FREES/WAN: Protocolo IPSEC, IP seguro

Alejandro Aguilera Uguet de Resayre
Juan Pedro Sánchez Martín

Objetivos y descripción del servicio

Linux FreeS/WAN es una implementacion de IPSEC e IKE para Linux.

IPSEC es **I**nternet **P**rotocol **SEC**urity. Usa criptografía fuerte para proveer servicios de cifrado y autenticación. La Autenticación asegura que los paquetes son del remitente correcto y no han sido cambiados en su tránsito. El cifrado previene la lectura indebida del contenido de los paquetes

.

Estos servicios permiten construir **túneles seguros a través de redes inseguras.** Todo lo que pasa por la red insegura es cifrado por el gateway IPSEC en un extremo y descifrado por el gateway IPSEC del otro extremo. El resultado es una **Red Virtual Privada** o VPN. Esto es una red que es segura aunque incluya máquinas en diferentes sitios conectadas por una conexión insegura a Internet.

1. Requisitos

FreeS/WAN se puede instalar sobre cualquiera de los siguientes Kernels 2.0.3X, 2.2.1X o 2.4.X, nosotros intentando no cambiar lo establecido hasta el momento en el laboratorio lo hemos instalado sobre el Kernel 2.2.19.

Necesitaremos también tener el parche para proveer el soporte para IPSEC al Kernel, que se llama *kernel-patch-freeswan,* ya que el Kernel no soporta este protocolo, lo bajamos de la direccion:
http://ftp.de.debian.org/debian-non-US/pool/non-US/main/f/freeswan/kernel-patch-freeswan_1.9-1_all.deb

Por supuesto se ha de bajar ha traves de ftp el paquete freeS/WAN desde:
http://ftp.de.debian.org/debian-non-US/pool/non-US/main/f/freeswan/freeswan_1.9-1_i386.deb

Adicionalmente nos hemos bajado el paquete *kernel-package* para facilitar la recompilacion del nucleo.

2. Arquitectura del servicio

Los protocolos usados son los 3 siguientes:

1.- **AH** (Authentication Header) provee un servicio de autentificación de paquetes.

2.- **ESP** (Encapsulating Security Payload) provee la encriptación y la autentificación.

3.- **IKE** (Internet Key Exchange) realiza la negociación de los parámetros de la conexión, algoritmos aceptables, tamaños de clave, configuración de claves. Provee todo lo necesario para establecer, renegociar claves, reparar o desconectar la conexión.

La implementación tiene tres partes principales

1.- **KLIPS** (kernel IPSEC) implementa AH, ESP, y el manejo de paquetes con el kernel

2.- **Pluto** (un daemon IKE) implementa IKE, negocia conexiones con otros sistemas.

3.- Varios scripts para proveer una interfaz entre el administrador y la máquina.

Paquetes IPSEC

IPSEC utiliza tres tipos de paquetes principalmente

IKE utiliza el **protocolo UDP sobre el puerto 500**.

ESP es un **protocolo número 50**

AH es un **protocolo número 51** .

ESP y AH no utilizan puertos.

Algunos protocolos como TCP y UDP necesitan el concepto de puerto. Otros protocolos, entre los que se encuentran ESP y AH, no. No existen puertos en los protocolos ESP o AH , y por tanto no se utiliza ningún puerto para ellos. Para estos protocolos *la idea de puerto es completamente irrelevante*.

3. Configuración

3.1 Instalación

La primera dificultad que nos encontramos es la instalación del servicio, ya que hay que aplicar un parche al Kernel y recompilarlo debido a que este no nos ofrece soporte para IPSEC.

Una vez que tengamos descomprimido el Kernel que vamos a utilizar e instalado el parche, deberemos copiar el script llamado *freeswan* que encontraremos en */usr/src/kernel-patches/all/apply/* al directorio raíz del código fuente del kernel y ejecutarlo.

