# Universidad Politécnica de Cartagena


## Escuela Técnica Superior de Ingeniería de Telecomunicación

## SEGURIDAD EN REDES DE COMUNICACIONES

# Práctica 3: Certificados Digitales para servidores web

**María Dolores Cano Baños**

**Natalio López Martínez**

# 1.Introducción

Debido a las carencias de seguridad del protocolo HTTP se implementa una capa adicional para el transporte de información que resuelve los aspectos de seguridad carentes en los servidores web.

Esta capa viene definida por el estándar TLS o Transport Layer Protocol que gracias a las librerias OpenSSL permite encriptar la información a través de las conexiones seguras SSL Secure Sockets Layer.

SSL es un estándar definido por Netscape Comunication Corporation y que actualmente implementan todos los servidores seguros. En el estándar SSL no solamente proporciona la encriptación y la confidencialidad de los datos sino que también proporcionan autenticación de cliente y servidor y garantiza la integridad de los datos a través del protocolo TCP/IP. Estas son las características deseables para que podamos considerar una conexión segura.

Como la mayoría de servidores (entre ellos apache) ya implementan estos estándares a través de las herramientas de OpenSSL sólo necesitamos de un Certificado válido para poder establecer conexiones de tipo HTTPS.

## 1.1.-Certificados Digitales

Para solucionar el problema de la Autentificación en las transacciones por Internet es necesario un sistema identificativo único de cada entidad o persona. Ya existen los sistemas criptográficos de clave asimétrica que se basan en el hecho de compartir una clave pública entre varios usuarios y otra privada, sólo conocida por el propietario. En general, esta clave se comparte mediante un directorio electrónico (normalmente en formato LDAP) o una página web.

Sin embargo, este modo de compartir presenta un inconveniente importante: nada nos garantiza que la clave pertenezca al usuario con el que está asociada. Por ejemplo, un hacker puede corromper la clave pública que aparece en el directorio remplazándola con su propia clave pública. Por consiguiente el hacker podrá descifrar todos los mensajes que se cifraron con la clave que aparece en el directorio. De aquí surgió la idea de implementar una especie de documento de identidad electrónica que identificara sin dudas a su emisor.

La solución a este problema condujo a la aparición de los **Certificados Digitales** o **Certificados electrónicos**, documentos electrónicos basados en la criptografía de clave pública y en el sistema de firmas digitales. Un certificado permite asociar una clave pública con una entidad (una persona, un equipo, etc…) para garantizar su validez. El certificado es como la tarjeta de identificación de la clave, emitido por una entidad llamada *Entidad de certificación* (que frecuentemente se abrevia **CA**, por sus siglas en inglés).

Las principales Autoridades Certificadoras actuales son:

- Verisign (www.verising.com).
- Filial de RSA Data Security Inc.
- Thawte (www.thawte.com).
- La Fabrica Nacional de Moneda y Timbre (www.fnmt.es).

La entidad de certificación es responsable de emitir los certificados, de asignarles una fecha de validez y de revocarlos antes de esta fecha en el caso de que la clave (o su dueño) estén en una situación de riesgo.

El sistema es análogo a otros de uso común, como el D.N.I., en el que una autoridad de confianza (estado o policía) atestigua que la persona portadora de dicho documento es quién dice ser.

### Estructura de los certificados

Los certificados son pequeños archivos divididos en dos partes:

- La parte que contiene la información
- La parte que contiene la firma de la entidad de certificación

La estructura de los certificados está estandarizada por la norma **X.509** (más precisamente, X.509v3) de la UIT, que define la información que contiene el certificado:

- La versión de X.509 a la que corresponde el certificado:
  El número de serie del certificado;
  El algoritmo de cifrado utilizado para firmar el certificado;
  El nombre (DN, siglas en inglés de *Nombre distinguido*) de la entidad de certificación que lo emite;
  La fecha en que entra en vigencia el certificado;
  La fecha en que finaliza el período de validez del certificado;
  El objeto de utilización de la clave pública;
  La clave pública del dueño del certificado;
  La firma del emisor del certificado (*huella digital*).

