SELinux Access Control

- Discretionary Access Control (DAC)

- Mandatoty Access Control (MAC)

  Type Enforcement, Security Context, Domain Transition

  User, Boolean, Logs,

- Role Based Access Control (RBAC)

- Confidentialité → Bell-La Padula (BLP), Multi-Level Security (MLS), Multi-Category Security (MCS)

- Labo SELinux & ZeroDay

Discretionary Access Control (DAC) : Linux

- Par défaut, Linux (et Windows) implémente le modèle de sécurité Discretionary Access Control = DAC
- Après authentification (username - password), chaque utilisateur possède un UID, un GID (proupe principal) et des groupes secondaires

  → **`whoami + id + groups (démo)`**
- Un fichier créé obtient les droits suivants : écriture pour le propriétaire + lecture pour tous
- Le propriétaire est celui qui créé le fichier
- Pour modifier les droits d'accès → **`(change mode =) chmod`**
- Ordre d'évaluation : Owner – Group – Other

File system permissions

```
ls –l /filename
-rw-r--r-- 1 root root 0 date hour filename
-rw-r--r-- 1 root root 0 date hour filename
```

directory (d) / file (-)

owner (UID=User + GID=Group)

owner – group – other

Ordre d'évaluation : OGO

read (r) / write (w) / execute (x)

length

last modification

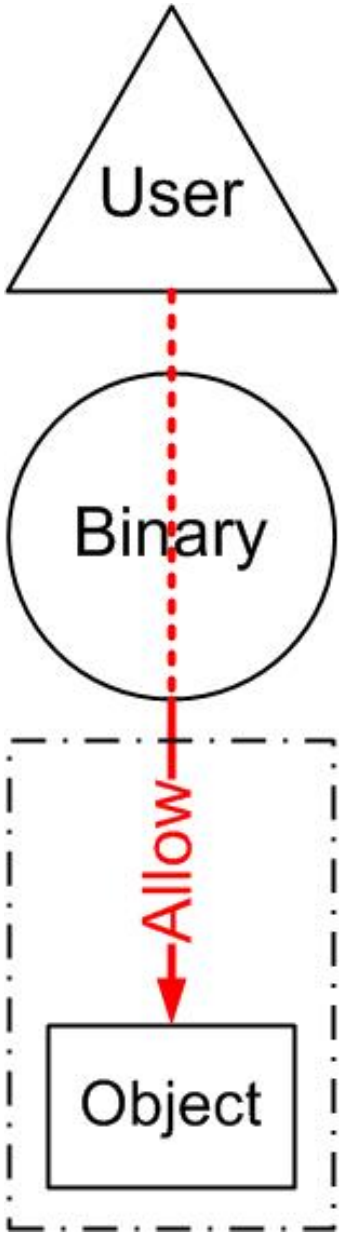
nb of hard link to inode

Permissions on Directories

- Le mécanisme conserve les bits read – write – execute mais ils ont des significations très différentes
- **Read** permet de voir le contenu du dossier (liste de fichiers)
- **Write** autorise la création et la suppression de fichiers
- **Execute** autorise l'accès aux fichiers (inode)

  Tous les dossiers du chemin (/etc/samba/smb.conf) doivent posséder cette autorisation
- L'autorisation execute sans read donne accès au fichier pour autant que l'utilisateur en connaisse le nom
- Pour ajouter ou supprimer il faut execute avec write

Discretionary Access Control : Principe



- Après authentification, l'utilisateur
- exécute la commande cat
  Processus
- **Règle de contrôle d'accès à la ressource pour User**
  Read – Write – Execute
- Fichier des salaires

Discretionary Access Control : Faille conceptuelle

- **Le binaire utilisé n'est pas contrôlé**
- Tout (n'importe quel) binaire peut être utilisé
- Il suffit qu'il hérite des droits appropriés
- Risque : usurpation d'identité, vol d'identité, …
  Vol de mot de passe
  Elévation des privilèges
- Avec XP-SP1, il suffisait d'envoyer un paquet (exploit ms03_026_dcom depuis Metasploit) pour obtenir un cmd avec les droits System !

