

**UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA**

**INGENIERIA EN CIENCIAS Y SISTEMAS**

**SEGURIDAD Y AUDITORIA DE SISTEMAS**



**ERICK ESTUARDO MEJIA HERNANDEZ**

**JOSUE ANTONIO MAQUIN CHOCOOJ**

**SELVIN IGNACIO PELICO TIUL**

**KEVIN ROLANDO ALFONSO MACZ MATA**

**Agosto-2025**

## Herramientas utilizadas

- Docker
- DVWA (Damn Vulnerable Web App)
- Kali Linux
- Nmap
- Nikto

## Configuración de red

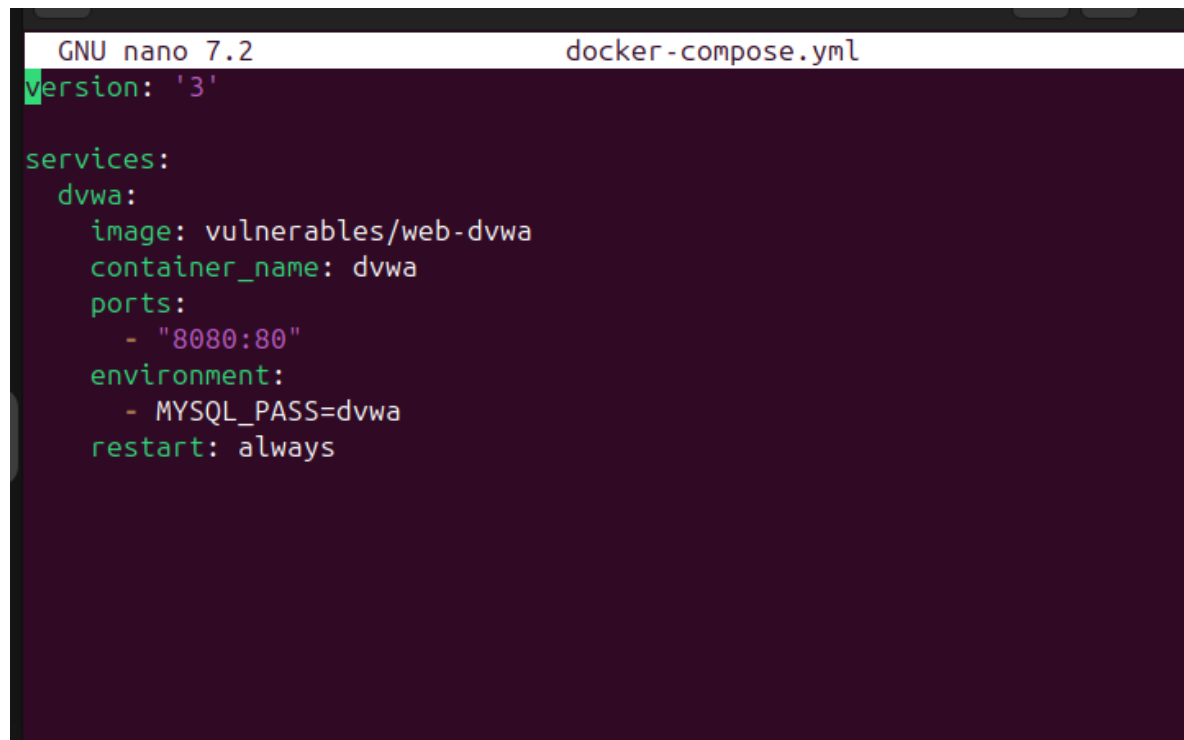
La máquina con Ubuntu Desktop (donde corre DVWA) y Kali Linux (atacante) fueron conectadas en la **\*\*misma red local\*\***, utilizando la opción de red en **\*\*puente (bridge)\*\*** en VMware, lo cual permitió el acceso a DVWA desde Kali mediante su IP: `http://192.168.0.13:8080`.

Ubuntu

```
mkdir dvwa-docker
```

```
cd dvwa-docker
```

```
nano docker-compose.yml
```



```
GNU nano 7.2                                docker-compose.yml
Version: '3'

services:
  dvwa:
    image: vulnerables/web-dvwa
    container_name: dvwa
    ports:
      - "8080:80"
    environment:
      - MYSQL_PASS=dvwa
    restart: always
```

