Master 2 SeCReTS
Concepts Sécurité et Réseaux
2014-2015

TP Sécurité Réseau : IPSec

Table des matières

Première partie
Généralités

1 Objectifs du TP

L'objectif de ce TP est de manipuler IPSec dans ses différents modes, et de mettre en application certaines des fonctionnalités vues en cours. En particulier, on utilisera les protocoles AH et ESP avec l'utilisation de clef partagées et de certificats. L'objectif final est de configurer IPSec en mode tunnel afin de permettre à un client d'accéder de manière sécurisée aux conteneurs LXC mis en place dans le TP précédent.

Ces manipulations se feront à l'aide de deux machines virtuelles.

2 Pré-requis

Pour cette première partie Vous devez utiliser **deux machines virtuelles** (celles du TP précédent peuvent convenir).

Les deux machines virtuelles (`vmclient` et `vmserver`) doivent avoir chacune une interface réseau rattachée à un réseau interne (« *intnet* » sous VirtualBox). Les adresses à configurer sur ce réseau sont :

- 192.168.155.1 pour `vmclient`
- 192.168.155.2 pour `vmserver`
- 255.255.255.0 est le masque de sous réseau à utiliser

Installez les packages *racoon* et *ipsec-tools* sur les deux machines.

#/etc/network/interfaces
auto ethX
iface ethX inet static
address 192.168.155.x
netmask 255.255.255.240

#/etc/hosts
192.168.155.1 vmclient
192.168.155.2 vmserveur

apt-get install ipsec-tools racoon

Deuxième partie

IPSec en mode transport

1 ESP et AH : paramétrage statique

On souhaite dans un premier temps que tout le trafic ICMP qui transite entre nos deux machines virtuelles sur le réseau interne soit chiffré avec l'algorithme *des-cbc*.

Écrire un fichier de règles pour configurer la SAD et la SPD et décrivez le.

```
# vmclient :
## Configuration de la SPD :
spdadd 192.168.155.1 192.168.155.2 icmp -P out ipsec esp/transport//require ;
spdadd 192.168.155.2 192.168.155.1 icmp -P in ipsec esp/transport//require ;
## Configuration de la SAD :
add 192.168.155.1 192.168.155.2 esp 0x301 -m transport -E des-cbc "12345678" ;
add 192.168.155.2 192.168.155.1 esp 0x302 -m transport -E des-cbc "12345678" ;

# vmserver :
## Configuration de la SPD :
spdadd 192.168.155.2 192.168.155.1 icmp -P out ipsec esp/transport//require ;
spdadd 192.168.155.1 192.168.155.2 icmp -P in ipsec esp/transport//require ;
## Configuration de la SAD :
add 192.168.155.1 192.168.155.2 esp 0x301 -m transport -E des-cbc "12345678" ;
add 192.168.155.2 192.168.155.1 esp 0x302 -m transport -E des-cbc "12345678" ;
```

Utilisez la commande *setkey* pour charger les règles et validez avec cette même commande que tout s'est bien déroulé.

```
setkey -f ipsec.conf
setkey -D
setkey -DP
```

- A l'aide d'une capture réseau, mettre en évidence que le trafic ICMP est bien chiffré
- Adaptez les règles précédentes pour que la clef utilisée pour la communication de `vmclient` vers `vmserver` soit différente que pour la communication de `vmserver` vers `vmclient`
- Que faut il faire avant de charger les nouvelles SA ?
- Authentifiez les paquets ESP

```
# vmclient :
flush ;
spdflush ;
spdadd 192.168.155.1 192.168.155.2 icmp -P out ipsec esp/transport//require ;
spdadd 192.168.155.2 192.168.155.1 icmp -P in ipsec esp/transport//require ;
add 192.168.155.1 192.168.155.2 esp 0x303 -m transport -E des-cbc "22345678" -A
hmac-md5 "2234567890123456" ;
add 192.168.155.2 192.168.155.1 esp 0x304 -m transport -E des-cbc "12345678" -A
hmac-md5 "1234567890123456" ;

# vmserver :
flush ;
spdflush ;
spdadd 192.168.155.1 192.168.155.2 icmp -P in ipsec esp/transport//require ;
```

```
spdadd 192.168.155.2 192.168.155.1 icmp -P out ipsec esp/transport//require;
add 192.168.155.1 192.168.155.2 esp 0x303 -m transport -E des-cbc "22345678" -A
hmac-md5 "2234567890123456";
add 192.168.155.2 192.168.155.1 esp 0x304 -m transport -E des-cbc "12345678" -A
hmac-md5 "1234567890123456";
```