Utilisateur veut changer son mot de passe

- Joe est un utilisateur standard → il ne dispose pas des droits root

- Il doit utiliser l'exécutable **`/usr/bin/passwd`**

```
[root@localhost liveuser]# ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 22812 16 jui  2010 /usr/bin/passwd
```

owner UID = root

owner's permissions = rws = Read Write SetUID

le processus qui exécute ce fichier passwd possède les droits root

```
[root@localhost liveuser]# ps -e -f | grep passwd
root      1972  1958  0 17:22 pts/0    00:00:00 passwd
```

- Le bit SetUID est couramment utilisé dans le monde Unix
- Il montre une **faiblesse conceptuelle** car **passwd a accès à toutes les ressources !!!**

Set-UID (SUID)

- Problématique

  Seul l'admin (root) peut modifier (écrire) le fichier **shadow**

  -rw-r----- 1 root shadow L D H /etc/shadow

  L' utilisateur doit pouvoir modifier son mot de passe avec **passwd**

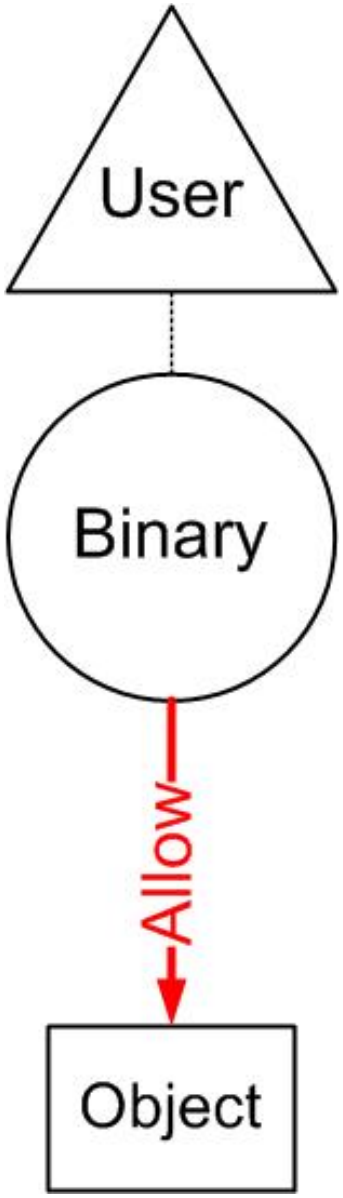
  -rwsr-xr-x 1 root root L D H /usr/bin/passwd

- Solution

  La commande **passwd** possède le bit s qui l'autorise à créer un processus dont l'utilisateur effectif est **root** permettant ainsi la modification du mot de passe

- Avec ce bit s, l'utilisateur de la commande possède les mêmes droits d'accès que le propriétaire (root)

Mandatory Access Control (MAC) : Principe



- Après authentification, l'utilisateur
- exécute la commande cat
  Processus
- **Règle de contrôle d'accès à la ressource pour cat**
  La règle se situe dans le noyau (pas dans la ressource)
  → **appels système ?!?!**
- Fichier des salaires

MAC : SELinux Type Enforcement (1/2)

- Illustration avec le programme **passwd** qui doit accéder au fichier **/etc/shadow**

passwd_t = attribut type du contrôle d'accès

allow

passwd_t Source type → process

shadow_t Target type → object

: file Object class → file

(ioctl read write getattr …) System calls

shadow_t = attribut type du contrôle d'accès

MAC : SELinux Type Enforcement (2/2)

- **Aucun accès par défaut (white-list)**

- Chaque processus et chaque objet possède un contexte de sécurité (**security context)** contenant les attributs de contrôle d'accès = user : role : **type**

