# Simulación de Ciberataques a Servicios Web en Entornos Controlados: Un Enfoque Seguro para la Evaluación de Vulnerabilidades

## Simulation of Cyberattacks on Web Services in Controlled Environments: A Secure Approach to Vulnerability Assessment

**Andrés Sebastián Almeida, Alison Paola Moncayo, Darwin Roberto Valdiviezo**
Universidad de las Fuerzas Armadas ESPE, Sangolquí, Ecuador.
{asalmeida4, apmoncayo2, drvaldiviezo2}@espe.edu.ec.

## Resumen

El aumento de ataques dirigidos a sistemas web ha generado una creciente preocupación en el ámbito de la seguridad informática, lo que ha impulsado la necesidad de investigar sus impactos en entornos controlados. Este artículo evalúa una aplicación web con vulnerabilidades ante diferentes ataques comunes, con la finalidad de evidenciar fallas en la seguridad del sistema y proponer soluciones para prevenirlas. Para lograrlo, se utilizó OWASP (Open Web Application Security Project) Juice Shop, una aplicación desarrollada por el proyecto OWASP, una comunidad internacional que promueve prácticas para mejorar la seguridad del software. Juice Shop permite realizar pruebas en un entorno diseñado para simular fallas comunes. Se evaluaron distintas vulnerabilidades, incluidos la inyección SQL, escaneo de puertos, ejecución de scripts maliciosos (XSS), carga de archivos maliciosos, recorrido por directorios y falsificación de peticiones en sitios cruzados (CSRF). Dichas simulaciones se ejecutaron en Kali Linux, dentro de una máquina virtual en VirtualBox sobre un host en Windows 11, utilizando Docker para desplegar el entorno web. Además, se utilizó Burp Suite como herramienta para interceptar solicitudes, modificar parámetros y explotar las fallas detectadas. Los ataques XSS demostraron la ejecución de código malicioso, lo que expuso debilidades en el filtrado de entradas. Sin embargo, la aplicación fue capaz de resistir los ataques de recorrido de directorios y carga de archivos maliciosos. Para hacer frente a la amenaza de ciberataques, este estudio subraya la importancia de incorporar poderosas medidas de ciberseguridad en el desarrollo de servicios web, más allá de las consideraciones típicas.

**Palabras clave:** *Ciberseguridad, Vulnerabilidades, Servicios Web, Entornos Virtualizados, Inyección SQL.*

## Abstract

The increase in attacks targeting web systems has generated growing concern in the field of computer security, driving the need to investigate their impacts in controlled environments. This article assesses a web application vulnerable to various common attacks to identify security flaws in the system and propose solutions to mitigate them. To achieve this, we utilized the OWASP (Open Web Application Security Project) Juice Shop, an application developed by the OWASP project. This international community promotes best practices to enhance software security. Juice Shop allows testing in an environment designed to simulate common flaws. Various vulnerabilities were evaluated, including SQL injection, port scanning, cross-site scripting (XSS), malicious file upload, directory traversal, and cross-site request forgery (CSRF). These simulations were run on Kali Linux, within a VirtualBox virtual machine on a Windows 11 host, using Docker to deploy the web environment. Also, Burp Suite was used as a tool to intercept requests, modify parameters, and exploit the detected flaws. XSS attacks demonstrated the execution of malicious code, exposing weaknesses in input filtering. However, the application was able to withstand directory traversal and malicious file upload attacks. To address the threat of cyberattacks, this study emphasizes the importance of incorporating robust cybersecurity measures into web service development, extending beyond typical considerations.

**Keywords:** *Cybersecurity, Vulnerabilities, Web Services, Virtualized Environments, SQL Injection.*

Fecha de Recepción: 05/09/2024 - Aceptado: 20/12/2024 - Publicado: 31/12/2024
ISSN: 2477-9253 – DOI: http://dx.doi.org/10.24133/RCSD.VOL09.N04.2024.05

## I. Introducción

A medida que el Internet y sus aplicaciones web se expanden en alcance, complejidad y uso diario, la ciberseguridad se ha convertido en una de las principales prioridades debido al aumento de ataques que afectan la integridad, y en algunos casos la disponibilidad de los sistemas. Según Ukwandu et al. (2020), los ataques cibernéticos en la actualidad pueden causar desde robos de datos hasta el colapso de servicios web. La prueba simulada en un entorno controlado ha surgido como una herramienta efectiva para poder analizar las vulnerabilidades y buscar estrategias que ayuden a prevenirlas (Straub, 2020; Gorgone Carvajal, 2023). Esto subraya la importancia de adoptar medidas proactivas para reforzar la seguridad de los servicios web y la infraestructura subyacente, además, estas prácticas permiten estudiar ataques comunes como escaneo de puertos, inyección SQL y XSS, replicando escenarios reales sin poner en riesgo sistemas productivos (Fuertes Díaz & Macas Carrasco, 2023).

En Ecuador, los ciberataques han aumentado en los últimos años, afectando tanto al sector público como privado. Echeverría et al. (2020) reportan más de 40 millones de intentos de intrusión en solo una semana, lo que evidencia una alarmante vulnerabilidad. Estudios locales demostraron que los usuarios, especialmente los millennials, carecen de conciencia suficiente sobre la ciberseguridad, lo cual hace que estén en primera fila ante riesgos de ataques de spoofing o sniffing (Augusto & Agudelo, 2018; Anu & Vimala, 2018).

Este estudio tiene como objetivo analizar las vulnerabilidades que puede presentar un servicio web, entre ellas, el análisis de puertos abiertos, inyección SQL, ejecución de scripts (XSS), carga de archivos, recorrido por directorios y falsificación de peticiones en sitios cruzados (CSRF). Con ello se busca evidenciar que los servicios web no deben centrarse únicamente en el desarrollo funcional, sino también incorporar consideraciones de ciberseguridad. La simulación se ha realizado en un entorno controlado, con el fin de evitar afectaciones reales que puedan implicar consecuencias éticas o legales.

