ROYAUME DU MAROC

مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle et de la Promotion du Travail

VPN IPSec sous Gnu/Linux

www.ofppt.info

OFPPT

DIRECTION **R**ECHERCHE ET **I**NGENIERIE DE **F**ORMATION

SECTEUR NTIC

Sommaire

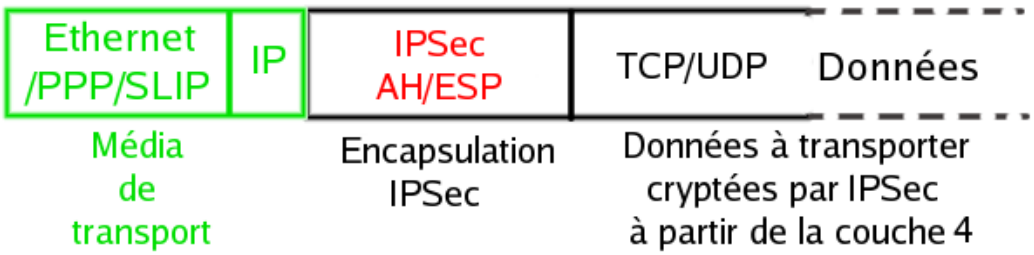
1. Fonctionnement

1.1. Mode de transport

IPSec est un protocole défini par l'IETF permettant de sécuriser les échanges au niveau de la couche réseau. Il s'agit en fait d'un protocole apportant des améliorations au niveau de la sécurité au protocole IP afin de garantir la confidentialité, l'intégrité et l'authentification des échanges.

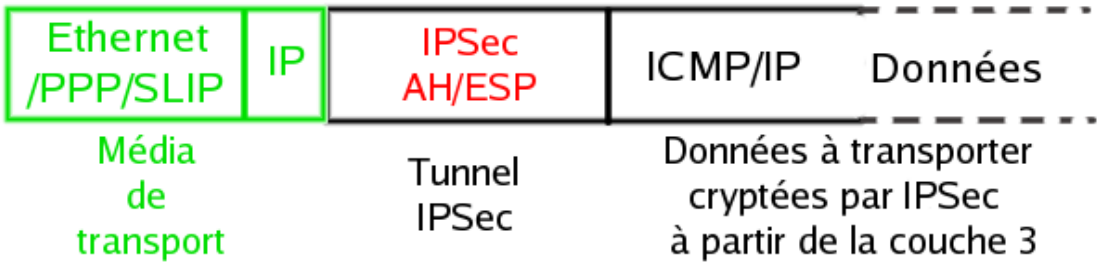
Il existe deux modes pour IPSec :

- le mode transport permet de protéger principalement les protocoles de niveaux supérieurs :
  - IPSec récupère les données venant de la couche 4 (TCP/transport), les signe et les crypte puis les envoie à la couche 3 (IP/réseau). Cela permet d'être transparent entre la couche TCP et la couche IP et du coup d'être relativement facile à mettre en place.
  - Il y a cependant plusieurs inconvénients :
    - l'entête IP est produite par la couche IP et donc IPSec ne peut pas la contrôler dans ce cas.
    - Il ne peut donc pas masquer les adresses pour faire croire à un réseau LAN virtuel entre les deux LAN reliés
    - cela ne garantie donc pas non plus de ne pas utiliser des options Ips non voulues



- le mode tunnel permet d'encapsuler des datagrammes IP dans des datagrammes IP
  - les paquets descendent dans la pile jusqu'à la couche IP et c'est la couche IP qui passe ses données à la couche IPSec. Il y a donc une entête IP encapsulée dans les données IPSec et une entête IP réelle pour le transport sur Internet (on pourrait imaginer que ce transport se fasse sur de l'IPX ou NetBIOS puisqu'il n'y a pas de contrainte dans ce mode)
  - Cela a beaucoup d'avantages :
    - l'entête IP réelle est produite par la couche IPSec. Cela permet d'encapsuler une entête IP avec des adresses relative au réseau virtuel et en plus de les crypter de façon à être sûr qu'elles ne sont pas modifiées.
    - On a donc des adresses IP virtuelles donc tirant partie au mieux du concept de VPN
    - On a le contrôle total sur l'entête IP produite par IPSec pour encapsuler

ses données et son entête IPSec.



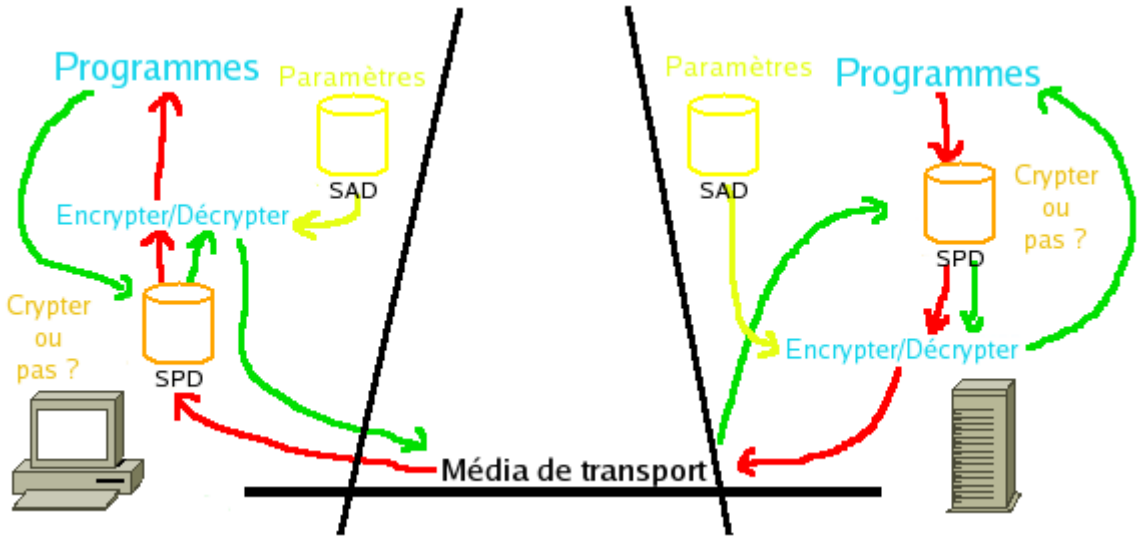
1.2. Les composantes d'IPSec

Le protocole IPSec est basé sur quatre modules :

