



Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

VPN con KAME IPsec y kernel 2.6 (1038 lectures)

Per **Jaume Sabater**, [Primetime](http://www.linuxsilo.net/) (<http://www.linuxsilo.net/>)

Creado el 07/01/2006 17:21 modificado el 07/01/2006 17:21

Este tutorial pretende mostrar cómo montar una VPN (del inglés, Virtual Private Network) entre dos máquinas usando la implementación [KAME IPsec](#)⁽¹⁾ disponible en la rama 2.6 del [kernel de Linux](#)⁽²⁾. Paso a paso se configurará una VPN en modo túnel entre dos puertas de enlace usando [IPsec](#)⁽³⁾, [Racoon](#)⁽⁴⁾ y una clave compartida (del inglés, Shared Secret o Shared Key). En este tutorial se ha escrito a partir de una VPN montada entre dos máquinas con [Debian GNU/Linux](#)⁽⁵⁾ Sarge y se ha usado [Netfilter/IPtables](#)⁽⁶⁾ para el filtrado.

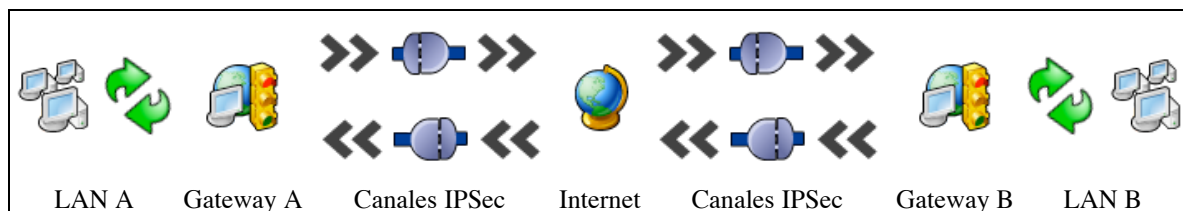
Índice

- [Introducción](#)
- [Instalación y configuración de una VPN](#)
- [Configuración de una VPN en modo túnel](#)
- [Sobre los algoritmos de cifrado y autenticación](#)
 - [¿Qué es la criptografía?](#)
 - [¿Qué es la criptografía de clave secreta?](#)
 - [¿Qué es la criptografía de clave pública?](#)
 - [¿Qué es Diffie-Hellman?](#)
 - [¿Qué son SHA y SHA-1?](#)
 - [¿Qué son MD2, MD4 y MD5?](#)
- [Bibliografía](#)
- [Historial de revisiones](#)

Introducción

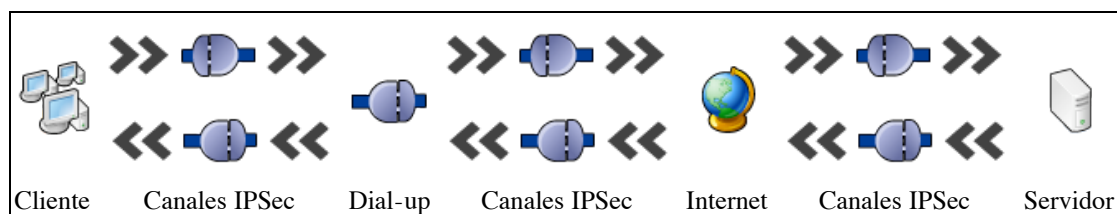
IPsec puede operar de dos maneras diferentes:

1. Modo túnel (del inglés, Tunnel mode). El propósito de este modo es establecer una comunicación segura entre dos redes remotas sobre un canal inseguro. Por ejemplo:



IPsec en modo túnel entre las dos puertas de enlace A y B

2. Modo transporte (del inglés, Transport mode). El propósito de este modo es establecer una comunicación segura punto a punto, entre dos hosts y sobre un canal inseguro. Por ejemplo:



IPsec en modo transporte entre un host cliente y uno servidor

Normalmente las VPNs se configuran en modo túnel.

IPsec funciona a partir de dos bases de datos:

- SPD (Security Policy Database, en inglés *Base de datos de políticas de seguridad*). Estas políticas le dicen a IPsec **cuando** debe o no debe actuar sobre un paquete IP.

2. SAD (Security Association Database, en inglés *Base de datos de asociaciones de seguridad*). Estas asociaciones le dicen a IPSec **cómo** debe crear el canal entre las dos máquinas.

Una SP o política de seguridad (del inglés, Security Policy) está formada, básicamente, por lo siguientes aspectos:

- Rango de direcciones de origen. Si la dirección de origen del paquete está dentro de este rango, entonces...
- Rango de direcciones de destino. Si la dirección de destino del paquete está dentro de este rango, entonces...
- Protocolo de alto nivel. Si el protocolo del paquete es tcp, udp, icmp, cualquiera, entonces...
- Política que se aplicará. Que consta de la dirección del canal a crear (entrada o salida) y la acción a tomar (*discard*, *none* o *ipsec*).

En el caso de aplicarse la política *ipsec* sobre el paquete, deberán especificarse los siguientes datos:

- Protocolo IPSec a usar: *ah*, *esp* o *ipcomp*.
- Modo IPSec que se usará (túnel o transporte). En el caso de *túnel* se deberán especificar las direcciones IPs de inicio y fin del mismo.
- Nivel IPSec:
 - *default*: nivel estándar.
 - *use*: usar una asociación de seguridad (SA) sólo si está disponible, o darle al paquete el tratamiento habitual en su defecto.
 - *require*: el uso de la asociación de seguridad es obligatorio para que se envíe el paquete.

Nota: si no hay una asociación de seguridad disponible, el kernel solicita una negociación de asociación de seguridad al demonio IKE (racoon).