## LEVANTAR CONTENEDOR

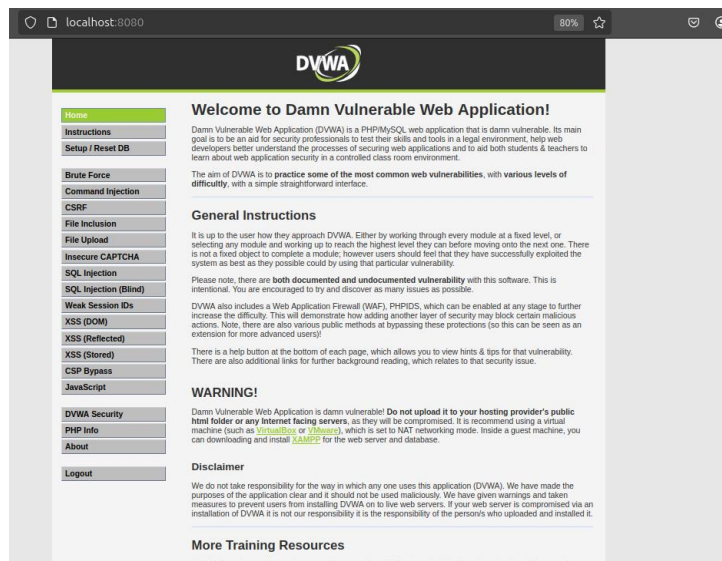
```
Aug 7 01:08
Terminal
josuex@josuex:~/Desktop$ mkdir dvwa-docker
josuex@josuex:~/Desktop$ cd dvwa-docker
josuex@josuex:~/Desktop/dvwa-docker$ nano docker-compose.yml
josuex@josuex:~/Desktop/dvwa-docker$ docker-compose up -d
Creating network "dvwa-docker_default" with the default driver
Pulling dvwa (vulnerables/web-dvwa:latest)...
latest: Pulling from vulnerables/web-dvwa
3e17c6eae66c: Pulling fs layer
0c57df616dbf: Pulling fs layer
0c57df616dbf: Downloading [>
3e17c6eae66c: Downloading [>
3e17c6eae66c: Downloading [=]
929kB/45.13MBtting
3e17c6eae66c: Downloading [=]
1.396MB/45.13MBtting
3e17c6eae66c: Pull complete
0c57df616dbf: Extracting [=====]
50.69MB/130.5MB
e9968e5981d2: Download complete
2cd72dba8257: Download complete
6cff5f35147f: Download complete
098cfff43466: Download complete
b3d64a33242d: Download complete
248B/240B
```

## CONTENEDOR CORRIENDO

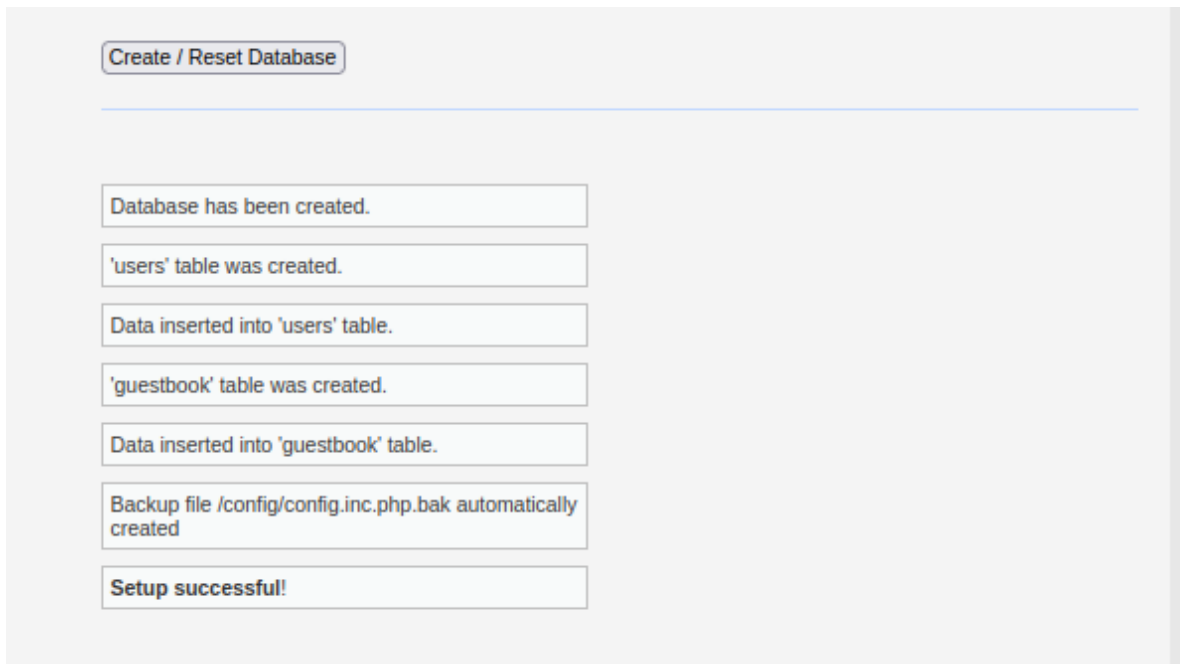
```
josuex@josuex:~/Desktop/dvwa-docker$ docker ps
CONTAINER ID   IMAGE                     COMMAND                  CREATED        STATUS
PORTS         NAMES
39a5f6d40c0d   vulnerables/web-dvwa     "/main.sh"              35 minutes ago Up 5 minutes
0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   dvwa
josuex@josuex:~/Desktop/dvwa-docker$
```

