IPSec - Partie 2

Version du 19 septembre 2024, 10:19

Objectifs

> *L'objectif de ce TP est de réaliser des tunnels IPSec en utilisant IKE grâce à StrongSwan.*

1 Des tunnels avec IKE

1.1 Préparation

Dans le cadre de cette seconde partie nous allons repartir de la même architecture que pour la première partie. Vous pouvez donc réutiliser le script `buil_architecture` fourni lors du premier TP.

Nous aurons également besoin du packetage `strongswan` que vous pouvez installer par apt avec quelques packages complémentaires.

```
$ sudo apt install strongswan strongswan-pki libcharon-extra-plugins \
libcharon-extauth-plugins strongswan-swanctl charon-systemd
```

Vous aurez également besoin de configurer une petite pki pour mettre en place des certificats entre les deux machines $h1$ et $h2$.

1.2 Mise en place de la PKI et génération des certificats

StrongSwan a l'avantage de venir avec une petite PKI intégrée dont nous allons nous servir pour générer les certificats utilisés par les deux extrémités de notre tunnel VPN.

La première étape consiste à construire la PKI à l'aide des commandes suivantes :

```
$ mkdir -p pki/{cacerts,certs,private}
$ pki --gen --type rsa --size 4096 --outform pem > pki/private/ca-key.pem
$ pki --self --ca --lifetime 3650 --in pki/private/ca-key.pem \
    --type rsa --dn "CN=VPN root CA" --outform pem > pki/cacerts/ca-cert.pem
```

La première lignes consiste à créer un répertoire *pki* dans votre répertoire courant, vous pouvez bien sur adapter les commandes en fonction de vos besoins.

Ces commandes vous permettent d'obtenir un certificat de l'autorité (`ca-cert.pem`) et une clé privée RSA de 4096bit pour cette autorité (`ca-key.pem`).

Le certificat `ca-cert.pem` doit être déployé sur $h1$ et sur $h2$ dans le répertoire `/etc/swanctl/x509ca/`. Si vous travaillez sur la solution à base de netns, vous devez donc les déployer dans les répertoire de chaque terminaux (exemple : `/etc/netns/h1/swanctl/x509ca/` pour $h1$)

Une fois la PKI initialisée, vous pouvez l'utiliser pour générer vos certificats et en particulier dans le cas qui nous intéresse, le certificat qui servira pour notre hote $h1$ (ip : 10.10.10.1). Dans un premier temps commençons par générer une clé secrète pour $h1$:

```
pki --gen --type rsa --size 4096 --outform pem > pki/private/h1Key.pem
```

Cette clé devra être placé dans le répertoire `/etc/swanctl/private/` (ou pour netns dans `/etc/netns/h1/swanctl/private` pour $h1$)

Cette clé nommée $h1Key.pem$ sera utilisée pour la mise en place de tunnel. On génère ensuite le certificat de $h1$

```
pki --pub --in pki/private/h1Key.pem --type rsa | pki --issue \
--cacert pki/cacerts/ca-cert.pem --cakey pki/private/ca-key.pem \
--lifetime 1825 --dn "CN=10.10.10.1" --san 10.10.10.1 \
--san @10.10.10.1 --outform pem > pki/certs/h1.pem
```

Le certificat doit être placé dans : `/etc/swanctl/x509/` (ou pour netns dans `/etc/netns/h1/swanctl/x509/` pour $h1$)

Refaite ensuite les mêmes opérations pour générer les clés et certificats de $h2$ et placez les dans les répertoires adaptés.

1.3 Configuration des deux extrémités d'un tunnel host-to-host avec IKE

Sur $h1$ vous devez ensuite configurer la partie StrongSwan pour définir les algorithmes de chiffrement à utiliser ainsi que l'autre extrémité du tunnel (ici 10.0.0.1).

Vous devez ensuite configurez le fichier `/etc/swanctl/swanctl.conf` (ou pour netns le fichier `/etc/netns/h1/swanctl/swanctl.conf` pour $h1$) de la manière suivante :

```
connections {
        host-host {
            remote_addrs = 10.0.0.1

            local {
                auth=pubkey
```

```
            certs = h1.pem
        }
        remote {
            auth = pubkey
            id = "CN=10.0.0.1"
        }
        children {
            net-net {
                start_action = trap
            }
        }
    }
}
```

Faites ensuite la même manipulation coté $h2$ en changeant les adresses IP remote par celle de $h1$.

Une fois les fichiers de configurations et les certificats correctement positionnés vous pouvez mettre en place votre tunnel. Pour les personnes qui travaillent en netns, il est nécessaire de faire une petite modification supplémentaire pour permettre aux différents démons de strongswan de s'exécuter en parallèle.

Pour les netns (les killall ne sont à faire que sur $h1$) :

```
$ sudo ip netns exec h1 bash
$ sudo killall charon
$ sudo killall starter
$ sudo mkdir /tmp/h1
$ sudo mount --bin /tmp/h1 /run
$ sudo ipsec start
$ swanctl --load-creds
$ swanctl --load-conns
```

Refaite la même chose sur $h2$ (sauf la partie killall). Remarque : Les répertoire /tmp/h1 et /tmp/h2 ne sont à créer que la première fois sur chacun des deux hôtes. Si vous relancer les netns ensuite, vous n'avez que le bind à faire.

La commande `swanctl -load-creds` vous permet de charger les certificats. Elle vous renvoie normalement les emplacements des fichiers chargés. La commande `swanctl -load-conns` vous permet de charger la connexion. Elle vous affiche en retour, si la mise en place de la connexion s'est bien passée ou non.

Une fois votre configuration terminée, faites un ping entre $h1$ et $h2$ et capturez le traffic pour visualiser les échanges IKE.

Si l'échange ping ne fonctionne pas, vous pouvez visualiser la négociations IKE grâce à la commande `swanctl -log` sur $h1$ et sur $h2$. Le premier passage d'un paquet vous affichera la négociation IKE et vous indiquera les potentielles erreurs que vous avez. Attention, ça ne fonctionne qu'au premier paquet échangé.

1.4 Mise en place d'un tunnel IPSec entre deux sites

Ce tunnel sera mis en place entre les routeurs $r1$ et $r2$ et dans un premier temps il va être nécessaire de générer des clés privées RSA et des certifcats pour ces deux routeurs sur le modèle de ce qui a été fait dans la section précédente.

Une fois les certifcats et clés créés, voyons la configuration de la connexion entre $r1$ et $r2$ en prenant l'exemple du fichier `swanctl.conf` de $r1$:

```
connections {
    net-net {
        remote_addrs = 172.16.1.2

        local {
            auth = pubkey
            certs = r1.pem
        }
        remote {
            auth = pubkey
            id = "CN=172.16.1.2"
        }
        children {
            net-net {
                local_ts  = 10.10.10.0/24
                remote_ts = 10.0.0.0/24
                start_action = trap
            }
        }
    }
}
```

Déployez cette configuration sur $r1$ et $r2$ et faites des tests entre $h1$ et $h2$ pour voir si la communication fonctionne et si vous arrivez à visualiser les échanges IKE et ESP. Vous pouvez ensuite remettre en place votre tunnel entre $h1$ et $h2$ pour visualiser la combinaison de tunnels.

1.5 Configuration Road Warrior

Vous pouvez ensuite consultez le github de StrongSwan pour tester des configurations différentes comme le mode EAP ou le mode Road Warrior : `https://github.com/strongswan/strongswan`