- *IP Authentification Header* (**AH**) gère
  - l'intégrité : on s'assure que les champs invariants pendant la transmission, dans l'entête IP qui précède l'entête AH et les données
  - l'authentification pour s'assurer que l'émetteur est bien celui qu'il dit être
  - la protection contre le rejeu : un paquet intercepté par un pirate ne peut pas être renvoyé
  - il ne gère pas la confidentialité : les données sont signées mais pas cryptées

- *Encapsulating Security Payload* (**ESP**)
  - en mode transport, il assure
    - confidentialité : les données du datagramme IP encapsulé sont cryptées
    - authentification : on s'assure que les paquets viennent bien de l'hôte avec lequel on communique (qui doit connaître la clé associée à la communication ESP pour s'authentifier)
    - l'unicité optionnelle contre le rejeu des paquets
    - l'intégrité des données transmises
  - en mode tunnel, c'est l'ensemble du datagramme IP encapsulé dans ESP qui est crypté et subit les vérifications suivantes. On peut donc se passer de AH.

- *Security Assocation* (**SA**) définit l'échange des clés et des paramètres de sécurité. Il existe une SA par sens de communication. Les paramètres de sécurité sont les suivants :
  - protocole AH et/ou ESP
  - mode tunnel ou transport
  - les algo de sécurité utiliser pour encrypter, vérifier l'intégrité
  - les clés utilisées
- La SAD (*Security Association Database*) stocke les SA afin de savoir comment traiter les paquets arrivant ou partant. Elles sont identifiées par des triplets :
  - adresse de destination des paquets
  - identifiant du protocole AH ou ESP utilisé
  - un index des paramètres de sécurité (Security Parameter Index) qui est un champ de 32bits envoyer en clair dans les paquets
- La SPD (Security Policy Database) est la base de configuration de IPSec. Elle permet



de dire au noyau quels paquets il doit traiter. C'est à sa charge de savoir avec quel SA il fait le traitement.

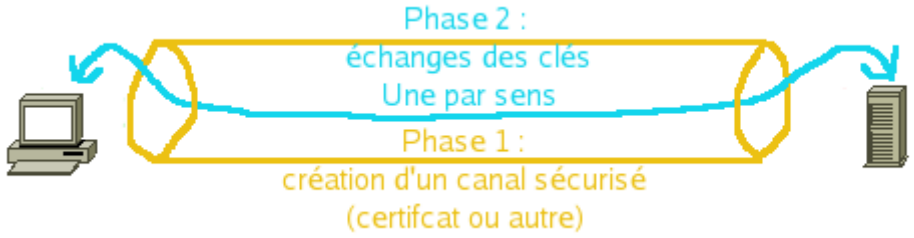
En résumé, le SPD indique quels paquets il faut traiter et le SAD indique comment il faut traiter un paquet sélectionné.

1.3. L'échange des clés

L'échanges des clés nécessaires au cryptage des données dans IPSec peut se faire de trois façons différentes :

- à la main : pas très pratique
- IKE (Internet Key Exchange) : c'est un protocole développé pour IPSec. ISAKMP (Internet Security Association and Key Management Protocol) en est la base et a pour rôle la création (négociation et mise en place), la modification et la suppression des SA. Elle se compose de deux phases :
    - la première permet de créer un canal sécurisé (par Diffie-Hellman) et authentifié à travers duquel on échange un secret pour dériver les clés utilisées dans la phase 2.
    - la seconde permet de mettre en place IPSec avec ses paramètres et une SA par sens de communication. Les données échangées sont protégées par le canal mis en place dans la phase 1.

A l'issue de ces deux phases, le canal IPSec est mis en place.



1.4. Informations complémentaires : accès à distances par les utilisateurs

1.4.1. Authentification IPSec de la phase 1

La phase 1 d'IPSec fait parti du protocole IKE (IPSec Key Exchange) géré par le démon racoon. Son but est d'authentifier les machines en présence (client et serveur) afin de définir une clé maître pour sécurisé la phase 2 d'IPSec. Le but de la phase 2 est, quant à elle, de dériver les clés utilisées

pour les échanges de trafic par IPSec.

La phase 1 d'IPSec offre deux méthodes d'authentification :

- les clés prépartagés. Plus difficile à maintenir mais rapide à mettre en place. Moins sûr.
- les certificats.Plus facileà maintenir et moins rapide à mettre en place. Plus sûr.

1.4.2. Xauth

Xauth est une extension IKE qui se passe après la phase 1 et ajoute une authentification login/mot de passe sécurisé par la phase 1 (mot de passe pas en clair).

