Práctica 2.6. Proxy inverso y balanceo de carga con SSL en NGINX

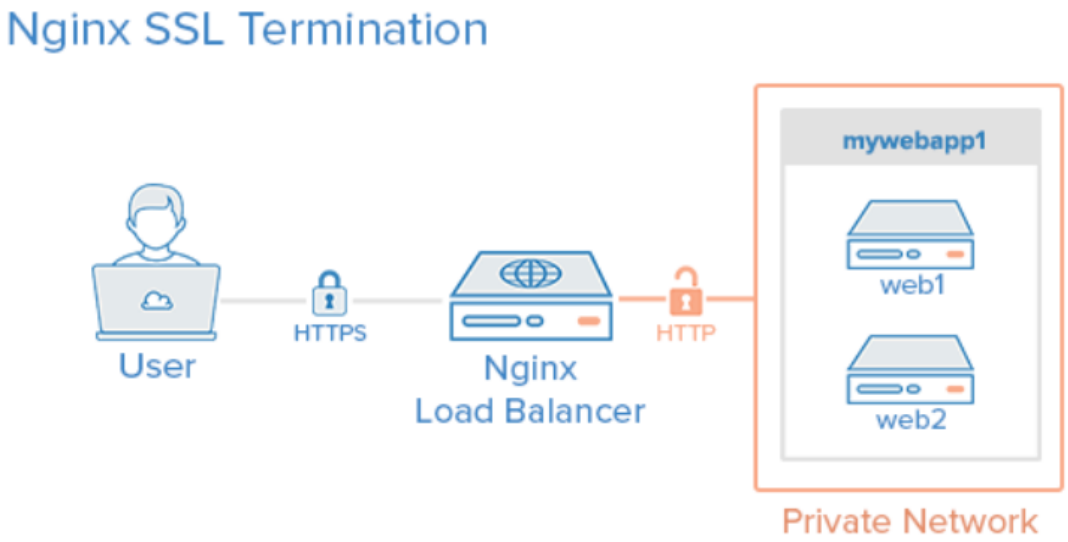
LA PRÁCTICA 2.5 DEBE ESTAR REALIZADA

1. Introducción

A partir de las prácticas anteriores hemos llegado a un escenario donde un proxy inverso actúa de intermediario entre dos servidores web Nginx, balanceando la carga entre ellos.

Ya dijimos que una importante función que podía tener un proxy inverso era realizar el cifrado y descifrado de SSL para utilizar HTTPS en los servidores web. De esta forma se aliviaba la carga de trabajo de los servidores web, ya que es una tarea que consume recursos.

En definitiva, tendríamos un esquema como este:



Podría llegarse a pensar que en términos de seguridad no es adecuado que el tráfico de red entre el balanceador de carga y los servidores web vaya sin cifrar (HTTP). Sin embargo, pensando en un caso real, la red privada y el proxy inverso/balanceador de carga, además de estar en la misma red privada, suelen estar administrados por las mismas personas de la misma empresa, por lo que no supone un peligro real que ese tráfico vaya sin cifrar.

