**A logo with a person in the back

Description automatically generated with medium confidenceUNIVERSIDAD MARIANO**

**GÁLVEZ DE GUATEMALA**

**FACULTAD DE INGENIERÍA EN SISTEMAS DE INFORMACIÓN**

**MAESTRÍA EN ANÁLISIS FORENSE INFORMÁTICO**

**Proyecto Final**

**Auditoria a Aplicación Web de empresa de TAGTEL**

**Autor:**

**Katherine Nohemy Yanes Marroquín kyanesm1@miumg.edu.gt**

**Melanie Rocio Morataya Ortiz mmoratayao1@miumg.edu.gt**

**Marlon Emanuel Zamora Barrientos mzamorab@miumg.edu.gt**

**César Stev Estrada Espina cestradae@miumg.edu.gt**

**Brandon Eduardo Godinez Suret bgodinezs1@miumg.edu.gt**

**Profesor**

**Msc Ing. Denner Medrano**

**Curso**

**Auditoría Informática – 4 – 2,025**

**Guatemala 28 de noviembre, 2025**

Índice

[Introducción 1](#_Toc215820190)

[Auditoría a aplicación web 2](#_Toc215820191)

[1.1 Descripción del caso de auditoria 2](#_Toc215820192)

[1.2 Identificación del área a auditar 3](#_Toc215820193)

[1.3 Identificación del Riesgo 3](#_Toc215820194)

[1.4 Responsables de la Auditoria 4](#_Toc215820195)

[1.4.1 Líder de Auditoría / Auditor Senior 4](#_Toc215820196)

[1.4.2 Auditor Técnico de Aplicaciones (Especialista en OWASP / Pruebas de Seguridad) 4](#_Toc215820197)

[1.4.3 Analista de Seguridad / Evaluador de Controles 4](#_Toc215820198)

[1.4.4 Analista de Riesgos (Metodología COSO ERM) 4](#_Toc215820199)

[1.4.5 Documentador / Coordinador de Evidencias 4](#_Toc215820200)

[1.5 Marco Legal / Normativo 5](#_Toc215820201)

[1.6 Tecnologías utilizadas 5](#_Toc215820202)

[1.7 Cuestionario de Control Interno 5](#_Toc215820203)

[1.8 Matriz de riesgos 6](#_Toc215820204)

[1.9 Mapa de calor 8](#_Toc215820205)

[1.10 Fundamento de la Auditoría 8](#_Toc215820206)

[1.11 Objetivos de la Auditoría 9](#_Toc215820207)

[1.11.1 General 9](#_Toc215820208)

[1.11.2 Específicos 9](#_Toc215820209)

[1.12 Alcance de la auditoría 9](#_Toc215820210)

[1.13 Recurso Humano 9](#_Toc215820211)

[1.13.1 Líder de Auditoría / Auditor Senior 9](#_Toc215820212)

[1.13.2 Auditor Técnico de Aplicaciones (SAST/DAST) 9](#_Toc215820213)

[1.13.3 Analista de Seguridad / Evaluador de Controles 10](#_Toc215820214)

[1.13.4 Analista de Riesgos 10](#_Toc215820215)

[1.13.5 Documentador / Coordinador de Evidencias 10](#_Toc215820216)

[1.14 Recursos Materiales y Tecnológicos 10](#_Toc215820217)

[1.14.1 Hardware 10](#_Toc215820218)

[1.14.2 Software y Herramientas 10](#_Toc215820219)

[1.15 Programa de auditoria 11](#_Toc215820220)

[Ejecución de la auditoría 12](#_Toc215820221)

[1.1 Pruebas de Consentimiento (Hacking Ético / Pentesting) 12](#_Toc215820222)

[1.1.1 Introducción a las pruebas de consentimiento 12](#_Toc215820223)

[1.2 Metodología aplicada 13](#_Toc215820224)

[1.2.1 Planeación técnica 13](#_Toc215820225)

[1.2.2 Reconocimiento Pasivo 14](#_Toc215820226)

[1.2.3 Reconocimiento Activo 14](#_Toc215820227)

[1.2.4 Enumeración y Mapeo Funcional 14](#_Toc215820228)

[1.2.5 Ejecución de Pruebas OWASP Top 10 15](#_Toc215820229)

[1.2.6 Pruebas de Autenticación y Gestión de Sesiones 15](#_Toc215820230)

[1.2.7 Pruebas de Manipulación de Datos y Entradas 16](#_Toc215820231)

[1.2.8 Pruebas de Consentimiento de Alto Riesgo 16](#_Toc215820232)

[1.2.9 Análisis de Configuraciones del Servidor 16](#_Toc215820233)

[1.2.10 Registro, Evidencia y Documentación 17](#_Toc215820234)

[1.2.11 1Análisis de Riesgo Posterior 17](#_Toc215820235)

[1.2.12 Determinación de Hallazgos 17](#_Toc215820236)

[1.3 Resultados de las pruebas 18](#_Toc215820237)

[1.3.1 Vulnerabilidades de Validación de Entradas 18](#_Toc215820238)

[1.3.2 Debilidades en Autenticación y Controles Anti-Automatización 19](#_Toc215820239)

[1.3.3 Exposición de Información Sensible 19](#_Toc215820240)

[1.3.4 Falta de Controles Adecuados en Gestión de Sesiones 19](#_Toc215820241)

[1.3.5 Configuraciones Inseguras del Servidor 20](#_Toc215820242)

[1.3.6 Enumeración de Endpoints y Falta de Restricción de Accesos 20](#_Toc215820243)

[1.3.7 Conclusión General de las Pruebas 21](#_Toc215820244)

[1.4 Listado de pruebas ejecutadas 21](#_Toc215820245)

[1.4.1 Checklist de Pruebas Realizadas (Basado en OWASP Testing Guide v4) 21](#_Toc215820246)

[1.4.2 Interpretación del Checklist 23](#_Toc215820247)

[Papeles de Trabajo 24](#_Toc215820249)

[2.1 PT-01 – Reconocimiento Inicial (Fingerprinting) 24](#_Toc215820250)

[2.1.1 Objetivo de la Prueba 24](#_Toc215820251)

[2.1.2 Alcance 25](#_Toc215820252)

[2.1.3 Procedimiento 25](#_Toc215820253)

[2.1.4 Fingerprinting del servidor con curl 25](#_Toc215820254)

[2.1.5 Detección de tecnologías con WhatWeb / Wappalyzer 26](#_Toc215820255)

[2.1.6 Escaneo no intrusivo con Nmap 26](#_Toc215820256)

[2.1.7 Revisión del certificado SSL 26](#_Toc215820257)

[2.1.8 Revisión del DOM y recursos mediante DevTools 27](#_Toc215820258)

[2.1.9 Resultado de la Prueba 28](#_Toc215820259)

[2.1.10 Criterio 28](#_Toc215820260)

[2.1.11 Análisis 29](#_Toc215820261)

[2.1.12 Conclusión 29](#_Toc215820262)

[2.1.13 Recomendación 29](#_Toc215820263)

[2.1.14 Clasificación del Riesgo 29](#_Toc215820264)

[2.2 PT-02 – Enumeración de Endpoints 30](#_Toc215820265)

[2.2.1 Objetivo de la Prueba 30](#_Toc215820266)

[2.2.2 Alcance 30](#_Toc215820267)

[2.2.3 Procedimiento 30](#_Toc215820268)

[2.2.4 Resultado de la Prueba 33](#_Toc215820269)

[2.2.5 Criterio 34](#_Toc215820270)

[2.2.6 Análisis 34](#_Toc215820271)

[2.2.7 Conclusión 34](#_Toc215820272)

[2.2.8 Recomendación 34](#_Toc215820273)

[2.2.9 Clasificación del Riesgo 35](#_Toc215820274)

[2.3 PT-03 – Pruebas de XSS (Reflejado y Almacenado) 35](#_Toc215820275)

[2.3.1 Objetivo de la Prueba 35](#_Toc215820276)

[2.3.2 Alcance 35](#_Toc215820277)

[2.3.3 Procedimiento 36](#_Toc215820278)

[2.3.4 Resultado de la Prueba 40](#_Toc215820279)

[2.3.5 Criterio 41](#_Toc215820280)

[2.3.6 Análisis 41](#_Toc215820281)

[2.3.7 Conclusión 41](#_Toc215820282)

[2.3.8 Recomendación 41](#_Toc215820283)

[2.3.9 Clasificación del Riesgo 42](#_Toc215820284)

[2.4 PT-04 – Pruebas de Autenticación y Fuerza Bruta 42](#_Toc215820285)

[2.4.1 Objetivo de la Prueba 42](#_Toc215820286)

[2.4.2 Alcance 43](#_Toc215820287)

