ANEXO 1: VPN CA

Índice general

1. Nivel de red: VPN con OpenVPN. Autoridad Certificadora X.509 **1**

Capítulo 1

Nivel de red: VPN con OpenVPN. Autoridad Certificadora X.509

Contents

1.1. DKLAB1 y Autoridad Certificadora 4

1.1.1. Instalación de OpenVPN y OpenSSL 4

1.1.2. Creación de la CA y los certificados X.509 del servicio VPN . . 4

Configuración de Easy-RSA/OpenSSL 4

Creación del material criptográfico de la CA 12

Creación del par de claves, petición de certificado y certificado para openvpn 13

Certificado para el rol de OpenVPN tls-server (dklab1); parámetros DH 13

Certificados para el rol de OpenVPN tls-client (dklab2): 14

Creación de CRL (Certificates Revocation List) 15

Verificación 16

Verificar el estado de revocado de un certificado en la lista revocación 17

TLS Protocols Test Suite 17

Anotaciones 19
1.1.3. Distribución de los recursos criptográficos creados 19
1.1.4. Creación de la cuenta no privilegiada openvpn 20
1.1.5. Configuración de OpenVPN en modo tls-server: /etc/openvpn/casafx-tls-server.conf 20
1.1.6. Integración en el sistema de configuración de red de Debian . . 27
1.1.7. Lanzamiento del servicio 28
1.2. DKLAB2 29
1.2.1. Instalación de OpenVPN 29
1.2.2. Distribución de los recursos criptográficos desde la CA 29
1.2.3. Creación de la cuenta no privilegiada openvpn 30
1.2.4. Configuración de OpenVPN en modo tls-client: /etc/openvpn/casafx-tls-client.conf 30
1.2.5. Integración en el sistema de configuración de red de Debian . . 34
1.2.6. Lanzamiento del servicio 35
1.3. Tests 35

Una idea motiva principalmente la existencia de este capítulo: si vamos a desplegar una infraestructura de servicios compleja, costosa, y vamos a depender de ella, hagamos que esté siempre disponible.

Así, si una computadora se encontrase lejos geográficamente por una eventualidad, podría establecer una red privada virtual protegida con alguna máquina de nuestra organización (bien utilizando un segmento de red especial y las consiguientes rutas, o bien en el mismo segmento de red etc). El resultado es que la computadora alejada de la organización, vuelve a tener disponibles toda la infraestructura de servicios (de la que depende) como si estuviera allí.

Puesto que nuestro despliegue contempla sólo dos máquinas, no podemos demostrar el despliegue de los extremos de la VPN utilizando otras máquinas dedicadas[1]. Por tanto dklab1 y dklab2, que ya se comunican entre sí a través de la red IP v4 192.168.1.0/24 según se configuró en el capítulo anterior, crean encima una nueva red en la que dklab1 será un extremo de la VPN y dklab2 el otro, siendo el segmento IP v4 de VPN formado el 10.168.1.0/24. Será sobre este segmento de red donde se configurarán posteriormente todos los servicios.

Existen otros problemos que afectan al nivel de red y que podemos intentar solucionar de forma ventajosa[2]. No podemos centrarnos en todos ellos, así que lo hacemos en el que nos ha parecido más importante.

Por último, y como veremos a continuación, nuestra solución de VPN utiliza el estándar TLS para proveer mecanismos de seguridad, y TLS depende de certificados X.509. Así, en este capítulo se creará para nuestra organización una Autoridad Certificadora (CA), que será responsable de cualquier certificado X.509 que necesitemos para el servicio actual o cualesquiera otros servicios que despleguemos más tarde y lo necesiten también.

[1]Efectivamente, no tiene sentido replicar los servicios en dklab1 y dklab2, a la vez que se hace depender a las computadoras de dklab1 para acceder a la red: si dklab1 cae, no podrían beneficiarse de la replicación en dklab2. En un despliegue en producción, el/los extremo/s de la VPN deberían estar en otras máquinas.

[2]Por ejemplo, se nos ocurre la posibilidad de tener varios servidores DHCP que cooperen entre sí y con LDAP para la configuración de red automática del resto de computadoras de nuestra hipotética organización.

1.1. DKLAB1 y Autoridad Certificadora

1.1.1. Instalación de OpenVPN y OpenSSL

```
apt-get install openvpn openssl
```

1.1.2. Creación de la CA y los certificados X.509 del servicio VPN

Una Autoridad Certificadora (CA) es, al fin y al cabo, un par de llaves privada y pública, ésta última con su correspondiente armadura X.509 y firma. Adicionalmente, CA se refiere también a la estructura de directorios y ficheros de configuración que nos ayudan a usarla, es decir, a crear o como mínimo firmar otros certificados.

Pues bien, respecto a este segundo significado, nuestra CA será un conjunto de scripts y estructura de directorios llamado "Easy-RSA", proveído por el paquete openvpn, y que usa a OpenSSL como backend para generar los recursos criptográficos. Nuestra hoja de ruta es clara:

1. Primero hemos de (comprender y) adaptar la configuración de Easy-RSA a nuestros intereses.
2. Entonces crearemos el material criptográfico de la CA que hemos mencionado.
3. Por último lo utilizaremos para crear y firmar los certificados que necesita openvpn en dklab1 y dklab2.

Configuración de Easy-RSA/OpenSSL

Inicialmente hemos de copiar el directorio de Easy-RAS a una ubicación y permisos convenientes:

```
cp -R /usr/share/doc/openvpn/examples/easy-rsa /usr/local/lib/
cd /usr/local/lib/easy-rsa/2.0/
chmod go-x -R .
less README.gz
```

El comando openssl dispone de una variedad de subcomandos ("openssl ca", "openssl req", "openssl verify"...) cuyo comportamiento se configura en ficheros openssl.cnf[3], de sintaxis tipo "INI". Existe uno "system-wide" en /etc/ssl/openssl.cnf, pero nosotros usaremos uno propio, en concreto modificaremos convenientemente el que viene con Easy-RSA.

