Seguridad en Redes

Práctica 2.3. Certificados digitales y modelos de confianza

Preparación del entorno

Las prácticas 2.1, 2.2 y 2.3 se pueden realizar en cualquier computador con S.O. Linux o en cualquier máquina virtual Linux. Si ya dispones de él, tan sólo es necesario instalar (en caso de que no estén instalados) los paquetes `openssl` y `gnupg`.

La segunda opción, si no dispones de un S.O. Linux propio, es descargar el servicio virtualizado `SER.ova` del campus virtual e importarlo con VirtualBox en tu PC. Esta máquina virtual (MV) ya cuenta con los paquetes `openssl` y `gnupg` instalados. Para esta práctica sólo se necesita una única MV (sin clones). Se recomienda disponer de conexión a Internet en la MV, por lo que el "Adaptador de red 1" (correspondiente a la interfaz `eth0` de la MV) debe configurarse en modo NAT.

NOTAS IMPORTANTES:
1. este documento está editado con MS Word y algunos caracteres especiales, especialmente el guion (-) o las comillas (″), a veces utilizan una codificación diferente a la que se usa en Linux. Por tanto, si haces un "copia y pega" del comando desde el documento a la consola de Linux, éste podría fallar. En tal caso, se recomienda revisar estos caracteres especiales o reescribir el comando completo.
2. Algunos comandos son demasiado largos y están separados en dos o más líneas dentro de este documento. Cuando se copien en la consola de Linux, deben escribirse en una única línea.

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 siguiente 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
```

Este comando crea, por defecto, una nueva clave privada de tipo RSA (archivo `userkey.pem`) y una solicitud de firma de certificado (archivo `usercsr.pem`), que incluye la correspondiente clave pública y la firma del propio solicitante

Si ya disponemos de una clave privada generada anteriormente, podemos usar la opción `-key`. Esta clave puede ser distinto tipo (RSA, DSA, o curva elíptica).

El contenido de la solicitud de firma de certificado 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 (por ejemplo, `user1` y `user2`).
- 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 certificados 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 la clave pública de un usuario con identificador `<id>`, se usa el siguiente comando:

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

Este comando usará, por defecto, la primera clave privada que se encuentre en el anillo de claves para realizar la firma.

Para firmar con otra clave privada, distinta a la primera, se usa la opción `--local-user`:

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

En este caso se firmará la clave pública del usuario con identificador `<id2>` usando la clave privada del usuario con identificador `<id1>`.

Para ver el nivel de **confianza** en el propietario de una clave pública y la **validez** la misma, se usa el siguiente comando:

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

Para cambiar un nivel de confianza en el propietario de una clave pública, se usa el siguiente comando:

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

Si queremos comprobar la validez de todas las claves públicas, se puede usar el siguiente comando:

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

Para mostrar un resumen de la red de confianza, se usa el siguiente comando:

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

Ejercicio 6

- Crea tres parejas de claves con nombres de usuario `user1`, `user2` y `user3` (utiliza el comando `gpg --gen-key` y usa la contraseña "seguridad" para las tres). Comprueba que, inicialmente, el nivel de confianza en los tres usuarios es absoluta y la validez de sus claves públicas también es absoluta (para ello, usa el comando: `gpg --edit-key <id> quit`), sustituyendo `<id>` por `user1`, `user2` y `user3`, respectivamente). Esto es así porque los tres usuarios que hemos creado se consideran usuarios locales y, por defecto, se les asigna nivel de confianza absoluta.
- Establece la confianza en `user2` a *total* (4) (mediante el comando `gpg --edit-key user2 trust quit`) y a continuación comprueba el nivel de confianza y la validez de su clave. Observarás que la clave de `user2` deja de ser válida (`validez: desconocido`) ya que ahora no se considera un usuario local y su clave pública no está firmada por ningún otro usuario de confianza absoluta o total.
- Firma la clave pública de `user2` con la clave privada de `user1` (mediante el comando `gpg --sign-key --local-user user1 user2`). A continuación, comprueba de nuevo el nivel de confianza y la validez de su clave de `user2` (ahora su clave si se considera válida con `validez: total`)
- Establece la confianza en `user3` a *desconocida* (1). Comprueba el nivel de confianza y la validez de su clave. A continuación, firma la clave pública de `user3` con la clave privada de `user2` y comprueba de nuevo la validez de su clave. Observa que la clave de `user3` se considera válida (`validez: total`), ya que esta firmada por un usuario de confianza total.
- Cambia la confianza en `user2` a *dudosa* (3). Comprueba la validez de las claves. ¿Cuál es ahora la validez de la clave de `user3`? ¿Por qué?
- Crea dos parejas de claves adicionales con nombres de usuario `user4` y `user5` (usa la contraseña "seguridad" para ambas), establece su nivel de confianza a *dudosa* (3) y firma ambas claves públicas con la clave privada de `user1` (para que sean consideradas válidas). Comprueba el nivel de confianza y validez de las claves de estos dos nuevos usuarios. A continuación, firma la clave pública de `user3` con las claves privadas de `user4` y `user5`. Comprueba la validez de las claves. ¿Cuál es ahora la validez de la clave de `user3`? ¿Por qué?
- Muestra el resumen de la red de confianza (comando `gpg --check-trustdb`) e intenta interpretar la salida de dicho comando.

Entrega #6: Una vez finalizado completamente el ejercicio anterior, copia y entrega la salida de los comandos que muestran el nivel de confianza y la validez de las claves de los cinco usuarios creados, así como la salida del comando que muestra el resumen de la red de confianza.