Instalación SSL

BLAS BARRAGÁN ROMÁN 2N DAW SEMI

PROFESORA: MANUELA SANTANDREU BENAVENT

Índice

Que es SSL?

Por sus siglas (Secure Sockets Layer). Hablamos de un protocolo de cifrado que nos garantiza la seguridad de las comunicaciones a través de internet.

Nos ofrece un canal seguro entre dos ordenadores o dispositivos que operen a través de internet o en una red interna.

Uno de sus usos mas frecuentes es el de proteger la comunicación entre un navegador web y un servidor web.

Este protocolo, cambia la dirección del sitio web HTTP a HTTPS.

Uno de sus usos mas frecuentes es para proteger la comunicacion entre un navegador web y un servidor web.
Este protocolo, cambia la direccion del sitio web HTTP a HTTPS

SSL es compatible con los siguientes principios de seguiridad:

- Cifrado: protege la transmision de datos.
- Autenticacion: garantiza que el servidor al que se conecta es, en efecto, el servidor correcto.
- Integridad de los datos: garantiza que los datos solicitados o enviados son realmente los datos legitimos.

Podemos utilizar SSL para proteger:

- Transacciones de pago a través de internet.
- Trafico en una intranet, como compartir archivos o conexiones a bases de datos.
- Servidores de correo electrónico.
- Conexiones entre los clientes de correo electrónico y sus servidores.
- Transferencia de archivos mediante HTTPS y servicios SFTP.

Instalación

Requisitos previos:

- Acceso a un servidor Ubuntu 20.04 con usuario ROOT sudo habilitado.
- Apache instalado en el servidor.

Habilitar modulo SSL:

- Para poder utilizar los certificados SSL, debemos habilitar el modulo de Apache que le da soporte al cifrado SSL

- Habilitamos mod_ssl con el siguiente comando:

```
$ sudo a2enmod ssl
```

```
barragan@server-blas:/$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled
```

- Reiniciamos Apache para activar el modulo:

```
$ sudo systemctl restart
```

```
barragan@server-blas:/$ sudo systemctl restart
barragan@server-blas:/$ _
```

Creación de certificados:

- Crearemos un nuevo certificado que albergará la información básica de nuestro sitio junto con el archivo de claves necesario para que el servidor trate de manera segura los datos encriptados.

- Creamos la clave y sus archivos con el siguiente comando:

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/nombre-archivo.key -out /etc/ssl/certs/nombre-archivo.crt
```

```
barragan@server-blas:/$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-autofirmado.key -out /etc/ssl/certs/apache-autofirmado.crt
Generating a RSA private key
.....+++++
.....+++++
writing new private key to '/etc/ssl/private/apache-autofirmado.key'
-----
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.
-----
```

- Este comando consta de:

    - openssl: herramienta para crear y administrar certificados, claves y otros archivos de OpenSSL.
    - req -x509: especifica que deseamos usar la firma de certificados (CSR) X.509. “X.509” es un estándar de infraestructura de claves públicas al que se adhieren SSL y TLS para la administración de claves y certificados.
    - -nodes: omite la opción de proteger nuestro certificado con una contraseña para que Apache pueda leer el archivo, sin intervención del usuario, cuando se inicie el servidor.
    - -days 365: tiempo durante el cual el certificado será válido. Muchos navegadores modernos rechazan certificados válidos por más de un año.
    - -newkey: especifica que vamos a generar una clave ademas del propio certificado
    - rsa:2048 le indica que la clave será RSA de 2048 bits.
    - -keyout: indica dónde se colocará el archivo de clave (.key).
    - -out: indica dónde se colocará el archivo de certificado (.crt).

Creación de certificados:

- El comando nos realiza un pequeño cuestionario donde debemos indicar:

```
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Valencia
Locality Name (eg, city) []:Valencia
Organization Name (eg, company) [Internet Widgits Pty Ltd]:BBR_DEV
Organizational Unit Name (eg, section) []:BBR
Common Name (e.g. server FQDN or YOUR name) []:misitio.com
Email Address []:webmaster@misitio.com_
```

- Country Name (2 letter code) [XX]: Codigo del pais - España **[ES]**
- State or Province Name (full name) []: Provincia - Valencia
- Locality Name (eg, city) [Default City]: Ciudad - **Valencia**
- Organization Name (eg, company) [Default Company Ltd]: Nombre de la compañia - **BBR_DEV**
- Organizational Unit Name (eg, section) []: Departamento - **BBR**
- Common Name (eg, your name or your server's hostname) []: Dominio - **misitio.com**
- Email Address []: Email - **webmaster@misitio.com**

Configuración de Apache:

- Una vez tenemos el certificado creado, configuraremos nuestro servidor.
  - Tanto si ya tenemos un VirtualHost en nuestro Apache como si no, debemos realizar los siguientes pasos para su correcta configuración:

- Abrimos el archivo de configuración de nuestro sitio y modificamos el archivo para que incluya estos datos. Si el archivo es nuevo, aparecerá vacío y como mínimo debemos añadir esta información.

`$ sudo nano /etc/apache2/sites-available/your_domain_or_ip.conf`

```
<VirtualHost *:443>

  ServerName misitio.com
  DocumentRoot "/var/www/html/misitio.com"

# Activacion SSL
        SSLEngine On
        SSLCertificateFile /etc/apache2/certs/apache-autofirmado.crt
        SSLCertificateKeyFile /etc/apache2/certs/apache-autofirmado.key