[3] http://www.phildev.net/ssl/opensslconf.html

Nuestra configuración permite tener tres tipos de certificados: el de la CA, el de tls-servers y el de tls-clients. Las diferencias entre ellos se establecen según los atributos y extensiones que se añaden a su armadura X.509, que constituye en sí misma una metainformación susceptible de ser utilizada por mecanismos de autenticación y autorización[4]. De los atributos nos interesa:

- Subject. Está formado por un DN[5] X.500, es decir una concatenación de pares identificador=valor separados por comas definido en los estándares X.500 para identificar nodos en sistemas de directorio, de forma que cada par constituya un RDN[6] (relative DN). Uno de los RDN, llamado CN (de "Common Name") se utiliza como identificador del certificado en sí, y puede dividirse en varias partes para definir esa identidad desde varios puntos de vista, por ejemplo CN=<nombre>/emailAddress=<dirección de correo>.

- Issuer. Con la misma estructura que el anterior, define no el certificado en sí sino a la CA que lo firmó (luego el certificado de esa CA tiene como contenido del atributo Subject a lo que en este certificado es contenido del atributo Issuer).

De las extensiones (informalmente, otros atributos), nos interesan:

- X509v3 Subject Alternative Name: permite añadir (o reemplazar) identidades en el atributo Subject, tal como direcciones de correo electrónico, nombres de dominio DNS, etc.

- X509v3 Basic Constraints: identifica si Subject es una CA, es decir si puede firmar y su profundidad de niveles de certificación.

- X509v3 Key Usage: indica el propósito de la clave pública, por ejemplo "Digital Signature"(verificación de firmas en servicios de autenticación, integridad de datos ...), "Key Encipherment"(cifrado de claves simétricas de sesión ...) etc.

- Netscape Cert Type: nos servirá para indicar que el certificado es para el rol de tls-server y evitar ciertos ataques man-in-the-middle. Efectivamente openvpn estará pendiente de que un tls-client no intente autenticarse con un certificado de ese tipo.

Finalmente, ya podemos definir la estrategia tras nuestro fichero de configuración openssl.cnf (estas notas serán, a su vez, la base para orientarnos en posteriores usos de nuestra CA, al desplegar otros servicios que necesiten certificados X.509):

- La llave privada del par de la CA firma todos los certificados. Por tanto, el certificado raiz de nuestra CA es autofirmado y verifica tanto a sí mismo como a todos los demás. Debiera haber una copia del certificado en todas las máquinas involucradas.

[4]PKI, private key infraestructure.
http://tools.ietf.org/html/rfc5280.
http://ospkibook.sourceforge.net/docs/OSPKI-2.4.7/OSPKI-html/ospki-book.htm
[5]De "Distinguished Name'én el sistema X.500,DN
[6]De "Relative Distinguished Name'én el sistema X.500, RDN

- El atributo Subject se compondrá con los valores de las variables en el fichero "vars". Efectivamente, podemos instruir a openssl para recoger valores como variables de entorno gracias a la sintaxis $ENV::<nombrevariable>. Por tanto, antes de ejecutar los scripts de Easy-RSA, exportaremos esas variables para que estén disponibles. El caso del RDN CN es especial, pues se compone como CN=<nombre>/emailAddress=<dirección de correo>, donde <nombre>será el argumento pasado al script de Easy-RSA, y ya la dirección de correo será la definida en una variable de entorno, KEY_EMAIL, si bien:
  - Al crear la CA, le pasamos al script un valor especial de la variable KEY_EMAIL: env KEY_EMAIL=camaster@casafx.dyndns.org ./ca-build
  - Al crear el resto de certificados, no le pasamos al script un nuevo valor de KEY_EMAIL, luego recoge el exportado desde "vars", KEY_EMAIL=vpnmaster@casafx
- Respecto a las extensiones, previamente aclararemos que un script (build-ca) crea la CA, otro (build-key-server) los certificados de tls-server y otro (build-key o build-key-pkcs12) los de tls-client. Pues bien, congruentemente, una sección del openssl.cnf (CA_default) define las extensiones y atributos para la CA, otra (server) para los certificados de tls-server y otra (usr_cert) para los de tls-client, tal que cada script use la sección pertinente en cada caso. Entonces:
  - La extensión "X509v3 Key Usage", tendrá asignado "Digital Signature" en todos los certificados. Adicionalmente, tendrá asignado el permiso para "Key Encipherment" pero sólo a los certificados para tls-server.
  - La extensión "X509v3 Basic Constraints"[7] será FALSE excepto para el certificado de la CA.
  - La extensión "Netscape Cert Type" con valor "SSL Server" se añadirá a los certificados para tls-server.
- Puesto que dklab1 y dklab2 tienen el certificado de la CA, ambos pueden verificar los certificados presentados por la otra parte y así la identidad del CN. Adicionalmente, los clientes suelen comprobar para el servidor algún tipo de coincidencia entre su nombre de dominio y su certificado:
  - El nombre de dominio usado en la URL debe estar en el CN del atributo Subject, o al menos ser subdominio de un CN utilizando "wildcard-matching" (por ejemplo, algo como que <subdominio>.<dominio>.<tld>casaría con *.<dominio>.<tld>). O, finalmente, casa en la extensión Subject Alternative Name.
  - Anotación: puede ocurrir tanto que el cliente no inspeccione la coincidencia en el CN si existe el atributo Subject Alternative Name, como que existen clientes que no implementan aún la inspección de Subject Alternative Name. Por tanto, cuando se usa Subject Alternative Name y el CN es un nombre de dominio, puede ser conveniente añadir el contenido del CN al Subject Alternative Name.

