

---

## **Práctica 1.4**

Implantación de Aplicaciones Web

Curso 2025/2026

# Índice general

<b>1</b>	<b>HTTPS. Creación y configuración de un certificado SSL/TLS autofirmado en Apache</b>	<b>1</b>
1.1	Instalación del servidor web Apache . . . . .	1
1.2	Creación del certificado autofirmado . . . . .	2
1.3	Cómo automatizar la creación de un certificado autofirmado . . . . .	3
1.4	Cómo consultar la información del sujeto del certificado . . . . .	4
1.5	Cómo consultar la fecha de caducidad del certificado . . . . .	4
1.6	Configuración de un VirtualHost con SSL/TSL en el servidor web Apache . . . . .	4
1.7	Tareas a realizar . . . . .	6
1.8	Entregables . . . . .	7
1.8.1	Documento técnico . . . . .	7
1.8.2	Scripts de Bash . . . . .	8
<b>2</b>	<b>Referencias</b>	<b>9</b>
<b>3</b>	<b>Licencia</b>	<b>10</b>

# 1 HTTPS. Creación y configuración de un certificado SSL/TLS autofirmado en Apache

En esta práctica vamos a crear un **certificado SSL/TLS autofirmado** con la herramienta `openssl`. Una vez creado vamos a configurar el [servidor web Apache](#) para que utilice dicho certificado.

El proceso de creación de un certificado autofirmado consta de los siguientes pasos:

1. Crear una clave privada y un certificado autofirmado.
2. Configurar la clave privada y el certificado autofirmado en el servidor web.

Tenga en cuenta que cuando un cliente acceda desde un navegador web a un sitio web que utiliza un certificado autofirmado, se mostrará un mensaje de advertencia indicando que **el certificado no es de confianza, porque no ha sido emitido por una [autoridad de certificación \(CA\)](#) reconocida**.

Si el cliente acepta el certificado, el navegador web lo almacenará en su almacén de certificados y no volverá a mostrar el mensaje de advertencia.

No se recomienda utilizar certificados autofirmados en sitios web públicos, por lo tanto, esta práctica se utiliza con fines educativos para conocer con detalle cómo funciona el proceso de creación y configuración de un certificado autofirmado.

El proceso de creación de un certificado emitido por una [autoridad de certificación \(CA\)](#), consta de los siguientes pasos:

1. Crear una clave privada.
2. Crear una solicitud de certificado o *Certificate Signing Request (CSR)* y enviarla a una autoridad de certificación (CA).
3. La CA valida la solicitud y emite un certificado.
4. El certificado se instala en el servidor web.

## 1.1. Instalación del servidor web Apache

En primer lugar deberemos tener instalado un [servidor web Apache](#) en nuestra máquina. Si todavía no lo hemos instalado, podemos hacerlo con los siguientes comandos:

```
1 sudo apt update
2 sudo apt install apache2 -y
```

## 1.2. Creación del certificado autofirmado

Para crear un certificado autofirmado vamos a utilizar la utilidad `openssl`.

Este es el comando que vamos a utilizar:

```
1 sudo openssl req \  
2 -x509 \  
3 -nodes \  
4 -days 365 \  
5 -newkey rsa:2048 \  
6 -keyout /etc/ssl/private/apache-selfsigned.key \  
7 -out /etc/ssl/certs/apache-selfsigned.crt
```

Vamos a explicar cada uno de los parámetros que hemos utilizado.

- `req`: El subcomando `req` se utiliza para crear solicitudes de certificado en formato PKCS#10. También puede utilizarse para crear certificados autofirmados, que será el uso que le daremos en esta práctica.
- `-x509`: Indica que queremos crear un certificado autofirmado en lugar de una solicitud de certificado, que se enviaría a una autoridad de certificación.
- `-nodes`: Indica que la clave privada del certificado no estará protegida por contraseña y estará sin encriptar. Esto permite a las aplicaciones usar el certificado sin tener que introducir una contraseña cada vez que se utilice.
- `-days 365`: Este parámetro indica la validez del certificado. En este caso hemos configurado una validez de 365 días.
- `-newkey rsa:2048`: Este parámetro indica que queremos generar una nueva clave privada RSA de 2048 bits junto con el certificado. La longitud de clave de 2048 bits es un estándar razonable para la seguridad en la actualidad.
- `-keyout /etc/ssl/private/apache-selfsigned.key`: Indica la ubicación y el nombre del archivo donde se guardará la clave privada generada. En este caso, hemos seleccionado que se guarde en la ruta `/etc/ssl/private/apache-selfsigned.key`.
- `-out /etc/ssl/certs/apache-selfsigned.crt`: Indica la ubicación y el nombre del archivo donde se guardará el certificado. En este caso, hemos seleccionado que se guarde en la ruta `/etc/ssl/certs/apache-selfsigned.crt`.