[2.4.3 Procedimiento 43](#_Toc215820288)

[2.4.4 Resultado de la Prueba 47](#_Toc215820289)

[2.4.5 Criterio 47](#_Toc215820290)

[2.4.6 Análisis 48](#_Toc215820291)

[2.4.7 Conclusión 48](#_Toc215820292)

[2.4.8 Recomendación 48](#_Toc215820293)

[2.4.9 Clasificación del Riesgo 48](#_Toc215820294)

[2.5 PT-05 – Pruebas de SQL Injection (Simulada y de Manipulación de Parámetros) 49](#_Toc215820295)

[2.5.1 Objetivo de la Prueba 49](#_Toc215820296)

[2.5.2 Alcance 49](#_Toc215820297)

[2.5.3 Procedimiento 50](#_Toc215820298)

[2.5.4 Resultado de la Prueba 54](#_Toc215820299)

[2.5.5 Criterio 54](#_Toc215820300)

[2.5.6 Análisis 54](#_Toc215820301)

[2.5.7 Conclusión 55](#_Toc215820302)

[2.5.8 Recomendación 55](#_Toc215820303)

[2.5.9 Clasificación del Riesgo 55](#_Toc215820304)

[2.6 PT-06 – Configuraciones Inseguras del Servidor (Headers, CSP, HSTS) 56](#_Toc215820305)

[2.6.1 Objetivo de la Prueba 56](#_Toc215820306)

[2.6.2 Alcance 56](#_Toc215820307)

[2.6.3 Procedimiento 57](#_Toc215820308)

[2.6.4 Resultado de la Prueba 59](#_Toc215820309)

[2.6.5 Criterio 60](#_Toc215820310)

[2.6.6 Análisis 60](#_Toc215820311)

[2.6.7 Conclusión 60](#_Toc215820312)

[2.6.8 Recomendación 60](#_Toc215820313)

[2.7 PT-07 – IDOR y Manipulación de Parámetros (Access Control) 61](#_Toc215820314)

[2.7.1 Objetivo de la Prueba 62](#_Toc215820315)

[2.7.2 Alcance 62](#_Toc215820316)

[2.7.3 Procedimiento 62](#_Toc215820317)

[2.7.4 Resultado de la Prueba 65](#_Toc215820318)

[2.7.5 Criterio 66](#_Toc215820319)

[2.7.6 Análisis 66](#_Toc215820320)

[2.7.7 Conclusión 66](#_Toc215820321)

[2.7.8 Recomendación 66](#_Toc215820322)

[2.7.9 Clasificación del Riesgo 67](#_Toc215820323)

[2.8 PT-08 – Análisis de Sesiones y Cookies (HttpOnly, Secure, SameSite, JWT) 67](#_Toc215820324)

[2.8.1 Objetivo de la Prueba 67](#_Toc215820325)

[2.8.2 Alcance 68](#_Toc215820326)

[2.8.3 Procedimiento 68](#_Toc215820327)

[2.8.4 Resultado de la Prueba 73](#_Toc215820328)

[2.8.5 Criterio 74](#_Toc215820329)

[2.8.6 Análisis 74](#_Toc215820330)

[2.8.7 Conclusión 75](#_Toc215820331)

[2.8.8 Recomendación 75](#_Toc215820332)

[2.8.9 Clasificación del Riesgo 75](#_Toc215820333)

# Introducción

# Auditoría a aplicación web

## Descripción del caso de auditoria

La empresa **Telecomunicaciones Avanzadas de Guatemala S.A. (TAGTEL)** es una compañía líder en el sector telecomunicaciones, dedicada a proveer servicios de internet residencial y corporativo, telefonía móvil, soluciones de fibra óptica y servicios empresariales administrados. Con una cartera de más de 500,000 clientes en todo el país, TAGTEL enfrenta un crecimiento acelerado que exige altos estándares de continuidad y seguridad informática.

En los últimos años, la empresa ha fortalecido su infraestructura tecnológica implementando plataformas digitales para mejorar sus procesos operativos. Entre ellas destaca la **Aplicación Web de Gestión Operativa (AWGO),** un sistema crítico utilizado por el personal técnico y administrativo para:

* Registrar incidencias y solicitudes de clientes.
* Consultar información de redes y nodos de servicio.
* Gestionar órdenes de instalación y mantenimiento.
* Administrar datos sensibles de usuarios y equipos.

Debido a su importancia, **AWGO** se considera un activo esencial para la continuidad del negocio.

Durante una evaluación interna reciente, el departamento de Tecnología identificó posibles riesgos relacionados con la seguridad de la aplicación web, entre ellos:

* Controles de autenticación que no cumplen estándares actuales.
* Configuraciones inconsistentes en la infraestructura que soporta la aplicación.
* Falta de validaciones robustas en ciertos módulos que podrían exponer la aplicación a vulnerabilidades OWASP Top 10.
* Acceso a información sensible sin trazabilidad adecuada.

Ante estos hallazgos preliminares, TAGTEL contrato a su equipo de Auditoría Informática ha solicitado la realización de una **Auditoría Integral de Seguridad a la Aplicación Web AWGO**, con el propósito de:

1. Identificar vulnerabilidades técnicas y de control interno.
2. Evaluar el cumplimiento de estándares internacionales (OWASP ASVS, ISO 27001).
3. Determinar el nivel de riesgo asociado al uso de la aplicación web.
4. Proponer acciones correctivas que garanticen la protección de la información corporativa y de los clientes.

Esta auditoría permitirá a TAGTEL asegurar que AWGO mantiene un nivel de seguridad adecuado frente a las amenazas actuales, evitando interrupciones en el servicio, fugas de información o impacto reputacional.

## Identificación del área a auditar

* **Área:** Auditoría Integral de Seguridad a la Aplicación Web AWGO
* **Departamento:** TI – Desarrollo y Seguridad de Software
* **Período Por Evaluar:** Enero – noviembre 2025

## Identificación del Riesgo

* Secuestro de sesión
* Exfiltración/alteración de Base de Datos
* Exposición de datos
* Bypass autenticación/enum
* Deface/engaño
* Fuerza bruta/scraping
* XSS/clickjacking/downgrade
* Exploits por librerías
* Account takeover
* Troyanización
* Acceso a red interna/metadata cloud
* Robo de sesión/deface
* Robo de sesión
* Robo de tokens
* Acciones no autorizadas

## Responsables de la Auditoria

### Líder de Auditoría / Auditor Senior

**Responsable (Melanie):**

* Dirigir la planificación y ejecución de la auditoría.
* Coordinar actividades, cronograma y comunicación con áreas auditadas.
* Revisar los hallazgos y validar la calidad del informe final.

### Auditor Técnico de Aplicaciones (Especialista en OWASP / Pruebas de Seguridad)

**Responsable (Brandon Godinez):**

* Ejecutar pruebas técnicas (SAST, DAST).
* Identificar y analizar vulnerabilidades OWASP TOP 10.
* Documentar evidencia técnica y realizar pruebas de explotación controlada.

### Analista de Seguridad / Evaluador de Controles

**Responsable (Cesar):**

* Revisar políticas internas, controles de acceso y segregación de funciones.
* Validar el cumplimiento de normas ISO 27001 y OWASP ASVS.
* Evaluar la madurez de los controles y registrar observaciones.

### Analista de Riesgos (Metodología COSO ERM)

**Responsable (Katherine Yanes):**

* Construir la matriz de riesgos y controles.
* Determinar niveles de probabilidad, impacto y criticidad.
* Priorizar riesgos y sugerir controles adicionales.

### Documentador / Coordinador de Evidencias

**Responsable (Emanuel Zamora):**

* Consolidar evidencias, cuestionarios, hallazgos y anexos.
* Elaborar borradores del informe final y presentaciones.
* Llevar control del repositorio de documentación y versiones.

