

Túneles Cifrados con FreeS/WAN

Daniel Esteban Coletti

1. Introducción

FreeS/WAN es una implementación de los protocolos IPSec (IP Security) en Linux. IPSec provee servicio de cifrado y autenticación a nivel IP (Internet Protocol). En otras palabras, IPSec le da a ipv4 algo que no tiene, seguridad de datos (cifrado¹) y autenticación.

En este documento voy a tratar de explicar lo más claramente posible cómo utilizar esta implementación y también voy a escribir sobre experiencias mías utilizando FreeS/WAN. Pero antes algunos comentarios:

1.1. Nomenclatura (Algunas formas que tengo de escribir):

- Cuando veas un # significa que es una línea de comando, y el # es el prompt
- Todo lo encerrado entre corchetes “[]” son datos opcionales
- Todo lo encerrado entre mayor y menor “< >” son datos obligatorios
- Si bien está escrito en español, no utilizo español estricto dado que no fuerzo la traducción de palabras que son más bien utilizadas en su forma original (inglés) al hablar. Soy argentino y no escribo de “tu”, sino de “vos”

2. Nota importante

Este artículo puede contener errores y vos deberías asumir que los contiene, lo escrito acá no implica ningún tipo de responsabilidad de mi parte, de ninguna persona y/o empresa mencionada. Cualquier tipo de consecuencia que surja de seguir estas indicaciones son responsabilidad *tuya* o de quien las siga/haga. En otras palabras, yo no me hago cargo de nada, cualquier cosa que hagas será *tu* responsabilidad.

3. Sobre VPNs

Las VPNs o Virtual Private Networks (Redes Privadas Virtuales) son una forma de emular el uso de una red privada sobre otra red (generalmente pública). Una red privada es como la red que tenemos en casa o la oficina², la idea de las VPNs es lograr que diferentes puntos de una red pública como internet, parezcan ser puntos de una red propia, en otras palabras es tener la posibilidad de transmitir paquetes a números de ip privados a través de internet.

Por ejemplo, el ADSL que tengo en casa es la puerta de enlace a las estaciones de trabajo y servidores de mi sector de RR.HH., el modem que tiene mi amigo en su casa es el acceso al sector de soporte técnico, y finalmente el vínculo con número de IP fija que tengo en la oficina del centro es el acceso a mis servidores de correo, intranet y autenticación.

Bueno, entonces mirando la figura podemos permitirnos imaginar una conexión directa entre el equipo de RRHH (mi casa) al CVS donde están los fuentes de mi software, ahora, esto en realidad no se puede hacer a través de internet, las direcciones 192.168.X.X no son “ruteables” (son una de las tantas direcciones para exclusivo uso de redes internas, y ningún router de internet va a saber a dónde enviar paquetes que provengan y/o vayan a éstas direcciones). Por otro lado, si esto pudiera hacerse, llevarlo a cabo sería una locura, navegar en mi intranet, autenticarme contra el servidor NIS/LDAP, revisar los correos ... sería como anunciar sobre todo lo que ocurre en mi red local a la red pública (muchos podrían ver mis datos sin problema alguno) y por último, cualquiera podría hacerse pasar por mí (o cualquiera de los usuarios de mi red local). Por lo tanto, es sumamente importante poder cifrar los datos que envío desde y hacia los diferentes puntos de mi red local virtual y también que los diferentes puntos de mi red local puedan estar seguros que quien los envía es parte de la red.

En términos generales, esto es lo que hace en una VPN que cifra el tráfico que pasa a través de ella (porque también hay VPN que no hacen el cifrado).

4. Sobre FreeS/WAN

Como dije antes, el proyecto FreeS/WAN es la implementación de IPSec en Linux. El proyecto comprende dos grandes áreas, una es el código que se agrega al núcleo de Linux (actualmente es un parche separado ya que no integra el núcleo) y la otra parte es el código de las herramientas que el usuario utiliza para hacer que se establezcan los túneles (entre otras cosas).

Los fuentes de este proyecto se puede obtener de sitio oficial de FreeS/WAN (<http://www.freeswan.org>).

Dado que FreeS/WAN es una implementación de IPSec en Linux podríamos inferir que no es la única implementación de IPSec en Linux, y que además IPSec esta implementado en otros sistemas operativos. Que yo sepa no hay otra implementación de IPSec para Linux, y en cuanto a la segunda inferencia, es justamente por esto que es bueno utilizar IPSec al realizar una VPN, existen muchos otros sistemas operativos que tienen implementado IPSec (además de que es sumamente seguro) y esto permite que con Linux podamos establecer túneles cifrados contra otras redes que tengan sistemas operativos diferentes. En el sitio de FreeS/WAN hay una linda tablita de interoperabilidad, recomiendo que la revise para saber contra qué se pueden establecer túneles.