[7] http://www.openssl.org/docs/apps/x509v3_config.html#Basic_Constraints_

```
cp openssl.cnf ../openssl.cnf.orig
vim openssl.cnf # openssl usara' este .cnf en lugar
                # del "system-wide" /etc/ssl/openssl.cnf
```

```
####fx:
SAN="email:camaster@casafx.dyndns.org"
####endfx
HOME=...
```

A continuación viene la configuración para el subcomando "openssl ca" de administración de autoridades certificadoras. No modificaremos nada, en general se describe la estructura de directorios de la CA, su período de validez, y la variable "x509_extensions" apunta a la sección donde se especificarán las extensiones que se añadirán a los certificados firmados por esta CA[8], sirviendo de información de autorización PMI[9]:

```
...
[ca]
default_ca = CA_default

[CA_default]
...
x509_extensions = usr_cert
default_days    = 3650
...
```

Le sigue olítica en las peticiones de certificado. La CA atenderá las peticiones de certificado que cumplan determinadas reglas. Así, por un lado la sección [policy_match] se refiere a la política de acuerdo entre los RDN del DN asignado al campo "Subject" en la petición de certificado en comparación al certificado de la CA:

[8] A través de órdenes como: openssl ca -key ca.crt -in request.csr -out cert.crt, que sabrán lanzar nuestros scripts.

[9] Véase http://ospkibook.sourceforge.net/docs/OSPKI-2.4.7/OSPKI-html/pkix-concepts.htm#CERTIFICATE-USING-SYSTEMS-AND-PMI

```
...
[policy_match]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
commonName              = supplied
...
```

Por otro lado, [policy_anything] configura la política para la estructura del DN asignado al campo "Subject": RDN obligatorios para formarlo, etc

```
...
[policy_anything]
countryName             = optional
stateOrProvinceName     = optional
organizationName        = optional
commonName              = supplied
...
```

Configuración para el subcomando "openssl req" (creación de petición de certificado): longitud, secciones para valores por defecto en el DN, cómo serán los prompts para pedir otros valores. La sección para extensiones no es la misma que la que se definió en CA_default pues aquellas son extensiones para los certificados, y éstas lo son para las peticiones. Por ello allí se apuntaba a la sección "usr_cert", y sin embargo ahora apuntamos a la sección "v3_ca" (ambas aparecerán luego). Cuando hay una extensión declarada para la petición pero no para el certificado, se obbia (es decir, no se pasa al certificado) excepto si se especificase "copy_extensions = copy", cosa que no ocurre en esta configuración.

```
...
[req]
default_bits            = $ENV::KEY_SIZE
distinguished_name      = req_distinguished_name
x509_extensions = v3_ca
...
[ req_distinguished_name ]
...
emailAddress_default            = $ENV::KEY_EMAIL
commonName_max                  = 64
...
```

Por fin, las anunciadas extensiones que estarán presentes en los certificados que firme la CA, conformando su PMI:

```
...
[ usr_cert ]
basicConstraints=CA:FALSE
#(Las dos siguientes no vienen en el /etc/ssl/openssl.cnf por defecto)
keyUsage = digitalSignature
extendedKeyUsage=clientAuth
...
####fx:
#subjectAltName= ${ENV::SAN}
####endfx
...
```

Como se ha visto, no usamos "Subject Alternative Name" en certificados para tls-client.

La sección para extensiones tls-server sí añade esa extensión:

```
####fx:
[ server ]
basicConstraints=CA:FALSE
nsCertType                      = server
...
extendedKeyUsage=serverAuth
keyUsage = digitalSignature, keyEncipherment
####fx:
subjectAltName=${ENV::SAN}
####endfx
```

Para las extensiones que estarán presentes en las peticiones:

```
...
[ v3_req ]
...
basicConstraints = CA:FALSE
####fx:
subjectAltName=${ENV::SAN}
####endfx
...
```

Hay un tercer grupo de extensiones que definir: las extensiones que estarán presentes en el propio certificado de la CA (ca.crt). Establecemos "basicConstrains" a true. Obbiamos "subjectAltName".

```
[ v3_ca ]
...
basicConstraints = CA:true
####fx:
#subjectAltName=${ENV::SAN}
####endfx
```

```
cp openssl.cnf ../openssl.cnf.mod
vim vars
```

Vamos ya con las variables:

```
cp vars ../vars.orig
vim vars
```

```
...
#Nota: Podri'amos doblar el taman~o de clave de 1024 a 2048 asumiendo
#una penalizacio'n temporal:
export KEY_SIZE=1024
#Por defecto los certificados tienen 10 an~os de peri'odo de validez
export CA_EXPIRE=3650
export KEY_EXPIRE=3650
####fx:
#-export KEY_COUNTRY="US"
#-export KEY_PROVINCE="CA"
#-export KEY_CITY="SanFrancisco"
#-export KEY_ORG="Fort-Funston"
#-export KEY_EMAIL="me@myhost.mydomain"
export KEY_COUNTRY="ES"
export KEY_PROVINCE="JAEN"
export KEY_CITY="Ubeza"
export KEY_ORG="CASAFX"
#... o cuales otros valores se necesiten.
export KEY_EMAIL="vpnmaster@casafx.dyndns.org"
####endfx
```

```
cp vars ../vars.modif
```

Creación del material criptográfico de la CA

- Si estamos interesados en saber exactamente qué llamadas se hacen a openssl, no hay problema en trazar los scripts mientras los usamos como se indica a continuación[10]:

```
export PS4='+(${BASH_SOURCE}:${LINENO}): ${FUNCNAME[0]:+${FUNCNAME[0]}(): }'
set -o xtrace               #set -vx puede ser tambie'n uqtil.
export SHELLOPTS
sed -i s/\#\!\\/bin\\/sh/\#\!\\/bin\\/bash/g pkitool
<nombrescript>              # 3>&1 1>&2 2>&3 | cat > /tmp/tee
```

- Adelantamos que, para revertir las trazas al terminar, se ha de hacer:

```
sed -i s/\#\!\\/bin\\/bash/\#\!\\/bin\\/sh/g pkitool
set +o xtrace
export -n SHELLOPTS
```

Vamos allá, el script build-ca creará el material criptográfico de nuestra CA:

```
source vars # con lo que las variables en 'vars' ya forman parte
            # de la shell actual.
./clean-all
unset SAN


env KEY_EMAIL=camaster@casafx.dyndns.org ./build-ca --pass
```

Acéptense los valores por defecto de las variables pulsando RET. Pedirá una passphrase que debemos recordar cada vez que usemos la llave privada de la CA más adelante. Durante las pruebas nosotros elegimos simplemente "asdf". Otras preguntas seguirán, pero se tomará la configuración a través de las variables que hemos configurado en el

[10] http://wiki.bash-hackers.org/scripting/debuggingtips#use_shell_debug_output

fichero “vars”, por tanto déjese el valor por defecto presentado pulsando RET. Nótese que el "Common Name" de nuestra CA será entonces "CASAFX CA".

