Servicios web seguros: certificados y análisis HTTPS

Contenido

[Recursos útiles 1](#_Toc202510716)

[Objetivo de la práctica 2](#_Toc202510717)

[Guion de la práctica 2](#_Toc202510718)

[Servicio web SSL 2](#_Toc202510719)

[Creación de un certificado autofirmado con openssl 2](#_Toc202510720)

[Fichero default-ssl.conf 4](#_Toc202510721)

[Stack Portainer 5](#_Toc202510722)

[Certificado para OWASP ZAP 7](#_Toc202510723)

[Entregables de la práctica 10](#_Toc202510724)

[Continuará… 11](#_Toc202510725)

# Recursos útiles

Recuerde el directorio de recursos compartidos disponible en:

<https://nascorformacion0-my.sharepoint.com/:f:/g/personal/juan_pinuela_docente_nascorformacion_com/EtGSBITnqFpJp6-XpG6REOQBFycP_G9zgcB89jxPymjV2Q?e=IBIEER>

Para esta práctica disponemos de una carpeta con nombre “**login-ssl”** donde podemos encontrar todos los ficheros utilizados y creados en este guion.

# Objetivo de la práctica

Esta práctica tiene como objetivo introducir al estudiante en los conceptos fundamentales de **seguridad en servicios web**, centrándose en la **capa de transporte segura (HTTPS)** y el uso de **certificados digitales** en un entorno de contenedores Docker. Su inclusión en el módulo de seguridad del curso de Big Data es clave porque permite la transición hacia entornos en la nube (como Azure), donde la gestión de certificados y la seguridad en las comunicaciones es crítica.

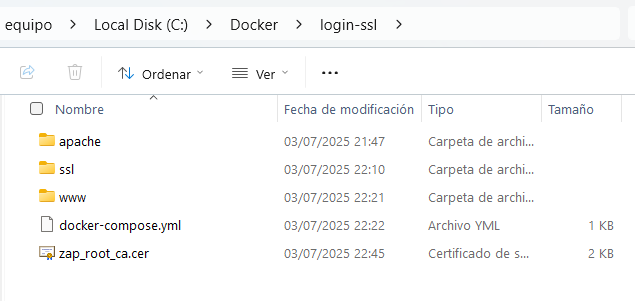
# Guion de la práctica

### Servicio web SSL

Tenemos como objetivo habilitar HTTPS con un certificado autofirmado en un contenedor PHP+Apache, creado desde Portainer.

En este caso vamos a ir un paso más y ya en Portainer hemos usado la creación de contenedor y de imágenes sobre las que luego crear contenedores ahora vamos directamente a la opción que nos queda de crear *stacks*.

La estructura de carpetas a crear será la siguiente:



Los ficheros utilizados para crear este guión en el equipo del docente están en la carpeta compartida pero los certificados por ejemplo deben ser creados por el estudiante.

### Creación de un certificado autofirmado con openssl

Necesitamos instalar por ejemplo GitBash (Git para Windows) desde: <<https://git-scm.com/download/win>> que viene equipado con openssl.

En lo que he visto en Mac openssl ya viene instalado. Si se ha instalado bien si arrancamos un terminal GIT en cualquier lado y hacemos:

openssl version

Debemos obtener la versión instalada.

Texto

El contenido generado por IA puede ser incorrecto.

La sentencia que utilizaremos es esta:

"C:\Program Files\Git\usr\bin\openssl.exe" req -x509 -nodes -days 365 -subj "/C=ES/ST=Madrid/L=Madrid/O=BigData/CN=localhost" -newkey rsa:2048 -keyout ssl\selfsigned.key -out ssl\selfsigned.crt