## Acceder a DVWA

<http://localhost:8080>



## PREISIONAR Create / Reset Database



## VISTA DESDE KALI LINUX

`nmap -sV -Pn 192.168.1.X -p 8080`

### Qué hace?

- `-sV`: Detecta versión del servicio
- `-Pn`: Omite ping
- `-p 8080`: Solo escanea ese puerto

```
josuessj@kali: ~/Escritorio
Archivo Acciones Editar Vista Ayuda
(josuessj@kali)-[~/Escritorio]
$ nmap -sV -Pn 192.168.0.13 -p 8080
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-07 01:20 CST
Nmap scan report for 192.168.0.13
Host is up (0.0094s latency).

PORT      STATE SERVICE VERSION
8080/tcp  open  http    Apache httpd 2.4.25 ((Debian))
MAC Address: 00:0C:29:1F:2D:67 (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.45 seconds
```

`nikto -h http://192.168.1.X:8080`

### ¿Qué hace?



- Escanea la aplicación buscando vulnerabilidades como:
  - Falta de headers de seguridad
  - Posibles inyecciones
  - Archivos expuestos
  - Métodos peligrosos habilitados

```

(josuessj@kali)~[~/Escritorio]
$ nikto -h http://192.168.0.13:8080
Nikto v2.5.0

Target IP:      192.168.0.13
Target Hostname: 192.168.0.13
Target Port:    8080
Start Time:     2025-08-07 01:23:41 (GMT-6)

Server: Apache/2.4.25 (Debian)
/: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
/: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
/: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
/: Cookie security created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
Root page / redirects to: login.php
No CGI Directories found (use '-C all' to force check all possible dirs)
Apache/2.4.25 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
/config/: Directory indexing found.
/config/: Configuration information may be available remotely.
/docs/: Directory indexing found.
/icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
/login.php: Admin login page/section found.
.gitignore: .gitignore file found. It is possible to grasp the directory structure.
8102 requests: 0 error(s) and 11 item(s) reported on remote host
End Time:      2025-08-07 01:23:55 (GMT-6) (14 seconds)

1 host(s) tested

```

## Vulnerabilidades encontradas por Nikto:

### 1. X-Frame-Options header is not present

- **Riesgo:** Permite ataques tipo *Clickjacking*
- **Mitigación:** Añadir X-Frame-Options: DENY o SAMEORIGIN en la configuración del servidor web

### 2. X-Content-Type-Options header is not set

- **Riesgo:** Puede permitir ataques por interpretación incorrecta de archivos (MIME sniffing)
- **Mitigación:** Añadir X-Content-Type-Options: nosniff

### 3. PHPSESSID created without the HttpOnly flag

- **Riesgo:** Las cookies pueden ser robadas por JavaScript (riesgo de XSS)
- **Mitigación:** Establecer la cookie con el flag HttpOnly

### 4. login.php encontrado

- **Info:** Página de autenticación identificada, posible objetivo de ataques de fuerza bruta o SQLi

### 5. Apache 2.4.25 es una versión desactualizada

- **Riesgo:** Podría contener vulnerabilidades ya conocidas
- **Mitigación:** Actualizar Apache a una versión más reciente (ej. 2.4.54 o superior)

### 6. /.git/ encontrado

- **Riesgo:** Permite acceso al historial del repositorio Git, posibles fugas de código fuente
- **Mitigación:** Bloquear acceso público a .git/ en la configuración del servidor

### 7. README, /config/, /icons/ y otros directorios visibles

- **Riesgo:** Permite acceso a archivos de configuración o recursos internos
- **Mitigación:** Deshabilitar el listado de directorios y proteger rutas sensibles

la aplicación **DVWA** permite ejecutar comandos SQL maliciosos a través de formularios, accediendo a datos sin autorización.

menú izquierdo, haz clic en:

SQL Injection

EN EL CAMPO USER ID ESCRIBIR:

1' OR '1'='1

### Vulnerability: SQL Injection

User ID:

ID: 1' OR '1'='1  
First name: admin  
Surname: admin

ID: 1' OR '1'='1  
First name: Gordon  
Surname: Brown

ID: 1' OR '1'='1  
First name: Hack  
Surname: Me

ID: 1' OR '1'='1  
First name: Pablo  
Surname: Picasso

ID: 1' OR '1'='1  
First name: Bob  
Surname: Smith

CON VALIDACIONES MINIMAS

### Vulnerability: SQL Injection

Click [here to change your ID.](#)

ID: 1' OR '1'='1  
First name: admin  
Surname: admin

**Conclusiones**

Se logró simular un entorno vulnerable utilizando DVWA y se comprobó que es posible explotar fallos como inyección SQL.