Al terminar, podemos evaluar los ficheros creados:

```
ls keys/
```

```
    ca.crt ca.key index.txt serial
```

Utilizamos los subcomandos de openssl para inspeccionar el material criptográfico generado:

```
# Certificado de la CA (llave pu'blica ma's armadura X.509, firmadas con su llave
openssl x509    -text -in keys/ca.crt     | ccze -A | less -R
# Llave privada de la CA:
openssl rsa     -text -in keys/ca.key     | ccze -A | less -R
```

Creación del par de claves, petición de certificado y certificado para openvpn

La ca.key privada firma cada petición dando lugar al resto de certificados. Las llamadas a openssl llevan el modificador "-nodes" y, por tanto, la llave privada no tendrá contraseña (menos seguro, pero más flexible al no requerir interacción con el usuario). En nuestro testbed ésto será adecuado, en otros despliegues quizás no y la solución es, como se hizo al crear la CA, añadir –pass al llamar al script para que éste evite -nodes al llamar a openssl.

Certificado para el rol de OpenVPN tls-server (dklab1); parámetros DH Usamos el script ./build-key-server:

```
SAN="DNS:dklab1.casafx.dyndns.org, DNS:openvpn1.casafx.dyndns.org" \
./build-key-server dklab1.casafx.dyndns.org
```

Acéptense los valores por defecto pulsando RET, emailAddress se puede sustituir por otra o no según convenga (por defecto será vpnmaster@casafx.dyndns.org). Finalmente pide firmar la petición de certificado:

```
    ...Sign the certificate? [y/n]:y
```

```
ls keys/
```

```
01.pem  dklab1.casafx.dyndns.org.key  index.txt.old
ca.crt  dklab1.casafx.dyndns.org.crt  index.txt                    serial
ca.key  dklab1.casafx.dyndns.org.csr  index.txt.attr               serial.old
```

Inspeccionamos el material criptográfico generado; nótese en la armadura del certificado cómo aparecen las extensiones de la sección [server] (y no de la sección [usr_cert]) de nuestro openssl.cnf; también debe aparecer varios subjectAltName.

```
openssl req     -text -in keys/dklab1.casafx.dyndns.org.csr | ccze -A | less -R
openssl x509    -text -in keys/dklab1.casafx.dyndns.org.crt | ccze -A | less -R
openssl rsa     -text -in keys/dklab1.casafx.dyndns.org.key | ccze -A | less -R
cat keys/index.txt keys/serial
```

OpenVPN usa el algoritmo Diffie-Hellman[11] para negociar una llave privada de sesión, el cual permite convertir una llave compartida en un número cualquiera de llaves criptográficas simétricas. Concretamente el proceso requiere que ambas partes tengan su par de llaves, como ocurre en nuestro caso, y combinando la privada de una y la pública de otra, ambos puedan computar una misma llave. Tal número puede ser usado para encriptar cualquier número de llaves de sesión con las que encriptar el resto de la comunicación. El proceso es parametrizable, de forma que necesitaremos crear unos parámetros Diffie-Hellman para el proceso haciendo de tls-server.

Creamos los parámetros Diffie-Hellman y los inspeccionamos a continuación:

```
./build-dh
ls keys/dh1024.pem
openssl dhparam -text -in keys/dh1024.pem | ccze -A | less -R
```

Certificados para el rol de OpenVPN tls-client (dklab2): Usaremos ./build-key-pkcs12 en lugar de ./build-key. Ambos añaden las extensiones según la sección [usr_cert] y no [server], pero ./build-key-pkcs12 además de generar el par criptográfico, utiliza el formato pkcs12 para reunir ca.crt, <cert>.crt y <priv>.key en un único fichero[12]:

```
./build-key-pkcs12 dklab2.casafx.dyndns.org
```

[11] http://tools.ietf.org/html/rfc2631

[12] "openssl pkcs12" permite operar de forma inversa también, extrayendo desde el .p12 a formato PEM cualquiera de los tres. Ejemplo: http://www.carbm1.com/?p=184

```
...
...Enter Export Password: <nada>
...
```

Vamos a crear otro certificado, simplemente para probar el proceso de revocación de un certificado. Aprovecharemos además para ponerle una passphrase esta vez:

```
./build-key-pkcs12 --pass cualquier_otro.casafx.dyndns.org
```

```
...Enter Export Password: hjkl
```

```
ls keys/
cat keys/serial keys/index.txt
openssl req  -text -in keys/dklab2.casafx.dyndns.org.csr | ccze -A | less -R
openssl x509 -text -in keys/dklab2.casafx.dyndns.org.crt | ccze -A | less -R
openssl rsa  -text -in keys/dklab2.casafx.dyndns.org.key | ccze -A | less -R
```

Creación de CRL (Certificates Revocation List)

Si, por ejemplo, una persona deja de pertenecer a nuestra organización y consecuentemente no debe acceder más a nuestra VPN, su certificado debería ser invalidado de alguna manera. La inclusión de ese certificado en una lista de certificados revocados (CRL) es, efectivamente, el mecanismo que hará al tls-server denegar el acceso a cualquier tls-client que presente el certificado revocado.

- Anotación al respecto de OSCP: openvpn sólo puede verificar si un certificado ha sido revocado a través de una lista de revocación local, no remotamente a través del protocolo OSCP.[13]

Revocación del certificado "cualquier_otro", reconstrucción de la lista CRL.

```
./revoke-full cualquier_otro.casafx.dyndns.org
grep otro.* keys/index.txt


openssl crl -text -in keys/crl.pem | ccze -A | less -R
```

[13] http://backreference.org/2010/05/09/ocsp-verification-with-openssl/

Puesto que openvpn no puede hacer más que comprobaciones en listas locales, la nueva lista CRL debe sustituir a la hipotética lista que hubiese anteriormente (aún no sabemos cómo, pero cuando configuremos openvpn en modo tls-server mostraremos cómo se le indica el fichero de la lista CRL).

