



UNIVERSIDAD DE GRANADA

Servidores Web de Altas Prestaciones
Práctica 3: Seguridad (Certificados SSL)

Miguel Torres Alonso

Mayo, 2024

Índice

1	Tareas Básicas - B1. Preparación del Entorno de Trabajo.	4
1.1	Crear directorios específicos para los archivos de configuración:	4
2	Tareas Básicas - B2. Creación de Certificados SSL.	4
2.1	Generación correcta del certificado SSL y la clave privada con OpenSSL siguiendo las especificaciones dadas.	4
2.1.1	Consulta realizada IA:	4
2.1.2	Respuesta de la IA:	4
2.1.3	Análisis detallado:	4
3	Tareas Básicas - B3. Configuración de Servidores Web Apache con SSL .	5
3.1	Configuración precisa del archivo de host virtual de Apache para atender peticiones HTTPS.	5
3.1.1	Consulta realizada IA:	5
3.1.2	Respuesta de la IA:	5
3.1.3	Análisis detallado:	6
3.2	Redacción adecuada del Dockerfile para los servidores Apache con soporte SSL.	6
3.2.1	Consulta realizada IA:	6
3.2.2	Respuesta de la IA:	6
3.2.3	Análisis detallado:	6
4	Tareas Básicas - B4. Configuración del Balanceador de Carga Nginx con SSL.	7
4.1	Elaboración correcta del Dockerfile y la configuración de Nginx para gestionar conexiones SSL.	7
4.1.1	Consulta realizada IA:	7
4.1.2	Respuesta de la IA:	7
4.1.3	Análisis detallado:	8
4.1.4	Consulta realizada IA:	8
4.1.5	Respuesta de la IA:	8
4.1.6	Análisis detallado:	8
5	Tareas Básicas - B5. Docker Compose para la Granja Web con SSL.	9
5.1	Desarrollo de un archivo docker-compose.yml funcional que despliegue todos los servicios y configuraciones definidos en la práctica.	9
5.1.1	Consulta realizada IA:	9
5.1.2	Respuesta de la IA:	9
5.1.3	Análisis detallado:	10
6	Tareas Básicas - B6. : Verificación y Pruebas del Escenario con SSL.	10
6.1	Ejecución efectiva del despliegue usando Docker Compose y verificación del correcto funcionamiento del entorno SSL.	10
7	Tareas Avanzadas - A1. Exploraciones Avanzadas de creación de certificados SSL.	11
7.1	Generar un certificado raíz (CA) y uno o más certificados intermedios (subCA) para entender cómo se construye una cadena de confianza.	11
7.1.1	Consulta realizada IA:	11
7.1.2	Respuesta de la IA:	11
7.1.3	Análisis detallado:	12
8	Tareas Avanzadas - A2. Optimización de la configuración SSL en los servidores web	13
8.1	Optimizar la configuración SSL en Apache para mejorar tanto la seguridad como el rendimiento de las conexiones seguras. Para ello, deshabilitar protocolos inseguros y cifrados débiles que pueden ser explotados por ataques.	13

8.1.1	Consulta realizada IA:	13
8.1.2	Respuesta de la IA:	14
8.1.3	Análisis detallado:	14
8.1.4	Consulta realizada IA:	14
8.1.5	Respuesta de la IA:	14
8.1.6	Análisis detallado:	14
8.1.7	Consulta realizada IA:	15
8.1.8	Respuesta de la IA:	15
8.1.9	Análisis detallado:	15
9	Tareas Avanzadas - A3. Configuración de Caché de Sesiones SSL y Tickets de Sesión en el balanceador.	16
9.1	Configurar Nginx para utilizar caché de sesiones SSL y tickets de sesión para mejorar la velocidad de las conexiones seguras repetidas, reduciendo el tiempo necesario para negociar la seguridad de la conexión.	16
9.1.1	Consulta realizada IA:	16
9.1.2	Respuesta de la IA:	16
9.1.3	Análisis detallado:	16
10	Tareas Avanzadas - A4. Optimización de conexiones HTTPS y cifrado en el balanceador.	17
10.1	Personalizar los protocolos SSL/TLS y suites de cifrado para equilibrar seguridad y rendimiento y activar HTTP/2 para mejorar la eficiencia de las conexiones HTTPS.	17
10.1.1	Consulta realizada IA:	17
10.1.2	Respuesta de la IA:	17
10.1.3	Análisis detallado:	17

1. Tareas Básicas - B1. Preparación del Entorno de Trabajo.

1.1. Crear directorios específicos para los archivos de configuración:

Seguimos los pasos del guión:

```
mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P3$ tree
.
├── P3-mtorres26-apache
├── P3-mtorres26-certificados
├── P3-mtorres26-nginx
└── web_mtorres26
    └── index.php

4 directories, 1 file
```

Figura 1: Estructura en árbol de los directorios de trabajo.