La entidad de certificación firma toda esta información (información + clave pública del solicitante) y esto implica que una **función hash** crea una huella digital de esta información y luego este hash se cifra con la clave privada de la entidad de certificación. La clave pública se distribuye antes de tiempo para permitir a los usuarios verificar la firma de la *entidad de certificación* con su clave pública.

Cuando un usuario desea comunicarse con otra persona, sólo debe obtener el certificado del receptor. Este certificado contiene el nombre y la clave pública del receptor, y está firmado por la entidad de certificación. De esta forma, es posible verificar la validez del mensaje aplicando, primero, la función hash a la información contenida en el certificado y, segundo, descifrando la firma de la entidad de certificación con la clave pública y comparando los dos resultados.

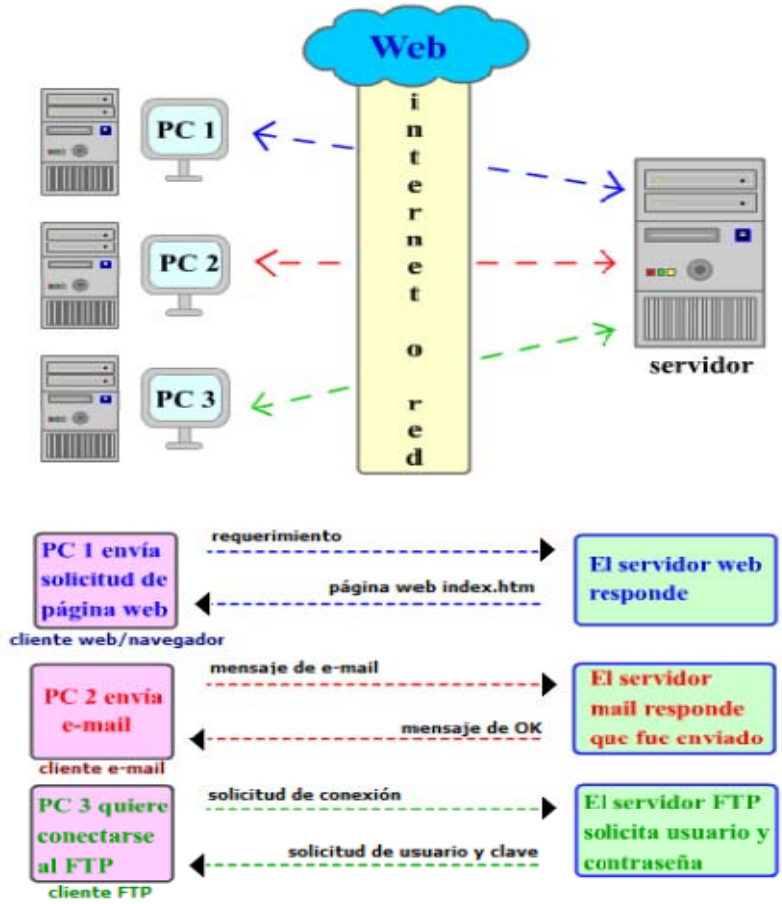
## 1.2.-Servidor Apache

La misión de un servidor web es básicamente la de traducir una URL (Uniform Resource Locator) a un nombre de archivo y enviar este archivo a través de Internet, o a traducir la URL por el nombre de un programa, correr ese programa y enviar la respuesta de vuelta. Cuando se conecta a una URL, lo que hace es enviar un mensaje a la máquina que tiene esa dirección. Lógicamente todos esperamos que la máquina a la que enviamos ese mensaje esté activa y lista para recibir y actuar sobre el mensaje enviado. Una URL tiene tres partes:

*scheme*>://<*host*>/<*path*>

Así, si <*scheme*> es *http*, significa que el navegador debe utilizar el protocolo http (Hypertext Transfer Protocol). Si <*host*> es *www.upct.es* y <*path*> es /, significa normalmente ir a la página superior de la máquina. La parte <*host*> puede contener una dirección IP o un nombre, que el navegador convertirá a una dirección IP. Las peticiones llegan al puerto 80 (puerto por defecto de HTTP) de la máquina <*host*>.

