet 2013

Résumé



Security Enhanced Linux (SELinux) est une technologie assez récente, pour sécuriser son système et son réseau. Elle implémente plusieurs modèles de contrôle d'accès tels que le modèle TE, RBAC et MCS/MLS. Le but de ce projet est tout d'abord d'acquérir un nouveau savoir dans le domaine de la sécurité de Linux. Puis dans un second temps de mettre en pratique ces connaissances dans un cas classique et concret, qui peut servir de base pour de futurs projets plus conséquents.

L'étude se déroule en onze semaines et a été réalisée en différentes étapes.

- Prise en main de Fedora 18, des outils d'analyse de règles et d'administration.
- Accomplissement du travail de laboratoire sur l'exploitation d'un exploit zéro day sur un serveur FTP.
- Accomplissement de mon propre travail de laboratoire visant à confiner différents groupes, suivant leurs rôles.

Au final j'ai réussi à :

- Mettre en place des règles SELinux autorisant l'accès à un/des objet(s) spécifique(s) pour des personnes spécifiques.
- Effectuer des transitions de domaines, pour pouvoir exécuter un service appartenant à un domaine différent du nôtre.
- Appliquer les deux points précédents dans un cas concret.
- Utiliser des outils propres à l'écriture de règle SELinux.
- Rechercher et tester des solutions pour déployer pfSense grâce à un serveur PXE.

J'ai commencé par effectuer un scénario classique d'accès à un répertoire, cette première approche pratique m'aura pris une semaine à réaliser. Elle m'a a permis entre autre de mieux comprendre comment les modules étaient créés et de me familiariser avec la commande sesearch et audit2allow.
Dès lors que j'ai compris le mécanisme, j'ai poussé les recherches plus loin, en essayant de comprendre comment on pouvait étendre nos privilèges dans un cas précis. J'ai alors commencé à rechercher dans le code source même des modules du système, et au bout de deux semaines, j'ai finalement compris comment cela était possible.
Au final, j'ai décidé de faire un projet qui regroupait toutes les notions acquises auparavant, que j'ai réussi à mener à bien en quelques semaines.

h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

Automne 2013
Session de bachelor

INGÉNIERIE DES TECHNOLOGIES DE L'INFORMATION
ORIENTATION – LOGICIELS ET SYSTÈMES COMPLEXES
SELINUX & PFSENSE

Descriptif :

Ce travail fait suite aux résultats présentés lors de la session 2012 par :

- Khaled Basbous → http://www.tdeig.ch/kvm/Basbous_ETB.pdf
- Michaël Golliet → http://www.tdeig.ch/kvm/Golliet_ETB.pdf

L'objectif de la première partie, consacrée à SELinux, est d'étudier les différentes solutions existantes pour générer et adapter un module de règles de contrôle RBAC.

Un administrateur système est amené à établir des rôles en fonction des besoins fonctionnels des entreprises. Ces rôles ont différents types et niveaux d'accès aux ressources.

La deuxième partie, en relation avec la solution pfSense, doit permettre de faciliter l'utilisation de ce logiciel lors des travaux de laboratoire effectués par les étudiants.

L'objectif consiste à supprimer le CD d'installation afin de permettre un chargement via PXE et d'offrir une interface (menu) appropriée aux besoins des étudiants.

Travail demandé :

Ce travail comprend les parties suivantes :

1) Établir un cas d'utilisation RBAC (rédacteur, lecteur, éditeur)
   Étude des différentes solutions (seedit, sepolgen, SLIDE...)
   Développement du module et intégration à la politique SELinux du système
   Test fonctionnel

2) Créer un Live CD approprié aux besoins du labo à partir du Live CD disponible sur www.pfsense.org/
   Utiliser le chargement PXE pour supprimer le support CD
   Spécifier les divers menus facilitant les travaux de laboratoire existants
   Si le temps le permet, définir puis configurer divers scénarios VPN (Virtual Private Network)

Sous réserve de modifications en cours du travail de Bachelor

Candidat :
M. CARLI KEVIN
Filière d'études : ITI
Orientation TIC

Professeur(s) responsable(s) :
Litzistorf Gérald

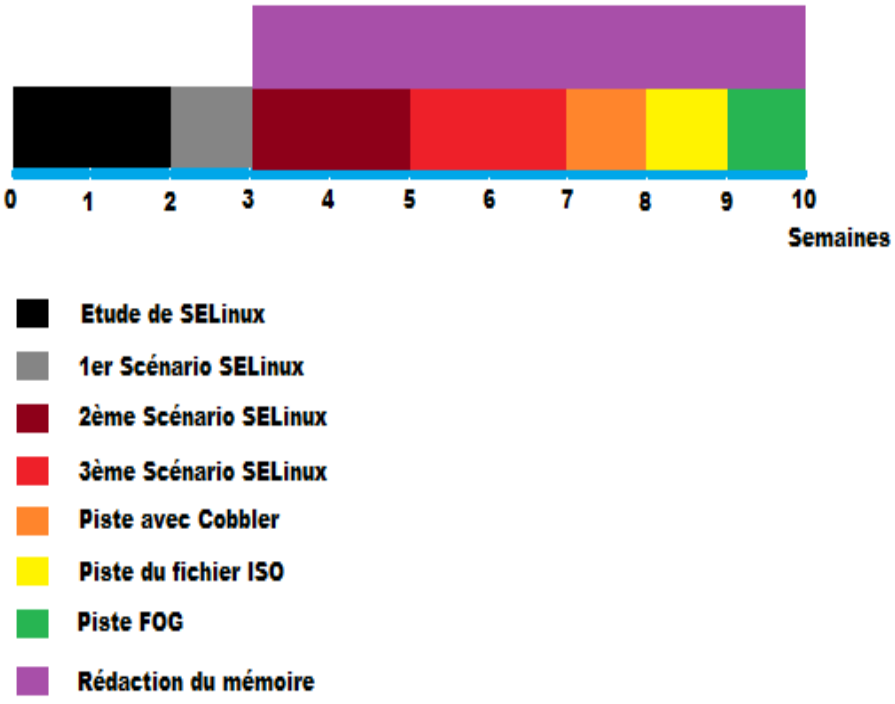
En collaboration avec :
Travail de bachelor soumis à une convention de stage en entreprise : non
Travail de bachelor soumis à un contrat de confidentialité : non

Timbre de la direction

Préambule

En préambule de ce mémoire, je souhaiterais remercier
Monsieur Litzistorf pour ses conseils utiles concernant la rédaction de ce mémoire.
Je remercie également M. Khaled Basbous pour m'avoir partagé son savoir de SELinux ainsi que pour ses nombreux conseils. Et M.Benoit-Georges Chalut pour m'avoir accordé du temps, et des réponses à mes questions.

Répartition du temps de tavail



L'environnement

J'ai installé la distribution Fedora 18 en 64 bits sur une machine physique du laboratoire.

Utilisateur : root	**Mot de passe : root**
Utilisateur : test	**Mot de passe : test**
Utilisateur : VivienBis	**Mot de passe : VivienBis**
Utilisateur : SamBis	**Mot de passe : SamBis**
Utilisateur : MaximeBis	**Mot de passe : MaximeBis**
Utilisateur : VivianneBis	**Mot de passe : VivianneBis**

Table des matières

Liste des Figures

1) Etude Théorique

1.1) Introduction

Dans tout système d'exploitation le contrôle d'accès est basé sur des règles de sécurité liant des objets avec des sujets. Dans SELinux le contexte de sécurité est définie selon trois éléments. On a tout d'abord l'utilisateur, le rôle, puis le type. Ce contexte est écrit de la façon suivante :

user:role:type

chaque élément de ce contexte doit bien entendu être un élément valide.
Grâce à la commande -Z qu'intègre ce module de sécurité, il est possible de modifier de nombreuses commandes existantes. Par exemple la commande *ls -Z,* liste tous les fichiers de l'endroit où on se trouve, mais rajoute chaque contexte de sécurité pour chaque fichier ou objet.

Tandis que dans Linux standard le contrôle d'accès à une ressource est régi par trois groupes de read/write/execute, un pour le propriétaire de la ressource, le suivant pour le groupe, et le dernier pour n'importe qui d'autre.
Dans SELinux ce contrôle est également présent, mais on a une couche en plus, celle du contexte de sécurité du TE.

On peut en conclure que ce module, nous offre une protection supplémentaire, à condition qu'il soit bien configuré. En effet comme décrit et démontré plus tard, une mauvaise configuration revient au même que de ne rien avoir.

La question qu'on peut donc se poser, est comment peut on alors bien configurer notre système ?
La réponse n'est pas triviale, et repose sur plusieurs aspects que je vais vous détailler. Tout d'abord nous allons voir la notion du Type Enforcement, puis la notion de rôle à travers Rôle Based Access Control, RBAC. Et finalement le MultiLevelSecurity ou MLS.

1.2) Le contrôle d'accès par le Type Enforcement

1.2.1) Introduction

Dans SELinux, chaque accès doit être explicitement autorisé. En effet par défaut rien n'est autorisé. La politique de la Liste Blanche y est de mise.
Contrairement à Linux standard, il n'y a donc pas de superutilisateur root. L'accès est défini pour un type, ou domaine et un objet en utilisant une règle *allow*.

La règle *allow* se compose de quatre éléments :

allow Source_Type Target_Type : Object_Class {Permissions}

Source_Type : Le domaine source essayant d'accéder à une ressource

Target_Type : Le domaine cible de la ressource étant accédé

Object_Class : Si il s'agit d'un fichier d'un répertoire, ou autres

Permissions : Le type d'accès qu'on autorise à la source d'effectuer sur la cible.

Exemple:

allow user_t bin_t : file {read getattr execute};

Ici on autorise un processus du type user_t à lire les attributs et exécuter un fichier du type bin_t.

La majorité de la politique SELinux est constituée de règles, qu'on appelle plus communément règles TE. Plusieurs types de règles existent, ici nous avons vu un exemple avec la règle allow, mais il est à noter que d'autres existent, comme par exemple des règles d'audit, et de transition.
Une bonne politique TE, peut contenir des milliers de règles, ainsi chaque accès de chaque processus pour chaque fichier est autorisé, s'il existent au moins une règle TE le permettant.

1.2.2) La Transition de type

Les règles de type transition aident à améliorer la sécurité de SELinux tout en étant invisibles pour l'utilisateur courant. Dans SELinux, par défaut, quand on crée un dossier par exemple, ce dossier se verra attribuer le même type que le dossier le contenant. Imaginons, que nous venons de créer le dossier Test : /Auteur/Test.
Le type du dossier Auteur est auteur_t. On aura donc notre dossier Test du type auteur_t. Pour pallier à ce défaut il existe la règle *type_transition.*

Cette règle n'autorise en rien un accès, elle permet juste un retypage d'un objet. Cependant pour rendre effective cette transition, il y a derrière quelques autres règles TE :

- La source doit avoir les permissions d'exécuter un fichier du type auquel elle veut transiter vers.
- La source doit pouvoir transiter vers le domaine de destination.

Voici donc un exemple d'une transition réussite :

allow auteur_t apache_exec_t : file execute ;
allow auteur_t apache_t : process transition ;
allow apache_t apache_exec_t : file entrypoint ;

type_transition auteur_t apache_exec_t : process apache_t ;

1.3) Role Based Access, ou RBAC

Le RBAC de SELinux est construit sur les bases du TE. L'utilité des rôles est de regrouper des règles TE, s'appliquant qu'à une certaine catégorie d'utilisateurs.
Par exemple dans une entreprise, il peut y avoir des comptables et des secrétaires. Ces deux groupes appartiendront à deux groupes distincts, et se verront ainsi attribuer des règles et des autorisations différentes.
Il est à noter qu'un utilisateur Linux, peut appartenir à plusieurs groupes SELinux.

Ce mécanisme, n'autorise pas un accès, cependant des privilèges sont attribués indirectement en associant un domaine à un ou plusieurs groupes.

Il existe plusieurs déclarations de règles, la première est la règle *user.*
Cette règle associe un utilisateur avec un rôle.
Sa syntaxe est la suivante :
user Kevin roles {auteur_t} ;
Ici nous associons l'utilisateur Kevin au rôle auteur_t. Grâce à cette règle, l'utilisateur Kevin et le rôle auteur_t, peuvent coexister dans un contexte de sécurité. Sans cette règle se contexte aurait été invalide.

La deuxième règle, est la règle *role,* qui quant à elle associe un rôle avec un type. Tout comme l'autre règle, celle-ci est nécessaire pour que le contexte de sécurité soit valide.
Elle s'écrit comme suit :

role auteur_r types auteur_t ;

1.4)Multilevel Security

MLS est une autre forme du modèle Mandatory Access control, MAC, qui est applicable dans certains genres de problèmes de sécurité.
Ce mécanisme rajoute deux champs additionnels au contexte de sécurité :
Un champ sensitivity et un champ regroupant aucune ou plusieurs catégories.

