# Implantación de sistemas seguros de despliegue de software

Tema 3. Laboratorio de ejercicios de Docker

# Índice general

# Capítulo 1

# Docker

En este capítulo nos centraremos en la creación de las imágenes Docker de las aplicaciones que se utilizaran en las siguientes sesiones del curso. Finalmente, utilizaremos Docker Compose para realizar el despliegue de la aplicación de manera automática y sincronizada.

## 1.1. DVWA

En esta sección, crearemos una imagen de Docker de la aplicación web Damn Vulnerable Web Application (DVWA) [1]. DVWA es una aplicación web escrita en PHP que utiliza MySQL como motor de base de datos y que contiene diferentes tipos de ejercicios y niveles para practicar vulnerabilidades web. Actualmente, existe una imagen de Docker ya generada por los propios autores en Docker Hub. Sin embargo, vamos a crear nuestro propia imagen desde cero.

En primer lugar, descargamos la aplicación del repositorio de Github [1] con el comando git clone https://... En la documentación del repositorio vemos un apartado en el que indica los pasos a seguir para instalar la aplicación en un entorno local. En nuestro caso, nos centraremos en Linux ya que es la imagen de Docker que utilizaremos como base. Como se ha comentado anteriormente, la aplicación consta de un servidor web con PHP y un motor de base de datos MySQL. Por tanto, crearemos dos imágenes de Docker, una para el servidor web y otra para el servidor de base de datos. Teniendo en mente como vamos a estructurar la aplicación, **¿qué instrucciones incluiríais en el Dockerfile para crear la imagen del servidor web? ¿Como crearíais la imagen de MySQL?**

El listing 1.1 muestra una posible solución para crear la imagen del servidor web. Partimos de una imagen de un servidor web Apache con PHP e instalamos las distintas extensiones que utilizará la aplicación web (i.e. mysqli, pdo, etc). Finalmente copiamos la aplicación web y concedemos los permisos requeridos para la realización de ciertos ejercicios de la aplicación.

Listing 1.1: Dockerfile para crear la imagen del servidor web DVWA

```
FROM php:7.0-apache
RUN docker-php-ext-install mysqli pdo pdo_mysql
RUN apt-get update && apt-get -y install libpng-dev
RUN docker-php-ext-install gd

WORKDIR /var/www/html/DVWA
COPY ./DVWA/ .
RUN chmod -R 755 .
RUN chmod 777 config hackable/uploads external/phpids/0.6/lib/IDS/
    tmp/phpids_log.txt

EXPOSE 80
```

Recordad dónde copiamos la aplicación en el Dockerfile (http://ip:puerto/DVWA).

Creamos la imagen de Docker y ejecutamos el contenedor siguiendo los comandos:

```
$ docker build -t dvwa .
$ docker run --name dvwa -p 9001:80 -d dvwa
```

En estos momentos, el contenedor está ejecutándose y deberíamos poder acceder a la aplicación web a través de un navegador utilizando la dirección del localhost y el puerto 9001. La aplicación web debería ser accesible pero al no haberla configurado todavía y al no tener activo el motor de base de datos, no funcionará correctamente. Por tanto, vamos a desplegar el motor de base de datos. Utilizaremos una imagen de MySQL. En Docker Hub, pueden verse las distintas variables de entorno que se le pueden pasar a la imagen de MySQL para definir la contraseña de administrador, usuario, contraseña, nombre de la base de datos, etc. Utilizando el siguiente comando definiremos las variables de entorno que son necesarias (los valores de estas variables se encuentran en el archivo de configuración de la aplicación web) y ejecutaremos nuestro contenedor:

```
$ docker run --hostname mysqldvwa -p 9002:3306 -e
    MYSQL_ROOT_PASSWORD=root -e MYSQL_USER=dvwa -e MYSQL_PASSWORD=
    p@ssw0rd -e MYSQL_DATABASE=dvwa --name mysqldvwa -d mysql --
    default-authentication-plugin=mysql_native_password
```

Creación y configuración del archivo config/config.inc.php. Revisar la configuración de conexión al motor de base de datos. Utilizar en vez de la dirección IP de localhost, el hostname que tendrá la máquina MySQL.