El servidor Apache (http://www.apache.org) es el servicio que se encarga de resolver las peticiones de las páginas de Internet de los clientes utilizando el protocolo de Internet http, como muestra la siguiente figura:



Apache es un servidor web mas utilizado en la actualidad en Internet, ocupando un lugar clave en la infraestructura de la red y cubre más de la mitad del mercado que se competidor directo Microsoft. Esto no solo es debido a que sea de libre distribución y por tanto gratuito, sino que su código es abierto, lo que significa que puede ser examinado por cualquiera, si hay errores miles de personas los detectan. Este hecho lo hace más fiable que cualquier software comercial. En particular, Apache es extensible a través de una tecnología establecida para escribir nuevos módulos (Tomcat, mod_ssl, etc…), que permiten añadir nuevas caracteristicas. El modulo ssl que utilizaremos añade capacidades criptográficas al servidor web Apache.

# 2. Objetivos

- Aprender a instalar un servidor web con facilidades criptográficas.
- Aprender a solicitar un certificado X.509 firmado por una Autoridad Certificadora..
- Aprender a instalar el certificado x.509.

# 3. Desarrollo de la práctica.

Antes de comenzar la descarga e instalaciones de las herramientas necesarias, procedemos a explicar el proceso para instalar y configurar un servidor web con facilidades criptográficas:

– Configurar un servidor web (descargándolo de Internet o comprando el software o un servidor que lo incluya).
– Instalarlo.
– Crear una pareja de claves pública y privada para el servidor.
– Opcionalmente, crear un certificado autofirmado para poner operación el servidor web de inmediato.
– Enviar la clave a una Autoridad Certificadora (CA).
– Enviar otros documentos de soporte a la Autoridad Certificadora.
– Recibir de la Autoridad Certificadora el certificado X.509 v3 firmado.
– Instalar el certificado en el servidor web.

Nosotros realizamos el proceso utilizando el servidor Apache + mod_SSL y VeriSign.

Siempre que se deban transferir datos confidenciales, como información de tarjetas de crédito, entre el servidor Web y el cliente, es aconsejable contar con una conexión segura y cifrada que incluya autenticación. mod_ssl proporciona un potente cifrado mediante los protocolos de nivel de zócalo con seguridad (SSL) y de seguridad del nivel de transporte (TLS) para la comunicación HTTP entre un cliente y un servidor Web. Mediante SSL/TSL se establece una conexión privada entre el servidor Web y el cliente. Se asegura la integridad de los datos y el cliente y el servidor pueden autenticarse mutuamente.

Para este propósito, el servidor envía un certificado SSL con información que prueba la identidad válida del servidor antes de responder a cualquier petición a una URL. A su vez, de esta manera se garantiza que el servidor es el único punto final correcto de la comunicación. Además, el certificado genera una conexión cifrada entre el cliente y el servidor que puede transportar información sin el riesgo de exponer contenido importante en formato de sólo texto.

mod_ssl no implementa los protocolos SSL/TSL por sí mismo, sino que actúa como interfaz entre Apache y una biblioteca SSL. En SUSE Linux, se emplea la biblioteca OpenSSL, la cual se instala automáticamente con Apache.

El efecto más visible de usar mod_ssl con Apache es que las URL van precedidas de https:// en lugar de http://.

## 3.1 Descarga e instalación del servidor web Apache + mod_SSL.

Vamos a ver los paquetes necesarios a instalar y el procedimiento a seguir para cada uno de ellos:

### Necesitamos instalar PHP5

Instalar PHP es simple:
linux:~ # yast2 -i php5 o puedes usar yast2 y en buscar poner: 'php5'.