1.4.3. Hybrid auth

L'authentification hybride est une autre extension IKE qui rend la phase 1 asymétrique. Durant la phase 1, la passerelle VPN peut utiliser un certificat alors que l'utilisateur distant n'a pas besoin de s'authentifier. Après la phase 1, on est donc dans la situation suivante :

- l'utilisateur distant sait qu'il parle bien à la bonne passerelle VPN
- la communication entre l'utilisateur distant et la passerelle VPN est sécurisée
- la passerelle VPN ne sait pas réellement si elle parle au bon client ou à un hacker

Après la phase 1, un échange Xauth peut intervenir pour authentifier de façon sûr l'utilisateur distant. Ensuite la phase 2 peut commencer.

Le niveau de sécurité de IPSec + Xauth + Hybrid est à peu près équivalent à une authentification par mot de passe en SSH.

1.4.4. Mode de configuration ISAKMP

Cela permet de donner automatiquement une configuration à un utilisateur authentifié de façon à ce que la configuration du VPN soit automatique. C'est encore une extension IKE qui autorise le serveur VPN à fournir une configuration réseau à la machine de l'utilisateur distant : adresse IP interne, adresse DNS, nom de domaine...

2. Implémentation Linux

Il existe plusieurs implémentation de IPSec dans Linux suivant la version du noyau :

- 2.4 et inférieur : FreeS/WAN (http://www.freeswan.org) et OpenSWAN (http://www.openswan.org)
- 2.6 et supérieur : support Natif (KAME-tools at http://ipsec-tools.sourceforge.net/), OpenSWAN ou isakmpd (de OpenBSD)

2.1. IPSec-tools (KAME-tools)

2.1.1. Installation

Pour utiliser IPSec en natif dans Linux, il faut avoir compiler son noyau avec les options (à y) (le critère de recherche dans config-*noyau* est indiqué entre parenthèses) :

- Network support
  - PF_KEY sockets (NET_KEY)
  - AH transformation (INET_AH)
  - ESP transformation (INET_ESP)
  - IPSec user configuration interface (XFRM_USER)
- Cryptographic API, en vrac : HMAC, Null, MD5, SHA1, DES et Triple DES, AES (CRYPTO_*)

Pour vérifier qu'une option *critère* est incluse dans le noyau ou dans un module de celui-ci :

```
cat /boot/config-`uname-r` |grep critère
```

Doit renvoyer au moins une ligne telle que : `CONFIG_critère=y` ou`CONFIG_critère=m`

Il faut aussi installer les IPSec-tools (http://ipsec-tools.sourceforge.net/) :

```
./configure --with-kernel-headers=/lib/modules/2.6.X/build/include
make
```

```
make install
```

où *X* est le troisième chiffre de la version de votre noyau.

2.1.2. Configuration de base

La configuration est assez compliquée mais peu se résumer de la façon suivante :

- on crée un script pour définir les paramètres IPSec à mettre dans la SAD (Security Association Database) et dans la SPD (Security Policy Database) :
  - SAD
    - les IP source et destination
    - le mode tunnel ou transport
    - les algos de hachage, cryptage et autres
    - des clés pour AH et ESP
  - SPD
    - les IP source et destination
    - le sens du traffic
    - si AH et/ou ESP sont requis
- on exécute le script avec la commande `setkey -f`.

i.Association de sécurité pour l'authentification

Une association de sécurité d'authentification a la forme suivante :

```
add IP_pub_machine1IP_pub_machine2 ah SPI1 -A algo_hash "clé_authentifaction";
```

Ceci indique que le traffic de *IP_pub_machine1* à *IP_pub_machine2* doit pouvoir être authentifier avec une entête AH signée avec l'algo de hachage *algo_hash* et la clé d'authentification *clé_authentification*. Cette association est repérée par l'index *SPI1* (un nombre). Les associations sont symétriques, c'est à dire qu'il faut la mettre identique sur les deux machines communicantes. De plus, elle n'agit que sur une sens de communication donc si l'on veut authentifier ou crypter dans les deux sens il faut une association par sens.

IP_pub_machine1 et *IP_pub_machine2* sont les IP de ces machines. La clé d'authentification est un nombre (décimal ou hexa) dont la taille en bits dépend de l'algo de hachage. *algo_hash* peut être choisi parmi : hmac-md5, hmac-sha1, keyed-md5, keyed-sha1, hmac-sha256, hmac-sha384, hmac-sha512, hmac-ripemd160, aes-xcbc-mac ou tcp-md5. Suivant la valeur de ce dernier, la taille de la clé peut être différente (md5 : 128bits, sha1 : 160bits...)

On peut aussi ajouter `-m tunnel` pour passer l'association en mode tunnel.

ii.Association de sécurité pour l'encryptage

Une association de sécurité d'encryptage a la forme suivante :

```
        add IP_pub_machine1IP_pub_machine2 esp SPI2 -E algo_crypt
"clé_encryptage";
```

Ceci indique que le traffic de *IP_pub_machine1* à *IP_pub_machine2* doit pouvoir être crypté (avec donc une entête ESP) avec l'algo de cryptage *algo_crypt* et la clé de cryptage *clé_encryptage*. Cette association est repérée par l'index *SPI2* (un nombre). Les associations sont symétriques, c'est à dire qu'il faut la mettre identique sur les deux machines communicantes. De plus, elle n'agit que sur une sens de communication donc si l'on veut authentifier ou crypter dans les deux sens il faut une association par sens.

IP_pub_machine1 et *IP_pub_machine2* sont les IP de ces machines. La clé d'encryptage est un nombre (décimal ou hexa) dont la longueur en bits dépend de l'algo d'encryptage. *algo_encryptage* peut être choisi parmi : des-cbc, 3des-cbc, blowfish-cbc, cast128-cbc,... Suivant la valeur de ce dernier, la taille de la clé peut être différente (des : 64bits, 3des : 192bits...)