Una asociación de seguridad (SA) está formada principalmente por los siguientes aspectos:

- Dirección IP de origen.
- Dirección IP de destino.
- Protocolo IPSec que debe usarse: *ah*, *esp* o *ipcomp*.
- Índice numérico (SPI, del inglés Security Parameter Index).
- Modo del protocolo: *tunnel*, *transport* o *any* (cualquiera de los dos).
- Algoritmos:
 - Autenticación y la clave
 - Cifrado y la clave
 - Cifrado y autenticación con las claves
 - Compresión

Nota: las asociaciones de seguridad (SA) sólo pueden usarse si están asociadas con una política, y esta asociación sólo puede establecerse si existe una política cuyas reglas tengan los mismos parámetros que la asociación de seguridad (SA). Aún así, puede haber una política de seguridad (SP) funcional sin una asociación de seguridad (SA) en uno de los dos casos siguientes:

1. Cuando la política de seguridad (SP) tiene el tipo *none* o *discard* y no haya ninguna asociación de seguridad (SA) que usar, pues IPSec no será aplicado.
2. Cuando la asociación de seguridad (SA) sea creada dinámicamente por el demonio IKE (racoon).

Instalación y configuración de una VPN

Antes de seguir deberemos asegurarnos de que tenemos un kernel 2.6 instalado y funcionando. Si es así, los siguientes paquetes deberían estar ya instalados:

- *cramfsprogs*: Tools for CramFs (Compressed ROM File System)
- *dash*: The Debian Almquist Shell
- *initrd-tools*: tools to create initrd image for prepackaged Linux kernel
- *module-init-tools*: tools for managing Linux kernel modules

A continuación se listan las opciones específicas del kernel que son necesarias para la VPN. Estas opciones activarán sus dependencias pero, de todos modos, se considera que el usuario ya tiene un kernel 2.6 con iptables funcionando.

Cryptographic options

- MD5 digest algorithm
- SHA1 digest algorithm
- DES and Triple DES EDE cipher algorithms
- Blowfish cipher algorithm
- AES cipher algorithms (i586)
- Deflate compression algorithm
- CRC32c CRC algorithm

Device Drivers: Networking support: Networking options

- PF_KEY sockets
- IP: AH transformation
- IP: ESP transformation

- IP: IPComp transformation
- IP: tunnel transformation
- IPsec user configuration interface

Device Drivers: Networking support: Networking options: Network packet filtering (replaces ipchains): IP: Netfilter Configuration

- IP tables support (required for filtering/masq/NAT)
- AH/ESP match support
- Packet filtering
- MASQUERADE target support

En el fichero `/usr/src/linux/.config` serían las siguientes opciones:

```

# Networking options
CONFIG_NET_KEY=y
CONFIG_INET_AH=y
CONFIG_INET_ESP=y
CONFIG_INET_IPCOMP=y
CONFIG_INET_TUNNEL=y

# IP: Netfilter Configuration
CONFIG_IP_NF_IPTABLES
CONFIG_XFRM_USER
CONFIG_IP_NF_MATCH_AH_ESP
CONFIG_IP_NF_FILTER
CONFIG_IP_NF_TARGET_MASQUERADE

# Cryptographic options
CONFIG_CRYPT=y
CONFIG_CRYPT_HMAC=y
CONFIG_CRYPT_MD5=y
CONFIG_CRYPT_SHA1=y
CONFIG_CRYPT_DES=y
CONFIG_CRYPT_BLOWFISH=y
CONFIG_CRYPT_AES_586=y
CONFIG_CRYPT_DEFLATE=y
CONFIG_CRYPT_CRC32C=y

```

Si, tras finalizar de leer el artículo, estamos seguros de que sólo vamos a usar un algoritmo de cifrado, por ejemplo Blowfish y no DES o AES, entonces obviamente el segundo no será necesario. De todos modos, el autor recomienda seleccionar todas las opciones para que cuando queramos cambiar de algoritmo no tengamos que recompilar el kernel y reiniciar la máquina.

Los paquetes necesarios para instalar y configurar una VPN con KAME IPsec sobre kernel 2.6 son los siguientes:

- *ipsec-tools*: IPsec tools for Linux
- *racoon*: IPsec IKE keying daemon
- *iproute*: Professional tools to control the networking in Linux kernels
- *iptables*: Linux kernel 2.4+ iptables administration tools

Nota: durante la instalación del paquete *racoon* elegiremos el método de configuración directo.

Asimismo, se recomienda la instalación de los siguientes paquetes:

- *dnsutils*: Clients provided with BIND
- *tcpdump*: A powerful tool for network monitoring and data acquisition
- *ntpdate*: The ntpdate client for setting system time from NTP servers

Por lo tanto, como *root*, ejecutaremos:

```
apt-get install ipsec-tools racoon iproute iptables dnsutils tcpdump ntpdate
```

Con el [kernel 2.4 de Linux](#)⁽⁷⁾ y [FreeS/WAN](#)⁽⁸⁾, la asociación del tráfico cifrado y las zonas se hacía fácil gracias a la presencia de pseudointerfaces con nombres del estilo *ipsecn* (p.e. *ipsecn0*). El tráfico cifrado saliente era enviado a través de un dispositivo *ipsecn* mientras que el tráfico cifrado entrante llegaba desde un dispositivo *ipsecn*. La implementación introducida en el [kernel 2.6](#)⁽⁹⁾ prescinde de estas pseudointerfaces. El tráfico saliente que va a ser cifrado y el tráfico entrante que debe ser descifrado deben compararse con las políticas en la SPD (del inglés, Security Policy Database) o de la SA (del inglés, Security Association) apropiada.