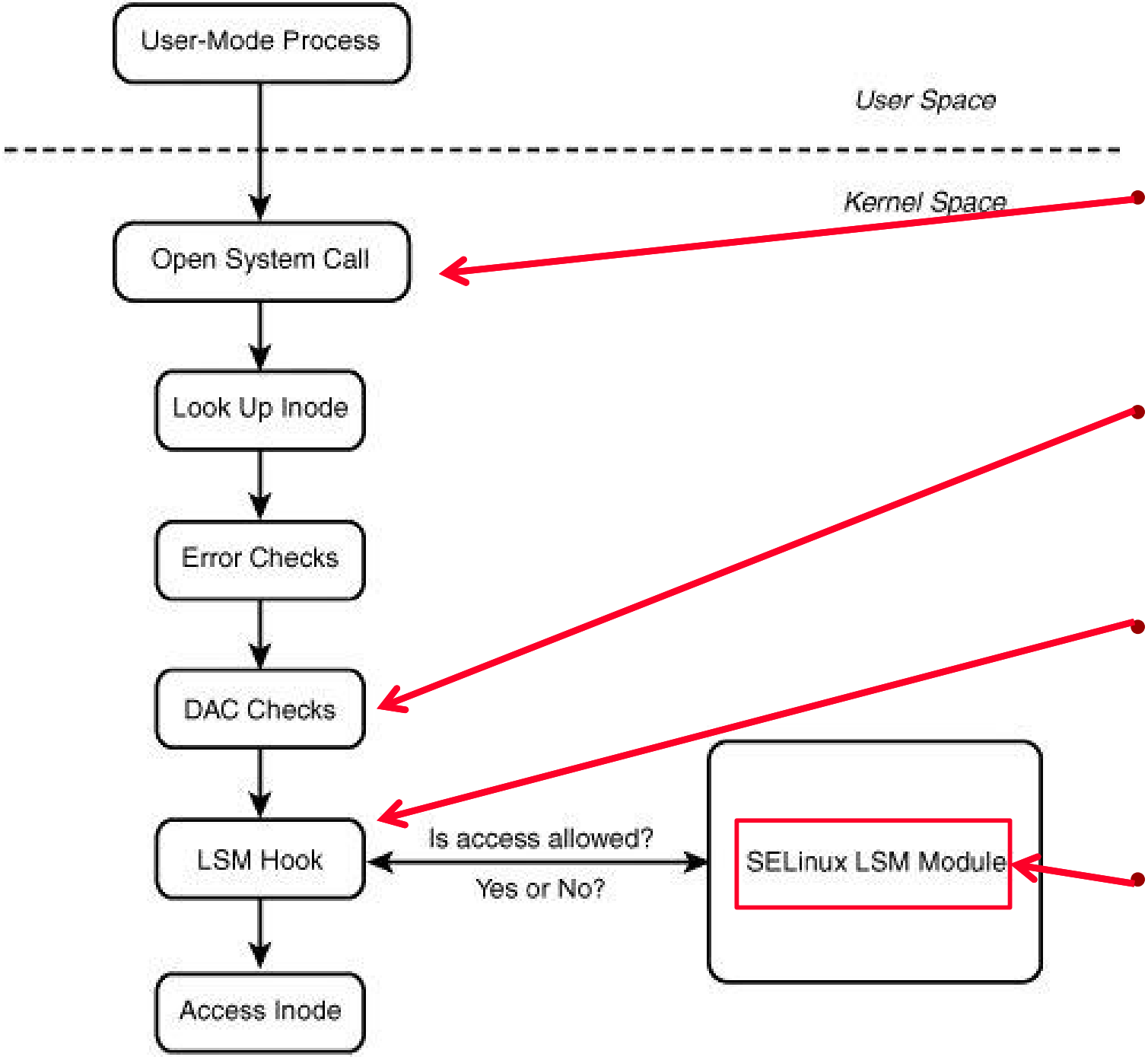
- Démo (Labo 3.1-3.3)

```
id -Z
ls –Z /etc/shadow
ps –eZ
```