2. Tareas Básicas - B2. Creación de Certificados SSL.

2.1. Generación correcta del certificado SSL y la clave privada con OpenSSL siguiendo las especificaciones dadas.

2.1.1. Consulta realizada IA:

'Cómo puedo generar un certificado autofirmado, con validez de 1 año, encriptación RSA de 2048 bits y sin necesidad de passphrase. La clave privada se llamará 'claveprivada.key' y el certificado 'certificado.crt'

2.1.2. Respuesta de la IA:

```
$ openssl genpkey -algorithm RSA -out claveprivada.key -pkeyopt \
rsa_keygen_bits:2048
```

```
$ openssl req -new -x509 -key claveprivada.key -out certificado.crt -days 365
```

Listing 1: Comandos necesarios para la creación de la clave privada y el certificado autofirmado

2.1.3. Análisis detallado:

El código que nos proporciona la IA es correcto por lo que vamos a adaptarlo a nuestro usuario de la UGR: 'mtorres26' y a los nombres que nos piden en el guión. En la figura 2 podemos ver el proceso de generación desde un terminal.

[illegible]

Figura 2: Captura mostrando el proceso de generación de la clave privada y el certificado.

3. Tareas Básicas - B3. Configuración de Servidores Web Apache con SSL .

3.1. Configuración precisa del archivo de host virtual de Apache para atender peticiones HTTPS.

3.1.1. Consulta realizada IA:

'Genera un archivo de configuración de Apache que escuche en el puerto 443, incluya las rutas al certificado y la clave privada y SSLEngine on'

3.1.2. Respuesta de la IA:

```
<VirtualHost *:443>
    ServerName tu_dominio.com
    DocumentRoot /var/www/html

    SSLEngine on
    SSLCertificateFile /ruta/a/certificado.crt
    SSLCertificateKeyFile /ruta/a/claveprivada.key

    # Configuracion adicional SSL opcional
    # SSLProtocol, SSLCipherSuite, etc.

    # Otras directivas de configuracion de Apache
    # Como Alias, Directory, etc.
</VirtualHost>
```

Listing 2: Respuesta de la IA.

3.1.3. Análisis detallado:

La IA nos proporciona algo muy similar a lo indicado en el guión, siendo esto último lo que vamos a usar finalmente (adaptando nombres).

3.2. Redacción adecuada del Dockerfile para los servidores Apache con soporte SSL.

3.2.1. Consulta realizada IA:

'Modifica el siguiente Dockerfile para que habilite SSL para Apache y copie dos certificados a la configuración de Apache:'

```
FROM debian:buster-slim
RUN apt-get update && \
    apt-get install -y apache2 php libapache2-mod-php net-tools iputils-ping && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
EXPOSE 80
CMD ["apachectl", "-D", "FOREGROUND"]
```

Listing 3: Dockerfile de la P1

3.2.2. Respuesta de la IA:

```
FROM debian:buster-slim

# Instalar Apache, PHP y otras herramientas necesarias
RUN apt-get update && \
    apt-get install -y apache2 php libapache2-mod-php net-tools iputils-ping && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Habilitar el modulo SSL de Apache
RUN a2enmod ssl

# Copiar los certificados al contenedor
COPY certificado.crt /etc/ssl/certificado.crt
COPY claveprivada.key /etc/ssl/claveprivada.key

# Exponer los puertos 80 y 443
EXPOSE 80
EXPOSE 443

# Comando de inicio para Apache
CMD ["apachectl", "-D", "FOREGROUND"]
```

Listing 4: Respuesta de la IA.

3.2.3. Análisis detallado:

Lo proporcionado por la IA es similar a lo indicado en el guión, siendo el del guión más completo. Por tanto vamos a hacer una combinación de lo que teníamos de la P1 con lo del guión y vamos a obviar la copia del certificado y la clave privada al directorio correspondiente de Apache, pues da error al estar en diferentes

carpetas e igualmente vamos a montar volúmenes en el *'docker-compose.yml'* posteriormente.

4. Tareas Básicas - B4. Configuración del Balanceador de Carga Nginx con SSL.

4.1. Elaboración correcta del Dockerfile y la configuración de Nginx para gestionar conexiones SSL.

Primero vamos a consultar a la IA generativa acerca del archivo de configuración de Nginx.

4.1.1. Consulta realizada IA:

Le vamos a pedir que modifique el archivo de configuración de la práctica anterior para adaptarlo a las exigencias de esta práctica.

'Modifica el siguiente código para que escuche en el puerto 443 ssl y especifique la ruta a un certificado y una clave privada' (código de la P2)

4.1.2. Respuesta de la IA:

```
events {
    worker_connections 1024; # Numero maximo de conexiones por trabajador
}
http {
    upstream backend_mtorres26 {
        server 192.168.10.2;
        server 192.168.10.3;
        server 192.168.10.4;
        server 192.168.10.5;
        server 192.168.10.6;
        server 192.168.10.7;
        server 192.168.10.8;
        server 192.168.10.9;
    }
    server {
        listen 443 ssl;
        server_name nginx_mtorres26;

        ssl_certificate /ruta/a/tu/certificado.crt;
        ssl_certificate_key /ruta/a/tu/claveprivada.key;

        access_log /var/log/nginx/nginx_mtorres26.access.log;
        error_log /var/log/nginx/nginx_mtorres26.error.log;

        location / {
            proxy_pass http://backend_mtorres26;
            proxy_set_header Cookie $http_cookie;
            proxy_hide_header Set-Cookie;
        }

        location /estadisticas_mtorres26 {
```

```
        stub_status on;
    }
}
}
```

Listing 5: Respuesta de la IA.