Configuración de una VPN en modo túnel

Supongamos que nos interesa que las máquinas de la subred local 192.168.0.0/24 sean capaces de comunicarse con los de la subred local 192.168.1.0/24. Ambas redes locales acceden a internet mediante sendas puertas de enlace 192.168.0.1 y 192.168.1.1, respectivamente. Las puertas de enlace tienen IPs públicas 213.96.80.51 y 80.36.214.182, respectivamente. En ambos gateways existen dos interfaces de red, una conectada a la LAN y otra a la WAN. La interfaz conectada a la red local es *eth0* y la interfaz conectada al router es *eth1* en ambos casos. Entonces, la topología de red es la siguiente:



Red A <-> Red B

Para conseguir esto necesitamos hacer dos cosas:

1. Abrir el cortafuegos de modo que se permita establecer un túnel IPsec (permitir los protocolos ESP y AH y el puerto UDP 500)
2. Permitir el tráfico a través del túnel.

Para permitir el tráfico por el túnel, empezaremos configurando las políticas de seguridad. Para gestionar dichas políticas, Debian cuenta con un demonio llamado *setkey*, manejable mediante el script */etc/init.d/setkey*, el cuál ejecuta las sentencias que se encuentren en el fichero */etc/ipsec-tools.conf* tras haber borrado las políticas que estén cargadas actualmente. Para una conexión de este tipo necesitaremos ocho entradas en el fichero */etc/ipsec-tools.conf* del gateway A (192.168.0.1):

```
spdadd 192.168.1.0/24 192.168.0.0/24 any -P in ipsec esp/tunnel/80.36.214.182-213.96.80.51/require;
spdadd 192.168.1.0/24 213.96.80.51/32 any -P in ipsec esp/tunnel/80.36.214.182-213.96.80.51/require;
spdadd 80.36.214.182/32 213.96.80.51/32 any -P in ipsec esp/tunnel/80.36.214.182-213.96.80.51/require;
spdadd 80.36.214.182/32 192.168.0.0/24 any -P in ipsec esp/tunnel/80.36.214.182-213.96.80.51/require;
spdadd 192.168.0.0/24 192.168.1.0/24 any -P out ipsec esp/tunnel/213.96.80.51-80.36.214.182/require;
spdadd 192.168.0.0/24 80.36.214.182/32 any -P out ipsec esp/tunnel/213.96.80.51-80.36.214.182/require;
spdadd 213.96.80.51/32 192.168.1.0/24 any -P out ipsec esp/tunnel/213.96.80.51-80.36.214.182/require;
spdadd 213.96.80.51/32 80.36.214.182/32 any -P out ipsec esp/tunnel/213.96.80.51-80.36.214.182/require;
```

El formato de la instrucción *spdadd* es el siguiente (de *man racoon.conf*):

```
spdadd [-46n] src_range dst_range upperspec policy
```

Las cuatro primeras entradas configuran la política para el túnel de entrada, mientras que las cuatro últimas las del túnel de salida. El comando *spdadd* añade políticas a la base de datos de políticas de seguridad. Respectivamente, las cuatro primeras sentencias establecen que:

- El tráfico entrante desde la subred 192.168.1.0/24 hacia la subred 192.168.0.0/24 será cifrado por *IPSec* usando el protocolo *ESP* en modo túnel, usando las direcciones de inicio y fin 80.36.214.182 y 213.96.80.51, respectivamente, y la asociación de seguridad será obligatoria o no se mandará el paquete.
- El tráfico entrante desde la subred 192.168.1.0/24 hacia el host 213.96.80.51 será cifrado por *IPSec* usando el protocolo *ESP* en modo túnel, usando las direcciones de inicio y fin 80.36.214.182 y 213.96.80.51, respectivamente, y la asociación de seguridad será obligatoria o no se mandará el paquete.
- El tráfico entrante desde el host 80.36.214.182 hacia el host 213.96.80.51 será cifrado por *IPSec* usando el protocolo *ESP* en modo túnel, usando las direcciones de inicio y fin 80.36.214.182 y 213.96.80.51, respectivamente, y la asociación de seguridad será obligatoria o no se mandará el paquete.
- El tráfico entrante desde el host 80.36.214.182 hacia la subred 192.168.0.0/24 será cifrado por *IPSec* usando el protocolo *ESP* en modo túnel, usando las direcciones de inicio y fin 80.36.214.182 y 213.96.80.51, respectivamente, y la asociación de seguridad será obligatoria o no se mandará el paquete.

Y las cuatro siguientes líneas establecen el canal de salida de manera análoga:

- El tráfico saliente desde la subred 192.168.0.0/24 hacia la subred 192.168.1.0/24 será cifrado por *IPSec* usando el protocolo *ESP* en modo túnel, usando las direcciones de inicio y fin 80.36.214.182 y 213.96.80.51, respectivamente, y la asociación de seguridad será obligatoria o no se mandará el paquete.
- El tráfico saliente desde la subred 192.168.0.0/24 hacia el host 80.36.214.182 será cifrado por *IPSec* usando el protocolo *ESP* en modo túnel, usando las direcciones de inicio y fin 80.36.214.182 y 213.96.80.51, respectivamente, y la asociación de seguridad será obligatoria o no se mandará el paquete.
- El tráfico saliente desde el host 213.96.80.51 hacia la subred 192.168.1.0/24 será cifrado por *IPSec* usando el protocolo *ESP* en modo túnel, usando las direcciones de inicio y fin 80.36.214.182 y 213.96.80.51, respectivamente, y la asociación de seguridad será obligatoria o no se mandará el paquete.
- El tráfico saliente desde el host 213.96.80.51 hacia el host 80.36.214.182 será cifrado por *IPSec* usando el protocolo *ESP* en modo túnel, usando las direcciones de inicio y fin 80.36.214.182 y 213.96.80.51, respectivamente, y la asociación de seguridad será obligatoria o no se mandará el paquete.

