Seguridad en Redes

Práctica 2.3. Certificados digitales y modelos de confianza

A. OpenSSL

A.1. Creación de una CA

Revisa el fichero de configuración por defecto (`/usr/lib/ssl/openssl.cnf`).

Aunque normalmente habría que adaptarla a nuestras necesidades, usaremos directamente la configuración por defecto, para lo cual, hay que crear algunos ficheros y directorios:

```
$ mkdir demoCA
$ mkdir demoCA/newcerts
$ mkdir demoCA/private
$ touch demoCA/index.txt
$ echo 01 > demoCA/serial
$ echo 01 > demoCA/crlnumber
```

Para crear un certificado raíz autofirmado a partir de una nueva clave privada, se usa el comando `req` con las opciones `-x509` y `-new`:

```
$ openssl req -x509 -new -days 3650 -keyout
demoCA/private/cakey.pem -out demoCA/cacert.pem
```

La nueva clave `demoCA/private/cakey.pem` es una clave RSA creada a partir de los parámetros especificados en el archivo de configuración. Si se usa la opción `-newkey` en lugar de `-new`, se pueden especificar los parámetros de la clave en la línea de comando, por ejemplo `-newkey rsa:1024`. Si queremos usar una clave ya existente, en lugar de crear una clave nueva, se debe usar la opción `-key` en lugar de `-keyout`.

Para visualizar el contenido del certificado raíz de la CA se puede usar la sigiente orden:

```
$ openssl x509 -in demoCA/cacert.pem -noout -text
```

Para visualizar la clave privada de la CA se puede usar la siguiente orden:

```
$ openssl pkey -in demoCA/private/cakey.pem -noout -text
```

Ejercicio 1

- Crea un certificado raíz autofirmado para la CA (con contraseña “seguridad”). Selecciona la opción por defecto para todos los campos salvo para *Common Name* (CN), donde debes poner “CA”.
- Visualiza el contenido del certificado raíz y el contenido de la clave privada de la CA

Entrega #1: Entrega el archivo con el certificado de la CA y el archivo con la clave privada de la CA creados en el ejercicio anterior

A.2. Creación de solicitudes de firma de certificado

Para crear una CSR (*Certificate Signing Request*), se usa el comando `req` con la opción `-new`:

```
$ openssl req -new -keyout userkey.pem -out usercsr.pem
```

También se pueden usar las opciones `-newkey` y `-key`.

La solicitud, que va firmada con la clave privada del solicitante, puede verse con:

```
$ openssl req -in usercsr.pem -noout -text
```

Se puede verificar la firma con:

```
$ openssl req -verify -in usercsr.pem
```

Ejercicio 2

- Crea dos CSRs para dos usuarios nuevos distintos (por ejemplo, `user1` y `user2`), con contraseña "seguridad". Selecciona la opción por defecto para todos los campos salvo para Common Name (CN), donde debes proporcionar un nombre único para cada solicitud.
- Visualiza y verifica ambos CSRs

Entrega #2: Entrega los dos archivos CSR creados en el ejercicio anterior

A.3. Creación y verificación de certificados

Para firmar una CSR con la CA por defecto y generar un certificado, se usa el comando `ca`:

```
$ openssl ca -in usercsr.pem -out usercert.pem
```

Antes de firmar el certificado se comprueba la firma de la solicitud y se pide confirmación.

Para verificar un certificado, se usa el comando `verify`, indicado con la opción `-CAfile` el fichero con los certificados de las CAs en las que se confía (en este caso, solamente una):

```
$ openssl verify -CAfile demoCA/cacert.pem usercert.pem
```

Para visualizar el contenido del certificado y extraer información del mismo (aunque normalmente se puede ver en el fichero del certificado), se puede usar el comando `x509`. Por ejemplo:

```
$ openssl x509 -in usercert.pem -noout -text
$ openssl x509 -in usercert.pem -noout -pubkey
```

Ejercicio 3

- Firma la solicitud del usuario `user1` creada en el ejercicio anterior para generar el correspondiente certificado.
- Visualiza el contenido de los archivos `demoCA/serial` y `demoCA/index.txt` y del directorio `demoCA/newcerts`
- Firma la solicitud del usuario `user2` creada en el ejercicio anterior para generar el correspondiente certificado.

- Visualiza de nuevo el contenido de los archivos `demoCA/serial` y `demoCA/index.txt` y del directorio `demoCA/newcerts`
- Verifica los certificados de ambos usuarios `user1` y `user2`.
- Visualiza el contenido de ambos certificados usando el comando `x509`

Entrega #3: Entrega los dos archivos de certificado firmados creados en el ejercicio anterior

A.4. Formatos de certificados

Existen varios formatos para almacenar codificar los certificados digitales:

- El formato PEM (*Privacy Enhaced Mail*) codifica el certificado X.509 en formato ASCII (Base64). Este es el formato por defecto que utiliza OpenSSL.
- El formato DER (*Distinguish Encoding Rules*) es similar a PEM, pero codificado en formato binario.
- El formato PKCS12 (*Public-Key Cryptography Standards*) permite almacenar el certificado X.509 junto con la clave privada en un solo fichero. Este formato se utiliza en muchos navegadores y clientes de correo.

