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

Rue de la Prairie 4
CH-1202 Genève

Tél. +41 (0)22 546 24 00
Fax +41 (0)22 546 24 10

hepia@hesge.ch
www.hesge.ch/hepia

Security Enhanced for Linux

Travail de Bachelor

Session 2012

-

Professeur responsable : LITZISTORF Gérald

Diplômant : BASBOUS Khaled

Filière Télécommunications

Laboratoire de transmission de données

Genève, le 22 juin 2012

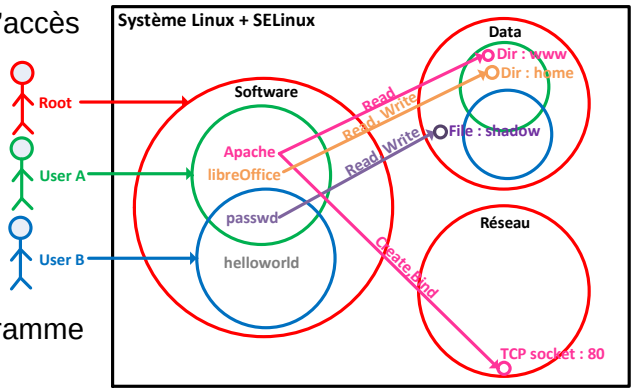
Hes·SO GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

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

Printemps 2012
Session de bachelor

Résumé: Security Enhanced for Linux (SELinux)

SELinux implémente plusieurs modèles de contrôle d'accès (MAC par TE, RBAC, MCS/MLS) qui améliorent grandement la sécurité des systèmes d'exploitation Linux. Le but de ce travail est, dans un premier temps, d'étudier d'un point de vue théorique le fondement des modèles présents dans SELinux. Puis, dans un second temps, d'analyser les règles fournies dans Fedora Core 16 (FC16) et de sélectionner les outils de gestion adéquats afin de faciliter l'administration du système. Ce travail sera poursuivi dans un programme exploratoire financé par la HES-SO.



L'étude s'est déroulée en huit semaines et a été réalisé en cinq étapes.

1. Lecture de l'excellent livre « SELinux by Example » qui traite de l'écriture de règles et qui décrit les différents mécanismes de contrôle.
2. Étude de nombreux documents et présentations trouvés sur le web.
3. Prise en main de Fedora 16, des outils d'analyse de règles et d'administration.
4. Définition de scénarios qui démontrent le fonctionnement du Type Enforcement (TE) et de Role Base Access Control (RBAC).
5. Réalisation d'un scénario où un système, sans et avec le renforcement SELinux, est attaqué en utilisant une porte dérobée.

J'ai choisi de commencer par étudier le modèle TE car c'est le modèle fondamental à la solution SELinux. Après avoir compris le concept de TE, je me suis attaqué à l'étude de la syntaxe et à la signification des règles TE. La compréhension des règles m'a été grandement bénéfique et m'a permis de simplifier mon travail lors de l'analyse des règles de FC16. L'avantage d'avoir commencé par TE est qu'il permet de comprendre aisément le modèle RBAC car il est implémenté avec un petit ajout à TE.

J'ai beaucoup utilisé l'excellent logiciel GUI d'analyse de règles APOL et ainsi que la commande indispensable d'administration semanage. Ces deux outils importants ont été utilisés pour :

- Démontrer TE, RBAC et MCS (Multi-Category Security).
- Expliquer la démarche que doit avoir un administrateur système lorsqu'il est confronté à une situation de blocage pour comprendre sa provenance et la résoudre.
- Restreindre les droits d'un utilisateur.

J'ai expliqué comment générer des règles à l'aide de l'outil polgengui pour sécuriser un logiciel auquel il n'existe pas de règles écrites dans la communauté, ainsi que la limitation de cette méthode.

Vers la fin du travail, j'ai récupéré les sources d'un serveur FTP vulnérable à un exploit Metasploit et décrit l'installation pour que les règles SELinux contenues dans F16 soient appliquées au serveur FTP. Au final j'ai récupéré un shell distant grâce à BackTrack 5 en exploitant cette faille et démontré l'intérêt de SELinux pour contrer cette attaque.

Diplômant :
M. BASBOUS KHALED
Classe : TE3
Filière d'études : Télécommunications
Ingénierie des Technologies de l'Information

Timbre de la direction

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

Printemps 2012
Session de bachelor

SECURITE DES SYSTEMES D'INFORMATION

SELINUX

Descriptif :

SELinux implémente divers modèles de sécurité (MAC, RBAC, Bell-LaPadulla) qui doivent être maîtrisés pour améliorer le niveau de sécurité des serveurs actuels. Trop de personnes estiment que SELinux est trop complexe par manque de connaissance. Désactiver SELinux aujourd'hui sur un serveur revient à ne pas être capable de régler la hauteur de son appuie-tête en voiture.
Les premiers travaux réalisés par les étudiants Gattuso - Roschi montrent un indéniable gain sécuritaire et des difficultés à en maîtriser tous les aspects. Chaque paquetage d'une distribution Linux est livré avec des règles SELinux qui peuvent parfois poser problème ; voir l'illustration par Fedora avec le serveur Apache
→ http://doc.fedora-fr.org/wiki/Installation_et_configuration_d%27Apache#SELinux_et_apache

Travail demandé :

1. Introduction
   Parcourir le livre SELinux by Example http://flylib.com/books/en/2.803.1.1/1/ pour identifier l'éventail des possibilités offertes ainsi que les outils proposés
   Etude du rapport http://www.tdeig.ch/Soir3/Rapport_Final/Roschi_Gattuso_SELinux.pdf
   Lien utile : https://fedoraproject.org/wiki/SELinux
   Estimation = 2 semaine

2. Comprendre le fonctionnement de SELinux avec les règles par défaut de Fedora 16
   Identifier les outils utiles
   Etudier quelques cas simples de règles fournies dans des packages
   Estimation = 2 semaines

3. Expliquer l'architecture de SELinux
   Utiliser la présentation de *School of Computer Science and Engineering, Seoul National University*
   - http://ssrnet.snu.ac.kr/course/sec2007-1/note/SELinux-2007.ppt
   - http://www.tdeig.ch/TB_2012/SELinux/SELinux-2007.ppt copie locale

   Voir les slides 29-34 qui illustrent l'évolution de l'architecture SELinux depuis le projet Flask (slide 30) jusqu'à son implémentation.
   Estimation = 2 semaines

4. Configurer une sécurité basée sur le modèle Bell-LaPadulla : *No information flow from 'high' security levels down to 'low' security level (confidentiality)*
   Estimation = 1 semaine

5. Thème à choix si le temps le permet

   Sous réserve de modification en cours du travail de Bachelor

Candidat :
M. BASBOUS KHALED
Filière d'études : --
Domaine de formation --

Professeur(s) responsable(s) :
Litzistorf Gérald

En collaboration avec : Nom de l'entreprise
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

PREAMBULE

En préambule de ce mémoire, je souhaiterais remercier sincèrement Monsieur Litzistorf pour ces valeureux conseils, qui s'est toujours montré disponible et à l'écoute tout au long de la réalisation de ce mémoire.

Je remercie également mes camarades pour leurs encouragements et pour les merveilleux moments passés ensemble.

Mes derniers remerciements iront vers ma famille, et surtout mes parents, qui m'auront permis de poursuivre mes études jusqu'à aujourd'hui.

Merci à toutes et à tous.

CONVENTION UTILISEES

Dans ce rapport le shell administrateur sera représenté par le caractère # en début de ligne :

```
1    #[root@SELinux ~]
```

Le shell utilisateur sera représenté par le caractère $ en début de ligne :

```
2    $[user@SELinux ~]
```

La réponse retournée par une commande CLI dans le shell est noté comme ceci :

```
3    Texte retournée dans le shell
```

Les règles SELinux sont reportées de cette façon :

```
allow source_type target_type : file { read write execute } ;
```

L'ENVIRONNEMENT

J'ai installé la distribution Fedora 16 en 64 bits sur une machine physique du laboratoire.

Nom de la machine : SELinux.HEPIA

Utilisateur : root **Mot de passe :** toortoor

Utilisateur : admin **Mot de passe :** nimdanimda

Utilisateur : test **Mot de passe :** tsettset

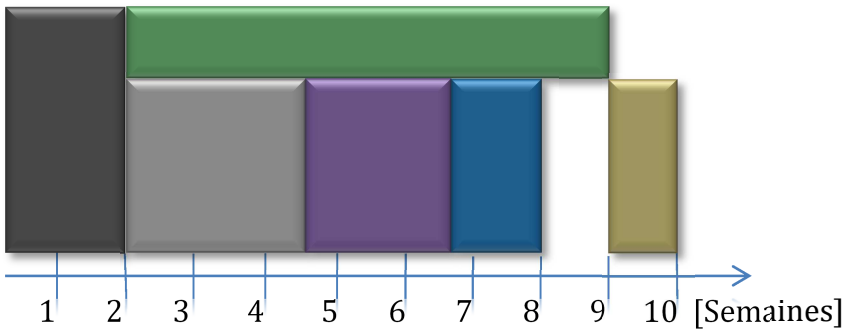
- Mettre à jour tous les paquets installés sur Fedora :

```
1    #[root@SELinux ~] yum -y update
```

- Installer les paquetages d'utilitaire utilisé dans cette étude, voir l'annexe A.3 pour plus d'informations sur ces paquetages :

```
1    #[root@SELinux ~] yum -y policycoreutils-gui
2    #[root@SELinux ~] yum -y setools
```

REPARTITION DU TEMPS DE TRAVAIL



- Etude du livre SELinux by Example
- Etude de la documentation du web
- Définition et réalisations des scénarios TE, RBAC, MCS
- Définition et réalisation du scénario hacking serveur FTP et génération de module
- Rédaction mémoire
- Préparation pour la défense

TABLE DES MATIERES

LISTE DES FIGURES

1 INTRODUCTION

1.1 ENJEUX DE LA SECURITE DES SYSTEMES D'EXPLOITATION

Les systèmes informatiques sont exposés à des menaces susceptibles d'altérer ou de détruire l'information, de la révéler à des tiers qui ne doivent pas en avoir accès. Si un système n'est pas sécurisé, les applications les plus robustes seront fragilisées et deviendront vulnérables une fois déployées sur celui-ci.

Depuis les années 2000, l'accès rapide aux informations à travers l'Internet et les partages de données ont augmenté de façon considérable. L'informatisation des entreprises a conduit au développement de logiciels internes et à l'utilisation de logiciels développés en externes.

L'indisponibilité des données, les incidents, les négligences et la malveillance peuvent conduire à des pertes monétaires importantes. La confidentialité et l'intégrité des données sensibles sont très souvent vitales pour l'image et la compétitivité de l'entreprise. Les menaces sont nombreuses et peuvent venir autant de l'intérieur que de l'extérieur de l'entreprise.

1.2 QUI NOUS VEUT DU MAL ?

"Si tu ignores à la fois ton ennemi et toi-même, tu ne compteras tes combats que par tes défaites. "

(Sun Tzu, général chinois, 6ème siècle av. J.-C.)

Ce travail se situe du côté de la défense et a comme objectif la sécurisation du système pour contrer les attaques d'un adversaire. Un adversaire est une entité qui essaie de contourner les infrastructures de sécurité. Les attaquants peuvent être des concurrents, des groupes malicieux, des hackers qui tentent de comprendre le système, des fanfarons ou des curieux. Ils ont plusieurs moyens d'attaque, cependant l'exploitation des failles des applications est un des plus puissants moyens.

1.3 LES PORTES D'ENTREES DU SYSTEME

"The necessity of operating system security to overall system security is undeniable … If it fails to meet this responsibility, system-wide vulnerabilities will result."[1]

Les exécutables sont les points d'entrées aux données du système. Ils sont développés :

- En interne
- En externe sur demande
- En externe destiné à la large distribution (gratuit ou payant)
- Par la communauté du libre

L'évolution incessante des applications augmente le risque d'ajout de failles au sein de ceux-ci. Beaucoup d'applications sont imparfaites et contiennent des erreurs de conception ou de programmation. Il existe des applications qui sont malicieuses, développées par nos adversaires, nommées « cheval de Troie ». Un cheval de Troie est programmé pour accomplir une tâche attendu afin de paraître légitime mais en arrière-plan

[1] P. Loscocco, S. Smalley, P. Muckelbauer, R. Taylor, S. Turner, J. Farrell. The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. In Proceedings of the 21st National Information Systems Security Conference, pp. 303314, October 1998, disponible http://cryptome.org/jya/paperF1.htm

il contient une fonction cachée qui porte atteinte au système. Par exemple, il capture en secret des données et les transmets sur le réseau.

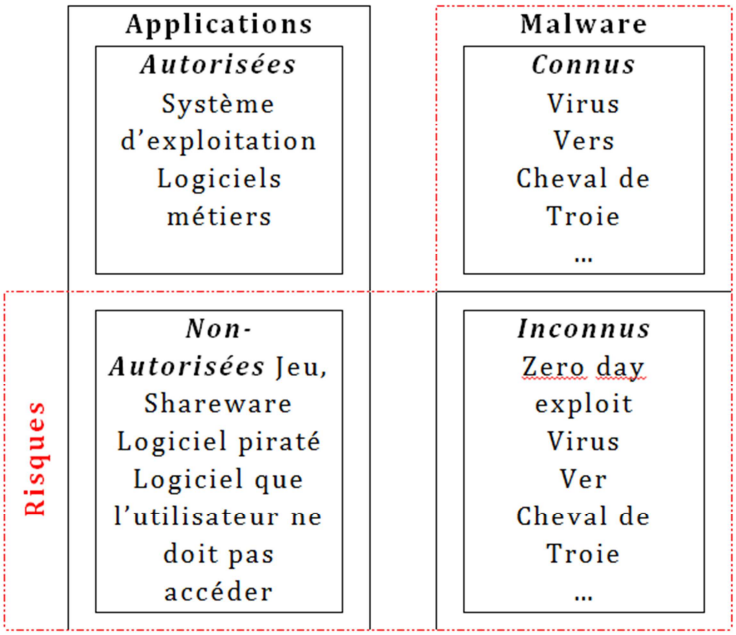


FIGURE 1: LES MENACES QUI PESENT SUR LES SYSTEMES INFORMATIQUE[2]

Les risques sur un système proviennent des logiciels touchés par des malwares qui circulent sur internet, des périphériques amovibles et des utilisateurs.

1.4 VERIFIABILITE DES APPLICATIONS

La création d'une application comporte des risques à tous les niveaux. Prenons l'exemple de la compilation d'un logiciel. Qu'est ce qui nous garantit que le compilateur n'insère pas une fonction non désirée au code source lors de la compilation ? À qui et à quoi pouvons-nous faire confiance ? Comment contrôler le contenu et la fonction de nos programmes ?

- Vérifier le code source du compilateur, de l'application générée
- Vérifier le binaire de l'application générée
- Demander à la communauté si un logiciel est sûr

Malgré toutes nos vérifications, notre système pourrait être touché par un virus et être reconfiguré. D'ailleurs, la vérification de toutes les applications utilisées d'un système est difficilement envisageable. Dès lors, une autre méthode pour sécuriser le système doit être utilisée. Brider les capacités des applications est une solution plus réaliste.

[2] Source de la figure l'illustration SecureWave

2 ANALYSE

2.1 OBJECTIFS DE SELINUX

"SELinux … represents the culmination of nearly 40 years of operating system security research"[3]

Security Enhanced for Linux est un module de sécurité supporté dans plusieurs distributions Linux. De plus en plus, il est activé par défaut dans les distributions. **Fedora Core** a été une des plateformes centrales dans laquelle la communauté a pu tester et intégrer ce module. Son champ d'action est la protection de l'**intégrité** et de la **confidentialité** d'un système d'après des **règles** définies.
Grâce à son utilisation, un système d'exploitation est moins exposé aux **exploits Zero-day**, aux attaques qui exploitent les vulnérabilités des logiciels et aux chevaux de Troie.
Malheureusement, SELinux est très souvent désactivé par les administrateurs système par manque de connaissance. La première suggestion sur le moteur de recherche Google lorsque nous tapons « SELinux » est « SELinux disable ». Nous verrons que SELinux est riche et

3 SELinux by Example p.4

puissant par ces fonctionnalités. Toutes les distributions Linux qui l'implémentent par défaut, par exemple Fedora 16, sont livrées avec un certain nombre de règles destinées au confinement du système. L'administrateur système a la possibilité de personnaliser ces règles pour mieux correspondre aux besoins.

2.2 MECANISMES DE CONTROLE D'ACCES, EN GENERAL

SELinux ajoute des méthodes de contrôle d'accès telles que :

- le contrôle d'accès obligatoire flexible, **Mandatory Access Control (MAC)**, nommé **type enforcement (TE)**
- le contrôle d'accès basé sur le rôle nommé **Role Base Access Control (RBAC)**
- le mécanisme de sécurisation multi-niveaux, **Multi-Level Security (MLS)**, mais il reste optionnel au déploiement.

2.2.1 DISCRETIONARY ACCESS CONTROL

La plupart des systèmes d'exploitation implémente un contrôle d'accès pour déterminer quel utilisateur a droit d'accéder à quelle ressource, à quel fichier. Linux utilise une méthode d'accès discrétionnaire, en anglais **Discretionary Access Control (DAC)** communément connu sous le nom de « file permission ». Cette méthode permet au propriétaire d'un fichier d'indiquer quel autre utilisateur a le droit d'accéder à son fichier. Dans Linux, un fichier a un propriétaire, un groupe et trois ensembles de permissions nommés les bits d'accès. Trois types d'accès sont possibles dans un ensemble de permissions :

1. le droit de lecture [r]
2. le droit d'écriture [w]
3. le droit d'exécution [x]

Un ensemble de permissions est alloué au propriétaire du fichier, un autre ensemble de permissions est alloué au groupe du propriétaire du fichier et un dernier ensemble de permissions est alloué aux autres utilisateurs du système.

```
1    $[user@SELinux ~]$ ls -l
```

Ensembles de permissions

```
1    -rw-r--r-- 1 user groupUser 14814 2006-07-26 13:25 test.txt
```

Propriétaire Groupe Autres Nom fichier

Dans l'exemple ci-dessus, le fichier test.txt peut être lu et écrit par son propriétaire c'est-à-dire user. Nous voyons ceci grâce à l'ensemble de

permission en rouge [rw-]. Ce fichier peut être lu [r--] par son groupe gourpUser et par tout le reste des utilisateurs du système.
Il existe deux niveaux de privilèges :

1. Root (accès total à l'ensemble des fichiers du système)
2. User (accès total à l'ensemble des fichiers de l'utilisateur en question uniquement)

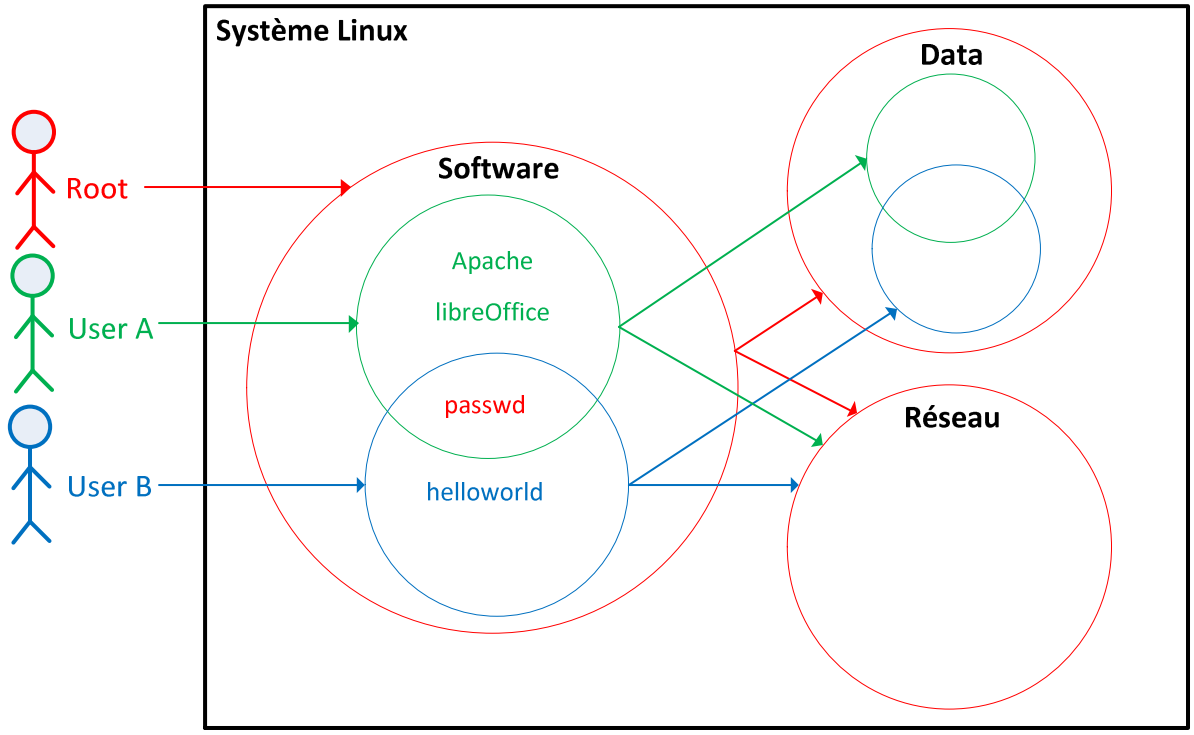


FIGURE 2: DROITS UTILISATEURS DANS LINUX AVEC LE MODELE STANDARD DAC

Dans la figure ci-dessus, l'utilisateur « User A » a accès à un ensemble d'applications. Ces applications ont accès à toutes les données de User A dû à l'**héritage des droits** de l'utilisateur et au réseau. Le super utilisateur « root » a accès à tout le système.
DAC possède une grande faiblesse car un programme hérite des droits (user rights) de l'utilisateur qu'il a lancé. Un programme vérolé peut ainsi créer, modifier ou supprimer des données en profitant de cet héritage de droit. L'utilisateur est obligé d'avoir une **confiance aveugle** dans le logiciel qu'il utilise.

2.2.2 MANDATORY ACCESS CONTROL

Dans un environnement renforcé par MAC, l'accès à n'importe quelles ressources est strictement contrôlé par le système d'exploitation selon les règles gérées ou écrites par l'administrateur système. Par exemple, le propriétaire d'un fichier n'a plus le droit d'exécuter un fichier qu'il possède sans que l'administrateur lui ait fourni ce droit auparavant. Pour améliorer la sécurité d'un système d'exploitation, il vaut mieux utiliser le **principe du moindre privilège**. Le principe du moindre privilège permet

de restreindre l'accès d'un programme au strict minimum pour accomplir sa fonction. Par exemple, le programme qui permet de modifier la configuration IP du système a uniquement accès aux fichiers de configuration du système concerné et à rien d'autre. Dans le cas où nous sommes en présence d'un logiciel serveur qui écoute sur un port, si ce programme contient une faille qui permet à un attaquant d'ouvrir une porte dérobée sur le système, le hacker aura accès aux ressources du système. Si le moindre privilège est de mise, le logiciel exploité pourra accéder uniquement aux ressources utiles au fonctionnement du programme exploité et à aucun autre élément du système. Le champ d'action de l'attaquant se retrouve ainsi fortement réduit.

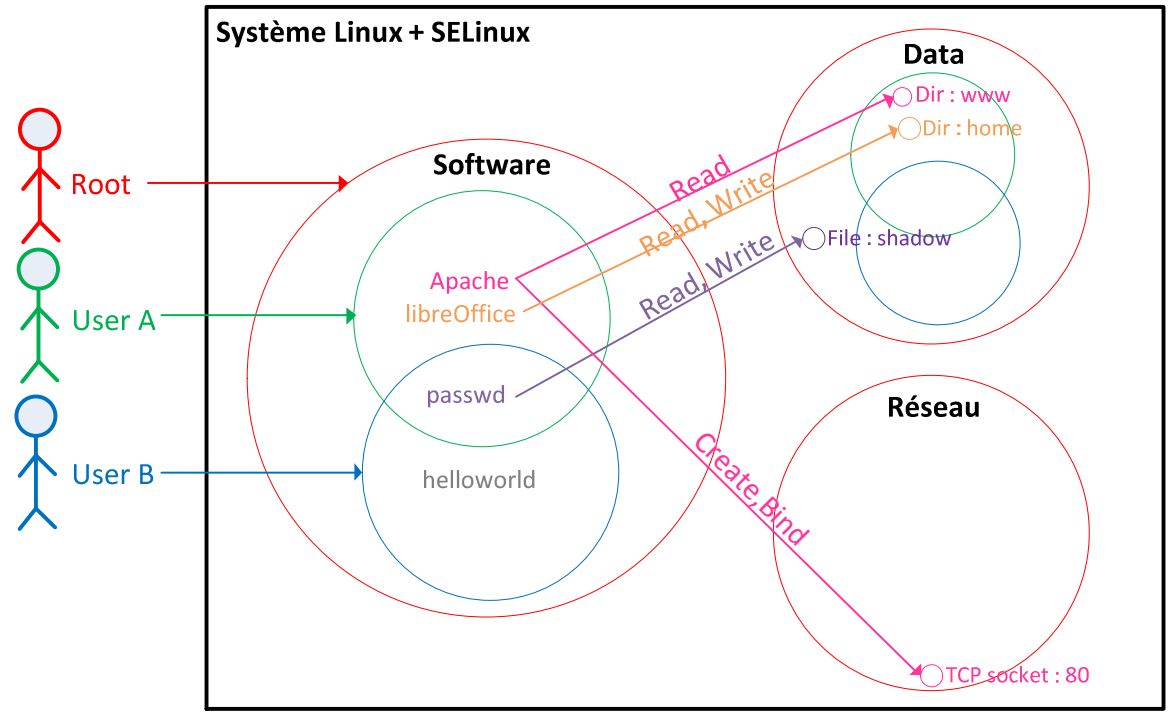


FIGURE 3: SYSTEME LINUX PROTEGE PAR LE MODELE MAC DE SELINUX

Dans la figure ci-dessus, nous avons de nouveau l'utilisateur « User A » qui a accès à certaines applications. Cette fois-ci, les applications ont été restreintes par un mécanisme MAC. Celles-ci n'ont plus accès à toutes les données de l'utilisateur et n'ont plus un accès libre au réseau.