|  |  |
| --- | --- |
| Parte | Explicación |
| openssl req | Lanza una solicitud de certificado (Certificate Signing Request) |
| -x509 | Genera un certificado X.509 directamente (no solo una CSR). X509 es un estándar internacional que define el formato de los certificados digitales.  Una CSR (Certificate Signing Request) es un archivo que contiene tu información de identidad (como nombre de dominio, organización, país, etc.) y tu clave pública. Este archivo:  Se envía a una autoridad certificadora (CA) para que te emita un certificado digital válido.  No es un certificado en sí, solo una solicitud para obtener uno.  Cuando usas la opción -x509 con openssl req, estás diciendo:  "No quiero solo una CSR, quiero que me generes directamente un certificado X.509 válido para pruebas". |
| -nodes | No cifra la clave privada (útil para desarrollo) |
| -days 365 | El certificado será válido por 365 días |
| -subj "..." | Define directamente los campos del certificado: país, ciudad, organización, CN |
| -newkey rsa:2048 | Genera una nueva clave RSA de 2048 bits |
| -keyout ssl/selfsigned.key | Archivo de salida de la clave privada |
| -out ssl/selfsigned.crt | Archivo de salida del certificado |

Una vez ejecutado el comando tendremos dos ficheros asociados al certificado:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

### Fichero default-ssl.conf

<IfModule mod\_ssl.c>

<VirtualHost \_default\_:443>

DocumentRoot /var/www/html

SSLEngine on

SSLCertificateFile /etc/ssl/certs/selfsigned.crt

SSLCertificateKeyFile /etc/ssl/private/selfsigned.key

<Directory /var/www/html>

Options Indexes FollowSymLinks

AllowOverride All

Require all granted

</Directory>

</VirtualHost>

</IfModule>

|  |  |
| --- | --- |
| Línea / Bloque | Descripción |
| <IfModule mod\_ssl.c> | Solo se aplica si el módulo SSL está habilitado en Apache (mod\_ssl). |
| <VirtualHost \_default\_:443> | Define un sitio HTTPS por defecto que escucha en el puerto 443. |
| DocumentRoot /var/www/html | Carpeta donde se encuentran los archivos del sitio web (HTML, PHP, etc.). |
| SSLEngine on | Activa el motor SSL para habilitar HTTPS. |
| SSLCertificateFile /etc/ssl/certs/selfsigned.crt | Ruta del certificado público autofirmado. |
| SSLCertificateKeyFile /etc/ssl/private/selfsigned.key | Ruta de la clave privada asociada al certificado. |
| <Directory /var/www/html> | Configura el acceso y permisos para el directorio raíz del sitio. |
| Options Indexes FollowSymLinks | Permite listar archivos y seguir enlaces simbólicos. |
| AllowOverride All | Permite que .htaccess sobrescriba esta configuración si existe. |
| Require all granted | Permite acceso a todos los usuarios (sin restricciones). |
| </VirtualHost> | Finaliza la definición del sitio seguro. |
| </IfModule> | Cierra la condición de aplicar solo si mod\_ssl está habilitado. |

### Stack Portainer

Una vez tengamos:

* Certificados en C:\Docker\login-ssl\ssl
* Configuración SSL (default-ssl.conf) en C:\Docker\login-ssl\apache
* Archivos PHP en C:\Docker\login-ssl\www (presumo)

Entra en Portainer y ve a Stacks > Add Stack

* Nombre del stack: login-ssl
* En Web editor, pega este docker-compose.yml (fichero disponible en la carpeta de recursos)

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Tenemos el siguiente significado para este fichero YML.

|  |  |
| --- | --- |
| Línea / Bloque | Descripción |
| version: '3.8' | Versión del esquema Docker Compose, compatible con Portainer y Docker modernos. |
| services: | Inicio de la definición de servicios (contenedores) en el stack. |
| php-apache-ssl: | Nombre del servicio que define el contenedor con Apache y PHP. |
| image: php:8.2-apache | Imagen base desde Docker Hub con Apache y PHP 8.2 preinstalado. |
| container\_name: php-apache-ssl | Nombre visible del contenedor dentro de Docker / Portainer. |
| ports: | Define el mapeo de puertos entre host y contenedor. |
| - "8443:443" | Expone el puerto HTTPS (443) del contenedor como 8443 en el equipo local. |
| volumes: | Monta carpetas del host dentro del contenedor para persistencia y configuración. |
| C:/Docker/login-ssl/www:/var/www/html | Código fuente PHP visible por Apache. |
| C:/Docker/login-ssl/apache:/etc/apache2/sites-available | Configuración de sitios de Apache, incluyendo default-ssl.conf. |
| C:/Docker/login-ssl/ssl:/etc/ssl/private | Certificado y clave SSL generados localmente. |
| command: | Sobrescribe el comando por defecto del contenedor al iniciarse. |
| sh -c "... | Ejecuta varios comandos encadenados dentro del contenedor: |
| a2enmod ssl | Activa el módulo SSL en Apache. |
| a2ensite default-ssl | Habilita el sitio definido en default-ssl.conf. |
| a2dissite 000-default | Desactiva el sitio HTTP por defecto (000-default.conf). |
| apache2-foreground | Lanza Apache en primer plano para mantener el contenedor activo. |