Instalación del editor nano: $ apt-get update && apt-get install nano

Si ejecutamos nuestro navegador y accedemos a la dirección y puerto comentados anteriormente, debería aparecernos algo similar a la Figura 1.1. Sin embargo, cuando pulsamos el botón Create/Reset Database, obtenemos un error, de que no ha sido posible establecer la comunicación con MySQL. **¿Por qué pensáis que se produce este error? ¿Cómo lo solucionaríais?**


**Figura 1.1**

El problema de comunicación se debe a qué la red a la que pertenecen por defecto los contenedores no realiza la traducción del nombre del contenedor a su dirección IP. Por tanto, cuando PHP intenta conectar a MySQL a través de su "hostname", éste no encuentra el servicio de base de datos. Para

solucionarlo, crearemos una nueva red de tipo bridge y añadiremos ambos contenedores a dicha red utilizando los siguientes comandos:

```
$ docker network create -d bridge dvwanet
$ docker network connect dvwanet mysqldvwa
$ docker network connect dvwanet dvwa
```

Si accedemos ahora a la aplicación y pulsamos el botón de crear o restablecer la base de datos, debería funcionar correctamente y una vez creada la base de datos nos debería redirigir a la web principal de la aplicación de manera que veréis algo similar a la Figura 1.2.

En caso de seguir sin funcionar, es posible que haya que reconfigurar el servicio DNS del Docker daemon. Para la realización del ejercicio podemos utilizar la IP de nuestro host directamente.

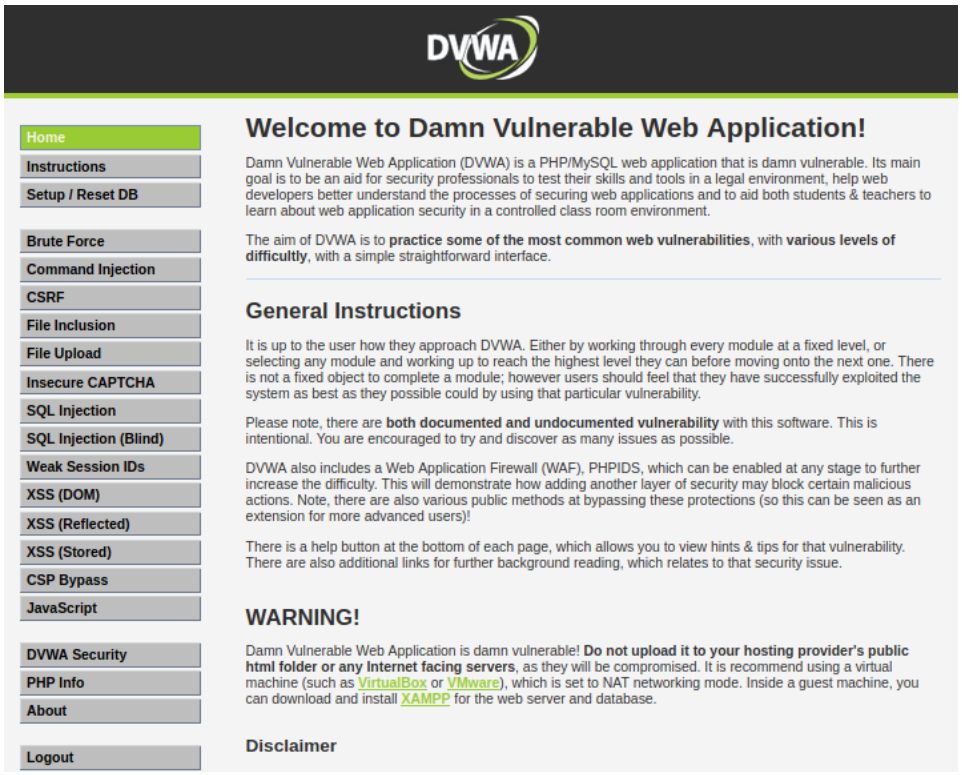


**Figura 1.2**

Las credenciales por defecto son **admin**/**password**

Por último, vamos a crear un archivo ".yml"para automatizar la creación y ejecución de ambas imágenes. Esto permite levantar ambos contenedores de manera sincronizada y facilitará su despliegue en el futuro. Además, al utilizar Docker Compose, no tendrás que acordarte de incluir las distintas

variables de entorno que hemos utilizado para MySQL, ya que las incluyes en el fichero y luego es simplemente realizar el despliegue con "docker-compose up". **¿Qué debéis incluir en el fichero de Docker Compose? ¿Que opciones añadiríais?**

El Listing 1.2 muestra una posible solución utilizando Docker Compose. Se puede observar que hemos creado 2 servicios, uno para el servidor web y otro para el servidor de bases de datos. Para el servicio de base de datos, indicamos que utilizaremos una imagen de MySQL, el puerto y qué variables de entorno usaremos en formato clave-valor. Para el servidor web, indicamos el nombre de la imagen, puerto y el directorio donde están las instrucciones para construir la imagen (Dockerfile).