2.2.3 ROLE BASED ACCESS CONTROL

Avec un contrôle d'accès basé sur le rôle, les permissions sont données à un utilisateur selon un rôle qu'il lui est associé. Un rôle représente un groupe d'utilisateurs et en même temps un ensemble de permissions. Les groupes d'utilisateurs sont construits selon leurs fonctions dans l'entreprise. Un comptable n'a pas besoin pour remplir sa fonction d'avoir accès aux fichiers où à la base de données qui concerne la conception d'un produit de l'entreprise. Un rôle comptable sera créé et regroupera tous

les employés de l'entreprise qui ont ce profil. Un utilisateur peut lui être associé plusieurs rôles. Par exemple, le directeur de l'entreprise peut avoir besoin de plusieurs rôles pour contrôler les activités de son entreprise.

2.2.4 MULTI-LEVEL SECURITY

Le but du mécanisme Multi-Level Security (MLS) est de forcer les processus du système à opérer dans un niveau de sécurité donné afin d'offrir la confidentialité. Une règle issue du modèle **Bell-La Padula** régie les communications entre les différents niveaux :

« No read up and no write down »

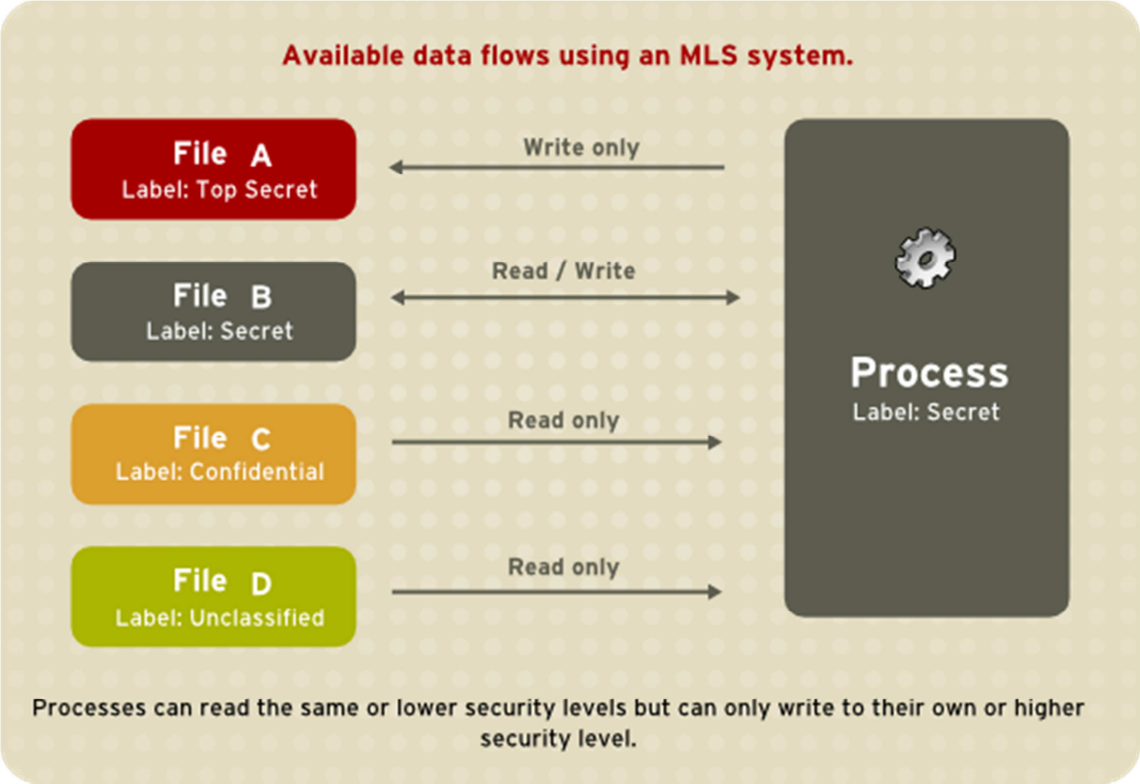


FIGURE 4: MODELE BELL-LA PADULA[4]

Ceci signifie qu'un processus de niveau inférieur ne peut pas lire à partir d'un fichier ou d'un processus d'un niveau supérieur et qu'un processus de niveau supérieur n'a pas le droit d'écrire dans un processus ou un fichier d'un niveau inférieur.

[4] Figure extraite du guide de déploiement RedHat (figure 47.3) : http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/sec-mls-ov.html

FIGURE 5: DIRECTION DU FLUX DE DONNEES DANS MLS[5]

Le but de ce modèle est d'empêcher que des données du niveau top secrète, par exemple, circulent vers les niveaux inférieurs. Les données d'un niveau top secret ne sont jamais écrites dans un niveau inférieur. Le flux de données est toujours montant.

2.3 ARCHITECTURE

Nous pouvons représenter le système Linux avec un modèle en couche. L'utilisateur n'a la possibilité d'effectuer des actions sur les fichiers et le matériel du système qu'en utilisant des applications. Les applications communiquent avec le noyau à l'aide des librairies qui effectuent des appels systèmes.

[5] Figure extraite du guide de déploiement RedHat (figure 47.2) : http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/sec-mls-ov.html

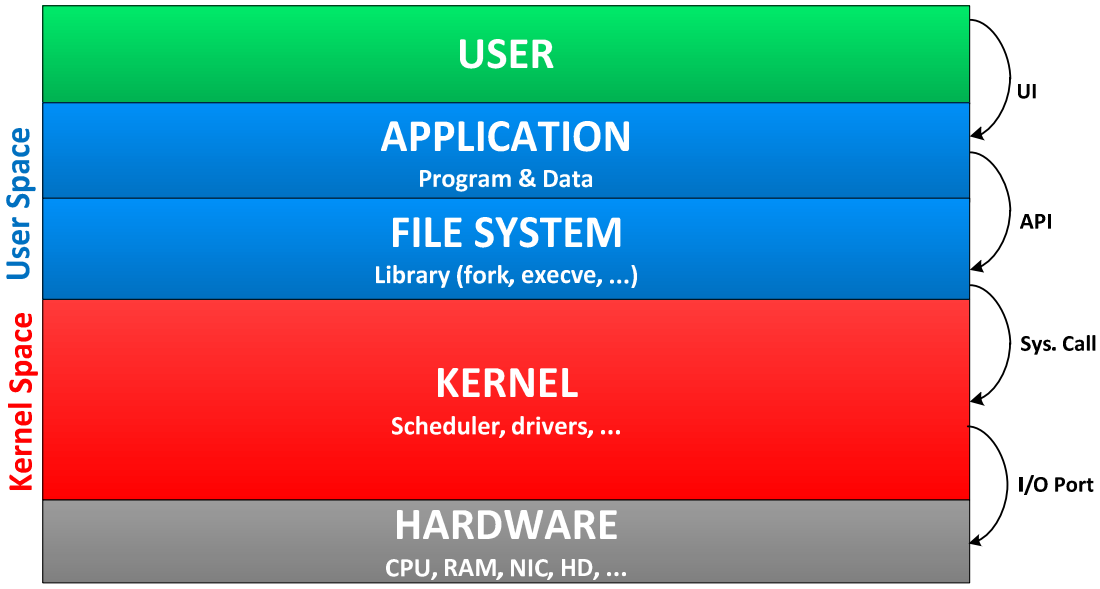


FIGURE 6: MODELE EN COUCHE DE LINUX[6]

SELinux contrôle si l'application qui effectue l'appel système a le droit d'accomplir telle action. Anciennement, SELinux fut introduit en tant que patch au kernel. Par la suite, Linus Torvalds conscient de la nécessité de sécurisé le système Linux avec d'autres mécanisme de contrôles que le mécanisme standard, proposa alors de créer une architecture modulaire qui permet de renforcer le contrôles d'accès. C'est ainsi que naquit le **Linux Security Modules (LSM)**. LSM fut intégré au noyau en version 2.6. Il permet à un module de sécurité, par exemple SELinux ou AppArmor, de s'enregistrer auprès de lui et ensuite de lui mettre à disposition des crochets, appelées hooks en anglais, pour capturer les appels système. Aucune sécurité n'est ajoutée par le LSM, tout le mécanisme du contrôle d'accès sera implémenté dans le ou les modules qui seront enregistrés auprès de lui.

[6] Sécurité Linux de Frédéric RAYNAL et Damien AUMAITRE
http://ws.edu.isoc.org/data/2007/147211836047d8efbe7b18d/linux-handout.pdf

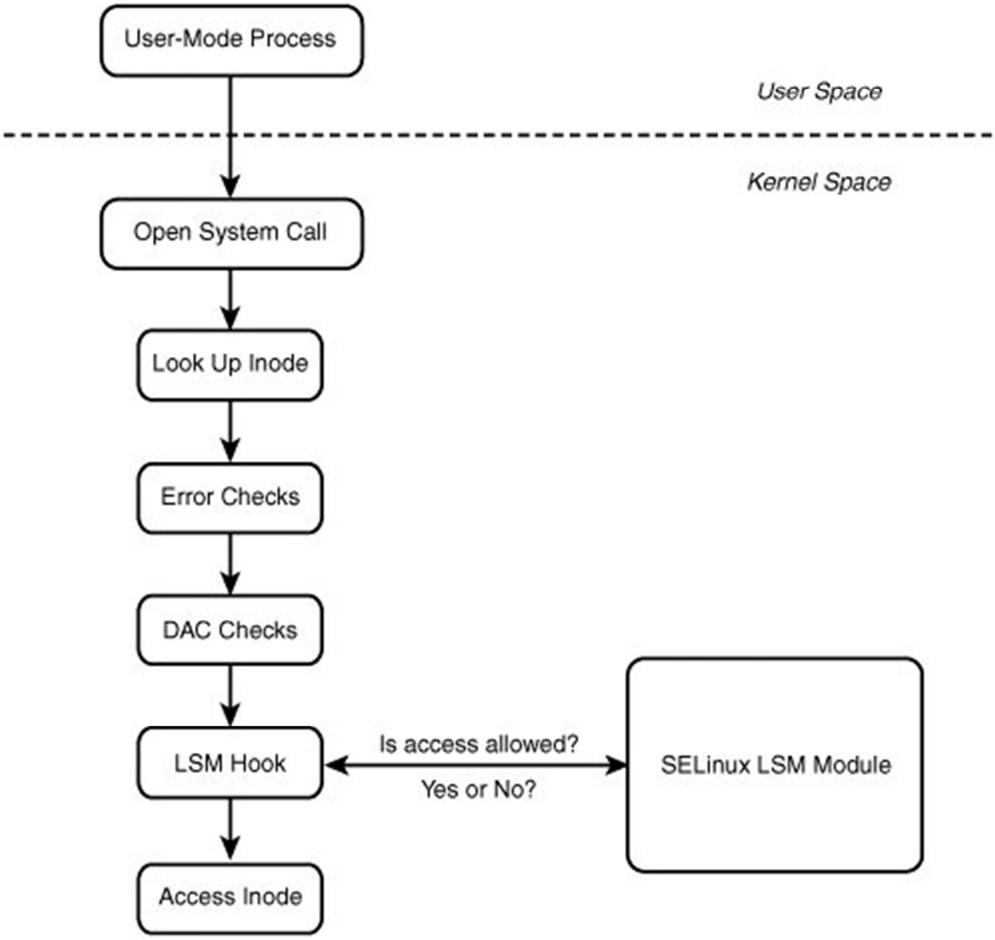


FIGURE 7: POSITION DU LINUX SECURITY MODULE DANS LE MODELE EN COUCHE[7]

Le LSM Hook intervient à la suite du contrôle standard Linux DAC. Le module SELinux ne remplace pas le DAC, mais il intervient après celui-ci. Par conséquent, SELinux ne peut que restreindre l'accès aux ressources du système.

[7] SELinux by Examples p.41 (figure 3-1)

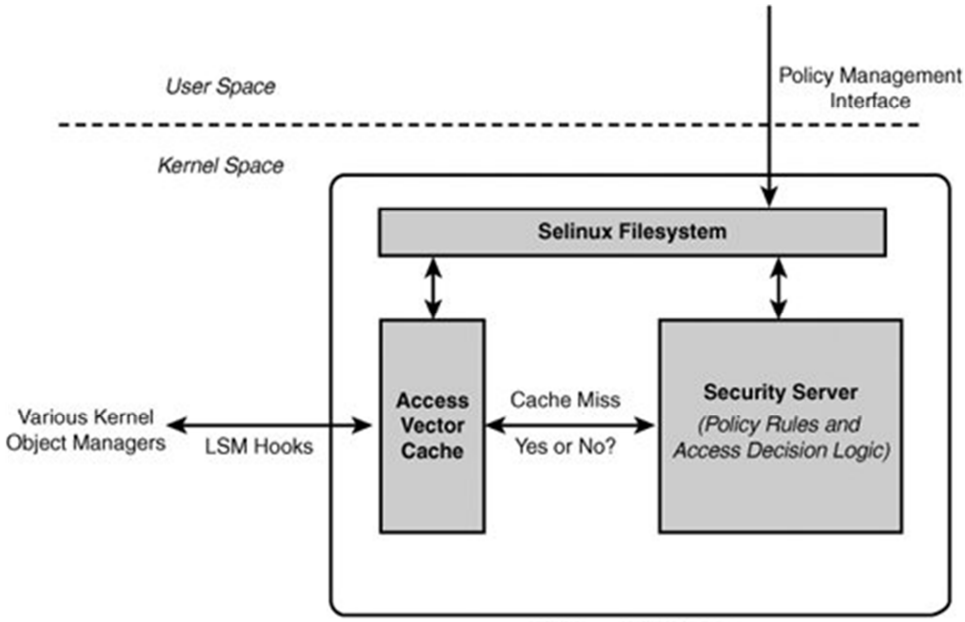


FIGURE 8: LE MODULE SELINUX[8]

Le système de fichier SELinux sert à contenir les règles actives du système qui seront chargé dans le **Security Server**, à récolter des statistiques de l'**AVC** et à d'autres fonctions. L'**Access Vector Cache** (AVC) contient seulement une partie des règles. Le cache réduit le temps du contrôle d'accès pour éviter d'impacter les performances du système. Dans le cas où l'AVC est incapable de répondre à une demande d'accès, celui-ci envoi une requête de recherche au Security Server.

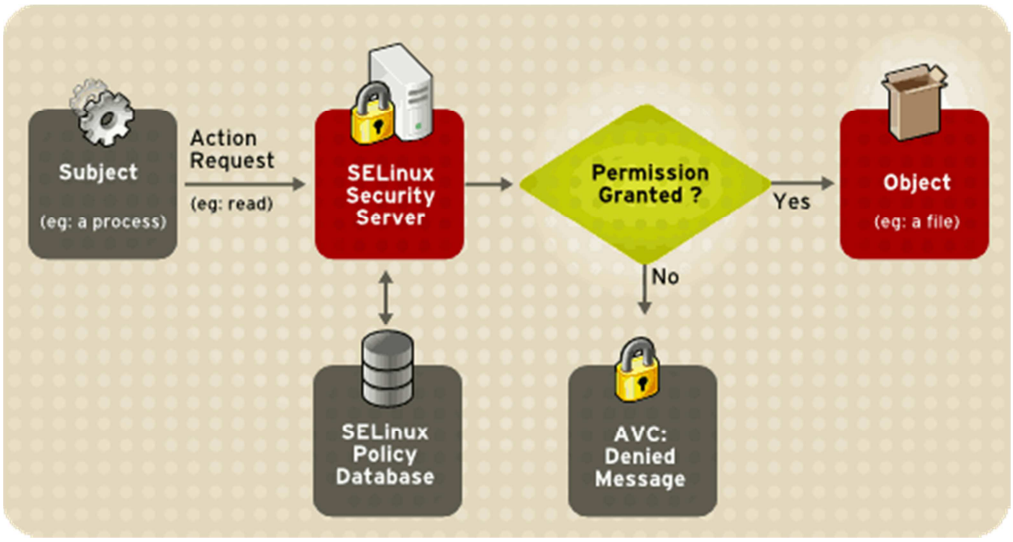


FIGURE 9: SELINUX SECURITY SERVER ET AVC LOG MESSAGE[9]

Quand un processus demande l'accès à un fichier, le serveur de sécurité et plus exactement l'AVC vérifie dans sa base de données et dans la base de

[8] SELinux by Example p.42 (figure 3-2)
[9] Figure extraite du RedHat Deployment Guide (figure 44.1) :
http://www.centos.org/docs/5/html/5.2/Deployment_Guide/s1-SELinux-overview.html

données du serveur de sécurité si une règle autorise l'accès, si aucune règle n'est trouvée, alors l'accès est refusé et puis le refus est enregistré par l'AVC dans un fichier log.

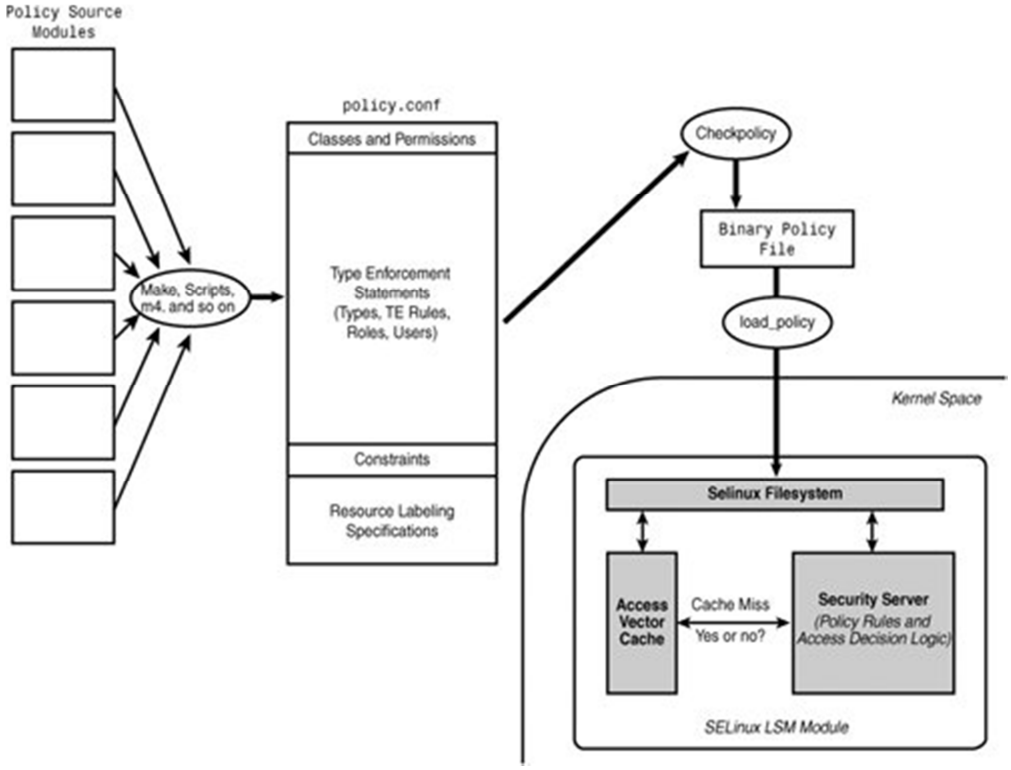


FIGURE 10: CHARGEMENT DE REGLES DANS LE MODULE SELINUX[10]

Les règles SELinux sont volumineuses (voir figure 16). Pour cette raison les règles sont modulaires et séparées dans plusieurs fichiers lors de leurs écritures. Un fichier policy.conf est créé et regroupera toutes les règles SELinux écrites dans les différents fichiers. Checkpolicy compile le fichier source policy.conf en un fichier binaire. Le chargement des règles dans le serveur de sécurité se fait ensuite en invoquant la commande load_policy.

2.4 ACTIVER SELINUX

Dans Linux, trois modes sont disponibles :

1. **Enforcing,** mode par défaut dans Fedora 16. SELinux est activé et renforce la sécurité du système.
2. **Permissive,** SELinux continu à contrôler et inscrire les refus potentiel dans les fichiers logs, mais par contre tous les accès sont autorisés comme si SELinux n'était pas en fonction.
3. **Disabled,** SELinux est désactivé.

[10] SELinux by Example p.52 (figure 3-6)

Mini labo

- **Afficher status de SELinux**

```
1    [root@SELinux ~]# sestatus
2
3    SELinux status:                 enabled
4    SELinuxfs mount:                /sys/fs/selinux
5    Current mode:                   enforcing
6    Mode from config file:          enforcing
7    Policy version:                 26
8    Policy from config file:        targeted
```

Le système de fichier SELinuxfs est monté dans le dossier /sys/fs/selinux (figure 8). Nous pouvons changer le mode de SELinux temporairement à l'aide de la commande setenforce. C'est pour cette raison qu'il y a 2 lignes en lien avec mode de SELinux (lignes 5 et 6). Le type de **politique est targeted**. Donc seule certaines applications critiques du point de vue sécurité sont limitées dans un domaine restreint et d'autres moins critiques sont dans des domaines plus vastes. Pour plus de détail sur le type de politique targeted lire l'annexe A.1.

- **Changer le mode de façon permanente**

```
1    [root@SELinux ~]# cat /etc/selinux/config
2
3    On systems with SELinux disabled, the SELINUX=disabled option is
     configured in /etc/selinux/config:
4    # This file controls the state of SELinux on the system.
5    # SELINUX= can take one of these three values:
6    #       enforcing - SELinux security policy is enforced.
7    #       permissive - SELinux prints warnings instead of enforcing.
8    #       disabled - No SELinux policy is loaded.
9    SELINUX=enforcing
10   # SELINUXTYPE= can take one of these two values:
11   #       targeted - Targeted processes are protected,
12   #       mls - Multi Level Security protection.
13   SELINUXTYPE=targeted
```

Pour modifier le mode de SELinux, éditer la ligne 9 du fichier config.

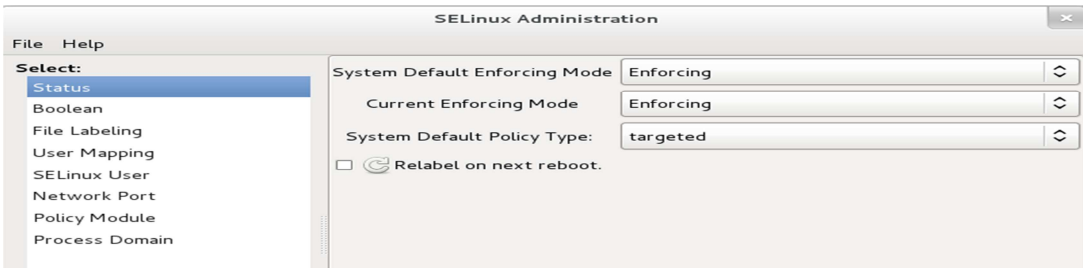
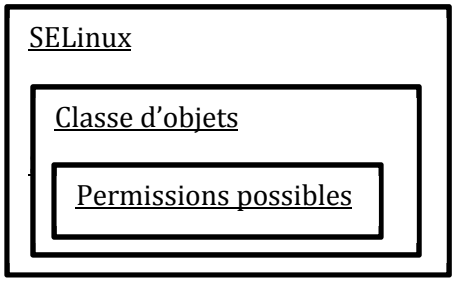


FIGURE 11: SELINUX MANAGEMENT

SELinux management n'est pas installer dans Fedora 16. Pour pouvoir profiter de la gestion GUI de SELinux il faut installer le paquetage policycoreutils-gui (voir annexe A.3).

2.5 CLASSES D'OBJETS ET PERMISSIONS

Les **classes d'objets**, sont des catégories de ressources qui peuvent être manipulées dans le système d'exploitation Linux. Par exemple les fichiers, les sockets, les tubes, les processus, les sémaphores, les tables d'une base de données et d'autres éléments ont une classe d'objets qui leurs corresponds à chacun. Le nombre de classes d'objets présent dépend de la version de SELinux et de la version du noyau Linux.
L'accès à ces objets est de type **liste blanche** et est défini à l'aide de **règles**. Le principe d'une liste blanche est que tout ce qui n'est pas explicitement permis est interdit. **Aucun accès n'est permis par défaut**. Une liste de **permissions** possible est liée à chaque classe d'objets.