On peut aussi ajouter `-m tunnel` pour passer l'association en mode tunnel.

iii.Politique de sécurité

(1)Paquets entrants

Une politique de sécurité a la forme suivante pour les paquets entrants (appliquée sur la machine 1) :

```
spdadd IP_pub_machine2 IP_pub_machine1 any -P IN ipsec

esp/mode/src-dst/require

ah/mode/src-dst/require;
```

Cela indique que le traffic venant de *IP_pub_machine2* et à destination de notre interface *IP_pub_machine1* doit être correctement authentifié et crypté et les paquets doivent être dans la mode *mode* (transport ou tunnel). Les paquets en clair (donc pas IPSec) venant de *IP_pub_machine2* seront détruits.

A noter que sur la machine 2, on doit inverser *IP_pub_machine1* et *IP_pub_machine2* et mettre OUT à la place de IN.

IP_pub_machine1 et *IP_pub_machine2* sont les IP de ces machines ou de leurs sous réseaux d'appartenance suivant la portée que l'on veut donner au VPN. La partie *src-dst* peut être omise si le mode est transport. Si le mode est tunnel, il faut préciser entre quelles machines *src-dst* (IP) se fait le tunnel.

(2)Paquets sortants

Une politique de sécurité a la forme suivante pour les paquets sortants (appliquée sur la machine 1) :

```
spdadd IP_pub_machine1 IP_pub_machine2 any -P OUT ipsec

esp/mode/src-dst/require

ah/mode/src-dst/require;
```

Cela indique que le traffic sortant de notre interface *IP_pub_machine1* et à destination de *IP_pub_machine2* doit être authentifié et crypté et les paquets doivent être émis dans la mode *mode* (transport ou tunnel).

A noter que sur la machine 2, il est préférable de mettre une règles dans laquelle *IP_pub_machine1* et *IP_pub_machine2* sont inversés et où IN est mis à la place de OUT.

IP_pub_machine1 et *IP_pub_machine2* sont les IP de ces machines ou de leurs sous réseaux d'appartenance suivant la portée que l'on veut donner au VPN . La partie *src-dst* peut être omise si le mode est transport. Si le mode est tunnel, il faut préciser entre quelles machines *src-dst* (IP) se fait le tunnel.

iv.Un exemple complètement statique en mode Tunnel (VPN)

Note : vous pouvez remplacer les clés par des clés générées par vous même avec /dev/urandom par exemple.

Voici un exemple de fichier de configuration statique pour le mode tunnel entre une machine 1 et une machine 2 qui permettent de relier les réseaux 1 et 2 (les différences sont en **rouge**). On remarquera que l'on utilise que ESP car en mode tunnel, ce protocole assure à la fois l'encryptage et l'authentification. Attention, seuls les paquets entre une machine du réseau 1 et du réseau 2 seront encryptés. Par contre, si on envoit un ping de *IP_pub_machine1* à *IP_pub_machine2*, il sera en clair car aucunes des IP *IP_pub_machine1* et *IP_pub_machine2* ne font parties des IP des réseau 1 et 2.

- Sur la machine 1 :

```
#!/usr/sbin/setkey -f

# Effacer les associations et les polices existantes
flush;
spdflush;

# ESP SAs doing encryption using 192 bit long keys (168 + 24 parity)
# and authentication using 128 bit long keys
add IP_pub_machine1IP_pub_machine2 esp 0x201 -m tunnel -E 3des-cbc
0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831
-A hmac-md5 0xc0291ff014dccdd03874d9e8e4cdf3e6;

add IP_pub_machine2IP_pub_machine1 esp 0x301 -m tunnel -E 3des-cbc
0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df
-A hmac-md5 0x96358c90783bbfa3d7b196ceabe0536b;

# Security policies
spdadd adresse_reseau1 adresse_reseau2 any -P out ipsec
           esp/tunnel/IP_pub_machine1-IP_pub_machine2/require;

spdadd adresse_reseau2 adresse_reseau1 any -P in ipsec
           esp/tunnel/IP_pub_machine2-IP_pub_machine1/require;

#il peut être nécessaire pour kernel >= 2.6.10 et ipsec-tools < 0.5

# d'autoriser les paquets à traverser IP_pub_machine1 et IP_pub_machine2
spdadd adresse_reseau2 adresse_reseau1 any -P fwd ipsec
esp/tunnel/IP_pub_machine2-IP_pub_machine1/require;
```

- Sur la machine 2 (même chose mais dans le sens **inverse**) :

```
#!/usr/sbin/setkey -f

# Effacer les associations et les polices existantes
flush;
spdflush;

# ESP SAs doing encryption using 192 bit long keys (168 + 24 parity)
# and authentication using 128 bit long keys
add IP_pub_machine1IP_pub_machine2 esp 0x201 -m tunnel -E 3des-cbc
0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831
-A hmac-md5 0xc0291ff014dccdd03874d9e8e4cdf3e6;

add IP_pub_machine2IP_pub_machine1 esp 0x301 -m tunnel -E 3des-cbc
0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df
-A hmac-md5 0x96358c90783bbfa3d7b196ceabe0536b;

# Security policies
spdadd adresse_reseau1 adresse_reseau2 any -P in ipsec
           esp/tunnel/IP_pub_machine1-IP_pub_machine2/require;

spdadd adresse_reseau2 adresse_reseau1 any -P out ipsec
           esp/tunnel/IP_pub_machine2-IP_pub_machine1/require;

#il peut être nécessaire pour kernel >= 2.6.10 et ipsec-tools < 0.5
```

```
# d'autoriser les paquets à traverser IP_pub_machine1 et IP_pub_machine2
spdadd adresse_reseau1 adresse_reseau2 any -P fwd ipsec
esp/tunnel/IP_pub_machine1-IP_pub_machine2/require;
```