On définit les sensibilités en utilisant le mot-clé sensitivity, comme cela :
sensitivity s0 ;
sensitivity s1 ;

Les catégories sont définies de la sorte :
category c0 alias blue ;
category c1 alias red ;

Et pour finir on définit une combinaison des deux éléments précédents de cette façon :
level s0:c0.c4 ;
level s1:c0.c2,c4 ;

A noter que le point signifie un intervalle continu de catégories, contrairement à la virgule qui représente un ensemble non continu.
Dans cet exemple s0 sera associé aux catégories c0,c1,c2,c3 et c4, et s1 sera associé aux catégories c0,c1,c2 et c4.

Le but principal d'une politique MLS est de forcer les processus du système à opérer dans un niveau de sécurité donné.

2) Etude Pratique

2.1) Premier Scénario : Auteur-Editeur-Lecteur

2.1.1) Explications

Un auteur appartient à un groupe. Ce groupe possède certains privilèges. Les auteurs sont représentés par des Hommes, les groupes par des ovales, et les privilèges par des notes rectangulaires.
Alice membre du groupe auteur pourra lire, écrire et créer des objets dans le répertoire Test.
Emilie membre du groupe Éditeur pourra lire et écrire des objets dans le répertoire Test.
Lisa membre du groupe Lecteur pourra seulement lire les objets du répertoire Test

2.1.2) Schéma

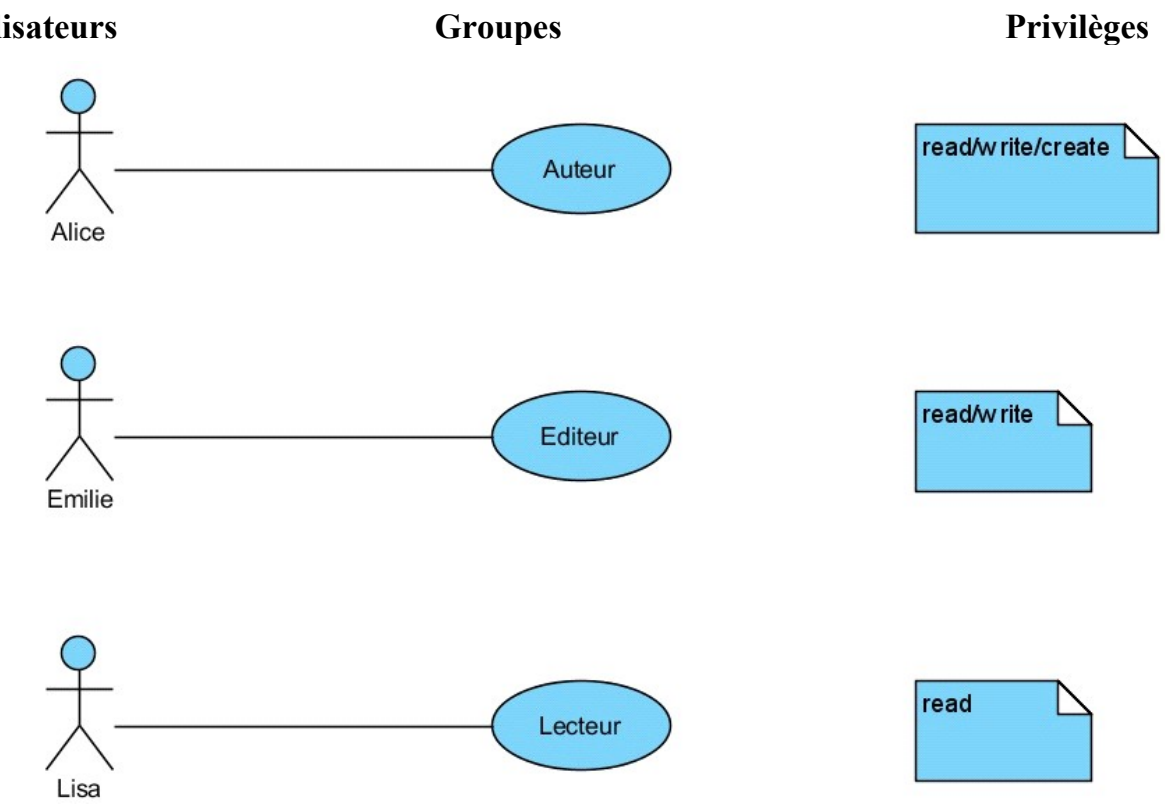


Illustration 1: Auteur-Editeur-Lecteur

2.1.3) Analyse

Comment effectuer ce travail ?
Pour mener à bien ce travail, plusieurs scénarios sont possibles.
- L'approche par Role Based Control Access
- L'approche par Multi-Category Security

Ici je vais vous présenter uniquement la première approche.

2.1.4) Prérequis

Pour comprendre ce dont je vais vous parler, des connaissances de base sur Linux et SELinux sont nécessaires. C'est pour cela que la lecture et l'accomplissement du laboratoire SELinux sont suggérés.
(cf Annexe A.1, p35).

2.1.5) Étapes

- Créer un dossier Journal qui va nous servir de base pour nos tests, car c'est à lui qu'on attribuera le type auteur_t, qui va nous servir pour nos règles *allow*. Ce répertoire Journal aura les droits 777 soir rwx pour tout le monde. Il appartiendra au type auteur_t.(cf Annexe 2.1.11) I).

- Créer trois rôles : auteur_r, editeur_r, et lecteur_r grâce à l'outil de génération de politiques SELinux, sepolgen.

- Créer trois utilisateurs Linux, nommons-les : Alice pour l'auteur, Emilie pour l'éditeur, et Lisa pour le lecteur.

- Créer trois utilisateurs SELinux, respectivement auteur_u, éditeur_u et lecteur_u. Qu'on associe ensuite aux trois utilisateurs Linux ainsi qu'à leurs rôles SELinux.

- Créer les règles Type Enforcement qui permettront de restreindre les privilèges de nos utilisateurs.

2.1.6) Réalisation

- Attribuer au répertoire Journal le droit chmod 777

- Installer sepolgen, présent dans le pacquage policycoreutils-gui.

- Pour y accéder, ouvrir les activités, onglets Outils Systèmes, Outil de génération de politiques SELinux.

- Dans la catégorie Login Users, choisir User Role, pour créer un nouveau rôle. Nommer notre rôle, par exemple en auteur, et cliquer sur suivant jusqu'à ce qu'on puisse terminer.

- Sepolgen nous dit qu'il a créé quatre fichiers. En effet un module d'une police est constitué de plusieurs fichiers. Un fichier .fc (File contexte), un fichier .te (Type Enforcement), qui contient des règles de sécurité, un fichier .if(Interface) et un fichier .sh qui lui va devoir être exécuté pour compiler et installer notre module.

- Créer trois utilisateurs SELinux, aller dans l'outil de gestion SELinux, dans l'onglet Identité SELinux, Ajouter, typer le nom de votre identité, par exemple auteur_u, et

rajouter lui les rôles staff_r pour pouvoir démarrer une session, system_r et le rôle que vous avaez créé précédemment (ie auteur_r).

- Aller dans l'onglet Correspondance d'utilisateurs, et rajouter nos utilisateurs Linux avec l'identité SELinux correspondante. Idem pour les deux rôles editeur_r et lecteur_r.

- Créer un fichier texte vierge, qu'on appelle *règles.te,* avec les règles nécessaires au bon fonctionnement de notre laboratoire. (cf Annexes 2.1.11) II).

Pour intégrer ce fichier au système exécuter ces commandes :
Créer le fichier .pp correspondant :

```
[root@localhost home]# make -f /usr/share/selinux/devel/Makefile regles.pp
```

Pour ajouter le module :

```
[root@localhost home]#semodule -i regles.pp
```

2.1.7) Tests

1) Vérifier que le dossier appartient au type auteur_t :

```
[root@localhost home]# ls -Z
drwxrwxrwx. test     test     unconfined_u:object_r:auteur_t:s0 Journal
```

2) Alice créée un fichier texte.

3) Emilie fait de même. Cela ne fonctionne pas.

4) Emilie écrit dedans. Cela marche.

5) Répéter les opérations 3 et 4 avec Lisa. Cela ne marche pas, Lisa ne peut que le lire.

2.1.8) Difficultés rencontrées

L'écriture du fichier règles.te ne fût pas très simple, de par la prise en main de cette syntaxe, qui est celle de l'écriture de règles, ainsi que par la découverte de tous les droits SELinux associés aux classes SELinux.

2.1.9) Liens et Références

http://www.tdeig.ch/kvm/Basbous_RTB.pdf(pages 1-52).

2.1.10) Conclusions

Les règles TE, ainsi que les trois rôles précédemment créés ont bien fonctionné.
Ce travail m'a permis d'approfondir mes connaissances dans l'écriture de règles TE, ainsi que dans la notion de rôle SELinux.
SELinux possède une panoplie de classes, de rôles et de domaines. Pour un exemple simple, on peut se permettre de créer nos propres rôles et règles, cependant dans un cas plus complexe, il est plus judicieux d'utiliser ce qui existe déjà. En effet la multitude de dépendances rendent l'écriture de règles vite complexe, et fastidieuse. Nous verrons un exemple par la suite de réutilisation de l'existant.

2.1.11) Annexes

Logiciel utilisé pour le schéma : Visual Paradigm for UML.

I)Si le répertoire Journal n'appartient pas au bon domaine, utiliser la commande suivante :

```
[root@localhost home]# chcon ~/Journal -t auteur_t
```

II)*Fichier règles.te*

```
module regles 1.0;

require {
        type editeur_t;
        type auteur_t;
        type lecteur_t;
        class file { entrypoint rename execute setattr read lock getattr create write execute_no_trans ioctl unlink open append };
        class dir { search setattr read write getattr remove_name open add_name create };
};

#============= editeur_t ==============

allow editeur_t auteur_t :dir { search read open write getattr };
allow editeur_t auteur_t :file { read write execute getattr open };

#============= lecteur_t ==============

allow lecteur_t auteur_t :dir { getattr search read open };
allow lecteur_t auteur_t :file { read execute getattr open };

#============= auteur_t ==============

allow auteur_t auteur_t :dir { read write search setattr getattr remove_name open add_name };
allow auteur_t auteur_t :file { rename execute setattr read lock getattr create write execute_no_trans ioctl unlink open append };
```

2.2) Deuxième Scénario : Transition de domaine

2.2.1) Objectifs

Reprendre le travail précédent et rajouter la fonctionnalité suivante :
- Permettre au domaine lecteur_t d'administrer un serveur apache via une transition vers le domaine webadm_t.

2.2.2) Étapes de Réalisation:

Ici deux cas distincts pourraient être envisageables :

- Le premier serait d'écrire les règles soi-même.

- Le second cas serait d'implémenter une architecture qui transiterait notre domaine lecteur_t en webadm_t, en se basant sur des règles et modules qui existent déjà. Cette architecture se nomme une interface. Dans cette alternative des outils tel que SELinux policy IDE, SLIDE, sont une aide précieuse.

2.2.4) Schéma

Utilisateur **contexte avant transition** **contexte après transition**

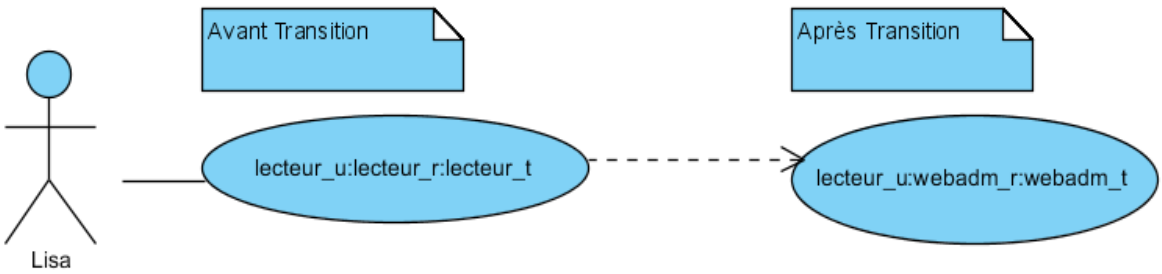


Illustration 2: Transition de domaine

.

2.2.5) Réalisation

J'ai commencé par le premier cas, mais comme dis au paragraphe Conclusions, du chapitre 2.2.10
« La multitude de dépendances rend l'écriture de règles vite complexe, et fastidieux. ».
La multitude venait ici du fait que pour exécuter le service Apache avec le bon domaine httpd_t et non unconfined_t, il fallait utiliser la commande :

```
[root@localhost home]# sudo service http start
```

Derrière la commande sudo se cachent de nombreuses règles !
Étude du second cas, soit la reprise de modules déjà existant pour compléter le mien. Pour se faire il m'a fallu un outil nommé SLIDE.