</VirtualHost>
```

Si no teníamos ya un VirtualHost configurado:

- Creamos la carpeta raíz del sitio con:

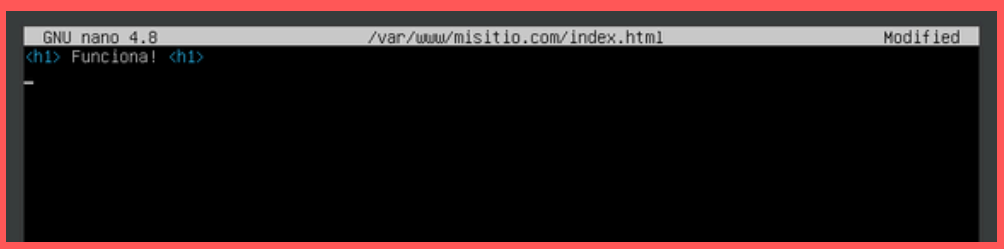
`$ sudo mkdir /var/www/your_domain_or_ip`

```
barragan@server-blas:/$ sudo mkdir /var/www/misitio.com
```

- Creamos un archivo HTML de bienvenida:

`$ sudo nano /var/www/your_domain_or_ip/index.html`

```
barragan@server-blas:/$ sudo nano /var/www/misitio.com/index.html
```



Configuración de Apache:

- Habilitamos el archivo de configuración correspondiente al sitio:

```
$ sudo a2ensite your_domain_or_ip.conf
```

```
barragan@server-blas:/$ sudo a2ensite misitio.com.conf
Site misitio.com already enabled
```

- Realizamos un test para comprobar que la configuración es correcta:

```
$ sudo apache2ctl configtest
```

```
barragan@server-blas:/$ sudo apache2ctl configtest
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0
.1.1. Set the 'ServerName' directive globally to suppress this message
Syntax OK
```

- Recargamos Apache para que acepte los cambios que hemos realizado:

```
$ sudo systemctl reload apache2
```

```
barragan@server-blas:/$ sudo systemctl reload apache2
```

Redirección HTTP a HTTPS:

- Si nuestro sitio sigue recibiendo solicitudes HTTP por el puerto 80, no responderá, ya que lo acabamos de configurar para responder en el 443 que usa HTTPS. Es buena practica, redireccionar esas solicitudes de forma automática con tal de forzar que todo el trafico se cifre o evitar conexiones sin respuesta.
- Vamos a modificar el archivo de configuración (.conf) de nuestro VirtualHost para ello:
- Abrimos el archivo con el siguiente comando y añadimos lo siguiente:

```
$ sudo nano /etc/apache2/sites-available/your_domain_or_ip.conf
```

```
  GNU nano 4.8                /etc/apache2/sites-available/misitio.com.conf

# Redireccionar http puerto 80 a https puerto 443 siempre
<VirtualHost *:80>
        ServerName misitio.com
        Redirect permanent / https://www.misitio.com/
</VirtualHost>
```

Redirección HTTP a HTTPS:

- Realizamos un test para comprobar que la configuración es correcta y recargamos Apache:

`$ sudo apache2ctl configtest`

```
barragan@server-blas:/$ sudo apache2ctl configtest
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0
.1.1. Set the 'ServerName' directive globally to suppress this message
Syntax OK
```

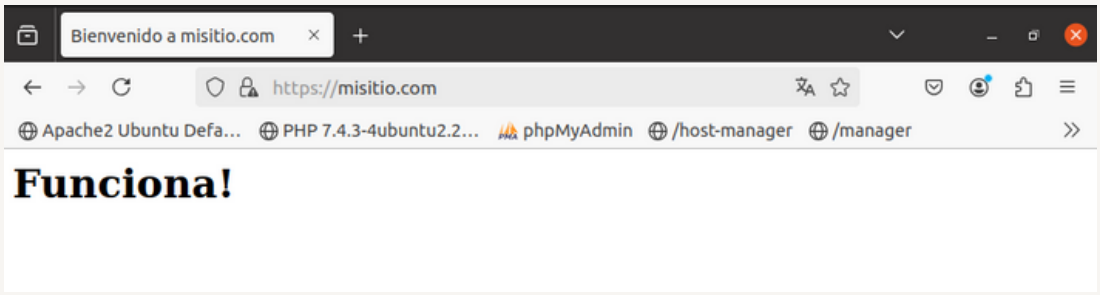
`$ sudo systemctl reload apache2`

```
barragan@server-blas:/$ sudo systemctl reload apache2
```

Prueba de resultados:

- Si hemos seguido todos los pasos correctamente, al abrir en cualquier navegador la URL https://your_domain_or_ip, después de crear una excepción de seguridad que nos indicará el propio navegador, podremos observar la pagina de bienvenida.

- Podemos observar la indicación de que nos encontramos en un sitio no seguro, debido a la característica auto firmada de nuestro certificado, ya que el navegador no lo considera de confianza, aunque nuestra información si esta cifrada



Fuentes

- **www.globalsign.com**

  GlobalSign by GMO
  ¿Qué es el SSL?.

- **www.digitalocean.com**

  Erin Glass y Brian Boucheron
  Cómo crear un certificado SSL autofirmado para Apache en Ubuntu 20.04.
  Published on August 20, 2020

- **es.wikipedia.org**

  Wagner, David y Bruce Schneier, «Analysis of the SSL 3.0 Protocol», The Second USENIX Workshop on Electronic Commerce Proceedings, USENIX Press, November 1996, pp. 29-40.
  https://es.wikipedia.org/wiki/Seguridad_de_la_capa_de_transporte