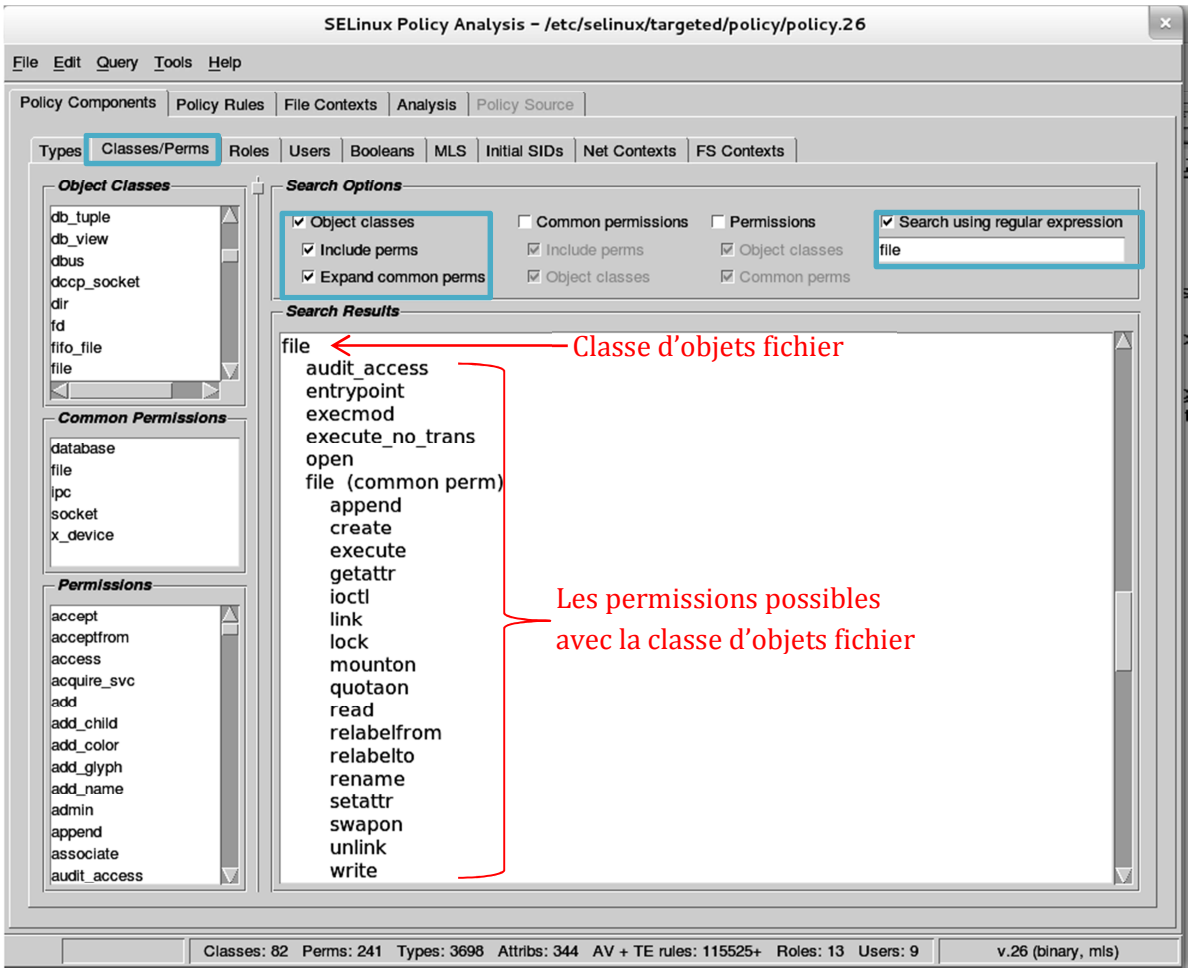


FIGURE 12 : APOL, UTILITAIRE GRAPHIQUE D'ANALYSE DES FICHIERS DE REGLES SELINUX

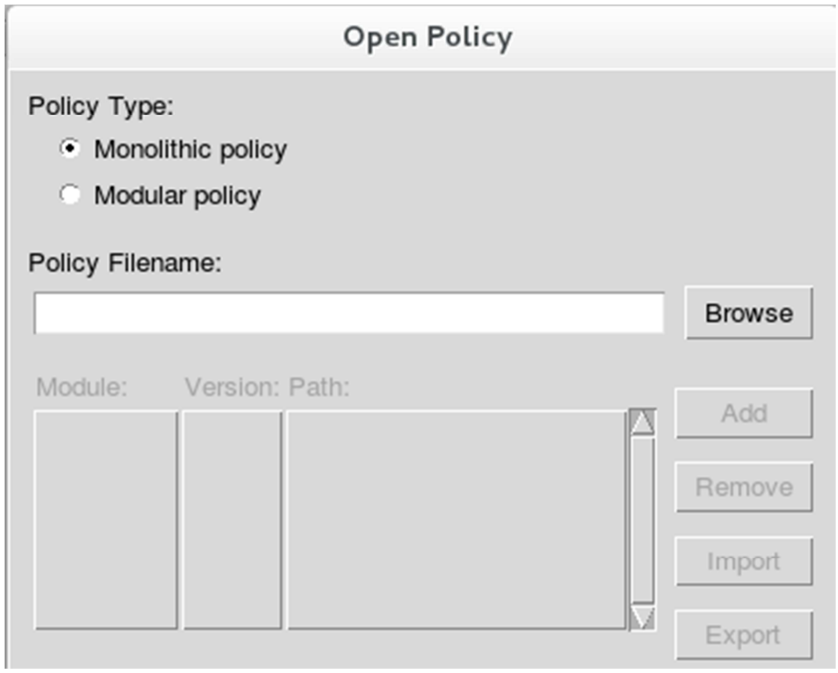
Dans l'onglet « Classes/Perms » nous avons les classes d'objets, les permissions communes et les permissions. Les permissions communes ne sont rien d'autre qu'un groupe de permissions. Attention les permissions communes peuvent avoir la même dénomination qu'une classe d'objets. Nous avons la possibilité de faire des recherches. Dans la prise d'écran ci-dessus, j'ai affiché toutes les classes avec le mot « file » dedans et j'ai demandé à ce que les permissions communes soient dépliées.
Bien que dans Linux tout est fichier, dans Fedora 16 il y a tout de même 82 classes d'objets et 241 permissions défini. Nous pouvons déjà nous rendre compte de la **fine granularité** des règles SELinux.
Nous pouvons diviser les classes d'objets en quatre catégories[11] :

- Les classes d'objets liées aux fichiers
- Les classes d'objets liées au le réseau
- Les classes d'objets liées au système V IPC (communication interprocessus)
- Les classes d'objets diverses

Mini labo

- **Démarrer APOL et charger les règles de Fedora 16**

APOL est capable d'analyser des fichiers de règles binaire ou des fichiers de règles source.



[11] SELinux by Example p. 67

- L'option monolithic policy est un fichier source ou binaire qui contient toutes les règles du système et ne dépend d'aucun autre fichier.
- L'option modular policy permet d'ajouter un fichier de base et des modules dépendant qui viennent étendre les règles contenu dans le fichier de base.

Je m'intéresse pour le moment au fichier de base de Fedora 16. Pour charger les règles de ce système :

- Ouvrir le menu file d'APOL
- Cocher Monolithic policy
- Charger le fichier binaire
  - /etc/selinux/targeted/policy/policy.[ver]

2.6 CONTEXTE DE SECURITE

Tous les ressources (fichiers, sockets,…) du système possèdent un **contexte de sécurité**. Celui-ci est composé de trois identificateurs obligatoires :

Utilisateur:Rôle:Type

Par convention, les identifiants possèdent un suffixe : _u pour l'utilisateur, _r pour le rôle et _t pour le type. La terminologie dans SELinux est quelque peu confuse. **Label de sécurité** et **domaine de sécurité** sont des synonymes de contexte de sécurité.

Le champ utilisateur contient l'identifiant de l'utilisateur SELinux créateur de l'objet. Ce champ n'est pas utilisé dans le contrôle de renforcement TE car ce sont uniquement le rôle et type qui sont utilisés.
Le champ rôle est valable pour les processus et pour les utilisateurs. Un fichier a toujours le même rôle object_r. Object_r est un rôle spécial codé en dur dans SELinux et est implicitement alloué à tous les fichiers.
Le champ type définit un domaine (de sécurité) pour les processus et un type pour les fichiers. Les règles SELinux permettent de définir quels types sont accessible à un domaine et quels domaines sont accessibles à un autre domaine. L'élément fondamental dans le contexte de sécurité pour appliquer le type enforcement est le champ type.
Pour chaque fichier, le contexte de sécurité est stocké comme attribut étendu[12] dans le système de fichier. La chaîne de caractères qui définit

12 http://en.wikipedia.org/wiki/Extended_file_attributes

chaque identificateur est déclaré dans les règles SELinux. Le contrôle des accès SELinux est basé sur le contexte de sécurité associé avec le sujet et le contexte de sécurité de l'objet ciblé. Le sujet est le processus qui demande l'accès à un objet, à une ressource.
L'identité et le rôle n'entre pas en compte directement dans la décision d'accès.

Mini labo

Plusieurs commandes de base Linux ont été modifiées par l'ajout de l'argument –Z. Cet argument permet d'afficher ou de modifier les informations lié au contexte de sécurité. Exemple des commandes modifié ps, ls, dir, mkdir, …

- **Affichage du contexte de sécurité d'un fichier :**

```
1    [test@SELinux ~]$ ls -lZ /bin/pwd
2    -rwxr-xr-x. root root system_u:object_r:bin_t       pwd
```

- **Affichage du contexte de sécurité, domaine de sécurité d'un processus en cours d'exécution :**
  - Lancer la commande passwd dans un terminal et dans un autre terminal lancer la commande ci-dessous :

```
1    [test@SELinux ~]$ ps -eZ | grep passwd
2    unconfined_u:unconfined_r:passwd_t 13212 pts/1 00:00:00 passwd
```

- **Affichage du contexte de sécurité du shell d'un utilisateur :**

```
1    [test@SELinux ~]$ id -Z
2    unconfined_u:unconfined_r:unconfined_t
```

Partie facultative

- **Récupérer les attributs étendus relié avec SELinux d'un fichier :**

```
1    [root@SELinux ~]# getfattr -n security.selinux /bin/pwd
2    getfattr: Removing leading '/' from absolute path names
3    # file: bin/pwd
4    security.selinux="system_u:object_r:bin_t:s0"
```

SELinux ne peux être utilisé qu'avec un système de fichier qui supporte les attributs étendus.

2.7 TYPE ENFORCEMENT

2.7.1 ACCESS VECTOR ET TYPE TRANSITION RULES

Type Enforcement est la charpente de SELinux pour implémenter le mécanisme MAC flexible. Il y a deux familles de règles type enforcement :

Access Vector Rules	Type Rules
allow	type_transition
neverallow	type_change
auditallow	type_member
auditdeny	
dontaudit	

Les deux types de règles primordiales pour le fonctionnement de TE sont les règles allow et type transition. Les autres règles sont en lien avec l'inscription des événements dans les fichiers log. La règle « allow » de la famille « Access Vector Rule » est la seule règle dans SELinux qui donne l'autorisation d'accès à une ressource donnée. Voici la syntaxe employée :

```
allow source_type target_type : object_class perm ;
```

Le « source_type » est le domaine du processus. Le « target_type » est le type de l'objet accédé par le processus. L'« object_class » est la classe de l'objet auquel nous voulons accéder. Le champ « perm » est la liste de permission(s) que le domaine est autorisé à effectuer sur la classe typée avec le champ « target_type ».

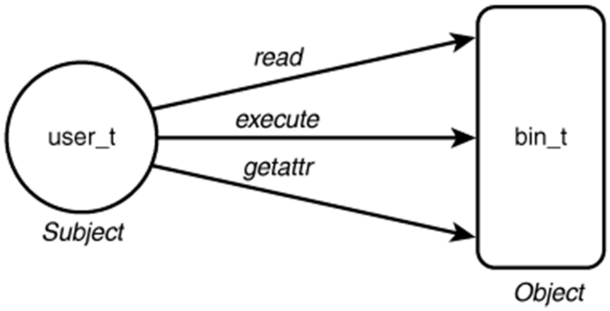


FIGURE 13: REPRESENTATION D'UNE REGLE ALLOW[13]

Une autre notion importante est la transition de domaine, en anglais « domain transition ». La transition de domaine est fondamentale pour appliquer le principe du moindre privilège et restreindre les possibilités d'accès d'une application. Avec les règles allow vector et type transition, nous pouvons réaliser ce qui a été expliqué dans la figure 3 de ce document.

[13] SELinux by Example p. 20 (figure 2-1)

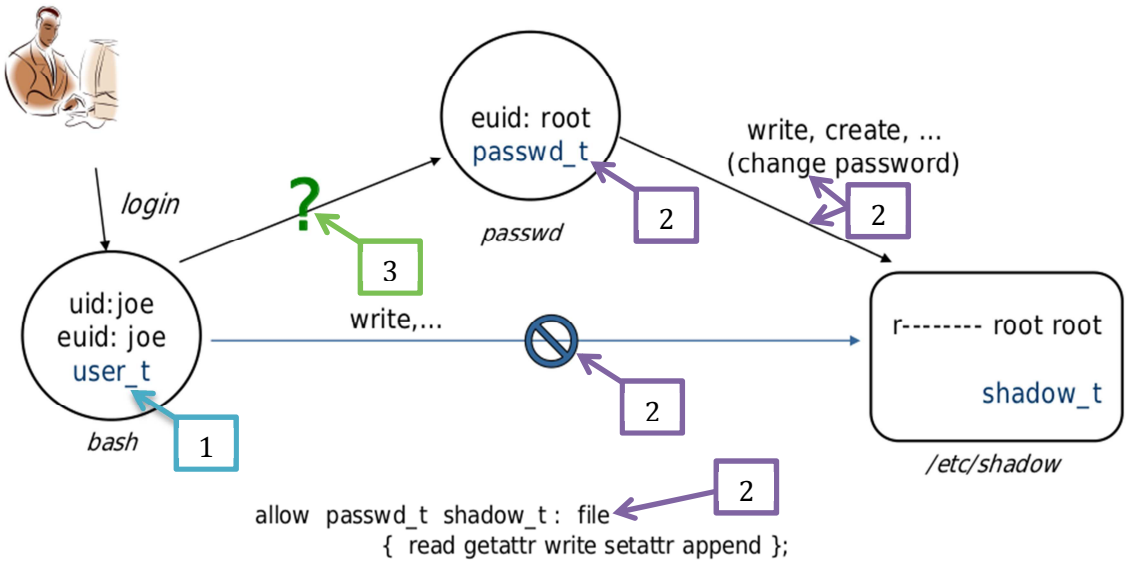


FIGURE 14: PROBLEMATIQUE TRANSITION DE DOMAINE[14]

Quand l'utilisateur Joe se connecte au système à travers le login, celui-ci se fait attribuer un domaine selon la politique mise en œuvre. Dans la figure ci-dessus, c'est le domaine user_t qui lui est attribué **[1]**. Le domaine user_t est utilisé pour restreindre les processus lancés par l'utilisateur. Le fichier shadow ne peut être accédé que depuis le domaine passwd_t **[2]**. Il faudra transiter du domaine user_t au domaine passwd_t pour permettre à Joe de modifier son mot de passe **[3]**. Bien évidemment la transition d'un domaine à un autre est strictement réglementée pour éviter qu'un processus change de domaine selon son bon vouloir. Voici comment la transition est orchestrée :

```
type_transition   source_type target_type : process default_type ;
```

Le « source_type » est le domaine d'un utilisateur ou d'un processus. Le « target_type » est le type d'un fichier exécutable. « Process » est la classe objets processus. « Default_type » est le domaine auquel le nouveau processus lancé transit lorsque le domaine « source_type » exécute un fichier binaire avec un type « target_type ».

Pour qu'une transition de domaine ait lieu, il faut trois règles « allow » pour l'autoriser et une règle « type transition ». Par exemple, au démarrage du système un processus d'initialisation dans le domaine init_t s'occupe de lancer les services. Imaginons que le processus d'initialisation doit lancer le serveur apache. Le serveur apache ne doit pas hériter du domaine du processus père et doit être lancé dans le domaine httpd_t pour qu'il soi restreint. Voici la règle de transition :

```
type_transition initrc_t httpd_exec_t : process httpd_t ;
```

[14] SELinux by Example p. 23 (figure 2-3)

Cette règle ne sera acceptée que si l'exécutable apache est marqué avec le type httpd_exec_t. Qu'une règle allow autorise le domaine initrc_t à exécuter un fichier marqué avec le type httpd_exec_t.

```
1)    allow initrc_t httpd_exec_t : file { getattr execute } ;
```

Qu'une règle allow défini que les fichiers de type httpd_exec_t sont des points d'entrés au domaine http_t :

```
2)    allow httpd_t httpd_exec_t : file entrypoint ;
```

Et pour finir qu'une règle autorise la transition du domaine initrc_t au domaine httpd_t :

```
3)    allow initrc_t httpd_t : process transition ;
```

La règle transition de type indique vers quel domaine le processus lancé doit transiter :

```
4)    type_transition init_rc httpd_exec_t : process httpd_t ;
```

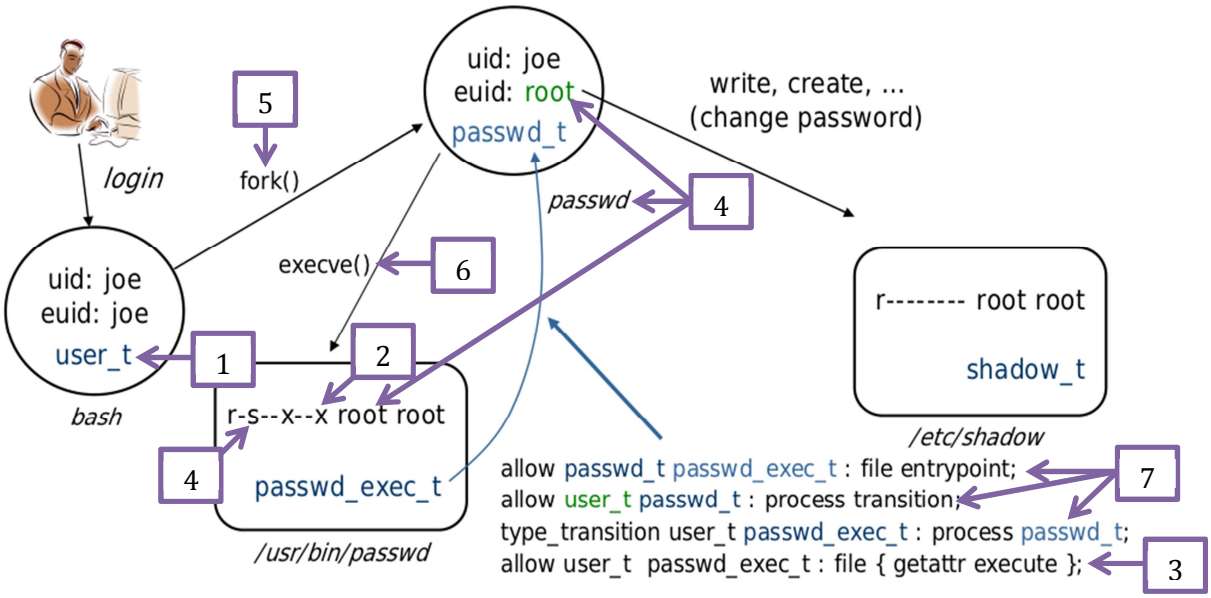
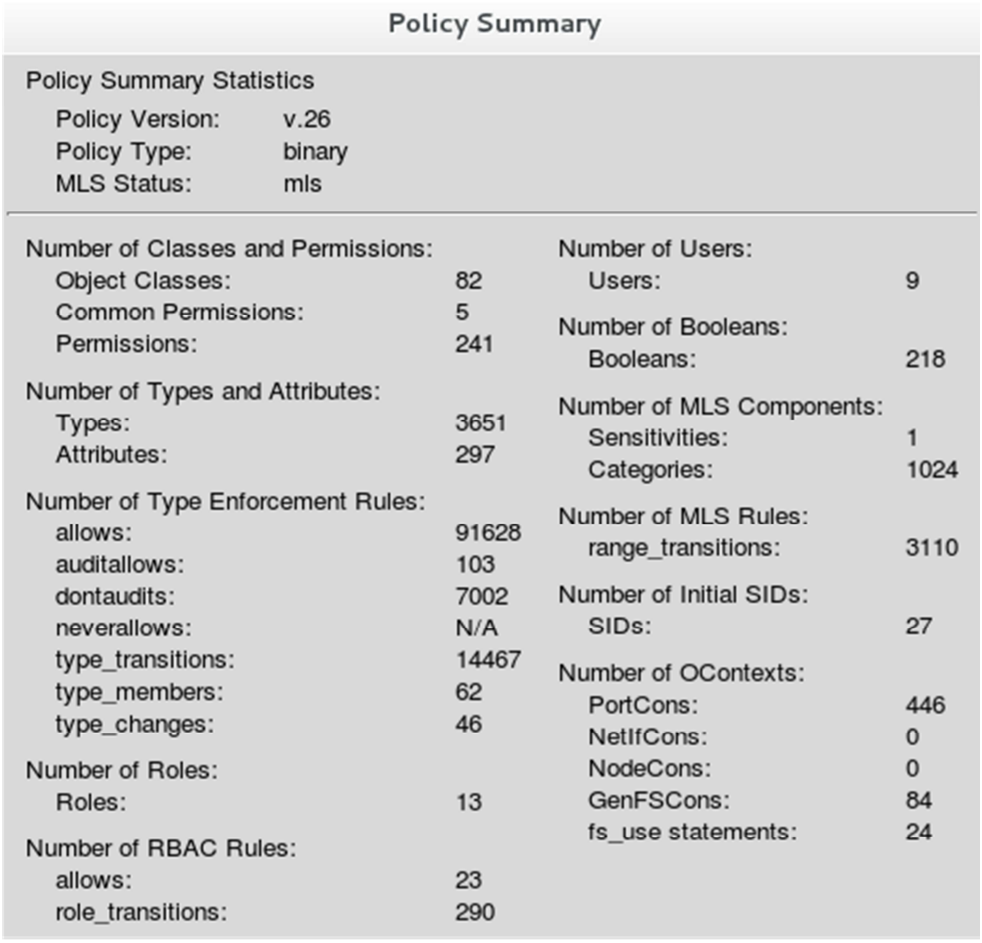


FIGURE 15: EXEMPLE DE TRANSITION DU DOMAINE USER_T A PASSWD_T[15]

Dans la figure ci-dessus, Joe se connecte au système. Un bash avec un domaine user_t lui est créé **[1]**. Joe à le droit d'exécuter le programme passwd car il a les droits DAC `r-s--x--x` sur l'exécutable, le dernier « x » indique que tout le monde a le droit d'exécuter passwd **[2]**. La dernière règle allow donne le droit au domaine user_t d'exécuter les fichiers marqués avec le type passwd_exec_t **[3]**. Le setuid bit « s », dans les droits d'accès, indique que le processus passwd doit avoir comme son utilisateur effectif égale au propriétaire du fichier shadow, ici root, afin qu'il puisse modifier son mot de passe **[4]**. Pour se faire, une copie conforme du processus bash est créé à l'aide de l'appel système fork() **[5]**.

[15] SELinux by Example p. 26 (figure 2-5)

La copie du bash lance l'appel système execve() pour charger le binaire du programme passwd à la place du binaire bash et change la valeur de l'utilisateur effectif à root [6]. La transition dans le domaine passwd_t est autorisée par les deux autres règles allow et la règle type_transition [7].



Policy Summary

Policy Summary Statistics

Policy Version:	v.26
Policy Type:	binary
MLS Status:	mls

Number of Classes and Permissions:	
Object Classes:	82
Common Permissions:	5
Permissions:	241
Number of Types and Attributes:	
Types:	3651
Attributes:	297
Number of Type Enforcement Rules:	
allows:	91628
auditallows:	103
dontaudits:	7002
neverallows:	N/A
type_transitions:	14467
type_members:	62
type_changes:	46
Number of Roles:	
Roles:	13
Number of RBAC Rules:	
allows:	23
role_transitions:	290
Number of Users:	
Users:	9
Number of Booleans:	
Booleans:	218
Number of MLS Components:	
Sensitivities:	1
Categories:	1024
Number of MLS Rules:	
range_transitions:	3110
Number of Initial SIDs:	
SIDs:	27
Number of OContexts:	
PortCons:	446
NetIfCons:	0
NodeCons:	0
GenFSCons:	84
fs_use statements:	24

FIGURE 16: STATISTIQUES DES REGLES INCLUSES DANS FEDORA 16 A L'AIDE D'APOL

Dans Fedora 16, il existe 91'628 règles allow et 14'467 règles de transition. Comme il faut explicitement donner des permissions à chaque domaine sur les types qu'il les concerne, un nombre important de règles est inscrit dans le système. Mais en fait, il y a encore beaucoup plus de règles mais celle-ci sont concaténées. Des artifices ont été trouvés pour regrouper plusieurs règles allow dans une seule règle. Les attributs sont un regroupement de plusieurs types. Une règle allow qui est définie à l'aide d'attribut peut générer énormément de règles allow simple. Ce que j'entends par règle allow simple, c'est une règle qui alloue un certain nombre de permissions à un domaine sur une classe d'objets d'un type donné.

Prenons un exemple, un attribut nommé « attribut_ex » regroupe les types abc_t, def_t et ghi_t. Un autre attribut nommé « application_domain_type »

regroupe tous les domaines des applications du système, celui-ci contient 240 domaines. Avec une règle du style :

```
allow application_domain_type attribut_ex : file {ioctl read getattr lock open}
```

La règle ci-dessus cache l'existence de 720 règles allow simples. Dans Fedora 16, il existe 3'651 types et 297 attributs. Il y a 13 rôles prédéfinis et 9 utilisateurs SELinux.

Mini labo

- **Transition de domaine**

```
1   [test@SELinux ~]$ id -Z
2   unconfined_u:unconfined_r:unconfined_t
```

Le domaine unconfined_t permet uniquement de restreindre les applications les plus exposées aux attaques. La plupart des applications lancées par un utilisateur du domaine unconfined_t seront lancé sans qu'il y ait des transitions vers d'autres domaines. Pour plus de détails sur le domaine unconfined, se référer à l'annexe A.1.

- Lancer le navigateur firefox et observer dans quel domaine il est lancé :

```
1   [test@SELinux ~]$ ps –eZ | grep firefox
2   unconfined_u:unconfined_r:unconfined_t  2034 ? 00:02:20 firefox
```

- Lancer la commande passwd dans un terminal. Lancer un deuxième terminal et relever le changement de domaine :

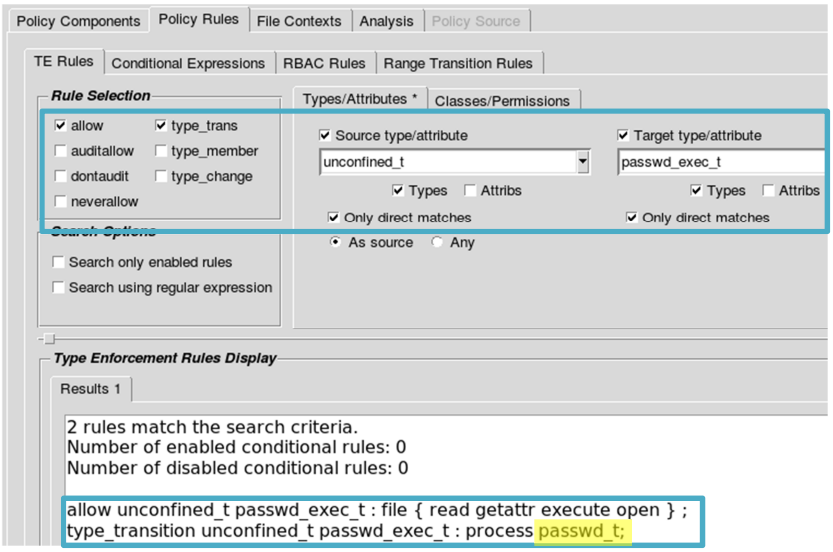
```
1   [test@SELinux ~]$ ps –eZ | grep passwd
2   unconfined_u:unconfined_r:passwd_t 13212 pts/1 00:00:00 passwd
```

- **Rechercher les règles qui autorisent la transition dans le domaine passwd :**
  - Observer le type du fichier lancer

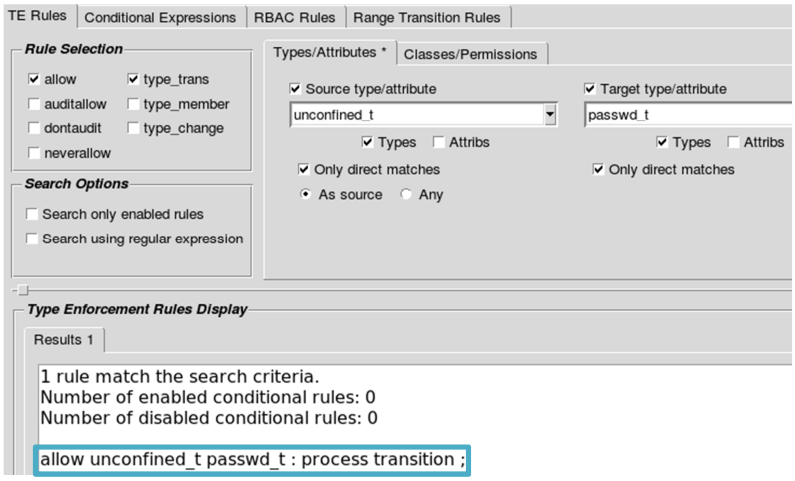
```
1   [test@SELinux ~]$ which passwd
2   /usr/bin/passwd
3   [test@SELinux ~]$ ls –lZ /usr/bin/passwd
4   -rwsr-xr-x. root root system_u:object_r:passwd_exec_t passwd
```

Nous avons obtenu au login un shell dans le domaine unconfined_t.