En el gateway B, 192.168.1.1, configuraremos el mismo fichero */etc/ipsec-tools.conf* pero intercambiando las entradas y las salidas (en las cuatro primeras entradas sustituiremos *in* por *out* y en las cuatro siguientes viceversa). El fichero resultante será el siguiente:

```
spdadd 192.168.1.0/24 192.168.0.0/24 any -P out ipsec esp/tunnel/80.36.214.182-213.96.80.51/require;
spdadd 192.168.1.0/24 213.96.80.51/32 any -P out ipsec esp/tunnel/80.36.214.182-213.96.80.51/require;
spdadd 80.36.214.182/32 213.96.80.51/32 any -P out ipsec esp/tunnel/80.36.214.182-213.96.80.51/require;
spdadd 80.36.214.182/32 192.168.0.0/24 any -P out ipsec esp/tunnel/80.36.214.182-213.96.80.51/require;
spdadd 192.168.0.0/24 192.168.1.0/24 any -P in ipsec esp/tunnel/213.96.80.51-80.36.214.182/require;
spdadd 192.168.0.0/24 80.36.214.182/32 any -P in ipsec esp/tunnel/213.96.80.51-80.36.214.182/require;
spdadd 213.96.80.51/32 192.168.1.0/24 any -P in ipsec esp/tunnel/213.96.80.51-80.36.214.182/require;
spdadd 213.96.80.51/32 80.36.214.182/32 any -P in ipsec esp/tunnel/213.96.80.51-80.36.214.182/require;
```

El siguiente paso es configurar el demonio IKE Racoon. Empezaremos por configurar la clave compartida entre hosts, editando el fichero */etc/racoon/psk.txt* en ambos gateways, de modo que quede como el siguiente:

```
213.96.80.51 shared_key
80.36.214.182 shared_key
```

Donde *shared_key* es una cadena de veinte caracteres ASCII aleatorios que podemos generar con el siguiente comando:

```
$ dd if=/dev/random count=20 bs=1 | xxd -ps
```

Nota: el comando *xxd* es parte del paquete *vim*.

Nota: *shared_key* debe tener el mismo valor en ambas líneas.

El siguiente paso es configurar el fichero de configuración del demonio ISAKMP Racoon con las direcciones de ambos extremos del túnel y las asociaciones de seguridad. Racoon negocia las asociaciones de seguridad por si mismo (SA de ISAKMP o SA de fase 1) y para el IPSec del kernel (SA de IPSec o SA de fase 2). El fichero está formado por una secuencia de directivas y sentencias. Cada directiva se compone de una etiqueta y las sentencias están enmarcadas entre llaves '{' y '}'. Las líneas que empiezan con '#' son comentarios. En la puerta de enlace A, 192.168.0.1, el fichero */etc/racoon/racoon.conf* quedaría tal que éste:

```
path pre_shared_key "/etc/racoon/psk.txt";
log notify;

listen
{
    isakmp 213.96.80.51;
    strict_address;
}

remote 80.36.214.182
{
    exchange_mode main;
    send_cr off;
    send_cert off;
    proposal {
        encryption_algorithm blowfish;
        hash_algorithm sha1;
        authentication_method pre_shared_key;
        dh_group 2;
    }
}

sainfo address 192.168.0.0/24 any address 192.168.1.0/24 any
{
    pfs_group 2;
    encryption_algorithm blowfish;
    authentication_algorithm hmac_sha1, hmac_md5;
    compression_algorithm deflate;
}

sainfo address 213.96.80.51/32 any address 192.168.1.0/24 any
{
    pfs_group 2;
    encryption_algorithm blowfish;
    authentication_algorithm hmac_sha1, hmac_md5;
    compression_algorithm deflate;
}

sainfo address 213.96.80.51/32 any address 80.36.214.182/32 any
{
    pfs_group 2;
    encryption_algorithm blowfish;
    authentication_algorithm hmac_sha1, hmac_md5;
    compression_algorithm deflate;
}

sainfo address 192.168.0.0/24 any address 80.36.214.182/32 any
{
    pfs_group 2;
    encryption_algorithm blowfish;
    authentication_algorithm hmac_sha1, hmac_md5;
    compression_algorithm deflate;
}
```

A continuación se explica el propósito de las directivas y sentencias más relevantes de las usadas en el fichero de configuración anterior. De ámbito general destacan estas dos directivas:

- **path pre_shared_key**: especifica el fichero que contiene las claves pre-compartidas para los diversos identificadores.
- **log**: define el nivel de log, que puede ser *notify*, *debug* o *debug2*. Por defecto se usa *notify*. Es importante tener en cuenta que, si se usa un nivel de logging demasiado elevado en una máquina lenta, pueden producirse timeouts durante la negociación.

De la directiva *listen* destacan las siguientes sentencias:

- **listen**: si no se especifica esta directiva, racoon escuchará en todas las direcciones disponibles.
- **isakmp**: el formato es *isakmp address [[port]]*. Si se especifica esta sentencia dentro de la directiva *listen*, racoon escuchará únicamente en la dirección especificada. El puerto por defecto es el 500, definido por la [IANA](#)⁽¹⁰⁾. Pueden establecerse varias sentencias *isakmp*.
- **strict_address**: esta sentencia hace necesario que todas las direcciones para ISAKMP estén asociadas con alguna interfaz de red.