- Labo 3.5 : Contexte de sécurité associé à un port TCP

- Attention de conserver ce *file labelling* lors d'une sauvegarde

SELinux Architecture



Appels système : ioctl read, …

Discretionary Access Control

Linux Security Module - Linus Torvalds

SELinux rules

Acces Vector Cache

`avstat`

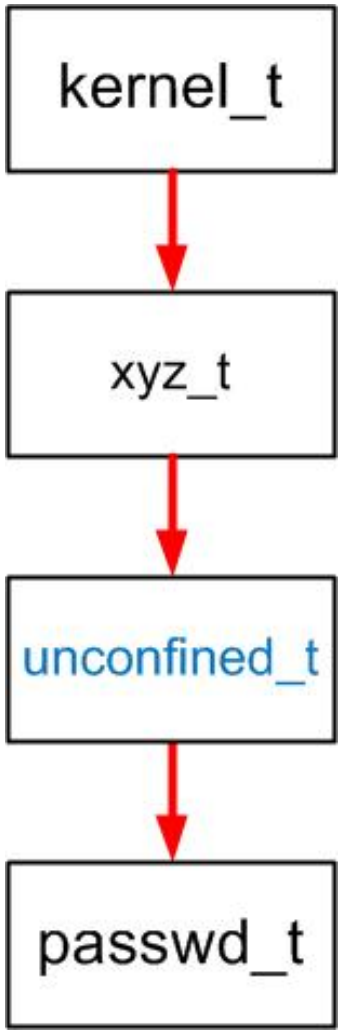
Problématique

- Sécuriser un système selon un modèle liste blanche est une opération gigantesque (… presque impossible)
- Fedora16 comprend 91’628 règles allow
- Fedora16 permet à de très nombreuses applications de fonctionner grâce au **domaine unconfined_t** et la règle

```
allow files_unconfined_type 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 }
```

- Ce mode (laxiste) par défaut est désigné par **Targeted**
- Il offre un modèle de sécurité liste blanche pour des applications critiques comme les mots de passe (passwd_t), un serveur web (httpd_t), …

Domaine unconfined_t



- Démarrage dans le domaine kernel_t
- Transition vers le domaine xyz_t
- Domaine par défaut de l'utilisateur (shell)
  très peu restreint
- Transition dans le domaine passwd_t pour changer de mot de passe
- Domaine restreint à l'usage de passwd (principe du moindre privilège)

Domain transition (Labo 3.4)

