Sécurité des infrastructures de virtualisation

SELinux

3e année cycle ingénieur STI, option 2SU, 2018 – 2019

https://tim.siosm.fr/cours

Objectifs

- Découvrir SELinux ;
- Gestion des contextes et des booléens ;
- Confinement des utilisateurs ;
- Ecriture d'un module pour un démon système.

Important : Garder une trace des différentes commandes utilisée et des configurations produites.

1 État de SELinux

Afficher l'état de SELinux et de la politique actuellement chargée :

```
$ sestatus -v
$ cat /etc/selinux/config
$ getenforce
$ seinfo
```

2 Contextes fichiers et processus

- **Question 1:** Sous quel contexte l'utilisateur vagrant s'exécute-il ?
- **Question 2:** Quel sont les contextes associés aux fichiers ou répoertoires suivants :
  - /usr/bin
  - /etc/shadow
  - /etc/password
  - /home
  - /home/vagrant
  - /root
  - /var/www
  - /tmp
- **Question 3:** Quel sont les contextes associés aux processus suivants :
  - démon sshd ;
  - démon httpd ;
  - init.

3 Gestion des booléens

Afficher l'état des booléens

```
$ getsebool -a
```

- Créer le dossier `/home/vagrant/public_html` et un ficher dans ce dossier ;
- Donner le droit à tout le monde de traverser le dossier `/home/vagrant` ;

— Commenter la directive `UserDir disable` et décommenter la directive `UserDir public_html` dans la configuration d'Apache (`/etc/httpd/conf.d/userdir.conf`) ;
— Redémarrer Apache.

Essayer de récupérer le contenu du fichier précedement créé avec la commande `curl` :

```
$ curl http://localhost/~vagrant/test
```

Ne pas oublier de consulter les logs pour obtenir des informations :

```
$ sudo journalctl -e
$ sudo tail /var/log/httpd/error_log
$ sudo tail /var/log/audit/audit.log
```

Chercher si une règle n'existe pas déjà dans la politique pour autoriser cet accès :

```
$ ls -alZ ~/public_html/
$ sesearch -v --allow -s httpd_t -t httpd_user_content_t
```

Afficher toutes les règles ajoutées par ce booléen :

```
$ sesearch --allow -b httpd_enable_homedirs
```

Activer le booléen pour autoriser Apache à lire le contenu du dossier public_html dans les répertoires utilisateurs :

```
$ sudo setsebool httpd_enable_homedirs on
$ curl http://localhost/~vagrant/test
```

4 Confinement d'un utilisateur

Ajouter un utilisateur toto :

```
$ sudo adduser toto
$ sudo passwd toto
```

Lister les associations nom de login -> utilisateur SELinux :

```
$ sudo semanage login --list
```

Lister les associations utilisateurs SELinux -> Rôles :

```
$ sudo semanage user --list
```

Ajouter une association login -> utilisateur SELinux confiné pour toto :

```
$ sudo semanage login --add -s user_u -r s0 toto
```

Vérifier l'ajout de l'association :

```
$ sudo semanage login --list
```

Se loguer sous cet utilisateur (**à distance avec SSH, pas avec `su`**!).

- **Question 4:** Sous quel contexte tourne le shell obtenu ?
- **Question 5:** Essayer de passer `root`. Que se passe-t-il ?

5 Écrire un module pour un démon système

— Importer le script python `pyserver.py` dans le dossier `/usr/local/bin/`
— Importer l'unit systemd `pyserver.service` dans le dossier `/etc/systemd/system/`
— Réstaurer les contextes SELinux par défaut pour ces fichiers :

```
$ sudo restorecon -Fv /usr/local/bin/pyserver /etc/systemd/system/pyserver.service
```

— Recharger la configuration de `systemd` :

```
$ sudo systemctl --daemon-relad
```

— Installer le démon `setroubleshoot` (à ne pas utiliser en production) pour s'aider à comprendre les erreurs SELinux :

```
$ sudo dnf install -y setroubleshoot-server
```

Utiliser sepolgen pour générer un caneva de politique SELinux pour notre démon :

```
$ sepolgen --init /usr/local/bin/pyserver # erreurs non importantes
```

Désactiver les règles `dontaudit` pour obtenir tous les logs d'erreur :

```
$ sudo semodule -DB
```

Compiler le module SELinux et le charger dans le noyau :

```
$ make -f /usr/share/selinux/devel/Makefile pyserver.pp
$ sudo /usr/sbin/semodule -i pyserver.pp
```

Démarrer/Redémarer le démon avec systemd :

```
$ sudo systemctl restart pyserver
```

Tester le serveur :

```
$ curl http://localhost:8000/
```

Regarder les erreurs dans les logs d'audit :

```
$ sudo tail -fn 10 /var/log/audit/audit.log
$ sudo journalctl -e _TRANSPORT=audit
```

Mettre à jour la politique en s'aidant de la commande `audit2allow` :

```
$ sudo cat /var/log/audit.log | audit2allow
```

Répéter les étapes précédentes jusqu'à ce que la politique soit complète.

Une fois terminée, il faut supprimer la directive rendant le domaine permissif pour rendre effective la protection avec SELinux.

Obtenir les sources complètes de la politique SELinux

Pour récupérer les sources de la politique et les recompiler :

```
$ dnf download --source selinux-policy
$ rpm --install selinux-policy-3.13.1-283.19.fc27.src.rpm 2>/dev/null # erreurs non importantes
$ sudo dnf install -y rpmdevtools
$ sudo dnf builddep -y selinux-policy
$ sudo dnf install -y @development-tools
```

```
$ cd ~/rpmbuild/SPECS
$ rpmbuild -bp selinux-policy.spec # erreurs non importantes
$ cd ~/rpmbuild/BUILD/serefpolicy-3.13.1
```

Références

- SELinux project Wiki : https://selinuxproject.org/page/Main_Page
- SELinux Notebook : http://freecomputerbooks.com/The-SELinux-Notebook-The-Foundations.html
- Blog de Dan Walsh : http://danwalsh.livejournal.com/