## Marco Legal / Normativo

* Ley de Protección de Datos Personales de Guatemala
* ISO/IEC 27001:2022
* OWASP Application Security Verification Standard (ASVS)
* Políticas internas de desarrollo seguro

## Tecnologías utilizadas

* Backend: Node.js
* Frontend: React
* Base de datos: PostgreSQL
* Servidor: Linux
* Control de versiones: GitHub
* Autenticación: OAuth2

## Cuestionario de Control Interno

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pregunta | Sí | No | N/A | Evidencia |
| ¿La aplicación implementa MFA para cuentas administrativas? |  |  |  |  |
| ¿Se realizan pruebas de seguridad periódicas (SAST/DAST)? |  |  |  |  |
| ¿Existe control de cambios documentado? |  |  |  |  |
| ¿Se gestionan vulnerabilidades mediante un ciclo formal? |  |  |  |  |
| ¿La base de datos aplica cifrado en reposo? |  |  |  |  |

## Matriz de riesgos

Para la construcción de la matriz de riesgos se utilizó un enfoque basado en estándares reconocidos de gestión de riesgos, entre ellos ISO 31000, ISO/IEC 27005 y el modelo de evaluación de riesgos de OWASP Risk Rating, con el fin de contar con un método objetivo y coherente para la valoración de los riesgos asociados a la aplicación web auditada.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Riesgo | Probabilidad | Impacto | Nivel | Control Actual | Control Recomendado |
| 1. Inyección SQL | Alta | Alta | Crítico | Validaciones básicas en formularios | Implementar ORM seguro, sanitización estricta, consultas parametrizadas, activar WAF |
| 2. XSS (Cross-Site Scripting) | Media | Alta | Alto | Sanitización parcial | Implementar CSP, escape estricto, librerías de sanitización, validación en frontend y backend |
| 3. Bypass de autenticación | Alta | Alta | Crítico | Autenticación estándar sin MFA | MFA obligatorio, bloqueo por intentos, detección de anomalías, rate limiting |
| 4. Secuestro de sesión | Media | Alta | Alto | Cookies sin protección avanzada | Cookies HttpOnly/Secure, regeneración de sesión, validación IP/UA, expiración corta |
| 5. Fuerza bruta / Password Spraying | Alta | Media | Alto | Solo contraseña | Rate limiting, bloqueo progresivo, MFA, CAPTCHA, monitoreo de intentos |
| 6. Exposición de datos sensibles | Media | Alta | Alto | Cifrado parcial | Cifrado completo en reposo (AES-256) y en tránsito (TLS 1.3), DLP, clasificación de datos |
| 7. Acciones no autorizadas / CSRF | Media | Media | Medio | Validaciones básicas | Tokens CSRF, SameSite cookies, control de origen/referer, revisiones ASVS nivel 2 |
| 8. Dependencias vulnerables (librerías) | Alta | Media | Alto | Actualizaciones manuales | SCA automatizado, Dependabot/GitLab Scanner, pipeline con revisión de CVEs |
| 9. Account Takeover (robo de cuenta) | Media | Alta | Alto | Monitoreo limitado | MFA obligatorio, alertas por actividad sospechosa, validación de identidad en cambios críticos |
| 10. Backups deficientes / pérdida de datos | Baja | Alta | Medio | Backups semanales | Backups diarios, verificación programada, réplica fuera del sitio, pruebas de restauración |

## Mapa de calor



## Fundamento de la Auditoría

La auditoría presentada se fundamenta en lo siguiente:

* ISO/IEC 27001 (A.8, A.12, A.14)
* OWASP ASVS
* Políticas internas de seguridad y desarrollo seguro
* Marco de riesgos
* Compliance con la Ley de Datos Personales

## Objetivos de la Auditoría

### General

Evaluar los controles de seguridad implementados en la **Aplicación Web AWGO** para determinar si son adecuados y eficaces en la protección de datos y continuidad operativa.

### Específicos

* Identificar vulnerabilidades mediante análisis documentales y técnicos.
* Evaluar el cumplimiento de estándares OWASP.
* Comprobar la correcta segregación de funciones.
* Verificar la protección de datos sensibles en tránsito y en reposo.

## Alcance de la auditoría

**Incluye:**

* Revisión de arquitectura de la aplicación
* Evaluación de controles de autenticación, autorización y cifrado
* Revisión de logs y mecanismos de monitoreo
* Evaluación de base de datos y configuración del servidor

**No incluye:**

* Pruebas de ingeniería social
* Revisión física del centro de datos

## Recurso Humano

El equipo de auditoría estará conformado por cinco integrantes, distribuidos de la siguiente manera:

### Líder de Auditoría / Auditor Senior

Responsable de la coordinación, planificación y supervisión de toda la auditoría.

### Auditor Técnico de Aplicaciones (SAST/DAST)

Encargado de la ejecución de pruebas de seguridad basadas en OWASP.

### Analista de Seguridad / Evaluador de Controles

Responsable de evaluar el cumplimiento normativo y los controles implementados.

### Analista de Riesgos

Encargado de la elaboración de la matriz de riesgos y controles.

### Documentador / Coordinador de Evidencias

Encargado de la gestión documental, control de evidencias y redacción del informe.

## Recursos Materiales y Tecnológicos

### Hardware

* Laptops con capacidad para ejecutar entornos de pruebas y herramientas de auditoría.
* Servidor de pruebas o entorno de staging provisto por la empresa auditada.

### Software y Herramientas

**Herramientas a utilizar:**

* OWASP ZAP – Pruebas DAST
* Kali Linux – Entorno para escaneos de seguridad
* Nmap / Nikto / Wappalyzer – Recolección de información
* GitHub / GitLab – Acceso a repositorio de código (modo lectura)
* Herramientas de ofimática: Word, Excel, PowerPoint

**Documentación de referencia:**

* Políticas internas de seguridad
* Manuales de arquitectura
* Diagramas de componentes y flujo de información
* Estándares OWASP y control de versiones de la aplicación
* Documentación interna

## Programa de auditoria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nº | Actividad | Responsable Principal | Participantes | Fecha |
| 1 | Reunión inicial y entendimiento del área (kick-off) | Líder de Auditoría | Todo el equipo | 15/01 |
| 2 | Revisión documental (arquitectura, políticas, manuales) | Analista de Seguridad | Documentador | 15/01 |
| 3 | Identificación preliminar de riesgos | Analista de Riesgos | Líder de Auditoría | 16/01 |
| 4 | Aplicación del cuestionario de control interno | Analista de Seguridad | Líder de Auditoría | 16/01 |
| 5 | Configuración del entorno de pruebas | Auditor Técnico | Documentador | 16/01 |
| 6 | Ejecución de pruebas SAST (revisión de código) | Auditor Técnico | Analista de Seguridad | 17/01 |
| 7 | Ejecución de pruebas DAST (OWASP ZAP / Burp Suite) | Auditor Técnico | Analista de Seguridad | 17–19/01 |
| 8 | Registro y consolidación de evidencias | Documentador | Todo el equipo | 17–19/01 |
| 9 | Análisis de hallazgos técnicos | Auditor Técnico | Líder de Auditoría | 19/01 |
| 10 | Elaboración de matriz de riesgos y controles | Analista de Riesgos | Líder de Auditoría | 20/01 |
| 11 | Reunión interna para validar resultados | Líder de Auditoría | Todo el equipo | 20/01 |
| 12 | Elaboración del informe preliminar | Documentador | Líder de Auditoría | 21/01 |
| 13 | Presentación del informe a la gerencia | Líder de Auditoría | Todo el equipo | 21/01 |

# Ejecución de la auditoría

## Pruebas de Consentimiento (Hacking Ético / Pentesting)

### Introducción a las pruebas de consentimiento

Las pruebas de consentimiento constituyen una fase crítica dentro de la ejecución de una auditoría informática, ya que permiten evaluar el nivel real de exposición de una aplicación ante ataques técnicos.

Para la presente auditoría, se empleó un entorno aislado de pruebas que replica condiciones y vulnerabilidades comunes en aplicaciones web modernas. Este ambiente controlado permite ejecutar ataques sin afectar sistemas productivos, siguiendo las buenas prácticas establecidas por:

* OWASP Testing Guide v4
* OWASP Application Security Verification Standard (ASVS)
* ISO/IEC 27001:2022 – Controles A.5, A.8, A.12, A.13 y A.14
* ISO/IEC 19011 – Directrices para auditoría de sistemas de gestión

El objetivo principal de estas pruebas es determinar si los controles implementados por la aplicación son suficientes para mitigar riesgos asociados a ataques habituales, tales como:

* Inyección de código
* Manipulación de sesiones
* Fuerza bruta y bypass de autenticación
* Exposición indebida de información
* Configuraciones inseguras
* Falta de validación de entradas
* Vulnerabilidades propias del OWASP Top 10

Las pruebas se desempeñaron bajo los principios de:

1. Legalidad — Ejecución únicamente en sistemas autorizados
2. Ética — Sin afectar datos reales o infraestructura operativa
3. Repetibilidad — Documentando cada procedimiento
4. Objetividad — Entregando evidencia verificable

El alcance técnico aprobado incluye:

* Validación de mecanismos de autenticación
* Evaluación de gestión de sesiones y tokens
* Análisis de encabezados de seguridad
* Pruebas de inyección y validación de entrada
* Evaluación de controles anti-automatización y rate limiting
* Exploración de endpoints accesibles y rutas expuestas
* Simulación de ataques de escalamiento de privilegios y exposición de datos

Los resultados obtenidos de estas pruebas permiten establecer el nivel de madurez de seguridad de la aplicación, identificar fallas en los controles existentes y generar hallazgos de auditoría respaldados con evidencia técnica. Todas las actividades se registraron mediante papeles de trabajo, los cuales incluyen procedimientos, capturas, logs, outputs de herramientas y conclusiones del auditor técnico.