Verificación

Los certificados tienen el atributo "Issuer", su "Keyid"... que indican qué CA se debe utilizar para verificarlo; pero ¿dónde debe estar el certificado de la CA para que una verificación se realice automáticamente?

Las librerías libssl[14] que usan la mayoría de las aplicaciones con necesidades criptográficas (entre ellas, como no, openssl) permiten buscar los certificados contra los que hacer verificaciones en /usr/lib/ssl/certs (que es un enlace simbólico a /etc/ssl/certs). A su vez el comando openssl también nos expone esa parte de la API con el subcomando verify.

Por tanto antes de llamar a "openssl verify" debemos mover el certificado de nuestra CA a /etc/ssl/certs. Además tenemos que enlazarlo como se muestra (sí, se utiliza un hash como nombre del enlace):

```
MYCA=casafx-ca.crt
cp -p keys/ca.crt /etc/ssl/certs/${MYCA}
cd /etc/ssl/certs/
MYCAHASH=`openssl x509 -in ${MYCA} -noout -hash`
ln -s ${MYCA} ${MYCAHASH}.0
```

Ya podemos verificarlo sin especificar el certificado de la CA:

```
cd /usr/local/lib/easy-rsa/2.0


openssl verify keys/dklab1.casafx.dyndns.org.crt
```

```
... OK
```

En otro caso, sin copiar y enlazar el hash etc, la salida errónea sería algo como:

```
... error 20 at 0 depth lookup:unable to get local issuer certificate
```

Algunas anotaciones:

- Lo ideal sería crear un paquete deb instalable (usando como referencia ca-certificates) e

[14] http://www.openssl.org/docs/ssl/ssl.html

instalarlo en todas las máquinas que lo necesiten y usen un SO Debian.

- Hemos decidido crear nuestra propia CA, pero no es estrictamente necesario; precisamente cuando ca-certificates está instalado, tenemos certificados correspondientes a las CA de empresas certificadoras comerciales, así como no comerciales (gratuita, caso de cacert.org[15]) que podríamos usar en lugar de crear una propia. Todos estos certificados del paquete ca-certificates se localizan en directorios bajo /usr/share/ca-certificates/, enlazados conjuntamente desde /etc/ssl/certs/.

Verificar el estado de revocado de un certificado en la lista revocación Es necesario concatenar ca.crt y la lista crl.pem en un fichero, como paso previo para que openssl pueda verificarlo; en nuestro ejemplo particular:

```
cat keys/ca.crt keys/crl.pem > crl+ca.pem
openssl verify -CAfile crl+ca.pem -crl_check \
        keys/cualquier_otro.casafx.dyndns.org.crt
```

```
... error 23 at 0 depth lookup:certificate revoked
```

TLS Protocols Test Suite

El programa openssl implementa, a través de sus subcomandos s_server y s_client, un pequeño framework para probar las negociaciones TLS, de forma que podamos circunscribirnos a esta parte de la comunicación en busca de problemas, dejando a un lado los que puedan surgir por requerimientos no cumplidos en el protocolo de la capa superior a TLS (openvpn en este caso). En distintas terminales y en el siguiente orden, ejecútese:

- Nota: s_client necesita que se le especifique la ruta al ca.crt, a pesar de lo comentado anteriormente sobre el resto de programas enlazados con libssl. Úsese -CAfile <file>, o incluso -CApath <dir>.

En una terminal, el servidor[16]:

```
openssl s_server -cert keys/dklab1.casafx.dyndns.org.crt \
                 -key keys/dklab1.casafx.dyndns.org.key -verify 1
```

[15] http://adam.shand.net/archives/2007/the_easy_way_to_generate_openssl_csrs_with_subjectaltnames/
[16] http://www.openssl.org/docs/apps/s_server.html

En otra terminal el cliente[17]:

```
openssl s_client -CAfile keys/ca.crt \
                 -cert keys/dklab2.casafx.dyndns.org.crt \
                 -key keys/dklab2.casafx.dyndns.org.key
```

... escríbase cualquier cosa, debiera aparecer inmediatamente en la otra terminal.

Adicionalmente, el mismo framework nos permite dar un paso más y probar su funcionamiento incluyendo protocolos de aplicación como http (el que nos interesará finalmente es openvpn, pero preliminarmente podemos ya probar con http).

```
# Pa'gina de prueba:
cat >test.html <<EOF
<html><body><h1> Ok
</h1></body></html>
EOF
```

```
openssl s_server -accept 4433 -WWW \
                 -cert keys/dklab1.casafx.dyndns.org.crt \
                 -key keys/dklab1.casafx.dyndns.org.key
```

En otra consola:

```
w3m https://localhost:4433/test.html
```

Al utilizar localhost en la URL, se dará un error de verificación debido a que el certificado está creado con otro nombre común, CN, y otros "Subject Alternative Name". Cuando instalemos el servicio DNS, https://dklab1.casafx.dyndns.org:4443/test.html sería una url válida para que w3m accediese a este "openssl s_server", pues libssl contra la que se enlaza w3m no protestaría por la discrepancia entre el CN/SAN del certificado y el dominio en la url. De todas formas, podemos usar a /etc/hosts provisionalmente:

```
echo '192.168.1.10 dklab1.casafx.dyndns.org' >> /etc/hosts
```

Ya, sin otro artefacto, la verificación será exitosa sólo si el certificado fue firmado por nuestra CA, y el cliente web silenciosamente nos mostrará el contenido de test.html:

[17] http://www.openssl.org/docs/apps/s_client.html

```
w3m https://dklab1.casafx.dyndns.org:4443/test.html
```

Revertimos el cambio sobre /etc/hosts:

```
sed -i s/192.168.1.20\ dklab1.casafx.dyndns.org//g
```

Anotaciones

Para asegurarnos que entendemos la gestión de la CA (en su acepción de estructura de ficheros y directorios), es buen ejercicio intentar identificar qué pasos habría que llevar a cabo para eliminar un certificado, es decir, dejar a la CA en un estado tal como si ese certificado no hubiera existido. Habría que:

- Echar un vistazo a keys/index.txt para ver qué identificador tiene.
- Borrar su keys/<id>.pem
- Borrar el material criptográfico de keys/
- Borrar su referencia en keys/index.txt, editar los índices (3ª columna) para que no haya un hueco libre en la secuencia. Igualmente para el index.txt.old.
- Decrementar el valor en keys/serial y keys/serial.old.