- **Domain** = type of a process =
  kernel_t, …, unconfined_t, passwd_t

1 Autoriser le domaine unconfined_t à exécuter un fichier de type passwd_exec_t

allow unconfined_t passwd_exec_t : file {execute}

2 Si 1 → autoriser la transition vers dom passwd_t

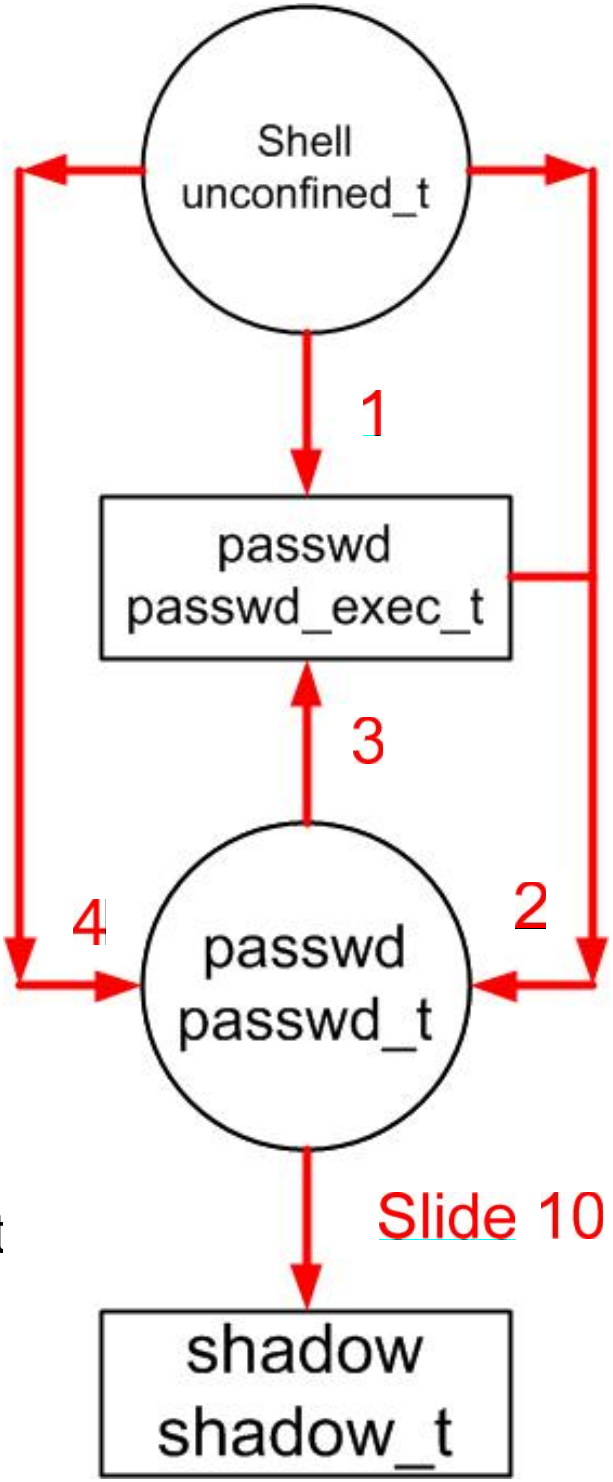
type_transition unconfined_t passwd_exec_t : process passwd_t

3 Autoriser le domaine passwd_t à utiliser un fichier de type passwd_exec_t comme point d'entrée

allow passwd_t passwd_exec_t : file {entrypoint}

4 Autoriser le dom unconfined_t à transiter au dom passwd_t

allow unconfined_t passwd_t : process transition



Recherches du Labo 3.4

- Les outils de recherche (**`sesearch, ...`**) sont essentiels car le nombre de règles est élevé

- Règle **`allow source_t target_t : object_class`**

```
sesearch --allow –s ..... –t ......
sesearch –A     –s ..... –t ......
```

- Règle **`type_transition source_t target_t : process`**

```
sesearch --type      –s ..... –t ......
sesearch –T          –s ..... –t ......
```

Utilisateurs par défaut (Labo 4.2) & prédéfinis (Labo 4.1)

- SELinux ignore les droits des comptes Linux → slide 12
- Il associe ces comptes à des **utilisateurs SELinux**

```
# semanage login -l
Login Name          SELinux User
__default__         unconfined_u
root                unconfined_u
```

- 9 **comptes prédéfinis** → `seinfo -u`

  unconfined_u : compte par défaut dans le domaine unconfined_t

  user_u : compte non privilégié permettant l'établissement d'une session. Pas d'accès aux ports en écoute. Pas de service

  staff_u : comme user_u + accès à setuid et getuid

- Outil **SELinux Management**

Restreindre un utilisateur (Labo 4.3) & comparer (4.4)

- Après authentification (Linux), l'utilisateur **paul** est lié (User Mapping) au compte SELinux **user_u**

- Labo → Switch User
  <Ctrl>+<Alt>+<F1> pour revenir à session 1
  <Ctrl>+<Alt>+<Fn> pour aller dans session n

- Comparaison
  185 règles de transition de domaine pour unconfined_t
  55 ……user_t

Booléens (Labo 5.1)

- Un booléen permet d'activer ou désactiver une(des) règle(s)
  `user_ping` autorise user_u à effectuer un ping
  `allow_user_exec_content` autorise l'exécution d'un application située dans le dossier personnel

- Il évite de devoir modifier une(des) règle(s)

- ~200 booléens à disposition
  Outils **SELinux Management, `semanage boolean -l`**

- Les outils **`audit2why`** et **`audit2allow`** peuvent résoudre un blocage en proposant de modifier un booléen → labo §5.2 – 5.3

