Manual de administrador de la Autoridad de Certificación (CA) v1.0

Índice

Descripción

Conjunto de páginas HTML y servlets escritos en Java que via HTTPS tratan de implementar una autoridad de certificación. Se acompañan de una serie de shell scripts comentados para la simplificación de las tareas del administrador. Está basado en utilidades proporcionadas por el Proyecto OpenSSL.
Creado por Gabriel Babiano Huete (gabrielbabiano@yahoo.es) para su Proyecto Fin de Carrera titulado "Estudio e implementación de una Autoridad de Certificación".

Características

Esta autoridad implementa las siguientes **funciones**:

- Certificación inmediata de Peticiones de Firma de Certificado (CSR) según el estándar PKCS#10. Sólo tendrá validez sólo un certificado por dirección de correo electrónico y por DistiguishedName (DN).
- Revocación de certificados firmados por esta autoridad. La Lista de Certificados Revocados (CRL) se actualizará automáticamente tras cada revocación.
- Búsqueda y entrega de certificados válidos de usuarios. Disponible según el estándar X.509 o PKCS#7. Tanto en codificación PEM como DER. Tanto en archivo como directamente al navegador.
- Entrega del certificado raiz de la autoridad. Disponible según el estándar X.509 o PKCS#7. Tanto en codificación PEM como DER. Tanto en archivo como directamente al navegador.
- Entrega de la Lista de Certificados Revocados (CRL) actualizada hasta el momento. Disponible según el estándar X.509 o PKCS#7. Tanto en codificación PEM como DER. Tanto en archivo como directamente al navegador.
- Entrega de la colección de certificados válidos en esta autoridad de certificación incluido el certificado raiz de la autoridad de certificación junto con la CRL actualizada según el estándar PKCS#7. Tanto en codificación PEM como DER. Tanto en archivo como directamente al

navegador.

La certificación de la identidad del usuario está limitada a la de su dirección de correo electrónico y esta autoridad no valida los datos incluidos en el DistiguishedName (DN) del certificado (de forma similar a los antiguos certificados de clase 1 de VeriSign). Dicha certificación de la dirección de correo electrónico se lleva a cabo con la entrega de una clave via email a la dirección de correo electrónico que pretende certificar su CSR.

La **política de las claves** es la siguiente:

- Para asegurar que quien pretende tanto que la autoridad le firme su CSR como la revocación de un certificado válido, la autoridad genera en cada ocasión una clave de tipo numérico y la remite instantáneamente a la dirección de correo electrónico que lo solicita.
- Por cada intento de certificación o de revocación (se lleve a cabo o no) se generará y enviará una clave distinta. La única clave válida será la última recibida en dicha dirección de correo electrónico.

Esta aplicación ha sido testada sobre una distribución SuSE Linux 8.0 Proffesional con los directorios por defecto.

Software necesario

- OpenSSL 0.9.6c-29 (procedente de la distribución SuSE Linux 8.0 Proffesional)
- Apache httpd 1.3.23-73 (procedente de la distribución SuSE Linux 8.0 Proffesional)
- mod-ssl 2.8.7-41 (procedente de la distribución SuSE Linux 8.0 Proffesional)
- jakarta-tomcat 4.0.1-1227 (procedente de la distribución SuSE Linux 8.0 Proffesional)
- mutt 1.3.27i-62 (procedente de la distribución SuSE Linux 8.0 Proffesional)
- Java 2 v1.4.0 JRE (Java Runtime Enviroment) o SDK (Standard Development Kit) (procedente de http://java.sun.com)
- jce_policy-1_4_0.zip (procedente de http://java.sun.com/products/jce/index-14.html)

Instalación

1. Instalar la distribución SuSE Linux 8.0 Proffesional con los mencionados paquetes en los directorios por defecto.
2. Instalar J2SDK 1.4.0_02
3. Instalar "jce_policy-1_4_0.zip"

Configuración

Ejecutar en la línea de comandos:

```
gab:~ #instalarCA.sh
```

Dicho script:

- Hace copia de seguridad de los archivos que se pudieran sobreescribir
- Configura OpenSSL

- Configura el servidor jakarta-tomcat
- Copia los archivos y directorios necesarios para la CA

Copiar al directorio /usr/share/ssl/demoCA/ una serie de archivos que serán de utilidad a la hora de generar números pseudo-aleatorios (ver página del manual sobre openssl genrsa). Es válido cualquier tipo de archivo, pero se recomienda archivos comprimidos (ya que así, poseen menos redundancia), y aunque el manual no indica el tamaño máximo o mínimo de estos archivos, funciona bien con ficheros de entre 2 y 5 Mbytes. En el CD del proyecto no se adjuntan dichos archivos para que sea el propio usuario el que los elija o cree, para aumentar de esta forma la "aleatoriedad".