2.2.6) Rapide Présentation

SLIDE est un projet de tresys Technology, qui a été réalisé sous licence GPL. Il a pour but de faciliter le processus d'élaboration de l'écriture de police SELinux. Pour ce faire il utilise l'environnement de développement d'éclipse. Il offre quelques fonctionnalités telles que :

- La coloration Syntaxique
- L'auto complétion
- Une recherche intégrée

- Télécharger et installer éclipse (cf Annexes 2.2.13) II et III) :

```
[root@localhost home]# tar-xvzf eclipse-SDK-4.2.2-linux-gtk.tar.gz -C /opt
```

- Ajouter les permissions pour touts les fichiers :

```
[root@localhost home]# chmod -R +r /opt/eclipse
```

- Créer un exécutable pour éclipse :

```
[root@localhost home]# touch /usr/bin/eclipse
[root@localhost home]# chmod 755 /usr/bin/eclipse
```

- Éditer le fichier créé :

Fichier /usr/bin/eclipse

```
#!/bin/sh
export ECLIPSE_HOME="/opt/eclipse"

$ECLIPSE_HOME/eclipse $*
```

- Lancer éclipse

- Une fois éclipse installé, il faut lui implémenter SLIDE. Pour ce faire il faut aller dans le menu help -> Install new Software et rajouter le site : (cf Annexe 2.2.13) I).

```
Name: Tresys Technology
URL: http://oss.tresys.com/eclipse-update/
```

- Créer un nouveau Projet : File -> New -> SLIDE Project.
- Clique droit sur le projet créé -> Add -> New -> SLIDE Module.
- SLIDE nous génère trois fichiers.
- Grâce à la fenêtre interfaces à droite et à l'option de recherche on trouve rapidement les macros utiles

Pour plus de détails sur SLIDE (cf Annexe A.2, p38)

```
sudo_role_template(lecteur, lecteur_r, lecteur_t)
webadm_role_change(lecteur_r)
```

Modification du fichier lecteur.te du module lecteur créé avec sepolgen :

lecteur.te

```
policy_module(lecteur, 1.0.0)

########################################
#
# Declarations
#
userdom_unpriv_user_template(lecteur)

role lecteur_r;
domain_type(lecteur_t)

sudo_role_template(lecteur, lecteur_r, lecteur_t)
webadm_role_change(lecteur_r)
```

Changement du fichier visudo

```
lisa ALL=(ALL) ROLE=webadm_r TYPE=webadm_t /sbin/service
lisa ALL=(lisa) ROLE=webadm_r TYPE=webadm_t ALL
```

Changement du fichier /etc/selinux/targeted/contextes/user/lecteur_u :

```
cp /etc/selinux/targeted/contextes/users/staff_u /etc/selinux/targeted/contextes/users/lecteur_u
sed -i 's/staff/lecteur/g' /etc/selinux/targeted/contexts/users/lecteur_u
```

lecteur_u

system_r:local_login_t:s0	lecteur_r:lecteur_t:s0 sysadm_r:sysadm_t:s0
system_r:remote_login_t:s0	lecteur_r:lecteur_t:s0
system_r:sshd_t:s0	lecteur_r:lecteur_t:s0 sysadm_r:sysadm_t:s0
system_r:crond_t:s0	lecteur_r:lecteur_t:s0
system_r:xdm_t:s0	lecteur_r:lecteur_t:s0
lecteur_r:lecteur_su_t:s0	lecteur_r:lecteur_t:s0
lecteur_r:lecteur_sudo_t:s0	lecteur_r:lecteur_t:s0
system_r:initrc_su_t:s0	lecteur_r:lecteur_t:s0
lecteur_r:lecteur_t:s0	lecteur_r:lecteur_t:s0
sysadm_r:sysadm_su_t:s0	sysadm_r:sysadm_t:s0
sysadm_r:sysadm_sudo_t:s0	sysadm_r:sysadm_t:s0

- Attribution du rôle webadm_r à Lisa.

2.2.7) Tests

Lisa peut lancer et administrer le serveur apache en transitant vers le domaine webadm_t. De plus les anciens droits du tp précédent sont conservés.

2.2.8) Première Piste Suivie

- Changement du fichier /etc/selinux/targeted/contextes/user/lecteur_u :

lecteur_u

system_r:local_login_t:s0	staff_r:staff_t:s0 sysadm_r:sysadm_t:s0
system_r:remote_login_t:s0	staff_r:staff_t:s0
system_r:sshd_t:s0	staff_r:staff_t:s0 sysadm_r:sysadm_t:s0
system_r:crond_t:s0	staff_r:staff_t:s0
system_r:xdm_t:s0	staff_r:staff_t:s0
staff_r:staff_su_t:s0	staff_r:staff_t:s0
staff_r:staff_sudo_t:s0	staff_r:staff_t:s0
system_r:initrc_su_t:s0	staff_r:staff_t:s0
staff_r:staff_t:s0	staff_r:staff_t:s0
sysadm_r:sysadm_su_t:s0	sysadm_r:sysadm_t:s0
sysadm_r:sysadm_sudo_t:s0	sysadm_r:sysadm_t:s0

- Changement du fichier /etc/sudoers

```
lisa ALL=(ALL) ROLE=webadm_r TYPE=webadm_t /sbin/service
lisa ALL=(lisa) ROLE=webadm_r TYPE=webadm_t ALL
```

- Attribuer le rôle webadm_r et staff_r à Lisa

2.2.9) Explications

Cette première approche marche. En effet comme le rôle staff_r possède les privilèges pour exécuter un sudo, je me suis dit que si le fichier lecteur_u était comme celui du staff_u, Lisa hériterait d'un id-Z :

```
lecteur_u:staff_r:staff_t
```

Et de ce fait pourrait exécuter sudo. Cependant ici elle acquiert trop de privilèges.

2.2.10) Deuxième Piste Suivie

- Obtenir les sources de webadm.te (cf Annexe 2.2.13) V).

- Écrire un fichier Transition.te qui reprend le fichier webadm.te pour avoir les mêmes privilèges.

Transition.te

```
policy_module(Transition, 0.0.1)
role Transition_r;
userdom_base_user_template(Transition)
allow Transition_t self:capability { dac_override dac_read_search kill sys_ptrace sys_nice };
files_dontaudit_search_all_dirs(Transition_t)
selinux_get_enforce_mode(Transition_t)
seutil_domtrans_setfiles(Transition_t)
logging_send_syslog_msg(Transition_t)
userdom_dontaudit_search_user_home_dirs(Transition_t)
```

- Obtenir les sources de webadm.if pour repérer la ligne importante.

- Pour administrer apache rajouter cette ligne dans le fichier Transition.te :

```
apache_admin(Transition_t, Transition_r)
```

- Écrire le fichier Transition.if d'après les sources d'un fichier .if généré par sepolgen

- Créer la macro Transition_role_change permettant de transiter vers le domaine Transition_t.

Transition.if

```
interface(`Transition_role_change',`
 gen_require(`
 role Transition_r;
 ')
 allow $1 Transition_r;
')
```

- Écrire un deuxième fichier .te qui reprend les sources du fichier staff.te pour avoir les droits sudo (cf Annexe 2.2.13) VI).

RoleTr.te

```
policy_module(RoleTr, 0.0.1)
role RoleTr_r;
userdom_unpriv_user_template(RoleTr)
sudo_role_template(RoleTr, RoleTr_r, RoleTr_t)
ssh_role_template(RoleTr, RoleTr_r, RoleTr_t)
kernel_read_ring_buffer(RoleTr_t)
kernel_getattr_core_if(RoleTr_t)
kernel_getattr_message_if(RoleTr_t)
kernel_read_software_raid_state(RoleTr_t)
auth_domtrans_pam_console(RoleTr_t)
libs_manage_shared_libs(RoleTr_t)
seutil_run_newrole(RoleTr_t, RoleTr_r)
netutils_run_ping(RoleTr_t, RoleTr_r)
domain_read_all_domains_state(RoleTr_t)
domain_getattr_all_domains(RoleTr_t)
domain_obj_id_change_exemption(RoleTr_t)
files_read_kernel_modules(RoleTr_t)
kernel_read_fs_sysctls(RoleTr_t)
modutils_read_module_config(RoleTr_t)
modutils_read_module_deps(RoleTr_t)
miscfiles_read_hwdata(RoleTr_t)
term_use_unallocated_ttys(RoleTr_t)
```

- Pour transiter de RoleTr_r a Transition_r rajouter cette ligne dans le fichier RoleTr.te :

```
Transition_role_change(RoleTr_r)
```

- Créer un nouvel utilisateur SELinux RoleTr_u qui permet à l'utilisateur Linux correspondant d'utiliser la transition.

- Exécuter la commande :

```
gen_user(RoleTr_u, user, Transition_r system_r RoleTr_r, s0, s0 -mls_systemhigh, mcs_allcats)
```

- Copier le fichier /etc/selinux/targeted/contexte/users/staff_u RoleTr_u pour permettre à l'utilisateur SELinux RoleTr_u de se loguer sur la machine.

- Remplacer chaque occurrence de «staff» par «RoleTr» grâce à la commande

```
sed -i 's/staff/editeur/g' /etc/selinux/targeted/contextes/users/editeur_u
```

- Créer un nouveau utilisateur Linux Kevin, lui associer comme user SELinux RoleTr_u.

- Rajouter dans le fichier /etc/sudoers :

```
Kevin ALL=(ALL) ROLE=Transition_r TYPE=Transition_t ALL
```

- Se loguer avec Kevin.

- Exécuter sudo service httpd start

- Vérifier que ça marche.

2.2.11) Explications

Cette méthode marche, car elle reprend les permissions du fichier webadm.te et webadm.if, lui permettant de gérer un serveur apache. Ainsi que les permissions du fichier staff.te qui nous permettent d'exécuter la commande sudo. De plus grâce à notre fichier Transition.if, nous avons désormais une macro nous permettant de changer de rôle.

La solution optimale reprend cette méthodologie, mais en simplifiant la chose. C'est à dire qu'on ne garde que les lignes de codes nécessaires.

2.2.12) Conclusions

Ce tp est un approfondissement de l'écriture de règles SELinux. Il m'a permis de me rendre compte de la multitude de règles qu'un module peut contenir et la complexité de celles-ci. Le fait de réécrire soit même des règles est un travail fastidieux et pas très judicieux.
Cependant si on cherche dans les sources des modules présents sur le système, on s'aperçoit que ces règles ont été écrites de façon à être réutilisés facilement, par le biais des macros, ou interfaces.

2.2.13) Annexes

I)Installation SLIDE Using the Eclipse Update Site
http://oss.tresys.com/projects/slide/wiki/download

II)Téléchargement d'ecplise
http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/juno/SR2/eclipse-jee-juno-SR2-linux-gtk-x86_64.tar.gz

III)Installation d'eclipse
http://www.if-not-true-then-false.com/2010/linux-install-eclipse-on-fedora-centos-red-hat-rhel/

IV)Aide Transition
http://selinux-mac.blogspot.ch/2009/06/selinux-lockdown-part-six-customized.html

V)Source webadm.te
http://oss.tresys.com/projects/clip/browser/packages/clip-selinux-policy/clip-selinux-policy/policy/modules/roles/webadm.te

VI)Source staff.te
http://oss.tresys.com/projects/clip/browser/packages/clip-selinux-policy/clip-selinux-policy/policy/modules/roles/staff.te

2.3) Troisième Scénario : Stockage-Management-Applicatif

2.3.1) Explications

Trois rôles SELinux

- Gérer les trois VMS
- Sauvegarder les disques VMS (Mount, pool) sur le disque distant
- Mise à jour du système

Quatre acteurs

- Deux membres du groupe Applicatif, qui vont gérer leurs VMS respectifs.
- Un membre du groupe Storage qui va gérer les disques de stockage, un local et un distant.
- Un membre du groupe Management qui va mettre à jour le système Linux

Conditions

- Un membre d'un groupe ne peut entreprendre des actions que par rapport à son groupe uniquement.
- Chaque membres se connecte via l'interface réseau du groupe Management.
- Politique du moindre privilèges.