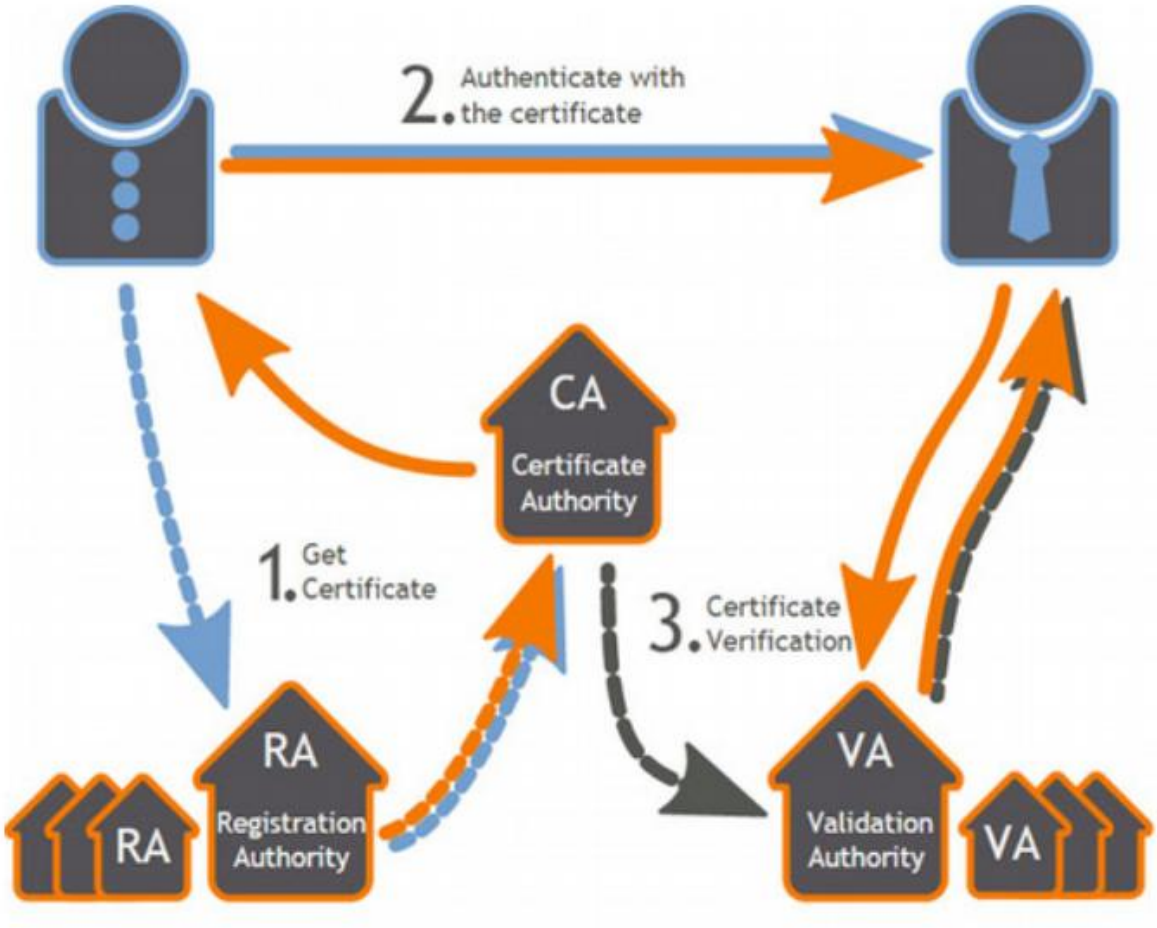
Podría cifrarse si fuera necesario, pero entonces pierde sentido que el proxy inverso se encargue del cifrado SSL para HTTPS, ya que haríamos el mismo trabajo dos veces.

Así las cosas, nos quedaremos con el esquema de la imagen de más arriba para la práctica.