4.1.3. Análisis detallado:

Efectivamente lo proporcionado en el guión es idéntico a lo respondido por la IA, por tanto lo añadimos a nuestro archivo de configuración sin olvidar que debemos mantener otro apartado *'server'* exactamente igual que el de la P2 configurado para HTTP.

Ahora vamos a proceder con el Dockerfile para Nginx.

4.1.4. Consulta realizada IA:

'Modifica el siguiente código para que cree un directorio 'ssl' en la carpeta de nginx, copie certificado y clave privada al directorio 'ssl', incluya un archivo de configuración a la ruta adecuada y exponga el puerto 443 (código del Dockerfile de la P2)'

4.1.5. Respuesta de la IA:

```
# Usa la imagen oficial de Nginx como base
FROM nginx:latest

# Crea un directorio 'ssl' en la carpeta de nginx
RUN mkdir /etc/nginx/ssl

# Copia el certificado y la clave privada al directorio 'ssl'
COPY certificado.crt /etc/nginx/ssl/
COPY claveprivada.key /etc/nginx/ssl/

# Copia el archivo de configuracion
COPY nginx.conf /etc/nginx/nginx.conf

# Exponer el puerto 443
EXPOSE 443

# Si es necesario, cambiar el comando predeterminado de Nginx
CMD ["nginx", "-g", "daemon off;"]
```

Listing 6: Respuesta de la IA.

4.1.6. Análisis detallado:

Es básicamente lo mismo que se nos indica en el guión, por tanto añadimos la respuesta a nuestro código adaptando los nombres de los ficheros y carpetas correctamente. Exponemos también el puerto 80 al igual que en el Dockerfile de Apache y obviemos también la copia de certificado y clave privada al directorio de Nginx correspondiente sin preocuparnos porque montaremos volúmenes para el correcto funcionamiento.

5. Tareas Básicas - B5. Docker Compose para la Granja Web con SSL.

5.1. Desarrollo de un archivo docker-compose.yml funcional que despliegue todos los servicios y configuraciones definidos en la práctica.

Vamos a pedirle a la IA que modifique el *docker-compose.yml* de la P2 con los requisitos que se nos pide en el guión.

5.1.1. Consulta realizada IA:

'Modifica el siguiente código para montar el directorio local certificados_mtorres en la carpeta /etc/apache2/ssl de cada servicio web y montar el archivo mtorres-nginx-ssl.conf en el servicio de nginx en /etc/nginx/nginx.conf (código del docker-compose de la P2)'

5.1.2. Respuesta de la IA:

```
services:
  web1:
    image: mtorres26-apache-image:p2
    container_name: web1
    build:
      context: .
      dockerfile: DockerfileApache_mtorres26
    volumes:
      - ./web_mtorres26:/var/www/html
      - ./certificados_mtorres:/etc/apache2/ssl
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2

# Los servicios web restantes tienen la misma configuración que web1,
# pero con diferentes direcciones IP

balanceador-nginx:
  image: mtorres26-nginx-image:p2
  container_name: balanceador-nginx
  build:
    context: .
    dockerfile: DockerfileNginx
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf
    - ./certificados_mtorres/mtorres-nginx-ssl.conf:/etc/nginx/nginx.conf
  networks:
    red_web:
      ipv4_address: 192.168.10.50
  ports:
    - "80:80"
  depends_on:
    - web1
```

- web2
- web3
- web4
- web5
- web6
- web7
- web8

Listing 7: Respuesta de la IA.

5.1.3. Análisis detallado:

Es parcialmente incorrecto, le faltan los siguientes elementos:

- En los contenedores Apache se debe montar el archivo de configuración en la carpeta de sitios de Apache.
- Apertura del puerto 443 en el balanceador Nginx.
- En el balanceador Nginx, el montaje del volumen de los certificados debe de montarse en el directorio del contenedor creado con tal propósito.