El resto del artículo se organiza de la siguiente manera: el apartado 2 presenta los trabajos relacionados a partir de los cuales se establecieron bases y pruebas similares para la detección de vulnerabilidades web. El apartado 3 describe los materiales y métodos empleados en cada una de las simulaciones. En el apartado 4 se analizan las formas de mitigación de cada ataque, junto con los resultados obtenidos a través de un análisis estadístico descriptivo, basado en la tasa de éxito, el tiempo de ejecución y la severidad (CVSS) de los ataques simulados. Finalmente, el apartado 5 expone las conclusiones del estudio y propone posibles líneas de trabajo futuro.

## II. Trabajos relacionados

El estudio de ciberseguridad en aplicaciones web cuenta con una amplia trayectoria, con numerosas investigaciones que emplean entornos controlados para corroborar vulnerabilidades habituales. En esta sección, se revisan investigaciones relacionadas a las vulnerabilidades analizadas en OWASP Juice Shop.

El uso de aplicaciones con vulnerabilidades incorporadas ha sido documentado en diversos estudios como una estrategia para la enseñanza de ciberseguridad. Vega-Oyola et al. (2022) analizaron entornos controlados orientados a la formación en ciberseguridad, observando mejoras en la capacidad de los estudiantes universitarios para identificar vulnerabilidades. Por otro lado, Son (2016) propuso un marco estructurado para la implementación de laboratorios virtuales que permiten prácticas en ciberseguridad dentro de contextos académicos, sin implicar riesgos legales.

De igual manera, las técnicas de inyección SQL han experimentado cambios en los últimos años. Sadegh & Tajalli (2013) analizaron patrones de inyección y técnicas de evasión en aplicaciones actuales, indicando que incluso sistemas con medidas básicas continúan siendo susceptibles a este tipo de ataques. Veeraburden y Bekaroo (2022), por su parte, realizaron un estudio comparativo de herramientas automatizadas para la detección de vulnerabilidades de inyección SQL, señalando que las tasas de detección varían según el contexto de implementación de los sistemas.

En la actualidad, las vulnerabilidades de Cross-Site Scripting (XSS) y Cross-Site Request Forgery (CSRF) continúan presentes como vectores de ataque en aplicaciones web. Lekies et al. (2023) documentaron la evolución de los ataques XSS en entornos JavaScript, identificando nuevos vectores que superan mecanismos de protección convencionales. En cuanto a CSRF, Barth et al. (2021) propusieron un enfoque de defensa en profundidad que integra el uso de tokens, encabezados personalizados y validación de origen, observando una reducción de vulnerabilidades en entornos de prueba controlados.

Durumeric et al. (2020) llevaron a cabo un estudio a gran escala acerca de la exposición de servicios en Internet, revelando que aproximadamente el 15% de los servidores web exponen servicios innecesarios, los cuales podrían ser aprovechados por atacantes cibernéticos.

Rafique et al. (2019) analizaron cómo los atacantes combinan diferentes vulnerabilidades para lograr la ejecución de código remoto mediante archivos maliciosos. Este estudio complementa nuestra investigación, donde se encontró que OWASP Juice Shop implementa controles efectivos contra estos ataques específicos.

Bau et al. (2021) establecieron un marco sistemático para evaluar la seguridad de aplicaciones web modernas, abordando tanto vulnerabilidades técnicas como problemas de configuración que se suelen pasar por alto.

Estudios recientes también han explorado nuevos paradigmas en seguridad web, como el de Ahmadi (2023), donde se analizaron los desafíos de seguridad específicos de las aplicaciones web progresivas (PWA) y las arquitecturas serverless, identificando nuevos vectores de ataque exclusivos para estas tecnologías. Por otro lado, Vighe (2024) estudió cómo la adopción e implementación de DevSecOps influye en la seguridad de las aplicaciones web, encontrando correlaciones positivas entre la integración temprana de pruebas de seguridad y la reducción de vulnerabilidades en entornos de producción.

Estos estudios han permitido contextualizar la presente investigación, la cual facilita una evaluación práctica y detallada de múltiples vectores de ataque en un sistema realista pero controlado.

## III. Materiales y métodos

Antes de definir las herramientas y técnicas empleadas, se plantearon las siguientes preguntas de investigación, alineadas con los objetivos del estudio:

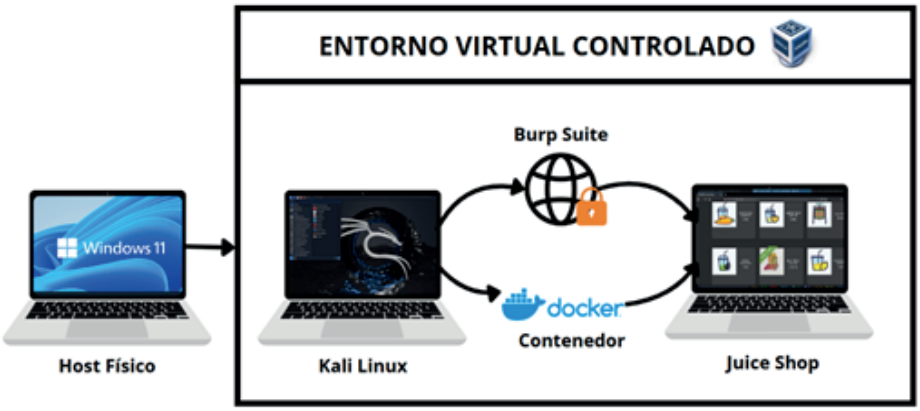
- RQ1: ¿Cuáles de las vulnerabilidades web más comunes pueden ser explotadas con éxito en un entorno controlado utilizando OWASP Juice Shop?
- RQ2: ¿Qué niveles de gravedad (Common Vulnerability Scoring System, CVSS) y tiempos de ejecución presentan los diferentes ataques realizados contra la aplicación vulnerable?