Posteriormente si tenemos instaladas las X haremos *make xconfig*, sino *make menuconfig*, aquí deberemos elegir las opciones para la compilación del núcleo, tenemos que fijarnos en:

REQUERIDOS	**DESACTIVADOS**	**RECOMENDADOS**	**OPCIONALES** (Según el uso)
Networking support	Kernel module loader	Network firewalls	Prompt for development ... code/drivers
Unix domain sockets	Socket filtering	IP: Optimize as router not host	Enable loadable module support
TCP/IP networking	IP: kernel level autoconfiguration	IP: TCP syncookie support	Set version information ....
IP: tunneling	IP: firewall packet netlink device	IP: large window support	Sysctl interface
Character devices -- random(4)	Novell IPX		Packet socket (Usado por tcpdump)
	Appletalk		Kernel/User netlink socket
	Network filesystems		Routing messages
	Kernel hacking		Netlink device emulation
			IP: advanced router
			IP: masquerading
			IPv6

El resto de parámetros no son importantes a la hora de instalar FreeS/WAN, aunque habrá que fijarse en ellos según los requisitos de nuestra máquina para no estropear nada.

Ahora podremos continuar con la compilación del Kernel siguiendo los pasos que nos dice el programa al salir grabando, o podemos hacer *make-kpkg* si hemos instalado el paquete *kernel-package*.

Una vez recompilado el Kernel hemos de actualizar nuestro fichero lilo.conf para arrancar con el nuevo kernel.

Hacemos lo siguiente:

cp /usr/src/kernel-sourcexxxx/arch/i386/boot/bzImage /boot/vmlinuz-2.X.X
cp /usr/src/kernel-sourcexxxx/vmlinux /boot/vmlinux-2.X.X
cp /usr/src/kernel-sourcexxxx/System.map /boot/System.map-2.X.X

vi /etc/lilo.conf

Cambiamos:

default=linux.new

Y añadimos:

image=/boot/vmlinuz-2.X.X
label=linux.new
read-only

Ejecutamos lilo y reseteamos la maquina.