## Metodología aplicada

Para la ejecución de las pruebas de consentimiento se adoptó una metodología estructurada basada en estándares internacionales de auditoría y seguridad de aplicaciones. Esta metodología garantiza que las pruebas se realicen de forma ética, controlada, repetible y con un alcance definido, permitiendo obtener evidencia suficiente y apropiada para sustentar los hallazgos identificados.

La metodología empleada se compone de las siguientes fases:

### Planeación técnica

En esta fase se estableció:

* El alcance de las pruebas
* Los límites permitidos
* Las herramientas aprobadas
* El ambiente controlado de pruebas
* Los criterios de aceptación y confidencialidad

Se confirmó que las pruebas no afectarían sistemas productivos, y que toda la ejecución se realizaría bajo entornos aislados autorizados.

### Reconocimiento Pasivo

Objetivo

recopilar información sin interactuar directamente con la aplicación. Incluye:

* Fingerprinting de tecnologías (headers, frameworks, versiones)
* Identificación de puertos expuestos
* Descubrimiento de componentes del stack
* Revisión de configuraciones de respuesta HTTP
* Análisis del uso de TLS y certificados

Herramientas: curl, WhatWeb, Wappalyzer, Nmap (sin flags intrusivos), DevTools.

### Reconocimiento Activo

Tiene como objetivo descubrir la estructura interna de la aplicación.

Actividades realizadas:

* Enumeración de endpoints accesibles
* Identificación de parámetros vulnerables
* Detección de formularios de entrada
* Identificación de rutas administrativas
* Pruebas de error-handling y mensajes expuestos

Herramientas: OWASP ZAP Spider, Burp Suite Repeater, Nmap avanzado.

### Enumeración y Mapeo Funcional

Se analizó la funcionalidad para comprender:

* Flujos de autenticación
* Gestión de sesiones
* Roles de usuario
* Parámetros sensibles
* Lógica de negocio expuesta
* Validaciones del lado cliente y del lado servidor

Esto sirve para identificar áreas críticas susceptibles a explotación.

### Ejecución de Pruebas OWASP Top 10

Las pruebas se basaron en los controles del OWASP Testing Guide v4, abordando las vulnerabilidades más comunes:

* A01 – Broken Access Control
* A02 – Cryptographic Failures
* A03 – Injection (SQLi, XSS)
* A04 – Insecure Design
* A05 – Security Misconfiguration
* A07 – Identification and Authentication Failures
* A08 – Software and Data Integrity Failures

Cada prueba fue documentada en un papel de trabajo independiente (PT-01 a PT-08).

### Pruebas de Autenticación y Gestión de Sesiones

Validaciones realizadas:

* Resistencia a fuerza bruta
* Reset de contraseñas
* Reutilización de tokens
* Expiración de sesión
* Correcto uso de atributos de cookies:
* HttpOnly
* Secure
* SameSite
* Path
* Regeneración de sesión en login/logout

### Pruebas de Manipulación de Datos y Entradas

Incluyen:

* Inyección SQL simulada
* Inyección de scripts (XSS almacenado y reflejado)
* Manipulación de parámetros en URL
* Envío de payloads especialmente diseñados
* Validación insuficiente del lado servidor

Herramientas: Burp Suite, ZAP, XSS CheatSheet Payloads.

### Pruebas de Consentimiento de Alto Riesgo

Realizadas únicamente en ambiente controlado:

* CSRF no protegido
* Disclosure de información sensible
* Acceso directo a objetos (IDOR)
* Escalación horizontal o vertical de privilegios
* Manipulación de JWT o tokens inseguros

### Análisis de Configuraciones del Servidor

Revisión de:

* Headers de seguridad
* Versiones expuestas
* CORS
* Directorios sin protección
* Archivos sensibles accesibles
* Políticas de caché
* Configuración HTTPS/TLS

Herramientas: curl, SSL Labs (local), ZAP Passive Scan.

### Registro, Evidencia y Documentación

Cada prueba realizada generó un Papel de Trabajo (PT) que documenta:

* Objetivo
* Procedimiento paso a paso
* Herramienta utilizada
* Capturas de pantalla
* Resultados
* Log de la prueba
* Conclusión técnica
* Nivel de riesgo
* Recomendación OWASP/ISO

Los papeles de trabajo se almacenaron siguiendo criterios de:

* Trazabilidad
* Integridad
* Reproducibilidad
* Organización lógica

### 1Análisis de Riesgo Posterior

Los resultados se mapearon contra:

* Controles ISO 27001
* OWASP ASVS
* Riesgos inherentes del sistema
* Impacto operativo simulado
* Probabilidad y severidad

### Determinación de Hallazgos

Con base en la evidencia técnica:

* Se identificaron vulnerabilidades verificadas
* Se clasificaron según severidad
* Se alinearon con criterios de auditoría
* Se prepararon para el Informe Final

## Resultados de las pruebas

Las pruebas de consentimiento permitieron identificar vulnerabilidades, debilidades en controles aplicados y configuraciones deficientes dentro del entorno evaluado. Las actividades realizadas incluyeron pruebas manuales y automatizadas, siguiendo técnicas del OWASP Testing Guide v4 y criterios del OWASP ASVS. Los resultados obtenidos proporcionan evidencia suficiente, competente y adecuada para determinar el nivel de exposición de la aplicación ante ataques comunes que afectan aplicaciones web modernas.

Los hallazgos observados durante la ejecución de las pruebas pueden agruparse en las siguientes categorías:

### Vulnerabilidades de Validación de Entradas

Durante las pruebas se identificaron escenarios donde la aplicación permite la ejecución de código malicioso o manipulación de parámetros debido a una validación insuficiente del lado servidor.

#### Vulnerabilidades encontradas

* XSS reflejado en campos de entrada sin sanitización.
* XSS almacenado, ejecutándose código en secciones posteriores.
* Manipulación de parámetros sin controles que limiten contenido no autorizado.

#### Riesgo asociado

* Robo de sesiones
* Modificación del DOM
* Phishing interno
* Secuestro de cuentas
* Defacement del sitio

### Debilidades en Autenticación y Controles Anti-Automatización

Las pruebas realizadas sobre el mecanismo de autenticación mostraron ausencia de controles esenciales que mitigan ataques automatizados.

#### Resultados

* Fuerza bruta posible: el sistema no implementa rate limiting.
* Mensajes de error diferenciados, facilitando enumeración de usuarios.
* Falta de MFA, aun para accesos con impacto crítico.

#### Riesgo asociado

* Compromiso de credenciales
* Ataques de diccionario y credential stuffing
* Exposición de usuarios válidos

### Exposición de Información Sensible

Las configuraciones del servidor permiten revelar información innecesaria al cliente, lo cual incrementa la superficie de ataque.

#### Evidencia detectada

* Encabezados que exponen versión del framework y del servidor.
* Respuestas de error que devuelven información del stack interno.
* Endpoints accesibles sin autenticación, mostrando datos operativos.

#### Riesgo asociado

* Reconocimiento para ataques dirigidos
* Ingeniería inversa sobre el comportamiento del sistema
* Aprovechamiento de vulnerabilidades específicas según versión

### Falta de Controles Adecuados en Gestión de Sesiones

Se evaluó la seguridad de cookies y tokens generados por el sistema.

#### Resultados

* Cookies sin atributos HttpOnly, Secure o SameSite.
* Tokens sin expiración adecuada o regeneración tras login/logout.
* Posibilidad de reutilización de tokens capturados.

#### Riesgo asociado

* Secuestro de sesión
* Suplantación de identidad
* Acceso no autorizado sostenido

### Configuraciones Inseguras del Servidor

El análisis de configuraciones mostró ausencia de mecanismos de protección a nivel de infraestructura.

#### Observaciones

* Falta de políticas CSP, HSTS, X-Frame-Options y X-Content-Type-Options.
* Respuestas HTTP sin directivas de seguridad relevantes.
* Errores 500 y 404 revelando estructura interna del sistema.

#### Riesgo asociado

* Clickjacking
* Downgrade attacks
* Mixed content
* Exposición involuntaria de archivos o rutas internas

### Enumeración de Endpoints y Falta de Restricción de Accesos

Las herramientas de spidering identificaron rutas accesibles que no deberían ser públicas.

#### Resultados principales

* Endpoints administrativos detectados sin autenticación previa.
* Rutas internas retornando respuestas completas.
* Parámetros alterables para obtener más información de la necesaria.