6. Tareas Básicas - B6. : Verificación y Pruebas del Escenario con SSL.

6.1. Ejecución efectiva del despliegue usando Docker Compose y verificación del correcto funcionamiento del entorno SSL.

Desplegamos los servicios y comprobamos que se están ejecutando (figura 3)

```
mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P3$ docker compose up -d
[+] Running 11/11
 ✓ Network p3_red_web      Created           0.1s
 ✓ Network p3_red_servicios Created           0.1s
 ✓ Container web1          Started          0.1s
 ✓ Container web7          Started          0.1s
 ✓ Container web3          Started          0.1s
 ✓ Container web5          Started          0.1s
 ✓ Container web8          Started          0.1s
 ✓ Container web6          Started          0.1s
 ✓ Container web4          Started          0.1s
 ✓ Container web2          Started          0.1s
 ✓ Container balanceador-nginx-ssl Started          0.0s

mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P3$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
ec981481399a   mtorres26-nginx-image:p3           "/docker-entrypoint..." 8 seconds ago  Up 5 seconds  0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443
5a811368b208   mtorres26-balanceador-nginx-ssl    "apachectl -D FOREGR..." 8 seconds ago  Up 6 seconds  80/tcp, 443/tcp
cd25d7c82229   mtorres26-apache-image:p3         "apachectl -D FOREGR..." 8 seconds ago  Up 5 seconds  80/tcp, 443/tcp
249c0d137fb0   mtorres26-apache-image:p3         "apachectl -D FOREGR..." 8 seconds ago  Up 6 seconds  80/tcp, 443/tcp
ebaf40ae1f38   mtorres26-apache-image:p3         "apachectl -D FOREGR..." 8 seconds ago  Up 5 seconds  80/tcp, 443/tcp
029e7443f260   mtorres26-apache-image:p3         "apachectl -D FOREGR..." 8 seconds ago  Up 6 seconds  80/tcp, 443/tcp
8f4c932bc697   mtorres26-apache-image:p3         "apachectl -D FOREGR..." 8 seconds ago  Up 5 seconds  80/tcp, 443/tcp
f6e6bd32f5f9   mtorres26-apache-image:p3         "apachectl -D FOREGR..." 8 seconds ago  Up 6 seconds  80/tcp, 443/tcp
d58d7e32e929   mtorres26-apache-image:p3         "apachectl -D FOREGR..." 8 seconds ago  Up 6 seconds  80/tcp, 443/tcp
```

Figura 3: Ejecución de los servicios

Escribimos en el navegador *'https://localhost'* y vemos nuestro *index.php* y que al recargar la página el balanceador realiza correctamente su trabajo. Si clickamos en el apartado del navegador que muestra información sobre el sitio web, observamos lo que se muestra en la figura 4



Figura 4: Información que nos muestra el navegador sobre nuestro certificado.

7. Tareas Avanzadas - A1. Exploraciones Avanzadas de creación de certificados SSL.

7.1. Generar un certificado raíz (CA) y uno o más certificados intermedios (sub-CA) para entender cómo se construye una cadena de confianza.

7.1.1. Consulta realizada IA:

'Genera clave privada de la CA y úsala para crear un certificado autofirmado que servirá como CA raíz. Segundo, crea una Autoridad Certificadora Intermedia (subCA) similar a la CA, crea una clave privada para la subCA y utilízala para generar una solicitud de certificado (CSR) de la subCA que será firmada por la CA raíz, estableciendo una cadena de confianza.'

7.1.2. Respuesta de la IA:

```
# Paso 1: Generar una clave privada para la CA raiz
openssl genrsa -out rootCA.key 2048

# Paso 2: Generar una solicitud de certificado (CSR) para la CA raiz
openssl req -new -key rootCA.key -out rootCA.csr

# Paso 3: Firmar el CSR con la clave privada de la CA raiz para generar el \
certificado raiz autofirmado
openssl x509 -req -days 365 -in rootCA.csr -signkey rootCA.key -out rootCA.crt

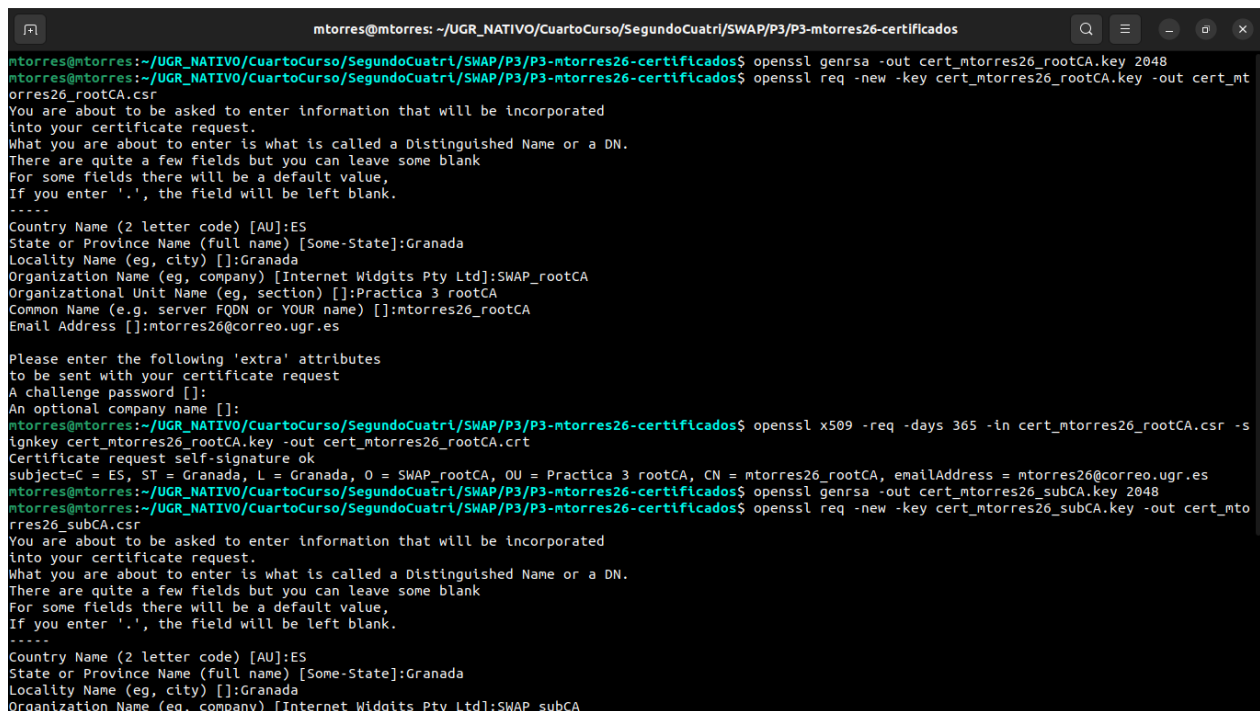
# Paso 4: Generar una clave privada para la subCA
openssl genrsa -out subCA.key 2048

# Paso 5: Generar una solicitud de certificado (CSR) para la subCA
openssl req -new -key subCA.key -out subCA.csr
```

```
# Paso 6: Firmar el CSR de la subCA con el certificado raiz y su clave privada \
para generar el certificado de subCA
openssl x509 -req -days 365 -in subCA.csr -CA rootCA.crt -CAkey rootCA.key \
-CAcreateserial -out subCA.crt
```