- RQ3: ¿Qué métricas operativas clave se pueden obtener al realizar pruebas de penetración sobre una aplicación web desplegada en Docker, y qué indican sobre su seguridad y desempeño?

La topología propuesta para la simulación contempla una máquina virtual que opera sobre un host físico con sistema operativo Windows 11, en el cual se utiliza VirtualBox. Dentro de esta máquina virtual está instalado Kali Linux, el cual cuenta con Docker y Burp Suite. Docker aloja la aplicación utilizada para vulnerar OWASP Juice Shop, mientras que Burp Suite se encarga de interceptar, modificar y reenviar solicitudes HTTP/HTTPS con el fin de ejecutar distintos vectores de ataque. La comunicación se realiza a través de una red virtual privada (Network Address Translation, NAT), lo que garantiza un entorno aislado y seguro. La estructura completa de este entorno se muestra en la Figura 1.

La arquitectura empleada en el sistema se basa en el despliegue de OWASP Juice Shop dentro de un contenedor Docker, ejecutado en un entorno virtual Kali Linux. Inicialmente, se instalan Docker y sus dependencias mediante comandos dentro de Kali, y posteriormente se ejecuta la imagen correspondiente de Juice Shop. Esta imagen contiene todos los componentes necesarios para ejecutar la aplicación, incluyendo Node.js y el framework Express. El sistema se ejecuta en el puerto 3000, lo que permite su acceso desde el navegador. Además, el contenedor mantiene una conexión interna con la base de datos del sistema, donde se almacena la información de usuarios y productos. Por esa razón, esta arquitectura nos proporciona un entorno seguro y sobre todo controlado, con el que podemos realizar las pruebas de vulnerabilidad sin infringir leyes ni exponer datos de personas reales.

**Figura 1:** *Topología utilizada en la simulación*

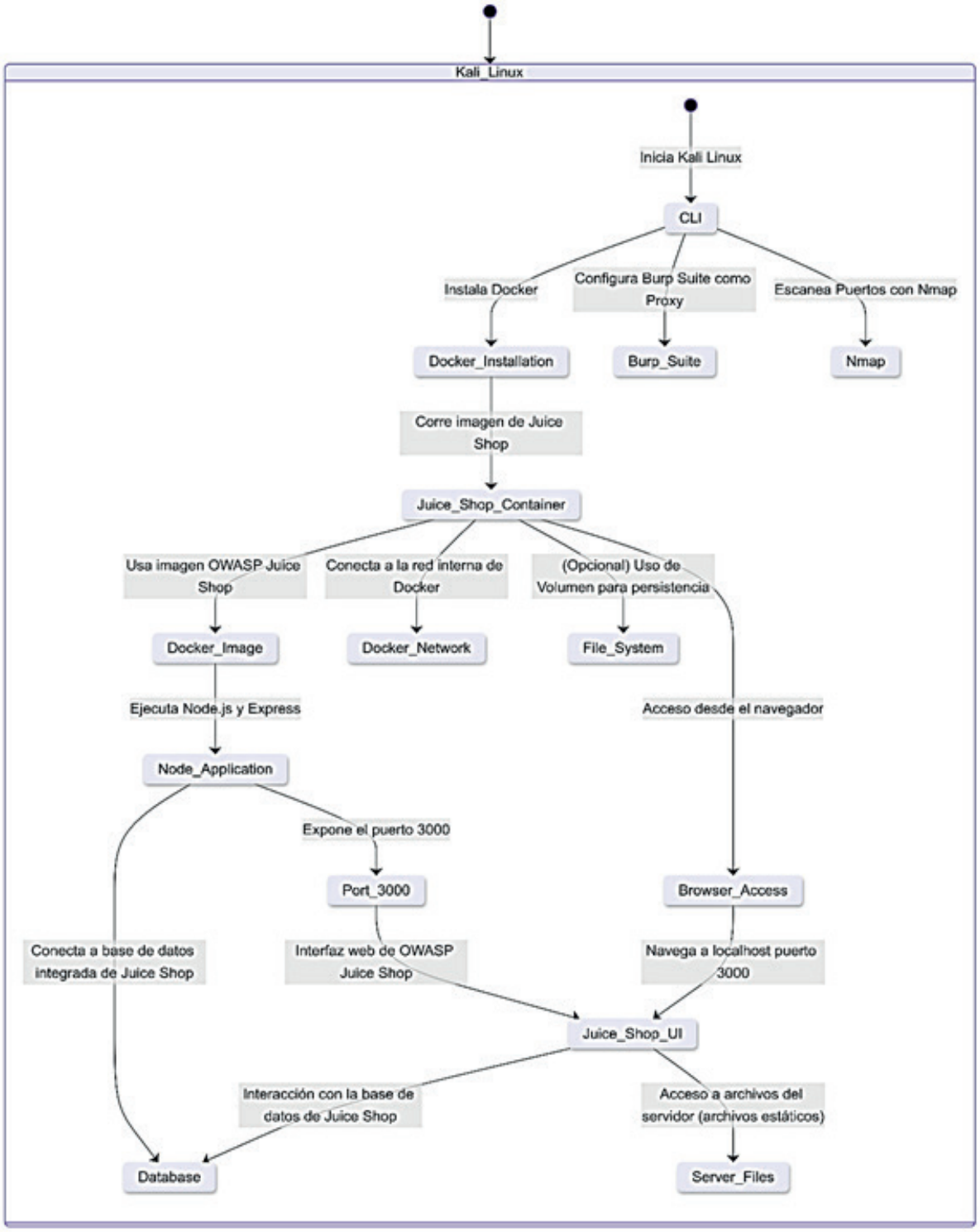



A continuación, se describe el flujo de comunicación establecido entre las herramientas utilizadas en el entorno de simulación, enfocándose en la interacción entre Kali Linux, Burp Suite y la aplicación OWASP Juice Shop:

- Kali Linux (Burp Suite) se comunica con la aplicación web vulnerable a través de la red virtual.
- Burp Suite intercepta y modifica las solicitudes enviadas a la aplicación web.
- La aplicación web vulnerable en Docker responde a las solicitudes enviadas por Burp Suite.