5. Cómo se hace (y qué se necesita)

Ya entendiendo que es una VPN y para qué se usa, vamos a empezar a ver cómo se usa FreeS/WAN y cómo se hace para armar la VPN. Para esto vamos a necesitar por lo menos dos equipos con Linux³ que se puedan conectar de algún modo a Internet (en forma directa⁴). Vamos a instalar FreeS/WAN en ámbos y, luego de determinar qué tipo de túneles vamos a establecer y el tipo de autenticación que se va a utilizar, vamos a poner a funcionar nuestra VPN.

Pero antes de hacer todo esto, aún tenemos que entender (bastante bien) que significa esto de “determinar que tipo de túneles vamos a establecer” y también esto de “tipo de autenticación que se va a utilizar”.

5.1. Tipos de túneles

La autenticación es el momento inicial cuando desde un nodo de la VPN se quiere establecer un túnel hacia otro nodo. Lo primero que tienen que hacer los nodos es asegurarse que cada uno es quien dice ser.

Para realizar esta tarea existen varios métodos. En FreeS/WAN he utilizado tres tipos diferentes, estos son: pre-shared-keys (claves compartidas preacordadas) donde los dos nodos saben de antemano qué clave van a utilizar, claves RSA (uso de claves asimétricas) para utilizar este método se generan claves asimétricas en cada nodo de la red y las partes públicas de cada clave son esparcidas por toda la red, de esta forma solamente aquellos que tengan la parte privada de la clave serán los verdaderos dueños de las partes públicas esparcidas, y por último la autenticación con certificados X509, donde sólo aquellos certificados que estén firmados por una entidad certificante en la cual nosotros confiemos, serán considerados como nodos válidos. Si bien todos los métodos tienen sus pros y contras, mi preferencia personal es la autenticación con certificados por su simpleza de administración y por su gran interoperabilidad con otras plataformas (es común que otras implementaciones de IPSec utilicen esta autenticación).

6. Instalación

FreeS/WAN tiene dos partes principales, el parche de ipsec para el núcleo (KLIPS) que implementa AH (Authentication Header -- Encabezado de Autenticación), ESP (Encapsulating Security Payload - No se como se traduce) y el manejo de paquetes dentro del núcleo. La otra parte son las herramientas de usuario (**pluto**). Pluto implementa IKE (Internet Key Exchange -- Intercambio de Llaves de Internet), es quien negocia las conexiones contra otros equipos.

Lo primero que hay que hacer es lograr que el núcleo entienda ipsec. Esto se puede hacer de varias formas, una es aplicando el parche a los fuentes del núcleo, compilarlo (con las opciones requeridas) y ejecutarlo, la otra forma es utilizar algún módulo de FreeS/WAN ya compilado y así hacer que el núcleo entienda ipsec utilizando la capacidad modular de los núcleos de Linux.

La gente de FreeS/WAN ha hecho un estupendo trabajo documentando todo el proceso de compilación, por lo que yo no voy a describir este método en detalle, lo único que sí voy a comentar (que no está en la documentación de FreeS/WAN) es que hay que tener cuidado si se están “probando” los métodos de instalación. Si uno decide instalar los paquetes y utilizar el módulo, ni siquiera hay que intentar compilar el parche y las herramientas, ya que el proceso para realizar esto sobrescribe los binarios de los paquetes sin hacer muchas preguntas. Mi recomendación es instalar los paquetes, es más fácil, más rápido y no hay que responder tantas preguntas.

En el sitio que produce los parches para X509 (<http://www.freeswan.ca>) generan paquetes RPMS (aún sino se va a utilizar este método de autenticación, los paquetes son válidos y se pueden usar con otros métodos), también generan los paquetes “Super Freeswan” que son paquetes que vienen con muchos parches “no oficiales” para FreeS/WAN (muy útiles). Una vez descargados del sitio web, se instalan los paquetes de esta forma.

```
#rpm -Uvh freeswan-<version>-module.rpm  
#rpm -Uvh freeswan-<version>.rpm
```

7. Entendiendo la configuración de FreeS/WAN

La gente de FreeS/WAN ideó una forma de escribir los archivos de configuración que al principio es un poco difícil de entender. Cuando ellos armaron esto quisieron hacer que los archivos de configuración de todos los nodos de la red VPN fueran iguales (o extremadamente parecidos), por lo que en vez de utilizar nombres de variables como “remoto” o “local” (dado que se deben invertir dependiendo el lugar donde uno esté) utilizaron variables como “izquierda” o “derecha”. De esta forma, no importaba qué nodo uno estuviera configurando, el “nodo izquierdo” siempre es el mismo en todos los puntos (lo mismo sucede con el derecho, por supuesto). Lo cierto es que la complejidad de las cosas no siempre juega a nuestro favor, con lo cual son menos las veces donde los archivos de configuración de los nodos son iguales, que las que lo son diferentes. Por otro lado, ellos prefirieron utilizar este tipo de sintaxis para disminuir la confusión que puede generar el uso de palabras como “local” o “remoto” al implementar la VPN (!?).

Por último hay muchos ejemplos en la documentación de FreeS/WAN que supone números de IP fijos, esto es algo que generalmente no esta accesible, por lo que en esta documentación te vas a encontrar con ejemplos más orientados a números de IP dinámicos.