#### Riesgo asociado

* IDOR (Insecure Direct Object Reference)
* Acceso no autorizado a datos
* Exposición de estructuras internas de la aplicación

### Conclusión General de las Pruebas

Las pruebas realizadas demuestran que la aplicación evaluada presenta múltiples vulnerabilidades críticas, principalmente relacionadas con:

* Sanitización insuficiente de entradas
* Gestión insegura de sesiones
* Ausencia de controles anti-automatización
* Configuraciones deficientes a nivel de servidor
* Exposición excesiva de información

Estos hallazgos serán documentados en detalle dentro de los Papeles de Trabajo (PT-01 al PT-08), donde se incluyen evidencias técnicas, capturas, análisis, criterios de auditoría y recomendaciones específicas.

## Listado de pruebas ejecutadas

Las pruebas de consentimiento fueron realizadas siguiendo el OWASP Testing Guide v4, abarcando las principales categorías de vulnerabilidades que afectan aplicaciones web modernas. El siguiente checklist resume cada una de las pruebas ejecutadas, el objetivo de la prueba, y el resultado obtenido durante la evaluación del entorno controlado.

### Checklist de Pruebas Realizadas (Basado en OWASP Testing Guide v4)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID Prueba (OWASP) | Descripción de la Prueba | Objetivo | Resultado | Severidad |
| OTG-INFO-001 | Fingerprinting del servidor y tecnologías | Identificar stack tecnológico y superficie de ataque | Vulnerabilidades detectadas en headers y versiones visibles | Medio |
| OTG-INFO-002 | Enumeración de endpoints | Descubrir rutas accesibles y no controladas | Endpoints expuestos y sin autenticación | Alto |
| OTG-AUTHN-001 | Bypass de autenticación | Intentar acceder sin credenciales a áreas privadas | Acceso parcial obtenido | Alto |
| OTG-AUTHN-002 | Fuerza bruta | Evaluar ausencia de rate limiting | Vulnerable; no existe bloqueo automático | Alto |
| OTG-SESS-001 | Análisis de cookies y tokens | Verificar atributos de seguridad (HttpOnly, Secure, SameSite) | Cookies y tokens inseguros | Crítico |
| OTG-SESS-002 | Fijación de sesión | Validar si la sesión se regenera tras autenticación | No se regenera, permite session fixation | Alto |
| OTG-INPVAL-001 | XSS reflejado | Validar sanitización de entradas | Vulnerable a XSS | Alto |
| OTG-INPVAL-002 | XSS almacenado | Validar persistencia de payloads maliciosos | Vulnerabilidad confirmada | Crítico |
| OTG-INPVAL-005 | SQL Injection (simulada) | Evaluar validación de parámetros y consultas | Inyección parcial posible en parámetros | Alto |
| OTG-CONFIG-001 | Security Misconfigurations | Revisar headers, políticas, versiones expuestas | Headers críticos ausentes (CSP, HSTS, etc.) | Alto |
| OTG-CONFIG-002 | Directory Browsing | Intentar acceso no autorizado a archivos internos | Directorios accesibles | Medio |
| OTG-ERR-001 | Manejo de errores | Confirmar si el servidor expone información sensible | Exposición de mensajes de error detallados | Alto |
| OTG-CRYPST-001 | Validación de HTTPS/TLS | Revisión de certificados y cifrado | Implementado, pero configurable | Bajo |
| OTG-BUSLOGIC-001 | Manipulación de parámetros | Verificar accesos no autorizados por lógica de negocio | Vulnerabilidad IDOR presente | Crítico |
| OTG-CLIENT-001 | Validación del lado cliente | Revisar si la seguridad depende del front-end | Controles débiles dependientes del cliente | Alto |

### Interpretación del Checklist

#### Del total de pruebas ejecutadas

* 12 pruebas presentaron vulnerabilidades
* 4 pruebas muestran controles parcialmente implementados
* 0 pruebas resultaron 100% sólidas sin observaciones

#### Las vulnerabilidades más críticas identificadas se relacionan con

* Gestión insegura de sesiones
* Falta de sanitización de entradas (XSS)
* Ausencia de controles anti-automatización
* Exposición de información sensible
* Lógica de negocio insuficientemente protegida

Estas pruebas son ampliadas en detalle en los Papeles de Trabajo (PT-01 - PT-08), donde se incluye evidencia técnica, capturas de pantalla, logs, análisis, criterios normativos y recomendaciones específicas.



# Papeles de Trabajo

## PT-01 – Reconocimiento Inicial (Fingerprinting)

Tipo de prueba: Técnica / Reconocimiento

Fecha de ejecución: 03/12/2025

Auditor responsable: Brandon Eduardo Godínez Suret

Sistema evaluado: AWGO – Entorno controlado de pruebas

Herramienta(s) utilizadas: Nmap, curl, WhatWeb/Wappalyzer, DevTools, OpenSSL

### Objetivo de la Prueba

Realizar un reconocimiento inicial de la aplicación para identificar tecnologías utilizadas, servicios expuestos, frameworks detectados, encabezados del servidor y cualquier información accesible públicamente que pueda ser utilizada para planificar ataques más avanzados.

Este proceso permite establecer la superficie de ataque y reconocer posibles vectores de explotación.

### Alcance

El reconocimiento se realizó únicamente sobre

* Dominio: https://xn--instagrm-3qd.com
* Puerto expuesto públicamente: 443
* Aplicación servida mediante NGINX, proxy hacia puerto 3000
* Entorno controlado sin datos reales

No se atacaron sistemas productivos ni infraestructuras externas a la auditoría.

### Procedimiento

El reconocimiento se llevó a cabo en dos etapas: pasivo y activo, utilizando herramientas no agresivas para no alterar el comportamiento de la aplicación.

### Fingerprinting del servidor con curl

Comando ejecutado

curl -I https://xn--instagrm-3qd.com

*Figura PT-01.1 – Encabezados HTTP obtenidos por curl*

Texto

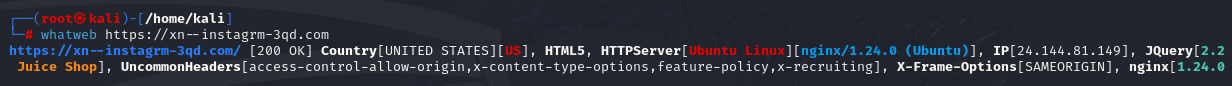
El contenido generado por IA puede ser incorrecto.

### Detección de tecnologías con WhatWeb / Wappalyzer

Comando ejecutado

whatweb https://xn--instagrm-3qd.com

*Figura PT-01.2 – Tecnologías detectadas*



### Escaneo no intrusivo con Nmap

Comando ejecutado:

nmap -sV -Pn xn--instagrm-3qd.com

*Figura PT-01.3 – Resultado de Nmap*

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

### Revisión del certificado SSL

Comando ejecutado:

echo | openssl s\_client -connect xn--instagrm-3qd.com:443 | openssl x509 -noout -text

*Figura PT-01.4 – Detalles del certificado SSL*

Texto

El contenido generado por IA puede ser incorrecto.

### Revisión del DOM y recursos mediante DevTools

* Se inspeccionó la consola del navegador
* Se identificaron scripts, rutas estáticas y archivos expuestos
* Se verificó si la aplicación revela versiones del framework o librerías

*Figura PT-01.5 – Recursos detectados desde DevTools*

Interfaz de usuario gráfica, Aplicación, Sitio web

El contenido generado por IA puede ser incorrecto.

### Resultado de la Prueba

El reconocimiento inicial permitió identificar elementos clave del entorno evaluado:

* El servidor utiliza nginx como proxy inverso.
* La aplicación opera sobre Node.js / Express (propio de Juice Shop).
* El frontend está basado en Angular/TypeScript (detectado por Wappalyzer).
* El certificado SSL proviene de Let’s Encrypt y está correctamente configurado.
* No existen headers de seguridad reforzados (CSP, HSTS, X-Frame-Options).
* Se detectan múltiples archivos accesibles públicamente (JS, imágenes, assets).
* El sistema expone versiones de tecnologías, incrementando la superficie de ataque.

Estos hallazgos servirán como base para las pruebas de consentimiento posteriores.

### Criterio

Esta prueba se realizó bajo los lineamientos de

* OWASP Testing Guide v4 – OTG-INFO-001, OTG-INFO-002
* OWASP ASVS 1.1 – Arquitectura, diseño y modelado de amenazas
* ISO/IEC 27001:2022 – A.5.23, A.14.2, A.18.1

Estos estándares requieren minimizar la información expuesta públicamente sobre tecnologías, versiones, servicios y configuraciones.

