PROJET PKI

Pour ce projet mettre en place une machine virtuelle Linux. Personnellement je travaille sur une machine virtuelle VirtualBox Ubuntu Serveur avec 2 cartes réseaux en mode NAT (et pas Réseau NAT). Si vous souhaitez savoir comment créer une machine sur VirtualBox cf. ce document (même si depuis la version de VirtualBox a changé).

Tous les fichiers sont disponibles dans mon dossier Git dans le dossier 00-Code. Tous les scripts sont à exécuter en `sudo`.

Vidéo explicative du projet dans le fichier 02-Video.pdf, le sommaire est disponible en description de la vidéo si vous souhaitez regarder que certaines parties.

Ordre d'exécution des scripts :

1- *A la main : faire l'étape I de ce document.*
2- creation_conteneur.sh *nbr_clients*
3- creation_data.sh
4- installation_mqtt.sh
5- paho_sur_lxc.sh
6- start_ecoute_mqtt
7- certificat_pki (installe les paquets, exécute le script python qui permet de générer le certificat sur le serveur et copie ce certificat sur les clients)
8- start_echange qui appelle echange_sym

I. Création des conteneurs (Client + Serveur)

J'ai décidé de mettre en place un script Bash qui permet de créer de 2 à 8 clients plus un serveur LXC Ubuntu. Le script prend un seul paramètre, n, le nombre de clients.

1. Mise en place du réseau

Ces entités vont communiquer à l'aide d'un bridge. On attribue une carte réseau de l'hôte à ce bridge.

a. Création du bridge

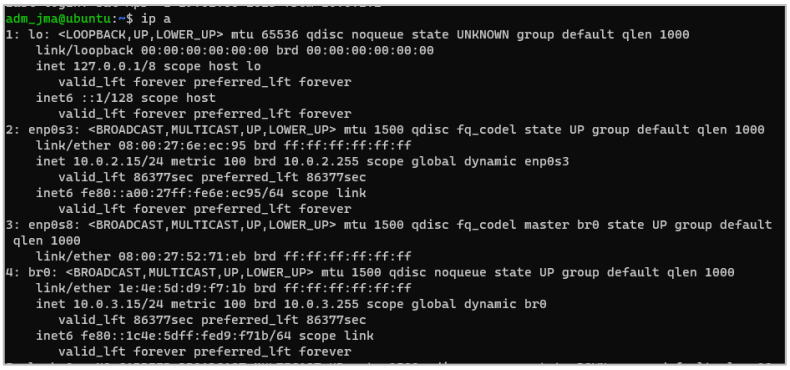
(Attention à l'indentation, la copie depuis ce document peut générer des problèmes)

```
nano /etc/netplan/*.yaml
network:
  ethernets:
    enp0s3:
      dhcp4: true
    enp0s8:
       dhcp4:false

  bridges:
    br0:
      dhcp4: true
      interfaces:
       -enps0s8
  version: 2
```

`netplan apply`

Pour vérifier les configurations rentrer la commande « `ip a` ». La seconde carte réseau ne devrait plus apparaître puisqu'elle est « absorbée » par le bridge `br0`. Voici ce que j'ai obtenu :



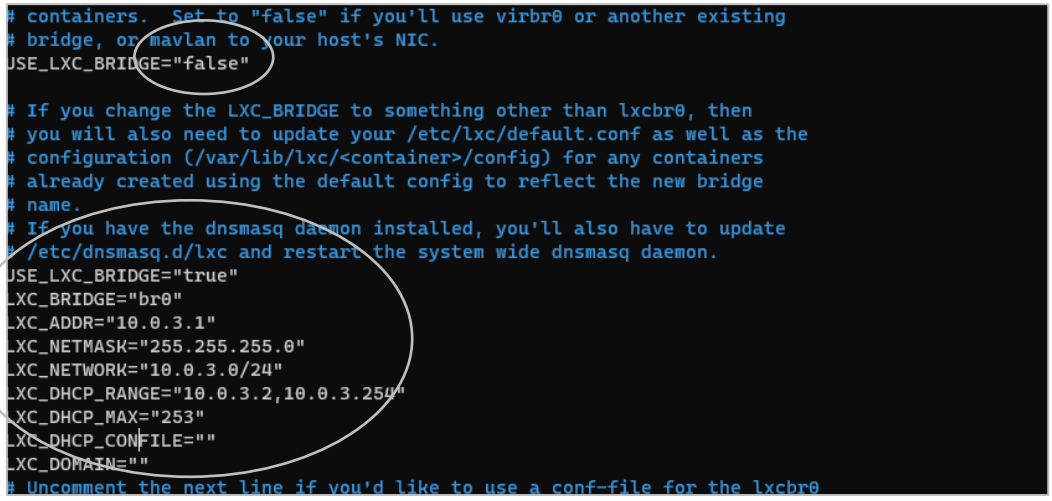
b. Téléchargement de LXC

```
apt-get update
apt install lxc lxc-templates -y
```