Para comprobar que la instalacion ha sido correcta ejecutamos ifconfig y observamos si nos aparece un interfaz virtual llamado ipsec0 de la siguiente manera:

```
eth0      Link encap:Ethernet  HWaddr 00:60:08:E9:1D:BE
          inet addr:10.0.2.2  Bcast:10.0.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:898 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1256 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:9 Base address:0xd400

ipsec0    Link encap:Ethernet  HWaddr 00:60:08:E9:1D:BE
          inet addr:10.0.2.2  Mask:255.255.255.0
          UP RUNNING NOARP  MTU:16260  Metric:1
          RX packets:338 errors:0 dropped:338 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:224 errors:0 dropped:0 overruns:0 frame:0
          TX packets:224 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

3.2 Configuración

A la hora de configurar FreeS/WAN debemos de tener en cuenta la configuracion de dos ficheros: ipsec.conf e ipsec.secrets

En ipsec.conf se encuentran todos los parámetros de la conexión, direcciones IP, identificadores, claves publicas, subredes y algunos parámetros mas generales, como los métodos de autentificación, los parámetros de los que hacer debug, etc……

En ipsec.secrets se encuentra la clave privada y ha de estar muy bien protegida, de forma que solo pueda leerla el administrador del sistema.

3.2.1 Ipsec.conf

Basicamente distinguimos 2 secciones, la de configuración(config setup) y la de conexión(conn), los parametros que se pueden configurar en cada una de ellas son los siguientes:

Seccion config:

Comienza por config setup sin ningun espacio antes, los siguientes parametros han de estar con sangria, los parametros mas importantes son:

interfaces	Interfaces virtuales y fisicos usados por IPSEC, pares "virtual=fisica" o %defaultroute
forwardcontrol	Como debe hacer el IP forwarding, por defecto es no, para lo que deberemos haber activado IPforward
syslog	Si se ha de realizar salida de errores al syslog
klipsdebug	Como se debe hacer la salida de logs de KLIPS, por defecto esta a none, si se quiere activar poner all
plutodebug	Lo mismo que el anterior pero para Pluto
pluto	Si se ha de iniciar Pluto al comienzo, por defecto yes
plutoload	Que conexiones se han de cargar en la base de datos interna de Pluto al arrancar, hay un valor especial que es %search
plutostart	Que conexiones hay que ejecutar al arrancar
uniqueids	Si el identificador de una conexión para un usuario ha de ser único, en el caso de existir dos conexiones para el mismo usuario, la nueva reemplazaría a la antigua. Por defecto no

Seccion conn:

Comienza con *conn %default* si a continuacion ponemos parametros validos para todas las conexiones o co*nn "nombre"*

type	Define el tipo de conexión los valores validos son *tunnel*(por defecto) o *transport*
compress	Si se quiere o no compresión de los datos, antes de la encriptación, por defecto *n*o
left/right	Dirección IP de cada participante en la conexión, valores especiales que se pueden dar son *%defaultroute* (dirección por defecto), *%any* (cualquiera) o *%opprotunistic* (localizar en el DNS).
left/rightsubnet	Subred de cada participante. Formato network/netmask (ej. 10.0.0.2/24)
left/rightnexthop	Siguiente maquina dentro de la red.
auto	Sirve para especificar que operación ha de ser realizada cuando IPSEC comienza. Los valores aceptados son: *add*(añadir a la BD de conexiones), *start*(comenzar la conexión), *route*(mirar la ruta), *ignore*(ignorar).
auth	Método de autentificación a utilizar: *esp*(por defecto) o *ah*
authby	Como se han de autentificar los gateways entre ellos: *secret* (por defecto), *rsasig*.
left/rightid	identificador de la maquina.
left/rightrsasig	La clave publica de la maquina utilizada para la autentificación mediante RSA. Un valor especial es *%dns* el cual busca las claves en el DNS.
pfs	Indica si se ha de usar Perfect Forward Secrecy, por defecto *yes*
keylife	Tiempo de vida de las claves utilizadas para autentificación/encriptación
keyingtries	Cuantos intentos se han de realizar a la hora de negociar una conexión.

3.2.2 ipsec.secrets

En este archivo se encontraran las claves privadas de nuestro gateway para la conexión, pueden existir distintas claves privadas dependiendo de la maquina con la que nos conectemos.

Es muy importante que solo nosotros tengamos acceso a este archivo de forma que lo que debemos hacer es cambiarle los privilegios de acceso haciendo : *chmod 660 /etc/ipsec.secrets*.

Actualmente soporta dos tipos de secretos: preshared secrets(clave compartida) o clave privada RSA.

La principal forma de configurar estas claves es de la siguiente forma:

(Id máquina o conexión IP IP) : tipo de clave(PSK o RSA)
CLAVE: "psk" o {RSAKEY}

3.2.3 Obtención de claves publicas y privadas.

Aunque las claves se pueden obtener desde otros programas como: pgp, openssl, freebsd,..., hemos utilizado la herramienta que ofrece FreeS/WAN para la generación de claves, esta se utiliza de la siguiente forma:

ipsec ranbits 192 > temp. Para encriptacion tipo 3DES.
ipsec ranbits 128 > temp. Para MD5.

3.3 Configuración de nuestro sistema

Nosotros hemos configurado el sistema utilizando, una facilidad que ofrece FreeS/WAN la cual permite crear un fichero nuevo para cada conexión, dejando los parámetros comunes en ipsec.conf y solo cambiando los parámetros particulares de cada conexión, este parámetro es include.
Se crearon dos conexiones una entre ambos servidores y otra para acceder a la subred 10.0.0.0 desde la maquina 10.0.2.2 usando como gateway la 10.0.2.1.

Asi mismo hemos creido mas conveniente el uso de algoritmos de clave publica y privada de forma que lo unico que han de conocer los dos gateways uno del otro sea su clave publica. Esta clave publica para mayor facilidad en su uso se ha colocado en el DNS ya configurado en el laboratorio, de forma que sea mas sencilla su configuración.

El ejemplo de los ficheros de configuración generados en el laboratorio para la prueba del software se recogen en el apéndice A de este documento.

4. Pruebas

Para verificar la puesta en funcionamiento del encriptado y desencriptado del nuevo protocolo IPSEC en el kernel 2.2.19 lo instalamos en el gateway de la subred y en una máquina de esa misma subred. En ambas máquinas la interfaz 'eth0' queda asociada a la interfaz virtual 'ipsec0'.Utilizamos la herramienta ping contra el gateway de la subred m1.plan.acme y central.acme, desde m2, de la siguiente forma:

```
ping   -p   feedfacedeadbeef   m1
ping   -p   feedfacedeadbeef   central
```

Desde el gateway de la subred, configurando la tarjeta ethernet 'eth0' en modo promiscuo, leemos la información de los paquetes que llegan al mismo en ambas interfaces (eth0 e ipsec0) mediante la herramienta tcpdump de la siguiente forma:

en la interfaz virtual **ipsec0** (salida desencriptada):

tcpdump –x –i ipsec0

obteniendo el siguiente resultado al leer los paquetes:

16:12:52.529265 m2.plan.acme > m1.plan.acme: icmp: echo request
4500 0054 0b30 0000 4001 5777 0a00 0202
0a00 0201 0800 2784 180d 0300 0aa1 ea3b
2c72 0800 **feed face dead beef feed face**
dead beef feed

resultado de la conexión punto a punto entre las 2 máquinas
…..
…..
16:12:29.549320 m2.plan.acme > central.acme: icmp: echo request
4500 0054 0b03 0000 4001 59a4 0a00 0202
0a00 0001 0800 ec35 170d 3200 f3a0 ea3b
50c0 0800 **feed face dead beef feed face**
dead beef feed

resultado de una conexión en la que el gateway enruta los datos después de desencriptarlos.

Se verifica por tanto que la interfaz virtual ipse0 encripta y desencripta la información de las paquetes. Por el contrario con la herramienta tcpdump escuchando en la interfaz eth0 se tiene el siguiente resultado:

en la interfaz **eth0:**

tcpdump -x -i eth0

16:13:06.529301 m2.plan.acme > m1.plan.acme: ip-proto-50 116
4500 0088 0b4d 0000 4032 56f5 0a00 0202
0a00 0201 9f5d f5d8 0000 0066 8c52 61b7
38bc dde2 27b2 f773 8e2f aceb d7ba d9b5
425f d189 1a7b
resultado de la conexión punto a punto entre las 2 máquinas
…..
…..
16:11:55.549289 m2.plan.acme > m1.plan.acme: ip-proto-50 116
4500 0088 0abd 0000 4032 5785 0a00 0202
0a00 0201 9f5d f5d7 0000 0033 9c80 caf0
0a04 d937 c517 074e e08d 8754 04ea 8e5c
e0a6 bf3b 63ce

resultado de una conexión en la que el gateway enruta los datos después de desencriptarlos.

Aquí se observa realmente los paquetes que iran a traves de la red de forma encriptada, ya que no se consigue entender nada

5. Gestión diaria

La gestión diaria si todo funciona bien se reduce a introducir nuevos usuarios en el sistema , para lo cual se nos ocurre que incluso podría crearse un script que realizara todos los pasos automáticamente.

A la hora de reiniciar los servicios se realiza como con el resto de demonios del sistema, ejecutando */etc/INIT.d/ipsec stop|start|restart*.

Existe ademas un comando denominado ipsec que sirve para lanzar y parar conexiones (*ipsec auto/manual --up/down*), o incluso observar el funcionamiento del sistema realizando una batería de pruebas y enseñando los logs generados por el sistema (*ipsec barf*).

APÉNDICE A

En este apéndice aparecen todos los ficheros de configuración generados para configurar el sistema, como ejemplo se pondran los de una sola maquina:
ipsec.conf:

```
# /etc/ipsec.conf - FreeS/WAN IPsec configuration file
# More elaborate and more varied sample configurations can be found
# in FreeS/WAN's doc/examples file, and in the HTML documentation.