8. Autenticación

Lo primero que hace IPSec a la hora de establecer un túnel es determinar si el equipo que intentar establecer un túnel con nosotros es quien dice ser, y que además es un equipo válido con el cual queremos establecer un túnel. Luego entre las partes deciden que claves van a utilizar y finalmente (luego de muchas otras cosas en el medio) el tunel se establece y ámbas partes empiezan a “hablar” con el protocolo ESP (y AH en algunos casos).

Para autenticar se puede usar muchos métodos, yo he realizado exitosamente pruebas con tres tipos de autenticacion: *pre shared keys* (claves compartidas preacordadas), firmas digitales RSA y certificados X509.

8.1. Pre Shared Keys

Este tipo de autenticación es cuando se utiliza una única clave en todos los nodos de la VPN, esta clave es una clave en texto plano. El nodo que esta recibiendo la conexión de otro nodo de la VPN determina que el nodo realizando la conexión es válido si tiene la clave correcta.

Los “pro” que tiene utilizar *pre shared keys* son dos: uno es la “facilidad” de configurar este tipo de autenticación, y dos que es un tipo de autenticación que la gran mayoría de implementaciones de IPSec tienen, por lo tanto es probable que tengamos más chance de interoperar con otras implementaciones si se utiliza este tipo de autenticación.

La gran contra que tiene este tipo de autenticación es la pobre administración de estas claves. Además que si hacemos túneles VPN con otras empresas u otros administradores, indefectiblemente tenemos que utilizar las mismas claves. Es un tipo de autenticación válido, pero debería ser el último recurso a utilizar.

8.2. Firmas digitales RSA

Las firmas digitales son firmas asincrónicas (eso quiere decir que la firma digital se divide en dos: una parte privada y otra pública), esta es una forma sencilla de autenticar entre dos Linux con FreeS/WAN y muy segura. Lo bueno de esta forma de autenticación es que cada nodo tiene su propia firma, esto hace que si en algún nodo se comprometió la parte privada de la firma, entonces se puede dar de baja ese nodo únicamente y el resto de los nodos de la red pueden seguir encriptando sin temor. El tamaño de las firmas por defecto es bastante generoso (2192 bits) aunque se puede agrandar si uno quiere.

El funcionamiento de estas firmas es simple, cada nodo arma su par de claves e intercambian la parte pública de la misma (o las publica en un DNS). Entonces cuando uno de los nodos recibe un pedido de conexión del otro, éste verificará si el nodo remoto es quien dice ser revisando el mensaje enviado. Este mensaje viene firmado digitalmente por el nodo que inicio la conexión, y el nodo receptor deberá determinar si esta firma es válida utilizando la clave pública del nodo remoto.

El proceso seguramente es mucho más complejo, pero realmente no lo conozco al detalle, recomiendo la lectura de documentación más detallada sobre esto o los mismo fuentes (?)).

El mantenimiento de estas claves puede llegar a ser algo tedioso (si hay muchos nodos), pero la facilidad de implementación vale la pena hacer el esfuerzo.

8.3. Certificados X509

Los certificados X509, a mi parecer, son la mejor opción a utilizar. La idea con los certificados es tener una CA (Certificate Authority - Autoridad Certificante) que actua de “validadora” de certificados, si un certificado (que provenga de cualquier lado en internet) esta firmado por la CA en la que nosotros confiamos, entonces asumimos que este certificado es un certificado válido, vale el comentario de que el nodo que esta recibiendo la conexión (generalmente *el concentrador*) no tiene la clave pública del nodo que inició la comunicación, éste solo revisará si el certificado esta firmado con la CA en la que se confía. Por otro lado, y antes de asumir la validez de un certificado firmado, se revisa lo que se llama la CRL (Certificate Revocation List - Lista de Certificados Revocados), que es un archivo donde están todos los certificados en los que *no* se confía más (y que han sido firmados en algún momento por nuestra CA), en esta lista van a parar todos los certificados que por alguna razón han sido comprometidos (alguien se robo una laptop o el nodo ha sido crackeado, o cualquier otra razón que resulte válida).

El único inconveniente (?) es que dar el primer paso con una CA y el **openssl** puede llegar a ser tedioso (pero no después de leer esta documentación por supuesto).

9. Configuración

FreeS/WAN posee un archivo principal de configuración y es el `ipsec.conf`, puede estar en `/etc` o en `/etc/freeswan` (depende la distribución de Linux). En este archivo se configura todo, hay otros archivos de configuración como el `ipsec.secrets` donde se pone la información de claves, firmas de RSA, y/o ubicación

de las claves de los certificados. Por último hay un directorio `/etc/ipsec.d` que se utiliza generalmente al trabajar con el parche de autenticación X509.

El `ipsec.conf` tiene dos tipos de secciones, la sección de “config” (configuración) y la sección de “conn” (conexiones). En este momento la única sección de configuración que se acepta en FreeS/WAN es la sección de config “setup”.