Choisir des algorithmes de chiffrement et d'authentification plus robustes et activez le chiffrement pour tout type de protocole.

```
# vmclient :
flush;
spdflush;
spdadd 192.168.155.1 192.168.155.2 any -P out ipsec esp/transport//require;
spdadd 192.168.155.2 192.168.155.1 any -P in ipsec esp/transport//require;
add 192.168.155.1 192.168.155.2 esp 0x305 -m transport -E aes-cbc "2234567890123456" -A
hmac-sha1 "22345678901234567890";
add 192.168.155.2 192.168.155.1 esp 0x306 -m transport -E aes-cbc "1234567890123456" -A
hmac-sha1 "12345678901234567890";

# vmserver :
flush;
spdflush;
spdadd 192.168.155.1 192.168.155.2 any -P in ipsec esp/transport//require;
spdadd 192.168.155.2 192.168.155.1 any -P out ipsec esp/transport//require;
add 192.168.155.1 192.168.155.2 esp 0x305 -m transport -E aes-cbc "2234567890123456" -A
hmac-sha1 "22345678901234567890";
add 192.168.155.2 192.168.155.1 esp 0x306 -m transport -E aes-cbc "1234567890123456" -A
hmac-sha1 "12345678901234567890";
```

Adapter les règles pour ne faire que de l'authentification et constatez le résultat à l'aide d'une capture réseau.

```
# vmclient :
flush;
spdflush;
spdadd 192.168.155.1 192.168.155.2 any -P out ipsec ah/transport//require;
spdadd 192.168.155.2 192.168.155.1 any -P in ipsec ah/transport//require;
add 192.168.155.1 192.168.155.2 ah 0x307 -m transport -A hmac-sha1 "22345678901234567890";
add 192.168.155.2 192.168.155.1 ah 0x308 -m transport -A hmac-sha1 "12345678901234567890";

# vmserver :
flush;
spdflush;
spdadd 192.168.155.1 192.168.155.2 any -P in ipsec ah/transport//require;
spdadd 192.168.155.2 192.168.155.1 any -P out ipsec ah/transport//require;
add 192.168.155.1 192.168.155.2 ah 0x307 -m transport -A hmac-sha1 "22345678901234567890";
add 192.168.155.2 192.168.155.1 ah 0x308 -m transport -A hmac-sha1 "12345678901234567890";
```

Que faut il faire pour que les règles soient ajoutées dès le démarrage de la machine ?

```
/etc/ipsec-tools.d/FILENAME.conf
```

2 Paramétrage dynamique : *Racoon*

2.1 Clefs partagées

Après avoir installé *Racoon* sur les deux machines virtuelles, éditez son fichier de configuration (`/etc/racoon/racoon.conf`) sur `vmclient` comme indiqué ci-dessous :

```
log notify;
path pre_shared_key "/etc/racoon/psk.txt";
remote 192.168.155.2 {
    exchange_mode main;
    proposal {
        encryption_algorithm aes;
        hash_algorithm sha1;
        authentication_method pre_shared_key ;
        dh_group 2;
    }
}
sainfo anonymous {
    pfs_group modp1024;
    encryption_algorithm aes;
    authentication_algorithm hmac_sha1;
    compression_algorithm deflate;
}
```

Que faut il mettre dans le fichier `/etc/racoon/psk.txt` et quelle syntaxe utiliser ?

```
# vmclient :
192.168.155.2 ipseckey
```

Recopier et adapter la configuration de *Racoon* sur `vmserver` puis démarrer le démon de chaque coté.

```
log notify;
path pre_shared_key "/etc/racoon/psk.txt";
remote 192.168.155.1 {
exchange_mode main;
proposal {
encryption_algorithm aes;
hash_algorithm sha1;
authentication_method pre_shared_key;
dh_group 2;
}
}
sainfo anonymous {
pfs_group modp1024;
encryption_algorithm aes;
authentication_algorithm hmac_sha1;
compression_algorithm deflate;
}

# vmserver :
192.168.155.1 ipseckey
```

Configurez ensuite la SPD de chaque machine virtuelle pour chiffrer les communications en utilisant IPSec en mode transport. Valider que les échanges sont bien chiffrés.

```
## vmclient :
spdadd 192.168.155.1 192.168.155.2 any -P out ipsec esp/transport//require;
spdadd 192.168.155.2 192.168.155.1 any -P in ipsec esp/transport//require;

## vmserver :
spdadd 192.168.155.2 192.168.155.1 any -P out ipsec esp/transport//require;
spdadd 192.168.155.1 192.168.155.2 any -P in ipsec esp/transport//require;
```

A partir d'une capture réseau, utilisez *Wireshark* pour déchiffrer les communications.