Esta topología es cerrada y segura, ideal para poder realizar pruebas de penetración y ciberataques en un entorno controlado, tal como se representa en la Figura 2.

**Figura 2:** *Arquitectura del sistema.*



### 3.1. Configuración de la aplicación web en Kali Linux

Se configuró el entorno en Kali Linux. Se actualizan los paquetes disponibles en el sistema a través del comando sudo apt-get update. Se instala Docker en el sistema operativo Kali Linux mediante sudo apt-get install docker.io, y se inicia el servicio Docker con sudo systemctl start docker. Luego, se descarga la imagen de Docker de la aplicación web que se usará, en este caso la de OWASP Juice Shop, utilizando el comando sudo docker pull bkimminich/juice-shop. Posteriormente, se ejecuta el contenedor que contiene el aplicativo web con sudo docker run -d -p 3000:3000 bkimminich/juice-shop. Con ello, ya se podrá acceder a la aplicación web a través de http://localhost:3000 mediante el navegador Mozilla Firefox.

Una vez realizados estos pasos, ya se dispone de un entorno web controlado, en el cual, se ha decidido desarrollar diversas actividades para con ello, demostrar, los riesgos a los cuales se puede ver expuesto un entorno real. Se utilizó este entorno dado por OWASP, puesto que es riesgoso el desarrollar un entorno para hacer este tipo de ataques y pruebas los cuales se presentan a continuación.

### 3.2. Identificación de puertos abiertos

El primer paso consistió en la identificación de puertos abiertos mediante un escaneo de red dirigido al aplicativo web. Esta acción permite descubrir los servicios expuestos y activos en dichos puertos. Un atacante puede utilizar esta información para determinar qué tecnologías o configuraciones están en uso, lo que facilita la preparación de vectores de ataque específicos, como exploits asociados a esos servicios o la explotación de configuraciones incorrectas.

Para este procedimiento, se inicia verificando la dirección IP de la instancia donde se encuentra desplegado el aplicativo web. Una vez identificada, se procede a realizar un escaneo de puertos utilizando la herramienta nmap, con el objetivo de detectar los puertos abiertos asociados a dicha dirección. Con esta información, es posible avanzar hacia las siguientes actividades del entorno de prueba, centradas en evaluar el comportamiento del aplicativo web frente a ciberataques y pruebas de penetración.

### 3.3. Inyección SQL

En esta fase se simuló un ataque de inyección SQL con el objetivo de evaluar la respuesta del aplicativo web ante intentos de manipulación de consultas a la base de datos. Para ello, se empleó la herramienta Burp Suite, configurada manualmente como proxy en la dirección 127.0.0.1:8080, utilizando el navegador para interceptar las solicitudes enviadas al aplicativo web.

En el aplicativo web se accedió al formulario de inicio de sesión, donde se ingresa una contraseña arbitraria y se modifica el campo de correo electrónico. La solicitud es interceptada con Burp Suite, y se reemplaza el valor original del correo por una serie de payloads de inyección SQL, incluyendo expresiones como ‘ OR 1=1 -- ‘. Estos valores fueron enviados al servidor con el fin de observar el comportamiento del sistema ante intentos de elusión de autenticación mediante manipulación de parámetros.

### 3.4. Secuencias de comandos entre sitios (XSS)

El ataque de Cross-Site Scripting (XSS) consiste en la inyección de código JavaScript malicioso dentro de campos o componentes del sitio web que interpretan contenido proporcionado por el usuario. Este tipo de vulnerabilidad puede ser utilizado para robar cookies, modificar la interfaz o redirigir al usuario hacia sitios externos sin su consentimiento.

Para esta prueba, se navegó dentro del aplicativo OWASP Juice Shop hacia componentes que permiten entradas de usuario, como la barra de búsqueda y el módulo de retroalimentación. En dichos campos, se ingresan diferentes cargas útiles (payloads) diseñadas para provocar la ejecución de scripts. Algunos de los vectores utilizados incluyen: <script>alert('XSS')</script>, <img src=x onerror=alert(1)> y <svg/onload=alert('XSS')>. Estas entradas fueron enviadas a través del navegador y monitorizadas con Burp Suite para observar cómo responde el sistema ante intentos de ejecución de código no autorizado.

### 3.5. Vulnerabilidad con carga de archivos

La vulnerabilidad de carga de archivos consiste en la posibilidad de subir archivos al servidor sin un control adecuado sobre su tipo o contenido, lo que podría permitir la ejecución de código no autorizado, acceso indebido o incluso el compromiso total del sistema. Un atacante podría aprovechar esta falla para introducir archivos que contengan scripts o instrucciones maliciosas que el servidor pueda interpretar y ejecutar. En este caso, se accedió a una sección del aplicativo web que permite la carga de archivos, específicamente la opción destinada a actualizar la foto de perfil del usuario.

Para esta prueba, se desarrolló un script PHP diseñado para ejecutar comandos en el servidor web mediante la URL. Este archivo es accedido desde el navegador incluyendo un parámetro con la instrucción deseada. A continuación, se intenta cargar el archivo como imagen de perfil en el aplicativo web, con el fin de evaluar el comportamiento del sistema ante la subida de archivos que no corresponden a formatos de imagen. La finalidad de este procedimiento es identificar si el sistema realiza algún tipo de validación del archivo enviado, ya sea por extensión o por tipo MIME (Multipurpose Internet Mail Extensions).

### 3.6. Recorrido de directorios

El recorrido de directorios es una técnica utilizada para acceder a archivos o directorios fuera del espacio permitido por la aplicación, mediante la manipulación de rutas relativas en las solicitudes del cliente. Este tipo de ataque busca explotar fallas en la validación de rutas para acceder a archivos del sistema que normalmente estarían restringidos.