On peut vérifier que les paquets partent et reviennent bien encryptés sur une *IP_interne1* avec `tcpdump host` *IP_pub_machine2* sur la machine 2:

- le ping : on voit bien que l'on envoie pas le paquet echo-request en clair. On remarquera que l'adresse de destination n'est pas celle de la machine *IP_interne1*. En effet, cette adresse se trouve dans le paquet IP crypté dans les données ESP. Il y a bien un tunnel crypté entre *IP_pub_machine2* et *IP_pub_machine2*

```
11:42:57.676757 IPIP_pub_machine2 > IP_pub_machine1:
ESP(spi=0x00000201,seq=0x52), length 116
```

- le pong : on voit un paquet crypté arriver et on voit bien un paquet ICMP echo-reply remonter. On rappelle que le mode tunnel encapsule l'entête IP réelle dans les données de ESP, ESP étant lui-même encapsulé dans un paquet IP entre les deux bout du tunnel ce qui fait que l'on croit recevoir deux paquets alors ce n'en est qu'un : on reçoit les données ESP cryptées qui sont décryptées et repassées à la couche IP :

```
11:42:57.677604 IPIP_pub_machine1 > IP_pub_machine2:
ESP(spi=0x00000301,seq=0x6f), length 116

11:42:57.677604 IPIP_interne1 > IP_pub_machine2: ICMP echo reply, id 6417,
seq 5, length 64
```

Cela peut se schématiser ainsi :

v.Un exemple complètement statique en mode Transport

Note : vous pouvez remplacer les clés par des clés générées par vous même avec /dev/urandom par exemple.

Voici un exemple de fichier de configuration statique pour le mode transport entre une machine 1 et une machine 2. **Attention, le mode transport assure l'encryptage ESP et l'authentification AH entre les machines 1 et 2 mais absolument pas entre les réseaux 1 et 2.**

- Sur la machine 1 :

```
#!/sbin/setkey -f
```

```
# Configuration for machine1

# Flush the SAD and SPD

flush;

spdflush;

# Attention: Use this keys
only for testing purposes!

# Generate your own keys!

# AH SAs using 128 bit long
keys

add machine1machine2 ah 0x200
-A hmac-md5

        0xc0291ff014dccdd03874d9e8e4cdf3e6;

add machine2machine1 ah 0x300
-A hmac-md5
```

```
        0x96358c90783bbfa3d7b196ceabe0536b;


                                        # ESP SAs using 192 bit long
keys (168 + 24 parity)

                                        add machine1machine2 esp
0x201 -E 3des-cbc

        0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;

                                        add machine2machine1 esp
0x301 -E 3des-cbc

        0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;


                                        # Security policies

                                        spdadd machine1machine2 any -
P out ipsec

                                        esp/transport//require

                                        ah/transport//require;
```

```
spdadd machine2machine1 any -
P in ipsec

esp/transport//require

ah/transport//require;
```

- Sur la machine 2 (même chose mais dans le sens **inverse**) :

```
#!/usr/sbin/setkey -f

# Configuration for machine2

# Flush the SAD and SPD

flush;

spdflush;

# Attention: Use this keys
only for testing purposes!

# Generate your own keys!
```

```
                                        # AH SAs using 128 bit long
keys

                                        add machine2machine1 ah 0x200
-A hmac-md5

        0xc0291ff014dccdd03874d9e8e4cdf3e6;

                                        add machine1machine2 ah 0x300
-A hmac-md5

        0x96358c90783bbfa3d7b196ceabe0536b;


                                        # ESP SAs using 192 bit long
keys (168 + 24 parity)

                                        add machine2machine1 esp
0x201 -E 3des-cbc

        0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;

                                        add machine1machine2 esp
0x301 -E 3des-cbc
```

```
        0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;


                                        # Security policies

                                        spdadd machine2machine1 any -
P in ipsec
                                        esp/transport//require

                                        ah/transport//require;


                                        spdadd machine1machine2 any -
P out ipsec
                                        esp/transport//require

                                        ah/transport//require;
```

On peut vérifier que les paquets partent et reviennent bien encryptés sur *machine1* avec `tcpdump host machine2`:

- le ping : on voit bien que l'on envoie pas le paquet echo-request en clair

```
11:07:19.411163 IPmachine1 > machine2: AH(spi=0x00000200,seq=0x8):
ESP(spi=0x00000201,seq=0x8), length 88
```

- le pong : on voit un paquet crypté arriver et on voit bien que ce n'est pas le paquet ICMP echo-reply en clair :

```
11:07:19.411830 IPmachine2 > machine1: AH(spi=0x00000300,seq=0x8):
ESP(spi=0x00000301,seq=0x8), length 88
```

vi.D'autres exemples de configuration IPSec

Pour plus d'informations sur les configurations avec IPSec-tools, voir http://www.ipsec-howto.org/x299.html.