### Apache

En este paso instalaremos Apache 2.2:

linux:~ # yast2 -i apache2
y el modulo PHP para Apache
linux:~ # yast2 -i apache2-mod_php5

También comprobamos que en el fichero de configuración la línea APACHE_MODULES contiene 'php5':

linux:~ # less /etc/sysconfig/apache2

El resultado es debería ser una linea similar a:

APACHE_MODULES="actions alias auth_basic authn_file authz_host authz_groupfile authz_default authz_user authn_dbm autoindex cgi dir env expires include log_config mime negotiation setenvif ssl suexec userdir php5"

Iniciamos el servidor Apache:
linux:~ # /etc/init.d/apache2 start

Starting httpd2 (prefork)  done

Para comprobar que versión de Apache tenemos instalada y donde están los ficheros de configuración:

```
linux:~ # httpd2 –V

Server version: Apache/2.2.8 (Linux/SUSE)
Server built:   Sep 25 2008 11:59:55
Server's Module Magic Number: 20051115:11
Server loaded:  APR 1.2.12, APR-Util 1.2.12
Compiled using: APR 1.2.12, APR-Util 1.2.12
Architecture:   32-bit
Server MPM:     Prefork
  threaded:     no
    forked:     yes (variable process count)
Server compiled with....
-D APACHE_MPM_DIR="server/mpm/prefork"
-D APR_HAS_SENDFILE
-D APR_HAS_MMAP
-D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
-D APR_USE_SYSVSEM_SERIALIZE
-D APR_USE_PTHREAD_SERIALIZE
-D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
-D APR_HAS_OTHER_CHILD
-D AP_HAVE_RELIABLE_PIPED_LOGS
-D DYNAMIC_MODULE_LIMIT=128
-D HTTPD_ROOT="/srv/www"
-D SUEXEC_BIN="/usr/sbin/suexec2"
-D DEFAULT_PIDLOG="/var/run/httpd2.pid"
-D DEFAULT_SCOREBOARD="logs/apache_runtime_status"
-D DEFAULT_LOCKFILE="/var/run/accept.lock"
-D DEFAULT_ERRORLOG="/var/log/apache2/error_log"
-D AP_TYPES_CONFIG_FILE="/etc/apache2/mime.types"
-D SERVER_CONFIG_FILE="/etc/apache2/httpd.conf"
```

El siguiente paso sería habilitar el módulo SSL para Apache. Esto lo hacemos mediante el script llamado ‘a2enflag’:

```
linux:~ # a2enflag SSL
```

Este commando lo que hace es modificar el fichero /etc/sysconfig/apache2:

APACHE_SERVER_FLAGS=""

Pasa a:  APACHE_SERVER_FLAGS=" SSL"

Esto se podría hacer de igual manera mediante un editor de texto.

Reiniciamos Apache:

```
linux:~ # /etc/init.d/apache2 restart

Syntax OK
Shutting down httpd2 (waiting for all children to terminate)        done
Starting httpd2 (prefork)                                           done
```

Comprobamos si la maquina escucha trafico 'http' y 'https':

```
linux:~ # netstat -lpt | grep "http"

tcp      0     0 *:www-http          *:*              LISTEN   3212/httpd2-prefork
tcp      0     0 *:https             *:*              LISTEN   3212/httpd2-prefork
```

***Comprobar si está desactivado el cortafuegos.***

## Activamos Secure Socket Layer ( SSL) para Apache. Creamos un Certificado 'ficticio'.

La creación de un certificado ficticio es fácil. Basta con llamar al guión /usr/bin/gensslcert. De esta forma se crean o sobrescriben los archivos siguientes:

- /etc/apache2/ssl.crt/ca.crt
- /etc/apache2/ssl.crt/server.crt
- /etc/apache2/ssl.key/server.key
- /etc/apache2/ssl.csr/server.csr

También se coloca una copia de ca.crt en /srv/www/htdocs/CA.crt para su descarga.