En esta prueba, se utiliza Burp Suite para interceptar y modificar solicitudes enviadas al servidor. A través de esta herramienta, se identificaron parámetros en las URLs o formularios susceptibles de aceptar rutas de archivos. Se sustituyeron dichos valores por cadenas que representan navegación hacia niveles superiores del sistema, como ../../../../etc./passwd, con el objetivo de evaluar si el servidor responde con acceso a archivos fuera del directorio de la aplicación. El procedimiento se realizó en un entorno controlado para observar la respuesta de la aplicación ante intentos de exploración no autorizada del sistema de archivos.

### 3.7. Falsificación de solicitudes entre sitios (CSRF)

La falsificación de solicitudes entre sitios (CSRF, por sus siglas en inglés) es una técnica que busca inducir a un usuario autenticado a ejecutar una acción no autorizada en una aplicación web. Este ataque explota la confianza que el sistema tiene en el navegador del usuario, mediante el envío de solicitudes manipuladas desde sitios externos.

Para esta prueba, se construyó un formulario HTML externo con los mismos parámetros que la aplicación vulnerable utiliza para realizar cambios, como la actualización de contraseñas. Este formulario se configuró para enviarse automáticamente al cargar la página, simulando una acción que el usuario no ha autorizado. El objetivo del procedimiento fue analizar si la aplicación valida el origen de la solicitud, y si requiere medidas de protección como tokens CSRF, cabeceras personalizadas o validación del origen.

### 3.8. Vulnerabilidad de manipulación de parámetros

La manipulación de parámetros es una vulnerabilidad que consiste en alterar los valores enviados en solicitudes HTTP, como formularios o URLs, con el fin de modificar el comportamiento de una aplicación web. Esta técnica puede ser utilizada por un atacante para realizar acciones no autorizadas, como acceder a información de otros usuarios, cambiar datos almacenados o elevar privilegios.

Para esta prueba, se accede a la funcionalidad de retroalimentación del aplicativo web, en la cual el usuario puede calificar la página mediante un valor de puntuación. Se utiliza Burp Suite para interceptar la solicitud HTTP correspondiente y se identifica el parámetro responsable del puntaje. Posteriormente, se modifica dicho valor antes de enviarlo al servidor, con el objetivo de evaluar cómo responde la aplicación ante la alteración de datos enviados desde el cliente.

## IV. Evaluación de Resultados y Discusión

### 4.1. Evaluación de Resultados

*RQ1: ¿Cuáles de las vulnerabilidades web más comunes pueden ser explotadas con éxito en un entorno controlado utilizando OWASP Juice Shop?*

Se ejecutaron cinco tipos de ataques diferentes con el objetivo de explorar las vulnerabilidades más comunes. La siguiente tabla muestra un resumen de cada ataque y su éxito.

**Tabla 1:** *Resumen de ataques realizados en OWASP Juice Shop*

| Tipo de Ataque | Descripción | Resultado | Vulnerabilidad Detectada |
|---|---|---|---|
| **SQL Injection** | Inyección SQL en el login para omitir autenticación | Acceso no autorizado | Sí |
| **XSS** | Código malicioso inyectado en barra de búsqueda | Alerta XSS activada | Sí |
| **File Upload** | Subida de archivo PHP con shell reversa | Archivo no ejecutado | No |
| **Directory Traversal** | Acceso a archivos restringidos del servidor | Acceso denegado | No |
| **CSRF** | Cambio de contraseña vía formulario externo | Contraseña modificada | Sí |

De los cinco ataques realizados, tres fueron exitosos: SQL Injection, XSS y CSRF. Esto confirma la vulnerabilidad crítica que tienen los sistemas web y cómo estas fallas pueden comprometer la seguridad de las aplicaciones cuando no se aplican controles adecuados.

Para la prueba de inyección SQL se empleó una serie de payloads diseñados para evadir el proceso de autenticación del sistema. Las solicitudes modificadas fueron interceptadas y enviadas utilizando Burp Suite. Los resultados obtenidos, incluyendo los códigos de respuesta, tiempos de ejecución y efectividad de cada intento, se resumen en la Tabla 2.

**Tabla 2:** *Resultados detallados de SQL Injection en Juice Shop*

| Payload | ¿Éxito? | Código | T1 | T2 | T3 | Long. | CVSS | Frecuencia |
|---|---|---|---|---|---|---|---|---|
| ' OR 1=1# | No | 500 | 14 | 12 | 15 | 1520 | 9.8 | 0/1 |
| ' OR 1=1/* | Sí | 200 | 26 | 16 | 19 | 1185 | 9.8 | 3/3 |
| ' OR "=' | No | 401 | 10 | 12 | 12 | 413 | 9.8 | 0/1 |
| ' OR 1=1 LIMIT 1-- | Sí | 200 | 13 | 16 | 14 | 1185 | 9.8 | 3/3 |
| ' AND 1=1-- | No | 401 | 10 | 9 | 9 | 413 | 9.8 | 0/1 |
| admin' OR 1=1-- | Sí | 200 | 13 | 13 | 14 | 1185 | 9.8 | 3/3 |

Tres de los payloads utilizados lograron ejecutarse correctamente, obteniendo tokens válidos y permitiendo el acceso al sistema con privilegios de administrador. Las respuestas generadas por el servidor presentaron una longitud uniforme de 1185 bytes y códigos de estado HTTP 200, lo que indica que no se aplicaron controles de validación eficaces sobre las entradas manipuladas durante el proceso de autenticación. En el ataque XSS se probaron diferentes vectores. La Tabla 3 presenta los resultados por ubicación, éxito y tiempo.