Cadre

- Trois VMS basés sur l'image Ubuntu Server du labo KVM
- Se connecter en SSH sur NIC3
- VMS utilise la NAT
- Un serveur NFS mis à disposition
- Aucune restriction pour l'administration en local de l'hyperviseur (Virt Manager)
- Sam possède le rôle Storage_r
- Vivien possède le rôle VmManager_r
- Maxime possède le rôle Manager_r

2.3.2) Schéma

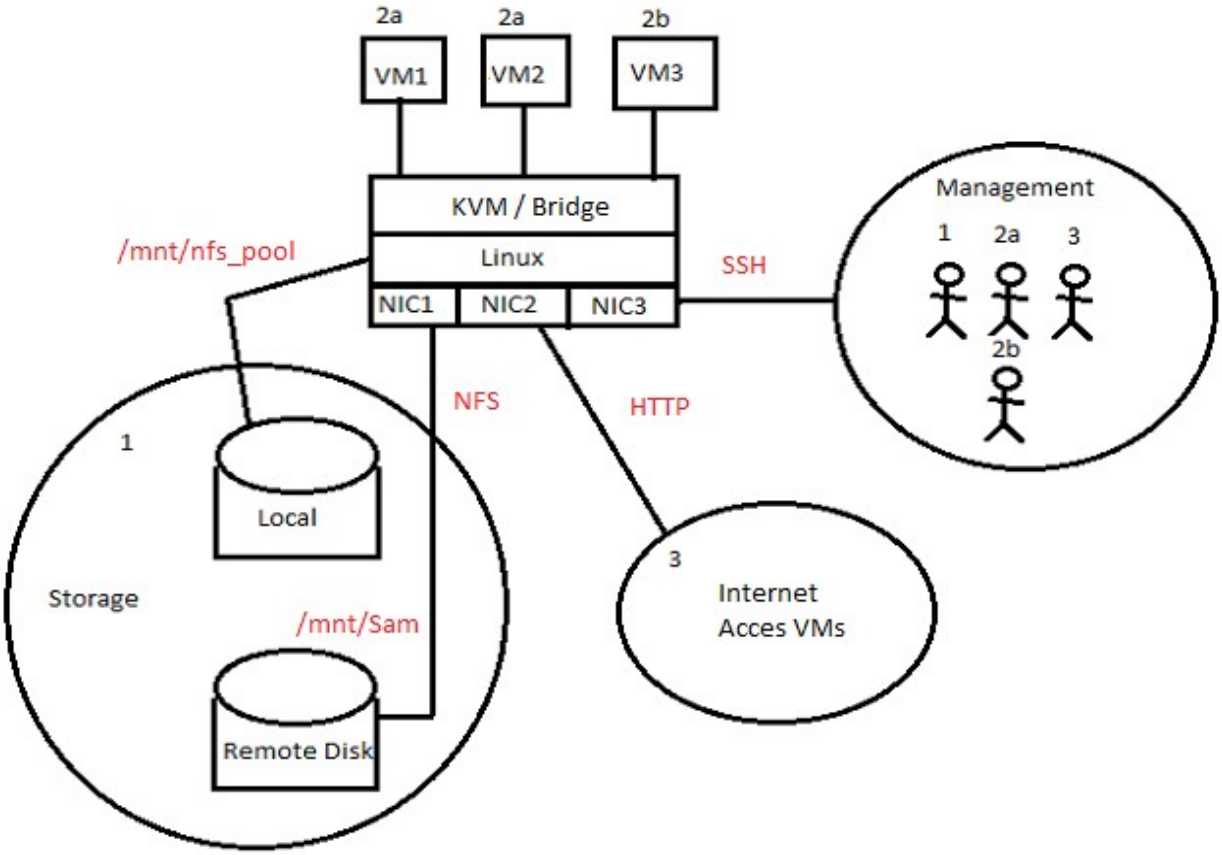


Illustration 3: Stockage-Management-Applicatif

2.3.3) Étapes de réalisation:

- Charger VMS
- Configuration du réseau
- Analyser les domaines

2.3.4) Réalisation

- Créer l'infrastructure suivant le schéma
- Écrire les règles SELinux

2.3.4.1) Créer l'infrastructure

- Installer virt manager
- Créer trois VMS (cf **1**)
- Configuration VMS (cf **2**)
- Création du pool local (cf **3**)
- Création du pool distant (cf **4**)
- Création des trois rôles SELinux / utilisateurs (cf 2.1 - Réalisation)

1) Outil utilisé: virt Manager

- Créer trois VMS

2) Configuration

Pour chaque VMS
Clique droit -> ouvrir Onglet Afficher -> Détails -> Ajouter un matériel -> Network :

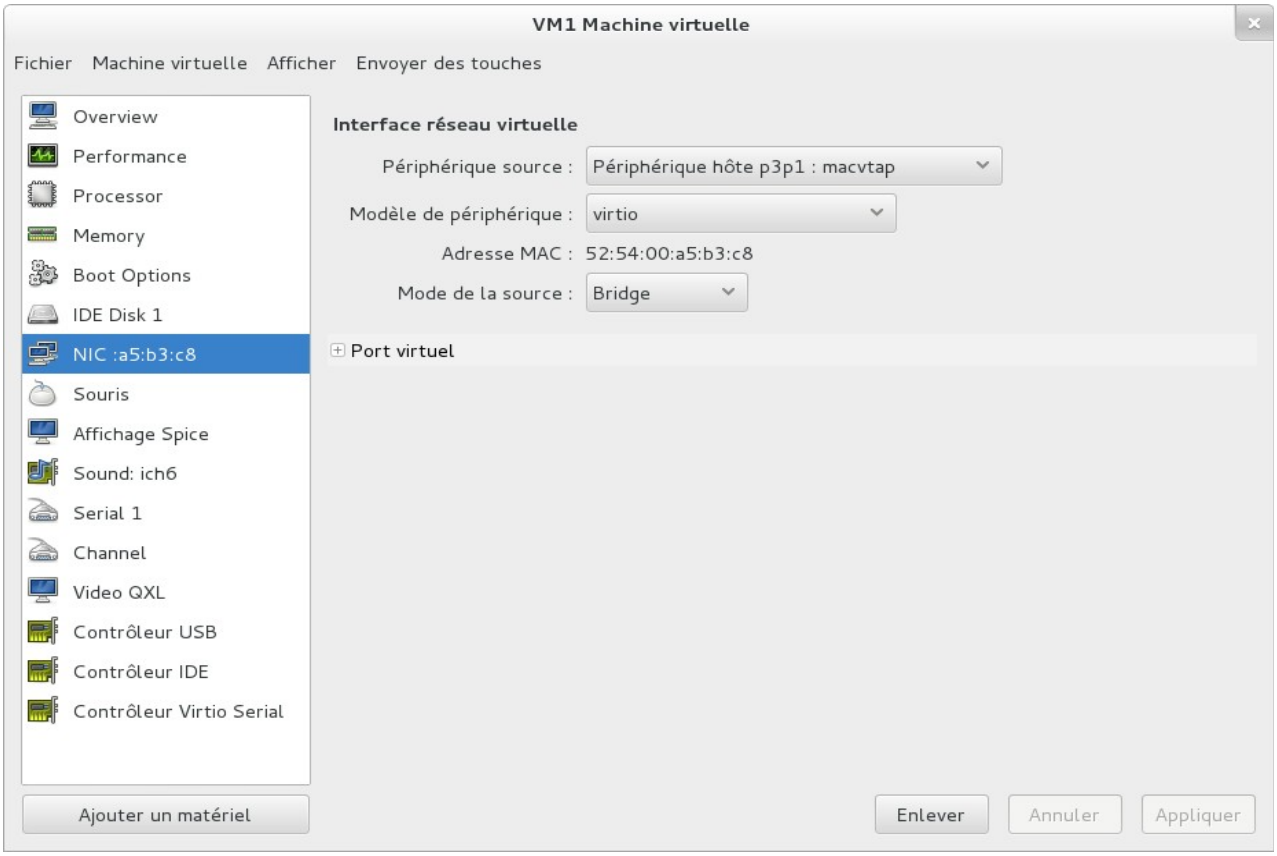


Illustration 4: Config VMS

Pour chaque VMS avoir cette configuration, si on considère que notre câble Ethernet est branché sur l'interface p3p1. Sinon changer.

3) Création du pool local
(cf Labo Live Migration chapitres 1, 2 et 3)
Chemin du pool: **/mnt/nfs_pool**

4)Création du pool distant

```
mount -t nfs4 10.2.1.1:/Sam /mnt/Sam
```

2.3.4.2)Écrire les règles SELinux

Pour le groupe Manager

Manager.te

```
policy_module(Manager, 1.0.0)

########################################
#
# Declarations
#
userdom_unpriv_user_template(Manager)

########################################
#
# Manager local policy
#

type Manager_port_t;
corenet_port(Manager_port_t)

sysnet_dns_name_resolve(Manager_t)
corenet_all_recvfrom_unlabeled(Manager_t)

allow Manager_t self:tcp_socket create_stream_socket_perms;
corenet_tcp_sendrecv_generic_if(Manager_t)
corenet_tcp_sendrecv_generic_node(Manager_t)
corenet_tcp_sendrecv_all_ports(Manager_t)
corenet_tcp_bind_generic_node(Manager_t)
corenet_tcp_bind_ssh_port(Manager_t)
corenet_tcp_connect_ssh_port(Manager_t)

allow Manager_t self:udp_socket { create_socket_perms listen };
corenet_udp_sendrecv_generic_if(Manager_t)
corenet_udp_sendrecv_generic_node(Manager_t)
corenet_udp_sendrecv_all_ports(Manager_t)
allow Manager_t Manager_port_t:udp_socket name_bind;
corenet_udp_bind_generic_node(Manager_t)
```

```
domain_use_interactive_fds(Manager_t)

files_read_etc_files(Manager_t)

miscfiles_read_localization(Manager_t)

role Manager_r;
domain_type(Manager_t)

sudo_role_template(Manager, Manager_r, Manager_t)
unconfined_role_change(Manager_r)
```

/etc/sudoers

```
maxime ALL=(ALL) ROLE=unconfined_r TYPE=unconfined_t /bin/yum
maxime ALL=(maxime) ROLE=unconfined_r TYPE=unconfined_t ALL
```

Tests

Depuis la session à Maxime :

- Ouvrir un terminal
- Type sudo yum update

Pour se connecter en ssh :

- Démarrer le service ssh sur Fedora
- Typer sudo service sshd start
- Depuis le poste Windows installer putty
- Se connecter à la machine en tant que maxime
- Typer sudo yum update
- Vérifier que ça marche

Pour le groupe VmManager

VmManager.te

```
policy_module(VmManager, 1.0.0)

########################################
#
# Declarations
#
userdom_unpriv_user_template(VmManager)

########################################
#
# VmManager local policy
#
```

```
type VmManager_port_t;
corenet_port(VmManager_port_t)

sysnet_dns_name_resolve(VmManager_t)
corenet_all_recvfrom_unlabeled(VmManager_t)

allow VmManager_t self:tcp_socket create_stream_socket_perms;
corenet_tcp_sendrecv_generic_if(VmManager_t)
corenet_tcp_sendrecv_generic_node(VmManager_t)
corenet_tcp_sendrecv_all_ports(VmManager_t)
corenet_tcp_bind_generic_node(VmManager_t)
corenet_tcp_bind_http_port(VmManager_t)
corenet_tcp_connect_http_port(VmManager_t)

allow VmManager_t self:udp_socket { create_socket_perms listen };
corenet_udp_sendrecv_generic_if(VmManager_t)
corenet_udp_sendrecv_generic_node(VmManager_t)
corenet_udp_sendrecv_all_ports(VmManager_t)
allow VmManager_t VmManager_port_t:udp_socket name_bind;
corenet_udp_bind_generic_node(VmManager_t)

domain_use_interactive_fds(VmManager_t)

files_read_etc_files(VmManager_t)

miscfiles_read_localization(VmManager_t)

role VmManager_r;
domain_type(VmManager_t)

sudo_role_template(VmManager, VmManager_r, VmManager_t)
unconfined_role_change(VmManager_r)
```

/etc/sudoers

```
vivien ALL=(ALL) ROLE=unconfined_r TYPE=unconfined_t /sbin/service
vivien ALL=(vivien) ROLE=unconfined_r TYPE=unconfined_t ALL
```

Tests

Depuis la session a Vivien :

- Ouvrir un terminal
- sudo service libvirtd start
- sudo virt-manager

Vérification que la VM est accessible depuis l extérieur par le port 80 :

- Installer apache2 sur la VM

```
sudo apt-get install apache2
```

- Lancer le serveur

```
/etc/init.d/apache2 start
```

- Depuis une machine Windows, lancer le navigateur
- Typer IPSERVEURUBUNTU:80

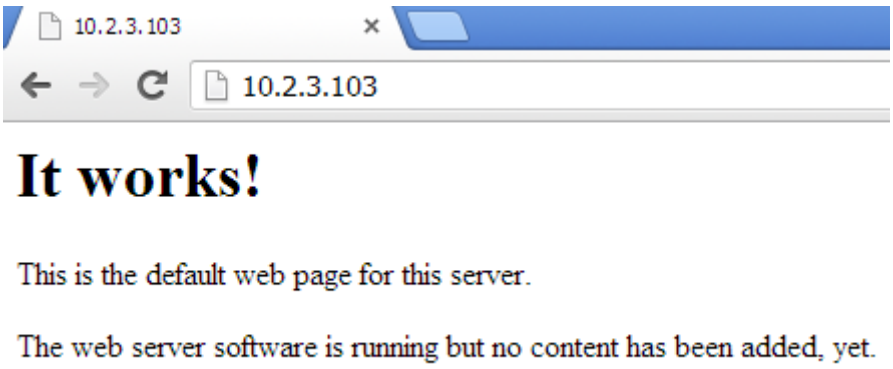


Illustration 5: Serveur VM

- Observer que la machine 10.2.3.103 n'est accessible que par le port 80

```
nmap -T4 -F 10.2.3.103

Starting Nmap 6.01 ( http://nmap.org ) at 2013-06-21 10:11 W. Europe Daylight Time
Nmap scan report for 10.2.3.103
Host is up (0.00s latency).
Not shown: 99 closed ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 52:54:00:0D:0A:A4 (QEMU Virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds
```

Illustration 6: Scan rapide nmap

Illustration 7: Scan intense nmap

Pour le groupe Storage

Rappel

- Pool Local situé : /mnt/nfs_pool
- Pool distant situé : /mnt/Sam

Storage.te

```
policy_module(Storage, 1.0.0)

##############################################
#
# Declarations
#
userdom_unpriv_user_template(Storage)

role Storage_r;
domain_type(Storage_t)



allow Storage_t mnt_t :dir { search setattr read write getattr remove_name open add_name create mounton };


allow Storage_t mnt_t :file { entrypoint rename execute setattr read lock getattr create write execute_no_trans ioctl unlink open append };

allow Storage_t nfs_t :dir { search setattr read write getattr remove_name open add_name create mounton };


allow Storage_t nfs_t :file { entrypoint rename execute setattr read lock getattr create write execute_no_trans ioctl unlink open append };

sudo_role_template(Storage, Storage_r, Storage_t)
mount_domtrans(Storage_t)
```