Al ejecutar el comando tendremos que introducir una serie de datos por teclado que se añadirán al certificado. Los datos que tenemos que introducir son los siguientes:

- **Código del país** de 2 caracteres que identifica el país donde se emite el certificado. Por ejemplo, `ES` para España.
- **Provincia** donde se emite el certificado.
- **Localidad** donde se emite el certificado.
- **Nombre de la organización** para la que se emite el certificado.
- **Nombre de la unidad o sección** de la organización.
- **Nombre del dominio** para el que se emite el certificado.

- **Email.**

**Ejemplo:**

Este ejemplo muestra la salida que obtendremos al ejecutar el comando anterior para crear el certificado autofirmado.

```
1 You are about to be asked to enter information that will be incorporated
2 into your certificate request.
3 What you are about to enter is what is called a Distinguished Name or a DN.
4 There are quite a few fields but you can leave some blank
5 For some fields there will be a default value,
6 If you enter '.', the field will be left blank.
7 -----
8 Country Name (2 letter code) [AU]:
9 State or Province Name (full name) [Some-State]:
10 Locality Name (eg, city) []:
11 Organization Name (eg, company) [Internet Widgits Pty Ltd]:
12 Organizational Unit Name (eg, section) []:
13 Common Name (e.g. server FQDN or YOUR name) []:
14 Email Address []:
```

### 1.3. Cómo automatizar la creación de un certificado autofirmado

Para automatizar la creación de un certificado autofirmado desde un script de *Bash*, podemos hacer uso del parámetro `-subj` que nos permite pasar los datos se adjuntan al certificado como argumentos desde la línea de comandos.

**Ejemplo:**

```
1 #!/bin/bash
2 set -x
3
4 # Configuramos las variables con los datos que necesita el certificado
5 OPENSSL_COUNTRY="ES"
6 OPENSSL_PROVINCE="Almeria"
7 OPENSSL_LOCALITY="Almeria"
8 OPENSSL_ORGANIZATION="IES Celia"
9 OPENSSL_ORGUNIT="Departamento de Informatica"
10 OPENSSL_COMMON_NAME="practica-https.local"
11 OPENSSL_EMAIL="admin@iescelia.org"
12
13 # Creamos el certificado autofirmado
14 sudo openssl req \
15     -x509 \
16     -nodes \
17     -days 365 \
18     -newkey rsa:2048 \
19     -keyout /etc/ssl/private/apache-selfsigned.key \
20     -out /etc/ssl/certs/apache-selfsigned.crt \
21     -subj "/C=$OPENSSL_COUNTRY/ST=$OPENSSL_PROVINCE/L=$OPENSSL_LOCALITY/O=
    $OPENSSL_ORGANIZATION/OU=$OPENSSL_ORGUNIT/CN=$OPENSSL_COMMON_NAME/
    emailAddress=$OPENSSL_EMAIL"
```

## 1.4. Cómo consultar la información del sujeto del certificado

```
1 openssl x509 -in /etc/ssl/certs/apache-selfsigned.crt -noout -subject
```

## 1.5. Cómo consultar la fecha de caducidad del certificado

```
1 openssl x509 -in /etc/ssl/certs/apache-selfsigned.crt -noout -dates
```

## 1.6. Configuración de un VirtualHost con SSL/TLS en el servidor web Apache

### Paso 1

Editamos el archivo de configuración del virtual host donde queremos habilitar el tráfico HTTPS.

En nuestro caso, utilizaremos el archivo de configuración que tiene Apache por defecto para SSL/TLS, que está en la ruta: `/etc/apache2/sites-available/default-ssl.conf`.