La sección “config setup” tiene toda la información que el software necesita al inicializarse.

Este es un ejemplo de esta sección:

```
config setup
    interfaces=%defaultroute
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
    uniqueids=yes
```

El valor más importante en este ejemplo es el del parámetro `interfaces`, el valor especial `%defaultroute` significa que la interface de red que **pluto** va a utilizar para establecer túneles cifrados, es la que utiliza la ruta por defecto (la que sale a internet generalmente). En el caso de que los túneles los estemos armando sobre otra red que no sea internet, posiblemente vaya a ser necesario cambiar este valor.

Los parámetros de debug, `klipsdebug` y `plutodebug`, se utilizan cuando hay problemas más complejos, los mensajes normales, que dicho sea de paso se graban en los logs, dan suficiente información para determinar dónde están los problemas comunes.

9.1. Secciones “conn”

Las secciones “conn” se utilizan para decirle a **pluto** que tipo de túneles se van a establecer o aceptar. Para empezar a explicar cómo configurar estas secciones voy a utilizar un ejemplo de *concentrador*. Generalmente se utilizan topologías de tipo estrella en las VPNs, todos los nodos (sucursales) quieren conectarse a una red LAN central (casa matriz) donde están los servidores centrales. El concentrador es el servidor que concentra todas las conexiones de los diferentes nodos VPN de la red.

Es importante destacar el hecho de que estas conexiones son indicadores de túneles que se van a *establecer* o *aceptar*, no hay una forma de hacer que FreeS/WAN *no* acepte conexiones, siempre llega a evaluar el hecho de establecer o no una conexión, pero no nos vamos a preocupar de esto ahora.

Bien, en lo que respecta a secciones “conn” hay una que es especial y es la sección `conn %default`, esta sección va a tener los valores por defecto de todas las secciones “conn”. Luego en cada sección “conn” se podrán sobrescribir los valores, pero en caso que se omitan se tomarán los escritos en esta sección.

Este es un ejemplo de sección `conn %default`

```
conn %default
```

```
keyingtries=0
authby=rsasig
```

El parámetro `keyingtries` determina cuantas veces **pluto** va a intentar para establecer un túnel, el valor cero indica que siga intentando eternamente. El parámetro `authby` determina la forma en que se va a estar haciendo la autentificación. El valor `secret` es cuando se utiliza pre-shared-keys (claves compartidas pre acordadas) y se utiliza el valor `rsasig` cuando se va a autenticar con una firma digital RSA (cuando se utilizan certificados X509 también se usa este tipo de valor).

Dependiendo el tipo de autentificación y los tipos de túneles que se vayan a utilizar van a existir una sección “conn” por nodo de VPN o no. Por ahora vamos a ver ejemplos que solo permiten un túnel por sección “conn” (que de hecho es lo más usual).

```
conn nodo-sop-tecnico
  left=%defaulttroute
  leftsubnet=192.168.1.0/24
  leftrsasigkey=0sAQO1wwYJq.....
  right=%any
  rightid=@amigo.xtech
  rightsubnet=192.168.33.0/24
  rightrsasigkey=0sAQNt1jXTSQ.....
  auto=add
```

Este ejemplo se lee de la siguiente manera (vamos a utilizar el valor `left` para el concentrador y el `right` para el nodo de soporte técnico)

Dado que el concentrador (`left`) esta conectado directamente a internet, se utiliza el valor `%defaulttroute` que le indica a **pluto** que tome el número de IP asignado a la interface que tiene la ruta por defecto (por ejemplo la interface `ppp0`, si es que nos conectamos a internet con un modem o línea ADSL). Si este número fuera un IP estático y fijo, entonces podríamos poner directamente el número como valor del campo.

El campo `leftsubnet` indica qué número de IP tiene la red local que esta del lado izquierdo del túnel, o sea, el número de IP de la red LAN del concentrador (donde están los servidores centrales). El parámetro `leftrsasigkey` indica la parte pública de la clave RSA del concentrador, ya vamos a ver esto más en detalle cuando veamos el tema de autentificación.

Con respecto al lado derecho (`right`), vemos que el parámetro `right`, que indica el número de IP que tiene el lado derecho del túnel, tiene un valor especial: `%any`, esto le indica a **pluto** que el lado derecho de este túnel va a poder ser cualquier número de IP (o sea un IP dinámico).

Important: ATENCIÓN!:

Cuando se utiliza este valor (`%any`) y la autentificación es por firmas digitales, es necesario agregar el campo `rightid=@<alguna_cadena>` (en el concentrador y en el nodo), si bien no estoy seguro cuál es la razón de esto, me imagino que es porque **pluto** (en el concentrador) no puede identificar qué “conn” utilizar cuando llega el requerimiento de conexión del nodo.

Al igual que el lado izquierdo, el derecho también tiene un subnet y una clave pública de RSA.