2.1.3. Configuration avec clés automatique par IKE et certificats pour utilisateur itinérant (serveur et client Linux)

Si le protocole IKE et son serveur racoon génèrent les clés à la volée et les SA pour la SAD, il ne gère pas les règles de police de sécurité à mettre dans la SPD. Et cela est évident, comment pourrait-t-il savoir ce que veut l'admin et entre quoi et quoi se fait l'encryptage.

i.Installation de ipsec-tools

Si vous souhaitez utilisez l'authentification hybride entre deux machines Linux (comme le montre la configuration suivante), il vous sera nécessaire de disposer d'un noyau 2.6 et de compiler les « ipsec-tools » dans une version supérieure à 0.5.2 (version packagée dans Debian Sarge, par exemple, mais qui ne contient pas ce type d'authentification hybride). Pour cela, il vous sera nécessaire d'installer les paquets suivants :

```
[root]# apt-get install libssl0.9.6 libssl-dev flex bison
libreadline5 libreadline5-dev libc6-dev bzip2 libpam0g-dev
```

On pourra télécharger le .tar.bz2 à l'adresse http://ipsec-tools.sourceforge.net/

(1)Compilation sur le client

```
[root]# tar xjvf ipsec-tools-*.tar.bz2

[root]# cd ipsec-tools-*

[root]# ./configure --enable-natt --enable-frag --enable-
hybrid \ --enable-dpd --enable-adminport \

--sysconfdir=/etc/racoon -localstatedir=/var \

--prefix=/usr

[root]# make

[root]# make install
```

(2)Compilation sur le serveur

```
[root]# tar xjvf ipsec-tools-*.tar.bz2

[root]# cd ipsec-tools-*

[root]# ./configure --enable-natt --enable-frag --enable-
hybrid \

--enable-dpd --sysconfdir=/etc/racoon --with-libpam \

--prefix=/usr

[root]# make

[root]# make install
```

ii.Générer son autorité de certification

Il peut être nécessaire de générer une autorité de certification maison pour signer ses propres certificats. Un certificat CA n'est rien de plus qu'un certificat autosigné :

```
[root]# cd /dossier/de/stockage


[root]# mkdir -p demoCA/newcerts
[root]# touch demoCA/index.txt
[root]# echo "00" > demoCA/serial
[root]# mkdir certs
[root]# openssl genrsa -des3 -out certs/ca.key 2048
```

Cette commande vous demande une passphrase pour protéger la clé privée crée.

```
[root]# chmod 600 certs/*.key
[root]# openssl req -days 3650 -x509 -key certs/ca.key -new -out
certs/ca.crt
```

Cette dernière commande vous demandera des informations sur le certificat de l'autorité CA crée et la passphrase pour décrypter la clé privée de l'autorité.

Ensuite, on va conserver ces deux clés afin de signer tous les certificats que l'on va générer par la suite.

iii.Génération des certificats du serveur et des clients

Note : le Common Name est celui du fichier, par exemple CN = client_ipsec, certificat client_ipsec.crt. De plus, Organization Name doit être le même pour le certificat CA et pour les certificats des clients et du serveur.

Il faut d'abord modifier la ligne contenant `dir` dans le fichier `/etc/openssl/openssl.cnf` (ou `/etc/pki/tls/openssl.cnf`) pour la faire pointer vers `/dossier/de/stockage/demoCA`.

Il faut générer un certificat pour toutes les machines qui vont communiquer (par exemple avec une clé de 1024 bits) :

```
[root]# /usr/bin/openssl genrsa -out certs/fichier.key
nb_bits_clé

[root]# chmod 600 certs/*.key
```

Il ne faut pas donner de passphrase car racoon ne sait pas les décrypter.

```
[root]# /usr/bin/openssl req -new -key certs/fichier.key -out
certs/fichier.csr

[root]# openssl ca -cert /dossier/de/stockage/ca.crt -keyfile
/dossier/de/stockage/ca.key -in certs/fichier.csr -out
certs/fichier.crt -days nb_jour_validité
```

Le certificat généré se trouve alors dans le fichier *fichier*.crt et la clé privée dans *fichier*.key.

iv.Mise en place des certificats et autres clés

Chaque machine doit avoir sa propre clé privée, son certificat et la certificat de la CA dans son répertoire /etc/certs/ (root:root 660) . Il doit aussi avoir le certificat des hôtes distants auxquels il se connecte.

Pour le serveur IPSec :

- le certificat de la CA : ca.crt
- sa clé privée et son certificat : serv_ipsec.key (root:root 600) et serv_ipsec.crt
- le certificat de tous ces clients : client_ipsec.crt

Pour les clients :

- le certificat de la CA : ca.crt
- sa clé privée et son certificat : client_ipsec.key (root:root 600) et client_ipsec.crt
- le certificat du serveur IPSec : serv_ipsec.crt

v.Configuration de racoon

Le démon racoon sert à gérer le protocole IPSec pour le mettre en place automatiquement par IKE. Il peut se lancer par `racoon -f /etc/racoon/racoon.conf` ou `/etc/init.d/racoon start`.

(1)Serveur

Il faudra copier le script `phase1-down.sh` dans le dossier `/etc/racoon` :

```
[root]# cp
/usr/share/doc/racoon/examples/samples/roadwarrior/server/phase1-down.sh
/etc/racoon/

[root]# chmod 755 /etc/racoon/*.sh
```

Dans un fichier `/etc/racoon/motd`, on pourra mettre un message de bienvenue aux utilisateurs se connectant.