7.1.3. Análisis detallado:

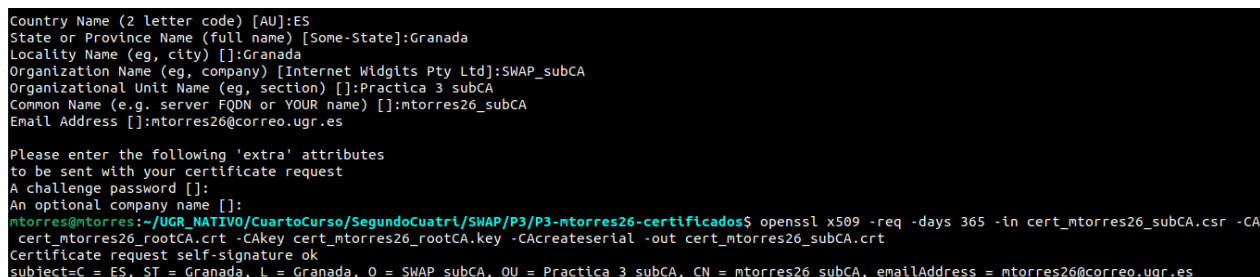
Prácticamente lo mismo que en el apartado 2.1.2, comandos muy similares pero adaptados a lo que se nos pide. Vamos a introducir los comandos y añadirlos en el *docker-compose.yml* para probarlo posteriormente¹. También debemos indicar en los archivos de configuración de Apache y Nginx la ruta a estos certificados.



```
mtorres@mtorres: ~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P3/P3-mtorres26-certificados
mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P3/P3-mtorres26-certificados$ openssl genrsa -out cert_mtorres26_rootCA.key 2048
mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P3/P3-mtorres26-certificados$ openssl req -new -key cert_mtorres26_rootCA.key -out cert_mtorres26_rootCA.csr
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]:Granada
Locality Name (eg, city) []:Granada
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SWAP_rootCA
Organizational Unit Name (eg, section) []:Practica 3 rootCA
Common Name (e.g. server FQDN or YOUR name) []:mtorres26_rootCA
Email Address []:mtorres26@correo.ugr.es

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P3/P3-mtorres26-certificados$ openssl x509 -req -days 365 -in cert_mtorres26_rootCA.csr -s
ignkey cert_mtorres26_rootCA.key -out cert_mtorres26_rootCA.crt
Certificate request self-signature ok
subject=C = ES, ST = Granada, L = Granada, O = SWAP_rootCA, OU = Practica 3 rootCA, CN = mtorres26_rootCA, emailAddress = mtorres26@correo.ugr.es
mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P3/P3-mtorres26-certificados$ openssl genrsa -out cert_mtorres26_subCA.key 2048
mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P3/P3-mtorres26-certificados$ openssl req -new -key cert_mtorres26_subCA.key -out cert_mtorres26_subCA.csr
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]:Granada
Locality Name (eg, city) []:Granada
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SWAP_subCA
```

Figura 5: Salida de terminal de los comandos introducidos.



```
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Granada
Locality Name (eg, city) []:Granada
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SWAP_subCA
Organizational Unit Name (eg, section) []:Practica 3 subCA
Common Name (e.g. server FQDN or YOUR name) []:mtorres26_subCA
Email Address []:mtorres26@correo.ugr.es

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P3/P3-mtorres26-certificados$ openssl x509 -req -days 365 -in cert_mtorres26_subCA.csr -CA
cert_mtorres26_rootCA.crt -CAkey cert_mtorres26_rootCA.key -CAcreateserial -out cert_mtorres26_subCA.crt
Certificate request self-signature ok
subject=C = ES, ST = Granada, L = Granada, O = SWAP_subCA, OU = Practica 3 subCA, CN = mtorres26_subCA, emailAddress = mtorres26@correo.ugr.es
```

Figura 6: Salida de terminal de los comandos introducidos.

Accedemos a <https://localhost> y al consultar los certificados podemos comprobar el correcto funcionamiento como se muestra en la figura 7.

