Master 2 SeCReTS
Concepts Sécurité et Réseaux
2016-2017

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 clefs partagées et de certificats. L'objectif final est de configurer IPSec en mode tunnel afin de permettre à un client (client1) d'accéder de manière sécurisée à un autre client (client2).

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

2 Pré-requis

Pour cette première partie Vous devez utiliser **4 machines virtuelles** (celles du TP précédent peuvent convenir).

Les deux machines virtuelles (`gateway1` et `gateway2`) doivent avoir chacune une interface réseau rattachée à un réseau interne (« *intnet0* » sous VirtualBox). Les adresses à configurer sur ce réseau sont :

- 10.1.0.1 pour `gateway1`
- 10.1.0.2 pour `gateway2`
- 255.255.255.0 est le masque de sous réseau à utiliser

La machine `client1` doit être reliée via le réseau « *intnet1* » à la machine `gateway1` avec les adresses suivantes :

- 10.1.1.1 pour `gateway1`
- 10.1.1.2 pour `client1`
- 255.255.255.0 est le masque de sous réseau à utiliser

La machine `client2` doit être reliée via le réseau « *intnet2* » à la machine `gateway2` avec les adresses suivantes :

- 10.1.2.1 pour `gateway2`
- 10.1.2.2 pour `client2`
- 255.255.255.0 est le masque de sous réseau à utiliser

Installez les packages *racoon* et *ipsec-tools* sur les deux passerelles via la commande `dpkg`, les packages se trouvant dans le repertoire `/root/packages`.

```
#/etc/network/interfaces
auto ethX
iface ethX inet static
address 10.1.1.x
netmask 255.255.255.0

#/etc/hosts
10.1.1.1 gateway1
10.1.1.2 client1

apt-get install ipsec-tools racoon
ou dpkg -i ipsec-tools*.deb racoon*.deb
```

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 passerelles sur le réseau interne `intnet0` soit chiffré avec l'algorithme *des-cbc*.

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

```
# gateway1 :
## Configuration de la SPD :
spdadd 10.1.0.1 10.1.0.2 icmp -P out ipsec esp/transport//require ;
spdadd 10.1.0.2 10.1.0.1 icmp -P in ipsec esp/transport//require ;
## Configuration de la SAD :
add 10.1.0.1 10.1.0.2 esp 0x301 -m transport -E des-cbc "12345678" ;
add 10.1.0.2 10.1.0.1 esp 0x302 -m transport -E des-cbc "12345678" ;

# gateway2 :
## Configuration de la SPD :
spdadd 10.1.0.2 10.1.0.1 icmp -P out ipsec esp/transport//require ;
spdadd 10.1.0.1 10.1.0.2 icmp -P in ipsec esp/transport//require ;
## Configuration de la SAD :
add 10.1.0.1 10.1.0.2 esp 0x301 -m transport -E des-cbc "12345678" ;
add 10.1.0.2 10.1.0.1 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 `gateway1` vers `gateway2` soit différente que pour la communication de `gateway2` vers `gateway1`
- Que faut il faire avant de charger les nouvelles SA ?
- Authentifiez les paquets ESP

```
# gateway1 :
flush ;
spdflush ;
spdadd 10.1.0.1 10.1.0.2 icmp -P out ipsec esp/transport//require ;
spdadd 10.1.0.2 10.1.0.1 icmp -P in ipsec esp/transport//require ;
add 10.1.0.1 10.1.0.2 esp 0x303 -m transport -E des-cbc "22345678" -A
hmac-md5 "2234567890123456" ;
add 10.1.0.2 10.1.0.1 esp 0x304 -m transport -E des-cbc "12345678" -A
hmac-md5 "1234567890123456" ;