El contenido del archivo será el siguiente:

```
1 <VirtualHost *:443>
2     #ServerName practica-https.local
3     DocumentRoot /var/www/html
4     DirectoryIndex index.php index.html
5
6     SSLEngine on
7     SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt
8     SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key
9 </VirtualHost>
```

Las directivas que hemos configurado son:

- `<VirtualHost *:443>`: Indica que este virtual host escuchará en el puerto 443 (HTTPS).
- `ServerName`: Indica el nombre de dominio y se utiliza para indicar al servidor web Apache qué peticiones debe servir para este virtual host. En nuestro ejemplo estamos utilizando el dominio `practica-https.local`.
- `DocumentRoot`: Es la ruta donde se encuentra el directorio raíz del host virtual.
- `SSLEngine on`: Configuramos que este virtual host utilizará SSL/TLS.
- `SSLCertificateFile`: Indica la ruta donde se encuentra el certificado autofirmado.
- `SSLCertificateKeyFile`: Indica la ruta donde se encuentra la clave privada del certificado autofirmado.

## Paso 2

Habilitamos el virtual host que acabamos de configurar.

```
1 sudo a2ensite default-ssl.conf
```

Tenga en cuenta que estamos utilizando el nombre de archivo **default-ssl.conf** porque estamos utilizando el archivo que tiene Apache por defecto para configurar un virtual host con SSL/TLS, pero en su caso puede ser otro.

## Paso 3

Habilitamos el módulo SSL en Apache.

```
1 sudo a2enmod ssl
```

## Paso 4

Configuramos el virtual host de HTTP para que redirija todo el tráfico a HTTPS.

En nuestro caso, el virtual host que maneja las peticiones HTTP está en el archivo de configuración que utiliza Apache por defecto para el puerto 80: `/etc/apache2/sites-available/000-default.conf`.

El contenido del archivo será el siguiente:

```
1 <VirtualHost *:80>
2     #ServerName practica-https.local
3     DocumentRoot /var/www/html
4
5     # Redirige al puerto 443 (HTTPS)
6     RewriteEngine On
7     RewriteCond %{HTTPS} off
8     RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
9 </VirtualHost>
```

Las directivas que hemos configurado son:

- **RewriteEngine On**: Habilita el motor de reescritura de URLs y nos permite usar reglas de reescritura.
- **RewriteCond %{HTTPS} off**: Esta directiva es una condición que comprueba si la petición recibida utiliza HTTPS o no. Si se cumple esta condición, entonces se ejecuta la siguiente línea.
- **RewriteRule ^ https://%{HTTP\_HOST}%{REQUEST\_URI} [L,R=301]**: Las reglas de reescritura tienen la siguiente sintaxis **RewriteRule Pattern Substitution [flags]**.
  - **Pattern**: Es el patrón que se debe cumplir en la URL solicitada para que la regla de reescritura se aplique. En este caso, `^` coincide con el principio de la URL, por lo que se aplicará a todas las solicitudes.
  - **Substitution**: Es la URL a la que se redirige la solicitud. En este caso, se utiliza el valor `https://%{HTTP_HOST}%{REQUEST_URI}` y por lo tanto se redirige la solicitud a HTTPS manteniendo el mismo nombre de dominio y URI.
  - **flags**: Son los flags que se pueden utilizar para modificar el comportamiento de la regla de reescritura. En este caso, el flag `[L,R=301]` indica que es una redirección permanente (Código de estado: 301).

Las directivas utilizan las siguientes **variables del servidor** que se obtienen de la cabecera de la petición HTTP:

- `#{HTTPS}`: Contiene el texto `on` si la conexión utiliza SSL/TLS o `off` en caso contrario.
- `#{HTTP_HOST}`: Contiene el nombre de dominio que se ha utilizado en la petición del cliente para acceder al sitio web.
- `#{REQUEST_URI}`: Contiene la URI que ha utilizado el cliente para acceder al sitio web. Por ejemplo, `/index.html`. Si la petición incluye parámetros éstos estarán almacenados en la variable `#{QUERY_STRING}`.

En la [documentación oficial del módulo `mod\_rewrite` de Apache](#) puede encontrar más información sobre estas directivas.