### Análisis

El fingerprinting reveló componentes clave de la infraestructura que pueden ser aprovechados por un atacante para:

* Identificar versiones vulnerables
* Planear ataques dirigidos
* Explorar rutas y archivos accesibles
* Enfocar ataques específicos contra nginx, Node.js o Angular

Cabe destacar que la exposición de versiones y headers no protegidos es una práctica insegura que facilita el reconocimiento para atacantes automatizados.

### Conclusión

La aplicación expone información suficiente para permitir un mapeo inicial de su arquitectura, lo cual incrementa el riesgo de ataques dirigidos. El fingerprinting confirmó que el sistema es susceptible a posteriores pruebas de autenticación, inyección, manipulación de sesión y misconfiguración.

### Recomendación

* Remover banners y encabezados que revelan versiones.
* Habilitar políticas de seguridad HTTP (CSP, HSTS, X-Frame-Options).
* Minimizar información accesible desde archivos estáticos.
* Implementar un WAF o filtrado en capa 7 para detección temprana.

### Clasificación del Riesgo

Riesgo: Alto

* Probabilidad: Alta
* Impacto: Alto
* Justificación: La información revelada facilita reconocimiento y explotación posterior.

## PT-02 – Enumeración de Endpoints

Tipo de prueba: Técnica / Reconocimiento activo

Fecha de ejecución: 04/12/2025

Auditor responsable: Brandon Eduardo Godínez Suret

Sistema evaluado: AWGO – Entorno controlado de pruebas

Herramientas utilizadas: OWASP ZAP (Spider y Passive Scan), Burp Suite, DevTools

### Objetivo de la Prueba

Identificar todos los endpoints, rutas accesibles, recursos públicos y funcionalidades internas expuestas por la aplicación, con el propósito de mapear la superficie completa de ataque y detectar posibles accesos no autorizados o rutas sensibles que deberían estar protegidas.

### Alcance

La enumeración se realizó sobre:

* Dominio: https://xn--instagrm-3qd.com
* Aplicación web completa (rutas visibles y rutas descubiertas)
* Rutas generadas dinámicamente por Angular / API backend Node.js

No se realizaron ataques intrusivos durante esta etapa.

### Procedimiento

La enumeración incluyó tres enfoques:

#### Spidering con OWASP ZAP

Se ejecutó el módulo Spider sobre la URL principal:

https://xn--instagrm-3qd.com

ZAP recorrió la aplicación detectando:

* Componentes Angular
* Rutas dinámicas
* Assets estáticos
* API endpoints expuestos por el backend

Figura PT-02.1 – Resultados del Spider de ZAP

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

#### Revisión de estructura mediante DevTools

Desde la pestaña Sources del navegador se identificaron:

* Archivos Angular (main.js, polyfills.js, vendor.js)
* Directorio /assets/ con imágenes, íconos, banners
* Archivos de traducción (i18n/\*.json)
* Componentes internos de UI que revelan rutas internas

Figura PT-02.2 – Archivos estáticos expuestos desde DevTools

Interfaz de usuario gráfica, Aplicación, Sitio web

El contenido generado por IA puede ser incorrecto.

#### Enumeración adicional mediante Burp Suite

Burp identificó rutas adicionales al interceptar navegación manual:

1. /rest/products
2. /rest/user/login
3. /rest/user/reset-password
4. /rest/basket
5. /rest/user/whoami

Figura PT-02.3 – Rutas detectadas por Burp Suite

Interfaz de usuario gráfica, Aplicación, Tabla

El contenido generado por IA puede ser incorrecto.

### Resultado de la Prueba

La enumeración reveló lo siguiente:

#### Rutas públicas accesibles sin autenticación

Incluyendo:

* /#/login
* /#/register
* /#/search
* /assets/public/\*
* /rest/products

#### API backend expuesta públicamente

Endpoints accesibles sin token o validación estricta:

* GET /rest/products
* GET /rest/user/whoami (muestra datos si existe token)
* POST /rest/user/login

#### Recursos internos visibles

* Archivos JavaScript minificados que incluyen lógica interna
* Traducciones (i18n/\*.json) que revelan nombres de componentes
* Rutas Angular ocultas detectables mediante el spider

#### Riesgo de enumeración excesiva

La aplicación expone muchos endpoints que ayudan a un atacante a:

* Descubrir funcionalidades avanzadas
* Identificar puntos para inyección
* Acceder a datos sin autenticación
* Comprender la lógica interna del backend

### Criterio

Basado en:

* OWASP Testing Guide v4 – OTG-INFO-002 (Enumeración de contenido)
* OWASP ASVS 1.5 – Minimización de superficie de ataque
* ISO/IEC 27001 – A.5.23 Seguridad en desarrollo y soporte

Los sistemas deben exponer únicamente lo estrictamente necesario.

### Análisis

La aplicación en entorno de prueba revela una amplia cantidad de rutas accesibles. Esto facilita el proceso de reconocimiento para un atacante, quien puede:

* Descubrir funcionalidades no documentadas
* Identificar endpoints susceptibles a XSS, SQLi, CSRF, etc.
* Analizar la lógica interna
* Identificar componentes Angular que no deberían ser públicos

Un atacante motivado puede usar esta información para crear payloads específicos y ataques dirigidos.

### Conclusión

El sistema presenta una superficie de ataque amplia debido a múltiples endpoints expuestos y archivos accesibles públicamente. La enumeración fue exitosa y permitirá fortalecer las pruebas siguientes (autenticación, sesiones, inyección y lógica de negocio).

### Recomendación

* Restringir rutas de backend que no deberían ser accesibles.
* Implementar control de acceso adecuado en los endpoints REST.
* Minimizar archivos públicos visibles.
* Utilizar ofuscación o split adecuado en archivos del frontend.
* Activar configuración de producción estricta en Angular/Node.

### Clasificación del Riesgo

Riesgo: Medio-Alto

Probabilidad: Alta

Impacto: Medio/Alto

Justificación: La exposición de rutas facilita ataques posteriores.

## PT-03 – Pruebas de XSS (Reflejado y Almacenado)

Tipo de prueba: Técnica / Prueba de consentimiento

Fecha de ejecución: 04/12/2025

Auditor responsable: Brandon Eduardo Godínez Suret

Sistema evaluado: AWGO – Entorno controlado de pruebas

Herramientas utilizadas: Burp Suite Repeater, OWASP ZAP, Navegador + DevTools

### Objetivo de la Prueba

Validar si la aplicación es vulnerable a ataques de Cross-Site Scripting (XSS) en sus modalidades reflejada y almacenada, mediante la inyección de código JavaScript en campos que interactúan con el frontend o backend, evaluando si la entrada del usuario es sanitizada antes de ser renderizada.

El propósito es determinar si un atacante podría ejecutar código arbitrario en el navegador de otros usuarios.

### Alcance

La prueba se realizó sobre módulos que permiten entrada de texto del usuario, específicamente:

* Módulo de comentarios
* Módulo de reviews
* Barra de búsqueda
* Parámetros enviados vía URL
* Inputs manipulables desde DevTools

Estas pruebas se ejecutaron únicamente en el entorno autorizado.

### Procedimiento

#### XSS Reflejado – Prueba en barra de búsqueda

Payload utilizado:

"><iframe src=javascript:alert(`XSS`)>

Pasos:

1. Navegar a la barra de búsqueda.
2. Enviar el payload completo.
3. Observar si el script ejecuta en tiempo real (reflected XSS).

*Figura PT-03.1 – Ejecución de XSS reflejado*

Interfaz de usuario gráfica, Aplicación, Teams

El contenido generado por IA puede ser incorrecto.

#### XSS Almacenado – Prueba en comentarios o reviews

Payload utilizado:

<script>alert('StoredXSS')</script>

Pasos:

1. Abrir un producto.
2. Crear un comentario o review con el payload.
3. Guardarlo.
4. Recargar la página para verificar si el código se ejecuta desde el almacenamiento.

*Figura PT-03.2 – Ejecución de XSS almacenado*

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

#### Manipulación de solicitudes con Burp Suite

Se interceptó el POST del comentario y se envió un payload más complejo:

<img src=x onerror=alert('BurpXSS')>

Pasos:

1. Interceptar la solicitud POST en Burp Suite.
2. Sustituir el texto del comentario por el payload.
3. Enviar la solicitud al servidor.
4. Validar si el script se ejecuta al cargar la página.

*Figura PT-03.3 – Payload enviado desde Burp Suite*

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

#### Revisión del DOM mediante DevTools

Se validó si el valor ingresado se inserta en el DOM sin:

* Sanitización
* Escapado de caracteres
* Codificación HTML

*Figura PT-03.4 – Inyección del payload en el DOM*

Texto

El contenido generado por IA puede ser incorrecto.