Podemos probar que todo va bien accediendo a <https://localhost:8443>, . Se nos avisará que es un certificado autofirmado; aceptamos y ya tenemos nuestro servicio seguro.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

### Certificado para OWASP ZAP

Cuando usas ZAP como proxy man-in-the-middle, intercepta el tráfico HTTPS y presenta su propio certificado, tu navegador lo bloqueará (porque es autofirmado) a menos que confíes en el certificado raíz de ZAP.

1. Iniciar ZAP con interfaz gráfica

2. Exportar el certificado de ZAP

* En ZAP, ir a Tools > Options > Dynamic SSL Certificates (en inglés) o Certificados SSL dinámicos en español.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Lo guardamos con nombre zap\_root\_ca.cer en la misma ruta del nuestro proyecto “C:\Docker\login-ssl”,

Como vemos, al comenzar con ---- BEGIN CERTIFICATE ---- se trata de un certificado PEM (Privacy Enhanced Mail), que contiene los datos codificados en Base64 y admite extensiones de fichero .pem, .crt, .cert y .key (para clave privada).

|  |  |  |
| --- | --- | --- |
| Extensión | Contenido típico | Cuándo se usa |
| .pem | Puede contener certificado, clave, o ambos | Usado en sistemas Unix/Linux y OpenSSL |
| .crt | Certificado público | Se usa con servidores Apache, NGINX, Postfix |
| .cer | Certificado público (igual que .crt) | En Windows o navegadores |
| .key | Clave privada | Debe mantenerse secreta; nunca compartir |
| .chain.pem | Certificados intermedios o cadena de confianza | En configuraciones SSL completas (como Let's Encrypt) |

La siguiente tabla compara este formato PEM con otros tipos de certificados:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Formato | Codificación | Extensión típica | Usos comunes | Legible por humanos |
| PEM | Base64 con encabezados (-----BEGIN) | .pem, .crt, .cer, .key | Apache, NGINX, OpenSSL, navegadores (Firefox) | ✅ Sí |
| DER | Binario (no texto) | .der, .cer | Windows, Java, certificados root | ❌ No |
| PFX / PKCS#12 | Binario con certificado + clave privada y cadena | .pfx, .p12 | Windows, IIS, importar en navegador, Java keystores | ❌ No |

3. Importar el certificado en el navegador

En Firefox, ir a la sección Privacidad y seguridad. Buscar el apartado Certificados y pulsar Ver certificados, Ir a la pestaña Autoridades, pulsar Importar y seleccionar el archivo exportado (zap\_cert.cer).

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Es importante marcar la casilla “Confiar en esta CA para identificar sitios web”

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Por su parte, para usarlo en en Chrome o Edge (Windows) tenemos que realizar los siguientes pasos.

* Pulsar Win + R, escribir certmgr.msc y presionar Enter
* Ir a la carpeta Entidades de certificación raíz de confianza > Certificados
* Hacer clic derecho en la lista > Todas las tareas > Importar
* Seleccionar el archivo zap\_cert.cer
* Completar el asistente y aceptar

4. Configurar el navegador para que use ZAP como proxy que ya lo tenemos realizado de la práctica anterior con la precaución de usar las mismas opciones de proxy para tráfico HTTPS