Suponemos que copiamos al directorio /usr/share/ssl/demoCA/ 3 archivos y los renombramos a: file1, file2 y file3. Para copiar archivos y renombrarlos a la vez, podemos utilizar el comando cp en la línea de comandos (cp archivo_origen archivo_destino), aunque si la ruta de los archivos es un tanto complicada, sería más recomendable el uso de cualquier utilidad para este fin: mc (Midnight Comander) en la propia línea de comandos o konqueror (similar al explorador de Microsoft Windows) en X-Windows, por ejemplo.

Generación de la clave privada RSA (de 2048 bits) de la propia CA (cakey.pem):

```
gab:~ # openssl genrsa -rand file1:file2:file3 2048
> /usr/share/ssl/demoCA/private/cakey.pem
9223693 semi-random bytes loaded
Generating RSA private key, 2048 bit long modulus
.....................................................++
.....................................................................++

e is 65537 (0x10001)
```

NOTA: en cuanto al tamaño de la clave privada tenemos que tener en cuenta que por defecto, openssl toma 512 bits. Sin la política de JCE sólo se permite criptografía fuerte (claves de hasta 1024 bits). Con la política de JCE de criptografía ilimitada, keytool llega a admitir una longitud de clave máxima de 2048 bits. Se ha utilizado un tamaño de clave de 2048 bits. En la siguiente tabla se muestra el tiempo de proceso que se requirió para los distintos tamaños de clave privada. Observar que a partir de un tamaño de clave de aproximadamente 8192 bits, el tiempo que requiere la creación de la clave privada se haría inadmisible.

Tamaño (bits)	Tiempo (hh:mm:ss)
2^{10}=1024	00:00:01
2^{11}=2048	00:00:02
2^{12}=4096	00:00:05
2^{13}=8192	00:05:50
2^{14}=16384	00:20:10

NOTA: se podría añadir la opción -des3 y encriptar la clave privada con triple DES al dejarla en el disco duro. Al incluir esta opción, se aumenta en seguridad, pero también es un gran inconveniente ya que cada vez que utilicemos la clave privada, como por ejemplo firmar las CSR, deberemos de introducir dicha clave, lo cual es muy engorroso y no siempre es posible.

Se desea ser la CA raiz por lo que se debe autofirmar nuestra clave pública. El comando que nos permite autofirmarnos (con nuestra clave privada) la clave pública y obtener cacert.pem es:

```
gab:~ # openssl req -new -x509 -key /usr/share/ssl/demoCA/private/cakey.pem -
days 365 > /usr/share/ssl/demoCA/cacert.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Spain
Locality Name (eg, city) []:Sevilla
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Universidad de
Sevilla
Organizational Unit Name (eg, section) []:Departamento de Telematica
Common Name (eg, YOUR name) []:Autoridad de Certificacion (CA)
Email Address []:root@192.168.0.3
```

NOTA: en stateProvinceName se ha introducido Spain en lugar de España y que no se ha utilizado las vocales tildadas en ningún campo. Si hubiésemos introducido España, al contener una ñ, no habría problemas para el keytool, pero sí para openssl ca ya que detectaría un caracter ilegal en el tipo ASN.1. Se ha intentado corregir esta situación, modificando a cualquiera de las opciones de la máscara del archivo de configuración situado por defecto en /usr/share/ssl/openssl.cnf:

```
# This sets a mask for permitted string types. There are several options.
# default: PrintableString, T61String, BMPString.
# pkix : PrintableString, BMPString.
# utf8only: only UTF8Strings.
# nombstr : PrintableString, T61String (no BMPStrings or UTF8Strings).
# MASK:XXXX a literal mask value.
# WARNING: current versions of Netscape crash on BMPStrings or UTF8Strings

# so use this option with caution!
string_mask = nombstr
```

Pero sin lograr el resultado esperado.