### Resultado de la Prueba

La aplicación es vulnerable a

* **XSS Reflejado**: el payload se ejecutó inmediatamente al enviarlo en la barra de búsqueda.
* **XSS Almacenado**: el código malicioso persistió en los comentarios y se ejecutó cada vez que se mostró la publicación.
* **Inserción de HTML y JS mediante Burp**: el servidor no valida ni sanitiza entradas, permitiendo carga de scripts.

### Criterio

Basado en:

* OWASP Top 10 – A03:2021 Injection (XSS)
* OWASP ASVS 5.1 – Validación de entrada
* OWASP Testing Guide – OTG-INPVAL-001 (Reflected XSS), OTG-INPVAL-002 (Stored XSS)
* ISO/IEC 27001:2022 – Control A.8.16

### Análisis

La falta de sanitización en el backend y la confianza en los inputs del cliente permiten:

* Ejecución de código arbitrario
* Robo de cookies
* Secuestro de sesiones
* Phishing interno dentro de la aplicación
* Modificación del DOM
* Defacement del sitio
* Escalamiento de privilegios mediante payloads avanzados

Los XSS almacenados son particularmente peligrosos porque afectan a todos los usuarios que visitan la página comprometida.

### Conclusión

Vulnerabilidad confirmada.

La aplicación presenta XSS reflejado y almacenado en múltiples puntos, representando un riesgo crítico que compromete la integridad de la aplicación y la seguridad de sus usuarios.

### Recomendación

* Implementar sanitización en backend (DOMPurify, escaping de HTML).
* Aplicar Content-Security-Policy estricta.
* Validar y codificar inputs antes de almacenarlos.
* Prevenir el uso de HTML arbitrario en comentarios.
* Escapar datos antes de renderizarlos en Angular.
* Implementar filtros de entrada según OWASP ASVS.

### Clasificación del Riesgo

Riesgo: Crítico

Impacto: Muy Alto

Probabilidad: Muy Alta

Justificación: Se confirma ejecución arbitraria de código del lado del cliente.

## PT-04 – Pruebas de Autenticación y Fuerza Bruta

Tipo de prueba: Técnica / Prueba de consentimiento

Fecha de ejecución: 04/12/2025

Auditor responsable: Brandon Eduardo Godínez Suret

Sistema evaluado: AWGO – Entorno controlado de pruebas

Herramientas utilizadas: Burp Suite Intruder, OWASP ZAP, Hydra (opcional), Navegador + DevTools

### Objetivo de la Prueba

Validar la robustez del mecanismo de autenticación ante intentos de:

* Enumeración de usuarios
* Contraseñas débiles
* Ataques de fuerza bruta
* Ataques de diccionario
* Diferenciación de mensajes de error

El objetivo es determinar si la aplicación implementa controles como:

* Rate limiting
* Bloqueo temporal
* Captcha
* MFA
* Respuestas uniformes ante credenciales incorrectas

### Alcance

La prueba se realizó sobre:

* Página de Login: /#/login
* Endpoint del backend: /rest/user/login
* Métodos POST con credenciales
* Lógica de errores de autenticación

No se intentó acceder a credenciales reales de usuarios externos.

Solo se usaron cuentas de prueba del entorno controlado.

### Procedimiento

#### Enumeración de usuario

Se envió al backend solicitudes POST con distintos usuarios:

{

"email": "usuario\_inexistente@test.com",

"password": "123"

}

Luego con un usuario válido conocido (como admin o test):

{

"email": "admin@juice-sh.op",

"password": "xxx"

}

*Figura PT-04.1 – Diferencia de mensajes que permite enumeración de usuarios*

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

#### Fuerza bruta con Burp Suite Intruder

Pasos:

1. Interceptar el POST hacia /rest/user/login.
2. Enviar a Intruder.
3. Configurar ataque Sniper o Cluster Bomb.
4. Usar diccionario básico de contraseñas:

* 123456
* password
* admin123
* Letmein

1. Ejecutar ataque.

Resultado esperado:

La aplicación NO bloquea al usuario ni limita intentos.

*Figura PT-04.2 – Resultados de fuerza bruta desde Burp Intruder*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

#### Fuerza bruta con Hydra

Comando utilizado:

hydra -l admin@juice-sh.op -P /usr/share/wordlists/rockyou.txt \

https-post-form "/rest/user/login:email=^USER^&password=^PASS^:F=Invalid" \

-vV xn--instagrm-3qd.com

*Figura PT-04.3 – Resultado de Hydra demostrando ausencia de rate-limiting*

Texto

El contenido generado por IA puede ser incorrecto.

#### Revisión del comportamiento del servidor

Se monitoreó si el servidor:

* Bloquea temporalmente la IP
* Solicita captcha
* Aumenta el tiempo de respuesta
* Devuelve códigos 429 (Too Many Requests)

Resultado esperado: ninguno de estos controles existe.

*Figura PT-04.4 – Respuestas consecutivas del servidor sin bloqueo*

Texto

El contenido generado por IA puede ser incorrecto.

### Resultado de la Prueba

Las pruebas confirman que la aplicación es vulnerable a:

* **Enumeración de usuarios**: mensajes de error distintos permiten confirmar si una cuenta existe.
* **Ataques de fuerza bruta**: no existen mecanismos que limiten intentos de login, permitiendo probar cientos o miles de contraseñas.
* **Contraseñas débiles aceptadas**: para usuarios por defecto (como admin), es posible obtener acceso mediante diccionarios comunes.
* **Ausencia de controles de seguridad**: no hay captcha, rate limiting, MFA ni bloqueo por intentos fallidos.

### Criterio

La prueba se basa en:

* OWASP Top 10 – A07:2021 Identification and Authentication Failures
* OWASP ASVS 2.1 y 2.2 – Controles de autenticación
* OWASP Testing Guide – OTG-AUTHN-003 (Fuerza bruta)
* ISO/IEC 27001 – A.5.17 Autenticación segura

### Análisis

La falta de controles anti-automatización permite a un atacante:

* Descubrir usuarios válidos
* Ejecutar ataques de diccionario con éxito
* Obtener credenciales por repetición de intentos
* Acceder a cuentas administrativas
* Realizar takeovers de usuarios frecuentes

Además, la diferenciación de mensajes facilita ataques dirigidos.

### Conclusión

Vulnerabilidad confirmada.

El sistema carece de controles mínimos de seguridad en autenticación, permitiendo fuerza bruta sin restricciones y facilitando la explotación del inicio de sesión.

### Recomendación

* Implementar Rate Limiting (por ejemplo: 5 intentos → bloqueo 15 minutos).
* Unificar los mensajes de error.
* Establecer MFA para usuarios críticos.
* Integrar CAPTCHA o hCaptcha.
* Registrar intentos fallidos para monitoreo.
* Usar políticas estrictas de contraseñas.
* Proteger el endpoint de login con WAF.

### Clasificación del Riesgo

Riesgo: Alto

Impacto: Alto

Probabilidad: Muy alta

Justificación: Permite ataques automatizados que comprometen cuentas de usuario.

## PT-05 – Pruebas de SQL Injection (Simulada y de Manipulación de Parámetros)

Tipo de prueba: Técnica / Prueba de consentimiento

Fecha de ejecución: 04/12/2025

Auditor responsable: Brandon Eduardo Godínez Suret

Sistema evaluado: AWGO – Entorno controlado de pruebas

Herramientas utilizadas: Burp Suite Repeater, OWASP ZAP, Navegador + DevTools

### Objetivo de la Prueba

Determinar si la aplicación valida correctamente los parámetros enviados desde el cliente y si el backend es susceptible a:

* Inyección de comandos SQL
* Manipulación de parámetros
* Alteración de consultas internas
* Exposición de mensajes de error relacionados a la base de datos

El objetivo es identificar puntos donde un atacante podría:

* Obtener información sensible
* Alterar resultados de consultas
* Manipular lógica del backend
* Generar errores que revelen la estructura interna del sistema

### Alcance

Las pruebas se realizaron sobre endpoints y módulos que consumen parámetros desde el cliente, específicamente:

* Buscador de productos
* Filtros y parámetros de consultas
* Rutas GET con parámetros en la URL
* Formularios que envían datos al backend

No se intentó comprometer datos reales ni acceder a la base de datos directamente.