De la directiva *remote* destacan las siguientes sentencias:

- **remote**: la directiva *remote* va seguida de una dirección o la palabra clave *anonymous*, según el formato *remote (address | anonymous) [[port]]*, y especifica los parámetros para la fase 1 de IKE para cada nodo remoto. El puerto por defecto es el 500. Si se especifica *anonymous*, las sentencias se aplicarán a todos los nodos a los que no se les apliquen otras sentencias *remote* específicas.
- **exchange_mode**: esta sentencia puede tener tres valores diferentes, *main*, *aggressive* y *base*, y define el modo de intercambio para la fase 1 cuando racoon es quien la inicia. También indica el modo de intercambio aceptado cuando racoon responde. Puede especificarse más de un modo, separados por comas. Todos los modos serán aceptados, pero el primero es el que se usará cuando racoon sea el iniciador de la negociación.
- **send_cr** y **send_cert**: si no se desea mandar una petición de certificado, debe ponerse a *off*, pues el valor por defecto es *on*. Lo cambiamos a *off* debido a que usamos una clave compartida y no certificados para establecer el túnel.
- **encryption_algorithm**: especifica los algoritmos de cifrado que se usarán en la fase 1. Esta sentencia es de obligatoria especificación. Los algoritmos pueden ser *des*, *3des*, *blowfish* o *cast128*.
- **hash_algorithm**: define el algoritmo de dispersión (del inglés, *hash*) que se usará durante la negociación de la fase 1. La definición de esta sentencia es obligatoria. El algoritmo puede ser *md5* o *sha1*.
- **authentication_method**: define el método de autenticación usado en la negociación de la fase 1 y es una sentencia que debe definirse obligatoriamente. Pueden darse los valores *pre_shared_key*, *rsasig*, *gssapi_krb*, *hybrid_rsa_server* o *hybrid_rsa_client*.
- **dh_group**: define el grupo que se usará para las exponenciaciones Diffie-Hellman. La definición de esta sentencia es obligatoria y puede coger uno de los valores *modp768*, *modp1024*, *modp1536*, *modp2048*, *modp3072*, *modp4096*, *modp6144* o *modp8192*. Alternativamente, pueden definirse los grupos DH como 1, 2, 5, 14, 15, 16, 17 o 18. Si se quiere usar el modo *aggressive* debe especificarse el mismo grupo DH en todas las propuestas.

De la directiva *sainfo* se usan las siguientes sentencias:

- **sainfo**: sigue el formato *sainfo (source_id destination_id | anonymous) [from idtype [string]]* y define los parámetros para la fase 2 de IKE (establecimiento de las asociaciones de seguridad de IPSec). *source_id* y *destination_id* se construyen según el formato *address address [/ prefix] [[port]] ul_proto*.
- **pfs_group**: define el grupo que se usará para las exponenciaciones Diffie-Hellman. La definición de esta sentencia es obligatoria y puede coger uno de los valores *modp768*, *modp1024*, *modp1536*, *modp2048*, *modp3072*, *modp4096*, *modp6144* o *modp8192*. Alternativamente, pueden definirse los grupos DH como 1, 2, 5, 14, 15, 16, 17 o 18.
- **encryption_algorithm**: especifica los algoritmos de cifrado que se usarán en la fase 2. Puede tomar los valores *des*, *3des*, *des_iv64*, *des_iv32*, *rc5*, *rc4*, *idea*, *3idea*, *cast128*, *blowfish*, *null_enc*, *twofish* o *rijndael* (usado con *ESP*).
- **authentication_algorithm**: define el algoritmo de autenticación usado en la negociación de la fase 2 y puede tomar los valores *des*, *3des*, *des_iv64*, *des_iv32*, *hmac_md5*, *hmac_sha1*, *non_auth* (usado con autenticación *ESP* y *AH*).
- **compression_algorithm**: define el algoritmo de compresión usado y puede tomar únicamente el valor *deflate* (usado con *IPComp*).

Por lo tanto, mediante las cuatro directivas *sainfo* que se configuran en este fichero */etc/racoon/racoon.conf* del gateway A, definimos cuatro asociaciones de seguridad para las conexiones salientes que se establecerán desde este host. Respectivamente:

- Las conexiones de la subred 192.168.0.0/24 a la subred 192.168.1.0/24.
- Las conexiones del host 213.96.80.51 (gateway A) a la subred 192.168.1.0/24.
- Las conexiones del host 213.96.80.51 (gateway A) al host 80.36.214.182 (gateway B).
- Las conexiones de la subred 192.168.0.0/24 al host 80.36.214.182 (gateway B).

El fichero */etc/racoon/racoon.conf* del gateway B (192.168.1.1/80.36.214.182) será muy parecido al anterior. Tan sólo debemos intercambiar las direcciones IP de las directivas *remote* y *listen* y modificar apropiadamente las directivas *sainfo* para que las direcciones IP sean las de origen de la subred 192.168.1.0/24 y el host 80.36.214.182 y las de destino las de la subred 192.168.0.0/24 y la del host 213.96.80.51. El fichero resultante sería el siguiente:

```
path pre_shared_key "/etc/racoon/psk.txt";
log notify;

listen
{
    isakmp 80.36.214.182;
    strict_address;
}

remote 213.96.80.51
{
    exchange_mode main;
    send_cr off;
    send_cert off;
    proposal {
        encryption_algorithm blowfish;
        hash_algorithm sha1;
        authentication_method pre_shared_key;
        dh_group 2;
    }
}

sainfo address 192.168.1.0/24 any address 192.168.0.0/24 any
{
    pfs_group 2;
```

```

    encryption_algorithm blowfish;
    authentication_algorithm hmac_sha1, hmac_md5;
    compression_algorithm deflate;
}

sainfo address 80.36.214.182/32 any address 192.168.0.0/24 any
{
    pfs_group 2;
    encryption_algorithm blowfish;
    authentication_algorithm hmac_sha1, hmac_md5;
    compression_algorithm deflate;
}

sainfo address 80.36.214.182/32 any address 213.96.80.51/32 any
{
    pfs_group 2;
    encryption_algorithm blowfish;
    authentication_algorithm hmac_sha1, hmac_md5;
    compression_algorithm deflate;
}

sainfo address 192.168.1.0/24 any address 213.96.80.51/32 any
{
    pfs_group 2;
    encryption_algorithm blowfish;
    authentication_algorithm hmac_sha1, hmac_md5;
    compression_algorithm deflate;
}