**Tabla 3:** *Resultados de ataques XSS en Juice Shop*

| Payload | ¿Éxito? | Tiempo (ms) | Ubicación | CVSS | Frecuencia |
|---|---|---|---|---|---|
| <script>alert('XSS')</script> | No | 0 (no ejecutado) | Customer Feedback | 7.4 | 0/1 |
| <img src=x onerror=alert(1)> | No | 0 (no ejecutado) | Customer Feedback | 7.4 | 0/1 |
| <svg/onload=alert('XSS')> | No | 0 (no ejecutado) | Customer Feedback | 7.4 | 0/1 |
| <iframe src="..."> | Sí | 5 | Barra de búsqueda | 7.4 | 1/1 |
| <script>alert('XSS')</script> | No | 0 (no ejecutado) | Barra de búsqueda | 7.4 | 0/1 |
| <img src=x onerror=alert(1)> | No | 0 (no ejecutado) | Barra de búsqueda | 7.4 | 0/1 |
| <svg/onload=alert('XSS')> | Sí | 1 | Barra de búsqueda | 7.4 | 1/1 |

Si bien los vectores de XSS más comunes fueron bloqueados por el sistema, otros menos convencionales, como <iframe> y <svg>, lograron ejecutarse correctamente, en particular dentro del componente de la barra de búsqueda. Este comportamiento sugiere la existencia de mecanismos de protección parcial, aplicados de forma no uniforme a lo largo de la aplicación.

En el ataque de Falsificación de Solicitudes entre Sitios (CSRF), se evaluaron formularios maliciosos con distintos objetivos. La Tabla 4 muestra los resultados en función de la acción afectada, la ubicación del recurso, el éxito del ataque y el tiempo de ejecución.

**Tabla 4:** *Resultados de ataques CSRF (Falsificación de solicitudes entre sitios) en Juice Shop*

| Payload | ¿Éxito? | Tiempo | Ubicación | CVSS | Frecuencia |
|---|---|---|---|---|---|
| **Formulario con auto-submit para cambiar la contraseña** | Sí | 200 ms | /rest/user/change-password | 7.5 | 1/1 |
| **Formulario con intento de cambiar el nombre de usuario** | No | 0 ms | /profile | --- | 0 |

Los resultados presentados en la Tabla 4 muestran que el ataque CSRF fue exitoso al intentar cambiar la contraseña del usuario mediante un formulario con autoenvío, aprovechando una vulnerabilidad presente en la ruta /rest/user/change-password. Esta acción corresponde a un nivel de severidad alto, con una calificación CVSS de 7.5. Por otro lado, el intento de modificar el nombre de usuario a través de la ruta /profile fue bloqueado, lo que indica que dicha funcionalidad cuenta con medidas de seguridad efectivas frente a este tipo de ataque.

*RQ2: ¿Qué niveles de gravedad (CVSS) y tiempos de ejecución presentan los diferentes ataques realizados contra la aplicación vulnerable?*

La tabla 5 presenta un resumen consolidado de los ataques realizados en OWASP Juice Shop, integrando los resultados obtenidos en los cuadros anteriores. Se evaluaron cinco tipos de vulnerabilidades: inyección SQL, Cross-Site Scripting (XSS), subida de archivos maliciosos, recorrido de directorios (Directory Traversal) y falsificación de solicitudes entre sitios (CSRF). Para cada una se reporta su severidad mediante la métrica CVSS v3.1, el tiempo promedio de ejecución y si fue detectada exitosamente en el entorno analizado.

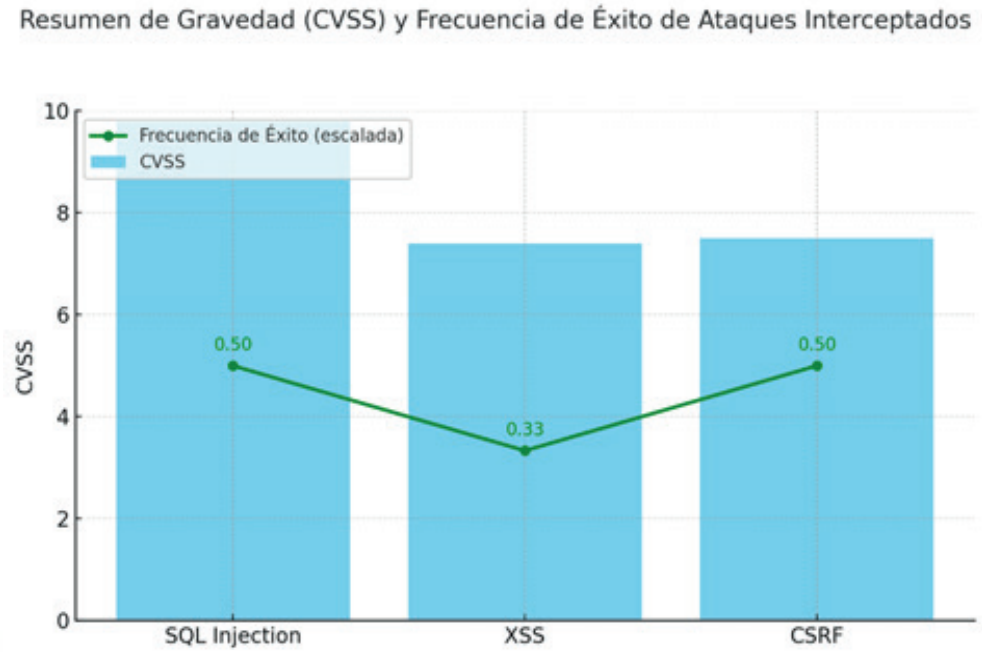
**Tabla 5:** *Gravedad (CVSS) y tiempo de ejecución de ataques*

| Ataque | CVSS v3.1 | Tiempo de Ejecución (ms) | Vulnerabilidad detectada |
|---|---|---|---|
| **SQL Injection** | 9.8 (Alta) | 15000 | Sí |
| **XSS** | 7.4 (Media) | 3 | Sí |

| | | | |
|---|---|---|---|
| **File Upload** | 8.6 (Alta) | -- | No |
| **Directory Traversal** | 7.5 (Alta) | -- | No |
| **CSRF** | 7.5 (Media) | 200 | Sí |

La Figura 3 presenta un resumen visual de las vulnerabilidades que lograron ser explotadas exitosamente durante las pruebas en OWASP Juice Shop. El análisis se enfocó en tres tipos de ataques: inyección SQL (SQL Injection), Cross-Site Scripting (XSS) y falsificación de solicitudes entre sitios (CSRF), los cuales fueron interceptados con éxito al menos una vez. Para cada caso se reporta su gravedad estimada mediante CVSS v3.1 (representada con barras) y su frecuencia de éxito relativa (proporción de payloads exitosos respecto al total) y la frecuencia de éxito fue normalizada con fines de representación visual,(representada como puntos verdes conectados).