Por último el parámetro `auto=add` indica que este túnel deberá agregarse a los túneles disponibles, pero **pluto** no hará nada con él al momento de inicializar todo el servicio (el valor de este campo puede contener `start` que indica que al inicializar el servicio, **pluto** intente establecer el túnel), y otros valores que no viene al caso explicarlos.

Este ejemplo es un ejemplo de túnel “LAN to LAN”, si no estuvieran los parámetros de `[right|left]subnet=` entonces sería un túnel “host to host” (por supuesto las combinaciones son también válidas: “host to LAN” y “LAN to host”, sacando y poniendo el parámetro `...subnet=` que corresponda).

```
conn al-centro
    left=centro.xtech.com.ar
    leftsubnet=192.168.1.0/24
    leftrsasigkey=0sAQ0lwwYJq.....
    right=%defaultroute
    rightid=@amigo.xtech
    rightsubnet=192.168.33.0/24
    rightrsasigkey=0sAQnt1jXTSQ.....
    auto=start
```

Esta sección “conn” es la que se utilizaría en el nodo de soporte técnico (la otra punta del túnel que acabamos de ejemplificar). Los campos que cambiaron su valor son `left=`, `right=` y `auto=`. Como podras ver, `left` sigue siendo el concentrador y `right` el nodo, pero los valores son diferentes. En `left=` se colocó el valor `central.xtech.com.ar` que es justamente el nombre al que responde el número de IP del nodo central. Si tuvieramos un número de IP fijo en central, podríamos utilizarlo, pero dado que este punto de la VPN también recibe un IP dinámico, no nos queda otra alternativa que utilizar un nombre y algún servicio de DNS dinámico. Para el caso de `right=` se utiliza el valor especial `%defaultroute` que indica exactamente lo mismo que explique más arriba para el caso del concentrador. En el caso del parámetro `auto=`, el valor `start` indica que al momento de inicializar el servicio, **pluto** intentará establecer el túnel.

9.2. Algunas notas sobre los ejemplos

Dado que en este ejemplo se están utilizando números de IP dinámicos en ambas puntas del túnel, un servicio de DNS dinámico es indispensable. De alguna manera los nodos tienen que saber contra quien deben establecer los túneles. Esta fuera del alcance de este documento explicar como utilizar los servicios de DNS dinámicos, pero en mi experiencia la mejor aplicación para hacer esto es DHIS o servicios como el que da <http://www.no-ip.com>

Por otro lado, es importante destacar que se dejaron de lado algunos parámetros de las secciones “conn” para hacer más sencilla la explicación inicial. Más adelante se entra en detalle y se explican las diferentes alternativas.

9.3. Configurando los tipos de autenticaciones

Cada un de los tipos de autenticación tiene su propia forma de configuración. En esta sección se detalla cada una de ellas.

9.3.1. Pre-shared-keys

Para configurar este tipo de autenticación es necesario poner en un solo lugar la clave única compartida. Este lugar es el archivo `/etc/ipsec.secrets`. En el caso donde los números de IP son fijos es posible poner una clave por conexión.

Para utilizar este tipo de autenticación se escribe lo siguiente en el archivo `/etc/ipsec.secrets`:

```
<direcciones_de_ip_que_intervienen> : PSK "clave"
```

Dado el dinamismo de los IPs, no hay mucha forma de saber el número de IP que va a tener cada punta de la red, con lo cual la única dirección de IP que vamos a poner es solamente la nuestra actual⁵

Por otro lado, vi en algún que otro documento que se pueden utilizar nombres acá, pero yo no lo he probado aún.

9.3.2. Firmas RSA digitales

Las firmas digitales no son tan difíciles de configurar y son más administrables. Para configurar este tipo de autenticación se debe ejecutar el siguiente comando:

```
# ipsec newhostkey --output /etc/ipsec.secrets
```

Este comando generará un archivo nuevo con un contenido parecido al siguiente:

```
: RSA {
# RSA 2192 bits fw.ny4487.com.ar Tue Jun 24 01:14:02 2003
# for signatures only, UNSAFE FOR ENCRYPTION
#pubkey=0sAQPY+BA6k+J7emr7+.....
#IN KEY 0x4200 4 1 AQPY+BA6k+J7em.....
Modulus: 0xd8f8103a93e27b7a6afbfa6c6b.....
PublicExponent: 0x03
PrivateExponent: 0x24295809c35069e9bc7f546.....
Prime1: 0xed2ef593c6.....
Prime2: 0xea2e8fe4f956.....
Exponent1: 0x9elf4e.....
Exponent2: 0x9c1f0a98a6.....
Coefficient: 0x497825b73814334.....
}
# do not change the indenting of that "}"
```

Si bien las líneas son largas y aleatorios sus números, hay una línea que es de interés, es la línea (comentada) que contiene la parte pública de la clave (la que comienza con `#pubkey=0sAQPY.....`), esta línea deberá ser copiada (tal cual esta, pero sin el `#pubkey`) al archivo `/etc/ipsec.conf` en el parámetro `[left|right]rsasigkey=` (en el que corresponda, por supuesto).