```

Antes de activar la VPN añadiremos opciones a la llamada al demonio *racoona* para que el log se haga en un fichero aparte y no en el *syslog*. Para ello modificaremos el script */etc/init.d/racoona* en ambos gateways y modificaremos el valor de la variable *RACOON_ARGS* en la línea 27, tal que:

```
RACOON_ARGS="-l /var/log/racoona.log"
```

De este modo podremos consultar el log en un fichero específico, */var/log/racoona.log* y no se nos mezclará con el log del sistema. Tras la configuración de estos ficheros reiniciaremos los demonios en ambos gateways:

```
/etc/init.d/setkey restart
/etc/init.d/racoona restart
```

En los logs deberían verse unas líneas tal que las siguientes:

```

$ tail -f /var/log/racoona.log | colorize
2005-05-27 16:34:15: INFO: @(#)ipsec-tools 0.5.2 (http://ipsec-tools.sourceforge.net)
2005-05-27 16:34:15: INFO: @(#)This product linked OpenSSL 0.9.7e 25 Oct 2004 (http://www.openssl.org/)
2005-05-27 16:34:15: INFO: 80.36.214.182[500] used as isakmp port (fd=6)
2005-05-27 16:34:15: INFO: 80.36.214.182[500] used for NAT-T

```

Nota: la dirección IP 80.36.214.182 será 213.96.80.51 dependiendo de en cuál de los dos gateways miremos el log.

Nota: *colorize* es un programa que colorea las líneas, especialmente pensado para facilitar la lectura de logs de la manera que se está haciendo en el caso arriba expuesto. Forma parte del paquete *colorize* en Debian.

Una vez hecho esto, la VPN propiamente dicha aún no se ha establecido, sino que lo hará (se negociarán las fases 1 y 2) cuando se efectúe la primera comunicación, por ejemplo un ping entre hosts. Una vez hecho esto podremos ver las líneas en el log donde se confirma que se ha establecido el túnel:

```

$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=108.7 ms

$ tail -f /var/log/racoona.log | colorize
2005-05-27 16:35:30: INFO: respond new phase 1 negotiation: 80.36.214.182[500]<=>213.96.80.51[500]
2005-05-27 16:35:30: INFO: begin Identity Protection mode.
2005-05-27 16:35:30: INFO: received Vendor ID: DPD
2005-05-27 16:35:31: INFO: ISAKMP-SA established 80.36.214.182[500]-
213.96.80.51[500] spi:cd02576d8b543669:71462f0526dab621
2005-05-27 16:35:32: INFO: respond new phase 2 negotiation: 80.36.214.182[0]<=>213.96.80.51[0]
2005-05-27 16:35:32: INFO: IPsec-SA established: ESP/Tunnel 213.96.80.51-
>80.36.214.182 spi=54491525(0x33f7985)
2005-05-27 16:35:32: INFO: IPsec-SA established: ESP/Tunnel 80.36.214.182-
>213.96.80.51 spi=212724624(0xcadeb90)

```

Nota: la respuesta inicial al ping tardará unos segundos pues el túnel de la VPN debe establecerse primero.

Finalmente, nos interesará establecer unas reglas en los cortafuegos de ambos gateways, de modo que el túnel ser permita únicamente entre esos dos hosts. Para ello podemos usar las siguientes reglas, que podemos incluir en las que ya tengamos:

```

IPTABLES=/sbin/iptables
INT_IFACE=eth0

```

```

EXT_IFACE=eth1
LOCAL_SUBNET=192.168.0.0/24
REMOTE_SUBNET=192.168.1.0/24
VPN_SRC=213.96.80.51
VPN_DST=80.36.214.182

# Enable packet forwarding
echo 1 > /proc/sys/net/ipv4/conf/all/accept_redirects
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/ip_dynaddr
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 0 > /proc/sys/net/ipv4/conf/all/accept_source_route
echo 0 > /proc/sys/net/ipv4/conf/all/log_martians
echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_all
echo 0 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses

${IPTABLES} --table nat --append POSTROUTING --source ${LOCAL_SUBNET} \
--destination ! ${REMOTE_SUBNET} --out-interface ${EXT_IFACE} \
--jump MASQUERADE

${IPTABLES} --append INPUT --in-interface ${EXT_IFACE} --proto udp \
--sport isakmp --dport isakmp --source ${VPN_DST} \
--destination ${VPN_SRC} --jump ACCEPT
${IPTABLES} --append INPUT --in-interface ${EXT_IFACE} --proto tcp \
--sport isakmp --dport isakmp --source ${VPN_DST} \
--destination ${VPN_SRC} --jump ACCEPT
${IPTABLES} --append INPUT --in-interface ${EXT_IFACE} --proto ah \
--source ${VPN_DST} --destination ${VPN_SRC} --jump ACCEPT
${IPTABLES} --append INPUT --in-interface ${EXT_IFACE} --proto esp \
--source ${VPN_DST} --destination ${VPN_SRC} --jump ACCEPT
${IPTABLES} --append OUTPUT --out-interface ${EXT_IFACE} --proto udp \
--sport isakmp --dport isakmp --source ${VPN_SRC} \
--destination ${VPN_DST} --jump ACCEPT
${IPTABLES} --append OUTPUT --out-interface ${EXT_IFACE} --proto tcp \
--sport isakmp --dport isakmp --source ${VPN_SRC} \
--destination ${VPN_DST} --jump ACCEPT
${IPTABLES} --append OUTPUT --out-interface ${EXT_IFACE} --proto ah \
--source ${VPN_SRC} --destination ${VPN_DST} --jump ACCEPT
${IPTABLES} --append OUTPUT --out-interface ${EXT_IFACE} --proto esp \
--source ${VPN_SRC} --destination ${VPN_DST} --jump ACCEPT

```

Nota: el parámetro del kernel `/proc/sys/net/ipv4/conf/all/rp_filter` bloquea el spoofing o engaño de direcciones IP y, debido a que ahora está deshabilitado, la comprobación deberán hacerla las propias reglas de *iptables*. Asimismo, estas reglas deberán aparecer debajo de las arriba escritas.