Attention ! Les scipts suivants supposent que vous n'avez pas d'autres conteneurs de créés.

c. Configuration de LXC sur l'hôte

Changer le fichier de configuration « `etc/default/lxc-net` » pour attribuer notre bridge `br0` aux conteneurs. Voici à quoi le fichier de configuration devrait ressemble après modifications :



Il faut également modifier le fichier « » ainsi :

2. Création du script de création des conteneurs

`./creation_conteneur.sh` *`nombre_clients`* : permet de créer les conteneurs LXC Ubuntu.
`./creation_data.sh` : permet de créer un fichier par client/serveur qui comprend l'adresse IP du conteneur et son nom.

Vous trouverez dans le dossier `src/debug` des fichiers bash permettant d'aider les tests. Ils sont appelés par les scripts du dossier `src`. Ce sont un peu des « fonctions » permettant d'alléger les fichiers principaux.

3. Mise en place des échanges par MQTT

`./installation_mqtt.sh` : met à jour les conteneurs et télécharges les modules nécessaires à la mise en place de MQTT. La librairie choisie est Mosquitto en Python.

Puis création des script Python « abonné » et « éditeur » dans le dossier `src_py`. Utilisation de la librairie `Paho` qui permet de mettre en place les échanges MQTT en Python. On copie ces deux scripts sur les conteneurs avec le script `paho_sur_lxc.sh`. Puisqu'il n'est pas possible de récupérer l'expéditeur d'un message en MQTT les messages envoyés sont des JSON. Une partie du message est le nom de l'expéditeur, l'autre le corps du message. Pour la suite du projet nous allons donc seulement chiffrer la deuxième partie du message.

II. PKI

1. Le serveur produit un certificat autosigné X509

Nous utilisons la classe pyca/cryptography de python pour générer le certificat autosigné X509.

Le script `certificat_pki.sh` installe les librairies nécessaires, génère un certificat auto-signé X509 sur le serveur (la PKI) qu'on copie en dure sur les clients avec sa clef publique.
Les clients créent leur propre clé privée et leur propre demande de signature de certificat (CSR). Ils ne révèlent pas leur clé privée à l'autorité de certification (PKI).
Les clients remettent leur CSR à la PKI et celle-ci leurs renvoient un certificat signé. L'envoie de ce fichier se fait toujours dans le script : `certificat_pki.sh`. La PKI réceptionne les CSR et génère donc les certificats. Les certificats ont généralement une durée de validité d'un an, bien qu'une autorité de certification accorde généralement quelques jours supplémentaires pour des raisons de commodité.
Le serveur renvoie donc le certificat signé ainsi que la signature aux clients. Les clients stockent leur certificat signé ainsi que leur signature dans leur dossier « `secure` ».

2. Echanges

Les échanges sont générés aléatoirement grâce au script `choix_msg.py`.

a. Si les clients ne se connaissent pas encore

Echange asymétrique

Celui qui initie l'échange envoie sa signature et son certificat à son destinataire.
Le destinataire vérifie la validité du certificat qu'il reçoit et extrait la clef publique.
Il génère ensuite un secret (mot de 32 bits, encrypté avec AES 128 et la clef publique de l'expéditeur) qu'il envoie à son expéditeur.
Ce dernier déchiffre ce secret et envoie un message d'acquittement à son destinataire pour le prévenir qu'il a bien récupérer le secret.
Les échanges symétriques sont désormais disponibles.

b. Les clients se connaissent

Echange symétrique

Le secret est partagé entre les deux clients, le fichier `SYM_OK.txt` est présent dans leur dossier `secure`. L'expéditeur a seulement besoin de crypter son message avec le secret qu'il partage avec son destinataire. Il l'envoie et son destinataire le décrypte de son côté.

L'historique des échanges des messages est stocké dans le fichier `historique.txt` sur chaque client.