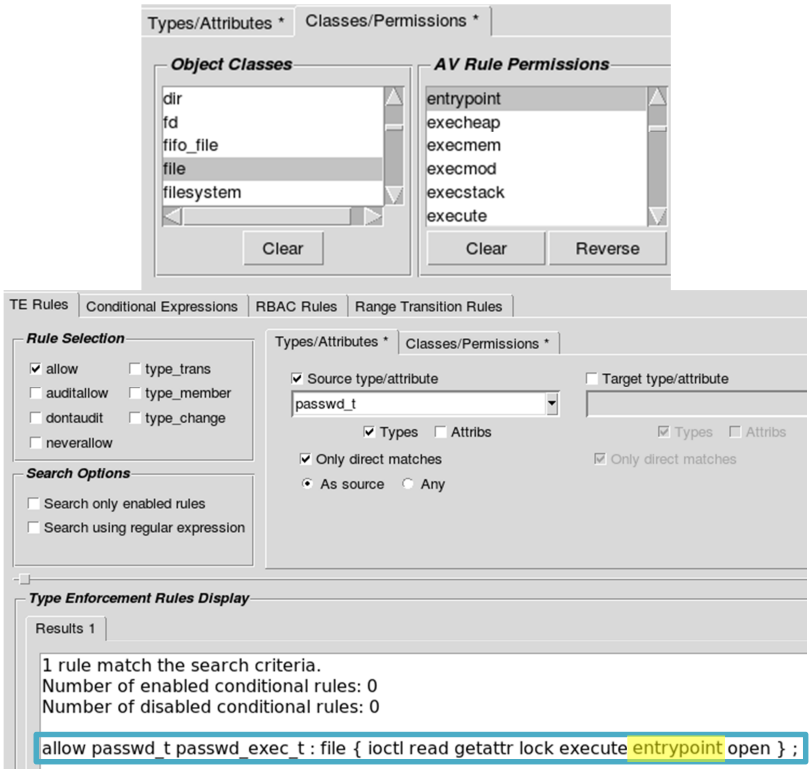
- À l'aide de ce shell lançer l'exécutable passwd (type passwd_exec_t).
- Ouvrir APOL et charger les règles du système comme vu dans le mini labo du chapitre 2.5.
- Rechercher des règles avec les informations obtenues (unconfined_t et passwd_exec_t) :



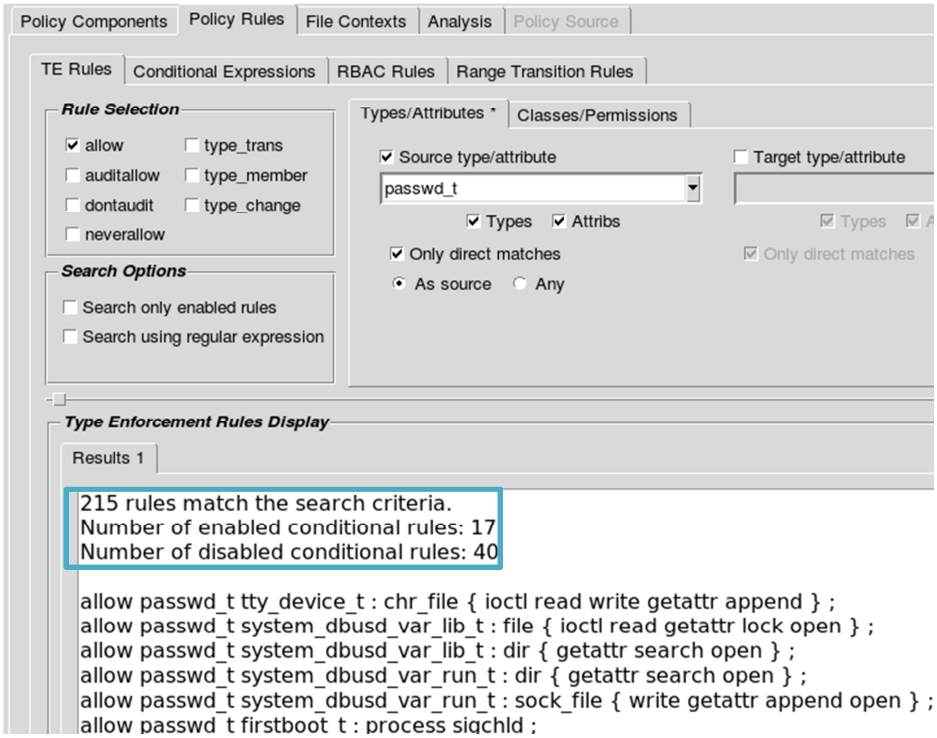
- Avec la règle de transition nous avons obtenu le domaine vers lequel nous transitons (passwd_t).



- Trouver le point d'entrée au domaine passwd_t. Je filtre sur l'objet fichier et la permission entrypoint pour avoir uniquement la règle qui permet d'accéder ce domaine :

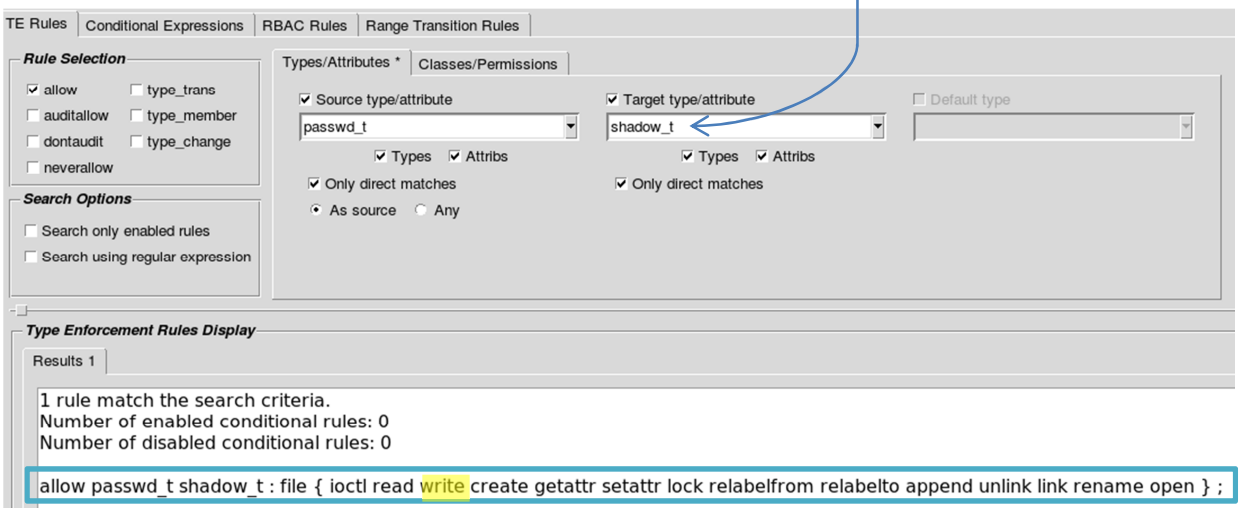


- Afficher la règle allow qui permet au domaine passwd_t d'écrire dans le fichier shadow :



J'ai obtenu trop de règles. Filtrer sur le fichier /etc/shadow pour n'avoir que la règle qui nous intéresse :

```
1    [test@SELinux ~]$ ls -lZ /etc/shadow
2    ----------. Root root system_u:object_r:shadow_t /etc/shadow
```



Constat : Le domaine passwd_t a bien le droit d'écrire dans un fichier de type shadow_t.

Complément d'informations

- Obtenir des statistiques en ligne de commande :

```
1    [test@SELinux ~]$ seinfo
2    Statistics for policy file: /etc/selinux/targeted/policy/policy.26
3    Policy Version & Type: v.26 (binary, mls)
4
5       Classes:            82    Permissions:       241
6       Sensitivities:       1    Categories:       1024
7       Types:            3698    Attributes:        344
8       Users:               9    Roles:              13
9       Booleans:          218    Cond. Expr.:       256
10      Allow:           93068    Neverallow:          0
11      Auditallow:        111    Dontaudit:        7086
12      Type_trans:      15152    Type_change:        62
13      Type_member:        46    Role allow:         23
14      Role_trans:        294    Range_trans:      3154
15      Constraints:        93    Validatetrans:       0
16      Initial SIDs:       27    Fs_use:             24
17      Genfscon:           84    Portcon:           441
18      Netifcon:            0    Nodecon:             0
19      Permissives:        52    Polcap:              2
```

Au démarrage du système, plusieurs transitions de domaine se produisent. Kernel_t est le domaine de départ. Le programme d'initialisation transite ensuite dans le domaine init_t pour poursuivre l'initialisation. Le domaine initrc_t a la tâche de lancer les services. Le domaine local_login_t transite dans le domaine associé à l'utilisateur. Et permettra, par exemple, de lancer le processus passwd dans le domaine passwd_t. Toutes ces transitions sont possible uniquement parce qu'il existe des règles pour les autoriser.

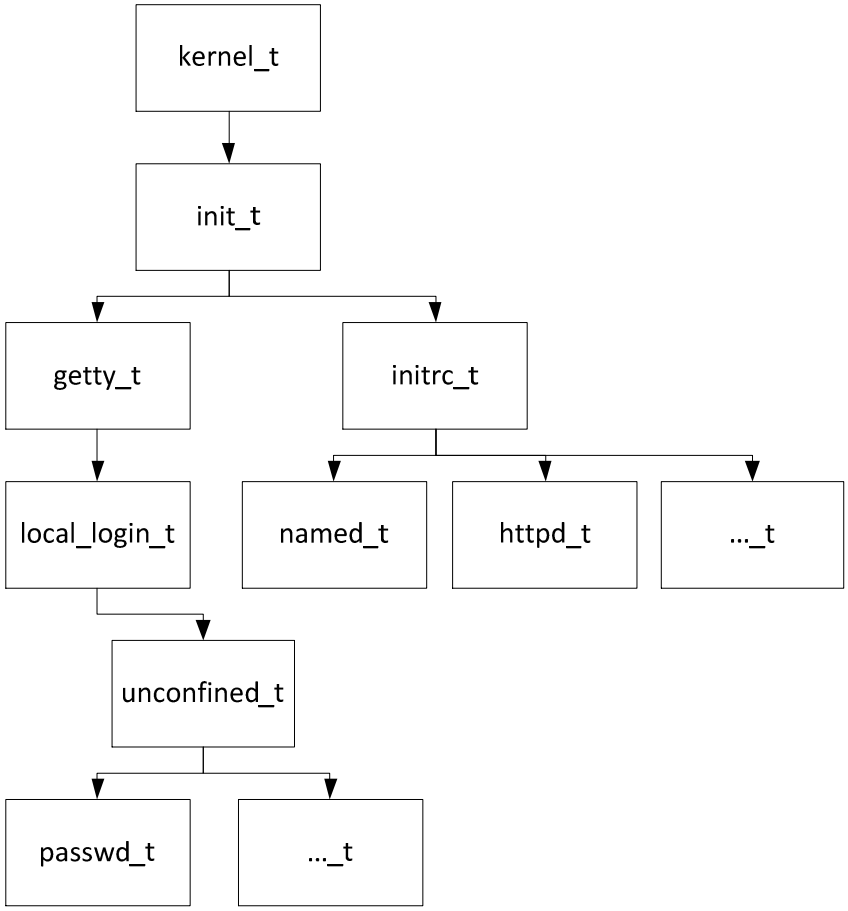


FIGURE 17: TRANSITION DE DOMAINE AU DEMARRAGE DU SYSTEME

2.7.2 MAPPING UTILISATEURS

Les utilisateurs Linux sont reliés avec un utilisateur SELinux avec une relation (1:1). Lors du login, l'utilisateur Linux lui est associé un utilisateur SELinux. Les utilisateurs Linux ne sont pas réutilisés parce que ceux-ci sont susceptibles de changer durant une session. Par exemple, l'appel de la commande su permet à un utilisateur Linux d'agir en tant qu'un autre utilisateur et ceci rend plus difficile la traçabilité des actions des utilisateurs. Un utilisateur SELinux est immuable durant une session.

- **Afficher la table de relation entre les utilisateurs Linux et SELinux :**

```
1    [root@SELinux ~]# semanage login -l
2    Login Name           SELinux User
3
4    __default__          unconfined_u
5    root                 unconfined_u
6    system_u             system_u
```

Par défaut dans Fedora 16, tous les utilisateurs sont reliés avec l'utilisateur SELinux unconfined_u par l'entrée __default__. Le root est spécifiquement relié avec l'utilisateur SELinux unconfined_u (ligne 5) pour lui laisser un accès souple au système même si l'administrateur système a décidé de restreindre les autres utilisateurs Linux en modifiant

la ligne __default__. Le domaine unconfined_t est très peu restreint par la définition de règles extrêmement laxistes. Par exemple :

```
allow unconfined_t file_type : file { ioctl read write create getattr
setattr lock relabelfrom relabelto append unlink link rename execute swapon
quotaon mounton execute_no_trans entrypoint open audit_access } ;
```

La règle ci-dessus fourni au domaine unconfined_t la permission sur la classe d'objets fichier les permissions suivante dans le cas où les fichiers sont tagués avec un type listé dans l'attribut file_type. À savoir que file_type regroupe tous les types de fichiers présents dans le système. Le domaine unconfined_t a été conçu pour éviter que les utilisateurs désactivent SELinux en disant pourquoi il me bloque l'accès à ces fichiers alors que je suis en root. Les transitions de domaine seront faites aux domaines plus restreints uniquement lors du lancement d'applications sensibles.
Pour renforcer la sécurité du système, nous pouvons à travers l'outil semanage faire en sorte que tous les utilisateurs standards soient automatiquement mappés à un utilisateur SELinux restreint, par exemple l'utilisateur SELinux user_u. Pour plus d'information sur les utilisateurs SELinux présent dans Fedora, veuillez-vous reporter à l'annexe A.2.

Mini labo

- **Restreindre un utilisateur Linux :**

Le domaine unconfined_t à accès à la commande su pour passer en root.

```
1    [test@SELinux ~]$ su root
2    Password:
3    [root@SELinux ~]#
```

- Restreindre l'utilisateur test en rajoutant une entrée spécifique dans la table de mapping des utilisateurs :

```
1    [root@SELinux ~]# semanage login -a -s user_u test
```

- Vérifier que le lien entre mon utilisateur Linux et utilisateur SELinux est bien présent dans la table :

```
1    [root@SELinux ~]# semanage login -l
2    Login Name           SELinux User
3
4    __default__          unconfined_u
5    root                 unconfined_u
6    system_u             system_u
7    test                 user_u
```

- Se déconnecter et se reconnecter au système avec l'utilisateur test :

```
1    [test@SELinux ~]# id -Z
2    user_u:user_r:user_t
```

L'utilisateur test est maintenant mappé à un utilisateur SELinux user_u.

- Ressayer de passer en root dans un terminal avec le compte test :

```
1    [test@SELinux ~]$ su root
2    bash: su: command not found...
```

```
3    Failed to search for file: Did not receive a reply. Possible
     causes include: the remote application did not send a reply, the
     message bus security policy blocked the reply, the reply timeout
     expired, or the network connection was broken.
```

Partie facultative

- **L'invariabilité d'un utilisateur SELinux lors d'une session :**
  - Appuyez sur la combinaison de touches <Ctrl><Alt><F2> pour obtenir une console texte. Se connecter en tant que root :

```
1    Fedora release 16 (Verne)
2    Kernel 3.3.5-2.fc16x86_64 (tty2)
3    SELinux login :
```

Nous obtenons le prompt root :

```
1    [root@SELinux ~]#
2    [root@SELinux ~]# id -Z
3    unconfined_u:unconfined_r:unconfined_t
```

  - Passer de l'utilisateur root vers l'utilisateur restreint test :

```
1    [root@SELinux ~]# su test
```

Le prompt a changé et nous sommes maintenant l'utilisateur Linux test.

  - Affichez le contexte de sécurité du bash :

```
1    [test@SELinux ~]$ id -Z
2    unconfined_u:unconfined_r:unconfined_t
```

Malgré le changement d'utilisateur Linux, l'utilisateur SELinux n'a pas changé et est resté unconfined_u.

2.7.3 LABELING, MARQUAGE DES FICHIERS

Les labels de sécurité sont essentiels à l'application des règles allow et type transition. Si SELinux est désactivé et nous l'activons ou le mettons en mode permissif, tous les fichiers, dossiers, sockets, devices, etc. sur le système vont être labellisés ou autrement dit marqués avec un contexte de sécurité. Très souvent un administrateur système aura besoin de modifier les labels des objets pour pouvoir personnaliser son système.
Il existe différentes manières pour définir un contexte à un objet :

- Règles de marquage stockées dans un fichier
  - /etc/selinux/targeted/contexts/files/file_contexts
- Règles type_transition
- Utilisation de l'API SELinux dans une application pour faire une demande explicite d'un label

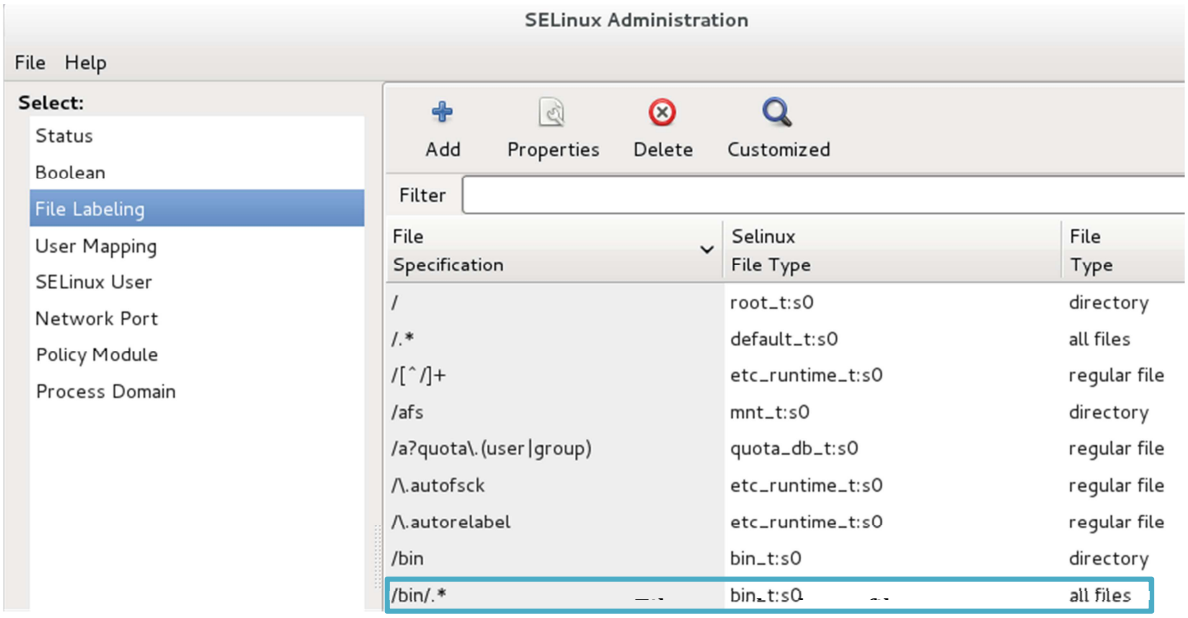
---

Mini labo

- **Observer le marquage d'un fichier selon les règles du fichier file_contexts :**
  - Relever le type du fichier exécutable pwd

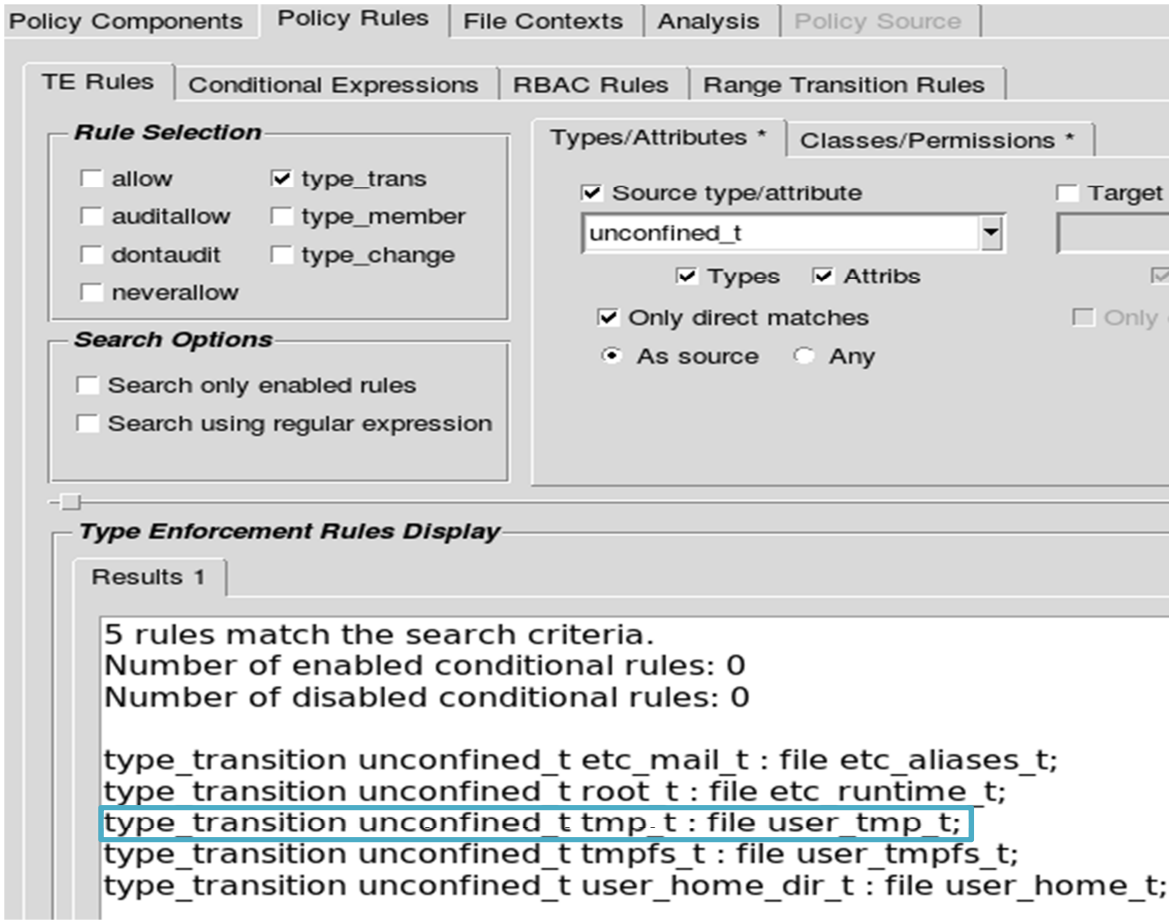
```
1    [root@SELinux ~]# ls -lZ /bin/pwd
2    -rwxr-x-r-x. root root system_u:object_r:bin_t    /bin/pwd
```

- Rechercher la règle qui a défini ce contexte à cet exécutable and SELinux management :



Dans SELinux Management nous pouvons relever la présence d'une ligne /bin/.* qui dicte un type bin_t au fichier contenu dans ce répertoire.

- **Observer un marquage d'un fichier selon une règle type transition :**
  - Rechercher une règle type_transition dans APOL et filtrer sur la classe file :



Cette règle dicte qu'à chaque fois que le domaine unconfined_t écrit un fichier dans un dossier marqué avec le type tmp_t, le fichier écrit dans ce dossier sera marqué avec le type user_tmp_t.

- Mettre en application la règle ci-dessus :
  - Contexte de sécurité du dossier tmp.

```
[root@SELinux /]# ls -dlZ /tmp
drwxrwxrwt. root root system_u:object_r:tmp_t        /tmp
```

Dans le cas où il n'existe pas de règle type_transition qui concerne ce dossier, si je crée un fichier dans le dossier tmp, le fichier héritera du label du dossier parent.

  - Créer un fichier dans tmp et relever le label de celui-ci :

```
[root@SELinux tmp]# touch test_label
[root@SELinux tmp]# ls -lZ test_label
-rw-r--r-- root root unconfined_u:object_r:user_tmp_t test_label
```

- **Donner la permission à un domaine d'accéder à une ressource en modifiant uniquement le marquage d'un fichier :**
  - Rechercher dans APOL les règles « allow » qui concerne le domaine semanage_t et filtrer sur la classe dossier :

```
allow semanage_t selinux_config_t : dir { ioctl read write create getattr
setattr...} ;
```

```
allow semanage_t selinux_var_lib_t : dir { ioctl read write create getattr
setattr ...} ;

allow semanage_t default_context_t : dir { ioctl read write getattr...} ;

allow semanage_t tmp_t : dir { ioctl read write getattr...} ;

allow semanage_t semanage_store_t : dir { ioctl read write create getattr
setattr...} ;

allow semanage_t semanage_tmp_t : dir { ioctl read write create getattr
setattr...} ;

allow semanage_t file_context_t : dir { ioctl read write getattr...} ;
```

Relever que semanage_t n'a pas le droit d'utiliser le dossier personnel (user_home_t) d'un utilisateur. Une fonctionnalité intéressante de semanage est l'export de la personnalisation réalisé sur la configuration du système dans un fichier.

- Créer un dossier qui devra contenir la sauvegarde de semanage dans le dossier personnel :

```
1   [root@SELinux ~]# mkdir /home/test/SELinuxConfig
2   [root@SELinux ~]# ls -ldZ /home/test/SELinuxConfig/
3   drwxr-xr-x. root root unconfined_u:object_r:user_home_t
    /home/test/SELinuxConfig/
```

- Essayer de sauvegarder la configuration de semanage dans le dossier créé :

```
1   [root@SELinux ~]# semanage -o /home/test/SELinuxConfig/conf.se
2   /usr/sbin/semanage: Permission denied
```

- Changer le type du dossier SELinuxConfig à un type que semanage y est autorisé d'écrire :

```
1 [root@SELinux ~]# chcon /home/test/SELinuxConfig/ -t
  semanage_store_t
2 [root@SELinux ~]# ls -ldZ /home/test/SELinuxConfig/
3 drwxr-xr-x. root root unconfined_u:object_r:semanage_store_t
  /home/test/SELinuxConfig/
```

- Relancer la sauvegarde dans SELinuxConfig :

```
1 [root@SELinux ~]# semanage –o /home/test/SELinuxConfig/conf.se
```

Chcon change le contexte d'un fichier ou dossier mais seulement jusqu'au prochain démarrage du système. Le fait que beaucoup de commandes SELinux effectuent des changements temporaires n'est pas pour augmenter le nombre de commandes et embêter l'utilisateur. C'est pour éviter qu'une fausse manipulation ou un changement pas assez réfléchi bloque tout le système. Pour changer le contexte d'un dossier d'une façon permanente et qui résiste au re-marquage et au redémarrage du système, utiliser la commande semanage qui elle écrie les changements dans le fichier file_contexts de SELinux.