El uso de herramientas como Nikto permitió identificar múltiples vulnerabilidades del servidor web. Finalmente, la explotación de la inyección SQL evidenció el riesgo de no validar correctamente las entradas del usuario.

Este reto demuestra la importancia de:

- Mantener los servicios actualizados
- Configurar cabeceras HTTP seguras
- Validar siempre la entrada de datos

### **Lecciones aprendidas**

- Cómo usar Docker para levantar entornos vulnerables
- Uso práctico de Nmap y Nikto en auditorías web
- Análisis de resultados y generación de mitigaciones

### **¿Cómo podría ser explotada DVWA y cómo mitigar los riesgos?**

#### **Introducción al contexto de explotación**

Durante el despliegue del entorno vulnerable, se utilizó DVWA (Damn Vulnerable Web Application) como aplicación de prueba para simular vulnerabilidades reales que pueden presentarse en sitios web mal protegidos.

El objetivo fue demostrar cómo un atacante, desde una máquina remota (Kali Linux), puede identificar debilidades de seguridad en un servidor expuesto, explotarlo y acceder a información sin autorización.

#### **Explotación de vulnerabilidad: Inyección SQL (SQLi)**

La vulnerabilidad encontrada corresponde al tipo Inyección SQL, una de las más comunes y críticas según el OWASP Top 10. Este fallo ocurre cuando una aplicación no valida ni sanitiza adecuadamente los datos ingresados por el usuario, permitiendo que un atacante modifique las consultas SQL internas.

#### **Paso a paso de la explotación:**

El atacante accede al módulo SQL Injection de DVWA.

Introduce el siguiente payload malicioso en el campo de ID:

bash

Copiar

Editar

1' OR '1'='1



La aplicación ejecuta la consulta SQL con ese valor sin verificar su validez. Esto altera la lógica de la consulta, que ahora se convierte en:

sql

Copiar

Editar

```
SELECT * FROM users WHERE id = '1' OR '1'='1';
```

Como '1'='1' siempre es verdadero, la base de datos devuelve todos los registros, mostrando información privada de varios usuarios como:

admin

Gordon Brown

Bob Smith, entre otros

### **Riesgos e impactos de la explotación**

Explotar esta vulnerabilidad permitiría a un atacante:

- Acceder a información confidencial (usuarios, contraseñas, correos)
- Autenticarse sin credenciales reales
- Modificar o eliminar registros de la base de datos
- Escalar privilegios o realizar movimientos laterales dentro del servidor
- Instalar puertas traseras (backdoors) para persistencia

### **Medidas de mitigación recomendadas**

#### **A nivel de desarrollo de la aplicación:**

- Validar y sanitizar todas las entradas del usuario antes de enviarlas a la base de datos.
- Usar consultas preparadas (Prepared Statements) o ORMs (Object-Relational Mapping) para construir las consultas SQL de forma segura.
- Aplicar técnicas de escape de caracteres y eliminación de entradas maliciosas.
- Implementar mensajes de error genéricos para evitar revelar detalles internos del sistema.

#### **A nivel de servidor web:**

Actualizar componentes obsoletos, como Apache (Nikto detectó Apache 2.4.25, versión vulnerable).

Configurar cabeceras de seguridad HTTP como:

- X-Frame-Options
- X-Content-Type-Options
- Content-Security-Policy
- Restringir el acceso a directorios internos como `.git/`, `/config/`, etc.
- Evitar el listado de directorios (Directory Indexing).

#### **A nivel de infraestructura:**

- Utilizar un WAF (Web Application Firewall) para filtrar ataques comunes.
- Registrar todos los intentos de acceso a páginas sensibles.
- Implementar autenticación multifactor en zonas críticas.
- Realizar auditorías y pruebas de penetración periódicas.

### **Reflexión final**

Este ejercicio demuestra que un simple error de validación de entrada puede exponer completamente la seguridad de una aplicación web. La explotación fue sencilla y rápida gracias a la falta de controles básicos. Si bien DVWA es un entorno de laboratorio, muchas aplicaciones reales presentan fallos similares por malas prácticas de desarrollo o falta de mantenimiento.

Implementar seguridad desde la etapa de diseño y desarrollo es clave para proteger los datos, la reputación y la continuidad operativa de cualquier organización.