Práctica de análisis de seguridad en servicios web con PHP y Docker

Contenido

[Guion de la práctica 2](#_Toc202423818)

[Instalando Docker en Windows 2](#_Toc202423819)

[Instalación de Portainer 2](#_Toc202423820)

[MySQL 3](#_Toc202423821)

[Apache con PHP 5](#_Toc202423822)

[Base de datos de autenticación y servicio PHP básico 8](#_Toc202423823)

[ZAP Browser 11](#_Toc202423824)

[Entregables de la práctica 14](#_Toc202423825)

[Opciones de ampliación para participantes que finalicen antes 14](#_Toc202423826)

[1. Implementación de la API con frameworks modernos 14](#_Toc202423827)

[2. Uso avanzado de OWASP ZAP siguiendo recursos oficiales 15](#_Toc202423828)

# 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>

Donde disponemos de una carpeta con npmbre “**Bases de datos”** donde podemos encontrar, por ejemplo la base de datos de autenticación utilizada en esta práctica en el fichero sample\_sql.sql: Además se dispone de distintos ejemplos de código PHP en la carpeta “Practica API Login”

# Guion de la práctica

## Instalando Docker en Windows

Instalamos Docker sobre Windows a partir de:

<https://docs.docker.com/desktop/install/windows-install/>

Como la instalación requiero tiempo y reiniciar el equipo, en este vídeo ya lo tenemos instalado destacando que, al menos por el momento, no hemos creado cuenta ni opciones específicas durante la instalación.

En lugar de utilizar el interfaz de aplicación, vamos a probar primero utilizando la línea de comandos. Para ello en la carpeta de trabajo que deseemos crear como, en este ejemplo “C:\Docker” abrimos un terminal desde el que ejecutamos:

>> docker version

Texto

El contenido generado por IA puede ser incorrecto.

La imagen de prueba habitual de funcionamiento de Docker es la disponible en:

<https://hub.docker.com/\_/hello-world>

Que como en la documentación que podemos ver en la página web, podemos ejecutar simplemente haciendo:

>> docker pull hello-world

>> docker run hello-world

## Instalación de Portainer

Portainer es una forma cómoda de gestionar entornos de contenedores distribuidos Docker. Se trata de un docker en sí y para su instalación vamos a:

<https://hub.docker.com/extensions/portainer/portainer-docker-extension>

Podemos instalarlo a través de Docker Desktop, haciendo *click* en las opciones que nos ofrece esta página web y que enlaza con la aplicación ya instalada. Sin embargo, en nuestro caso recurrimos a la línea de comandos, haciendo:

>> docker volume create portainer\_data

>> docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer\_data:/data portainer/portainer-ce:latest

Y podemos acceder a Portainer en: https://localhost:9443.Es importante en este punto configurar el entorno con la IP de nuestro ordenador local de forma que podamos acceder a cualquiera de los contenedores con la IP local 127.0.0.1.

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

## MySQL

El uso más básico de Docker consiste en crear un contenedor directamente desde Docker Hub. Por ejemplo, podemos buscar imágenes de MySQL y seguir las indicaciones disponibles en < <https://hub.docker.com/_/mysql/>>. Usamos la imagen denominada “mysql” y publicamos el puerto 3307 del host (nuestro equipo) con el puerto estándar de MySQL como es el 3306 que usará el contenedor.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Y donde además vamos a definir como ejemplo de uso de variable del entorno de este Docker el password del usuario root en:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Una vez configuradas estas opciones pulsamos “Deploy Container” y si todo es correcto podemos conectarnos a la base de datos de nuestro contenedor usando Heidi o cualquier cliente MySQL como se muestra en la siguiente imagen.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

En el caso de DBeaver es fundamental seleccionar la siguiente opción sobre uso de credenciales públicas “AllowPublicKeyRetrieval”.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

## Apache con PHP

El segundo ejemplo en orden de complejidad es usar un Dockerfile. Portainer no tiene una opción directa para subir un `Dockerfile`, puedes crear una imagen desde un `Dockerfile` y luego usar esa imagen para crear un contenedor. En el menú o sección “Images” de Portainer, pulsamos la opción “Build a New Image” usaremos en este ejemplo de Instalación de Apache PHP junto con las extensiones necesarias para conectividad con la base de datos MySQL la siguiente sintaxis.

FROM php:8.2-apache

RUN docker-php-ext-install mysqli pdo pdo\_mysql

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Una vez finalizado el proceso de construcción de la imagen “personalizada” de Apache+PHP con las extensiones requeridas, si todo ha ido bien aparecerá la nueva imagen junto con las anteriormente instaladas:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Y ya podemos crear nuestro contenedor utilizando esta imagen con:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

En este caso como opción adicional vamos a mapear el volumen típico de Apache /var/www/html con un directorio de nuestro equipo según la siguiente imagen:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En la ruta “C:/Docker/www/” podemos por ejemplo ubica un fichero info.txt con el texto “Hola” que podemos acceder una vez deplegado el contenedor con:

<http://localhost:8080/info.txt>

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

El contenido generado por IA puede ser incorrecto.