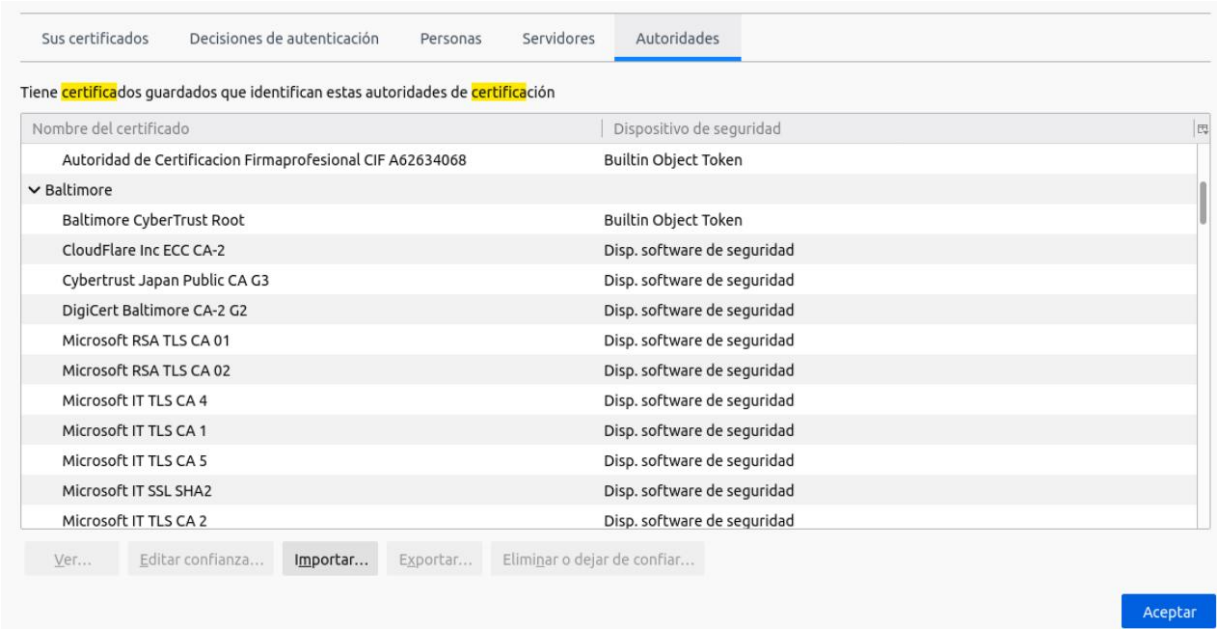
1.1 Certificados

HTTPS se basa en el uso de certificados digitales. Resumidamente, cuando entramos en una web vía HTTPS, ésta nos presenta un certificado digital para asegurar que es quién dice ser. ¿Cómo sabemos que ese certificado es válido? Debemos consultar a la Autoridad de Certificación (CA) que emitió ese certificado si es válido.

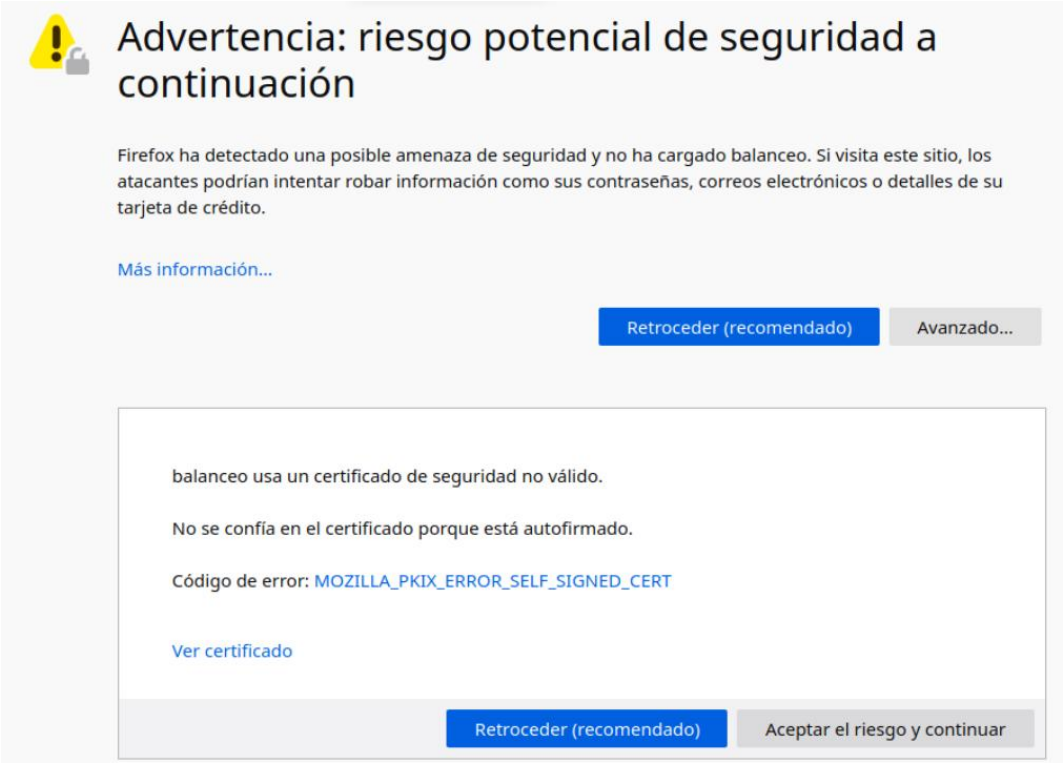
Las CA son entidades que emiten certificados y su funcionamiento se basa en la confianza. Confiamos en que los certificados emitidos y firmados por esas entidades son reales y funcionales.