¹En los archivos que se entregarán aparecerán comentadas todas las líneas de los distintos archivos que se deben de modificar para este apartado.



Figura 7: Visor de certificados del navegador.

8. Tareas Avanzadas - A2. Optimización de la configuración SSL en los servidores web

8.1. Optimizar la configuración SSL en Apache para mejorar tanto la seguridad como el rendimiento de las conexiones seguras. Para ello, deshabilitar protocolos inseguros y cifrados débiles que pueden ser explotados por ataques.

Primero necesitamos saber qué protocolos son seguros.

8.1.1. Consulta realizada IA:

¿Qué protocolos TLS son seguros??

8.1.2. Respuesta de la IA:

'TLS 1.2: Considerado seguro y ampliamente utilizado en la actualidad. Ofrece una buena seguridad y es compatible con una amplia gama de aplicaciones y dispositivos.

TLS 1.3: La última versión del protocolo TLS en el momento de esta respuesta. Introduce mejoras significativas en seguridad y rendimiento con respecto a TLS 1.2. Se recomienda su uso siempre que sea posible.

TLS 1.1 y versiones anteriores: Consideradas inseguras debido a vulnerabilidades conocidas, como POODLE (Padding Oracle On Downgraded Legacy Encryption). Se recomienda deshabilitar estas versiones y utilizar TLS 1.2 o superior siempre que sea posible.'

8.1.3. Análisis detallado:

Efectivamente está desaconsejado el uso TLSv1.1 y versiones anteriores como podemos comprobar en la página web del Instituto Nacional de Ciberseguridad.

Ahora, preguntamos a la IA acerca de los algoritmos de cifrado más seguros:

8.1.4. Consulta realizada IA:

'¿Qué algoritmos de cifrado son seguros?'

8.1.5. Respuesta de la IA:

AES (Advanced Encryption Standard): AES es un algoritmo de cifrado simétrico que ha sido ampliamente adoptado y es utilizado por muchas organizaciones y estándares de seguridad.

ChaCha20: ChaCha20 es un algoritmo de cifrado de flujo diseñado para ser rápido y seguro en una amplia gama de plataformas.

RSA (Rivest-Shamir-Adleman): RSA es un algoritmo de cifrado asimétrico que se utiliza para cifrar y firmar digitalmente datos.

ECC (Elliptic Curve Cryptography): ECC es una técnica de cifrado asimétrico que utiliza curvas elípticas para generar claves criptográficas. Ofrece una seguridad comparable a RSA pero con claves más cortas.

SHA-2 (Secure Hash Algorithm 2): SHA-2 es una familia de funciones hash criptográficas, que incluyen SHA-256, SHA-384 y SHA-512.

8.1.6. Análisis detallado:

La IA nos proporciona ciertos algoritmos seguros. En el guión se nos propone MD5, RC4 y 3DES, sin embargo, según las páginas:

<https://telefonicatech.com/blog/la-criptografia-insegura-que-deberias-dejar-de-usar> y <https://ciberseguridad.comillas.edu/funciones-de-hash-en-ciberseguridad-un-analisis-detallado/#:~:text=MD5,y%20se%20desaconseja%20su%20uso>. están obsoletos y se desaconseja su uso por problemas de seguridad y por tanto los vamos a prohibir en nuestro archivo. Por tanto, vamos a intentar usar los siguientes:

- AES.
- RSA.
- ChaCha.
- SHA-256.

8.1.7. Consulta realizada IA:

'Modifica el siguiente código para que utilice protocolos TLSv1.2 o posteriores y algoritmos de cifrado fuertes como AES, RSA o SHA-256'

```
<VirtualHost *:443>
    DocumentRoot /var/www/html
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/certificado_mtorres26.crt
    SSLCertificateKeyFile /etc/apache2/ssl/certificado_mtorres26.key
</VirtualHost>
```

Listing 8: Código que queremos que la IA modifique

8.1.8. Respuesta de la IA:

```
<VirtualHost *:443>
    DocumentRoot /var/www/html
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/certificado_mtorres26.crt
    SSLCertificateKeyFile /etc/apache2/ssl/certificado_mtorres26.key

    # Configuración para utilizar solo protocolos TLSv1.2 o posteriores
    SSLProtocol +TLSv1.2

    # Configuración para algoritmos de cifrado fuertes
    SSLCipherSuite HIGH:!aNULL:!MD5:!AES128:!AES256:!RSA:!SHA1:!SHA256
</VirtualHost>
```

Listing 9: Respuesta de la IA

8.1.9. Análisis detallado:

La IA falla un poco o no entiende bien la pregunta, por tanto metemos a mano los protocolos y algoritmos mencionados en los apartados anteriores. En la documentación oficial de Apache podemos observar recomendaciones para cifrados fuertes. Con *SSLHonorCipherOrder on* hacemos que se sigan las preferencias de cifrado del servidor en vez de las del cliente.

```
<VirtualHost *:443>
    DocumentRoot /var/www/html
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/certificado_mtorres26.crt
    SSLCertificateKeyFile /etc/apache2/ssl/certificado_mtorres26.key

    # Rutas para los certificados de la cadena de confianza
    # SSLCertificateFile /etc/apache2/ssl/cert_mtorres26_subCA.crt
    # SSLCertificateKeyFile /etc/apache2/ssl/cert_mtorres26_subCA.key

    # Configuración para utilizar solo protocolos TLSv1.2 o posteriores
    SSLProtocol -all +TLSv1.2 +TLSv1.3

    # Configuración para algoritmos de cifrado fuertes
    SSLCipherSuite HIGH:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:\
    ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:!aNULL:!RC4:!MD5
```

```
    SSLHonorCipherOrder on
</VirtualHost>
```

Listing 10: Archivo de configuración con protocolos seguros.