SI probaramos nuestro servidor con la siguiente configuración de conexión a la base de datos no tendremos éxito:

$host = '127.0.0.1:3307'; // Cambia esto si tu base de datos está en otro servidor

$db = 'mysql’; // Reemplaza con tu base de datos

…

Si analizamos el *container* de MySQL tenemos:

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Es necesario usar el siguiente fichero test\_database.php

<?php

// Configuración de la base de datos

$host = '172.17.0.3:3306'; // Cambia esto si tu base de datos está en otro servidor

$db = 'mysql'; // Reemplaza con tu base de datos

$user = 'root'; // Reemplaza con tu usuario

$pass = 'root'; // Reemplaza con tu contraseña

$charset = 'utf8mb4';

// Configuración de la conexión PDO

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";

$options = [

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION,

PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_ASSOC,

PDO::ATTR\_EMULATE\_PREPARES => false,

];

try {

// Crear una nueva conexión PDO

$pdo = new PDO($dsn, $user, $pass, $options);

// Ejecutar la consulta

$stmt = $pdo->query("SELECT NOW() AS current\_datetime");

// Obtener y mostrar el resultado

$row = $stmt->fetch();

echo "La fecha y hora actual es: " . $row['current\_datetime'];

} catch (\PDOException $e) {

// Manejo de errores

throw new \PDOException($e->getMessage(), (int)$e->getCode());

}

?>

Que invocándolo con: <http://localhost:8080/test_database.php> nos muestra la hora del sistema. Resulta por tanto que la conexión entre contenedores se hace a través de la red definida para los propios Dockers donde el contenedor que ejecuta MySQL usa la IP mostrada en la imagen previa y el puerto del propio contenedor que es el estándar 3306.

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

El contenido generado por IA puede ser incorrecto.

## Base de datos de autenticación y servicio PHP básico

Añadimos ahora la base de datos definida en el recurso complementario “sample\_sql” ya analizada en clases anteriores. Para ello usamos la opción “Archivo > Buscar archivo denominado …” de DBeaver y ejectutamos con el tercer botón una vez seleccionada la conexión tal como muestra la siguiente figura:

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Tenemos el siguiente esquema entidad / relación:

Diagrama

El contenido generado por IA puede ser incorrecto.

Creamos el servicio login\_mas\_inseguro.php disponible en la carpeta de recursos con el siguiente código (observar que la lógica no es correcta; porque no se verifica el salt y el algoritmo de hahing pero imaginar que hubiéramos hecho una lógica más sencilla sin esas consideraciones y simplemente convirtiendo a md5 la contraseña real):

<?php

// Conexión a la base de datos (sin protección)

$conn = mysqli\_connect("172.17.0.3", "root", "root", "ciberseguridad-pIII");

if (!$conn) {

http\_response\_code(500);

echo json\_encode(["error" => "Error de conexión a la base de datos."]);

exit;

}

// Solo permitir método POST

if ($\_SERVER["REQUEST\_METHOD"] !== "POST") {

http\_response\_code(405); // Método no permitido

echo json\_encode(["error" => "Método no permitido. Solo se acepta POST."]);

exit;

}

// Obtener datos del cuerpo del POST sin saneamiento

$username = $\_POST['username'] ?? '';

$password = md5($\_POST['password']) ?? '';

// Consulta deliberadamente vulnerable a inyección SQL

$sql = "SELECT \* FROM user\_login\_data WHERE LoginName = '$username' AND PasswordSalt = '$password'";

**file\_put\_contents("inyeccion.txt", $sql);**

$result = mysqli\_query($conn, $sql);

// Respuesta JSON en función del resultado

if ($result && mysqli\_num\_rows($result) == 1) {

echo json\_encode(["success" => true, "message" => "Login exitoso"]);

} else {

echo json\_encode(["success" => false, "message" => "Credenciales inválidas"]);

}

?>

Hemos destacado en negrita una línea que hace que la consulta realizada se almacene en un fichero de texto para que pueda afinar el como realizar la inyección y hacer un “bypass” del usuario y contraseña, pues incluso con la siguiente imagen podría necesitar pensar un poco para resolverlo.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

En los recursos de la unidad disponemos de dos ficheros de login más con nombre “login\_inseguro.php” y “login.php” para realizar distintas pruebas. Por ejemplo, con “login\_inseguro.php” podemos hacer desde Postman la siguiente petición:

|  |  |
| --- | --- |
| Key | Value |
| loginName | ' OR '1'='1 |
| password | cualquier cosa (ej. test) |

Si obtenemos un mensaje de contraseña no correcta vemos que no es necesario conocer el usuario (hemos saltado ese ataque) y podríamos forzar el login si acertamos la contraseña del primer usuario de base de datos (misma complejidad que si con técnicas de ingenería social conocemos el usuario real).

## ZAP Browser

La auditoría dinámica es clave para descubrir vulnerabilidades reales en aplicaciones web. OWASP ZAP, patrocinado por la comunidad OWASP, se consolida como una solución potente, gratuita y adaptable para realizar pruebas de seguridad.

ZAP (Zed Attack Proxy) es una herramienta de análisis de seguridad web que actúa como un proxy entre el navegador y el servidor web, permitiendo interceptar, visualizar y modificar el tráfico HTTP/HTTPS.