2.7.4 ANALYSE DES FICHIERS LOGS

Tout refus SELinux est logué par l'AVC dans ces deux fichiers logs :

- /var/log/audit/audit.log
- /var/log/message

Tous refus AVC est enregistré dans le fichier log à moins qu'il existe une règle explicite qui demande de ne pas logué un refus donné.
Les logs sont très utiles pour que l'administrateur ait connaissance d'une tentative d'une application ou d'un utilisateur de transgresser les règles définies. Mais ils permettent aussi à comprendre pourquoi un refus a eu lieu et permettent aussi de générer des règles automatiquement pour autoriser l'action refusée. Attention, tout de même aux règles proposées, elles peuvent être très laxistes et créer des trous de sécurité. La génération de règles à partir des logues à l'aide de la commande audit2allow ne crée jamais de nouveau type mais réutilise les types définis dans le système. Un système sans SELinux est toujours plus fragile qu'avec même si la configuration de SELinux comporte des trous de sécurité. L'idéal c'est d'avoir un système avec SELinux et des règles sans trous de sécurité.

Mini labo

Le domaine user_t n'a pas le droit d'effectuer un ping ou un traceroute par défaut.

```
1   [test@SELinux Desktop]$ ping www.tdeig.ch
2   ping: icmp open socket: Permission denied
```

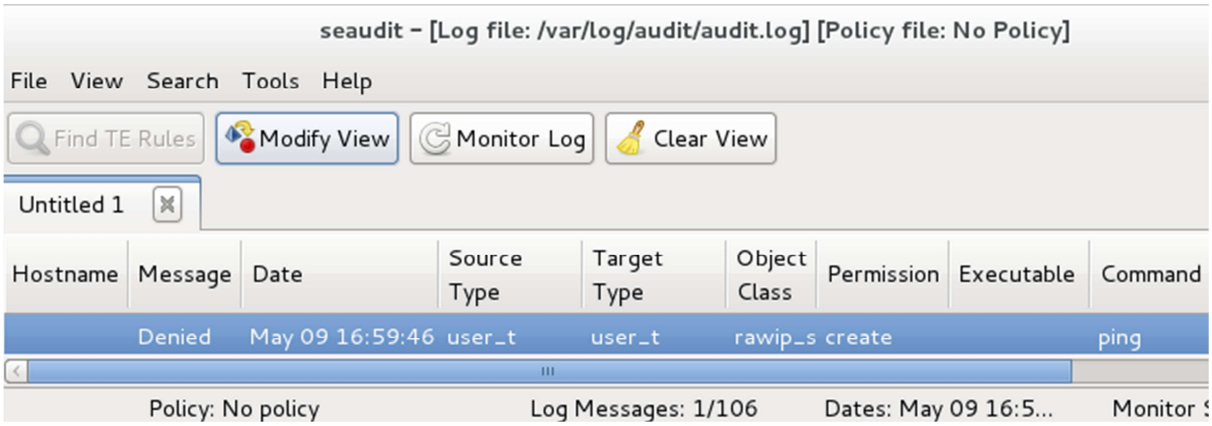
- **Rechercher le log généré par l'exécution de ping et obtenir une explication sur la cause du refus :**

```
1   [root@SELinux ~]# ausearch -x ping -ts today -m AVC | audit2why
2   type=AVC msg=audit(1336575586.310:764): avc:  denied  { create }
    for  pid=8821 comm="ping" scontext=user_u:user_r:user_t:s0
    tcontext=user_u:user_r:user_t:s0 tclass=rawip_socket
3   Was caused by:
4         Missing type enforcement (TE) allow rule.
5         You can use audit2allow to generate a loadable module to
    allow this access.
```

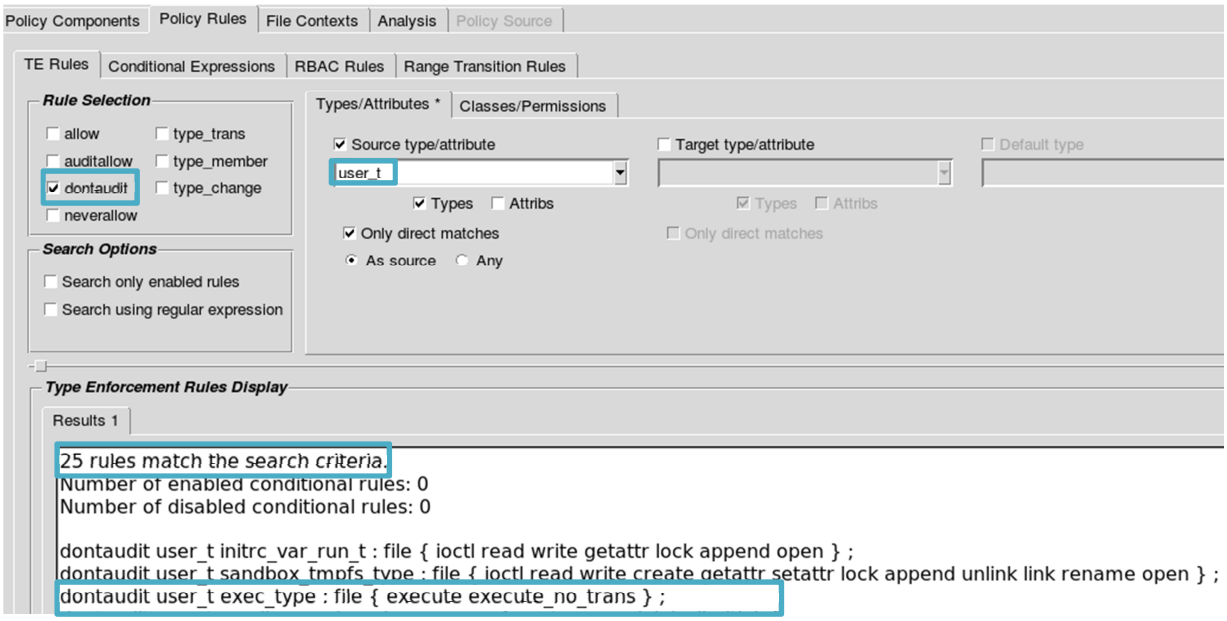
La commande ausearch permet de rechercher des logs contenus dans le fichier /var/log/audit/audit.log. La commande audit2why permet de donner une explication à une entrée de l'Access Vector Cache dans le fichier log.
Le message log ci-dessus nous indique la commande qui a causé le log. Le contexte source de l'exécutable. Quelle action a été interdite et dans quel contexte cible.

Partie facultative

Avec l'outil d'analyse de log, nous pouvons faire des filtres et des tris des logs. Voici à quoi ressemble ce même log dans seaudit :

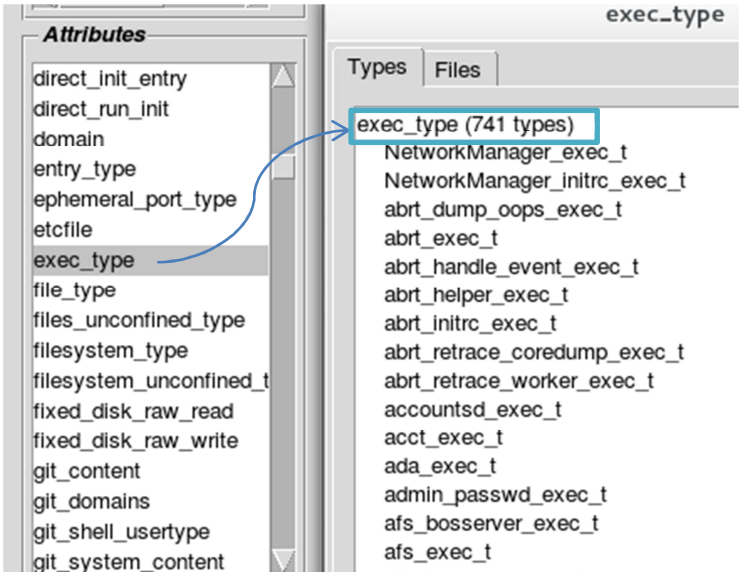


- **Rechercher le nombre de règles dontaudit dans APOL pour le domaine user_t :**



La règle qui a le plus d'impact sur inscription des logs de refus d'accès parmi les règles trouvées par APOL, voir la figure ci-dessus est celle-ci:

```
dontaudit user_t exec_type : file { execute execute_no_trans } ;
```



L'attribut exec_type regroupe 741 types qui sont les types des fichiers exécutables. Attention, les règles dontaudit sont une source de frustration. Par exemple, si refus d'accès à un exécutable atteint un utilisateur sans qu'il y ait une trace dans les logs, la compréhension du refus d'accès est moins évidente.

Une commande existe pour désactiver les règles dontaudit :

```
1    [root@SELinux ~]# semanage dontaudit off
```

Ou supprimer temporairement les règles dontaudit d'un module :

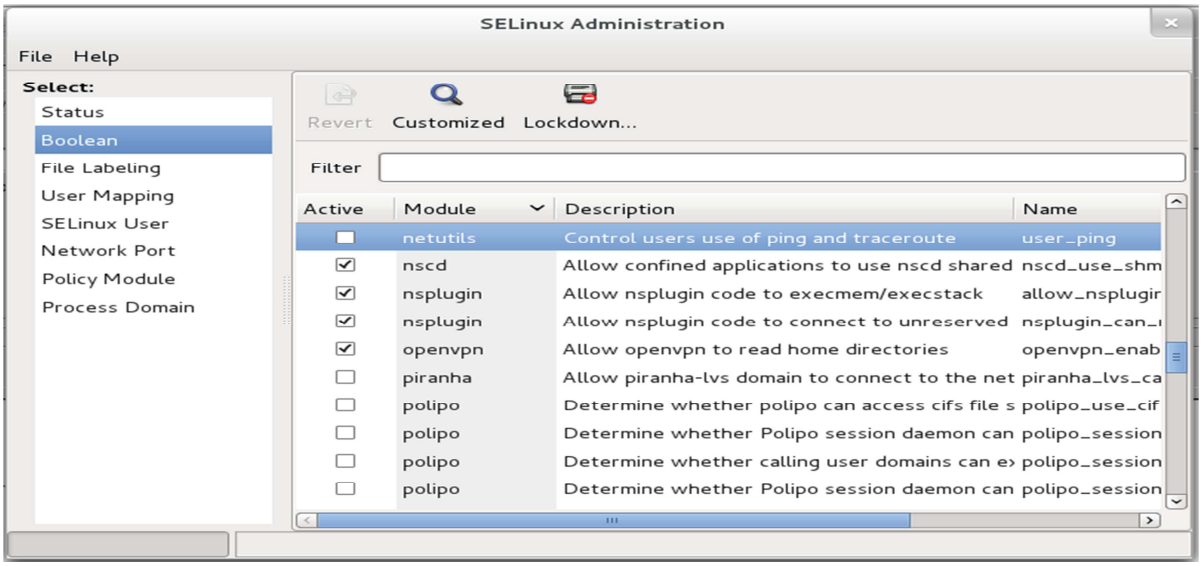
```
1    semodule -D
```

2.7.5 LES BOOLEENS

Les booléens servent à activer ou désactiver simplement et rapidement un certain nombre de règles dans un système en cours d'exécution. Ces booléens sont très souvent liés avec un programme ou un service. L'administration de SELinux se retrouve ainsi quelque peu simplifier.

Mini labo

Nous pouvons voir les booléens et une description de chacun d'eux dans l'outil SELinux Management.



- **Empêcher le domaine user_t à lancer des exécutables dans son dossier personnel :**

L'utilisateur user_u récupère un shell avec le domaine user_t qui a la permission de lancer des exécutables depuis son dossier personnel.

- Copier l'exécutable pwd dans le dossier personnel du user :

```
[root@SELinux ~]# cp /bin/pwd /home/test/
```

- Exécuter pwd copié depuis le compte test :

```
[test@SELinux ~]$ ./pwd
/home/test
```

- Changer l'état du booléen allow_user_exec_content à vrai :

```
[root@SELinux ~]# setsebool allow_user_exec_content true
```

- Vérifier que le booléen à bien changé d'état.

```
[root@SELinux ~]# getsebool -a | grep allow_user_exec_content
allow_user_exec_content --> on
```

- Ressayer d'exécuter la commande pwd :

```
[test@SELinux ~]$ ./pwd
bash: ./pwd: Permission denied
```

Partie facultative

- **Vérifier les entrées log causées par le lancement de pwd par l'utilisateur test :**

```
[root@SELinux bin]# ls -lZ pwd
-rwxr-xr-x. root root system_u:object_r:bin_t:s0       pwd
```

Dans la partie facultative du sous-chapitre analyse des fichiers log, j'ai expliqué qu'il existe une règle dontaudit pour tous les exécutables de base de Linux. Le type bin_t en faisait partie. Mais quand nous avons effectué la copie de l'exécutable pwd dans le dossier personnel de l'utilisateur test, pwd a hérité du contexte du dossier père.

```
[root@SELinux bin]# ls -lZ /home/test/pwd
-rwxr-xr-x. test test user_u:object_r:user_home_t   /home/test/pwd
```

Comme user_home_t ne fait pas parti de l'attribut exec_type, donc la règle dontaudit ne s'applique plus.

- Chercher la raison du refus :

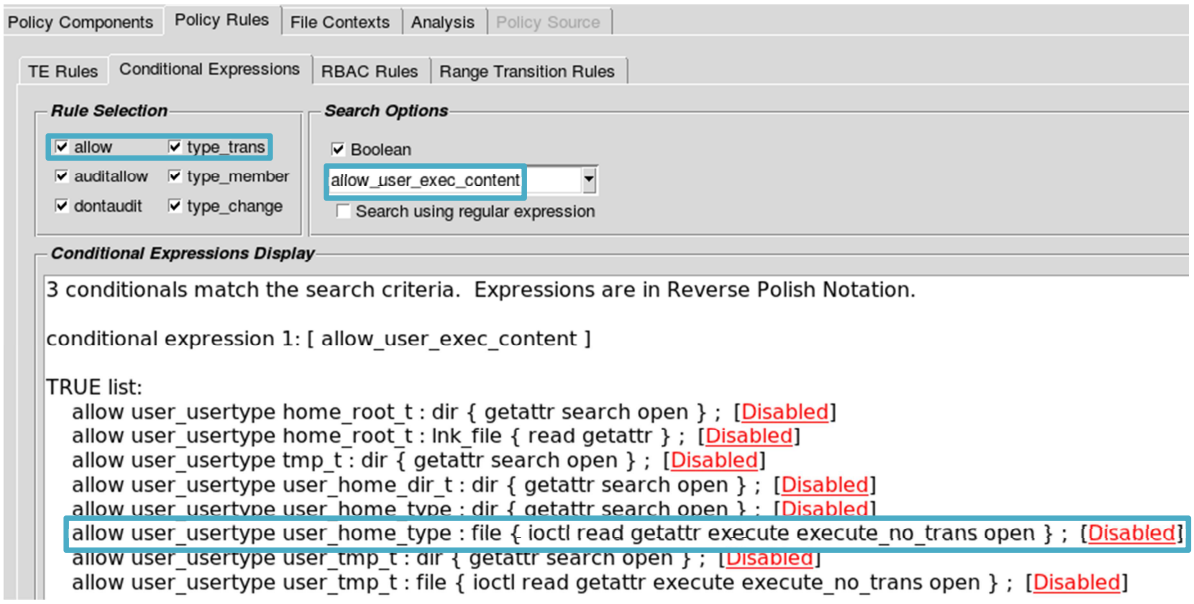
```
[root@SELinux bin]# ausearch -o user_home_t -ts today -m AVC
time->Thu May 10 11:15:42 2012
type=SYSCALL msg=audit(1336641342.828:547): arch=c000003e
syscall=59 success=no exit=-13 a0=1c1f130 a1=1de19f0 a2=1ded4e0
a3=10 items=0 ppid=4511 pid=5353 auid=1001 uid=1001 gid=1001
euid=1001 suid=1001 fsuid=1001 egid=1001 sgid=1001 fsgid=1001
tty=pts1 ses=4 comm="bash" exe="/bin/bash"
subj=user_u:user_r:user_t:s0 key=(null)
type=AVC msg=audit(1336641342.828:547): avc:  denied  { execute }
for  pid=5353 comm="bash" name="pwd" dev="dm-2" ino=787557
scontext=user_u:user_r:user_t:s0
tcontext=user_u:object_r:user_home_t:s0 tclass=file
```

- Renvoyer le log vers audit2allow pour trouver une solution :

```
[root@SELinux bin]# ausearch -o user_home_t -ts today -m AVC |
audit2allow
#============= user_t ==============
#!!!! This avc can be allowed using the boolean
'allow_user_exec_content'

allow user_t user_home_t:file execute;
```

Pour donner le pouvoir d'exécution au domaine user_t, audit2allow me suggère de changer l'état du booléen allow_user_exec_content.

- **Chercher dans APOL les règles liées avec un booléen :**
  - Rechercher les règles TE impliquées dans le booléen allow_user_exec_content :



- **Autoriser le domaine user_t à effectuer des ping :**

Il existe un booléen nommé user_ping qui permet de donner le droit de pinger au domaine user_t.

Ce booléen permet d'activer les règles suivantes et d'autres :

```
allow user_t ping_exec_t : file { read getattr execute open } ;  [Disabled]

type_transition user_t ping_exec_t : process ping_t;  [Disabled]
```

- Changer l'état du booléen à vrai de façon permanente :

```
[root@SELinux ~]# setsebool -P user_ping true
```

- Vérification que le booléen à bien changé d'état.

```
[root@SELinux ~]# getsebool -a | grep user_ping
user_ping --> on
```

Désormais le domaine user_t peut effectuer des ping et des traceroute.

```
[test@SELinux Desktop]$ ping www.tdeig.ch
PING www.tdeig.ch (129.194.184.80) 56(84) bytes of data.
64 bytes from 129.194.184.80: icmp_req=1 ttl=63 time=0.398 ms
```

2.7.6 LES MODULES

Les modules permettent d'étendre les règles d'un système. Il existe trois extensions de fichier lors de l'écriture d'un module :

- module.te
  - contient les règles TE
- module.fc
  - contient les contextes de fichiers à appliquer aux objets
- module.if
  - contient des interfaces pour permettre la réutilisation des règles

Une fois compilé, l'extension module.pp est donnée au fichier binaire généré.

Les règles qui s'appliquent à une application et la restreignent sont présentes soit :

- Dans le système pour les applications utilisées le plus couramment (serveur httpd, dns, ftp,…) et aux applications critiques (passwd, semanage, …)
- Fourni avec l'application et installé en même temps que celle-ci (exemple FreeIPA)
- Sur le web développées par la communauté du libre

Le souci c'est lorsqu'une application n'a pas de règles qui lui sont associées, l'application héritera ou restera dans le domaine de l'utilisateur qu'il a lancé. Par défaut c'est le domaine unconfined_t qui est associé aux utilisateurs d'où l'intérêt de restreindre les utilisateurs à un domaine user_t. L'objectif de restreindre une application est d'éviter qu'un hacker accède au système en profitant d'une faille dans celle-ci et d'éviter que cette application touche à des données ou récupère des données confidentielles.

Mini labo

- **Générer un module à l'aide d'audit2allow**

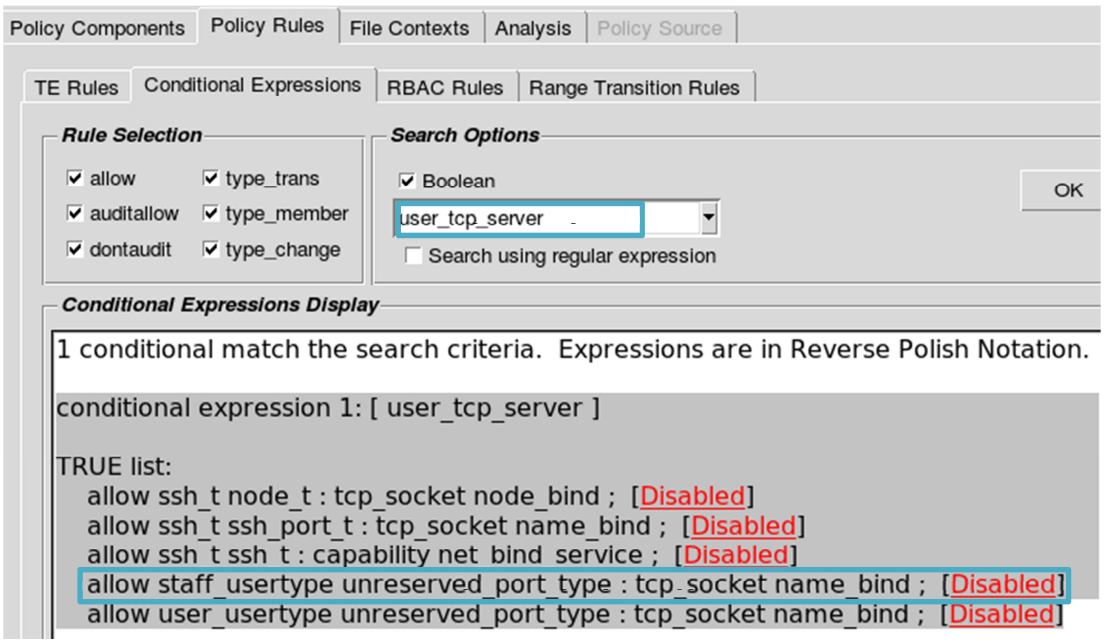
Dans ce projet j'utilise Dropbox pour partager et synchroniser mes fichiers avec le professeur. Dès que j'ai restreint mon compte test au domaine user_t, j'ai à chaque démarrage des messages log de SELinux qui empêche Dropbox de bind un socket tcp.

o Chercher une solution avec audit2allow :

Audit2allow nous retourne un booléen ou une règle qui peux résoudre le problème.

```
[root@SELinux ~]# ausearch -m AVC -x dropbox -ts today |
audit2allow
#============= user_t ==============
#!!!! This avc can be allowed using one of the these booleans:
#      user_tcp_server, allow_ypbind

allow user_t unreserved_port_t:tcp_socket name_bind;
```

Il existe un booléen user_tcp_server pour permettre cette action.



Les règles activées par ce booléen active aussi pour staff_usertype la possibilité de bind un tcp port. Si je ne veux pas donner ce droit à staff_usertype, je ne dois pas activer ce booléen. Je veux juste que user_t puisse bind un tcp socket dans le domaine unreserved_port.

Audit2allow a écrit automatiquement la règle qui résout le problème qui bloque Dropbox d'après les logs. L'option -M de la commande audit2allow permet de générer un module qui résout le problème.

o Générer un module pour autoriser dropbox à bind un tcp socket :

```
[root@SELinux ~]# ausearch -m AVC -x dropbox -ts today |
audit2allow -M dropbox
******************** IMPORTANT ***********************
To make this policy package active, execute:
semodule -i dropbox.pp
```

- Explorer les fichiers générés :

```
[root@SELinux ~]# ls -l
-rw-r--r--.  1 root   root    960 May 11 14:23 dropbox.pp
-rw-r--r--.  1 root   root    293 May 11 14:23 dropbox.te
```

Il y a deux fichiers crée. Un fichier généré avec les règles au format lisible (source) :

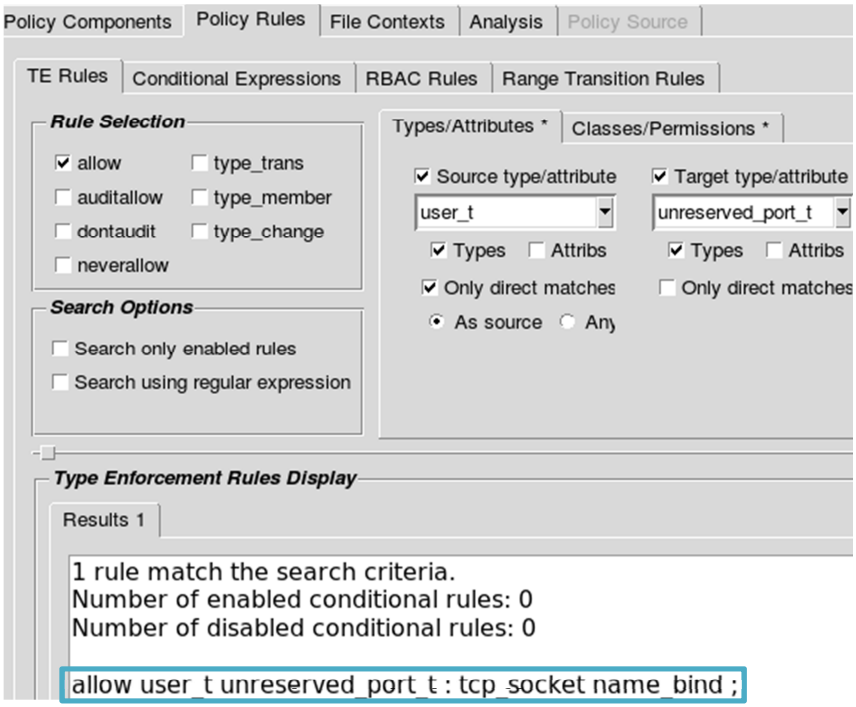
```
[root@SELinux ~]# cat dropbox.te
module dropbox 1.0;
require {
        type unreserved_port_t;
        type user_t;
        class tcp_socket name_bind;
}
#============= user_t ==============
#!!!! This avc can be allowed using one of the these booleans:
#     user_tcp_server, allow_ypbind
allow user_t unreserved_port_t:tcp_socket name_bind;
```

Le fichier dropbox.pp contient le module compilé.