**Figura 3:** *Resumen de gravedad (CVSS) y frecuencia de éxito de ataques interceptados en OWASP Juice Shop*




Del análisis se desprende que el ataque de inyección SQL posee la mayor severidad (CVSS 9.8), alineada con una frecuencia de éxito del 50\%, lo que sugiere que además de ser crítico, fue parcialmente viable en el entorno analizado. Por otro lado, el ataque XSS, aunque con una gravedad menor (CVSS 7.4), evidenció una frecuencia de éxito más baja (33\,\%), lo cual puede estar asociado a mecanismos de validación más robustos en ciertas rutas. En contraste, CSRF, con una severidad media (CVSS 7.5), alcanzó una frecuencia de éxito igual a la del SQLi (50\,\%), indicando una debilidad puntual en la protección contra solicitudes forjadas. Esta comparación evidencia que no siempre existe una relación directa entre la severidad teórica y la efectividad práctica de un ataque, por lo que ambos factores deben ser considerados conjuntamente al priorizar medidas de mitigación.

*RQ3: ¿Qué métricas operativas clave se pueden obtener al realizar pruebas de penetración sobre una aplicación web desplegada en Docker, y qué nos indican sobre su seguridad y desempeño?*

La Tabla 6 muestra las métricas operativas obtenidas durante las pruebas de penetración en una aplicación web desplegada en Docker. Estas métricas permiten evaluar la seguridad y el desempeño del sistema frente a distintos vectores de ataque, considerando aspectos como la frecuencia de éxito, el tiempo de respuesta y la severidad de las vulnerabilidades detectadas.

**Tabla 6:** *Métricas de los ataques exitosos en el entorno de pruebas de ataques detectados en OWASP Juice Shop*

| Métrica | Descripción | Valor | Interpretación |
|---|---|---|---|
| **Vulnerabilidades detectadas** | Tipos de ataques que lograron ejecución exitosa | 3 (SQLi, XSS, CSRF) | El entorno permitió explotación de 3 vectores distintos |
| **Frecuencia máxima de éxito** | Mayor proporción de payloads exitosos entre los intentos | 0.5 (SQLi y CSRF) | SQLi y CSRF alcanzaron el 50% de éxito en pruebas |
| **Tiempo de ejecución mínimo** | Payload con menor tiempo de respuesta registrado | 1 ms (XSS – svg/onload) | Algunos vectores de XSS responden casi instantáneamente |
| **Severidad más alta (CVSS)** | Valor CVSS más alto entre ataques detectados | 9.8 (SQL Injection) | SQLi representa un riesgo crítico |
| **Cantidad total de payloads probados** | Suma de todos los intentos de explotación registrados | 14 payloads (6 SQLi, 6 XSS, 2 CSRF) | Muestra el alcance de las pruebas en distintos vectores |

La Tabla 6 resume los aspectos más relevantes de los ataques exitosos detectados durante las pruebas. Se reportan la cantidad de vulnerabilidades explotadas, su frecuencia de éxito, tiempos de ejecución mínimos, nivel de severidad (CVSS) y el total de payloads utilizados. Estos indicadores permiten evaluar tanto el impacto como la eficacia de los vectores que lograron vulnerar el entorno.

### 4.2. Discusión

Los resultados obtenidos demuestran que SQL Injection, Cross-Site Scripting (XSS) y Cross-Site Request Forgery (CSRF) fueron explotados con éxito en OWASP Juice Shop. Esto evidencia la presencia de vulnerabilidades críticas cuando no se aplican controles de seguridad desde el desarrollo. En particular, SQL Injection representó el mayor riesgo con una severidad de CVSS 9.8 y permitió el acceso como administrador. Además, tanto SQLi como CSRF alcanzaron una frecuencia de éxito del 50 %, lo que refleja su alta efectividad en entornos vulnerables.

En el caso de XSS, a pesar de tener una menor gravedad (CVSS 7.4), el tiempo de ejecución registrado fue de solo 1 ms, lo que lo convierte en una amenaza silenciosa y difícil de detectar. Algunos vectores como <iframe> y <svg> lograron ejecutarse en la barra de búsqueda, lo que evidencia que los mecanismos de

filtrado aplicados no fueron uniformes en toda la aplicación. Por otro lado, el ataque CSRF fue exitoso al permitir el cambio de contraseña del usuario mediante un formulario con autoenvío, aprovechando la vulnerabilidad en la ruta /rest/user/change-password. Este comportamiento evidencia la ausencia de mecanismos de protección anti falsificación en dicha funcionalidad. Sin embargo, al intentar modificar el nombre de usuario a través de la ruta /profile, la solicitud fue bloqueada, lo que indica la existencia de controles más estrictos en esa sección específica.

En conjunto, estos hallazgos refuerzan la importancia de incorporar consultas preparadas, validaciones del lado del servidor y tokens CSRF en todas las rutas sensibles. Además, se valida el uso de entornos como Juice Shop para evaluar la seguridad de sistemas web mediante ataques reproducibles y medibles. Estos hallazgos refuerzan la necesidad de incorporar medidas de ciberseguridad desde etapas tempranas del desarrollo web, como parte de una estrategia de prevención continua.