5. Probar acceso, con la dirección ya mencionada de <https://localhost:8443>

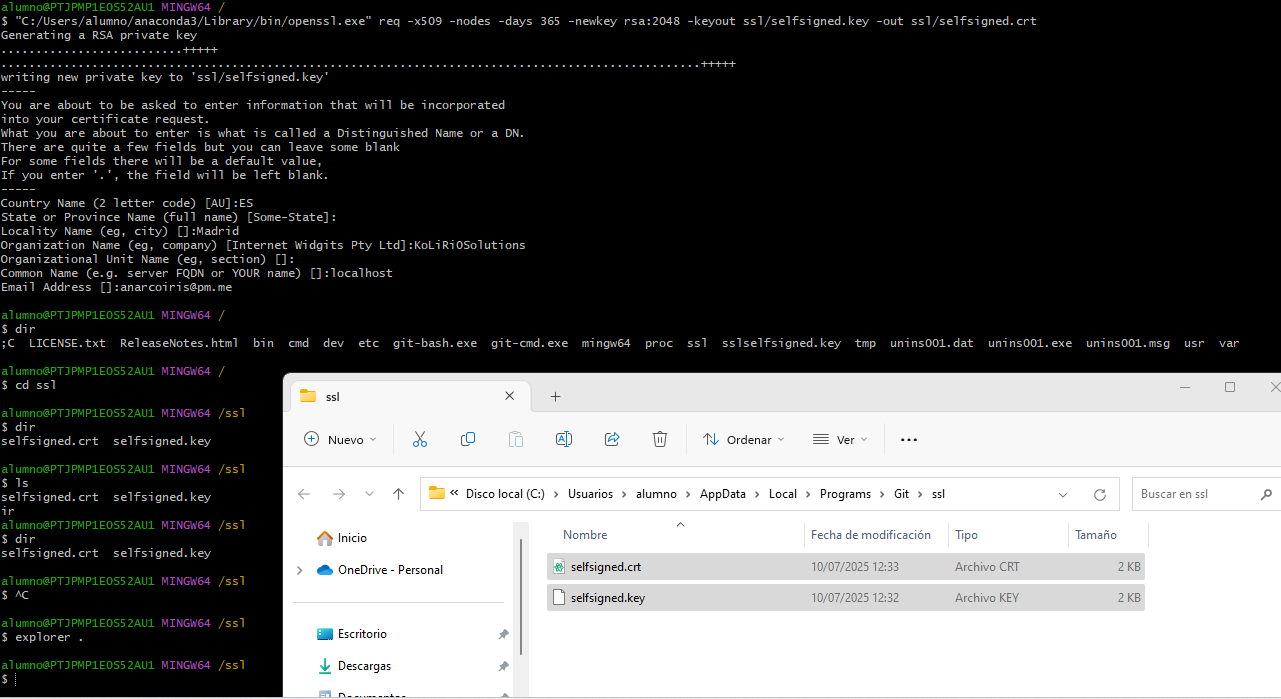
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