Nota: la cabecera IP de un paquete ESP no puede ser modificada por NAT (del inglés, *Network Address Translation*), por lo que no es posible combinar NAT con un canal IPSec (al menos directamente, pero existe NATT para ese propósito). Es decir, no debe haber NAT entre los dos gateways.

Sobre los algoritmos de cifrado y autenticación

¿Qué es la criptografía?

A medida que el campo de la criptografía ha ido avanzando, las líneas divisorias de lo que es o no criptografía se han vuelto borrosas. Hoy en día la criptografía puede considerarse como el estudio de las técnicas y aplicaciones que dependen de la existencia de problemas difíciles. El *criptoanálisis* es el estudio de cómo comprometer (saltarse) los mecanismos criptográficos, y la *criptología* (del griego, *kryptós lógos*, que significa *palabra oculta*) es la disciplina que combina la criptografía y el criptoanálisis. Para la mayoría de las personas, la criptografía hace referencia a mantener las comunicaciones privadas.

El cifrado es la transformación de los datos a una forma que sea casi imposible de leer por alguien que no tenga el conocimiento adecuado (la clave). Su propósito es asegurar la privacidad manteniendo la información oculta de cualquiera que no sea el destinatario, incluso de aquellos que tengan acceso a los datos cifrados. Descifrar es la acción opuesta a cifrar, y consiste en la transformación de los datos cifrados a una forma inteligible.

El cifrado y el descifrado generalmente requieren del uso de algún tipo de información secreta, conocida como clave. Algunos mecanismos de cifrado usan la misma clave tanto para cifrar como para descifrar, mientras que otros usan diferentes claves. Hoy en día la criptografía es más que cifrar y descifrar. La autenticación es una parte fundamental de la privacidad.

¿Qué es la criptografía de clave secreta?

La criptografía de clave secreta es conocida a veces como criptografía simétrica. Es la forma más tradicional de criptografía, en la cuál una única clave se usa tanto para cifrar como para descifrar el mensaje. La criptografía de clave secreta no sólo trata el cifrado, sino también la autenticación. Una de esas técnicas es la llamada *códigos de autenticación de mensajes (MAC)*.

El principal problema de los criptosistemas de clave secreta es conseguir que el remitente y el destinatario se pongan de acuerdo en la clave secreta que se usará sin que nadie más la descubra. Esto requiere un método mediante el cuál ambas partes puedan comunicarse sin

miedo a ser espiados. Sin embargo, la ventaja de la criptografía de clave secreta es que es generalmente más rápida que la criptografía de clave pública.

¿Qué es la criptografía de clave pública?

Para solventar el problema de la gestión de las claves en la criptografía de clave secreta, W. Diffie y M. Hellman crearon en 1976 el concepto de criptografía de clave pública. Los criptosistemas de clave pública tienen dos usos principales, el cifrado y las firmas digitales. En estos sistemas, cada persona recibe un par de claves, una llamada clave pública y la otra llamada clave privada. La clave pública es publicada, mientras que la privada se mantiene en secreto. La necesidad del remitente y el destinatario de compartir información secreta desaparece, pues todas las comunicaciones requieren únicamente las claves públicas, y nunca se transmite o comparte la clave privada. Cualquiera puede mandar un mensaje simplemente usando la información pública, pero el mensaje sólo puede descifrarse con la clave privada, que está únicamente en posesión del destinatario.

En un criptosistema de clave pública, la clave privada está siempre asociada matemáticamente a la clave pública. Por lo tanto, es posible realizar un ataque al sistema de clave pública deduciendo la clave privada de la pública. Típicamente, la defensa contra esto es conseguir que el problema de deducir la clave privada requiera la factorización de un número muy grande, algo computacionalmente intratable. Esta es la idea que hay detrás del criptosistema RSA de clave pública.

¿Qué es Diffie-Hellman?

El protocolo de acuerdo de claves Diffie-Hellman (también conocido como acuerdo de clave exponencial) fue desarrollado por Diffie y Hellman en 1976 y publicado en el sensacional documento "New Directions in Cryptography." El protocolo permite a dos usuarios intercambiar una clave secreta sobre un canal inseguro sin la necesidad de secretos previos.

El protocolo tiene dos parámetros de sistema p y g . Ambos son públicos y pueden ser usados por todos los usuarios de un sistema. El parámetro p es un número primo y el parámetro g (normalmente llamado generador) es un entero menor que p con la siguiente propiedad: para cada número n entre 1 y $p-1$, inclusive, hay una potencia k de g tal que $n = g^k \text{ mod } p$.

Supongamos que Alicia y José quieren acordar una clave secreta compartida usando el protocolo de acuerdo Diffie-Hellman. Procederían de la siguiente manera: Primero, Alicia genera un valor aleatorio privado a y José genera un valor aleatorio privado b . Tanto a como b son escogidos de entre los enteros. Entonces deben derivar sus valores públicos usando los parámetros p y g y sus valores privados. El valor público de Alicia es $g^a \text{ mod } p$ y el de José es $g^b \text{ mod } p$. Ahora deben intercambiar sus valores públicos y, finalmente, Alicia computa $g^{ab} = (g^b)^a \text{ mod } p$, y José computa $g^{ba} = (g^a)^b \text{ mod } p$. Debido a que $g^{ab} = g^{ba} = k$, Alicia y José ahora tienen una clave secreta compartida k .