2.2 Création d'une PKI

Utiliser les scripts `easy-rsa` présents sur `vmserver` pour créer :

- Une authorité de certification
- Un certificat serveur pour vmclient.test.org
- Un certificat serveur pour vmserver.test.org
- La `crl` associée

```
source ./vars
./clean-all
./build-ca
./build-key-server vmclient.test.org
./build-key-server vmserver.test.org
```

2.3 Utilisation des certificats

Adaptez la configuration de *racoon* pour remplacer l'utilisation de clefs partagés par des certificats.

```
log notify;
path certificate "/etc/racoon/certs";
remote 192.168.155.2 {
    exchange_mode main;
    verify_cert on;
    my_identifier asn1dn;
    certificate_type x509 "vmclient.test.org.crt" "vmclient.test.org.key";
    proposal {
        encryption_algorithm aes;
        hash_algorithm sha1;
        authentication_method rsasig;
        dh_group 2;
    }
    proposal_check obey;
}
sainfo anonymous {
    pfs_group modp1024;
    encryption_algorithm aes;
    authentication_algorithm hmac_sha1;
```

```
    compression_algorithm deflate;
}
```

```
cp ca.crt vmclient.test.org.crt vmclient.test.org.key /etc/racoon/certs
cd /etc/racoon/certs
ln -s ca.crt `openssl x509 -hash -noout -in ca.crt`.0
```

Troisième partie

IPSec en mode tunnel

Refaire la même chose qu'à l'étape précédente mais en mode tunnel afin d'accéder aux conteneurs LXC depuis `vmclient`. Quelles sont les règles à ajouter à la SPD ?

```
# vmclient :
spdadd 192.168.155.1 10.0.3.0/24 any -P out ipsec
esp/tunnel/192.168.155.1-192.168.155.2/require ;
spdadd 10.0.3.0/24 192.168.155.1 any -P in ipsec
esp/tunnel/192.168.155.2-192.168.155.1/require ;

route add -net 10.0.3.0/24 gw 192.168.155.2

# vmserver :
spdadd 192.168.155.1 10.0.3.0/24 any -P out ipsec
esp/tunnel/192.168.155.1-192.168.155.2/require ;
spdadd 10.0.3.0/24 192.168.155.1 any -P in ipsec
esp/tunnel/192.168.155.2-192.168.155.1/require ;

sysctl -w net.ipv4.ip_forwarding=1
```

Vous aurez dans cette partie à ajouter des règles de routage afin d'accéder aux services conteneurisé sur `vmserver` depuis `vmclient`.

Faire un schéma de l'architecture ainsi réalisée.

Quatrième partie

Pour les plus rapides

Vérifier l'authenticité d'un paquet et déchiffrez le à l'aide d'un script python. Pour plus de facilité vous pourrez vous remettre dans la situation II.1.

```
# Vérification d'authenticité
from Crypto.Hash import HMAC, MD5
key = "31323334353637383930313233343536".decode("hex")
esp = "000100xxxxx".decode("hex")
mac = HMAC.new(key, digestmod=MD5)
mac.update(esp)
print mac.hexdigest()[:24]

from Crypto.Hash import HMAC, SHA
data = "8551..."
key = "32323334353637383930313233343536".decode("hex")
iv = data[:32].decode("hex")
data_cipher = data[32 :].decode("hex")
aes1 = AES.new(key, AES.MODE_CBC, iv)
print aes1.decrypt(data_cipher)

# Déchiffrement DES
from Crypto.Cipher import DES
key = "3132333435363738".decode("hex")
data = "8b0d5e..."
data_cipher = data[16 :].decode("hex")
iv = data[:16].decode("hex")
des1 = DES.new(key, DES.MODE_CBC, iv)
result = des1.decrypt(data_cipher)
print result.encode("hex")

# Déchiffrement AES
from Crypto.Cipher import AES
key = "31323334353637383930313233343536".decode("hex")
data = "954fff.."
iv = data[:32].decode("hex")
data_cipher = data[32 :].decode("hex")
aes1 = AES.new(key, AES.MODE_CBC, iv)
print aes1.decrypt(data_cipher)
```