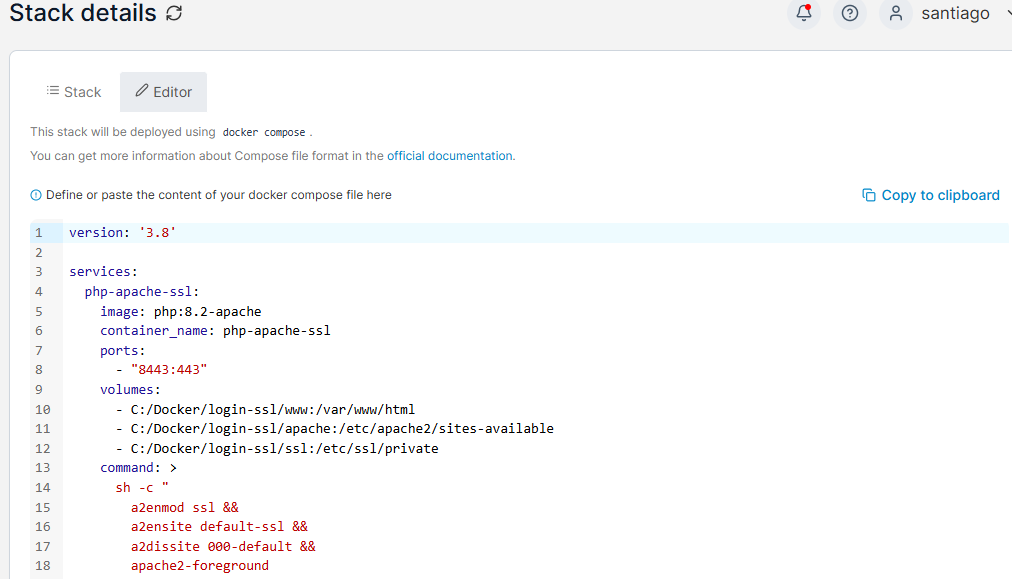
# Entregables de la práctica

Se debe entregar un documento que incluya capturas o evidencias de:

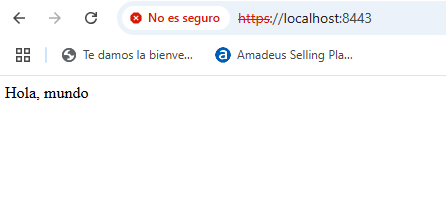
* La creación del certificado autofirmado en local (comando openssl o script).



* La estructura del stack en Portainer y sus rutas de volumen.



* El acceso funcional a https://localhost:8443 con advertencia de certificado aceptada.



* La exportación del certificado raíz de OWASP ZAP.
* La importación de ese certificado en el navegador (Firefox o Chrome).
* OWASP ZAP interceptando correctamente tráfico cifrado.

Opcionalmente: cualquier error o problema detectado durante el proceso, y cómo se resolvió.

# Continuará…

En módulo ya orientados a *cloud*, podríamos seguir trabajando con esta misma arquitectura basada en contenedores PHP + MySQL con SSL, pero llevándola a un entorno gestionado dentro de Azure. Allí tendriamos que:

* Subir imágenes Docker personalizadas a Azure Container Registry (ACR).
* Desplegar servicios web seguros con certificados gestionados desde Azure App Services o Azure Container Apps.
* Utilizar servicios como Azure Key Vault para el almacenamiento seguro de certificados y claves.

En Azure, además de servicios gestionados como App Services y Container Apps, hay varias opciones más modernas y escalables para desplegar servicios web como sobre todo son las **funciones serverless**. Comparando opciones en Azure tendremos las siguientes tecnologías:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Servicio Azure | Tipo | Uso principal | Certificados SSL | Escalado | Ideal para... |
| App Service | PaaS | Web apps y APIs tradicionales (PHP, Node) | Gestionados por Azure | Automático | Aplicaciones monolíticas o APIs REST estándar |
| Container Apps | Serverless de contenedores | Microservicios en Docker | Gestionados por Azure | Automático | APIs en contenedor, eventos, microservicios |
| Azure Functions | Serverless | Funciones bajo demanda (event-driven) | Gestionados por Azure | Dinámico (a cero) | Tareas ligeras, APIs sin servidor |
| AKS (Kubernetes) | IaaS gestionado | Contenedores orquestados a gran escala | Requiere configuración | Manual o automático | Empresas con arquitectura compleja |
| Static Web Apps + Functions | Serverless híbrido | Frontend estático + backend reactivo | Incluidos automáticamente | Automático | Web modernas con APIs ligeras |
| API Management | Pasarela/API Gateway | Gestión de APIs y seguridad avanzada | No aplica directamente | N/A | Control de versiones, autenticación y cuotas |

Un ejemplo de escenario específicamente relacionado con Big Data sería:

Un contenedor que ejecuta una API de login (como la que hemos desarrollado) desplegado en Azure Container Apps, se conecta con una Azure Function que, tras autenticar al usuario, procesa y normaliza datos recibidos, enviándolos a Azure Data Lake o una base de datos analítica.

|  |
| --- |
| Todo ello estará protegido mediante certificados gestionados en Azure Key Vault, y supervisado por Azure API Management, que controla el acceso, aplica políticas de seguridad y permite la trazabilidad del uso de las APIs.  Este enfoque refleja una **arquitectura realista, escalable y segura**, en línea con los desafíos actuales en entornos Big Data empresariales. |