Logs (Labo 5.2) & Module (Labo 5.3)

- En cas de refus lors d'une tentative d'accès, SELinux l'enregistre dans les logs de AVC (Access Vector Cache) → slide 12
- Extraire l'élément

```
ausearch -m AVC -c bash -ts today
```

- audit2why donne une explication textuelle du refus

```
ausearch -m AVC ... | audit2why
```

- audit2allow propose une solution que l'admin peut activer

```
ausearch -m AVC ... | audit2allow
```

- audit2allow génère un module

```
ausearch -m AVC ... | audit2allow –M toto
```

- Installer le module

```
semodule -i toto.pp
```

Role Based Access Control – RBAC (Labo 6)

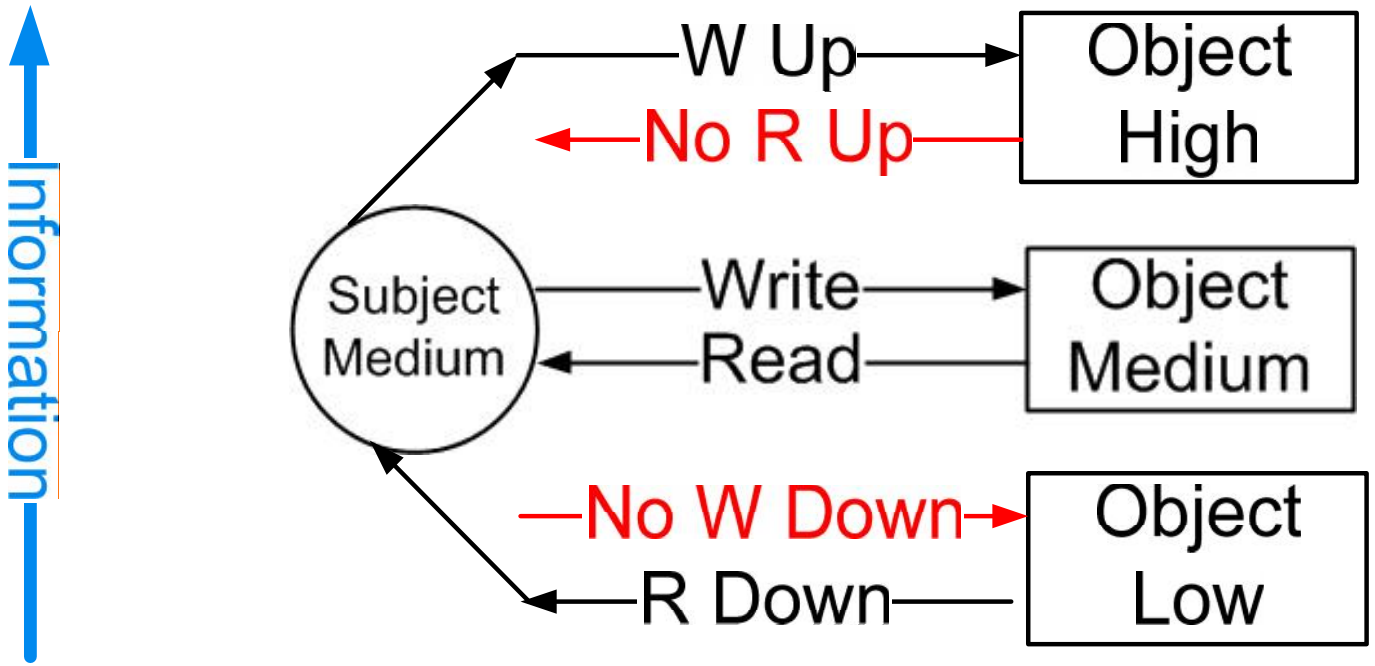
- **`seinfo -r`** → 13 rôles prédéfinis pour autoriser

`system_r`	le lancement de service, …
`webadm_r`	l'administration d'un serveur web
`staff_r`	l'établissement d'une session et l'accès au réseau
`user_r`	l'établissement d'une session et les logiciels sans droit admin (shell, navigateur, …)