/etc/sudoers

```
Sam ALL=(ALL) ROLE=Storage_r TYPE=Storage_t /bin/mount
Sam ALL=(Sam) ROLE=Storage_r TYPE=Storage_t ALL
```

- Par défaut la commande utilisée pour monter le pool distant nous met le dossier monter /mnt/Sam, avec comme propriétaire nobody
- Changer le propriétaire pour le dossier /mnt/Sam
- chown -R root /mnt/Sam
- Redémarrer
- Typer les commandes, et vérifier que ça marche:

```
[sam@localhost nfs_pool]# mkdir Mount
```

```
[sam@localhost nfs_pool]# sudo mount -t nfs4 10.2.1.1:/mnt/Sam /mnt/nfs_pool/Mount
```

- Note : Avec un répertoire dont le propriétaire est nobody, malgré l'absence de règles SELinux l'utilisateur Vivien arrivait à écrire dans le fichier /mnt/Sam de type nfs_t

2.3.4.3)Attibution des catégories MCS

Interfaces

lo : s0:c0
p2p1 : s0:c1
p3p1 : s0:c2,c3

Commandes :

```
[root@localhost test]# semanage interface -a -r s0:c0 -t unconfined_t lo
[root@localhost test]# semanage interface -a -r s0:c1 -t unconfined_t p2p1
[root@localhost test]# semanage interface -a -r s0:c2,c3 -t unconfined_t p3p1
```

Vérification :

```
[root@localhost Manager]# cat /etc/selinux/targeted/modules/active/interfaces.local
```

Utilisateurs

Vivien -> s0:c0,c2
Maxime -> s0:c0
Sam -> s0:c0,c1
Vivianne -> s0:c0,c3

Commandes :

```
[root@localhost test]# chcat -l – s0:c0,c2 vivien
[root@localhost test]#  chcat -l – s0:c0 maxime
[root@localhost test]#  chcat -l – s0:c0,c1 sam
```

Problème : si ces commandes ne marchent pas, ouvrir le gestionnaire SELinux, aller dans identités SELinux, cliquer sur l'identité à modifier et rajouter dans l'intervalle MLS/MCS : s0-s0:c0.c1023. Par défaut on à juste s0.

VMS

VM1 et VM2 -> s0:c2
VM3 -> s0:c3

- Double clique sur la VM pour l'ouvrir -> Afficher -> Détails
- Onglet Sécurité
- Modifier l'étiquette

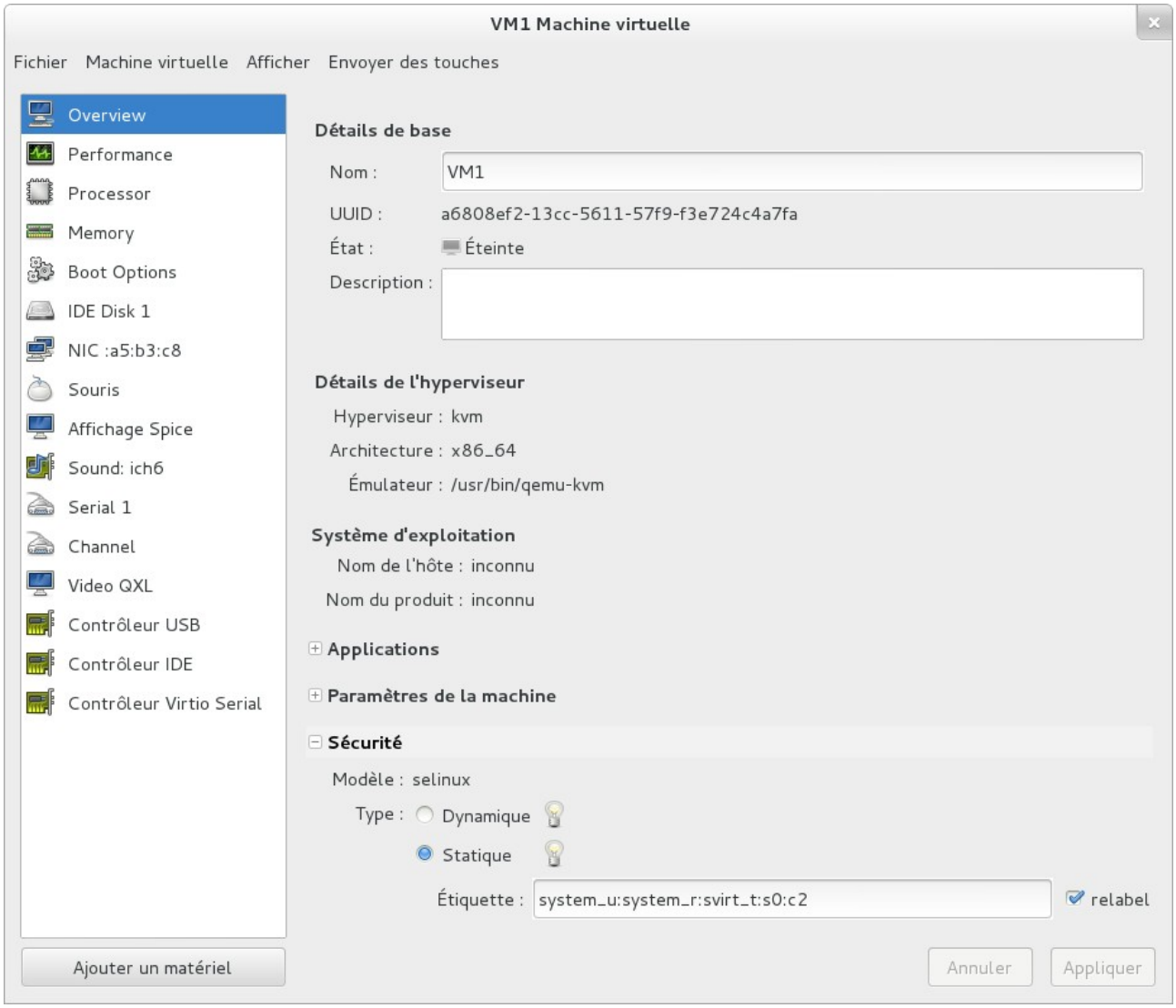


Illustration 8: MCS VMS

2.3.5) Problèmes rencontrés

Malgré le fait que les VMS soit bien assignées à des catégories distincts, un membre de la catégorie c2, par exemple peut lancer une VM de la catégorie c3.

Ceci est dû au fait que ce membre en question a lancé le service libvirtd en tant que unconfined. Le domaine unconfined_t étant non restreint, les catégories MCS sont donc ignorées. Ce qui explique notre problème.

- En cherchant dans les sources du module svirt, deux interfaces on l'air intéressantes :

```
virt_domtrans(VmManagerBis_t)
virt_manage_config(VmManagerBis_t)
```

La première permet une transition de domaine et la seconde permet de gérer les VMS.

- En les rajoutant dans le fichier VmManagerBis.te, tout en gardant la transition vers unconfined
- En modifiant le fichier /etc/sudoers

```
VivienBis ALL=(ALL) ROLE=unconfined_r TYPE=unconfined_t /bin/virt-manager
VivienBis ALL=(ALL) ROLE=unconfined_r TYPE=unconfined_t /sbin/service
VivienBis ALL=(VivienBis) ROLE=VmManagerBis_r TYPE=VmManagerBis_t ALL
```

On obtient déjà un meilleur résultat. L'utilisateur de la catégorie c2, peut toujours lancer une VM de la catégorie c3, mais cependant, en typant : id -Z ou sudo -u VivienBis sh puis id -Z, il appartient toujours à son groupe et domaine de base. Il à beaucoup moins de privilèges qu'avant.

Mais c'est un début de solution.

2.3.6)Conclusion

Ce Tp m'aura permis de mieux maîtriser les transitions de domaines, ainsi que de mieux connaître quels services sont lancés pour telle application, pour mieux attribuer les droits.
Il m'aura également permis d'avoir une approche avec les catégories MLS/MCS, et de les implémenter.
Ce tp est un exemple de cas concret d'utilisation de SELinux pour restreindre chaque groupe à leurs tâches.
Les objectifs auront été atteints hormis pour le groupe VmManager. (cf Problèmes rencontrés)

3) PfSense

3.1)Énoncé

- Créer un Live CD approprié aux besoins du labo à partir du Live CD disponible sur www.pfsense.org/

- Utiliser le chargement PXE pour supprimer le support CD

3.2)Analyse

Au laboratoire nous disposons de CDs d'installation de pfSense, ce qui n'est pas très pratique, car sur chaque poste il faut insérer le CD et répondre aux questions de l'installation. Une erreur peut vite survenir.

Le but serait donc dans un premier temps de supprimer le support CD en déployant pfSense via le serveur PXE, puis dans un second temps d'automatiser l'installation en ayant les réponses aux questions préconfigurées.

En ce qui concerne le déploiement de pfSense, j'ai suivi plusieurs pistes :

- La première piste était d'utiliser l'outil Cobbler, qui est un outil de déploiement qui facilite grandement la tâche. Cependant cette piste fût en réalité un piège car pfSense est basé sur du FreeBSD, et ce système d'exploitation est différent des distributions standard de Linux, que Cobbler prend en charge. De plus un serveur gérant le PXE est déjà présent dans le laboratoire.

- La seconde piste suivie se basait sur le serveur PXE du laboratoire, principalement sur le fichier */tftpboot/pxelinux.cfg/default* qui permet d'ajouter une entrée dans le menu PXE. Cette seconde piste ne fût pas concluante, car le fichier image de pfSense requiert que le cd d'installation soit présent.

- La troisième piste suivie consistait à installer pfSense sur une machine, puis d'effectuer une image de celle-ci grâce à l'outil FOG. Cette troisième piste a de grandes chances de fonctionner, cependant je n'ai pu la terminer, car j'ai perdu du temps, en ne passant pas directement par une machine virtuelle.

3.3) Première Piste : Déploiement via Cobbler d'une distribution Fedora18

3.3.1) Objectifs

Se familiariser avec l'outil Cobbler, en déployant une image LiveCD d'une distribution standard de Linux, Fedora18. Pour par la suite tester cela avec pfSense.

Pour m'aider j'ai lu le projet d'approfondissement de M. Chalut (cf Annexe 3.3.9) I)

3.3.2) Explications

Une image de Fedora18 est déployé via Cobbler. Les clients se connectent au switch, et obtiennent une adresse ip en 192.168.1, grâce au serveur dhcp de Cobbler.
Ensuite ils peuvent démarrer sur une image de Fedora18 via PXE.

3.3.3) Schéma

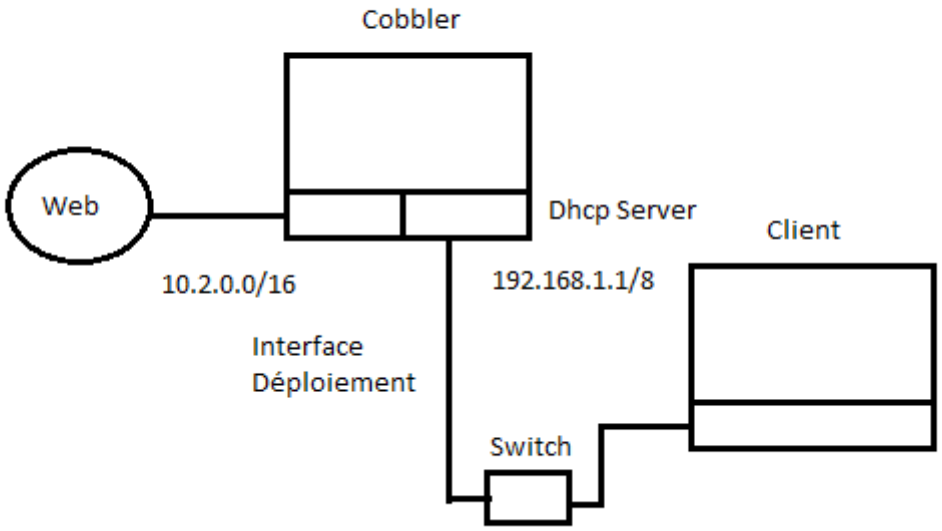


Illustration 9: Déploiement via Cobbler

3.3.4) Matériels

- Deux machines, une machine cliente, et une machine avec Cobbler installé faisant office de serveur
- La machine Cobbler possède deux interfaces physiques, une connecté au Web, 10.2.0.0/16. Et une autre 192.168.1.0/8 connecté à un réseau interne servant au déploiement
- Un switch

3.3.5) Réalisation

- Installation de Cobbler

```
[root@localhost Manager]# yum install cobbler
```

- Installation des prérequis (cf Annexe 3.3.9) II)
- Configuration de cobbler : Modification du fichier /etc/cobbler/*settings* et /etc/cobbler/dhcp.template (cf Annexe 3.3.9) III et IV).
- Démarrer les services :

```
[root@localhost Manager]# systemctl restart httpd.service
[root@localhost Manager]# systemctl restart cobblerd.service
```