- Installer le module dropbox.pp :

```
[root@SELinux ~]# semodule -i dropbox.pp
```

- Recharger le fichier /etc/selinux/targeted/policy/policy.[ver] dans APOL et rechercher la règle qui a été ajouté par l'installation du module :



- **Installer un paquetage qui contient des règles SELinux**
  - Installer la suite FreeIPA :

```
[root@SELinux ~]# yum install freeipa-server
```

- Lister les modules installés :

```
[root@SELinux ~]# semodule -l | grep ipa
ipa_dogtag 1.4
```

```
3    ipa_httpd  1.2
4    ipa_kpasswd1.0
```

Ces modules ont été installés automatiquement lors de l'installation de FreeIPA.

- **Générer un module pour restreindre une application :**
  - Désinstaller le module dropbox installé auparavant :

```
1    [root@SELinux ~]# semodule –r dropbox
```

  - Relever le domaine du processus dropbox :

```
1    [test@SELinux ~]$ ps -eZ | grep dropbox
2    unconfined_u:unconfined_r:unconfined_t 1716 ? 00:00:06 dropbox
3    user_u:user_r:user_t 3918 ?        00:00:13 dropbox
```

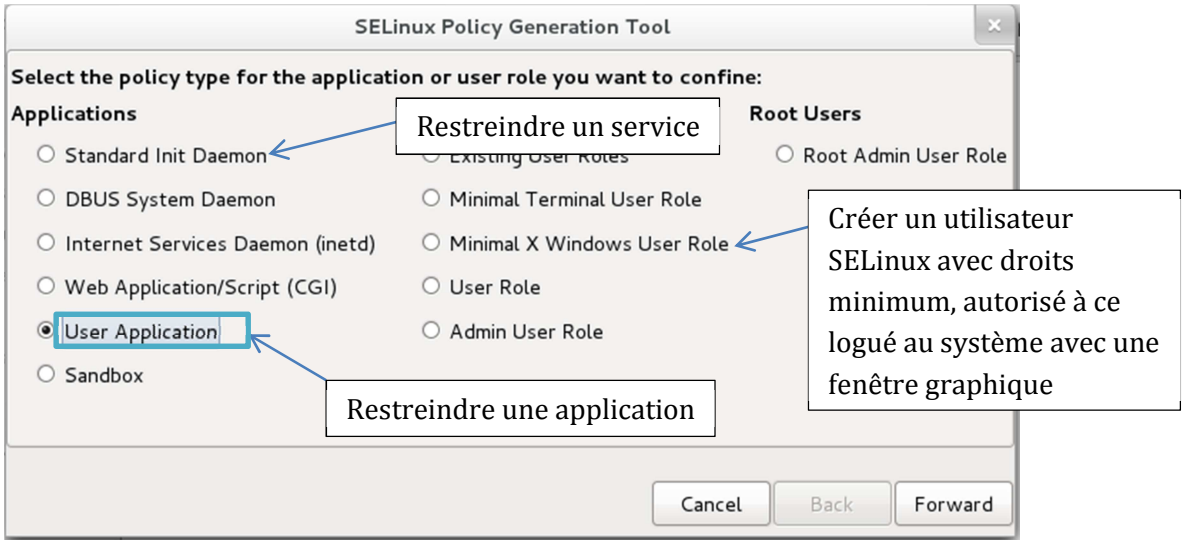
  - Fermer toutes les instances de dropbox.
  - Chercher les principaux fichiers et dossier de dropbox :

```
1    [root@SELinux ~]# find / -name "*dropbox*"
2    ...
3    /usr/bin/dropbox
4    ...
5    /home/test/.dropbox-dist
6    ...
7    /home/test/.config/autostart/dropbox.desktop
8    /home/test/.dropbox
9    ...
```

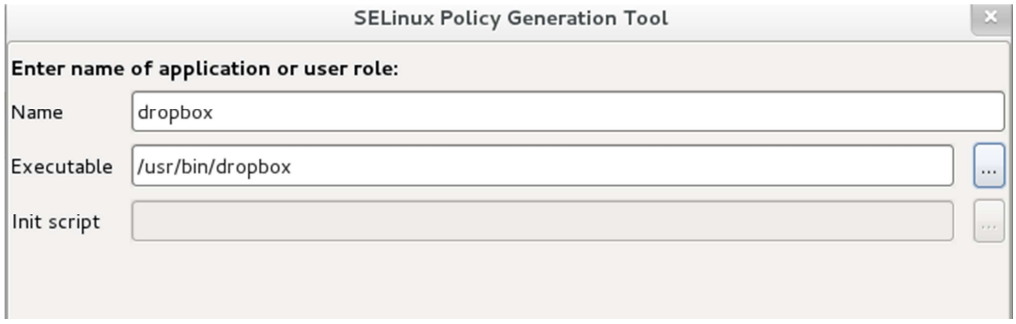
  - Démarrer polgengui avec un terminal root :

```
1    [root@SELinux ~]# selinux-polgengui
```

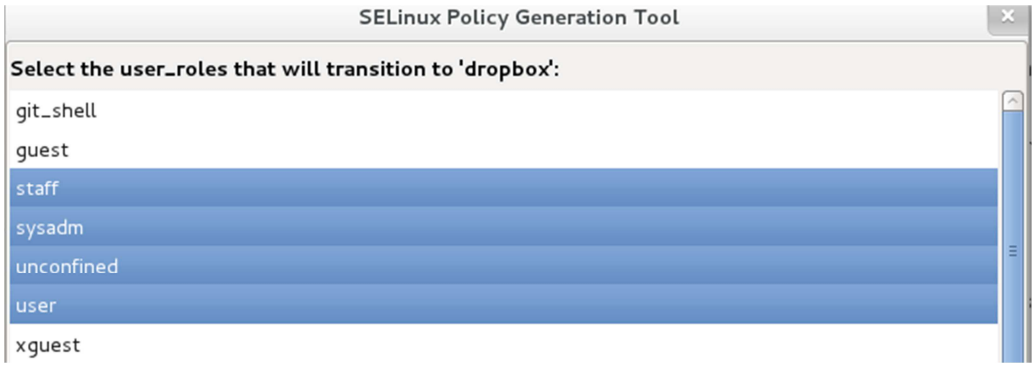
  - Sélectionner User Application :



  - Ajouter le nom du module et le chemin de l'exécutable à restreindre :

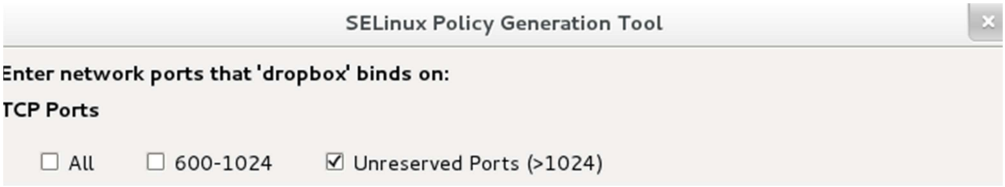


- Sélectionner les utilisateurs - rôles qui pourront transiter au domaine de l'application à restreindre :

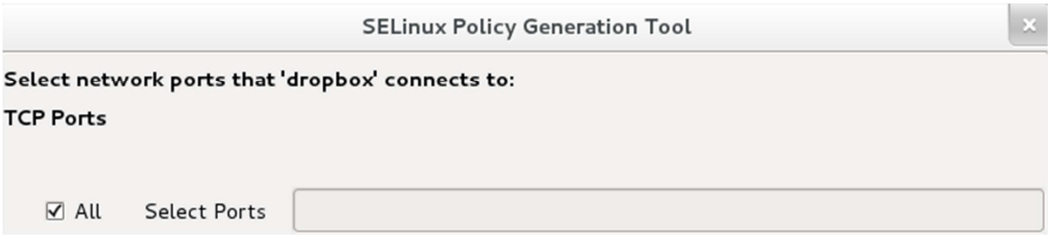


Dropbox est une application qui utilise le réseau, j'imagine qu'elle aura besoin de bind des ports non réservé.

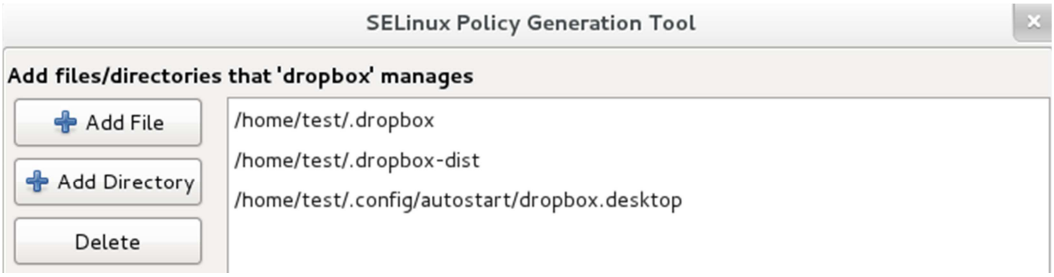
- Sélectionner les ports de type unreserved, pour autoriser l'application dropbox à utiliser la fonction bind() sur ceux-ci :



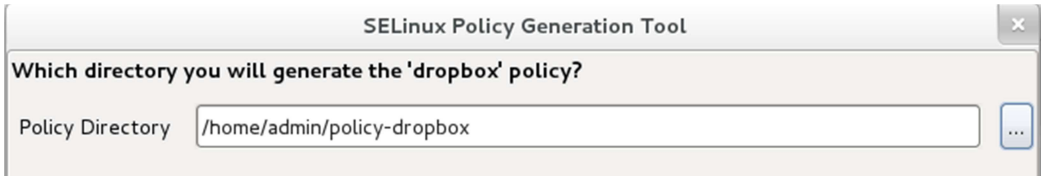
- Sélectionner les ports de type unreserved, pour autoriser l'application dropbox à utiliser la fonction connect() sur ceux-ci :



- Ajouter les fichiers et dossiers principaux de dropbox :



- Indiquer le dossier ou sera stocké le module généré :



- Afficher les fichiers générés :

```
[root@SELinux ~]# ls –l /home/admin/policy-dropbox
-rw-r--r--. 1 root root  326 Jun 20 16:59 dropbox.fc
-rw-r--r--. 1 root root 3308 Jun 20 16:59 dropbox.if
-rwxr-x---. 1 root root 1418 Jun 20 16:59 dropbox.sh
-rw-r--r--. 1 root root 1539 Jun 20 16:59 dropbox.te
```

- Installer les en-têtes et les librairies utilisées pour la compilation de SELinux :

```
[root@SELinux ~]# yum install libselinux-devel
```

- Installer le module généré :

```
[root@SELinux ~]# cd /home/admin/policy-dropbox
[root@SELinux policy-dropbox]# ./dropbox.sh
Building and Loading Policy
+ make -f /usr/share/selinux/devel/Makefile
Compiling targeted dropbox module
/usr/bin/checkmodule:  loading policy configuration from tmp/dropbox.tmp
/usr/bin/checkmodule:  policy configuration loaded
/usr/bin/checkmodule:  writing binary representation (version 13) to tmp/dropbox.mod
Creating targeted dropbox.pp policy package
rm tmp/dropbox.mod.fc tmp/dropbox.mod
+ /usr/sbin/semodule -i dropbox.pp
+ /sbin/restorecon -F -R -v /usr/bin/dropbox
/sbin/restorecon reset /usr/bin/dropbox context system_u:object_r:bin_t:s0->system_u:object_r:dropbox_exec_t:s0
+ /sbin/restorecon -F -R -v /home/test/.config/autostart/dropbox.desktop
+ /sbin/restorecon -F -R -v /usr/bin/dropbox
+ /sbin/restorecon -F -R -v /home/test/.dropbox
+ /sbin/restorecon -F -R -v /home/test/.dropbox-dist
```

- Mettre le domaine dropbox_t généré en permissive :

```
[root@SELinux ~]# semanage permissive -a dropbox_t
```

- Utiliser les fonctions de l'application afin de générer le plus de log AVC possible. Puis mettre à jour les fichiers du module générés :

```
[root@SELinux policy-dropbox]# ./dropbox.sh –update
Found avc's to update policy with

require {
        type staff_t;
        type staff_dbusd_t;
        type home_root_t;
        ...
        class process { siginh execstack noatsecure execmem rlimitinh };
        class unix_stream_socket connectto;
        class chr_file { read ioctl write getattr open };
        class shm { write unix_read unix_write read destroy create };
```

```
13        class file { rename execute setattr read lock create
    getattr execute_no_trans write ioctl unlink open append };
14        ...
15        class unix_dgram_socket { create ioctl };
16        class udp_socket { name_bind node_bind };
17        class dir { search setattr read write getattr remove_name
    open add_name };
18   }
19
20   #============= dropbox_t ==============
21   allow dropbox_t bin_t:file { execute getattr read open ioctl
    execute_no_trans };
22   allow dropbox_t bin_t:lnk_file { read getattr };
23   ...
24
25   #============= staff_t ==============
26   allow staff_t dropbox_t:process { siginh rlimitinh noatsecure };
27   allow staff_t dropbox_t:unix_stream_socket connectto;
28   ...
29   Do you want these changes added to policy [y/n]?
30   Y
```

- Étudier les règles proposés et cliquer sur [y] pour modifier les fichiers générés auparavant, puis personnaliser les règles dans les fichiers et les contextes de fichier.
- Enlever le domaine dropbox _t de la liste des permissives domaines :

```
1    [root@SELinux ~]# semanage permissive -d dropbox_t
```

- Réinstaller le module dropbox :

```
1    [root@SELinux policy-dropbox]# ./dropbox.sh
```

Relancer la mise à jour des fichiers générés à partir des messages AVC à plusieurs reprises si besoin. Une fois que le module est personnalisé, nous pouvons supprimer le domaine dropbox de la liste des permissives domaines.

Cette méthode est rapide et facile à utiliser pour restreindre une application. Les règles générées à partir des logs peuvent être trop laxiste. La difficulté est de connaître les besoins de l'application pour garder les fonctionnalités de celle-ci tout en appliquant le principe du moindre privilège. Polgen crée trois types lors de la restriction d'une application :

- Un type pour le fichier de l'exécutable de l'application (dropbox_exec_t)
- Un type pour le domaine dans lequel sera restreinte l'application (dropbox_t)
- Un type pour les fichiers qui peuvent être accédé par l'application (dropbox_rw_t)

Pour bien isoler et contrôles les accès de l'application, il faudra crée plus de types. Mais l'avantage d'utiliser polgen est d'avoir un canevas dans lequel nous ajouterons par la suite d'autres types et d'autres règles.

2.8 ROLE BASED CONTROL ACCESS

Dans la plupart des systèmes d'exploitation, la fonctionnalité de contrôle d'accès basé sur le rôle est centrale à l'allocation d'accès aux utilisateurs soit directement soit à travers une forme de groupe. SELinux n'alloue pas d'accès directement à travers RBAC. Dans SELinux, l'allocation d'accès aux ressources provient du type enforcement. À chaque rôle correspond une liste de domaines accessible. Une transition d'un domaine à un autre est possible pour un processus uniquement si le domaine de destination est listé dans un de ces rôles.

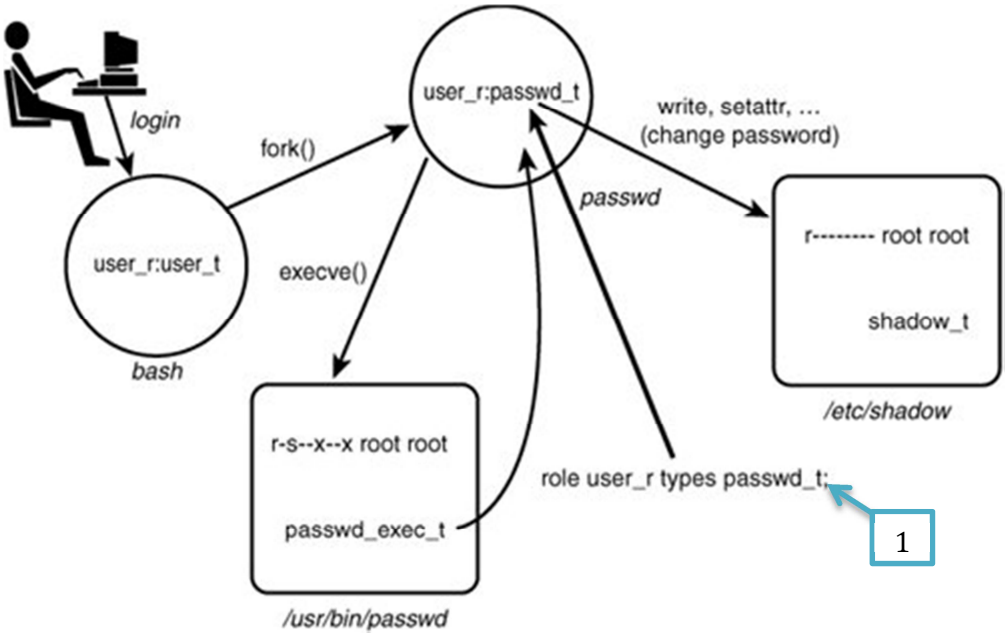


FIGURE 18: ROLE BASED ACCESS CONTROL, EXEMPLE PASSWD

Dans la figure ci-dessus, la transition vers le domaine passwd_t est possible non seulement parce qu'il y a les règles TE, mais parce qu'il y a le type ou le domaine passwd_t qui est listé dans le rôle user_r ce qui autorise la transition vers ce domaine [1].
Un utilisateur SELinux peut lui être associé plusieurs rôles avec une relation (1:n).

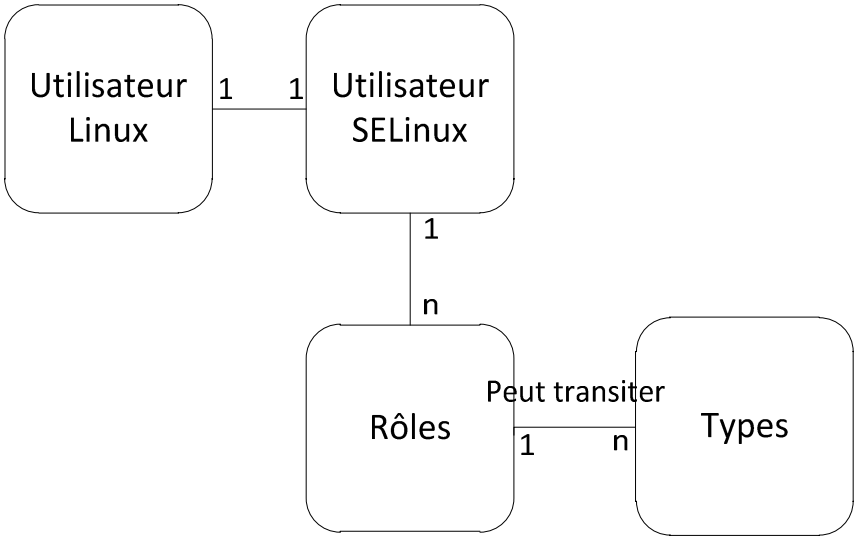


FIGURE 19: UTILISATEUR SELINUX - ROLES - TYPES

Les rôles sont utiles pour séparer les responsabilités des utilisateurs du système. Par exemple, nous pouvons avoir un rôle webadmin charger d'administrer le serveur web et un rôle bddadmin pour administrer les bases de données. La transition d'un rôle vers un autre est réglementée.

- Une transition peut devenir effective lorsqu'un programme est lancé et qu'une règle SELinux de transition de rôle existe pour ce programme.

```
role_transition staff_r su_exec_t sysadm_r ;
```

Lorsqu'un utilisateur avec le rôle staff_r lance un exécutable marqué avec le type su_exec_t, le processus démarré obtient un rôle sysadm_r

- Une transition de rôle peut-être demandée par une application à l'aide de l'API SELinux. La commande newrole permet ce type de transition.

- Une transition de rôle peut être faite en modifiant le fichier sudoers.

Mini labo

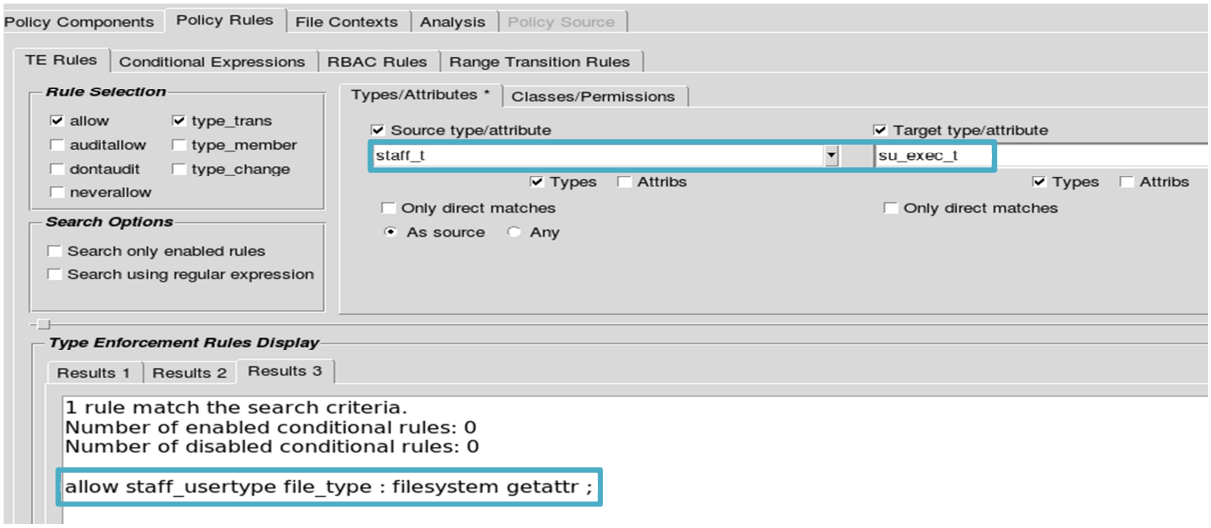
- **Afficher les utilisateurs SELinux avec les rôles qu'ils leurs sont associés :**

```
[root@SELinux ~]# semanage user -l

                Labeling
SELinux User    Prefix      SELinux Roles

git_shell_u     user        git_shell_r
guest_u         user        guest_r
root            user        staff_r sysadm_r system_r unconfined_r
staff_u         user        staff_r sysadm_r system_r unconfined_r
```

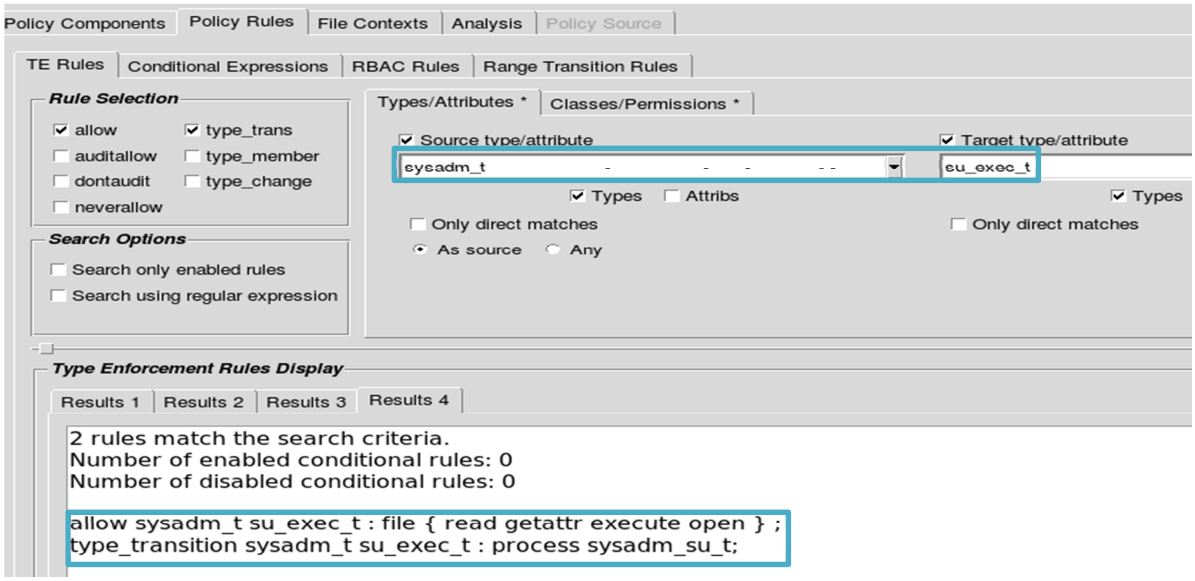
```
10   sysadm_u        user         sysadm_r
11   system_u        user         system_r unconfined_r
12   unconfined_u    user         system_r unconfined_r
13   user_u          user         user_r
14   xguest_u        user         xguest_r
```

- **Transiter d'un rôle à un autre en modifiant le fichier sudoers :**

Le rôle staff_r n'a pas le droit de lancer la commande su. Vérification avec APOL, je relève le domaine du rôle staff_r et j'effectue la recherche :

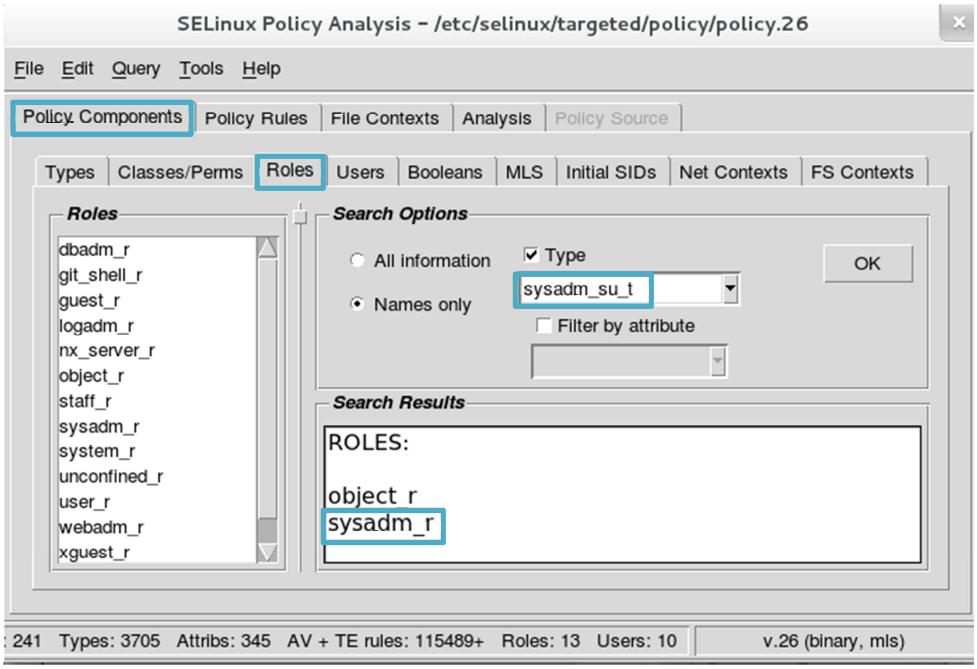


Le rôle sysadm_r a le droit de lancer la commande su, je filtre le résultat dans APOL avec la classe process, file et la permission execute :



Le domaine auquel nous transitons lorsque nous lançons la commande su depuis le domaine sysadm_t est nommé sysadm_su_t. Tous les domaines auxquels nous pouvons transiter sont inscrits dans le rôle. Ici nous transitons au domaine sysadm_su_t à partir du domaine sysadm_t.