1.1.3. Distribución de los recursos criptográficos creados

Previamente, recapitulemos qué necesita cada instancia de openvpn:
Dklab1 (rol de tls-server) necesita:

- Certificado de la CA
- Certificado de servidor dklab1.casafx.dyndns.org.crt
- Llave privada asociada al anterior
- Parámetros Diffie-Hellman

Dklab2 (rol de tls-client) necesita:

- Certificado de la CA
- Certificado de usuario dklab2.casafx.dyndns.org.crt
- Llave privada asociada al anterior.

Todos, a excepción de los parámetros DH, pueden almacenarse en un mismo fichero gracias al formato pkcs12.[18]. En dklab1 por tanto:

```
mkdir /etc/openvpn/ovpn-keys-casafx.dyndns.org
cp keys/{ca.crt,dklab1.casafx.dyndns.org.key} \
   keys/{dklab1.casafx.dyndns.org.crt,dh1024.pem,crl.pem} \
   /etc/openvpn/ovpn-keys-casafx.dyndns.org
ls -l /etc/openvpn/ovpn-keys-casafx.dyndns.org
```

Los permisos y propietarios son los correctos (a lo sumo, podríamos quitar el flag de escritura para la llave privada, el fichero ".key". Más adelante se referenciarán en el archivo de configuración de openvpn.

1.1.4. Creación de la cuenta no privilegiada openvpn

No queremos que los procesos openvpn se ejecuten bajo una cuenta privilegiada.

```
addgroup --system --no-create-home --disabled-login openvpn
adduser --system --no-create-home --disabled-login openvpn \
        --gid `getent group openvpn|awk -F: '{print $3}'`
id openvpn

touch /var/local/ifconfig-pool-persist_ovpnCASAFX
chown openvpn:root /var/local/ifconfig-pool-persist_ovpnCASAFX
chmod ug+wr,o-wrx /var/local/ifconfig-pool-persist_ovpnCASAFX
```

1.1.5. Configuración de OpenVPN en modo tls-server: /etc/openvpn/casa

tls-server.conf

OpenVPN puede gestionar varias VPN, donde cada una se configura a través de la existencia de un fichero con extensión .conf [19] en /etc/openvpn. Consúltese "man openvpn".

El fichero será autoexplicativo:

[18] http://www.rsa.com/rsalabs/node.asp?id%3D2138

[19] http://openvpn.net/index.php/open-source/documentation/howto.html#examples

```
vim /etc/openvpn/casafx-tls-server.conf
```

```
comp-lzo yes
keepalive 60 320

persist-tun
persist-key
; Tras iniciarse, leer llaves privadas y otras tareas, bajara' los
; privilegios y correra' como un usuario normal creado a tal efecto:
user openvpn
group openvpn
;;verb 3
```

```
server 10.168.1.0 255.255.255.0
;la anterior directiva "server", se expande a:
;mode server
;tls-server
;push topology [topology]
;if dev tun and (topology == net30 or topology == p2p)
;  ifconfig 10.168.1.1 10.168.1.2
;  ifconfig-pool 10.168.1.4 10.168.1.251
;  route 10.168.1.0 255.255.255.0
;  if client-to-client:
;    push "route 10.168.1.0 255.255.255.0"
;  else if topology == net30:
;    push "route 10.168.1.1"
;if dev tap or (dev tun and topology == subnet):
;  ifconfig 10.168.1.1 255.255.255.0
;  ifconfig-pool 10.168.1.2 10.168.1.254 255.255.255.0
;  push "route-gateway 10.168.1.1"
;
;... por tanto la ip para alcanzar esta ma'quina en la vpn es 10.168.1.1
```

```
;... si quisiesemos incluir la interfaz en un bridge (switch a nivel II),
;    habri'a q usar interfaces TAP y, en lugar de "server", "server-bridge":
;
;    server-bridge [ gateway netmask pool-start-ip pool-end-ip ]
;    Es decir, creari'amos el bridge en nuestro SO, lo
;    configurari'amos a nivel 3 con una ip y ma'scara, datos ambos
;    que pasari'amos a la directiva server-bridge aqui' para que
;    identifique y se una a ese bridge cuando openvpn fuese lanzado.
;    Puesto que gracias al bridge los clientes de openvpn recibiri'an una
;    IP en el mismo segmento de red que otras ma'quinas alcanzables por
;    los puertos del bridge, la directiva server-bridge recibe tambie'n
;    el rango de ip disponibles para dar.
```

```
;E'sto permite que los tls-clients puedan verse entre si',
;adema's de al tls-server:
client-to-client

ifconfig-pool-persist /var/local/ifconfig-pool-persist_ovpnCASAFX

;Si no se le da una ip a la interfaz del tls-server, los tls-client's
;podra'n verse entre si' (gracias a client-to-client) pero no a
;la ma'quina del tls-server. E'sto se consigue con ifconfig-noexec.
;ifconfig-noexec

;Si dklab1 estuviese conectado a una segunda red, y tuviese activado el
;forwarding, podemos pedir a los tls-clients cuando se conecten que
;an~adan una ruta para alcanzar esa red del lado de dklab1.
;nota: evidentemente esas otras ma'quinas deberi'an tener una ruta
;      para acceder a las de la vpn, etc.
;push "route 192.168.11.0 255.255.255.0"
;
```

```
;Efectivamente se pueden enviar acciones cada vez que se conecta un cliente;
;adema's e'sto puede hacerse en caliente y de forma especi'fica gracias a la
;funcionalidad client-config-dir de openvpn:
client-config-dir ccd-casafx.dyndns.org
;Y en una shell:
; cd /etc/openvpn
; mkdir ccd-casafx.dyndns.org
;echo 'ifconfig-push 10.168.1.2 255.255.255.0'    \
;      >> ccd-casafx.dyndns.org/dklab2.casafx.dyndns.org
;; Podemos mencionar en este punto que existe un mecanismo DEFAULT
;; para configuraciones cuando no existe un fichero especi'fico. En
;; ese caso la IP se asigna dina'micamente del rango definido con
;; ifconfig-pool y gracias a ifconfig-pool-persist se registra en
;; un fichero, ambas opciones declaradas anterioremente. Entonces
;; se ejecuta la configuracio'n del fichero DEFAULT si disponible:
;; echo 'push "route 10.11.1.0 255.255.255.252"' \
;;      >> ccd-casafx.dyndns.org/DEFAULT
;; ...en este ejemplo se an~adiri'a una ruta al SO. Aprovechamos
;; para mencionar que 1) existen tambie'n opciones iroute para declarar
;; rutas internas de OpenVPN u'tiles en ciertos casos complejos,
;; consu'ltese man openvpn, y 2) que el enrutamiento del SO depende
;; de que esa funcionalidad este' presente y activada; en el caso de
;; Linux se activa ejecutando en una shell las conocidas sentencias:
;; echo 1 > /proc/sys/net/ipv4/ip_forward # Activa hasta el pro'ximo reinicio
;; echo net.ipv4.ip_forward=1 >> /etc/sysctl.conf # Da soporte entre reinic.
```

```
;;
;; Tambie'n es resen~able, por u'ltimo, que en el caso de interfaces
;; TUN (caso que nosotros rechazamos en favor de TAP) y tal como
;; se expuso, las configuraciones IP que se creari'an seri'an punto
;; a punto, de forma que el comando ifconfig-push cambiari'a a algo
;; como:
;; echo 'ifconfig-push 10.168.1.2 10.168.1.1'    \
;;      >> ccd-casafx.dyndns.org/dklab2.casafx.dyndns.org

; Resumiendo, cada vez que se conecte un cliente, se buscan comandos
;que ejecutar en ccd-casafx.dyndns.org/file donde file es el Common Name
;dado al cliente a trave's de su certificado (era uno de los RDN en
;el atributo X.509 Subject). Si no existe tal fichero, se ejecutan
;los de DEFAULT previa asignacio'n dina'mica de una IP del pool definido.
;Se pueden hacer cambios en esos ficheros sin reiniciar openvpn.
; Por su lado, si algu'n cliente esta' conectado a una segunda red privada, y
;tuviese activado el forwarding de paquetes de una iface a otra, podemos
;indicarle a este tls-server que an~ada una ruta para poder alcanzarla;
;el comando para ello es route, existiendo la capacidad de configurar un
;enrutamiento interno adicional con iroute, consu'ltese man openvpn.
```

```
;max-clients X

dh   ovpn-keys-casafx.dyndns.org/dh1024.pem
;pkcs12 ovpn-keys-casafx.dyndns.org/dklab1.casafx.dyndns.org.p12
ca   ovpn-keys-casafx.dyndns.org/ca.crt
cert ovpn-keys-casafx.dyndns.org/dklab1.casafx.dyndns.org.crt
key  ovpn-keys-casafx.dyndns.org/dklab1.casafx.dyndns.org.key

crl-verify ovpn-keys-casafx.dyndns.org/crl.pem
;crl.pem debe ser tal, no puede estar vaci'o o la carga de openvpn fallara'
```

