TP1 SELinux

Exercice 1: Information système :

1. Utiliser la commande `id` pour retrouver le contexte SELinux "de l'utilisateur" (en fait, c'est le contexte SELinux du processus qu'on vient de lancer !).
2. Utiliser la commande `ps axZ | less` pour retrouver le contexte SELinux de chaque processus en cours d'exécution. Noter les différences.
3. Utiliser la commande `ls -lZ` pour retrouver le contexte SELinux de chaque fichier dans le répertoire maison, dans les répertoire `/bin`, `/tmp`, dans d'autres répertoires de votre choix. Noter les différences.
4. Utiliser la commande `cp -preserve=context` pour copier des fichiers de votre répertoire maison dans le répertoire `/temp` avec le contexte de sécurité d'origine.
5. Utiliser la commande `chcon` pour changer le contexte de sécurité d'un fichier. Que se passe-t-il si le contexte de sécurité n'existe pas dans la politique ?

(Le dernier composant du contexte de sécurité est le niveau de sécurité.)

Exercice 2: Contextes de sécurité des applications et de leurs fichiers :

1. Vérifier que le serveur Apache est bien démarré. Afficher son contexte de sécurité. Afficher aussi le contexte de sécurité des fichiers dans `/var/www/html`.
2. Créer un fichier dans `/var/www/html` et le récupérer avec `wget` (connexion localhost). Noter le comportement, pour le comparer avec le point suivant.
3. Créer un fichier `test2` dans le répertoire maison puis le copier dans `/var/www/html` en préservant son contexte de sécurité (option `-preserve` de `cp`). Puis essayer de récupérer le fichier avec `wget` – que se passe-t-il ? Afficher le message d'erreur SELinux et l'expliquer.

   Afficher le message d'erreur à l'aide du "SELinux Troubleshooter" et/ou en affichant la fin du logfile `/var/log/audit/audit.log`. (Si auditd pas démarré, chercher dans `/var/log/messages`, pareil pour le SELinux Troubleshooter).
4. Changer (à l'aide de la commande `chcon`) le contexte de sécurité de `test2` dans le contexte nécessaire pour que `httpd` puisse le fournir à `wget`, réessayer et noter le résultat.
5. Remettre le fichier `test2` dans son contexte original, arrêter le `httpd`, modifier le contexte du fichier `/usr/sbin/httpd` en `unconfined_exec_t` et relancer `httpd`. Noter le contexte de `httpd`. Réessayer de récupérer le fichier `test2` à l'aide de `wget` – y a-t-il de différence avec le point 3 ?
6. Changer le contexte de sécurité du volume monté à l'exo 1 en `system_u:object_r:samba_share_t`, à l'aide de l'option `-o defcontext=...` de `mount`, et noter le résultat. Essayer avec autres contextes et essayer ensuite d'afficher les fichiers sur le volume.
7. Essayer les mêmes manipulations avec SELinux permissif – soit à l'aide de la commande `setenforce`, soit en modifiant le fichier `/etc/selinux/config`. Noter les effets sur les messages d'erreur.

Exercice 3: Contextes de sécurité des processus lors des `exec` :

1. Lancer en exécution un petit programme écrit et compilé en C qui fait un `scanf` et, pendant son exécution, afficher son contexte de sécurité. (Vous aurez compris pourquoi il faut un `scanf` et pourquoi ce point ne peut pas être résolu avec des langages comme python ou java !).
2. Lancer le même programme dans le contexte `system_u:system_r:unconfined_t`, à l'aide de la commande `runcon`. Comparer avec le résultat du premier point de l'exo précédent.
3. À l'aide de la commande `semanage`, retrouver les correspondances user-role-domaine qui sont permises dans la politique. Puis essayer de lancer votre exécutable dans des contextes qui ne correspondent pas à ces règles – que se passe-t-il ?
4. Lancer en exécution une commande interactive dans `/bin` ou `/usr/bin` dont l'exécutable est étiquetté avec `bin_t` (genre `rm -i` ou autre). et, pendant son exécution, afficher son contexte.
5. Lancer aussi `passwd` et, pendant son exécution, afficher son contexte – que constate-t-on ? Quel est le contexte de l'exécutable `passwd` ?
6. Essayer de lancer une commande dans le contexte `system_u:system_r:kernel_t:` – que se passe-t-il ?
7. Essayer de lancer `ls` dans le contexte `passwd_t` – y a-t-il des erreurs ?

