Plan De Pruebas Seguridad -   
WebGoat

Índice

[0. Registro de Cambios 2](#_Toc73618711)

[1. Objeto 2](#_Toc73618712)

[2. Documentación 3](#_Toc73618713)

[3. Definiciones 3](#_Toc73618714)

[4. Descripción General 4](#_Toc73618715)

[**4.1.** **Dentro del alcance de las pruebas** 4](#_Toc73618716)

[**4.2.** **Fuera del alcance de las pruebas** 4](#_Toc73618717)

[**4.3.** **Objetivos** 4](#_Toc73618718)

[**4.4.** **Metodología de Pruebas** 5](#_Toc73618719)

[**4.5.** **Prerequisitos** 6](#_Toc73618720)

[4.5.1. Damn Vulnerable Web Application (DVWA) 6](#_Toc73618721)

[5. Herramientas e infraestructura de pruebas 7](#_Toc73618722)

[**5.1.** **Herramientas** 7](#_Toc73618723)

[**5.2.** **Infraestructura de pruebas** 7](#_Toc73618724)

[6. Seguimiento de defectos 7](#_Toc73618725)

[7. Entregables 8](#_Toc73618726)

[**7.1.** **Reporte Análisis estático** 8](#_Toc73618727)

[**7.2.** **Reporte análisis dinámico** 8](#_Toc73618728)

[**7.3.** **Informe resultado ejecución pruebas de seguridad** 8](#_Toc73618729)

# Registro de Cambios

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor (iniciales)** | **Versión** | **Fecha Registro** | **Cambio Realizado respecto a la versión anterior** |
| EJRN | 1 | 03/05/2021 | Versión inicial |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Objeto

Se pretende realizar pruebas de Seguridad sobre la aplicación [JuiceShop](https://github.com/bkimminich/juice-shop) con el objetivo de verificar que se cumplen los requisitos estipulados en el nivel 2 del [OWASP Application Security Verification Standard](https://github.com/OWASP/ASVS/raw/master/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0-en.pdf)

WebGoat es una aplicación insegura diseñada y mantenida por [OWASP](http://www.owasp.org/) para enseñar lecciones de seguridad sobre de aplicaciones web. JuiceShop está programada en ***JavaScript*** con los principales fallos de seguridad recogidos en el [OWASP top 10](https://owasp.org/www-project-top-ten/). Sus principales objetivos son ayudar a los profesionales de la seguridad a poner a prueba sus habilidades y herramientas en un entorno legal, ayudar a los desarrolladores web a comprender mejor los procesos de protección de las aplicaciones web y ayudar a los profesores / estudiantes a enseñar / aprender sobre seguridad de las aplicaciones web en un entorno controlado.

Este documento describe las estrategias, procesos, flujos de trabajo y metodologías que van a ser usadas para planificar, ejecutar y organizar las pruebas de seguridad.

# Documentación

Requisitos de seguridad para aplicaciones y servicios web: [OWASP Application Security Verification Standard 4.0-en](https://github.com/M0l1n3ta/PFG/blob/master/Reportes/Requisitos%20Seguridad%20-%20OWASP%20ASVS_v4.xlsx)

# Definiciones

# Descripción General

## **Dentro del alcance de las pruebas**

Las pruebas de este proyecto se abarcarán desde distintos grupos de prueba:

El análisis dinámico de la aplicación se realizará ejecutando el plan de pruebas, donde se generarán peticiones para cubrir las siguientes vulnerabilidades:

* SQL Injection
* Broken Authentication
* Sensitive Data Exposure
* Xml External Entities (XXE)
* Cross-Site Request Forgery (CSRF) attack
* Cross-Site Scripting (XSS)

## **Fuera del alcance de las pruebas**

Fuera del alcance del alcance quedan el resto de las vulnerabilidades que presenta la aplicación.

Dichas vulnerabilidades podrán ser incluidas dentro del alcance, si así se estima oportuno para futuras iteraciones a través de la actualización de este plan de pruebas.

## **Objetivos**

El principal objetivo de las pruebas de seguridad es aumentar la confianza del software bajo prueba. Las pruebas se centran en comprobar que el software no es vulnerable a los ataques más comunes en cuanto aplicaciones de software se refiere.

Un objetivo secundario sería detectar posibles defectos funcionales que pueda tener el software.

## **Metodología de Pruebas**

Una vez fijados las aplicaciones bajo prueba, y los objetivos deseados, definimos la metodología de va a ser aplicada para las pruebas de seguridad.

Dentro de las fases definidas en apartado 2.1.2 para la ejecución del proceso de pentesting ejecutaremos todas las fases menos la de explotación y postexplotación.

***Alcance y términos de la prueba de intrusión.***

Para cada una de las aplicaciones crearemos un documento definición del plan de pruebas de seguridad donde se detallará toda la información de las pruebas de seguridad a ejecutar.