- Anotación sobre la interfaz administrativa: la línea "management localhost 11944" abre dicho puerto y permite, a cualquiera que pueda conectarse -por ejemplo con telnet-, modificar el comportamiento de openvpn[20]. Deshabilítese en un entorno en producción excepto cuando se sepa lo que se está haciendo.

En la parte final, efectivamente, hemos referenciado todo el material criptográfico necesario, y la lista de control de acceso en su estado local.

Como se indicó en los comentarios, gracias al sistema CCD de openvpn dklab2 puede recibir una ip fija tal como 10.168.1.2 (quedando, tal como se declaró, la .1 para dklab1):

```
cd /etc/openvpn
mkdir ccd-casafx.dyndns.org
echo 'ifconfig-push 10.168.1.2 255.255.255.0' \
      >> ccd-casafx.dyndns.org/dklab2.casafx.dyndns.org
```

1.1.6. Integración en el sistema de configuración de red de Debian

El objetivo es que los scripts ifup/ifdown actúen sobre nuestra VPN y ésta se establezca al inicio. Siguiendo las instrucciones de "man openvpn" en debian:

[20] http://openvpn.net/index.php/open-source/documentation/miscellaneous/79-management-interface.html

```
vim /etc/network/interfaces
```

```
...
iface eth0 inet static
      address 192.168.1.10
      netmask 255.255.255.0
####fx:
      post-up ifup ovpnCASAFX-SRV
####endfx
...


####fx:
auto
iface ovpnCASAFX-SRV inet manual
  openvpn casafx-tls-server
####endfx
```

El primer bloque hace referencia a que la interfaz de red ovpnCASAFX-SRV que representa la VPN, depende de la interfaz física eth0 (la que interconecta las máquinas virtuales) y es seguro ordenar que sea levantada a continuación. Por ello, para que ese ifup funcione definimos abajo un bloque "iface" de nombre ovpnCASAFX-SRV donde se indica que es de tipo openvpn y su fichero .conf en /etc/openvpn.

1.1.7. Lanzamiento del servicio

```
invoke-rc.d openvpn status || invoke-rc.d openvpn start
```

1.2. DKLAB2

1.2.1. Instalación de OpenVPN

```
apt-get install openvpn
```

1.2.2. Distribución de los recursos criptográficos desde la CA

Usamos scp, y seguimos la política de /etc/ssl/certs para que el certificado de la CA pueda ser usado para verificaciones por libssl. Recordemos que la instancia openvpn en dklab2, en su rol de tls-client, necesita:

- Certificado de la CA
- Certificado de usuario dklab2.casafx.dyndns.org.crt
- Llave privada asociada a la anterior.

```
mkdir -p /etc/openvpn/ovpn-keys-casafx.dyndns.org/

scp 192.168.1.20:/usr/share/doc/openvpn/examples/easy-rsa/\
keys/{ca.crt,dklab2.casafx.dyndns.org.crt} \
    192.168.1.20:/usr/share/doc/openvpn/examples/easy-rsa/\
keys/{dklab2.casafx.dyndns.org.key,dklab2.casafx.dyndns.org.p12} \
    /etc/openvpn/ovpn-keys-casafx.dyndns.org

cd /etc/ssl/certs/
MYCA=casafx-ca.crt
cp -p /etc/openvpn/ovpn-keys-casafx.dyndns.org/ca.crt ./${MYCA}
MYCAHASH=`openssl x509 -in ./${MYCA} -noout -hash`
ln -s ./${MYCA} ./${MYCAHASH}.0
```

Los permisos y propietarios son los correctos.