# basic configuration
config setup
      # THIS SETTING MUST BE CORRECT or almost nothing will work;
      # %defaultroute is okay for most simple cases.
      interfaces="ipsec0=eth0"
      # Debug-logging controls:  "none" for (almost) none, "all" for lots.
      klipsdebug=none
      plutodebug=none
      # Use auto= parameters in conn descriptions to control startup actions.
      plutoload=%search
      plutostart=%search
      # Close down old connection when new one using same ID shows up.
      uniqueids=yes



# defaults for subsequent connection descriptions
conn %default
      # How persistent to be in (re)keying negotiations (0 means very).
      keyingtries=0
      # RSA authentication with keys from DNS.
      authby=rsasig
      leftrsasigkey=%dns
      rightrsasigkey=%dns

include /etc/ipsec/*.ipsec
```

Y un ejemplo de configuración de una de las conexiones:

```
conn sample
        # Left security gateway, subnet behind it, next hop toward right.
        left=10.0.2.2
        leftrsasigkey=%dns
        leftid=@m2.plan.acme
        # Right security gateway, subnet behind it, next hop toward left.
        right=10.0.2.1
```

```
        rightrsasigkey=%dns
        rightid=@m1.plan.acme
        rightsubnet=10.0.0.0/24

      # To authorize this connection, but not actually start it, at startup,
# uncomment this.
        auto=start
```

Ahora podemos observar las lineas añadidas a la configuracion del DNS en /etc/bind/db.plan.acme:

```
m1    IN    KEY   0x4200      4     1
AQOQvZDOhF24WuEpxHl6ciipyQ5HcNswBuBeKU0Vx5FGg0SoF7zDzY05O8fiXnXaN9rhWkLrF71Cq
6WHO8SFdo9AHd07r96QSpkvuqeJuj7EBEgQZB11L//KI9FJYbW8yCj7XlQmKepr2sE/yH2zRBPSt8
hFYPTkJ5vCv6RIIZHcXw==
m2    IN    KEY   0x4200      4     1
AQOyQK2n3BKfgFQ4MiIknU8U/wyeahPqEgfZbr7wPHnMbcUUIUVcpYJdPPURBw2i7xtL5ZHxwKJkU
AxQsQqkNAHvtUwxtAlmLcWXsET8m3rSusEMbEKN1Qp9yq8etRHNfX1hXvcrqTQMusQtcAfxSh7qZV
gbD/TfBlsnmyzYGE/pkw==
```

El cual pone las claves publicas para ambas maquinas. Las claves privadas las encontraremos en el fichero ipsec.secrets, y estara configurado de la siguiente forma:

```
: RSA {
      # RSA 1024 bits   m2   Tue Nov  6 20:42:12 2001
      # for signatures only, UNSAFE FOR ENCRYPTION
      #pubkey=0sAQOQvZDOhF24WuEpxHl6ciipyQ5HcNswBuBeKU0Vx5FGg0SoF7zDzY05O8fiX
nXaN9rhWkLrF71Cq6WHO8SFdo9AHd07r96QSpkvuqeJuj7EBEgQZB11L//KI9FJYbW8yCj7XlQmKe
pr2sE/yH2zRBPSt8hFYPTkJ5vCv6RIIZHcXw==
      Modulus:0x90bd90ce845db85ae129c4797a7228a9..................
      PublicExponent: 0x03
      # everything after this point is secret
      PrivateExponent: 0x181f98226b64f40f2586f6143f1306c6f6.............
      Prime1: 0xc9b2211c976713791add5528cc4b05b3...................
      Prime2: 0xb7b5cb53078118d4a9334828cd.....................
      Exponent1: 0x8676c0bdba44b7a611e8e3708.......................
      Exponent2: 0x7a7932375a56108dc622301b33.......................
      Coefficient: 0xb9a73d529d68f8cfefa7740163.......................
      }
 En este fichero no se dejan las claves enteras para una mayor claridad del
contenido.
```