Inicializar la CA ejecutando:

```
gab:~ # /usr/share/ssl/iniciaCA.sh
rm: cannot remove `/usr/share/ssl/demoCA/importante.txt': no such file or
directory
rm: cannot remove `/usr/share/ssl/demoCA/emails.txt': no such file or directory
rm: cannot remove `/usr/share/ssl/demoCA/serial*': no such file or directory
rm: cannot remove `/usr/share/ssl/demoCA/index.txt*': no such file or directory
rm: cannot remove `/usr/share/ssl/demoCA/miIndex.txt': no such file or directory

rm: cannot remove `/usr/share/ssl/demoCA/newcerts/*': no such file or directory
rm: cannot remove `/usr/share/ssl/demoCA/files/*': no such file or directory
rm: cannot remove `/usr/share/ssl/demoCA/importante.txt: no such file or
directory
Using configuration from /usr/share/ssl/opnessl.cnf
```

Para poder utilizar el protocolo HTTPS, hay que crear un par de claves RSA (en este caso de 2048 bits) en un keystore con el alias tomcat:

```
gab:~ # /usr/java/j2sdk1.4.0_02/bin/keytool -genkey -alias tomcat -keyalg RSA -
keystore /opt/jakarta/conf/.keystore -keysize 2048
Escriba la contraseña del almacén de claves: miclave
```

```
¿Cuáles son su nombre y su apellido?
[Unknown]: 192.168.0.3
¿Cuál es el nombre de su unidad de organización?
[Unknown]: Departamento de Telematica
¿Cuál es el nombre de su organización?
[Unknown]: Universidad de Sevilla
¿Cuál es el nombre de su ciudad o localidad?
[Unknown]: Sevilla
¿Cuál es el nombre de su estado o provincia?
[Unknown]: Spain
¿Cuál es el código de país de dos letras de la unidad?
[Unknown]: ES
¿Es correcto CN=192.168.0.3, OU=Departamento de Telematica, O=Universidad de
Sevilla, L=Sevilla, ST=Spain, C=ES?
[no]: y

Escriba la contraseña clave para
(INTRO si es la misma
```

Notar que cuando se pregunta el nombre y apellido, en realidad se nos pregunta acerca del CommonName (CN) y ahí hemos de introducir el nombre de la máquina a la que se accede de forma "literal", es decir, que es distinto poner 192.168.0.3, www.us.es o us.es, por ejemplo.

Para crear a partir de la CSR X.509 en formato PEM siguiendo el estándar PKCS#10:

```
gab:~ # /usr/java/j2sdk1.4.0_02/bin/keytool -certreq -alias tomcat -
keystore /opt/jakarta/conf/.keystore -file /opt/jakarta/conf/tomcat.csr
Escriba la contraseña del almacén de claves: miclave
```

En nuestro caso, y a falta de una CA raiz que nos firme el CSR la firmará nuestra propia CA de forma manual.

```
gab: # cd /usr/share/ssl
gab:/usr/share/ssl # openssl ca -in /opt/jakarta/conf/tomcat.csr -
out /opt/jakarta/conf/tomcat.cer -notext
Using configuration from /usr/share/ssl/openssl.cnf
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName :PRINTABLE:'ES'
stateOrProvinceName :PRINTABLE:'Spain'
localityName :PRINTABLE:'Sevilla'
organizationName :PRINTABLE:'Universidad de Sevilla'
organizationalUnitName:PRINTABLE:'Departamento de Telematica'
commonName :PRINTABLE:'192.168.0.3'
Certificate is to be certified until Nov 17 08:56:53 2003 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Instalamos el certificado de la Autoridad de Certificación (CA) en el keystore

```
gab:/usr/share/ssl # /usr/java/j2sdk1.4.0_02/bin/keytool -import -alias ca -
keystore /opt/jakarta/conf/.keystore -file /usr/share/ssl/demoCA/cacert.pem
Escriba la contraseña del almacén de claves: miclave
Propietario: EMAILADDRESS=root@192.168.0.3, CN=Autoridad de Certificacoin (CA),
OU=Departamento de Telematica, O=Universidad de Sevilla, L=Sevilla, ST=Spain,
```

```
C=ES
Emisor: EMAILADDRESS=root@192.168.0.3, CN=Autoridad de Certificacoin (CA),
OU=Departamento de Telematica, O=Universidad de Sevilla, L=Sevilla, ST=Spain,
C=ES
Número de serie: 0
Válido desde: Sun Nov 17 16:31:38 CET 2002 hasta: Mon Nov 17 16:31:38 CET 2003
Huellas digitales del certificado:
MD5: ED:ED:D8:DB:F4:84:53:58:6E:DC:37:7E:A8:8F:53:6B
SHA1: 7E:6A:D5:09:69:64:D2:93:28:4E:1C:51:B9:8F:DF:BB:DE:EF:43:8B
¿Confiar en este certificado? [no]: y
Se ha añadido el certificado al almacén de claves
```

Una firmada la CSR por la CA, obtenemos el certificado (/opt/jakarta/conf/tomcat.cer).

Se importa el certificado a la keystore bajo el mismo alias de tomcat:

```
gab:~ # /usr/java/j2sdk1.4.0_02/bin/keytool -import -alias tomcat -
file /opt/jakarta/conf/tomcat.cer -keystore /opt/jakarta/conf/.keystore
Escriba la contraseña del almacén de claves: miclave Se ha añadido el
certificado al almacén de claves
```

Configurar el firewall (si se va a activar) para poder permitir los servicios HTTP y HTTPS hacia el exterior, en caso que se desee hacerlo.

Comprobar la correcta configuración en el navegador dirigiéndonos por el protocolo HTTPS (puerto 443) al directorio /CA/ de la máquina en que instalado. En este ejemplo nos dirigiríamos hacia:

```
gab:~ # mozilla https://192.168.0.3/CA
```

Uso

Arranque

Para arrancar el servidor jakarta se debe ejecutar:

```
gab:~ # setDefaultJava --devel SunJava2
gab:~ # source setJava --devel SunJava2
gab:~ # /opt/jakarta/bin/startup.sh
Guessing CATALINA_HOME from catalina.sh to /opt/jakarta/bin/..
Setting CATALINA_HOME to /opt/jakarta/bin/..
Using CLASSPATH: /opt/jakarta/bin/../bin/bootstrap.jar:/opt/jakarta/bin/../common
stdext.jar:/opt/jakarta/bin/../common/lib/jndi.jar:/opt/jakarta/bin/../common/lib
common.jar:/opt/jakarta/bin/../common/lib/naming-resources.jar:/opt/jakarta/bin/.
0.9.7.0.jar:/opt/jakarta/bin/../common/lib/xerces.jar:/opt/jakarta/bin/../server/
1.2.jar:/opt/jakarta/bin/../server/lib/servlets-common.jar:/opt/jakarta/bin/../se
invoker.jar:/opt/jakarta/bin/../server/lib/servlets-manager.jar:/opt/jakarta/bin/
webdav.jar:/opt/jakarta/bin/../server/lib/tomcat-ajp.jar:/opt/jakarta/bin/../serv
util.jar:/opt/jakarta/bin/../server/lib/warp.jar:/opt/jakarta/bin/../lib/jasper-c
factory.jar:/usr/java/j2sdk1.4.0_02/lib/tools.jar
Using CATALINA_BASE: /opt/jakarta/bin/..
Using CATALINA_HOME: /opt/jakarta/bin/..
Using JAVA_HOME: /usr/java/j2sdk1.4.0_02
```

Parada

Para parar el servidor jakarta:

```
gab:~ # /opt/jakarta/bin/shutdown.sh
Guessing CATALINA_HOME from catalina.sh to /opt/jakarta/bin/..
Setting CATALINA_HOME to /opt/jakarta/bin/..
Using CLASSPATH: /opt/jakarta/bin/../bin/bootstrap.jar:/opt/jakarta/bin/../common
stdext.jar:/opt/jakarta/bin/../common/lib/jndi.jar:/opt/jakarta/bin/../common/lib
common.jar:/opt/jakarta/bin/../common/lib/naming-resources.jar:/opt/jakarta/bin/.
0.9.7.0.jar:/opt/jakarta/bin/../common/lib/xerces.jar:/opt/jakarta/bin/../server/
1.2.jar:/opt/jakarta/bin/../server/lib/servlets-common.jar:/opt/jakarta/bin/../se
invoker.jar:/opt/jakarta/bin/../server/lib/servlets-manager.jar:/opt/jakarta/bin/
webdav.jar:/opt/jakarta/bin/../server/lib/tomcat-ajp.jar:/opt/jakarta/bin/../serv
util.jar:/opt/jakarta/bin/../server/lib/warp.jar:/opt/jakarta/bin/../lib/jasper-c
factory.jar:/usr/java/j2sdk1.4.0_02/lib/tools.jar
Using CATALINA_BASE: /opt/jakarta/bin/..
Using CATALINA_HOME: /opt/jakarta/bin/..
Using JAVA_HOME: /usr/java/j2sdk1.4.0_02
```

Reinicializado

Para el reinicializado de la CA podemos ejecutar:

```
gab:~ # /usr/share/ssl/iniciaCA.sh
```

Dicho script:

- Borra los certificado emitidos, la CRL antigua y los fichero índice.
- Mantiene la clave privada y el certificado de la CA.
- Vuelve a crear una CRL vacía.
- El número de serie vuelve a ser 01.

NOTA: esta operación es definitiva y tras el borrado, no se pueden recuperar los datos tras el reinicio

---

Desinstalación

Para desinstalar la Autoridad de Certificación (CA) se ejecutará:

```
/usr/share/ssl/desinstalarCA.sh
```