La seguridad del protocolo depende del problema del logaritmo discreto. Asume que el cálculo de la clave compartida $k = g^{ab} \text{ mod } p$ es inasequible computacionalmente dados los valores públicos $g^a \text{ mod } p$ y $g^b \text{ mod } p$ cuando el número primo p es suficientemente grande.

¿Qué son SHA y SHA-1?

El *Secure Hash Algorithm (SHA)*, el algoritmo especificado en el estándar de dispersión segura o *Secure Hash Standard (SHS, FIPS 180)*, fue desarrollado por NIST. SHA-1 es una revisión de SHA que fue publicada en 1994 y que corregía un error no publicado de SHA. Su diseño es muy similar a de la familia de algoritmos de dispersión MD4 desarrollada por Rivest. SHA-1 está también explicado en el estándar ANSI X9.30 (parte 2).

El algoritmo toma un mensaje de menos de 264 bits de longitud y produce un mensaje resumido de 160 bits. El algoritmo es ligeramente más lento que MD5, pero la mayor longitud del resumen (en inglés, *digest*) resultante lo hace más seguro ante ataques de fuerza bruta. SHA es parte del proyecto Capstone.

¿Qué son MD2, MD4 y MD5?

MD2, MD4 y MD5 son algoritmos de resumen de mensajes desarrollados por Rivest. Su propósito son las aplicaciones de firma digital en las cuales un mensaje muy grande tiene que ser "comprimido" de manera segura antes de asignársele una clave privada. Estos tres algoritmos toman un mensaje de longitud arbitraria y producen un mensaje resumido de 128 bits. Pese a que las estructuras de estos algoritmos son considerablemente parecidas, el diseño de MD2 es bastante diferente del de MD4 y MD5. MD2 fue optimizado para máquinas de 8 bits, mientras que MD4 y MD5 está enfocado a máquinas de 32 bits. La descripción y el código fuente de estos algoritmos puede encontrarse en Internet en los RFCs 1319-1321.

MD2 fue desarrollado por Rivest en 1989. El mensaje es primero rellenado de modo que su longitud sea divisible por 16. Una verificación de 16 bytes es añadida luego al mensaje y el valor de dispersión es computado sobre el mensaje resultante. Rogier and Chauvaud han encontrado que pueden ocurrir colisiones si se omite el cálculo de la verificación. Este es el único resultado criptoanalítico conocido de MD2.

MD4 fue desarrollado por Rivest en 1990. El mensaje es rellenado primero para asegurarse que su tamaño más 64 sea divisible por 512. Una representación binaria de 64 bits de la longitud original del mensaje es concatenada entonces al mensaje. El mensaje es procesado en bloques de 512 bits según la estructura iterativa de Damgard/Merkle y cada bloque es procesado en tres ocasiones diferentes. Ataques a MD4 basados en la omisión de la primera o la tercera ronda fueron rápidamente desarrollados por Den Boer, Bosselaers y otros. Dobbertin ha demostrado como pueden ocurrir colisiones en MD4 en menos de un minuto en un PC típico. En un trabajo reciente, Dobbertin (*Fast Software Encryption*, 1998) ha demostrado que una versión reducida de MD4 en la cual la tercera ronda de compresión no se lleve a cabo pero el resto se deje igual es reversible. Por lo tanto, claramente MD4 puede considerarse roto.

MD5 fue desarrollado por Rivest en 1991. Es básicamente un MD4 con cinturón de seguridad y, pese a ser ligeramente más lento que MD4, es mucho más seguro. El algoritmo consiste en cuatro rondas de compresión diferentes, con leves variaciones de diseño respecto de MD4. El tamaño del mensaje comprimido, así como los requerimientos de relleno, son los mismos. Den Boer y Bosselaers han encontrado pseudocolisiones en MD5. Un trabajo más reciente de Dobbertin ha extendido las técnicas usadas tan eficientemente en el análisis de MD4 para hallar colisiones en las funciones de compresión de MD5.

Van Oorschot y Wiener han trabajado en la búsqueda por fuerza-bruta de colisiones en las funciones de dispersión y estiman que una máquina diseñada específicamente para la búsqueda de colisiones (con un coste de 10 millones de dólares en 1994) tardaría una media de 24 días en encontrar una colisión. Las técnicas generales pueden aplicarse también a otras funciones de dispersión.

Bibliografía

- [Adam Sherman Online: Linux 2.6 IPsec VPNs](#)⁽¹¹⁾
- [How to easily build a VPN with KAME IPsec \(Kernel 2.6\)](#)⁽¹²⁾
- [Debian IPsec Micro-Howto](#)⁽¹³⁾
- [IPSEC using Linux Kernel 2.6](#)⁽¹⁴⁾
- [RSA Laboratories Crypto FAQ](#)⁽¹⁵⁾

Historial de revisiones

Fecha	Versión	Cambios
10/08/2005	1.0	Documento inicial

Lista de enlaces de este artículo:

1. <http://www.kame.net/>
2. <http://www.kernel.org/>
3. <http://ipsec-tools.sourceforge.net/>
4. <http://www.kame.net/racoon/>
5. <http://www.debian.org/>
6. <http://www.netfilter.org/>
7. <http://www.kernel.org/pub/linux/kernel/v2.4/>
8. <http://www.freeswan.org/>
9. <http://www.kernel.org/pub/linux/kernel/v2.6/>
10. <http://www.iana.org/>
11. <http://www.sherman.ca/archives/2004/11/21/linux-26-ipsec-vpns/>
12. http://users.cjb.net/ipsec/index.en_us.html
13. <http://www.fukt.bth.se/~teddy/debian-ipsec>
14. <http://shorewall.net/IPSEC-2.6.html>
15. <http://www.rsasecurity.com/rsalabs/node.asp?id=2153>

E-mail del autor: jsabater_ARROBA_linuxsilo.net

Podrás encontrar este artículo e información adicional en: <http://bulma.net/body.phtml?nIdNoticia=2269>