Vérification dans la liste de rôle si ce domaine est listé dans le rôle sysadm_t :



Effectivement sysadm_su_t est présent dans le rôle sysadm_r.

- Associer l'utilisateur test à l'utilisateur SELinux staff_u :

```
1    [root@SELinux ~]# semanage login -a -s test_u test
2    [root@SELinux ~]# semanage login -l
3    Login Name                SELinux User
4    __default__               unconfined_u
5    root                      unconfined_u
6    system_u                  system_u
7    test                      staff_u
```

- Vérifier le contexte du bash :

```
1    [test@SELinux ~]# id -Z
2    staff_u:staff_r:staff_t
```

- Associer l'utilisateur test à l'utilisateur SELinux test_u :

```
3    [root@SELinux ~]# semanage login -a -s test_u test
4    [root@SELinux ~]# semanage login -l
5    Login Name                SELinux User
6    __default__               unconfined_u
7    root                      unconfined_u
8    system_u                  system_u
9    test                      test_u
```

- Vérifier le contexte du bash :

```
10   [test@SELinux ~]# id -Z
11   test_u:staff_r:staff_t
```

- Essayer de lancer la commande su :

```
1    [test@SELinux ~]# su
2    bash : su : command not found...
```

- Modifier le fichier /etc/sudoers[16] :

La commande visudo permet de modifier le fichier sudoers et de contrôler sa syntaxe. Il est fortement déconseillé de modifier le contenu de ce fichier à l'aide d'un éditeur de texte simple.

```
1    [root@SELinux ~]$ visudo
2    ## Sudoers allows particular users to run various commands as
3    ## the root user, without needing the root password.
4    ##
5    ## Examples are provided at the bottom of the file for
     collections
6    ## of related commands, which can then be delegated out to
     particular
7    ## users or groups.
8    ##
9    ## This file must be edited with the 'visudo' command.
10   ...
```

Je rajoute une ligne dans sudoers en dessous de la ligne root ALL=(ALL) ALL :

test ALL=(ALL) ROLE=sysadm_r TYPE=sysadm_t /bin/bash

Cette ligne signifie que l'utilisateur test peu exécuter la commande bash depuis n'importe quel terminal et agir comme n'importe quel utilisateur Linux. De plus lors de l'exécution de cette commande un changement de rôle devra se faire vers le rôle sysadm_r et un transit vers le domaine sysadm_t.

- Se connecter à nouveau avec l'utilisateur test et lancer la commande sudo dans un bash :

```
1    [test@SELinux ~]# sudo bash
2    [root@SELinux users]#
```

- Vérifier si la transition c'est bien faite :

```
1    [root@SELinux ~]# id -Z
2    test_u:sysadm_r:sysadm_t
```

Maintenant j'ai accès à la commande su qui n'était pas accessible avec le rôle staff_r.

- **Transition de rôle en utilisant la commande newrole :**
  - Installer newrole :

```
1    [root@SELinux ~]# yum –y install policycoreutils-newrole
2    [test@SELinux ~]# newrole –r sysadm_r
3    [test@SELinux ~]# id -Z
4    test_u:sysadm_r:sysadm_t
```

Partie facultative

- **Créer un utilisateur SELinux :**
  - Ajouter un utilisateur nommé test_u et lui associer les rôles staff_r et sysadm_r :

```
[root@SELinux ~]# semanage user -a -R "staff_r sysadm_r" test_u
[root@SELinux ~]# semanage user -l

                Labeling
SELinux User    Prefix      SELinux Roles

git_shell_u     user        git_shell_r
guest_u         user        guest_r
root            user        staff_r sysadm_r system_r unconfined_r
staff_u         user        staff_r sysadm_r system_r unconfined_r
sysadm_u        user        sysadm_r
system_u        user        system_r unconfined_r
test_u          user        staff_r sysadm_r
unconfined_u    user        system_r unconfined_r
user_u          user        user_r
xguest_u        user        xguest_r
```

Avant, nous avons créé l'utilisateur SELinux test_u en lui associant deux rôles. C'est une association de type (1:n). Pourquoi notre système associe le rôle staff_r et non le rôle sysadm_r par défaut. L'association du rôle est gérée par le dossier /etc/selinux/targeted/contexts/users.

  - Afficher le contenu de ce dossier :

```
[root@SELinux ~]# ls -l /etc/selinux/targeted/contexts/users
total 24
-rw-r--r--. 1 root root 253 Apr 24 13:21 guest_u
-rw-r--r--. 1 root root 389 Apr 24 13:21 root
-rw-r--r--. 1 root root 514 Apr 24 13:21 staff_u
-rw-r--r--. 1 root root 578 Apr 24 13:21 unconfined_u
-rw-r--r--. 1 root root 353 Apr 24 13:21 user_u
-rw-r--r--. 1 root root 307 Apr 24 13:21 xguest_u
```

  - Afficher le contexte de l'utilisateur SELinux guest_u :

```
[root@SELinux ~]# cat
/etc/selinux/targeted/contexts/users/guest_u
guest_r:guest_t:s0          guest_r:guest_t:s0
system_r:crond_t:s0         guest_r:guest_t:s0
system_r:initrc_su_t:s0     guest_r:guest_t:s0
system_r:local_login_t:s0   guest_r:guest_t:s0
system_r:remote_login_t:s0  guest_r:guest_t:s0
system_r:sshd_t:s0          guest_r:guest_t:s0
```

Nous pouvons d'abord remarquer le contexte de sécurité du login local. L'association se trouve à droite avec le rôle guest_r et le domaine guest_t. Nous pouvons prendre exemple sur celui-ci pour créer un contexte pour l'utilisateur test_u :

```
[root@SELinux ~]# cd /etc/selinux/targeted/contexts/users/
```

Créer un fichier test_u dans le dossier users et ajouter la ligne sysadm_r:sysadm_t:s0 sysadm_r:sysadm_t:s0 > test_u :

```
[root@SELinux users]# echo sysadm_r:sysadm_t:s0
sysadm_r:sysadm_t:s0 > test_u
```

Au login local, nous voulons que le shell de l'utilisateur test soi lancé avec le rôle sysadm_r et avec le domaine sysadm_t :

```
1    [root@SELinux users]# echo system_r:local_login_t:s0
     sysadm_r:sysadm_t:s0 >> test_u
```

Pour le login distant, nous avons la possibilité de personnaliser le contexte de sécurité du shell obtenu. Par exemple, pour plus de sécurité, assigner un contexte de sécurité avec le rôle staff_r pour avoir moins de droit que le rôle sysadm_r lors d'une connexion distante :

```
1    [root@SELinux users]# echo system_r:remote_login_t:s0
     staff_r:staff_t:s0 >> test_u
2    [test@SELinux users]# id -Z
3    test_u:sysadm_r:sysadm_t
```

Avec, c'est modification nous avons le rôle sysadm_r après la connexion local. Vérifier si nous avons désormais le droit d'exécuter la commande su :

```
1    [test@SELinux ~]$ su
2    Password:
3    [root@SELinux ~]#
```

2.9 MULTI-LEVEL SECURITY / MULTI-CATEGORY SECURITY[17]

Le Type Enforcement permet de protéger l'intégrité des données du système. Dans le passé, il a même servi à garantir la confidentialité des données du système. Mais, dans le but d'améliorer la clarté des règles, c'est préférable de séparer la protection de l'intégrité des données de ceux qui se rattache à la confidentialité des données. Ainsi, la partie qui contient les règles d'autorisation d'accès entre domaine et type, allow rules, est focalisée sur la protection de l'intégrité du système.
Le but du mécanisme *Multi-Category Security (MCS)* est de protégé la confidentialité des données. MCS est une fonctionnalité supplémentaire pour prévenir les fuites accidentelles ou délibérées de données secrètes. Elle est basée sur le mécanisme Multi-Level Security. MLS est compliqué à utiliser, il est adapté surtout à une utilisation militaire donc elle ne concerne pas la problématique de la plupart des utilisateurs. C'est pour cette raison que MCS est utilisé de base dans Fedora 16 et dans les distributions Red Hat. MCS utilise les attribues MLS, les niveaux de sécurité et les catégories de sécurité, dans son propre modèle pour une utilisation plus commune. Du fait que MCS ne contrôle que la permission d'accès aux fichiers uniquement, des fuites de données peuvent avoir lieu

dans le cas où deux processus collabore et transfert des informations sensé être confidentiel à travers des sockets ou des pipes entre eux. La conception de MCS a été faite ainsi pour éviter de restreindre toutes formes de communications interprocessus (IPC). La restriction des IPC pourrait bloquer beaucoup de programmes et rendre le système difficile à utiliser. Le mécanisme MLS est conçu pour restreindre les IPC et c'est une des raisons de sa complexité d'utilisation. Pour imager, imaginons qu'un processus est une personne, MCS ne peux pas interdire à une personne mal intentionnée qui a accès aux données confidentielles de les partager à une autre personne qui n'y a pas le droit d'accès. C'est d'ailleurs ce qui s'est passé lors du vol de données bancaires confidentielles Suisse et qui ont été vendu au fisc allemand en 2012[18]. La sécurisation d'un système informatique ne pourra jamais empêcher ce genre d'incident mais elle peut restreindre la diffusion de l'information aux personnes concernées tout au plus.

Dès l'activation de MLS/MCS les attributs MLS sont ajoutés à chaque contexte de sécurité :

Utilisateur:Rôle:Type:**MLS/MCS**

Nous allons nommer labels MCS les attributs MLS des fichiers dans le cadre de l'utilisation de MCS. Dans MCS, chaque utilisateur lui est assigné une ou plusieurs catégories. Il y a qu'un seul niveau de sécurité utilisé ce qui nous affranchi de la règle « no read up and no write down ». MCS a 1024 catégories qui peuvent être assigné à des processus ou fichiers.

Un fichier peut avoir un label MCS s0:c0.c3. Le s0 dans le label MCS n'est pas utilisé car c'est le niveau de sensibilité liée avec le mécanisme MLS. Dans MCS, nous n'utilisons qu'un seul niveau de sensibilité, le niveau s0. La partie qui est utilisée par MCS est c0.c3 qui signifie un ensemble de catégories allant de la catégorie 0 à la catégorie 3 incluse.

Un processus peut avoir un label MCS à deux niveaux. Un haut et un bas niveau. Le haut niveau est un super-ensemble qui contient l'ensemble bas niveau. Par exemple un processus peut avoir un label s0-s0:c0.c5, ce qui signifie que le bas niveau est la sensibilité s0 sans catégories et le haut niveau c'est la sensibilité s0 avec les catégories allant de la catégorie c0 à la catégorie c5. Un ensemble de catégories disjointes est aussi autorisé. Le label s0:c1,c3 signifie la catégorie c1 et la catégorie c1 sont dans le label. Le haut niveau détermine le droit d'accès aux fichiers. Le bas niveau détermine le niveau par défaut des fichiers crée par le processus. Pour accéder un fichier, un processus doit avoir un label MCS qui domine le label MCS du fichier à accéder. Voir annexe A.7 pour un exemple pratique.

---

[18] Article sur la fuite de données bancaire en Suisse :
http://www.tdg.ch/suisse/interet-fisc-allemand-cd-voles-faiblit/story/18303721?track

3 DIFFICULTES RENCONTREES

La documentation sur internet est trop redondante et en anglais. J'ai lu plusieurs centaines de pages afin de trouver des nouvelles informations.

Pour effectuer des analyses avancées dans APOL, par exemple analyse des transitions de domaines, il faut charger un fichier de permissions. J'ai cherché un moment avant de trouver comment récupérer ce fichier. Au final c'est un fichier qui vient dans les setools mais APOL ne le récupère pas automatiquement. (/usr/share/setools-3.3/apol_perm_mapping_ver24).

La recherche de règle pour comprendre pourquoi un accès est refusé n'est souvent pas triviale, il faut avoir bien compris le concept pour arriver à définir une méthodologie de recherche.

Pendant mes testes de SELinux, et à force de modifier la configuration du système, j'ai bloqué mon système. Il n'était plus possible de se connecter au système car j'avais supprimé plein de règles pour renforcer le système. Par conséquent l'utilisation du système était devenue trop restrictive. Pour débloquer la situation, j'ai désactivé SELinux dans grub (gestionnaire d'amorçage) et réinstallé la targeted policy.

Certains outils d'analyse des règles, par exemple sesearch, sont mal documentés. Après avoir cherché un moment la définition de certains

éléments affichés j'ai finalement récupéré le code source de l'outil pour comprendre leurs définitions.

4 CONCLUSION

Les objectifs de ce travail ont été atteints. Je réalise que les règles de référence et les extensions SELinux sont écrites uniquement par quelques experts. L'écriture de règles est très complexe car elle doit être réalisée en connaissant les ressources et les éléments utilisées dans le logiciel à restreindre, connaître le contenu de la reference policy pour réutiliser l'existant et faire attention à ne pas créer un trou de sécurité dans le système.

L'étude de l'écriture de règles n'était pas dans l'énoncé du travail, mais j'ai tout de même passé du temps afin de comprendre de quoi elles sont constituées. Comme SELinux est riche en fonctionnalités, il fera l'objet d'étude ultérieure dans un projet exploratoire de la HES-SO. Ce travail a été la préparation et la collecte de documents et de connaissance permettant d'aller plus loin dans une analyse prochaine plus ciblé sur SELinux. La documentation sur SELinux est volumineuse et extrêmement redondante. Il m'a fallu beaucoup d'effort pour faire le tri et sélectionner les textes les plus consistants. J'ai été à plusieurs reprises dans le code source de commandes pour chercher la signification des éléments affichés à l'écran car la documentation était encore malheureusement incomplète.

SELinux a une faiblesse, par exemple un serveur FTP qui partage des fichiers et donne accès à ces fichiers en écriture. Dans le cas où ce serveur

est attaqué et qu'une faille existe dans ce logiciel, le hacker aura accès en écriture aux fichiers partagés ! Par contre, il aura uniquement accès aux ressources accessibles par le serveur FTP et à rien d'autre.

J'ai été convaincu par la plus-value sécuritaire apporté par SELinux à un système d'exploitation. Dorénavant, j'imagine difficilement un serveur en production tourné sans ce module. Comme dit au début de ce document, vérifier le code des applications qui tournent sur un système est difficilement réalisable, par contre appliquer le principe du moindre privilège sur ces applications nous garantit en grande partie que les applications ne sont plus libre de faire ce que bon lui semble avec les données du système.

Ce travail m'a permis de travailler sur des mécanismes de contrôle d'accès non étudiés à HEPIA. Vers la fin du projet, j'ai eu des bons contacts avec la communauté SELinux à travers le canal IRC. Je regrette de ne pas avoir fait usage dès le début du projet.

Khaled BASBOUS
Genève, le 22 juin 2012

5 LIENS & REFERENCES

- **Pour comprendre le concept et la syntaxe des règles j'ai lu l'excellent livre :**

  Livre « SELinux by Example » *de Frank MAYER, Karl MACMILLAN et David CAPLAN.*

  Chapitre 1, Mécanisme de contrôles d'accès.
  Chapitre 2, Qu'est-ce qu'apporte SELinux.
  Chapitre 4, Classes d'objets et permissions.
  Chapitre 5, Syntaxes des règles Type Enforcement

  La reference policy maintenue et écrite par Tresys :
  http://oss.tresys.com/projects/refpolicy
  Site très avancé sur l'écriture de règles SELinux :
  http://selinuxproject.org/page/Main_Page

- **Étude de l'architecture de LSM :**

  SELinux by Example.
  Chapitre 3, Implémentation et architecture de SELinux
  Étude de NSA Security Enhanced Linux de Mathieu BAEUMLER, David HOEUNG et Laurent VALLAR
  http://www.labri.fr/perso/fleury/courses/SS/download/papers/selinux-fr.pdf

- **Pour comprendre le concept TE dans le cadre de Fedora :**

Le journal de Daniel WALSH ingénieur sécurité RedHat spécialisé dans SELinux et très actif dans la communauté SELinux :
http://danwalsh.livejournal.com/
Tutorial sur SELinux en français :
http://www.linuxcertif.com/doc/Comprendre%20SELinux/
Présentation de Eli Billauer « Breaking the Ice with SELinux":
http://www.haifux.org/lectures/200/selinux.pdf
Le rapport de Xavier Gattuso et de Sylvain Roschi écrit dans le cadre du cours de M. LITZISTORF

- **Pour comprendre le concept MCS/MLS :**
  Guide complet de déploiement Red Hat, Document complet et très intéressant :
  http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/
  Guide utilisateur de SELinux RedHat :
  http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html-single/Security-Enhanced_Linux/index.html
  Article très bien écrit de Russell Coker :
  http://www.linuxjournal.com/article/9408

- **Pour générer une policy avec polgen :**
  L'article "Guided Policy Generation for Application" de David R. Harris, Brian T. Sniffen et John D. Ramsdell
  http://www.mitre.org/work/tech_papers/tech_papers_06/06_0046/06_0046.pdf

- **Utilisé dans la construction du scénario d'une attaque :**
  L'article "Guided Policy Generation for Application" de David R. Harris, Brian T. Sniffen et John D. Ramsdell
  http://www.mitre.org/work/tech_papers/tech_papers_06/06_0046/06_0046.pdf

- **Exploit d'une porte dérobée dans le FTP :**
  Exploit sur proftpd :
  http://www.aldeid.com/wiki/Exploits/proftpd-1.3.3c-backdoor

- **Comparatif modules de sécurité linux (LSM) :**
  http://www.cyberciti.biz/tips/selinux-vs-apparmor-vs-grsecurity.html

A ANNEXES

A.1 SYSTEME TARGETED ET STRICT

« Strict Policy » est une politique où aucun accès n'est autorisé par défaut. Les règles SELinux doivent garantir des droit à travers des règles « allow ». Le principe du moindre privilège est de mise. Chaque processus est restreint dans un domaine. Le dossier où sont stockés les règles et les fichiers de configuration lié avec « Strict Policy » est /etc/selinux/strict/.
La « Targeted Policy » est un politique où seules les applications les plus exposées aux attaques sont restreintes dans des domaines. SELinux est un système qui implémente un contrôle d'accès de type liste blanche. Mais rien n'empêche la création d'un domaine qui comprend tous les éléments du système. Et par la suite créer une règle qui donne accès à tous les types du système. Le domaine non restreint « unconfined_t » est un domaine qui fait partie de la targeted policy. Tous ce qui est démarré dans ce domaine « unconfined_t », tourne comme si SELinux était désactivé. Par défaut les processus sont dans le domaine « unconfined_t ». Par contre, certains processus transit à un domaine restreint lors de leurs démarrage. Par exemple un serveur web httpd est démarré dans le domaine « unconfined_t ». Une règle transition de type l'obligera à transiter dans un domaine « httpd_t » afin de le limiter aux ressources nécessaires à son fonctionnement.

Depuis Red Hat Enterprise Linux 5 et Fedora Core 5, le système « Strict Policy » a été l'équivalent du système « Targeted Policy » avec le domaine « unconfined_t » en moins. À partir de Fedora 8, le système « Strict Policy » a été fusionné dans le « Targeted Policy ». Nous pouvons revenir à un système strict en désactivant le module « unconfined ».

A.2 UTILISATEUR SELINUX DISPONIBLE DANS FEDORA[19]

The guest_u SELinux user:

This profile is used for users that need to be tightly controlled. The guest_u SELinux user can only log in using OpenSSH. Guest users have no access to network resources, setuid, setgid programs.

The xguest_u SELinux user:

This profile is identical to that of guest_u. The exception is that Xguest users can only log in to Xwindows and cannot log in using OpenSSH. Another exception of Xguest users is that this partical user can access HTTP port using a SELinux restricted instance of Mozilla Firefox.

The user_u SELinux user:

The user_u SELinux user resembles a ordinary unprivileged SELinux confined user. This user can log in using Xwindows and OpenSSH, has access to network resources, but cannot use setuid and setgid programs.

The staff_u SELinux user:

This SELinux user is identical to user_u except that staff_u can access setuid and getgid programs. The staff_u SELinux user can also stat all process on the system amongst other minor extra privileges compared to user_u.

The sysadm_u SELinux user:

This user is designed for SELinux restricted root login, which is not recommended. This SELinux user is used in a Multi-Level Security Environment where there is no unconfined_u.

---

[19] http://selinux-mac.blogspot.ch/2009/06/selinux-lockdown-part-one-confined.html

The unconfined_u SELinux user:

The unconfined_u SELinux user is the environment where all Linux users are mapped to be default in Fedora Targeted policy. This user is to a large extend exempted from SELinux confinement. The exception is Memory Execution Protections.

Real Linux users, not root, should not be mapped to the unconfined_u SELinux user group if you want to improve security on your system. In many scenarios having unconfined users on a system creates a gaping hole in security.

Root logins should be prohibited always. Root should only be able to log in using the terminal in case of an emergency. In Fedora, the Linux user root is mapped to unconfined_u. This means that root logins are almost not protected by SELinux.

The improve the security of root logins one could map the root Linux user to the sysadm_u SELinux user. Although this does not provide much security over unconfined_u, and root will be able to bypass SELinux security.

A.3 PAQUETS SELINUX DISPONIBLE

```
1   [root@SELinux ~]# yum search selinux
2   Loaded plugins: langpacks, presto, refresh-packagekit
3   ================================================================
    ========== N/S Matched: selinux
    ================================================================
    ===========
4   crossfire-selinux.x86_64 : SELinux policy files for crossfire
5   dokuwiki-selinux.noarch : SElinux support for dokuwiki
6   drraw-selinux.noarch : SELinux context for drraw
7   freeipa-server-selinux.x86_64 : SELinux rules for freeipa-server
    daemons
8   gcl-selinux.x86_64 : SELinux policy for GCL images
9   ladvd-selinux.x86_64 : SELinux policy module supporting ladvd
10  libselinux.i686 : SELinux library and simple utilities
11  libselinux.x86_64 : SELinux library and simple utilities
12  libselinux-devel.i686 : Header files and libraries used to build
    SELinux
13  libselinux-devel.x86_64 : Header files and libraries used to build
    SELinux
14  libselinux-python.x86_64 : SELinux python bindings for libselinux
15  libselinux-python3.x86_64 : SELinux python 3 bindings for
    libselinux
16  libselinux-ruby.x86_64 : SELinux ruby bindings for libselinux
```

```
17   libselinux-static.x86_64 : Static libraries used to build SELinux
18   libselinux-utils.x86_64 : SELinux libselinux utilies
19   mod_selinux.x86_64 : Apache/SELinux plus module
20   php-pecl-selinux.x86_64 : SELinux binding for PHP scripting
     language
21   pki-selinux.noarch : Certificate System - PKI Selinux Policies
22   pure-ftpd-selinux.x86_64 : SELinux support for Pure-FTPD
23   sagator-selinux.noarch : SELinux support for SAGATOR
24   selinux-policy.noarch : SELinux policy configuration
25   selinux-policy-doc.noarch : SELinux policy documentation
26   selinux-policy-minimum.noarch : SELinux minimum base policy
27   selinux-policy-mls.noarch : SELinux mls base policy
28   selinux-policy-targeted.noarch : SELinux targeted base policy
29   websvn-selinux.noarch : SELinux context for websvn
30   xpilot-ng-selinux.x86_64 : SELinux policy files for the xpilot-ng
     game server
31   checkpolicy.x86_64 : SELinux policy compiler
32   eclipse-setools.x86_64 : Eclipse plugin wrapper for SETools Java
     policy analysis tools for SELinux
33   eclipse-slide.noarch : SELinux policy editing plugin for Eclipse
34   libsemanage.i686 : SELinux binary policy manipulation library
35   libsemanage.x86_64 : SELinux binary policy manipulation library
36   libsepol.i686 : SELinux binary policy manipulation library
37   libsepol.x86_64 : SELinux binary policy manipulation library
38   mailgraph-selinux.noarch : A RRDtool frontend for Mail statistics
39   mcstrans.x86_64 : SELinux Translation Daemon
40   policycoreutils.x86_64 : SELinux policy core utilities
41   policycoreutils-gui.x86_64 : SELinux configuration GUI
42   policycoreutils-python.x86_64 : SELinux policy core python
     utilities
43   policycoreutils-restorecond.x86_64 : SELinux restorecond utilities
44   policycoreutils-sandbox.x86_64 : SELinux sandbox utilities
45   queuegraph-selinux.noarch : A RRDtool frontend for Mail statistics
46   setools.x86_64 : Policy analysis tools for SELinux
47   setools-console.x86_64 : Policy analysis command-line tools for
     SELinux
48   setools-devel.i686 : Policy analysis development files for SELinux
49   setools-devel.x86_64 : Policy analysis development files for
     SELinux
50   setools-gui.x86_64 : Policy analysis graphical tools for SELinux
51   setools-libs.i686 : Policy analysis support libraries for SELinux
52   setools-libs.x86_64 : Policy analysis support libraries for
     SELinux
53   setools-libs-java.i686 : Java bindings for SELinux policy analysis
54   setools-libs-java.x86_64 : Java bindings for SELinux policy
     analysis
55   setools-libs-python.x86_64 : Python bindings for SELinux policy
     analysis
56   setools-libs-tcl.x86_64 : Tcl bindings for SELinux policy analysis
57   setroubleshoot.x86_64 : Helps troubleshoot SELinux problems
58   setroubleshoot-server.x86_64 : SELinux troubleshoot server
```

Installation utilitaire GUI

- Pour simplifier l'étude de SELinux, installer paquetage d'utilitaires SELinux.
- Ouvrir terminal et passer en root

```
1    # su
```

```
2    # yum -y install policycoreutils-gui.x86_64
```

Dans ce paquetage, il y a deux outils graphiques, le premier est SELinux Management et le deuxième est SELinux policy generation tool. Pour le moment, je m'intéresse uniquement à la gestion de SELinux et non à l'écriture de nouvelles règles. Donc je vais laisser SELinux policy generation tool de côté pour le moment.
SELinux Management est un outil crée par Dan Walsh, ingénieur sécurité très actif dans SELinux de l'entreprise RedHat.

- Installer un paquetage composé d'une suite d'outil open source développé par Tresys.