***Recolección de información.***

En esta fase ejecutaremos el análisis estático de dependencias y generaremos un reporte del análisis estático de código. Los resultados del análisis estático servirán como base para crear un plan de pruebas para el análisis dinámico.

***Análisis de vulnerabilidades.***

En esta fase ejecutaremos el análisis dinámico de código a partir del plan de pruebas generado con la información obtenida en la fase anterior.

En esta fase ejecutarán dos veces el escáner de análisis dinámico con distinto número de reglas:

* **Escáner regular:** para descubrir todas las posibles rutas validados debajo de los dominios a evaluar a partir del plan de pruebas definido con los datos de la fase anterior
* **Escáner completo:** A partir del escáner regular, para obtener el reporte definitivo después de revisar los errores encontrados para descartar los no relevantes y los falsos positivos.

***Generación de informes.***

Como resultado el proceso de ejecución de las pruebas de seguridad generaremos los siguientes documentos.

* Definición del plan de pruebas de seguridad
* Reporte análisis estático de código
* Plan pruebas para el análisis dinámico
* Reporte análisis dinámico
* Informe resultado ejecución pruebas de seguridad

## **Prerrequisitos**

* + 1. JuiceShop App

La aplicación debe estar desplegada en el entorno de pruebas, en nuestro caso se ejecutará a través del siguiente docker compose:

FROM node:12 as installer

COPY . /juice-shop

WORKDIR /juice-shop

RUN npm i -g typescript ts-node

RUN npm install --production --unsafe-perm

RUN npm dedupe

RUN rm -rf frontend/node\_modules

FROM node:12-alpine

ARG BUILD\_DATE

ARG VCS\_REF

LABEL maintainer="Bjoern Kimminich <bjoern.kimminich@owasp.org>" \

    org.opencontainers.image.title="OWASP Juice Shop" \

    org.opencontainers.image.description="Probably the most modern and sophisticated insecure web application" \

    org.opencontainers.image.authors="Bjoern Kimminich <bjoern.kimminich@owasp.org>" \

    org.opencontainers.image.vendor="Open Web Application Security Project" \

    org.opencontainers.image.documentation="https://help.owasp-juice.shop" \

    org.opencontainers.image.licenses="MIT" \

    org.opencontainers.image.version="12.7.2" \

    org.opencontainers.image.url="https://owasp-juice.shop" \

    org.opencontainers.image.source="https://github.com/bkimminich/juice-shop" \

    org.opencontainers.image.revision=$VCS\_REF \

    org.opencontainers.image.created=$BUILD\_DATE

WORKDIR /juice-shop

RUN addgroup --system --gid 1001 juicer && \

    adduser juicer --system --uid 1001 --ingroup juicer

COPY --from=installer --chown=juicer /juice-shop .

RUN mkdir logs && \

    chown -R juicer logs && \

    chgrp -R 0 ftp/ frontend/dist/ logs/ data/ i18n/ && \

    chmod -R g=u ftp/ frontend/dist/ logs/ data/ i18n/

USER 1001

EXPOSE 3000

CMD ["npm", "start"]

Ejecutando el siguiente comando:

docker run --rm -p 3000:3000 bkimminich/juice-shop

# Herramientas e infraestructura de pruebas

## **Herramientas**

Entre las distintas herramientas de análisis, para la implementación de nuestra infraestructura de pruebas haremos uso de las siguientes herramientas:

* SonarQube
* Dependency-check

La herramienta seleccionada para la ejecución del análisis dinámico será **OWASP ZAP** a partir de cuyo reporte, junto con el del análisis estático, se revisarán los defectos encontrado para verificarlos o descartarlos.

## **Infraestructura de pruebas**

Como entorno de pruebas para la ejecución de los análisis de código; haremos uso de una máquina física y de un contenedor de Docker con la siguientes características y herramientas instaladas en cada una de ellas:

|  |  |  |
| --- | --- | --- |
| **Maquina** | **Física** | **Contenedor** |
| **Sistema Operativo** | Windows 10 Pro | Debian GNU/Linux 10 (buster) |
| **Herramientas** | OWASP Zap 2.10 Dependency-check  SonarScaner 4.6.2 | SonarQube 8.2  PostGresSQL 13.3 |

# Seguimiento de defectos

No se realizará seguimiento de los defectos.

# Entregables

Las actividades realizadas durante el proceso de ejecución de este plan de pruebas se generarán una serie entregables que se detallan a continuación:

## **Reporte Análisis estático**

Se entregará previo al inicio de la ejecución de las pruebas de seguridad.

## **Reporte análisis dinámico**

Se entregará previo al inicio de la ejecución de las pruebas de seguridad.

## **Informe resultado ejecución pruebas de seguridad**

Después de cada una de las intervenciones se plasmará en un documento el estado de la aplicación en cuanto a seguridad se refiere.

Este documento incluirá los resultados de las pruebas realizadas, dificultades encontradas durante la ejecución, los defectos detectados y toda información que el equipo de pruebas considere relevante.