- S'assurer qu'il soient bien lancer

```
[root@localhost Manager]# systemctl status httpd.service
[root@localhost Manager]# systemctl status cobblerd.service
```

- Exécuter la commande :

```
[root@localhost Manager]# cobbler check
```

- Corriger les points à modifier
- Exécuter la commande :

```
[root@localhost Manager]# cobbler sync
```

- Teste du bon fonctionnement du système avec une distribution LiveCD Fedora18 (cf Annexe 3.3.9) V)
- Téléchargement d'une image Fedora18.iso
- Installation de livecd-tools
- Exécution des commandes :

```
[root@localhost Manager]# livecd-iso-to-pxeboot NomDeImage.iso
```

- Cette commande génère deux fichiers que nous devons copier

```
[root@localhost Manager]# mkdir -p /srv/livecd
[root@localhost Manager]# cp /home/labotd/tftpboot /srv/livecd/vmlinuz0
[root@localhost Manager]#cp /home/labotd/tftpboot/initrd0.img  /srv/livecd/initrd0.img
```

- Ajout d'une distribution

```
[root@localhost Manager]# cobbler distro add –name=fedora18 –kernel=/srv/livecd/vmlinuz0 – initrd=/srv/livecd/initrd0.img –kopts='home/labotd/NomDeImage.iso rootfstype=iso9660 rootflags=loop!text!lang!ksdevice
```

- Ajout d'un profil

```
[root@localhost Manager]# cobbler profile add –name=fedora18 --distro=fedora18
```

3.3.6) Problèmes rencontrés

- Erreur : httpd does not appear to be running and proxying cobbler à l'exécution de la commande cobbler check
- Erreur : dhcpd -t failed à l'exécution de la commande cobbler sync
- Lors du démarrage de mon pc client celui si obtenait l'adresse ip 192.168.1.0, car j'avais typé la commande :

```
[root@localhost Manager]# cobbler system add –name=live_network –ip-address=192.168.1.0 --profile=fedora18
```

3.3.7) Solutions

- Modifier le fichier /etc/cobbler/*settings* à la ligne **server,** mettre comme paramètre : **localhost**
- Modifier le fichier /etc/cobbler/dhcp.template s'assurer que les adresses ips soient justes.
- Modifier le fichier /etc/cobbler/dhcp.template, après l'exécution de la commande remplacer le paramètre du champ *fixed-address*. Lui mettre une ip comprise dans l'intervalle définit.

3.3.8) Conclusion

Ce Tp m'a permis de me familiariser avec Cobbler, et de comprendre quels étaient les fichiers importants pour le démarrage PXE.
J'ai également appris les différences entre une version LiveCD, et un DVD pour une même distribution Linux.
Maintenant que j'ai appris à déployer un LiveCD via Cobbler je vais tester avec pfSense,vu que lui n'est que sous cette forme.

3.3.9) Annexes

I) http://www.tdeig.ch/kvm/Chalut_RPA.pdf
II) http://www.cobblerd.org/manuals/2.2.3/3/1_-_Prerequisites.html
III)
Fichier /etc/cobbler/settings

```
server: localhost
next_server: 192.168.1.1
```

Fichier /etc/cobbler/dhcp.template

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    option routers             192.168.1.1;
    option domain-name-servers 192.168.1.1;
    option subnet-mask         255.255.255.0;
```

```
range dynamic-bootp        192.168.1.50 192.168.1.70;
filename                                        "/pxelinux.0";
default-lease-time         21600;
max-lease-time             43200;
next-server                $next_server;
```

IV) http://www.cobblerd.org/manuals/2.2.3/2_-_Cobbler_Quickstart_Guide.html

V)https://github.com/cobbler/cobbler/wiki/How%20To%20Boot%20Live%20CDs

3.4) Deuxième Piste : Fichier ISO

3.4.1) Objectifs

L'outil Cobbler n'étant pas adapté à la situation, il m'a fallu trouver une alternative.
Le laboratoire possédant déjà un serveur PXE, l'objectif ici est de modifier les fichiers de configuration de celui-ci pour rajouter notre pfSense.

3.4.2)Recherches

- Sur le web : http://doc.pfsense.org/index.php/NetBoot_Embedded_%28soekris%29
- Sur le site officiel de pfSense

3.4.3)Réalisation

- Modification du fichier : 10.2.1.1/tftpboot/pxelinux.cfg/default, rajout de :

```
LABEL pfSense
 KERNEL memdisk
 INITRD pfsense/pfSense-kc.iso
 APPEND iso raw
```

3.4.4)Test

- Démarrer un pc sur le PXE et sélectionner pfSense

3.4.5) Problèmes rencontrés

- Trying to mount root from cd9660:/dev/iso9660/pfSense

Après plusieurs recherches j'en ai conclu que le fichier iso avait besoin du cd pour démarrer.

3.4.6)Conclusion

Cette solution n'est pas réalisable, on ne peut pas démarrer via PXE sur du FreeBSD comme sur une distribution standard, par exemple Fedora18.
Une alternative serait d'installer pfSense sur un pc, puis grâce à FOG de faire une image de celui-ci, qu'on déploierai.

3.5)Troisième piste : Utilisation de FOG

3.5.1) Objectifs

- Installer pfSense sur un poste client
- Installer FOG sur un poste serveur
- Effectuer une image du poste client grâce à FOG
- Mettre cette image sur le serveur PXE

3.5.2)FOG en bref

- FOG est un outil qui permet de cloner, faire des images d'un système.
- Pour plus d'informations (cf Annexes 3.5.5) I)

3.5.3)Réalisation

- Installation de pfSense sur une machine Windows. (cf Annexes 3.5.5) II)
- Installation de FOG sur une machine Fedora18. (cf Annexes 3.5.5) III)

3.5.4) Problèmes rencontrés

- Problème de configuration de FOG, quand un client démarre sur le pxe, il voit le pxe du laboratoire et non celui ou il y a FOG.
- Pour utiliser un serveur tftp et dhcp à part il faut configurer FOG sur deux postes, client et serveur, d'une autre manière en spécifiant d'utiliser le serveur dhcp de FOG sur le poste serveur. Problème, quand on démarre on ne peut pas trouver le serveur tftp, du coup on n'arrive pas à démarrer sur le PXE.
- En utilisant la virtualisation, cela ne fonctionne pas non plus, car pfSense ne peut pas démarrer par PXE.

3.5.5)Solution

- Modifier le code source de pfSense.

3.5.6)Annexes

I)Description de FOG : http://www.fogproject.org/?q=node/1
II)Laboratoire pfSense
III)Installation et configuration de FOG :

http://www.fogproject.org/wiki/index.php/Integrating_FOG_into_an_Existing_Network_in_non_intrusive_mode

http://www.fogproject.org/wiki/index.php?title=Modifying_existing_DHCP_server_to_work_with_FOG

4) Problèmes rencontrés

La documentation sur internet concernant SELinux étant redondante et en anglais, j'ai passé beaucoup de temps à trouver ce que je cherchais.

Pour une simple transition de domaine, il m'a fallu regarder dans les sources de modules existant pour voir comment cela était implémenté.
De plus lors de la troisième étude pratique j'ai fait des transitions de domaine vers le domaine unconfined à tort et à travers, ce qui fût une mauvaise idée, vu qu'au final j'ai dû remanier les règles pour que les privilèges attribués soient bien ceux qu'on attendait et pas plus.

En ce qui concerne pfSense, comme ce que je cherchais à faire avait été réalisé que par peu de personnes, qui plus est des experts, les recherches furent longues, de nombreuses pistes ont été suivies, et n'ont pas abouti.
Notamment la première piste avec l'outil Cobbler, qui en y repensant n'était pas adéquat, et qui aurait pu être évité si j'avais tenu compte du fait que pfSense est basé sur un système FreeBSD.
De même pour la seconde piste. En typant l'erreur sur internet on n'obtient pas de réponse précise sur ce qu'il faut faire.

Finalement en essayant d’intégrer mon serveur FOG directement dans le réseau du laboratoire j'ai fais face à des problèmes de configuration et de DHCP. Et au niveau virtualisation, mes compétences dans ce domaine n'étant pas poussé j'ai mis du temps à mettre en place l'architecture adéquate.

5) Conclusion

Les objectifs de ce travail ont été atteints. Je réalise maintenant pourquoi les utilisateurs lambda de Linux préfèrent désactiver SELinux.

En effet SELinux possède de nombreuses règles complexes, qui sont écrites par des experts, essayer de les réécrire soit même est quelque chose de difficile, mais cependant tout à fait faisable si on reste au niveau permissions sur des objets et transitions de domaines.

En ce qui me concerne je ne me vois pas désactiver SELinux. Ce dispositif est réellement un atout majeur pour la sécurité de son système. En effet imaginons qu'on télécharge une nouvelle version d'un logiciel, rien ne nous garantit que ce dit logiciel ai été codé correctement. Combien de temps faut il pour qu'un exploit zéro day soit trouvé? Personne ne le sait. Cependant grâce à SELinux on peut appliquer le principe du moindre privilège et ainsi être garantit que la personne qui nous piratera ne pourra pas faire ce que bon lui semble. En quelque sorte, on a une garantie que nos données personnelles ne soient pas à la portée de quiconque.
Au final sa complexité et son incroyable potentiel, auront égayé ma curiosité et mon intérêt pour ce système.

Pour la partie pfSense, j'ai apprécié de me retrouver face à un problème majeur. Je l'ai cependant résolu en recherchant et testant diverses alternatives et en fournissant un travail assez conséquent, qui m'a demandé beaucoup de temps.
J'ai également appris qu'un travail de recherche nécessitait d'être méthodique et logique. Il faut suivre un ordre précis, en commençant par un rassemblement de toutes les éventuelles solutions possibles, un tri de celles-ci et au final la mise en place de la bonne solution. Essayer de mettre en place la première solution venue va certainement se finir en échec.

6)Annexes

A.1)Prise de connaissance

Avant de commencer à faire des travaux pratiques, voici les étapes que j'ai suivi :

- Accomplissement du laboratoire SELinux
- Lecture en Intégralité du Mémoire de Khaled Basbous
- Lecture du Livre SELinux by Example

A.1.1)Accomplissement du laboratoire SELinux

Dans se laboratoire on y apprend les principaux mécanismes de sécurité offerts par SELinux.

A.1.2)Type Enforcement

- Découverte des commandes :

```
id- Z, ls- Z, ps -eZ
```

Pour relever les différents contextes de sécurité.

- Utilisation de la commande sesearch pour trouver des règles de sécurité, par exemple pour trouver la règle autorisant le domaine unconfined_t à exécuter un fichier de type passwd_exec_t, on utilise la commande :

```
sesearch -A -s unconfined_t -t passwd-exec_t
```

A.1.3) Gestion des utilisateurs

- Création d'utilisateurs, et attribution d'un rôle, voire un changement de rôle.
- Découverte d'une interface GUI, dans SELinux Management :

Avec une partie Identité SELinux qui nous permet de créer une nouvelle identité, ainsi que de lui assigner des rôles.

Une partie Correspondance d'utilisateurs, qui nous permet d'ajouter un nouvel utilisateur et de lui associer une identité SELinux.

A.1.4) Gestion des blocages

- Découverte de la commande :

```
ausearch -m AVC -ts today
```

Qui nous permet de voir les logs, et ainsi savoir quelles actions ont été bloquées et pourquoi, par exemple :

```
avc: denied { add_name } for pid=12891 comm="touch" name="testdeux.txt" scontexte=editeur_u:editeur_r:editeur_t:s0 tcontexte=unconfined_u:object_r:auteur_t:s0 tclass=dir
```

Ici on voit qu'un utilisateur faisant partit du domaine editeur_t à essayer de créer un fichier, commande touch, nommé testdeux.txt, dans un répertoire appartenant au type auteur_t. Et qu'il n'a pas réussi.

- Découverte de la commande audit2allow :

```
ausearch -m AVC -c touch -ts today | audit2allow -M module
```

Pour créer la règle manquante, qui à causé le blocage.

A.1.5) Role Based Access Control

- Modification du fichier /etc/sudoers via la commande visudo
- Compréhension de la syntaxe de ce fichier, pour nous permettre d'y écrire dedans.
- Par exemple, les deux lignes suivantes :