### Paso 5

Para que el servidor web Apache pueda hacer la redirección de HTTP a HTTPS es necesario habilitar el módulo `rewrite` en Apache.

```
1 sudo a2enmod rewrite
```

### Paso 6

Reiniciamos el servicio de Apache

```
1 sudo systemctl restart apache2
```

### Paso 7

Una vez llegado a este punto, es necesario comprobar que el puerto 443 está abierto en las reglas del firewall para permitir el tráfico HTTPS.

### Paso 8

Accede desde un navegador web al nombre de dominio que acabas de configurar. En nuestro caso será: `https://practica-https.local`.

## 1.7. Tareas a realizar

En esta práctica tendremos que realizar la instalación de la [pila LAMP](#) y la **configuración de un certificado SSL/TLS autofirmado** en el [servidor web Apache](#), en una instancia EC2 de [Amazon Web Services \(AWS\)](#) con la última versión de [Ubuntu Server](#).

A continuación se describen **muy brevemente** algunas de las tareas que tendrá que realizar.

1. Crea una instancia EC2 en AWS.
2. La **Amazon Machine Image (AMI)** que vamos a seleccionar para esta práctica será una **Community AMI** con la última versión de **Ubuntu Server**.

3. Cuando esté creando la instancia deberá configurar los puertos que estarán abiertos para poder conectarnos por SSH y para poder acceder por HTTP/HTTPS.
  - SSH (TCP)
  - HTTP (TCP)
  - HTTPS (TCP)
4. Crea un par de claves (pública y privada) para conectar por SSH con la instancia. También puedes hacer uso de las claves que te proporciona AWS Academy (*vockey.pem*).
5. Crea una dirección **IP elástica** y asígnala a la instancia EC2.
6. Una vez que haya iniciado su instancia deberá hacer uso de los **scripts de bash** que diseñó en las prácticas anteriores para automatizar la instalación de la pila LAMP.
7. Automatice la creación y configuración de un certificado autofirmado SSL/TLS con la utilidad `openssl` para el servidor web Apache.
8. Busque cuál es la dirección IP elástica de su instancia y compruebe que puede acceder a ella desde un navegador web.

## 1.8. Entregables

Deberá crear un repositorio en [GitHub](#) con el nombre de la práctica y añadir al profesor como colaborador.

El repositorio debe tener el siguiente contenido:

- Un **documento técnico** con la descripción de todos los pasos que se han llevado a cabo.
- Los **scripts de Bash** que se han utilizado para automatizar la creación y configuración de un certificado SSL/TLS autofirmado en el servidor web Apache.

Además del contenido anterior puede ser necesario crear otros archivos de configuración. A continuación se muestra un ejemplo de cómo puede ser la estructura del repositorio:

```
1  .├──
2  README.md├──
3  conf├──
4    000-default.conf├──
5    └── default-ssl.conf├──
6  scripts├──
7    .env├──
8    install_lamp.sh├──
9    setup_selfsigned_certificate.sh
```

### 1.8.1. Documento técnico

El documento técnico `README.md` tiene que estar escrito en [Markdown](#) y debe incluir **como mínimo** los siguientes contenidos:

- Descripción del proceso de creación y configuración del certificado SSL/TLS autofirmado en el servidor web Apache.

### 1.8.2. Scripts de Bash

El directorio `scripts` debe incluir los siguientes archivos:

- `.env`: Este archivo contiene todas las variables de configuración que se utilizarán en los scripts de Bash.
- `install_lamp.sh`: Script de Bash con la automatización del proceso de instalación de la pila LAMP.
- `setup_selfsigned_certificate.sh`: Script de Bash con la automatización del proceso de creación y configuración de un certificado SSL/TLS autofirmado en Apache.

## 2 Referencias

- [¿Qué es HTTPS?](#). Documentación oficial de Cloudflare.
- [Book: OpenSSL Cookbook](#). Ivan Ristić.
- [Infraestructura de clave pública \(PKI\)](#). Wikipedia.
- [Autoridad Certificadora \(CA\)](#). Wikipedia.
- [X.509](#). Wikipedia.
- [RSA](#). Wikipedia.
- [Módulo mod\\_rewrite de Apache](#). Documentación oficial de Apache.

## **3 Licencia**

Esta página forma parte del curso Implantación de Aplicaciones Web de José Juan Sánchez Hernández y su contenido se distribuye bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.