9. Tareas Avanzadas - A3. Configuración de Caché de Sesiones SSL y Tickets de Sesión en el balanceador.

9.1. Configurar Nginx para utilizar caché de sesiones SSL y tickets de sesión para mejorar la velocidad de las conexiones seguras repetidas, reduciendo el tiempo necesario para negociar la seguridad de la conexión.

9.1.1. Consulta realizada IA:

'Modifica el siguiente código para configurar Nginx para utilizar caché de sesiones SSL y tickets de sesión para mejorar la velocidad de las conexiones seguras repetidas (archivo de config. de Nginx)'

9.1.2. Respuesta de la IA:

```
# Configuración global para cache de sesiones SSL y tickets de sesión
    ssl_session_cache shared:SSL:10m;
    ssl_session_tickets on;
```

Listing 11: Respuesta de la IA

9.1.3. Análisis detallado:

Efectivamente con dichos parámetros podemos mejorar la velocidad de las conexiones. Si miramos la documentación de Nginx nos ofrece también habilitar conexiones 'keep-alive' y un 'timeout' de 10 minutos, por lo que los añadimos.

```
server{
    # puerto 443 para https
    listen 443 ssl;
    ssl_certificate /etc/nginx/ssl/certificado_mtorres26.crt;
    ssl_certificate_key /etc/nginx/ssl/certificado_mtorres26.key;

    # Rutas para los certificados de la cadena de confianza
    # ssl_certificate /etc/nginx/ssl/cert_mtorres26_subCA.crt;
    # ssl_certificate_key /etc/nginx/ssl/cert_mtorres26_subCA.key;

    location / {
        proxy_pass http://backend_mtorres26;
        proxy_set_header Cookie $http_cookie;
        proxy_hide_header Set-Cookie;
    }

    # Configuraciones para mejorar velocidad de conexiones repetidas
    keepalive_timeout 70;
    ssl_session_cache shared:SSL:10m;
    ssl_session_tickets on;
    ssl_session_timeout 10m;
```



```
}
```

Listing 12: Apartado server del archivo de config. de Nginx.

10. Tareas Avanzadas - A4. Optimización de conexiones HTTPS y cifrado en el balanceador.

10.1. Personalizar los protocolos SSL/TLS y suites de cifrado para equilibrar seguridad y rendimiento y activar HTTP/2 para mejorar la eficiencia de las conexiones HTTPS.

10.1.1. Consulta realizada IA:

‘¿Como puedo personalizar los protocolos SSL/TLS y suites de cifrado para equilibrar seguridad y rendimiento y activar HTTP/2 para mejorar la eficiencia de las conexiones HTTPS en un archivo de configuración de Nginx?’

10.1.2. Respuesta de la IA:

```
server {  
    listen 443 ssl http2; # Habilitar HTTP/2 en el puerto 443  
    server_name ejemplo.com;  
  
    ssl_certificate /ruta/al/certificado.crt;  
    ssl_certificate_key /ruta/a/claveprivada.key;  
  
    # Configuración de protocolos SSL/TLS  
    ssl_protocols TLSv1.2 TLSv1.3; # Utilizar solo TLSv1.2 y TLSv1.3  
    ssl_prefer_server_ciphers on; # Preferir los cifrados del servidor  
  
    # Configuración de suites de cifrado  
    ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:\nECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:\nECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256';\n    # Cifrado preferido para compatibilidad y seguridad  
  
    # Configuración adicional de seguridad  
    ssl_session_cache shared:SSL:10m; # Cache de sesiones SSL  
    ssl_session_tickets on; # Tickets de sesión SSL  
  
    # Otras configuraciones del servidor...  
}
```

Listing 13: Respuesta de la IA.