```
Kevin ALL=(ALL) ROLE=webadm_r TYPE=webadm_t /bin/service
Kevin ALL=(Kevin) ROLE=webadm_r TYPE=webadm_t ALL
```

Permette à Kevin de lancer un service en tant que root mais en transitant vers le rôle webadm_r et le domaine webadm_t, et d'autoriser Kevin à pouvoir lancer toutes les commandes autorisées pour le rôle webdam_r.

A.1.5)Lecture en Intégralité du Mémoire de Khaled Basbous

A.1.5.1)Liens
www.tdeig.ch/kvm/Basbous_RTB.pdf

A.1.5.2)Utilités

Ce mémoire, permet d'approfondir nos connaissances sur SELinux.
On y découvre :

- Une introduction sur SELinux, c'est à dire les différents mécanismes de contrôle d'accès, que ce soit le Discretionary Access Control, le Mandatory Access Control, ou le Role Based Access Control.
- Des notions avancés sur le Type Enforcement, et le Role Based Access.

Ce mémoire est à la fois théorique et pratique.

A.1.6)Laboratoire Protection_contre_une_attaque_0Day

Dans ce Laboratoire on effectue les étapes suivantes :

- Récupérer les sources d'un serveur ftp faillible et l'installer sur un poste Fedora avec une configuration par défaut
- Récupérer l'utilitaire metasploit et l'utiliser pour obtenir un accès sur la machine Fedora
- Observer que malgré la présence de SELinux mal configuré, ou l'absence de celui-ci, le pirate obtient un contrôle quasi total du système.
- Changer le contexte d'exécution du serveur ftp, dans un domaine plus confiné
- Répéter l'opération 3- et se rendre compte que le pirate ne peut plus faire grand chose, malgré l'obtention d'un shell distant.

A.1.7)Lecture du Livre SELinux by Example

A.1.7.1)Chapitres

- Chapitre 2 Concepts
- Chapitre 5 Type Enforcement
- Chapitre 6 Roles and Users
- Chapitre 13 Managing an SELinux System
- Chapitre 14 Writing Policy Modules

A.2) Tutoriel SLIDE

Une fois SLIDE installé, le lancer avec la commande :