Listing 1.2: Archivo docker-compose.yml para automatizar la construcción y desplegado de ambos contenedores.

```yaml
version: '3'
services:
   mysqlserver:
      container_name: mysqldvwa
      image: mysql
      restart: always
      hostname: mysqldvwa
      ports:
        - 9002:3306
      command: --default-authentication-plugin=mysql_native_password
      environment:
         MYSQL_ROOT_PASSWORD: root
         MYSQL_USER: dvwa
         MYSQL_PASSWORD: p@ssw0rd
         MYSQL_DATABASE: dvwa
   webserver:
      container_name: dvwa
      build: .
      image: dvwa
      restart: always
      ports:
        - 9001:80
      depends_on:
        - mysqlserver
```

Por último, desplegaremos ambos servicios utilizando "docker-compose up". Si nos fijamos en la salida que se muestra a través de la consola de comandos, podemos ver que al igual que hacíamos de manera manual, Docker Compose crea una red interna para ambos servicios.

**Ejercicio: ¿Cómo podríamos configurar de forma automática la aplicación DVWA (config/config.inc.php)?**

```
COPY config.inc.php config/
```

## 1.2. bWAPP

Buggy Web Application (bWAPP) [2] es una aplicación desarrollada con diferentes problemas de seguridad para practicar pentesting web. En total, presenta más de 100 vulnerabilidades web y cubre todos los riesgos incluidos en el OWASP Top 10. Al igual que la aplicación anterior, está desarrollada en PHP y utiliza una base de datos MYSQL. Es posible descargar la aplicación en una máquina virtual ya preparada o instalarla en modo local utilizando Apache, MySQL y PHP.

De manera a similar a como hicimos anteriormente, vamos a crear la imagen Docker de la aplicación vulnerable. Los pasos a seguir son los siguientes:

- Descargamos el archivo comprimido bWAPP_latest.zip del siguiente enlace: https://sourceforge.net/projects/bwapp/files/bWAPP
- Descomprimimos la aplicación. En el directorio resultante, se encuentra un archivo llamado INSTALL.txt donde se indican los principales pasos a seguir (para generar nuestro Dockerfile).
- Entramos a la aplicación (subcarpeta bWAPP) y modificamos el archivo que se encuentra en admin/settings.php cambiando la dirección del servidor (ip:puerto).
- Creamos el archivo Dockerfile con las instrucciones de instalación de Apache, PHP y de la aplicación.

La aplicación en el Dockerfile copiadla en /var/www/html/bWAPP

- Creamos el archivo docker-compose.yml para desplegar de manera automática y sincronizada el contenedor del servidor web y del servidor MySQL.
- Desplegamos ambos contenedores con el comando "docker-compose up".

Una vez creado las imágenes de Docker y desplegado sus contenedores, si accedemos en el navegador a la dirección de localhost, al puerto indicado deberíamos visualizar correctamente la aplicación. Para comprobar que todo funciona correctamente, es necesario acceder al archivo install.php para que genere la base de datos y las tablas necesarias. Una vez generada la base de datos, ya podremos hacer login en la aplicación. La Figura 1.3 muestra la pantalla principal de la web donde se pueden observar los distintos tipos de vulnerabilidades que presenta la aplicación para practicar.

Las credenciales por defecto son **bee**/**bug**


**Figura 1.3**

## 1.3. OWASP Mutillidae II

OWASP Mutillidae II [3] es una aplicación web vulnerable que presenta más de 40 vulnerabilidades y ejercicios para practicar pentesting web. En este caso, si navegamos por el repositorio del proyecto podemos observar que incluye los archivos de Docker necesarios para crear las imágenes y realizar el despliegue de los contenedores.

Por tanto, descargaremos el repositorio a nuestra máquina local (git clone https://...) y realizaremos el despliegue de los contenedores. Podemos observar que, en la estructura del proyecto, existen varios ficheros Dockerfile y el docker-compose.yml. Estos ficheros, al igual que sucedía en las aplicaciones anteriores, indican cómo se va a construir las imágenes y cómo se va a realizar el despliegue de las misma.

**Ejercicio: Revisad los ficheros e identificad las distintas acciones que realizan para crear las imágenes. Podréis observar que son muy parecidos a los anteriormente creados.**

Finalmente, realizar el despliegue de los contenedores con Docker Compose y comprobar que tenéis acceso a la aplicación.

Cuidado con los puertos que ya tengamos activos en nuestro host. Es recomendable editar el docker-compose.yml con puertos externos del rango 9000, por ejemplo

# Bibliografía

[1] D. team, "Damn vulnerable web application." [Online]. Available: https://github.com/digininja/DVWA

[2] M. Mesellem, "Buggy web application." [Online]. Available: https://sourceforge.net/projects/bwapp/

[3] "Owasp multillidae ii." [Online]. Available: https://github.com/webpwnized/mutillidae-docker