Los navegadores web tienen precargadas las Autoridades de Certificación en las que confían por defecto a la hora de navegar por webs HTTPS:



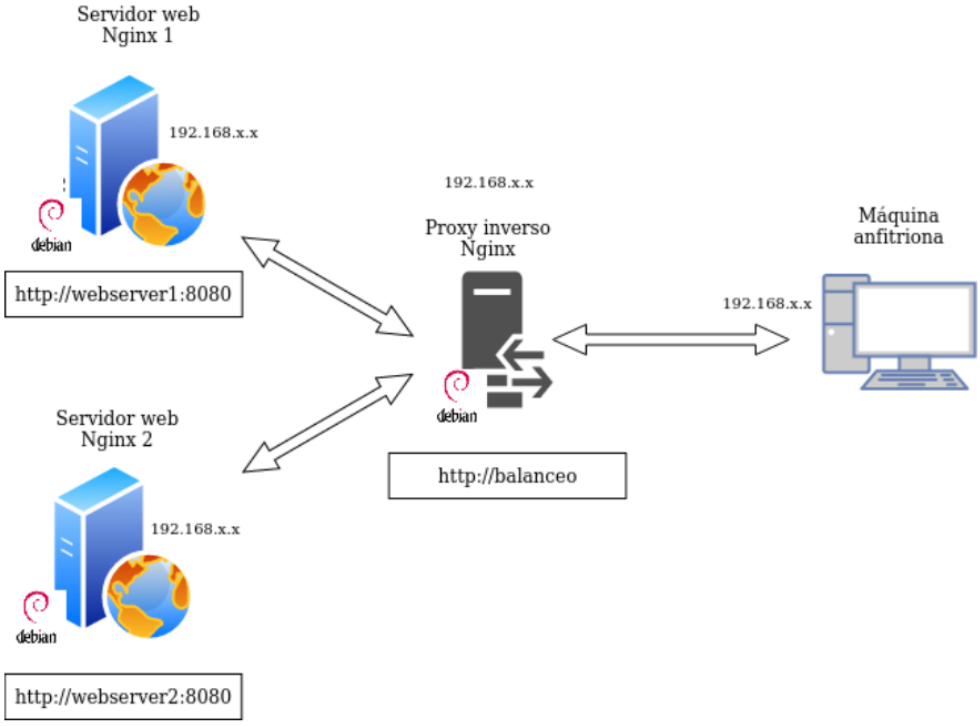
Si accedemos a una web cuyo certificado no haya sido emitido y firmado por una de estas entidades, nos saltará el famoso aviso:



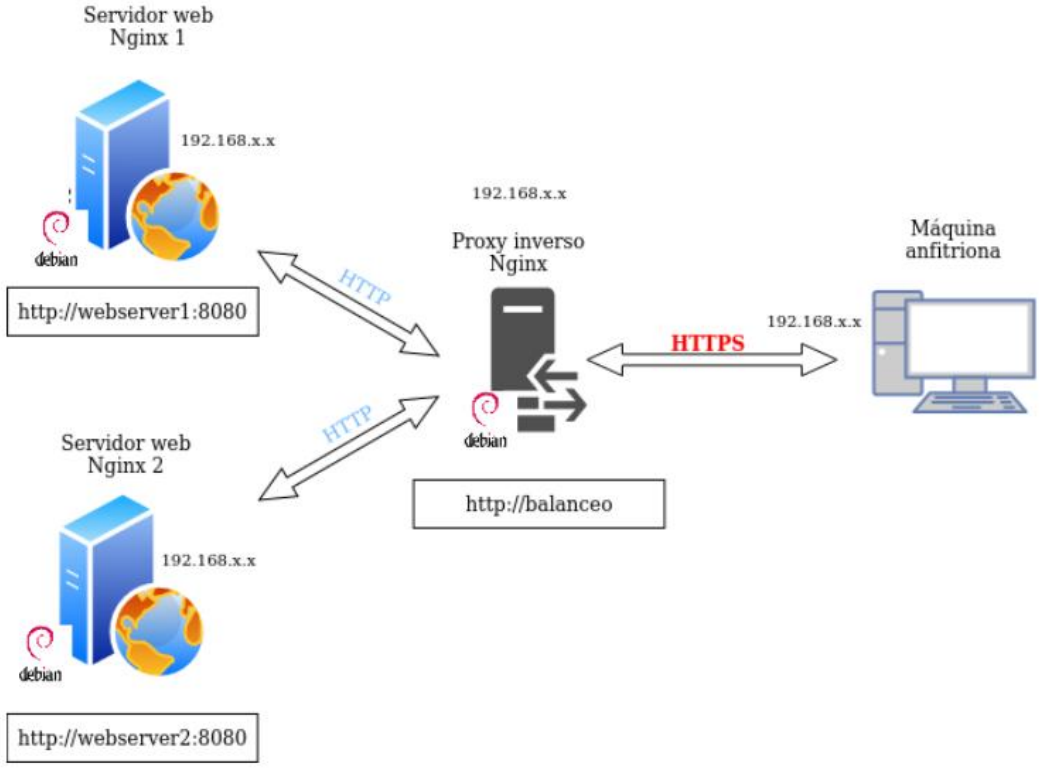
Ya que, si el certificado no ha sido emitido y firmado por una CA de confianza, puede que se trate de una web maliciosa que nos suponga un riesgo de seguridad, como bien dice el aviso.

2. Tarea

Partimos de la configuración exacta de la práctica anterior, que recordemos era esta:



Por lo que en esta práctica simplemente debemos añadir la configuración SSL para el cifrado en el Proxy Inverso:



2.1 Creación de certificado de *autofirmado*

Nosotros no utilizaremos certificados de ninguna CA de confianza, básicamente porque:

- Nuestros servicios no están publicados en Internet
- Estos certificados son de pago

Así pues, nosotros crearemos nuestros propios certificados y los firmaremos nosotros mismos como si fuéramos una CA auténtica para poder simular este escenario.

Esto provocará que cuando accedamos por HTTPS a nuestro sitio web por primera vez, nos salté el aviso de seguridad que se comentaba en la introducción.

En este caso no habrá peligro puesto que estamos 100% seguros que ese certificado lo hemos emitido nosotros para esta práctica, no hay dudas.