1.2.3. Creación de la cuenta no privilegiada openvpn

```
addgroup --system --no-create-home --disabled-login openvpn
adduser --system --no-create-home --disabled-login openvpn \
        --gid `getent group openvpn|awk -F: '{print $3}'`
id openvpn
```

1.2.4. Configuración de OpenVPN en modo tls-client: /etc/openvpn/casaf tls-client.conf

Procuraremos que sea autoexplicativo:

```
vim /etc/openvpn/casafx-tls-client.conf
```

```
management localhost 11944

dev-type tap
dev ovpnCASAFX

comp-lzo yes
keepalive 60 320

persist-tun
persist-key
user openvpn
group openvpn
;verb 3

client
;...se expande a e'sto (pull es para que el multi server nos
;                       de' las opciones como rutas etc     )
;pull
;tls-client

proto tcp
```

```
;En una seccio'n connection se puede usar: bind, connect-retry,
;connect-retry-max, connect-timeout, float, http-proxy,
;http-proxy-option, http-proxy-retry, http-proxy-timeout, local, lport,
;nobind, port, proto, remote, rport, socks-proxy, and socks-proxy-retry .
;Fuera de una seccio'n connection una de esas hace de default de todas.
; Puesto que el openvpn tls-server en dklab1 esta' a la escucha
;en la direccio'n 192.168.1.10 y puerto 1194, nuestra seccio'n
;connection tiene este aspecto:
<connection>
  remote 192.168.1.10 1194
</connection>

; Ejemplo de seccio'n connection usando http-proxy y otro puerto:
;<connection>
;  remote openvpn.casafx.dyndns.org 8888 tcp-client
;  http-proxy proxy.ujaen.es 3128
;  http-proxy-retry
;;  Cualquier cambio de puerto, como el 8888 en este ejemplo, exige usar
;;  iptables para redireccionar el tra'fico: openvpn no permite
;;  escuchar en ma's de un puerto asi' que, a su llegada a dklab1,
;;  netfilter debiera redirigirlo al puerto definido para aquel, 1194.
;</connection>

; Para limitar el logging (por defecto a syslog), podemos hacer que so'lo
; se repitan n mensajes de la misma categori'a, que so'lo se muestren
; errores fatales, o deshabilitarlo:
mute 2
;verb 0
;log /dev/null
```

```
;Si comprueba que el certificado del servidor lleva el atributo
;nsCertType y de valor "server".
remote-cert-tls server

;Si usas nombres de dominio en lugar de IP, no dejes de intentar
;resolver el nombre:
;resolv-retry infinite

;No parece funcionar algo como e'sto si tls-client de un tls-server.
;ifconfig 10.168.1.3 255.255.255.0
;No es problema puesto que usamos una IP fija gracias al sistema CCL
;de OpenVPN que usamos y comentamos detalladamente en el .conf de dklab1.
; En aquel, recordamos, esta' configurado el pool de IP dado por:
; server 10.168.1.0 255.255.255.0
; y 10.168.1.1 es la IP asignada en la VPN a dklab1.
```

```
;El formato pkcs12 lleva ca.crt,cert,key en un mismo fichero. Por cierto,
;si intentamos usar el mismo para varios tls-client, el tls-server intentara'
;darle la misma IP a todos ellos: hay que crear un par key/cert por cada
;tls-client. No obstante, para pruebas, se puede avisar al tls-server de que
;se comparten certs con "duplicate-cn" en su .conf:
pkcs12 ovpn-keys-casafx.dyndns.org/dklab2.casafx.dyndns.org.p12

;;o el anterior(todo en uno) o los siguientes:
;ca   ovpn-keys-casafx.dyndns.org/ca.crt
;cert ovpn-keys-casafx.dyndns.org/dklab2.casafx.dyndns.org.crt
;key  ovpn-keys-casafx.dyndns.org/dklab2.casafx.dyndns.org.key

;README.Debian: Don't specify a 'writepid' option in the .conf files,
;or the init.d script won't be able to stop/reload the tunnels.
```

- Anotación sobre la interfaz administrativa: la línea "management localhost 11944" abre dicho puerto y permite, a cualquiera que pueda conectarse -por ejemplo con telnet-, modificar el comportamiento de openvpn[21]. Deshabilítese en un entorno en producción excepto cuando se sepa lo que se está haciendo.

1.2.5. Integración en el sistema de configuración de red de Debian

```
vim /etc/network/interfaces
```

[21] http://openvpn.net/index.php/open-source/documentation/miscellaneous/79-management-interface.html

```
...
iface eth0 inet static
      address 192.168.1.20
      netmask 255.255.255.0
####fx:
      post-up ifup ovpnCASAFX
####endfx
...

####fx:
auto
iface ovpnCASAFX inet manual
  openvpn casafx-tls-client
####endfx
```

1.2.6. Lanzamiento del servicio

```
invoke-rc.d openvpn status || invoke-rc.d openvpn start
```

1.3. Tests

```
ifconfig ovpnCASAFX-SRV
echo status | socat - tcp4:localhost:11944
ping -c2 10.168.1.1
ping -c2 10.168.1.2
grep openvpn /var/log/syslog | ccze -A | less -R
```

```
ifconfig ovpnCASAFX
echo status | socat - tcp4:localhost:11944
ping -c2 10.168.1.2
ping -c2 10.168.1.1
grep openvpn /var/log/syslog | ccze -A | less -R
```

LyX