## V. Conclusiones y Trabajo Futuro

Las pruebas de penetración realizadas sobre la aplicación OWASP Juice Shop en un entorno controlado permitieron identificar vulnerabilidades críticas asociadas a ataques comunes como inyección SQL, Cross-Site Scripting (XSS) y falsificación de solicitudes entre sitios (CSRF). La explotación exitosa de estas fallas evidenció la ausencia de controles robustos de validación y autenticación en ciertas rutas del sistema, lo que posibilitó accesos no autorizados y ejecución de acciones sensibles. En contraste, funcionalidades como la carga de archivos y el recorrido de directorios demostraron contar con medidas de protección efectivas ante los vectores utilizados. El análisis también permitió obtener métricas operativas relevantes, como la frecuencia de éxito de los payloads, tiempos de respuesta mínimos y niveles de severidad CVSS. Estas métricas facilitaron una evaluación comparativa de los vectores y ayudaron a visibilizar los riesgos asociados a cada uno. Se confirmó que ataques como SQLi y CSRF, con una frecuencia de éxito del 50%, representan amenazas recurrentes en aplicaciones con validaciones incompletas.

Como trabajo futuro, se propone ampliar la cobertura de pruebas mediante el uso de herramientas automatizadas para detección de vulnerabilidades, así como incorporar técnicas de fuzzing y escaneo de seguridad continua dentro del ciclo de desarrollo (DevSecOps). Asimismo, se sugiere evaluar otras plataformas diseñadas con vulnerabilidades controladas, con el fin de contrastar resultados y validar la aplicabilidad de los hallazgos en diferentes entornos.

## Referencias

Ahmadi, S. (2024). Challenges and solutions in network security for serverless computing. International *Journal of Current Science Research and Review*, 7(1), 218–229. https://dx.doi.org/10.47191/ijcsrr/V7-i1-23

Anu, P., & Vimala, S. (2018). A survey on sniffing attacks on computer networks. Proceedings of 2017 International Conference on Intelligent Computing and Control (I2C2 2017), 1–5. https://doi.org/10.1109/I2C2.2017.8321914

Augusto, C., & Agudelo, R. (2018). Arquitectura para automatizar respuesta a incidentes de seguridad de la información relacionados con ataques internos mediante la ejecución de técnicas spoofing [Trabajo académico].

Barth, A., Jackson, C., & Mitchell, J. C. (2021). Robust defenses for cross-site request forgery. *ACM Transactions on Information and System Security*, 24(1), 1–38. https://doi.org/10.1145/1455770.1455782

Bau, J., Bursztein, E., Gupta, D., & Mitchell, J. (2021). State of the art: Automated black-box web application vulnerability testing. *IEEE Symposium on Security and Privacy,* 332–345. https://doi.org/10.1109/SP.2010.27

Binbusayyis, A. (2024). Reinforcing network security: Network attack detection using random grove blend in weighted MLP layers. *Mathematics*, 12(11), 1720. https://doi.org/10.3390/MATH12111720

Chávez, J. (2011). Simulación y análisis de mecanismos de defensa ante los ataques de denegación de servicios (DoS) en redes de área local convergentes [Tesis de grado, Escuela Politécnica Nacional]. http://bibdigital.epn.edu.ec/handle/15000/4282

Echeverría, J., Espinoza, C., & Zambrano, D. (2020). Evaluación del nivel de ciberseguridad en usuarios jóvenes ecuatorianos [Estudio técnico].

Fernandes, P., Ciardhuáin, S., & Antunes, M. (2024). Unveiling malicious network flows using Benford's law. *Mathematics,* 12(15), 2299. https://doi.org/10.3390/MATH12152299

Fuertes Díaz, W. M., & Macas Carrasco, M. A. (2023). Ciberseguridad [Tesis de grado, Universidad de las Fuerzas Armadas ESPE]. http://repositorio.espe.edu.ec/handle/21000/36481

González, D. A., Pérez, M. E. G., & Bernal, L. P. (2016). Detección y mitigación de ataques ARP en la red corporativa de la división territorial Holguín, ETECSA. *Telemática,* 15(1), 62–68.

Gorgone Carvajal, A. (2023). Simulación de ciberataques con GNS3 para validar la robustez de protocolos industriales [Tesis de grado]. https://hdl.handle.net/10630/27491

Lekies, S., Stock, B., & Johns, M. (2023). 25 million flows later: Large-scale detection of DOM-based XSS. Proceedings of the 2023 ACM SIGSAC Conference, 1193–1204. https://doi.org/10.1145/2508859.2516703

Nakata, M., & Otsuka, A. (2021). Security testing environments for web services [Estudio técnico].

Rafique, S., Humayun, M., Gul, Z., & Abbas, A. (2015). Systematic review of web application security vulnerabilities detection methods. *Journal of Computer and Communications*, 3(9), 28–40. http://dx.doi.org/10.4236/jcc.2015.39004

Sadegh, M., & Tajalli, B. (2013). Study of SQL injection attacks and countermeasures. International Journal of Innovative Research in Computer and Communication Engineering. http://dx.doi.org/10.7763/IJCCE.2013.V2.244

Son, J. (2016). Virtual lab for online cyber security education. Communications of the IIMA. http://dx.doi.org/10.58729/1941-6687.1200

Straub, J. (2020). Cyberwarfare simulations: An overview of platforms and challenges [Informe técnico].

Ukwandu, E., Aghenta, L., & Okwechime, I. (2020). A systematic review of cyberattacks on critical infrastructure. [Estudio de revisión].

Vega-Oyola, C., Tapia, E., & Gallardo, F. (2022). Análisis de factores de seguridad informática mediante la metodología OWASP v4.2: Caso de estudio ISTJOL. *Revista EETES*, 6(1). http://dx.doi.org/10.33970/eetes.v6.n1.2022.293

Veeraburden, K., & Bekaroo, G. (2022). Security in web applications: A comparative analysis of key SQL injection detection techniques. 2022 4th International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM). http://dx.doi.org/10.1109/ELECOM54934.2022.9965264

Vighe, S. (2024). Security for continuous integration and continuous deployment pipeline. International Research Journal of Modernization in Engineering Technology and Science. http://dx.doi.org/10.56726/IRJMETS50676