Se deberá hacer esto mismo en el otro nodo y en ambos `/etc/ipsec.conf` deberán contener la parte pública de la clave de los dos nodos.

9.3.3. Certificados X509

Este método de autenticación es el más complicado de armar, pero el más sencillo de mantener en el tiempo.

Para configurar este método hay que hacer los siguientes pasos:

1. Armar una CA (Certificate Authority)
2. Generar un certificado para cada nodo
3. Firmar los certificados con la CA creada en el paso 1.

1. Armar una CA

Para armar una CA es necesario utilizar el comando **openssl** para generar un certificado firmado por uno mismo (en inglés: self-signed). Este comando hace justamente esto, generando por un lado una clave privada `cakey.pem` y por el otro el certificado público firmado `cacert.pem`.

```
# openssl req -x509 -newkey rsa:2048 -keyout cakey.pem -out cacert.pem
```

El comando pregunta una serie de cosas, el único campo importante (bah!, con el que hay que tener especial cuidado) es el “Organization Name” o “Compañía”. El mismo dato que se ingresa acá hay que ingresarlo a la hora de crear un requerimiento de certificado (esto es para hacer más fácil el procedimiento).

El archivo `cakey.pem` hay que moverlo al directorio `/etc/ipsec.d/private` y el `cacert.pem` al directorio `/etc/ipsec.d/cacerts`.

1. Generar un certificado por nodo

Una vez que esta armado el certificado de la CA se pueden empezar a armar los certificados de los nodos. En realidad lo que se hace es generar un “requerimiento de certificado”, esto tiene una validez por defecto de 30 días. Este requerimiento de certificado es lo que se hace firmar por la CA y en el proceso de creación del requerimiento se crea la parte privada del certificado.

Este comando crea el requerimiento de certificado y su clave privada correspondiente.

```
# openssl req -newkey rsa:1024 -keyout hostAliciakey.pem  
-out hostAliciareq.pem
```

Este comando formula una serie de preguntas, la importante, para que mantener las cosas simples, es la pregunta de “Organization Name” (Nombre de la Organización), acá es importante ponerle lo mismo que se ha puesto al armar el certificado de la CA.

1. Firmar los certificados

Antes de enviar este certificado al nodo correspondiente vamos a firmarlo con la CA que se creó anteriormente. Esta es la parte más complicada (?!). Antes de firmar los requerimientos de certificado con la CA tenemos que hacer algunos ajustes.

Se copia el archivo `openssl.cnf` al directorio `/etc/ipsec.d`. Hay que editar este archivo y cambiar la variable “dir” bajo la sección “[CA_default]”, se debe reemplazar la cadena “./demoCA” por “./” únicamente; y también hay que cambiar la variable “certificate” y ponerle el valor “\$dir/cacerts/cacert.pem”. Por último se deberá crear un archivo vacío llamado `index.txt` en el directorio `/etc/ipsec.d` y crear otro archivo, llamado `serial` con el número 01 en su interior⁶.

Con estos ajustes estamos listos para firmar los requerimientos de certificados. El siguiente, es el comando que se utiliza.

```
# openssl ca -in hostAlicereq.pem -out hostAlicecert.pem -notext
-config ./openssl.cnf
```

Una vez que los requerimientos de certificados están firmados (por lo tanto ya se los considera “un certificado”) éstos ya están listos para ser enviados a los nodos de nuestra VPN. Si todo el proceso se realizó en la misma máquina, entonces hay que tener en cuenta que habrá que enviar con el certificado firmado la clave privada del mismo. En el caso de que se haya generado el requerimiento de certificado en el nodo y luego se lo transfirió a la CA para ser firmado, lo único que hay que enviar al nodo es el certificado firmado (ya que la clave privada del certificado está en el nodo). Por supuesto, la recomendación es que el requerimiento de certificado se realice en el nodo y se transfiera la parte pública al equipo donde esta la CA.

Como último comentario sobre este tipo de autenticación debo destacar que no es necesario que los certificados firmados residan en otro equipo que no sea el nodo al cual pertenece este certificado. El concentrador solamente va a aceptar conexiones desde nodos que tengan certificados firmados por la CA. Lo único que sí deben tener todos los nodos es la parte pública del certificado de la CA (el `cacert.pem`), por razones que no vienen al caso detallar.

10. VPN ejemplo

Para bajar todo lo explicado a tierra vamos a ejemplificar una VPN de nodos LAN-to-LAN con IPs dinámicos en todas las puntas (así es más divertido). Esta VPN va a autenticar con certificados x509 y va a hacer uso de un sistema de DNSs dinámicos para fijar el nombre del concentrador (que es el único nombre importante). El nodo central de la VPN está en Córdoba y sus nodos remotos residen en Bs.As., Rosario, Tucumán y Salta.

10.1. Configuración del concentrador y un nodo paso por paso

Lo primero que se necesita es armar los certificados de la CA, y luego el del nodo concentrador. Más tarde, en cada uno de los nodos se generarán los requerimientos de certificado para ser firmados por la CA. La CA va a estar en el mismo equipo que el concentrador, o sea en el equipo de Córdoba.