Para convertir de formato PER a formato DER, o viceversa, se usa el comando `x509` con las opciones `-inform` y `-outform`, respectivamente. Por ejemplo:

```
$ openssl x509 -in usercert.pem -out usercert.der -outform DER
$ openssl x509 -in usercert.der -inform DER -out usercert.pem
```

Para exportar un certificado, junto con su clave, a formato PKCS12 se usa el comando `pkcs12`:

```
$ openssl pkcs12 -export -in usercert.pem -inkey userkey.pem -
out usercert.p12
```

Ejercicio 4

- Convierte los certificados creados en el ejercicio 3 a formato DER y PKCS12

Entrega #4: Entrega los archivos de certificados en formato DER y PKCS12 creados en el ejercicio anterior

A.5. Revocación de certificados

Para revocar un certificado, se usa el comando `ca` con la opción `-revoke`:

```
$ openssl ca -revoke usercert.pem
```

Para generar una nueva CRL (*Certificate Revocation List*), se usa el comando `ca` con la opción `-gencrl`:

```
$ openssl ca -gencrl -out crl.pem
```

(NOTA: cada vez que se revoca un certificado, es necesario reconstruir la CRL usando la orden anterior, para incluir la nueva revocación)

Para examinar la CRL, se usa el comando `crl` con la opción `-text`:

```
$ openssl crl -in crl.pem -noout -text
```

Para verificar la CRL, se usa el comando `crl` con la opción `-CAfile`:

```
$ openssl crl -CAfile demoCA/cacert.pem -in crl.pem
```

Para verificar un certificado comprobando que no esté en la CRL, se usa el comando `verify` con la opción `-crl_check`, indicando con la opción `-CRLfile` el fichero con la CRL:

```
$ openssl verify -crl_check -CAfile demoCA/cacert.pem -CRLfile
crl.pem usercert.pem
```

Si esto no funciona (por tratarse de una versión anterior), hay que crear un fichero con los certificados de las CAs y las CRLs e indicarlo con la opción `-CAfile`:

```
$ cat demoCA/cacert.pem crl.pem > cacrl.pem
$ openssl verify -CAfile cacrl.pem -crl_check usercert.pem
```

Ejercicio 5

- Comprueba la base de datos `demoCA/index.txt` para los certificados de los usuarios `user1` y `user2` creados en el ejercicio 3. Observa que ambos certificados están marcados como válidos (comienza por letra `V`)
- Revoca el certificado de `user1` y comprueba de nuevo la base de datos `demoCA/index.txt`. Observa que el certificado de este usuario está marcado como revocado (comienza por letra `R`)
- Genera una CRL
- Examina el contenido de la CRL y comprueba que ésta incluye la revocación del certificado del `user1`
- Verifica el certificado de `user1` con el comando `verify -crl_check` y comprueba que éste está revocado
- Revoca el certificado de `user2` y comprueba de nuevo la base de datos `demoCA/index.txt`. Observa que el certificado de este usuario está marcado como revocado (comienza por letra `R`)
- Si verificas el certificado de `user2` con el comando `verify -crl_check` comprobarás que éste todavía no está incluido en CRL y por tanto no aparece como revocado. Para ello es necesario reconstruir la CRL.
- Genera de nuevo la CRL para incluir la revocación del certificado de `user2`
- Examina el contenido de la CRL y comprueba que ésta incluye la revocación de los certificado de `user1 y user2`
- Verifica el certificado de `user2` con el comando `verify -crl_check` y comprueba que éste está revocado

Entrega #5: Entrega el archivo CRL creado en el ejercicio anterior

B. GnuPG

B.1. Firma de claves (*Web of trust*)

Para firmar una clave una vez que se ha verificado su validez, se usa el comando `--sign-key`:

```
$ gpg --sign-key <id>
```

Para firmar con otra clave (por defecto, se usará siempre la primera), se usa la opción `--local-user`:

```
$ gpg --sign-key --local-user <id1> <id2>
```

Se firmará la clave con identificador `id2` con la clave con identificador `id1`.

Para asignar un nivel de confianza en el propietario de la clave, se usa el comando `--edit-key` con el subcomando `trust`:

```
$ gpg --edit-key <id> trust quit
```

Para mostrar la validez de las claves, hay que utilizar la opción `show-uid-validity`:

```
$ gpg --list-options show-uid-validity --list-keys
```

Para recalcular la validez de las claves y mostrar un resumen de la red de confianza, se usa el comando `--check-trustdb`:

```
$ gpg --check-trustdb
```

Ejercicio 6

- Crea tres claves con nombre `user1`, `user2` y `user3` (utiliza el comando `gpg --full-gen-key` y usa la contraseña "seguridad"). Firma la clave de `user2` con `user1` y la clave de `user3` con `user2`. Establece la confianza en `user2` a *total* (4) y la confianza en `user3` a *desconocida* (1) para que sea calculada. ¿Cuál será la validez de `user3`? Comprueba la validez de las claves.
- Cambia la confianza en `user2` a *dudosa* (3). ¿Cuál será ahora la validez de `user3`? Comprueba la validez de las claves.
- (Opcional) Añade otras dos claves, fírmalas con `user1` (para que sean válidas), establece su confianza a *dudosa* (3) y firma con ellas la clave `user3`. ¿Cuál será ahora la confianza en `user3`? Comprueba la validez de las claves.

Entrega #6: Copia y entrega la salida del comando que muestra la validez de las claves y del comando que muestra el resumen de la red de confianza.