Pour le serveur le fichier `/etc/racoon/racoon.conf` contient :

```
#config donnée depuis
# /usr/share/doc/racoon/examples/samples/roadwarrior/server/racoon.conf

#chemin où sont stockés les certificats et clés privées
path certificate "/etc/certs";

#indique les clients distants autorisés
remote anonymous {
          #mode nécessaire à un utilisateur itinérant
         exchange_mode aggressive;
          #certificat du serveur et clé privée
         certificate_type x509 "serv_ipsec.crt" "serv_ipsec.key";
          #certificat de l'autorité de certification
         ca_type x509 "ca.crt";
          #indique d'utiliser le CN des certificats pour s'identifier
         my_identifier asn1dn;
          #obéir à la proposition de jeu de cryptage des clients
         proposal_check obey;
          #générer les règles IPSec à la connexion des clients
         generate_policy on;
          #permet de traverser une passerelle NAT entre le serveur et le
client
         nat_traversal on;
          #indique le délai de régénération des clés
         dpd_delay 20;
          #indique que l'on peut fragmenter les paquets IKE et autres
         ike_frag on;
          #indique un script pour nettoyer la connexion à la déconnexion
         script "/etc/racoon/phase1-down.sh" phase1_down;
          #proposition de jeu de cryptage aux clients pour la phase 1
         proposal {
                 encryption_algorithm 3des;
                 hash_algorithm sha1;
```

```
                authentication_method hybrid_rsa_server;
                dh_group 2;
        }
}

#indique la configuration réseau à donner aux clients
mode_cfg {
          #première adresse à attribuer au client
        network4 première_IP_réseau_VPN;
          #nombre d'IP donc de clients possibles
        pool_size nb_clients;
          #masque du réseau des clients VPN
        netmask4 masque_réseau_VPN;
          #vérifier les login/mots de passes à partir de PAM sur le serveur
        auth_source pam;
          #adresse IP du DNS pour les clients
        dns4 IP_serveur_DNS;
          #adresse IP du WINS pour les clients
        wins4 IP_serveur_WINS;
          #chemin du fichier contenant le message de bienvenu pour les
clients
        banner "/etc/racoon/motd";
}

#jeu de cryptage pour la phase 2 d'IPSec (dont trafic)
sainfo anonymous {
        pfs_group 2;
        lifetime time 12 hour;
        encryption_algorithm 3des;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}
```

(2)Client

Il faudra copier le script `phase1-down.sh` et `phase1-up.sh` dans le dossier `/etc/racoon` :

```
[root]# cp /usr/share/doc/racoon/examples/samples/roadwarrior/server/*.sh
/etc/racoon/

[root]# chmod 755 /etc/racoon/*.sh
```

Pour le serveur le fichier `/etc/racoon/racoon.conf` contient :

```
#config donnée depuis
# /usr/share/doc/racoon/examples/samples/roadwarrior/client/racoon.conf

#chemin du fichier contenant les certificats et clés privées
path certificate "/etc/certs";

#indique de maintenir les correspondances NAT sur la passerelle
# en envoyant des paquets « vides » toutes les 5 secondes
timer {
        natt_keepalive 5sec;
}

#permet le contrôle de racoon par la commande racoonctl (en root)
```

```
listen {
        adminsock "/var/racoon/racoon.sock" "root" "operator" 0660;
}

#configuration pour la connexion au VPN
remote IP_serveur_VPN {
          #seul mode pour utilisateur itinérant
        exchange_mode aggressive;
          #certificat de l'autorité de certification
        ca_type x509 "ca.crt";
          #obéir au jeu de cryptage du serveur
        proposal_check obey;
          #autorise à traverser une passerelle NAT entre le client et le
serveur
        nat_traversal on;
          #autorise la fragmentation des paquets
        ike_frag on;
          #autorise le serveur à envoyer la configuration pour nous
        mode_cfg on;

          #ne pas vérifier les identifiants
        verify_identifier off;
          #vérifier les certificats
        verify_cert on;

          #certificat et clé privée du client
        certificate_type x509 "client_ipsec.crt" "client_ipsec.key";
          #utiliser le CN des certificats pour le serveur
        my_identifier asn1dn;
          #utiliser le CN des certificats pour les clients
        peers_identifier asn1dn;

          #script pour activer la configuration envoyée par le serveur
        script "/etc/racoon/phase1-up.sh" phase1_up;
          #script pour désactiver la configuration à la déconnexion
        script "/etc/racoon/phase1-down.sh" phase1_down;
          #initier la connexion au serveur VPN
        passive off;
          #jeu de cryptage pour la phase 1
        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method hybrid_rsa_client;
                dh_group 2;
        }
}

#jeu de cryptage pour la phase 2 (dont trafic)
sainfo anonymous {
        pfs_group 2;
        lifetime time 12 hour ;
        encryption_algorithm 3des;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate ;
}
```

(3)Mémorisation du mot de passe sur le client

Si vous voulez ne pas avoir à taper votre mot de passe Xauth à chaque connexion, vous pouvez le mettre dans le fichier des clés prépartagées.

Pour cela, dans le fichier `/etc/racoon/racoon.conf`, il faut ajouter :

```
#chemin et nom du fichier contenant des clés prépartagées
path pre_shared_key "/etc/racoon/psk.txt";
```

et dans sa section `remote` :

```
xauth_login "nom_utilisateur";
```

Puis dans le fichier `/etc/racoon/psk.txt` :

nom_utilisateurmot_de_passe

Ceci vous permettra de vous connecter simplement en faisant :

```
[root]# racoonctl vc

IP_serveur_VPN
```

(4)Notes

Note : les certificats qu'il soit au format PEM ou au format .key/.pub doivent absolument être décrypté (et donc lisible uniquement par l'utilisateur sous lequel tourne racoon) car racoon n'a pas la possibilité de vous demander le mot de passe.

Note pour Windows : les certificats pour Windows sont au format PKCS#12 :

```
[root]# openssl pkcs12 -export -inkey fichier_key.pem -
certfile /etc/certs/cacert.pem -in fichier_cert.pem -out
windows_client.p12
```

vi.Démarrage

Racoon va générer les SA et les règles de police pour vous.

Le démarrage se fait comme suit :

```
[root]# /etc/init.d/racoon start
```

2.1.4. Connexion et déconnexion au VPN

Il est nécessaire d'exécuter les commandes suivantes :

- pour se connecter

```
[root]# racoonctl vc -u utilisateur IP_ou_nom_VPN
```

- pour se déconnecter

```
[root]# racoonctl vd IP_ou_nom_VPN
```

Il vous sera alors nécessaire de donner votre mot de passe tel que définit dans /etc/passwd sur le serveur VPN.