Siguiendo lo explicado en la sección de armado de CA, vamos a ejecutar la siguiente seguidilla de comandos

```
(me posiciono en el directorio correspondiente)
# cd /etc/ipsec.d
(armo la CA)
# openssl req -x509 -newkey rsa:2048 -keyout
```

```

    private/cakey.pem -out cacerts/cacert.pem
(armando el certificado del concentrador)
# openssl req -newkey rsa:1024 -keyout private/CbaKeyConcentrador.pem
    -out CbaReq.pem
(responder todas las preguntas y poner el mismo 'Organization'
    que se puso al hacer la CA)
# cp /usr/share/ssl/openssl.cnf .
# vi openssl.cnf
(cambio las cosas que tengo que cambiar dentro del archivo
    y firmo el certificado)
# openssl ca -in CbaReq.pem -out CertCbaSigned.pem -notext
    -config ./openssl.cnf

```

-- listo el pollo en el concentrador --

Ahora vamos al nodo y ejecutamos los siguientes comandos

```

# cd /etc/ipsec.d
# openssl req -newkey rsa:1024 -keyout private/BsAsKey.pem
    -out BsAsReq.pem (responder todas las preguntas)
(transfiero el archivo BsAsRep.pem al concentrador)
# scp BsAsReq.pem dcoletti@cordoba.no-ip.org:/tmp

```

en el concentrador ejecuto

```

# cp /tmp/BsAsReq.pem /etc/ipsec.d
# cd /etc/ipsec.d
(firmo el certificado)
# openssl ca -in BsAsReq.pem -out CertBsAsSigned.pem -notext
    -config ./openssl.cnf
(copio el certificado firmado al nodo de Buenos Aires)
# scp CertBsAsSigned.pem root@<nro. de ip de BA>:/tmp
(y ya que estoy tambien envio la publica de la CA)
# scp cacerts/cacert.pem root@<nro. de ip de BA>:/tmp

```

Por último, agrego la nueva conn al FreeS/WAN del concentrador, edito el archivo `/etc/ipsec.conf` y agrego lo siguiente

```

conn desde-bsas
    left=%defaultroute
    leftsubnet=192.168.0.0/24
    leftcert=CertCbaSigned.pem
    right=%any
    rightsubnet=192.168.1.0/24
    auto=add

```

Dado que por cada nodo se va agregar una conn nueva, y en todas las secciones conn hay que poner los mismos valores en `left`, `leftsubnet` y `leftcert`, directamente podemos agregar estos valores en la seccion `%default`, y no poner estas lineas en todas las secciones conn de cada nodo.

De más esta decir que el proceso de crear nuevos certificados y firmarlos por la CA se debe repetir por cada nodo de la red.

(sigo ejecutando en el nodo Buenos Aires, ubico los certificados donde deben estar)

```
# mv /tmp/CertBsAsSigned.pem /etc/ipsec.d
# mv /tmp/cacert.pem /etc/ipsec.d/cacerts
```

(edito el archivo /etc/ipsec.conf y agrego lo siguiente)

```
conn hacia-cba
    right=%defaultroute
    rightsubnet=192.168.0.0/24
    rightcert=CertBsAsSigned.pem
    left=cordoba.no-ip.org
    leftsubnet=192.168.1.0/24
    leftid="/C=AR/ST=Cordoba/O=Xtech S.A./CN=cba.xtech.com.ar"
    auto=auto
```

Como se puede ver en la sección `conn hacia-cba` se utilizó el `leftid` para indicar quién es Córdoba, esto se utiliza para no tener que tener el certificado del concentrador en cada nodo. Al momento de autenticarse, las dos puntas intercambian certificados, pero cada nodo tiene que poder determinar que los certificados son válidos, por lo tanto es importante que el `leftid` tenga el valor del campo `Subject`⁷ del certificado de Córdoba.

Por otro lado en el campo `left=`, que es donde va el número de IP del lado izquierdo de la conexión, tiene el nombre del equipo, este nombre es un nombre dinámico (por eso tiene ese dominio).

Por último es importante que se note que en ambas secciones “conn” (la del concentrador y la del nodo) los campos `left*` y `right*` se mantuvieron con la misma información (excepto cuando usan `%defaultroute` ya que este valor especial indica que tome el número de IP de la interface por donde esta establecida la ruta por defecto).

El problema más escabroso por resolver es el qué hacer cuando las conexiones a internet se caen. Hay que tener en cuenta que IPSec no tiene forma de determinar que un túnel no funciona más, por lo tanto es importante hacer algún tipo de programa que vaya determinando que el túnel sigue funcionando y en el caso de que no funcione más, que reinicie todo. También habrá que armar un programa para monitorear la conexión a internet del concentrador. Estos programas generalmente son shell scripts.