Pour comprendre ce qui se passe lors des messages de type `avc denied` se rappeler de la notion de transition de type au lancement en exécution d'un programme !

Exercice 4: Les attributs étendus Linux qui permettent au SELinux de gérer les contextes se trouvent dans `/proc/[noprocessus]/attr`

1. Aller dans le pseudo-répertoire du shell (`/proc/self/`) et comparer le contenu du pseudo-fichier `attr/current` avec le résultat de la commande `id -Z`.
2. Essayer de modifier le contexte du shell en `user_u:system_r:unconfined_t:s0` en modifiant le "pseudo-fichier" `attr/current` à l'aide d'`echo` – que se passe-t-il ?

   Essayer aussi avec `system_u:system_r:unconfined_t:s0`.
3. Le contexte des processus fils peut être fixé en modifiant le pseudo-fichier `attr/exec`. Essayer de placer tous les contextes ci-dessus dans ce fichier, et puis essayer de lancer des commandes – que se passe-t-il ? Que faire alors ?
4. Le contexte de fichiers par défaut créés par le shell courant peut être modifiable en modifiant `attr/fscreate`. Modifer ce contexte par défaut en `system_u:object_r:user_home_t:s0` et créer des fichiers dans le répertoire maison (pas dans `/proc/self` !). Regarder le contexte des fichiers créés.
5. Le contexte de sécurité des fichiers peut être aussi retrouvé à l'aide de la commande `getfattr` – comparer son affichage avec celui de `ls -Z`.

Exercice 5: Default user mapping, commande `semanage`, booléens et la modification dynamique de la politique.

1. Afficher le "user mapping" à l'aide de la commande `sudo semanage login -l`, puis la correspondance user-rôle à l'aide de `semanage user -l`. Noter les correspondances.

2. Se connecter avec `ssh` sur la même machine sous différents utilisateurs et noter le contexte du shell créé par ssh. Regarder le contexte affiché dans `attr/prev` – c'est le contexte de qui ? Fermer la session `ssh`.

3. Fermer la connexion ssh, puis, pour un compte `lambda` déjà existant ou nouvellement créé, rajouter une nouvelle connexion par défaut dans `staff_u` à l'aide de `semanage`. Se connecter en ssh sur le compte `lambda` et noter à nouveau le contexte – y a-t-il des changements ? Se déconnecter, essayer de rajouter un autre user-mapping avec `semanage` pour le compte `lambda` – est-ce possible ? Supprimer tous les user-mapping pour `lambda` pour revenir à la situation initiale.

4. Faire de même pour le compte `root` – y a-t-il des modifications ? Ne pas oublier de revenir à la situation initiale !

5. Afficher la valeur des booléens de la politique avec `getsebool -a`, puis la valeur du booléen `ssh_sysadm_login`.

   Les mêmes booléens sont accéssibles dans le pseudo-répertoire `/selinux/booleans`.

6. Modifier le user-mapping pour le compte `lambda` en `sysadm_u` et essayer de se connecter sous ssh – est-ce possible ? Essayer aussi de se connecter sous le rôle `sysadm_r` (commande `ssh -l lambda/sysadm_r localhost`) et noter l'effet.

7. Modifier le booléen `ssh_sysadm_login` et refaire les opérations précédentes – y a-t-il du nouveau ?

8. Si on reboote après avoir fait une modification de booléen comme on l'a fait ci-dessus, la modification ne va pas être prise en compte au redémarrage. La prise en compte au redémarrage se fait si on fait cette modif à l'aide de `system-config-securitylevel`, ou en rajoutant l'option `-P` à `setsebool`.

Les modifications à prendre en compte au redémarrage sont aussi dans `/etc/selinux/targeted/modules/active/booleans.local`.