La connexion (échange de clés) se fait ensuite à la seconde tentative d'émission de paquets au travers du VPN. Ne soyez donc pas surpris si la première tentative de ping échoue et que la seconde réussie.

2.2. Problème de NAT, de fragmentation et de delai de connexion

2.2.1. NAT

Si le client se trouve derrière une passerelle assurant le NAT tel que c'est souvent le cas lorsque l'on a un réseau local et une passerelle pour accéder au net, vous ne pourrez pas utiliser IPSec directement. En effet, les paquets sont encapsulé dans des paquets ESP (au dessus de IP) qui par définition ne possède pas de port est sont donc très difficile à NATer. Dans ce cas, on utilisera NATT afin d'encapsuler les paquets ESP dans des paquets UDP pour passer les passerelles.

Pour ce faire,dans le fichier `/etc/racoon/racoon.conf`, dans la section `remote`, on mettre `nat_traversal on;`

De plus, il peut arriver que l'échange IKE ne se fasse pas si le port UDP du NAT change du fait de la faible persistance des associations de NAT dans les passerelles. Il peut donc être utile de faire des envoie de paquets « vide » pour faire persister l'association NAT sur la passerelle traversée.

Pour ce faire, on ajoutera ceci dans le fichier `/etc/racoon/racoon.conf` :

```
timer {

natt_keepalive 5sec; #10 seconde maxi

}
```

2.2.2. Fragmentation

La fragmention intervient lorsque l'on utilise, par exemple, une liaision DSL qui suppose par défaut que les paquets UDP ne sont pas de grande taille. La fragmentation va donc arriver plus fréquemment dans un IPSec NATé.

Il y a deux types de fragmentation possible pour IPSec :

- la fragmentation des paquets IKE lors de l'établissement des clés IPSec. On peut choisir la fragmentation IKE qui est une extension de ce protocole
- la fragmentation des paquets ESP lors des communications. Dans ce cas, il faut simplement fragmenter les paquets IP encapsuler dans ESP pour faire des paquets UDP/ESP de plus petite taille.

Dans le fichier `/etc/racoon/racoon.conf`, dans la section `remote`, on mettra `ike_frag on;`

2.2.3. Delai de connexion

Il peut arriver que la connexion entre le client et le serveur ne soit pas très fiable et donc que des déconnexions arrivent. Pour cela, on peut forcer le serveur IKE à renouveller les clés régulièrement.

Pour cela, dans le fichier `/etc/racoon/racoon.conf`, on ajoutera dans la section `remote`, `dpd_delay 20;`

2.3. Et iptables dans tout ça...

NOTE : la configuration suivante est valable uniquement pour le mode Tunnel. Le mode transport à un intérêt limité dans le VPN.

2.3.1. La passerelle IPSec machine 1

Si l'on veut limiter le forward aux paquets cryptés, il est nécessaire de marquer les paquets esp et/ou ah entrants :

```
iptables -t mangle -A PREROUTING -p 50 -j MARK --set-mark 1

iptables -t mangle -A PREROUTING -p 51 -j MARK --set-mark 1
```

Déjà, il faut autoriser ESP ou AH ou les deux en fonction de la configuration d'IPSec :

```
#ESP

iptables -A OUTPUT -p 50 -s IP_pub_machine1 -d IP_pub_machine2
-j ACCEPT

iptables -A INPUT -p 50 -s IP_pub_machine2 -d IP_pub_machine1
-j ACCEPT

#AH

iptables -A OUTPUT -p 51 -s IP_pub_machine1 -d IP_pub_machine2
-j ACCEPT

iptables -A INPUT -p 51 -s IP_pub_machine2 -d IP_pub_machine1
-j ACCEPT
```

Ensuite, il faut autoriser IKE :

```
iptables -A INPUT -p udp --sport 500 --dport 500 -s
IP_pub_machine2-j ACCEPT

iptables -A OUTPUT -p udp --sport 500 --dport 500 -d
IP_pub_machine2 -j ACCEPT

#si on fait du NAT-T

iptables -A INPUT -p udp --sport 4500 --dport 4500 -s
IP_pub_machine2 -j ACCEPT

iptables -A OUTPUT -p udp --sport 4500 --dport 4500 -d
IP_pub_machine2 -j ACCEPT
```

Ensuite, il faut autoriser le traffic entre les deux réseaux internes (si les paquets sont cryptés) :

```
# autoriser le transite par la passerelle

iptables -A FORWARD -m mark --mark 1 -j ACCEPT



# effectuer la NAT des paquets traversant

iptables -t nat -A POSTROUTING -s adresse_réseau1 -o
interface_publique -j SNAT --to-source IP_publique_machine1
```

```
#active le forwarding

echo 1 > /proc/sys/net/ipv4/ip_forward
```

Enfin, il faut indiquer à la machine 1, la route vers le réseau interne 2. Note, si la machine 1 n'est pas la passerelle par défaut du réseau 1, il faut indiquer une route vers la machine 1 sur la passerelle par défaut :

```
route add adresse_réseau2 gateway IP_pub_machine2
```

2.3.2. La passerelle IPSec machine 2

La configuration est la même mais dans l'autre sens : on inverse adresse_réseau1 par adresse_réseau2, etc...

2.3.3. En plus sur les passerelles IPSec

Ensuite, on peut faire des règles de filtrages sur la table filter comme si on était une simple passerelle.

2.3.4. En plus sur le client

Le client est considéré avec la configuration précédente mais sans les règles de forward.