10.1.3. Análisis detallado:

Añadir la directiva `http2` donde escucha el servidor (*`listen 443 ssl http2;`*) está obsoleto, hay que añadirla en una orden aparte. Vamos a usar los suites de cifrado que nos propone la IA porque son algoritmos de cifrado seguros. También vamos a añadir la directiva *`ssl_prefer_server_ciphers on;`* para que los cifrados del servidor tengan prioridad y vamos a prohibir el uso de algoritmos de cifrado obsoletos como los comentados en el apartado 8.1.6.

```
server{  
    # puerto 443 para https  
    listen 443 ssl;  
    ssl_certificate /etc/nginx/ssl/certificado_mtorres26.crt;  
    ssl_certificate_key /etc/nginx/ssl/certificado_mtorres26.key;  
  
    # Rutas para los certificados de la cadena de confianza  
    # ssl_certificate /etc/nginx/ssl/cert_mtorres26_subCA.crt;  
    # ssl_certificate_key /etc/nginx/ssl/cert_mtorres26_subCA.key;  
  
    location / {  
        proxy_pass http://backend_mtorres26;  
        proxy_set_header Cookie $http_cookie;  
        proxy_hide_header Set-Cookie;  
    }  
  
    # Configuraciones para mejorar velocidad de conexiones repetidas. Tarea A3  
    keepalive_timeout 70;  
    ssl_session_cache shared:SSL:10m;  
    ssl_session_tickets on;  
    ssl_session_timeout 10m;  
  
    # Configuraciones para aumentar seguridad. Tarea A4  
    http2 on;  
    ssl_protocols TLSv1.2 TLSv1.3; # Protocolos seguros a usar  
    ssl_prefer_server_ciphers on; # Preferir los cifrados del servidor  
    ssl_ciphers HIGH:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-EC  
    # Suites de cifrado seguras  
}
```

Listing 14: Archivo de config. de Nginx con seguridad añadida.

Procedemos a comprobar con las herramientas del navegador si la versión es HTTP/2 (figura 8).

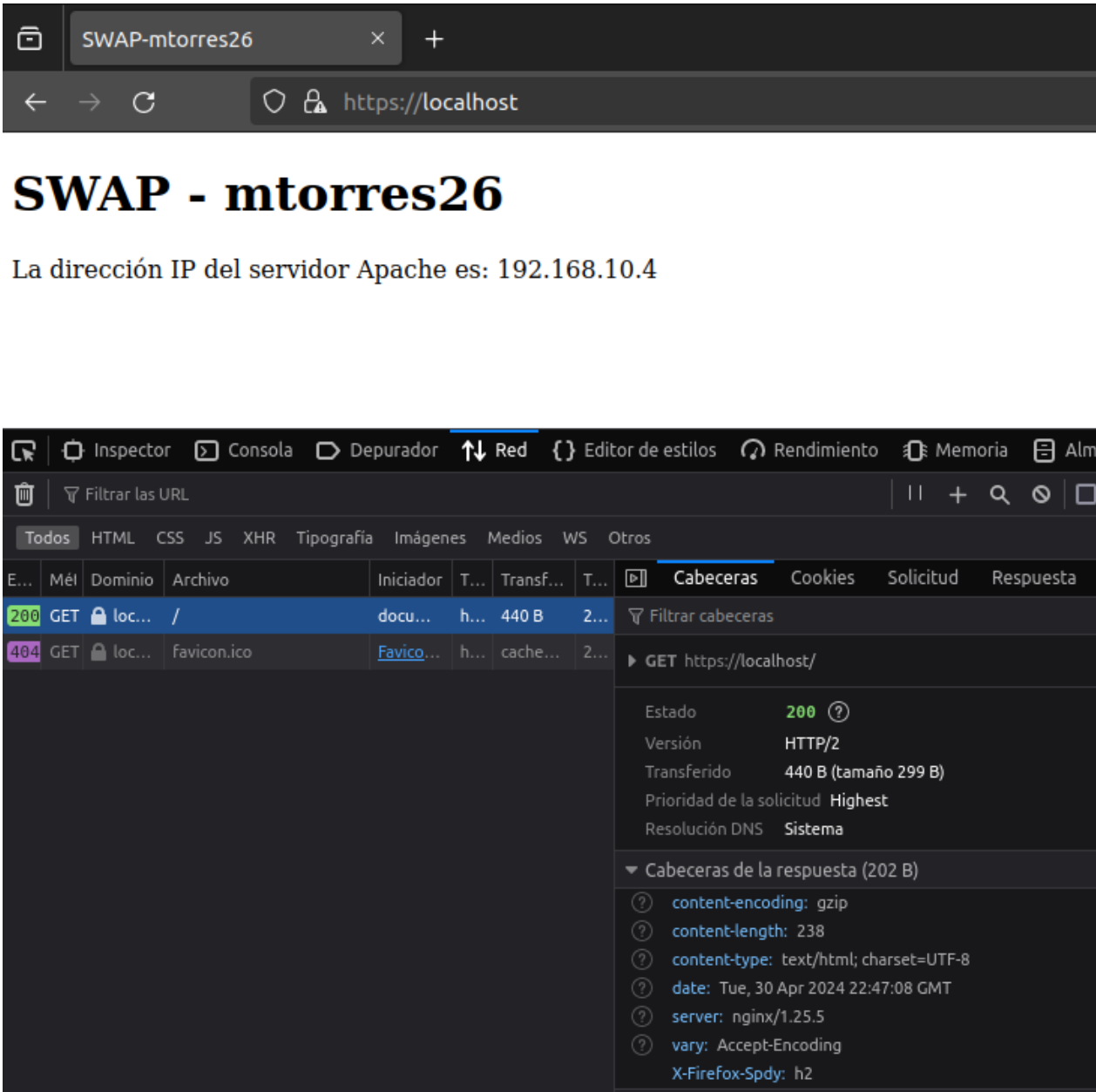


Figura 8: Inspeccionamiento de la versión de HTTP2.