 Importante:

Los certificados ficticios no se deben emplear nunca en un sistema de producción. Sólo se deben utilizar con fines de prueba.

```
    linux:~ # gensslcert
comment         mod_ssl server certificate
name
C               XY
ST              unknown
L               unknown
U               web server
O               SuSE Linux Web Server
CN              linux-iifk.site
email           webmaster@linux-iifk.site
srvdays         730
CAdays          2190
```

```
creating CA key ...
710687 semi-random bytes loaded
Generating RSA private key, 2048 bit long modulus
..................+++
...........+++
e is 65537 (0x10001)

creating CA request/certificate ...
`/etc/apache2/ssl.crt/ca.crt' -> `/srv/www/htdocs/CA.crt'

creating server key ...
710687 semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
................++++++
...++++++
e is 65537 (0x10001)

creating server request ...
creating server certificate ...
Signature ok
subject=/C=XY/ST=unknown/L=unknown/O=SuSE Linux Web Server/OU=web
server/CN=linux-iifk.site/emailAddress=webmaster@linux-iifk.site
Getting CA Private Key

Verify: matching certificate & key modulus

Verify: matching certificate signature
/etc/apache2/ssl.crt/server.crt: OK
```

## Configuración de hosts virtuales

El término *host virtual* hace referencia a la capacidad de Apache para proporcionar servicio a varios identificadores de recursos universales (URI, del inglés Universal Resource Identifiers) desde el mismo equipo físico. Esto significa que varios dominios, como www.ejemplo.com y www.ejemplo.net pueden ejecutarse desde un mismo servidor Web en un equipo físico.

Es una costumbre habitual emplear hosts virtuales para ahorrar esfuerzos administrativos (sólo es necesario realizar el mantenimiento de un servidor Web) y gastos de hardware (no es necesario emplear un servidor dedicado para cada dominio). Los hosts virtuales pueden estar basados en nombres, en IP o en puertos.

Accedemos al directorio de Apache donde se encuentran los virtual hosts:

```
linux:~ # cd /etc/apache2/vhosts.d/
```

Copiamos el fichero vhost-ssl.template a otro con el nombre de tu maquina. En nuestro ejemplo ‘linux’.

```
linux:/etc/apache2/vhosts.d # cp vhost-ssl.template linux.conf
```

Reiniciamos el servicio de Apache:

```
    linux:~ # /etc/init.d/apache2 restart
Syntax OK
Shutting down httpd2 (waiting for all children to terminate)      done
Starting httpd2 (prefork)                                         done
```

**Testeamos el servidor para conexiones http:**

Para ello creamos el fichero test.php .

```
linux:~ # echo "<?php phpinfo(); ?>" >> /srv/www/htdocs/test.php
```

Abrimos el navegador y comprobamos si acepta conexiones http:

```
http://ip.address.ofyour.box/test.php
```

Nos aparecerá una pantalla similar a esta:

**Testeamos el servidor para conexiones https:**

```
https://ip.address.ofyour.box/test.php
```

Podemos verificar que módulos se encuentran presentes obteniendo la lista de los módulos apache:

```
linux:~ # httpd2 -M
```

## 3.2 Diferencia entre el uso de http y https.

Para poder apreciar con mayor claridad la diferencia entre ambos protocolos, vamos a realizar un formulario web, que nos solicite un usuario y contraseña, y que una vez insertados nos de un saludo de bienvenida.

Este formulario se hará en php y tendrá una apariencia similar al ejemplo:


Código ejemplo ( para vuestra ayuda) ( más información www.webestilo.com/php):

```
<!-- Manual de PHP de WebEstilo.com -->
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>
<FORM ACTION="procesa2.phtml" METHOD="POST">
Introduzca su nombre:<INPUT TYPE="text" NAME="nombre"><BR>
Introduzca sus apellidos:<INPUT TYPE="text" NAME="apellidos"><BR>
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</body>
</html>
```

procesa2.phtml

```
<!-- Manual de PHP de WebEstilo.com -->
<html>
```

```
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>
El nombre que ha introducido por GET es: <?php echo $_GET['nombre']," ",$_GET['apellidos']
?><br>
El nombre que ha introducido por POST es: <?php echo $_POST['nombre']," 
",$_POST['apellidos'] ?>
<br>
</body>
</html>
```

Una vez realizado el formulario, lo cargaremos con ambos protocolos http y https, para que mediante el analizador de red “Wireshark” podamos comprobar la diferencia en cómo se pasan los parámetros de usuario y contraseña.