# gateway2 :
flush ;
spdflush ;
spdadd 10.1.0.1 10.1.0.2 icmp -P in ipsec esp/transport//require ;
```

```
spdadd 10.1.0.2 10.1.0.1 icmp -P out ipsec esp/transport//require;
add 10.1.0.1 10.1.0.2 esp 0x303 -m transport -E des-cbc "22345678" -A
hmac-md5 "2234567890123456";
add 10.1.0.2 10.1.0.1 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.

```
# gateway1 :
flush;
spdflush;
spdadd 10.1.0.1 10.1.0.2 any -P out ipsec esp/transport//require;
spdadd 10.1.0.2 10.1.0.1 any -P in ipsec esp/transport//require;
add 10.1.0.1 10.1.0.2 esp 0x305 -m transport -E aes-cbc "2234567890123456" -A
hmac-sha1 "22345678901234567890";
add 10.1.0.2 10.1.0.1 esp 0x306 -m transport -E aes-cbc "1234567890123456" -A
hmac-sha1 "12345678901234567890";

# gateway2 :
flush;
spdflush;
spdadd 10.1.0.1 10.1.0.2 any -P in ipsec esp/transport//require;
spdadd 10.1.0.2 10.1.0.1 any -P out ipsec esp/transport//require;
add 10.1.0.1 10.1.0.2 esp 0x305 -m transport -E aes-cbc "2234567890123456" -A
hmac-sha1 "22345678901234567890";
add 10.1.0.2 10.1.0.1 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.

```
# gateway1 :
flush;
spdflush;
spdadd 10.1.0.1 10.1.0.2 any -P out ipsec ah/transport//require;
spdadd 10.1.0.2 10.1.0.1 any -P in ipsec ah/transport//require;
add 10.1.0.1 10.1.0.2 ah 0x307 -m transport -A hmac-sha1 "22345678901234567890";
add 10.1.0.2 10.1.0.1 ah 0x308 -m transport -A hmac-sha1 "12345678901234567890";

# gateway2 :
flush;
spdflush;
spdadd 10.1.0.1 10.1.0.2 any -P in ipsec ah/transport//require;
spdadd 10.1.0.2 10.1.0.1 any -P out ipsec ah/transport//require;
add 10.1.0.1 10.1.0.2 ah 0x307 -m transport -A hmac-sha1 "22345678901234567890";
add 10.1.0.2 10.1.0.1 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 passerelles, éditez son fichier de configuration (`/etc/racoon/racoon.conf`) sur `gateway1` comme indiqué ci-dessous :

```
log notify;
path pre_shared_key "/etc/racoon/psk.txt";
remote 10.1.0.2 {
    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 ?

```
# gateway1 :
10.1.0.2 ipseckey
```

Recopier et adapter la configuration de *Racoon* sur `gateway2` puis démarrer le démon de chaque coté.

```
log notify;
path pre_shared_key "/etc/racoon/psk.txt";
remote 10.1.0.1 {
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;
}

# gateway2 :
10.1.0.1 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.

```
## gateway1 :
spdadd 10.1.0.1 10.1.0.2 any -P out ipsec esp/transport//require;
spdadd 10.1.0.2 10.1.0.1 any -P in ipsec esp/transport//require;

## gateway2 :
spdadd 10.1.0.2 10.1.0.1 any -P out ipsec esp/transport//require;
spdadd 10.1.0.1 10.1.0.2 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 `gateway2` pour créer :

- Une authorité de certification
- Un certificat serveur pour gateway1.uvsq.org
- Un certificat serveur pour gateway2.uvsq.org
- La `crl` associée

```
source ./vars
./clean-all
./build-ca
./build-key-server gateway1.uvsq.org
./build-key-server gateway2.uvsq.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 10.1.0.2 {
    exchange_mode main;
    verify_cert on;
    my_identifier asn1dn;
    certificate_type x509 "gateway1.uvsq.org.crt" "gateway1.uvsq.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 gateway1.uvsq.org.crt gateway1.uvsq.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 à `client2` depuis `client1`. Quelles sont les règles à ajouter à la SPD ?

```
# gateway1 :
spdadd 10.1.1.0/24 10.1.2.0/24 any -P out ipsec
esp/tunnel/10.1.0.1-10.1.0.2/require ;
spdadd 10.1.2.0/24 10.1.1.0/24 any -P in ipsec
esp/tunnel/10.1.0.2-10.1.0.1/require ;


# gateway2 :
spdadd 10.1.2.0/24 10.1.1.0/24 any -P out ipsec
esp/tunnel/10.1.0.2-10.1.0.1/require ;
spdadd 10.1.1.0/24 10.1.2.0/24 any -P in ipsec
esp/tunnel/10.1.0.1-10.1.0.2/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 `gateway2` depuis `gateway1`.

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)
```