Esta capacidad la convierte en una pieza esencial para realizar auditorías de seguridad, ya que facilita la detección de vulnerabilidades como inyecciones SQL, XSS, inclusión de archivos locales, etc.

Descargamos ZAP desde: <https://www.zaproxy.org/download/>

Funcionalidades claves:

* **Escaneo activo y pasivo**. Automatiza la búsqueda de vulnerabilidades conocidas y auditadas sin interferir con la app o simulando ataques reales.
* **Spidering y descubrimiento**. Explora rutas ocultas y estructura de la aplicación mediante crawling y diccionarios personalizados.
* **Fuzzing y breakpoints**. Permite personalizar ataques y modificar solicitudes en tiempo real para detectar validaciones insuficientes.
* **Informes detallados**. Genera documentación con evidencia técnica que facilita la mitigación por el equipo de desarrollo.

**Los pasos para la configuración son**;

1. **Editar archivo hosts**

Para interceptar tráfico a un Docker (también ubicado en localhost) hay varias opciones.

Podemos, por ejemplo, añadir en el fichero hosts (<https://en.wikipedia.org/wiki/Hosts_(file)> ), la siguiente línea:

127.0.0.1 miapi.local

1. **Proxy OWASP ZAP**

Por defecto es localhost en el puerto 8080. Podemos cambiarlo al 8888 en:

Herramientas > Opciones > Red > Servidores / Proxies locales

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

1. **Redirigir tráfico en navegador o de Postman**

En Firefox o cualquier navegador debemos configurar ZAP para usarlo como proxy proporcionado en 127.0.0.1 y mismo puerto

Ajustes > Configuración de red > Configuración manual del proxy

En Postman

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Una vez así configurado si hago una llamada desde Postman a: <http://miapi.local:8080/login.php>

se intercepta desde ZAP como se muestra a continuación donde además ya se muestran varias alertas avisando de problemas de configuración en las cabeceras principalmente:

Interfaz de usuario gráfica, Texto, Aplicación, Word

El contenido generado por IA puede ser incorrecto.

Seleccionamos la URL en la parte superior izquierda y con el botón derecho, click en Fuzzer como se muestra a continuación:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Y con ayuda del siguiente video podemos ver como realizar un fuzzing sobre el servicio login.php que debería llevar al siguiente resultado final.

<https://www.youtube.com/watch?v=bNI5R92JTCo>

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

# Entregables de la práctica

Cada estudiante deberá entregar una documentación en formato PDF que incluya:

1. Capturas de pantalla que evidencien:

* El correcto despliegue de los servicios en Docker (MySQL y Apache+PHP).
* La ejecución y respuesta de las pruebas sobre el login básico (login.php) y la versión vulnerable (login\_inseguro.php).
* La configuración de OWASP ZAP interceptando las peticiones HTTP/HTTPS hacia el servicio.
* El ataque de fuzzing desde ZAP mostrando intentos válidos e inválidos.
* Las etiquetas o coincidencias definidas en ZAP para marcar accesos exitosos o respuestas clave (como "OK" o "Contraseña incorrecta.").

1. Breve explicación (2-3 líneas por imagen) indicando:

* Qué se está mostrando.
* Qué se ha conseguido verificar (acceso correcto, inyección, enumeración, etc.).

El objetivo de la entrega es demostrar el funcionamiento completo del servicio y de las técnicas de auditoría aplicadas, no solo a nivel técnico sino también de razonamiento.

# Opciones de ampliación para participantes que finalicen antes

Aquellos participantes que completen los objetivos principales de la práctica, pueden optar por realizar alguna de las siguientes **ampliaciones voluntarias**, que permiten profundizar en técnicas reales de desarrollo seguro y auditoría de APIs web:

## 1. Implementación de la API con frameworks modernos

En entornos profesionales, las APIs no se desarrollan con PHP plano, sino utilizando frameworks que facilitan el enrutado, el manejo de errores, la seguridad y la estructura del proyecto.

Se propone como ampliación:

* Reescribir la API de login utilizando un framework ligero como [**PHP Slim**](https://www.slimframework.com/).
* Alternativamente, se puede replicar el funcionamiento básico de la API con otros lenguajes y frameworks modernos como:
  + **Python + Flask** o **FastAPI**
  + **Java + Spring Boot**

Se debe conservar la lógica de hashing, uso de salt y conexión a la base de datos MySQL en Docker.

## 2. Uso avanzado de OWASP ZAP siguiendo recursos oficiales

Los participantes pueden profundizar en el uso de **OWASP ZAP** siguiendo el vídeo formativo de INCIBE:

🔗 <https://www.youtube.com/watch?v=1InikJ74654>

Algunas ideas para esta ampliación:

* Uso del **modo automático (Active Scan)** contra el servicio desplegado.
* Generación de un **informe de vulnerabilidades** completo en HTML o PDF.
* Automatización de pruebas con scripts ZAP o mediante su **API REST**.
* Detección de cabeceras inseguras, formularios sin tokens CSRF o recursos accesibles no protegidos.