Veamos pues el proceso para generar los certificados y las claves asociadas a ellos (privada/pública). En primer lugar, debemos crear el siguiente directorio:

```
/etc/nginx/ssl
```

Podemos crear el certificado y las claves de forma simultánea con un único comando, donde:

- **openssl**: esta es la herramienta por la línea de comandos básica para crear y administrar certificados, claves y otros archivos OpenSSL.
- **req**: indica que se va a crear y procesar un certificado CSR (*Certificate Signing Request*) o un certificado X.509.
- **-x509**: Esto modifica aún más el subcomando anterior al decirle a la herramienta que queremos crear un certificado X.509 autofirmado, en lugar de generar una solicitud de firma de certificado, como sucedería normalmente.
- **-nodes**: Esto le dice a OpenSSL que omita la opción de asegurar nuestro certificado con contraseña. Necesitamos que Nginx pueda leer el archivo sin la intervención del usuario cuando se inicia el servidor. Una contraseña evitaría que esto sucediera ya que tendríamos que introducirla a mano después de cada reinicio.
- **-days 365**: esta opción establece el tiempo durante el cual el certificado se considerará válido. Lo configuramos para un año.
- **-newkey rsa: 2048**: Genera una nueva clave privada RSA de 2048 bits junto con el certificado. No creamos la clave necesaria para firmar el certificado en un paso anterior, por lo que debemos crearla junto con el certificado.
- **-keyout**: este parámetro le dice a OpenSSL dónde colocar el archivo de clave privada generado que estamos creando.
- **-out**: Esto le dice a OpenSSL dónde colocar el certificado que estamos creando.

El comando completo sería así:

```
sergio@Debian-DAW:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/server.key -out /etc/nginx/ssl/server.crt
.....+...+......++++++++++++++++++++++++++++++++++++++++*.+......+...+.....+...+............+...........+.+........+......+.+...+.........+..+
++++*...+...+..+...+....+..+......+............+......+.+..............++++++
.+...+........+...+.........+......+...+...........+.+...........+.......+..+...+......+.+......++++++++++++++++++++++++++++++++++++++++*.....
.........+.....+..............+....+.............+......+.+...+...+...++++++++++++++++++++++++++++++++++++++++*....+..+....+........+.+.....+
+.....+......+.........+..+.+.....+.......+..+......+.+.........+......+.................+..+....+..+.......+...+...+...........+.........+.
........+............+...+..++++++
-----
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.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Alicante
Locality Name (eg, city) []:Alicante
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IES Doctor Balmis
Organizational Unit Name (eg, section) []:2DAW - DEAW - SERGIO
Common Name (e.g. server FQDN or YOUR name) []:balanceo
Email Address []:sergio@mail.com
```

Os solicitará que introduzcáis una serie de parámetros. Debéis introducir los mismos parámetros que en la imagen excepto en el "*Organizational Unit Name*" que veis recuadrado en amarillo. Ahí deberéis poner *2DAW – DEAW – Vuestronombre.*

2.2 Configuración SSL en el proxy inverso

De la práctica anterior, dentro del directorio `/etc/nginx/sites-available` ya debéis tener el archivo de configuración llamado "*balanceo*". Es precisamente aquí donde realizaremos la configuración para que el acceso al sitio web se realice mediante SSL (HTTPS).

Dentro del bloque `server {…}` debéis cambiar el puerto de escucha (listen 80) por lo que veis en la imagen de abajo, añadiendo las siguientes líneas de configuración también, de tal forma que quede:

```
listen 443 ssl;
ssl_certificate /etc/nginx/ssl/server.crt;
ssl_certificate_key /etc/nginx/ssl/server.key;
ssl_protocols   TLSv1.3;
ssl_ciphers     ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AESA128:ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS;
server_name balanceo;
access_log /var/log/nginx/https_access.log;
```

Donde le estáis diciendo que:

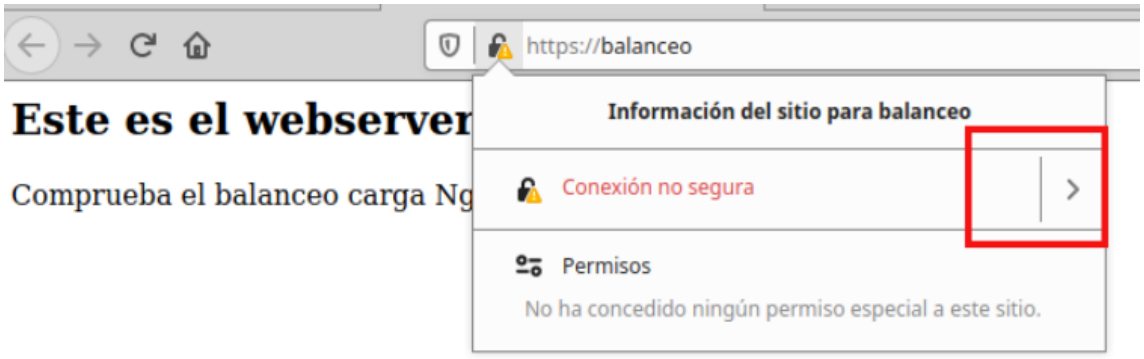
- Escuche en el puerto 443 → **Puerto por defecto de HTTPS**
- El directorio donde está el certificado que habéis generado anteriormente
- El directorio donde está la clave privada que habéis generado anteriormente
- Los protocolos y tipos de cifrados que se pueden utilizar → Estas son las versiones de protocolos y los tipos de cifrados considerados seguros hoy en día (hay muchos más pero no se consideran seguros actualmente)
- `server_name` ya lo teníais de la práctica anterior, no hace falta tocarlo

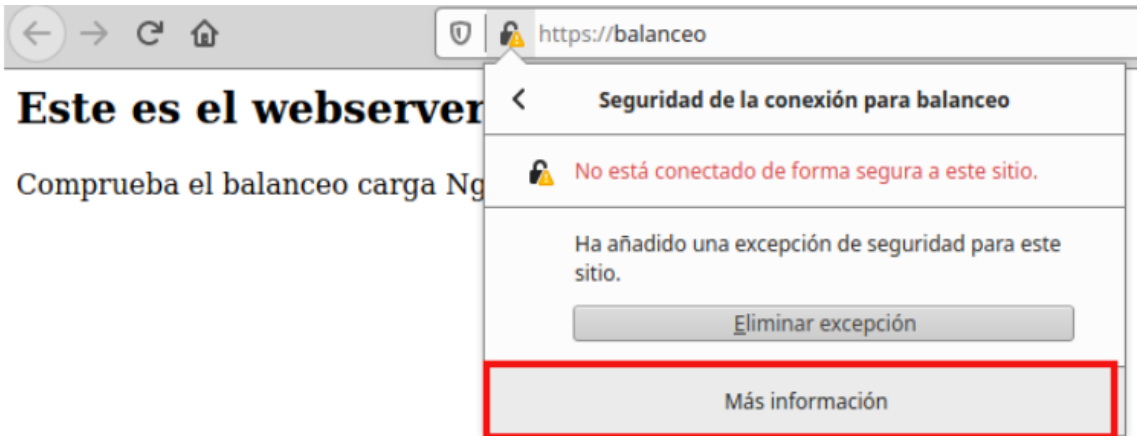
- El archivo donde se guardan los logs cambia de nombre, ahora será https_access.log

Recordad que, tras modificar cualquier configuración de un servicio, hay que reiniciar el servicio, en este caso Nginx.