Ce paquetage contient un outil essentiel à l'analyse et au débogage des règles SELinux. Il s'agit d'APOL. En plus de cet outil, ce paquetage contient d'autres outils intéressant tels que seaudit, sediff, findcon.

```
1    # yum -y install setools-gui.x86_64
```

Ci-dessous un référencement des outils SELinux disponible[20] :
The following is a brief description of the main SELinux packages:

- *policycoreutils-python* — provides utilities such as **semanage**, **audit2allow**, **audit2why** and **chcat**, for operating and managing SELinux.
- *policycoreutils* — provides utilities such as **restorecon**, **secon**, **setfiles**, **semodule**, **load_policy**, and **setsebool**, for operating and managing SELinux.
- *policycoreutils-gui* — provides **system-config-selinux**, a graphical tool for managing SELinux.
- *selinux-policy* — provides the SELinux Reference Policy. The SELinux Reference Policy is a complete SELinux policy, and is used as a basis for other policies, such as the SELinux targeted policy; refer to the Tresys Technology SELinux Reference Policy page for further information. This package also provides the **/usr/share/selinux/devel/policygentool** development utility, as well as example policy files.
- *selinux-policy-**policy*** — provides SELinux policies. For targeted policy, install selinux-policy-targeted. For MLS, install selinux-policy-mls.
- *setroubleshoot-server* — translates denial messages, produced when access is denied by SELinux, into detailed descriptions that are viewed with **sealert** (which is provided by this package).

[20] http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/chap-Security-Enhanced_Linux-Working_with_SELinux.html#sect-Security-Enhanced_Linux-Working_with_SELinux-SELinux_Packages

- *setools-console* — this package provides the Tresys Technology SETools distribution, a number of tools and libraries for analyzing and querying policy, audit log monitoring and reporting, and file context management[8]. The setools package is a meta-package for SETools. The *setools-gui* package provides the **apol**, **seaudit**, and **sediffx** tools. The setools-console package provides the **seaudit-report**, **sechecker**, **sediff**, **seinfo**, **sesearch**, **findcon**, **replcon**, and **indexcon** command line tools. Refer to the Tresys Technology SETools page for information about these tools.
- *libselinux-utils* — provides the avcstat, getenforce, getsebool, matchpathcon, selinuxconlist, selinuxdefcon, selinuxenabled, setenforce, togglesebool tools.
- *mcstrans* — translates levels, such as s0-s0:c0.c1023, to an easier to read form, such as SystemLow-SystemHigh. This package is not installed by default.

To install packages in Red Hat Enterprise Linux, as the Linux root user, run the yum install package-name command. For example, to install the mcstrans package, run the yum install mcstrans command. To upgrade all installed packages in Red Hat Enterprise Linux, run the yum update command.

A.4 DOCUMENTATION DE LA REFERENCE POLICY

```
1    [root@SELinux ~]# yum info selinux-policy-doc.noarch
2    Loaded plugins: langpacks, presto, refresh-packagekit
3    Installed Packages
4    Name        : selinux-policy-doc
5    Arch        : noarch
6    Version     : 3.10.0
7    Release     : 86.fc16
8    Size        : 14 M
9    Repo        : installed
10   From repo   : updates
11   Summary     : SELinux policy documentation
12   URL         : http://oss.tresys.com/repos/refpolicy/
13   License     : GPLv2+
14   Description : SELinux policy documentation package
```

Exemple de module :

```
1    [root@SELinux ~]# ls -l /usr/share/doc/selinux-policy-3.10.0/
2    total 36
3    -rw-r--r--. 1 root root   185 Apr 24 13:30 example.fc
4    -rw-r--r--. 1 root root  1033 Apr 24 13:30 example.if
5    -rw-r--r--. 1 root root   516 Apr 24 13:30 example.te
6    drwxr-xr-x. 2 root root 20480 May 24 14:00 html
```

```
7    -rw-r--r--. 1 root root   195 Apr 24 13:30 Makefile.example
```

A.5 REINSTALLATION TARGETED POLICY

```
# setenforce 0
# rm -rf /etc/selinux/targeted
# yum -y reinstall selinux-policy selinux-policy-targeted
# restorecon -R -v /etc/selinux/targeted
# setenforce 1
```

A.6 DESACTIVER LE MODULE UNCONFINED

Dans Fedora 16, il y a des modules qui ne sont pas lié à une application mais lié avec plusieurs. Par exemple le module unconfined permet de limiter les possibilités d'un utilisateur unconfined :

```
1    [root@SELinux ~]# semodule -l | grep unconfined
2    unconfined  3.3.0
3    unconfineduser    1.0.0
```

Si je désinstalle ce module il y aura des impacts sur la sécurité du système. Nous allons regarder ces changements liés à la désinstallation de ce module :

Nous avons ci-dessus les règles liés avec l'attribut unconfined_domain_type. Il y en a 69 règles et il sont très généraliste et donne l'accès à un grand nombre de répertoires.
Exemple de règle:

```
allow unconfined_domain_type user_home_dir_t : dir { ioctl read write
getattr lock add_name remove_name search open } ;
```

Donne accès à tous les types contenus dans l'attribut unconfined_domain_type sur les ressources labellisé avec le type user_home_dir_t. Le label user_home_dir_t est donné au répertoire personnel des utilisateurs.

L'attribut unconfined_domain_type regroupe 60 types.
Un autre point important dans TE, ce sont les transitions de domaine. Je vais chercher le nombre de règles de transition lié avec le unconfined_domain_type et je mets un filtre sur la classe process :

Il y a 2872 règles actives de transition de domaine autorisée. Maintenant je désactive le module unconfined et je compare :

```
1    [root@SELinux ~]# semodule -d unconfined
```

Je relève le nombre de type contenu dans attribut après que j'ai désactivé le module unconfined :

Il n'y a plus que 10 types compris dans cet attribut. Les règles de transition ont aussi subi des modifications voici le rendu de APOL :

835 rules match the search criteria.
Number of enabled conditional rules: 6
Number of disabled conditional rules: 0

type_transition anaconda_t NetworkManager_initrc_exec_t : process initrc_t;
type_transition anaconda_t abrt_helper_exec_t : process abrt_helper_t;
type_transition anaconda_t abrt_initrc_exec_t : process initrc_t;
type_transition anaconda_t admin_passwd_exec_t : process sysadm_passwd_t;
type_transition anaconda_t afs_initrc_exec_t : process initrc_t;
type_transition anaconda_t aiccu_initrc_exec_t : process initrc_t;
...
Nous sommes passés de 2872 règles à 835 règles. La règles allow utilise l'attribut unconfined_domain_type et comme celui-ci contient moins de types, les règles ont moins d'étendu.

A.7 EXEMPLE MCS[21]

Use the semanage login -a command to assign Linux users to SELinux user identities:

```
1    # semanage login -a james
2    # semanage login -a daniel
3    # semanage login -a olga
```

Red Hat Enterprise Linux and SELinux are preconfigured with several default categories, but to make effective use of MCS, the system administrator typically modifies these or creates further categories to suit local requirements.
SELinux maintains a mapping between internal sensitivity and category levels and their human-readable representations in the setrans.conf file. The system administrator edits this file to manage and maintain the required categories.
Use the chcat -L command to list the current categories:

[21] Deployment Guide RedHat

```
1    [root@dhcp-133 tmp]# chcat -L
2    s0:c0 CompanyConfidential
3    s0:c3 TopSecret
4    s0
5    s0-s0:c0.c255 SystemLow-SystemHigh
6    s0:c0.c255 SystemHigh
```

To modify the categories or to start creating your own, modify the /etc/selinux/<selinuxtype>/setrans.conf file. For the example introduced above, add the Marketing, Finance, Payroll, and Personnel categories as follows (this example uses the targeted policy, and irrelevant sections of the file have been omitted):

```
1    [root@dhcp-133 tmp]# vi /etc/selinux/targeted/setrans.conf
2    s0:c0=Marketing
3    s0:c1=Finance
4    s0:c2=Payroll
5    s0:c3=Personnel
```

Use the chcat -L command to check the newly-added categories:

```
1    [root@dhcp-133 tmp]# chcat -L
2    s0:c0 Marketing
3    s0:c1 Finance
4    s0:c2 Payroll
5    s0:c3 Personnel
6    s0
7    s0-s0:c0.c255 SystemLow-SystemHigh
8    s0:c0.c255 SystemHigh
```

After you make any changes to the setrans.conf file, you need to restart the MCS translation service before those changes takes effect. Use the following command to restart the service:

```
1    [root@dhcp-133 ~]# service mcstrans restart
```

Now that the required categories have been added to the system, you can start assigning them to SELinux users and files. To further develop the example above, assume that James is in the Marketing department, Daniel is in the Finance and Payroll departments, and Olga is in the Personnel department. Each of these users has already been assigned an SELinux login.

Use the chcat command to assign MCS categories to SELinux logins:

```
1    [root@dhcp-133 ~]# chcat -l -- +Marketing james
2    [root@dhcp-133 ~]# chcat -l -- +Finance,+Payroll daniel
3    [root@dhcp-133 ~]# chcat -l -- +Personnel olga
```

For example, if the company director also requires a user account with access to all categories, follow the same procedure as above:

Create a user account for the company director (Karl)

```
1    [root@dhcp-133 ~]# useradd karl
2    [root@dhcp-133 ~]# passwd karl
```

Assign the user account to an SELinux login

```
1    [root@dhcp-133 ~]# semanage login -a karl
```

Assign all the MCS categories to the new login

```
1    [root@dhcp-133 ~]# chcat -l --
     +Marketing,+Finance,+Payroll,+Personnel karl
```

Use the chcat command to verify the addition of the new user:

```
1    [root@dhcp-133 ~]# chcat -L -l daniel james olga karl
2    daniel: Finance,Payroll
3    james: Marketing
4    olga: Personnel
5    karl: Marketing,Finance,Payroll,Personnel
```

MCS category access is assigned during login. Consequently, a user does not have access to newly-assigned categories until they log in again. Similarly, if access to a category is revoked, this is only apparent to the user after the next login. At this point we have a system that has several user accounts, each of which is mapped to an SELinux user identity. We have also established a number of categories that are suitable for the particular deployment, and assigned those categories to different users.

All of the files on the system, however, still fall under the same category, and are therefore accessible by everyone (but still according to the standard Linux DAC and TE constraints). We now need to assign categories to the various files on the system so that only the appropriate users can access them.

For this example, we create a file in Daniel's home directory:

```
1    [daniel@dhcp-133 ~]$ echo "Financial Records 2006" >
     financeRecords.txt
2    [daniel@dhcp-133 ~]$ ls -Z financeRecords.txt
3    -rw-r--r-- daniel daniel user_u:object_r:user_home_t
     financeRecords.txt
```

Notice that at this stage the file has the default context for a file created in the user's home directory (user_home_t) and has no categories assigned to it. We can add the required category using the chcat command. Now when you check the security context of the file, you can see the category has been applied.

```
1    [daniel@dhcp-133 ~]$ chcat -- +Finance financeRecords.txt
2    [daniel@dhcp-133 ~]$ ls -Z financeRecords.txt
3    -rw-r--r-- daniel daniel root:object_r:user_home_t:Finance
4    financeRecords.txt
```

In many cases, you need to assign more than one category to a file. For example, some files may need to be accessible to users from both the Finance and Payroll departments.

```
1     [daniel@dhcp-133 ~]$ chcat -- +Payroll financeRecords.txt
2    [daniel@dhcp-133 ~]$ ls -Z financeRecords.txt
3    -rw-r--r-- daniel daniel root:object_r:user_home_t:Finance,Payroll
4    financeRecords.txt
```

If a user who is assigned to a different category tries to access the file, they receive an error message:

```
1    [olga@dhcp-133 ~]$ cat financeRecords.txt
2    cat: financeRecords.txt: Permission Denied
```

A.8 CODE DU MODULE DROPBOX

- **dropbox.fc**

```
1
2    /home/test/.config/autostart/dropbox.desktop       --
      gen_context(system_u:object_r:dropbox_rw_t,s0)
3
4
5    /home/test/.dropbox(/.*)?
      gen_context(system_u:object_r:dropbox_rw_t,s0)
6
7
8    /home/test/.dropbox-dist(/.*)?
      gen_context(system_u:object_r:dropbox_rw_t,s0)
9
10
11   /usr/bin/dropbox       --
      gen_context(system_u:object_r:dropbox_exec_t,s0)
```

- **dropbox.if**

```
1
2    ## <summary>policy for dropbox</summary>
3
4
5    ########################################
6    ## <summary>
7    ##     Transition to dropbox.
8    ## </summary>
9    ## <param name="domain">
10   ## <summary>
11   ##     Domain allowed to transition.
12   ## </summary>
13   ## </param>
14   #
15   interface(`dropbox_domtrans',`
16    gen_require(`
17          type dropbox_t, dropbox_exec_t;
18    ')
19
20    corecmd_search_bin($1)
21    domtrans_pattern($1, dropbox_exec_t, dropbox_t)
22   ')
23
24
25   ########################################
26   ## <summary>
27   ##     Search dropbox rw directories.
28   ## </summary>
29   ## <param name="domain">
30   ##     <summary>
31   ##     Domain allowed access.
32   ##     </summary>
33   ## </param>
34   #
35   interface(`dropbox_search_rw_dir',`
36    gen_require(`
37          type dropbox_rw_t;
```

```
38    ')
39
40    allow $1 dropbox_rw_t:dir search_dir_perms;
41    files_search_rw($1)
42  ')
43
44  ########################################
45  ## <summary>
46  ##     Read dropbox rw files.
47  ## </summary>
48  ## <param name="domain">
49  ##     <summary>
50  ##     Domain allowed access.
51  ##     </summary>
52  ## </param>
53  #
54  interface(`dropbox_read_rw_files',`
55    gen_require(`
56          type dropbox_rw_t;
57    ')
58
59    allow $1 dropbox_rw_t:file read_file_perms;
60    allow $1 dropbox_rw_t:dir list_dir_perms;
61    files_search_rw($1)
62  ')
63
64  ########################################
65  ## <summary>
66  ##     Manage dropbox rw files.
67  ## </summary>
68  ## <param name="domain">
69  ##     <summary>
70  ##     Domain allowed access.
71  ##     </summary>
72  ## </param>
73  #
74  interface(`dropbox_manage_rw_files',`
75    gen_require(`
76          type dropbox_rw_t;
77    ')
78
79    manage_files_pattern($1, dropbox_rw_t, dropbox_rw_t)
80  ')
81
82  ########################################
83  ## <summary>
84  ##     Create, read, write, and delete
85  ##     dropbox rw dirs.
86  ## </summary>
87  ## <param name="domain">
88  ##     <summary>
89  ##     Domain allowed access.
90  ##     </summary>
91  ## </param>
92  #
93  interface(`dropbox_manage_rw_dirs',`
94    gen_require(`
95          type dropbox_rw_t;
96    ')
```

```
97
98    manage_dirs_pattern($1, dropbox_rw_t, dropbox_rw_t)
99   ')
100
101
102  ########################################
103  ## <summary>
104  ##     Execute dropbox in the dropbox domain, and
105  ##     allow the specified role the dropbox domain.
106  ## </summary>
107  ## <param name="domain">
108  ##     <summary>
109  ##     Domain allowed to transition
110  ##     </summary>
111  ## </param>
112  ## <param name="role">
113  ##     <summary>
114  ##     The role to be allowed the dropbox domain.
115  ##     </summary>
116  ## </param>
117  #
118  interface(`dropbox_run',`
119   gen_require(`
120         type dropbox_t;
121   ')
122
123   dropbox_domtrans($1)
124   role $2 types dropbox_t;
125  ')
126
127  ########################################
128  ## <summary>
129  ##     Role access for dropbox
130  ## </summary>
131  ## <param name="role">
132  ##     <summary>
133  ##     Role allowed access
134  ##     </summary>
135  ## </param>
136  ## <param name="domain">
137  ##     <summary>
138  ##     User domain for the role
139  ##     </summary>
140  ## </param>
141  #
142  interface(`dropbox_role',`
143   gen_require(`
144         type dropbox_t;
145   ')
146
147   role $1 types dropbox_t;
148
149   dropbox_domtrans($2)
150
151   ps_process_pattern($2, dropbox_t)
152   allow $2 dropbox_t:process signal;
153  ')
154
155
```

```
156  ########################################
157  ## <summary>
158  ##     All of the rules required to administrate
159  ##     an dropbox environment
160  ## </summary>
161  ## <param name="domain">
162  ##     <summary>
163  ##     Domain allowed access.
164  ##     </summary>
165  ## </param>
166  ## <param name="role">
167  ##     <summary>
168  ##     Role allowed access.
169  ##     </summary>
170  ## </param>
171  ## <rolecap/>
172  #
173  interface(`dropbox_admin',`
174   gen_require(`
175         type dropbox_t;
176   type dropbox_rw_t;
177   ')
178
179   allow $1 dropbox_t:process { ptrace signal_perms };
180   ps_process_pattern($1, dropbox_t)
181
182   files_search_etc($1)
183   admin_pattern($1, dropbox_rw_t)
184
185  ')
```

- **dropbox.te**

Pour les macros utilisées sont issue de plusieurs fichiers, principalement :

- dropbox.if
- /usr/share/selinux/devel/include/support/

```
1    policy_module(dropbox, 1.0.0)
2
3    ########################################
4    #
5    # Declarations
6    #
7
8    type dropbox_t;
9    type dropbox_exec_t;
10   application_domain(dropbox_t, dropbox_exec_t)
11   role system_r types dropbox_t;
12
13   permissive dropbox_t;
14
15   type dropbox_rw_t;
16   files_type(dropbox_rw_t)
17
18   ########################################
19   #
20   # dropbox local policy
21   #
22
23   allow dropbox_t self:fifo_file manage_fifo_file_perms;
```

```
24  allow dropbox_t self:unix_stream_socket
    create_stream_socket_perms;
25
26  manage_dirs_pattern(dropbox_t, dropbox_rw_t, dropbox_rw_t)
27  manage_files_pattern(dropbox_t, dropbox_rw_t, dropbox_rw_t)
28
29  sysnet_dns_name_resolve(dropbox_t)
30  corenet_all_recvfrom_unlabeled(dropbox_t)
31
32  allow dropbox_t self:tcp_socket create_stream_socket_perms;
33  corenet_tcp_sendrecv_generic_if(dropbox_t)
34  corenet_tcp_sendrecv_generic_node(dropbox_t)
35  corenet_tcp_sendrecv_all_ports(dropbox_t)
36  corenet_tcp_bind_generic_node(dropbox_t)
37  corenet_tcp_bind_all_unreserved_ports(dropbox_t)
38  corenet_tcp_connect_all_ports(dropbox_t)
39
40  domain_use_interactive_fds(dropbox_t)
41
42  files_read_etc_files(dropbox_t)
43
44  miscfiles_read_localization(dropbox_t)
45
46  optional_policy(`
47   gen_require(`
48         type staff_t;
49         role staff_r;
50   ')
51
52   dropbox_run(staff_t, staff_r)
53  ')
54
55  optional_policy(`
56   gen_require(`
57         type sysadm_t;
58         role sysadm_r;
59   ')
60
61   dropbox_run(sysadm_t, sysadm_r)
62  ')
63
64  optional_policy(`
65   gen_require(`
66         type unconfined_t;
67         role unconfined_r;
68   ')
69
70   dropbox_run(unconfined_t, unconfined_r)
71  ')
72
73  optional_policy(`
74   gen_require(`
75         type user_t;
76         role user_r;
77   ')
78
79   dropbox_run(user_t, user_r)
80  ')
```

- **dropbox.sh**

```
1   #!/bin/sh -e
2
3   DIRNAME=`dirname $0`
4   cd $DIRNAME
5   USAGE="$0 [ --update ]"
6   if [ `id -u` != 0 ]; then
7   echo 'You must be root to run this script'
8   exit 1
9   fi
10
11  if [ $# -eq 1 ]; then
12   if [ "$1" = "--update" ] ; then
13         time=`ls -l --time-style="+%x %X" dropbox.te | awk '{
    printf "%s %s", $6, $7 }'`
14         rules=`ausearch --start $time -m avc --raw -se dropbox`
15         if [ x"$rules" != "x" ] ; then
16               echo "Found avc's to update policy with"
17               echo -e "$rules" | audit2allow -R
18               echo "Do you want these changes added to policy
    [y/n]?"
19               read ANS
20               if [ "$ANS" = "y" -o "$ANS" = "Y" ] ; then
21                     echo "Updating policy"
22                     echo -e "$rules" | audit2allow -R >> dropbox.te
23                     # Fall though and rebuild policy
24               else
25                     exit 0
26               fi
27         else
28               echo "No new avcs found"
29               exit 0
30         fi
31   else
32         echo -e $USAGE
33         exit 1
34   fi
35  elif [ $# -ge 2 ] ; then
36   echo -e $USAGE
37   exit 1
38  fi
39
40  echo "Building and Loading Policy"
41  set -x
42  make -f /usr/share/selinux/devel/Makefile || exit
43  /usr/sbin/semodule -i dropbox.pp
44
45  # Fixing the file context on /usr/bin/dropbox
46  /sbin/restorecon -F -R -v /usr/bin/dropbox
47  # Fixing the file context on
    /home/test/.config/autostart/dropbox.desktop
48  /sbin/restorecon -F -R -v
    /home/test/.config/autostart/dropbox.desktop
49  # Fixing the file context on /home/test/.dropbox
50  /sbin/restorecon -F -R -v /home/test/.dropbox
51  # Fixing the file context on /home/test/.dropbox-dist
52  /sbin/restorecon -F -R -v /home/test/.dropbox-dist
```

- **Patch dropbox.te**

```
1
2    require {
3     type staff_t;
4     type staff_dbusd_t;
5     type home_root_t;
6     type ld_so_t;
7     type usr_t;
8     type data_home_t;
9     type node_t;
10    type fonts_t;
11    type user_devpts_t;
12    type shell_exec_t;
13    type tmp_t;
14    type random_device_t;
15    type proc_t;
16    type gconf_home_t;
17    type xdm_var_run_t;
18    type proc_net_t;
19    type fonts_cache_t;
20    type sysfs_t;
21    type unreserved_port_t;
22    type user_home_dir_t;
23    type urandom_device_t;
24    type xserver_t;
25    type dropbox_t;
26    type ldconfig_exec_t;
27    type config_home_t;
28    type xdm_home_t;
29    type bin_t;
30    type user_home_t;
31    type fs_t;
32    type tmpfs_t;
33    type system_dbusd_var_lib_t;
34    class process { siginh execstack noatsecure execmem rlimitinh };
35    class unix_stream_socket connectto;
36    class chr_file { read ioctl write getattr open };
37    class shm { write unix_read unix_write read destroy create };
38    class file { rename execute setattr read lock create getattr
     execute_no_trans write ioctl unlink open append };
39    class filesystem getattr;
40    class sock_file { write getattr setattr read create unlink };
41    class lnk_file { read getattr };
42    class unix_dgram_socket { create ioctl };
43    class udp_socket { name_bind node_bind };
44    class dir { search setattr read write getattr remove_name open
     add_name };
45   }
46
47   #============= dropbox_t ==============
48   allow dropbox_t bin_t:file { execute getattr read open ioctl
     execute_no_trans };
49   allow dropbox_t bin_t:lnk_file { read getattr };
50   allow dropbox_t config_home_t:dir { read getattr setattr };
51   #!!!! The source type 'dropbox_t' can write to a 'file' of the
     following types:
52   # dropbox_rw_t, dropbox_home_t
53
54   allow dropbox_t config_home_t:file { write getattr open };
```

```
55  allow dropbox_t data_home_t:dir search;
56  allow dropbox_t fonts_cache_t:file { read getattr open };
57  allow dropbox_t fonts_t:dir getattr;
58  allow dropbox_t fonts_t:file { read getattr open };
59  allow dropbox_t fs_t:filesystem getattr;
60  allow dropbox_t gconf_home_t:dir search;
61  allow dropbox_t home_root_t:dir getattr;
62  allow dropbox_t ld_so_t:file execute_no_trans;
63  allow dropbox_t ldconfig_exec_t:file { read execute open
    execute_no_trans };
64  allow dropbox_t node_t:udp_socket node_bind;
65  allow dropbox_t proc_net_t:file read;
66  allow dropbox_t proc_t:file { read getattr open };
67  allow dropbox_t random_device_t:chr_file { read getattr open ioctl
    };
68  allow dropbox_t self:process { execstack execmem };
69  allow dropbox_t self:shm { write unix_read unix_write read destroy
    create };
70  allow dropbox_t self:unix_dgram_socket { create ioctl };
71  allow dropbox_t self:unix_stream_socket connectto;
72  allow dropbox_t shell_exec_t:file { read execute open getattr
    execute_no_trans };
73  allow dropbox_t staff_dbusd_t:unix_stream_socket connectto;
74  allow dropbox_t sysfs_t:file { read open };
75  allow dropbox_t system_dbusd_var_lib_t:file { read getattr open };
76  #!!!! The source type 'dropbox_t' can write to a 'dir' of the
    following types:
77  # dropbox_rw_t, dropbox_home_t
78
79  allow dropbox_t tmp_t:dir { write remove_name read add_name };
80  #!!!! The source type 'dropbox_t' can write to a 'file' of the
    following types:
81  # dropbox_rw_t, dropbox_home_t
82
83  allow dropbox_t tmp_t:file { write getattr read create unlink open
    };
84  allow dropbox_t tmpfs_t:file { read write };
85  allow dropbox_t unreserved_port_t:udp_socket name_bind;
86  #!!!! This avc can be allowed using the boolean 'global_ssp'
87
88  allow dropbox_t urandom_device_t:chr_file { read getattr open
    ioctl };
89  allow dropbox_t user_devpts_t:chr_file { read write getattr ioctl
    };
90  allow dropbox_t user_home_dir_t:dir { read search open getattr };
91  #!!!! The source type 'dropbox_t' can write to a 'dir' of the
    following types:
92  # dropbox_rw_t, dropbox_home_t
93
94  allow dropbox_t user_home_t:dir { search setattr read write
    getattr remove_name open add_name };
95  allow dropbox_t user_home_t:file { rename execute setattr read
    lock create execute_no_trans write ioctl unlink open };
96  allow dropbox_t user_home_t:sock_file { write getattr setattr read
    create unlink };
97  allow dropbox_t usr_t:file { read getattr open };
98  allow dropbox_t xdm_home_t:file { read getattr ioctl append };
99  allow dropbox_t xdm_var_run_t:dir search;
100 allow dropbox_t xdm_var_run_t:file { read getattr open };
```

```
101  allow dropbox_t xserver_t:unix_stream_socket connectto;
102
103  #============= staff_t ==============
104  allow staff_t dropbox_t:process { siginh rlimitinh noatsecure };
105  allow staff_t dropbox_t:unix_stream_socket connectto;
```