11. Uso de las herramientas de FreeS/WAN

FreeS/WAN provee un comando **ipsec** que se utiliza para indicarle a **pluto** qué hacer con cada “conn”, incluso para indicarle nuevas configuraciones sin tener que reiniciar el servicio de ipsec. Dado que hay bastante información escrita sobre el comando (a través de las páginas del manual en línea), no voy a dar detalles sobre el comando, simplemente voy a describir algunas de las formas del comando más utilizadas (o por lo menos las que más utilizo yo).

--- mostrar las rutas encriptadas ya establecidas, acá van a aparecer los túneles una vez que estos se hayan establecido ---

```
# ipsec eroute
```

--- agrega una "conn" a las conexiones disponibles al demonio **pluto** que se esta ejecutando, este comando se utiliza cuando se agregó una sección "conn" al `/etc/ipsec.conf` y no se desea -- o no se puede -- bajar el servicio de ipsec ---

```
# ipsec auto --add <conn>
```

-- se le indica a **pluto** que arme el túnel "conn" --

```
# ipsec auto --up <conn>
```

-- se le indica a **pluto** que elimine el túnel "conn" de los túneles disponibles en memoria, esto da de baja el túnel si éste estuviera establecido --

```
# ipsec auto --delete <conn>
```

--- determina el estado de las conexiones, de todas las conexiones, las establecidas y las declaradas ---

```
# ipsec whack --status
```

--- reinicia todo el servicio de ipsec ---

```
# ipsec setup restart
```

12. Ejemplos de Configuración

En esta última sección voy a mostrar cómo configurar VPNs en diferentes escenarios utilizando FreeS/WAN. Con suerte estos escenarios cubren los casos más utilizados.

12.1. LAN wireless

Si bien este no es un escenario tan común hoy en día, con la reducción de costos de las placas y nodos wireless, las redes locales wireless van a empezar a ser más numerosas. Por otro lado, la tecnología de cifrado que usan las redes wireless es muy mala, con lo cual poner otro tipo de cifrado es sumamente importante.

Hay que recordar que es muy fácil ver los datos (o *sniffear*) en una LAN wireless, solo basta con tener una placa wireless, estar relativamente cerca de la red LAN (en el edificio de al lado, por ejemplo) y ponerse a escuchar.

Armar una VPN dentro de una red LAN es bastante sencillo, la idea es armar tuneles Host to LAN, los Hosts seran las PCs de la red wireless y la LAN es el comodin 0.0.0.0/0 o sea todas las redes. De esta forma nos aseguramos que para llegar a cualquier otra dirección TCP/IP el equipo PC va a salir por el túnel.

La configuración de la PC cliente será la siguiente (siendo `right` el concentrador y `left` el equipo PC):

```
conn algateway
    left=%defaultroute
    leftcert=MiCertificado.pem
```

```
right=192.168.1.1
rightsubnet=0.0.0.0/0
auto=start
```

Del lado del concentrador la configuración recomendada sería la siguiente

```
conn todoslosnodos
left=%any
leftcert=ConcentradorCert.pem
right=192.168.1.1
rightsubnet=0.0.0.0/0
auto=add
```

Acá es donde se pone divertido. Ya habiendo leído el documento sabemos que podríamos utilizar otros tipos de autenticación, como por ejemplo, *Pre Shared Keys* y firmas asimétricas RSA, las contras que tienen estos métodos de autenticación es justamente la administración. Si quisieramos utilizar PSK tendríamos que poner en el archivo `ipsec.secrets` una línea con una clave diferente por cada nodo (porque no pensaban darle la misma clave a todos los nodos, no?), y si quisieramos utilizar RSA, entonces tendríamos que tener una sección `conn` por cada nodo (lo cual hace sumamente extenso el archivo de configuración ---cosa que particularmente me molesta---).

13. Referencias

- Sitio de FreeS/WAN (<http://www.freeswan.org>)
- Sitio donde generan SuperFreeS/WAN (<http://www.freeswan.ca>)
- RFCs (http://www.freeswan.org/freeswan_trees/freeswan-1.99/doc/rfc.html#RFCs.tar.gz)
- <http://www.dhis.org> (Sistema para manejar DNS dinámico)
- Sitio original donde está este documento (<http://www.colettis.com.ar/~daniel>)

Notes

1. Cuando hablo de tráfico cifrado, me refiero a tráfico encriptado, es que simplemente estoy usando la palabra correcta (encriptar no existe en español)
2. No es solo esto, pero este es uno de los ejemplos
3. Todos los ejemplos van a ser utilizando Linux en todas las puntas de la VPN
4. Que obtengan un número de IP público y válido al conectarse
5. obviamente vamos a tener que armar algún script que cambie este valor cuando la conexión a internet se caiga o cambie la dirección por algún motivo
6. Si querés entender por qué se hacen estos cambios, probá firmar el certificado sin hacerlos (y mirá los errores con los que te encontrás) o lee la documentación de openssl
7. Este valor se puede obtener utilizando el comando “# openssl x509 -in CertCbaSigned.pem -noout -subject”