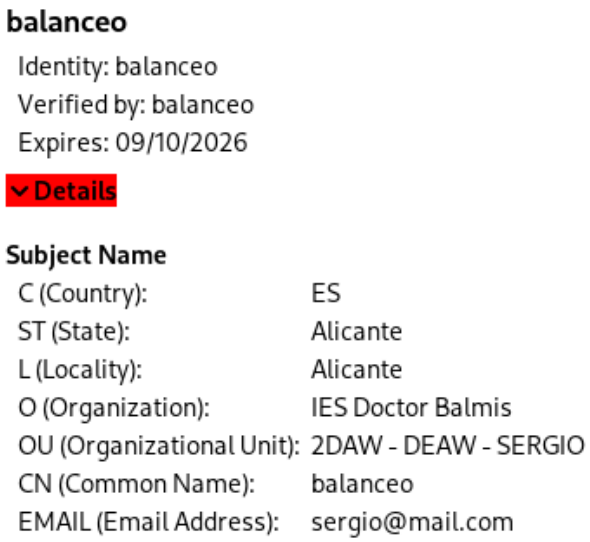
3. Comprobaciones

- Si accedéis ahora a *https://balanceo* os debería saltar un aviso de seguridad debido a que nuestro certificado es autofirmado, como comentábamos anteriormente.
- Si añadís una excepción podréis acceder al sitio web y recargando repetidamente la página con F5, veréis que el balanceo de carga se hace correctamente accediendo mediante HTTPS.
- Para comprobar que los datos del certificado son, efectivamente, los vuestros podéis comprobarlo así. Pulsando en el candado de la barra de búsqueda:





Y por último, ver certificado para ver los detalles:



Si ahora intentáis acceder a **http://balanceo (sin https)**, ¿deberíais poder acceder? ¡Comprobadlo!

3.1 Redirección forzosa a HTTPS

Para que, indistintamente de la forma por la que accedamos al sitio web balanceo, siempre se fuerce a utilizar HTTPS, necesitaremos una configuración adicional.

Necesitamos añadir un bloque "*server*" adicional y separado del otro, al archivo de configuración de "balanceo". Algo así:

```
server {
listen 80;
server_name balanceo;
access_log /var/log/nginx/http_access.log;
return 301 https://balanceo$request_uri;

}
```

Con esta configuración le estamos diciendo que:

- Escuche en el puerto 80 (por defecto para HTTP)
- Que el nombre al que responderá el servidor/sitio web es *balanceo*
- Que guarde los logs de este bloque en ese directorio y con ese nombre
- Cuando se recibe una petición con las dos condiciones anteriores, se devuelve un código HTTP 301:
    - **HTTP 301 *Moved Permanently***, es un código de estado de HTTP que indica que el host ha sido capaz de comunicarse con el servidor pero que el recurso solicitado ha sido movido a otra dirección permanentemente. Es muy importante configurar las redirecciones 301 en los sitios web y

para ello hay diferentes métodos y sintaxis para realizar la redirección 301.

- La **redirección 301** es un código o comando insertado por un Webmaster que permite redirigir a los usuarios y buscadores de un sitio web de un sitio a otro.

Es decir, lo que estamos haciendo es que cuando se reciba una petición HTTP (puerto 80) en `http://balanceo`, se **redirija** a `https://balanceo` (HTTPS)

ACTIVIDADES

1. Para esta actividad, es importante que en el vídeo mostréis la conexión HTTPS y que cuando hacéis la petición a HTTP os redirija a HTTPS.

2. Hemos configurado nuestro proxy inverso con todo lo que nos hace falta pero no nos funciona y da un error del tipo *This site can't provide a secure connection, ERR_SSL_PROTOCOL_ERROR.*

Dentro de nuestro server block tenemos esto:

```
server {

    listen 443;

    ssl_certificate /etc/nginx/ssl/enrico-berlinguer/server.crt;

    ssl_certificate_key /etc/nginx/ssl/enrico-berlinguer/server.key;

    ssl_protocols TLSv1.3;

    ssl_ciphers
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:ECDH+3DES:DH+3DES:RSA+
AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS;

    server_name enrico-berlinguer;

    access_log /var/log/nginx/https_access.log;


    location / {

        proxy_pass http://red-party;

        }

    }
```

3. Imaginad que intentamos acceder a nuestro sitio web HTTPS y nos encontramos con el siguiente error:

Investiga qué esta pasando y cómo se debe solucionar.