- **Scénario : jean doit pouvoir administrer le serveur web**
- Créer **`webadm_u`** possédant les rôles **`staff_r, webadm_r`** et **`system_r`**
- **`semanage user -l`** **relation SELinux User – SELinux Role(s)**
- Lier **jean** à **`webadm_u`**
- Autoriser les actions de **jean** dans le fichier sudoers

Modèle Bell-LaPadulla (confidentialité)

- Illustration dans un contexte militaire
- Classification des données : High – Medium – Low Confidentiality
- **Flux des données**



- SELinux MLS (Multilevel security)

Multi-Category Security – MCS (Labo 7)

- MCS simplifie le modèle précédent en définissant **un seul niveau s0** (sensitivity) et des **catégories non hiérarchiques c0, c1, …**

- **Scénario**

  Le dossier A est réservée à jean

  paul et jean ont accès au dossier B

  → créer 2 catégories c0 c1 et 3 identifiants A B AB

- Assigner une(des) catégorie(s) à jean et paul
- Modifier les catégories de **`webadmin_u`** et **`user_u`**
- Configurer les catégories des dossiers A et B

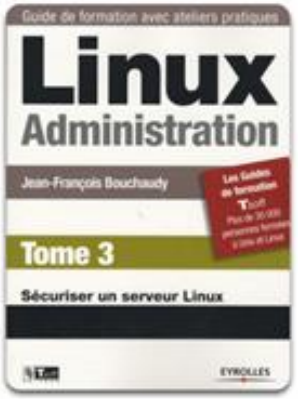
- MCS activé par défaut dans Fedora16
- L'accès aux fichiers reste du type DAC (read – write – execute)

Outils CLI & GUI

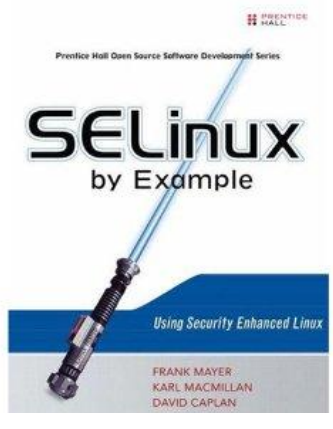
- – Z contexte de sécurité → §3.1 – 3.3
- ausearch recherche dans les logs → §5.2
- chcat change category → §7.3
- chcon change context → §5.4
- seinfo query tool → §4.1, §6.1
- semanage administration → §4.2, §5.1, §6.2
- semodule création module → §5.3
- sesearch recherche dans les règles → §3.4 – 3.5

- **GUI SELinux Management** → §3.5, §4.1-3, §5.1, §6.2, §7.2
- GUI SELinux Audit Log Analysis → §5.2

Références



- Linux Administration Tome 3 par J-F Bouchaudy
- Travaux du labo → http://www.tdeig.ch/SELinux/
  Excellent document Linux_Certif_Comprendre_SELinux.htm
- **https://fedoraproject.org/wiki/SELinux**
- Fedora13 FAQ
  http://docs.fedoraproject.org/en-US/Fedora/13/html/SELinux_FAQ/index.html
- Fedora13 SELinux
  http://docs.fedoraproject.org/en-US/Fedora/13/html/Security-Enhanced_Linux/
- SELinux
  Livre http://flylib.com/books/en/2.803.1.1/1/



Conclusion … et suite

- Selon Bouchaudy 3 : *La maîtrise de la sécurité SELinux échappe en grande partie à l'administrateur. Ce dernier se contente d'appliquer une politique globale*

- Des outils (audit2why-allow) proposent des solutions aux blocages

- Les booléens apportent une certaine souplesse sans exiger l'écriture de règles

- Labo SELinux Exercices pratiques

- Labo ZeroDay Optionnel

  Etude d'une faille de type Reverse Shell présente dans un serveur ftp

  Sécuriser ce serveur ftp avec SELinux