```
[root@localhost test]# eclipse
```

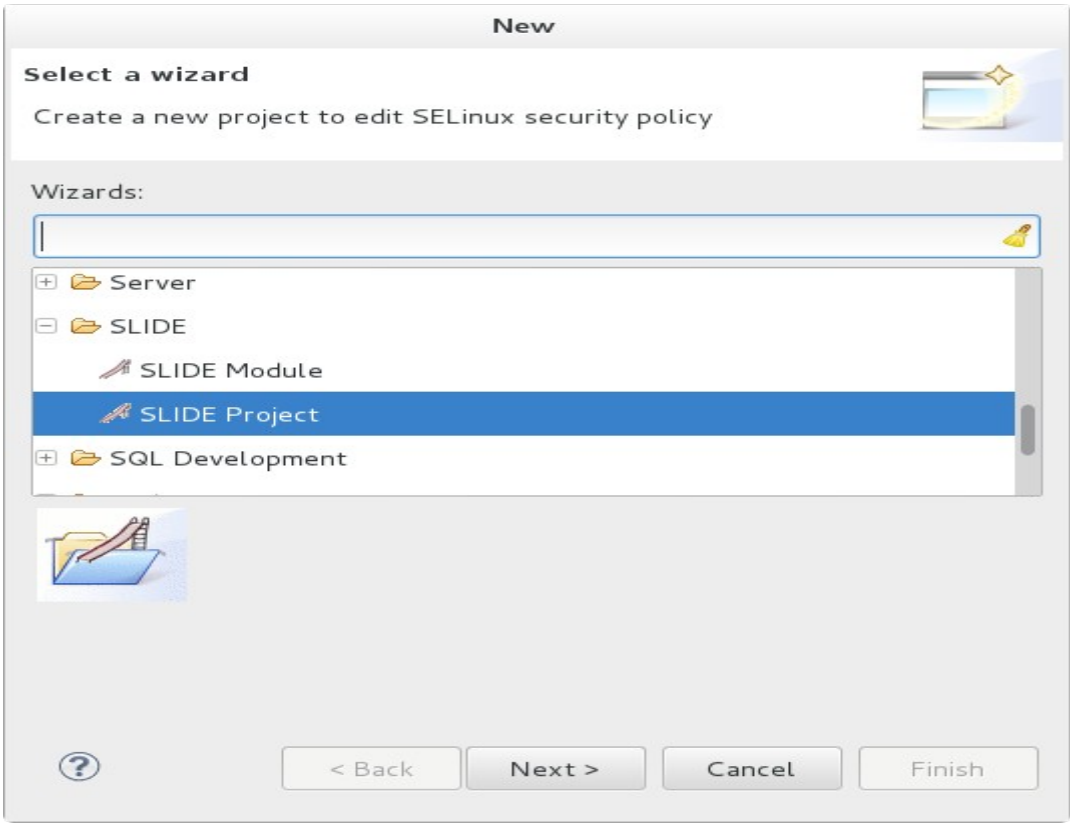


Illustration 10: Slide Project

- Aller dans : File-> New-> Other
- Choissir SLIDE Project -> Next
- Nommer le projet
- Choisir sa location, laisser par défaut en général
- Choisir le type du projet : Police Module Référence, si on veut éditer un module. Full Reference Policy Project, si on veut éditer une police complète.
- Next, laisser la police par défaut puis Finish.
- Clique droit sur le project créer -> add -> new -> SLIDE Module

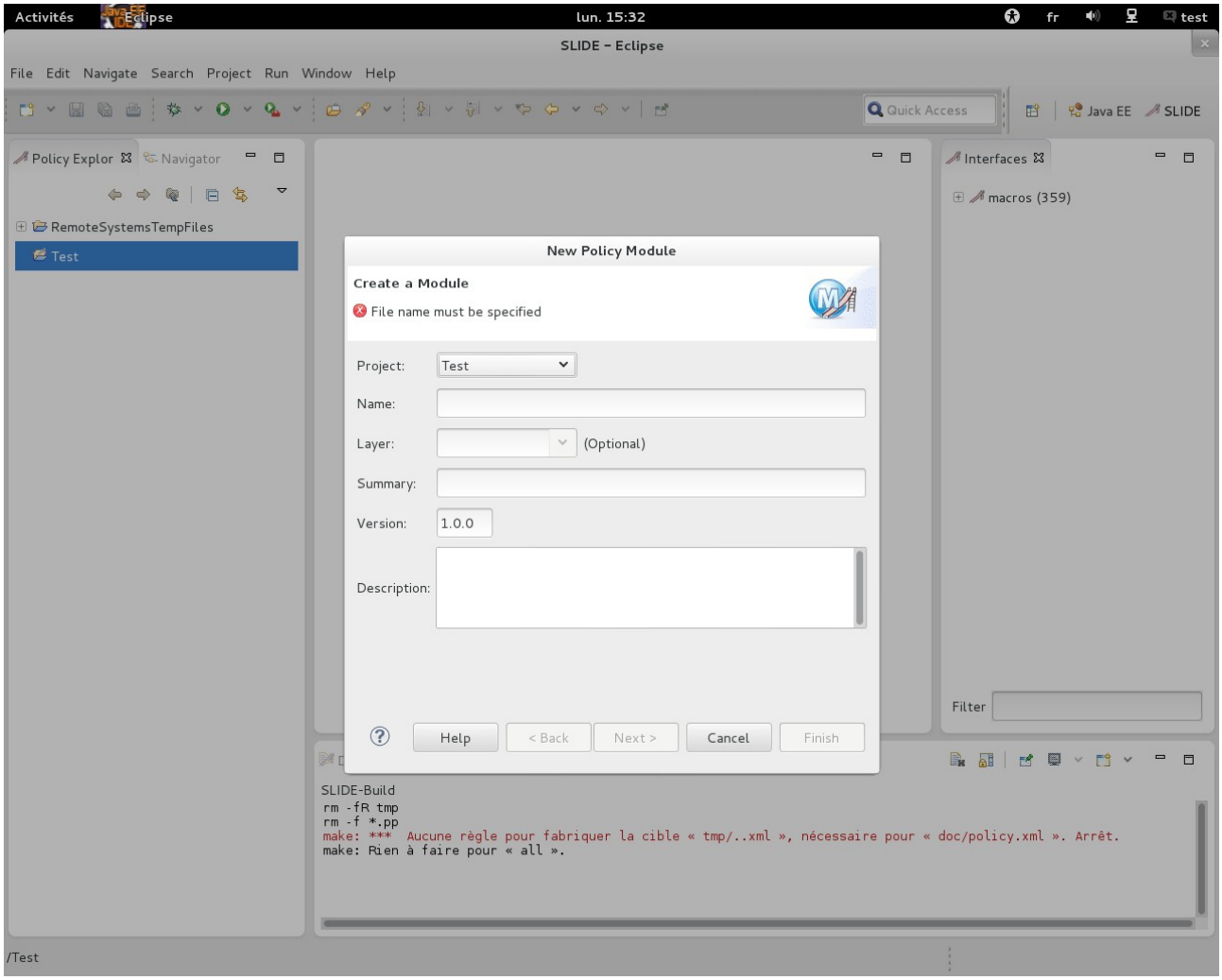


Illustration 11: Slide Module

- Spécifier un nom de projet ainsi qu'un résumé -> Finish
- Voici à quoi ressemble notre projet.

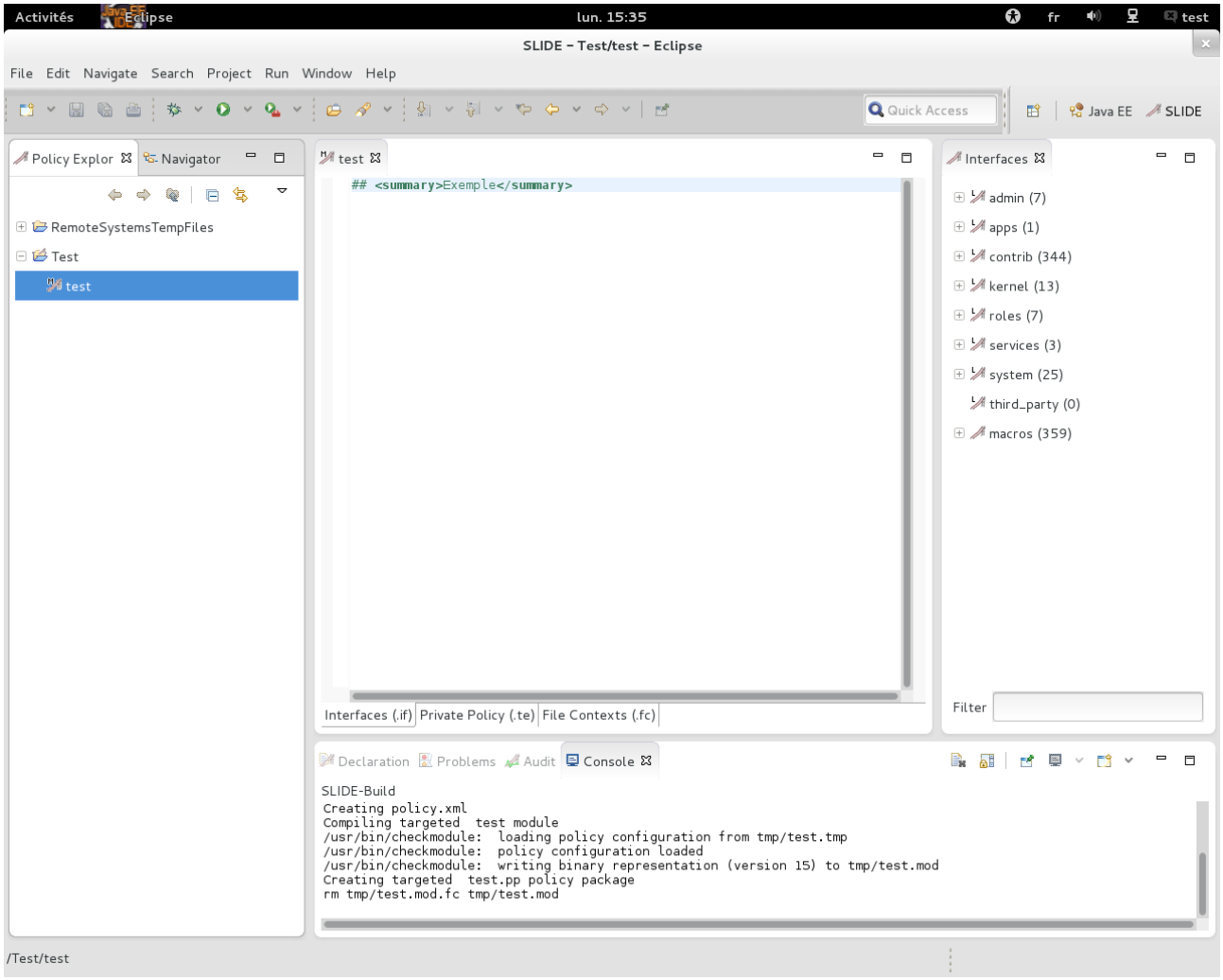


Illustration 12: Interface du projet

- Deux parties nous intéressent:

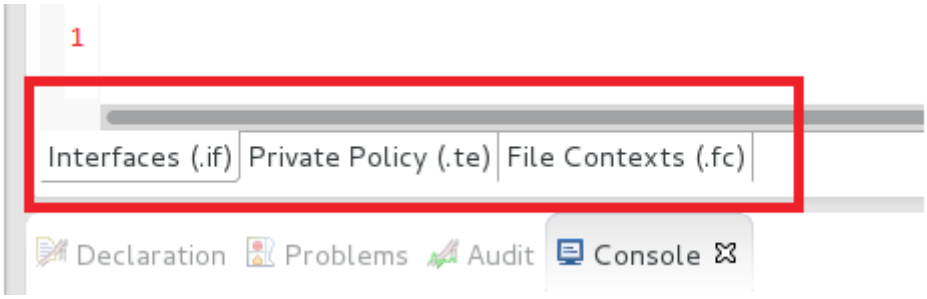


Illustration 13: Fichiers du projet

Ici on voit nos trois fichiers que SLIDE nous a créé.
On peut naviguer de l'un à l'autre en cliquant dessus. On pourra rajouter nos interfaces soit dans le fichier .if soit dans le fichier .te.

Une fois les macros pour la transition de domaine repérées je les ai rajoutés dans le fichier .te.

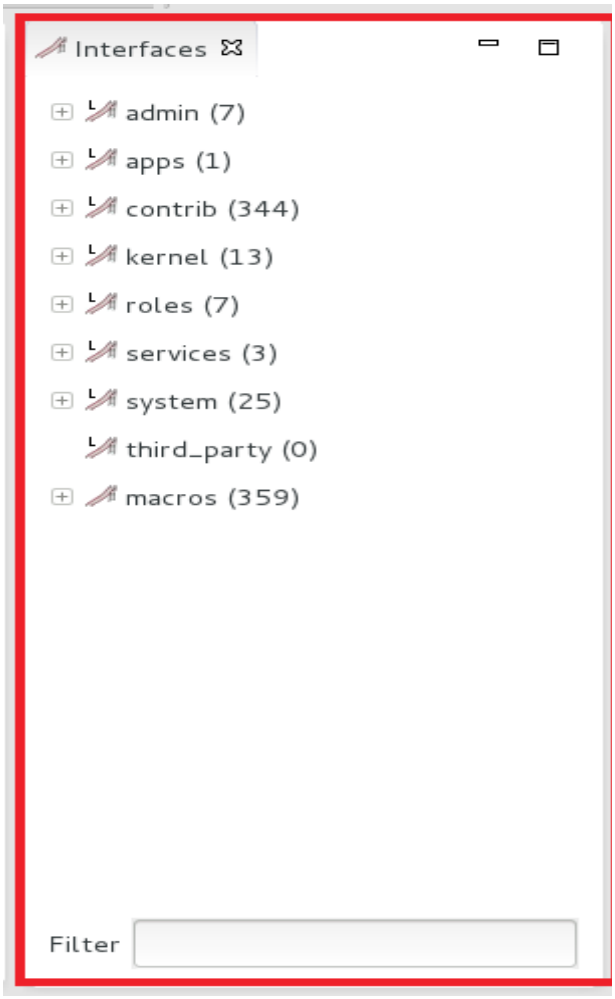


Illustration 14: Interfaces

Et ici on voit les interfaces qu'on va pouvoir utiliser, ainsi qu'une barre de recherche pour appliquer un filtre. Par exemple si on veut utiliser une interface pour gérer un serveur apache, on va filtrer sur le critère webadm.

- Pour rajouter une interface, une fois celle-ci trouvé, faire clique droit dessus -> add to current module.
- Une fois le projet terminé, cliquer sur Run -> Run configurations

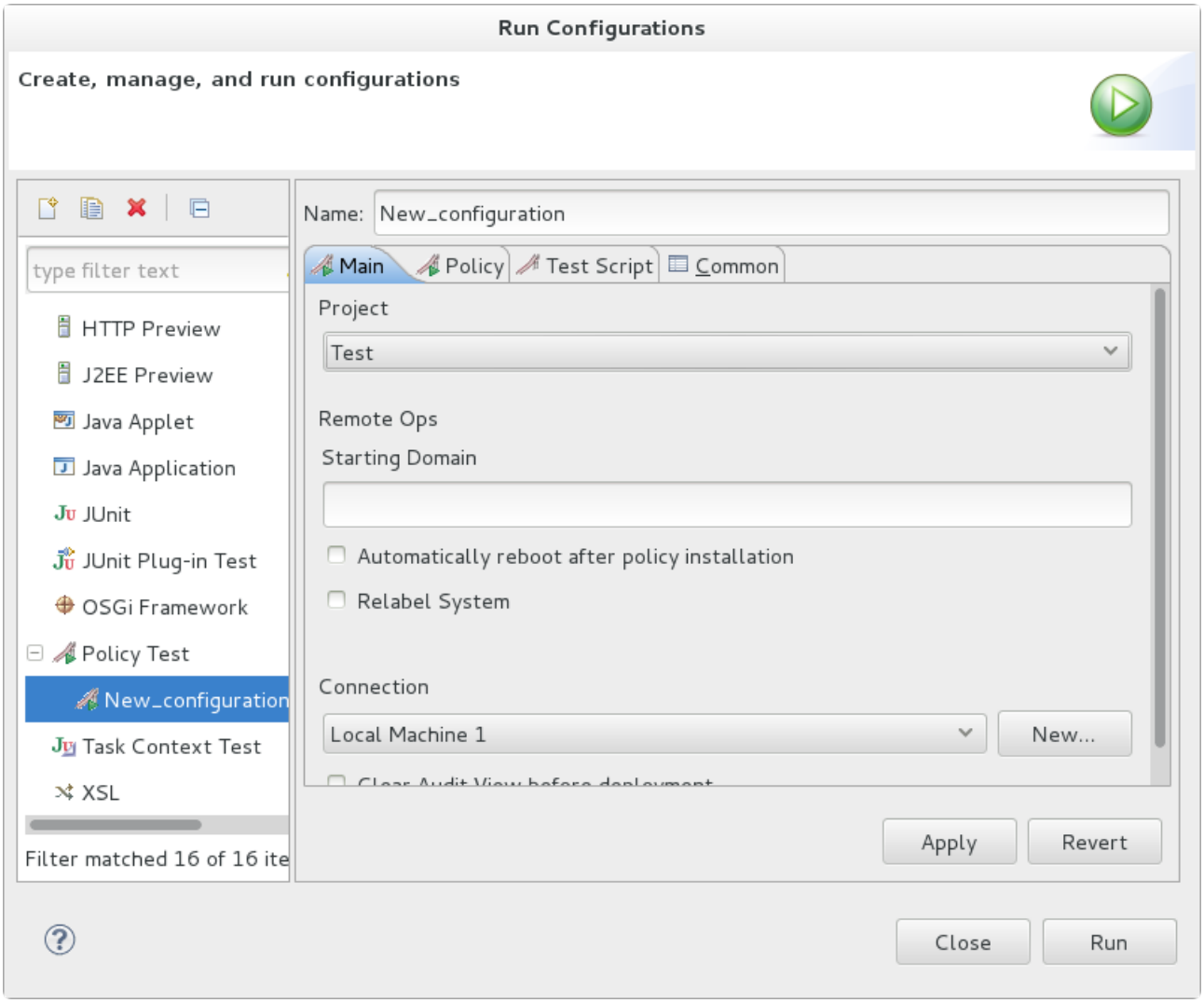


Illustration 15: Configuration du démarrage

- Aller dans Policy Test
- New_configurations
- Dans l'onglet Main, sélectionner le projet à lancer et choisir une connexion.
- Pour cela, cliquer sur New -> add -> ok
- Dans l'onglet Policy

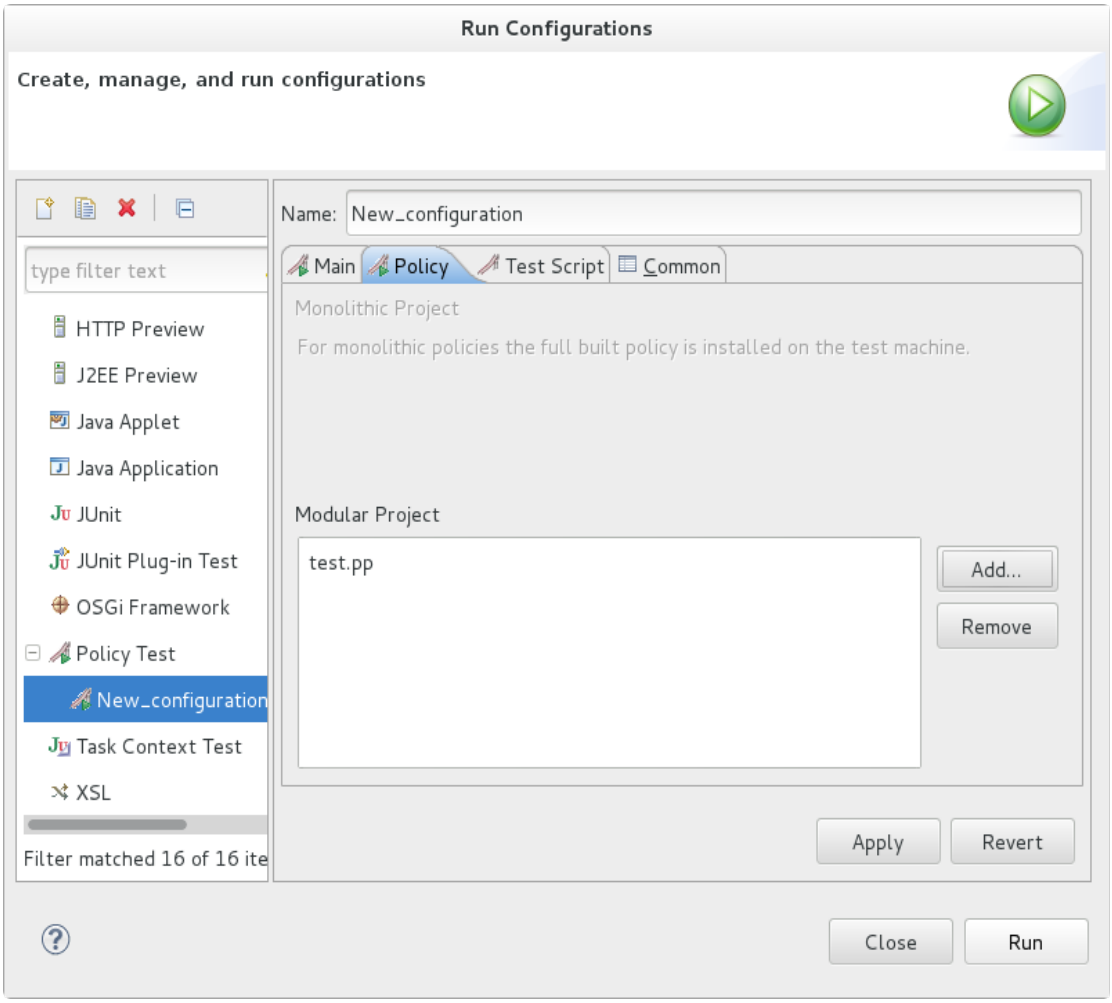


Illustration 16: Slide- nouvelle configuration

- Cliquer sur Add
- Ajouter notre module
- Ouvrir un nouveau Terminal
- Installer slideRemote : http://oss.tresys.com/projects/slide/wiki/download
- Lancer la commande slideRemote

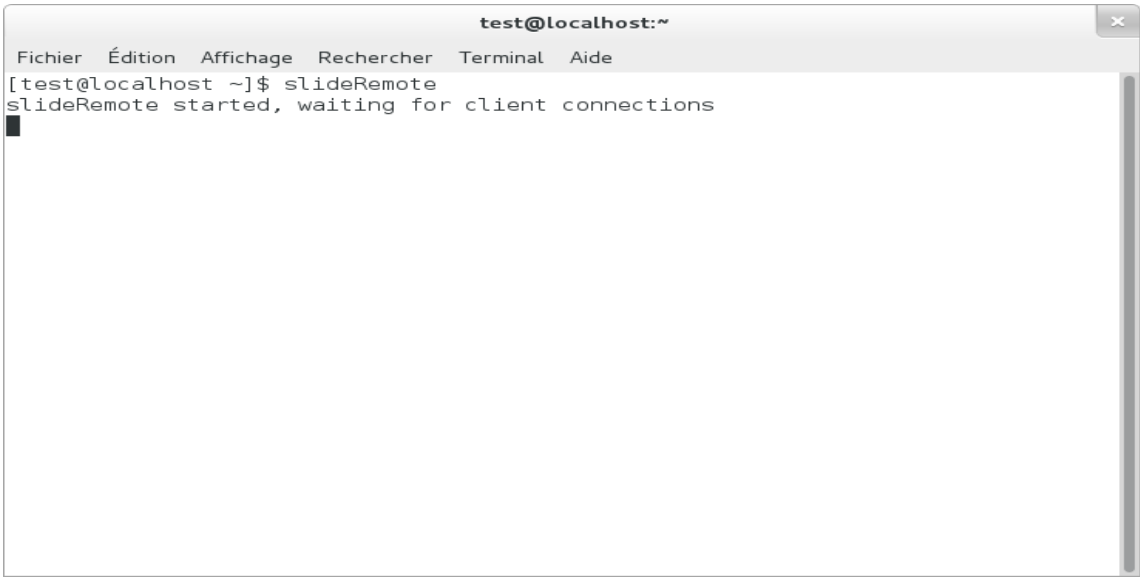


Illustration 17: Slide Remote

- Retourner sur eclipse
- Cliquer sur Run

Si notre module est correct il devrait être correctement implémenté dans le système.

A noter que SLIDE Remote est un processus démon que SLIDE utilise pour installer une police et pour retrouver les messages d'audits.
Il n'est pas nécessaire. On peut s'en passer en allant directement dans le répertoire où notre fichier .pp est crée, et utiliser la commande :

```
[root@localhost test]# semodule -i NomDuFichier.pp
```