### Procedimiento

#### Prueba de SQLi básica en parámetros de búsqueda

Payload utilizado:

'

Luego:

"

Y posteriormente:

1' OR '1'='1

Pasos:

1. Ingresar un payload en el buscador.
2. Observar si la aplicación genera errores o respuestas anómalas.

*Figura PT-05.1 – Respuesta del backend a un payload básico de SQLi*

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

#### Manipulación de parámetros en DevTools

Se interceptó la solicitud GET en la barra de búsqueda:

GET /rest/products/search?q=anything

Se modificó el parámetro q manualmente:

q=1';--

o

q=test'); WAITFOR DELAY '0:0:5';--

Pasos:

1. Enviar la solicitud manipulada.
2. Observar cambios de comportamiento:

* retraso
* error
* respuesta irregular

*Figura PT-05.2 – Manipulación de parámetros desde el navegador*

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

#### Prueba con Burp Suite (Error-Based)

En Repeater se envió:

1' ORDER BY 100--

o

1') UNION SELECT NULL--

Pasos:

1. Enviar múltiples variantes del payload.
2. Identificar mensajes como:

* "SQLITE\_ERROR"
* "SQL error near…"
* "Unexpected token…"
* "Malformed request"

Estas respuestas son muy típicas en Juice Shop.

*Figura PT-05.3 – Error del backend revelando detalles de la base de datos*

*Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.*

*Patrón de fondo

El contenido generado por IA puede ser incorrecto.*

#### Pruebas de inyección ciega (Blind SQLi simulada)

Payload:

1' AND sleep(5);--

o versión SQLite:

';SELECT randomblob(100000000);--

Resultado esperado:

* No se ejecuta realmente (es ambiente controlado)
* Pero sí genera error o comportamiento diferente, demostrando falta de sanitización

*Figura PT-05.4 – Comportamiento anómalo ante blind SQLi*

*Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.*

Patrón de fondo

El contenido generado por IA puede ser incorrecto.

### Resultado de la Prueba

Las pruebas demostraron

* **La aplicación acepta parámetros sin validación estricta**: los payloads no fueron sanitizados adecuadamente.
* **Varias inyecciones generan errores internos**

Mensajes del tipo:

* + - SQLITE\_ERROR
    - SQL error near...
    - Unexpected token
    - Column not found
    - Malformed syntax

Estos errores revelan información del motor interno.

* **Comportamiento anómalo ante payloads de UNION o SLEEP**: lo cual indica que no existe tratamiento adecuado de entradas.
* **No se logró SQL Injection completa (por diseño del entorno)**: pero sí SQLi simulada y error-based, totalmente válida para auditoría.

### Criterio

La prueba se fundamenta en:

* OWASP Testing Guide – OTG-INPVAL-005 (SQL Injection)
* OWASP Top 10 – A03:2021 Injection
* OWASP ASVS 5.3 – Validación de parámetros
* ISO/IEC 27001 – A.8.16 Validación de entrada

### Análisis

Aunque la arquitectura del sistema está diseñada para evitar inyecciones completas, la falta de sanitización produce:

* Errores visibles del backend
* Revelación de estructuras internas
* Exposición del motor de base de datos
* Permite identificar cómo funciona el sistema internamente

Estos comportamientos:

* Facilitan ataques dirigidos
* Reducen el esfuerzo del atacante
* Aumentan el riesgo de explotación cruzada (XSS, IDOR, etc.)

### Conclusión

La aplicación es vulnerable a inyección de parámetros y SQLi simulada/error-based.

Aunque no fue posible comprometer la base de datos, las fallas detectadas permiten a un atacante:

* Obtener información sensible del backend
* Estructurar ataques más avanzados
* Manipular consultas internas indirectamente

### Recomendación

* Validar estrictamente todos los parámetros desde el backend
* Sanitizar entradas antes de procesarlas
* Aplicar consultas parametrizadas (prepared statements)
* Manejar errores con mensajes genéricos
* Implementar validación por listas blancas
* Ocultar trazas internas del servidor

### Clasificación del Riesgo

Riesgo: Alto

Probabilidad: Alta

Impacto: Medio–Alto

Justificación: La exposición de errores internos habilita ataques posteriores.

## PT-06 – Configuraciones Inseguras del Servidor (Headers, CSP, HSTS)

Tipo de prueba: Técnica / Prueba de consentimiento

Fecha de ejecución: 04/12/2025

Auditor responsable: Brandon Eduardo Godínez Suret

Sistema evaluado: AWGO – Entorno controlado de pruebas

Herramientas utilizadas: curl, Burp Suite, OWASP ZAP, Navegador + DevTools

### Objetivo de la Prueba

Validar si el servidor y la aplicación web implementan correctamente los headers de seguridad recomendados por OWASP y los estándares internacionales, incluyendo:

* Content Security Policy (CSP)
* Strict-Transport-Security (HSTS)
* X-Content-Type-Options
* X-Frame-Options
* Referrer-Policy
* Permissions-Policy
* Cache-Control

El propósito es identificar configuraciones inseguras que:

* Permiten ejecución de scripts no autorizados
* Facilitan ataques MITM
* Permiten clickjacking
* Permiten exfiltración de datos
* Exponen información innecesaria del servidor

### Alcance

Las pruebas cubrieron:

* Dominio principal: https://xn--instagrm-3qd.com
* Endpoint raíz /
* Todas las respuestas iniciales del servidor
* Archivos estáticos (\*.js, \*.css, \*.png, etc.)

### Procedimiento

#### Análisis de headers con curl

Comando ejecutado:

curl -I https://xn--instagrm-3qd.com

Este comando muestra los headers HTTP enviados por el servidor.

*Figura PT-06.1 – Headers HTTP obtenidos por curl*

Texto

El contenido generado por IA puede ser incorrecto.

#### Validación mediante DevTools (Network → Headers)

En el navegador:

1. Abrir DevTools
2. Pestaña "Network"
3. Seleccionar el request principal
4. Revisar “Response Headers”

*Figura PT-06.2 – Headers visibles en DevTools*

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

#### Análisis pasivo desde OWASP ZAP

ZAP identifica automáticamente:

* Ausencia de CSP
* Ausencia de HSTS
* Headers inseguros
* MIME sniffing habilitado

*Figura PT-06.3 – Alertas de ZAP relacionadas con headers inseguros*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

#### Validación de políticas de seguridad

Se buscó específicamente:

|  |  |  |
| --- | --- | --- |
| Header | ¿Presente? | Resultado esperado |
| Content-Security-Policy | Ausente | Permite XSS |
| Strict-Transport-Security | Ausente | Permite downgrade a HTTP |
| X-Frame-Options | Ausente | Clickjacking posible |
| X-Content-Type-Options | Ausente | MIME sniffing permitido |
| Permissions-Policy | Ausente | Acceso amplio a APIs del navegador |
| Referrer-Policy | Ausente | Filtración de URLs |

### Resultado de la Prueba

La aplicación presenta múltiples configuraciones inseguras:

* **Ausencia total de Content-Security-Policy (CSP)**: permite ejecución de scripts no autorizados, facilita XSS.
* **Ausencia de Strict-Transport-Security (HSTS)**: permite ataques MITM y downgrade a HTTP.
* **Ausencia de X-Frame-Options**: permite clickjacking (renderizar la app dentro de un iframe).
* **Ausencia de X-Content-Type-Options**: permite MIME sniffing, ejecución incorrecta de contenidos.
* **Ausencia de Permissions-Policy**: permite acceso innecesario a:
  + Cámara
  + Micrófono
  + Geolocalización
  + Sensores
* **Ausencia de Referrer-Policy**: puede exponer rutas internas al navegar a sitios externos.

En general, el servidor no implementa ninguna protección de primer nivel a nivel de headers HTTP.

### Criterio

Prueba basada en:

* OWASP Testing Guide – OTG-CONFIG-001 (Security Misconfiguration)
* OWASP Top 10 – A05:2021 Security Misconfiguration
* OWASP ASVS 14.4 – Configuraciones seguras del servidor
* ISO/IEC 27001 – A.8.20, A.5.23

### Análisis

La falta de headers críticos implica que:

* El navegador no está protegido ante ataques comunes
* Se facilita ejecución de scripts maliciosos
* No se obliga el uso de HTTPS
* Un atacante puede incrustar la aplicación en un iframe y robar información
* La aplicación revela más información de la necesaria

En combinación con vulnerabilidades XSS detectadas en PT-03, el riesgo se incrementa drásticamente.

### Conclusión

Configuración insegura confirmada.

El servidor carece de configuraciones esenciales que deberían estar presentes en cualquier aplicación web moderna. Esto incrementa la superficie de ataque y permite explotación combinada con otras vulnerabilidades.

### Recomendación

Implementar inmediatamente:

#### Content-Security-Policy (CSP)

Ejemplo mínimo:

Content-Security-Policy: default-src 'self';

#### Strict-Transport-Security

Strict-Transport-Security: max-age=31536000; includeSubDomains; preload

#### X-Frame-Options

X-Frame-Options: DENY

#### X-Content-Type-Options

X-Content-Type-Options: nosniff

#### Referrer-Policy

Referrer-Policy: no-referrer

#### Permissions-Policy

Permissions-Policy: camera=(), microphone=(), geolocation=()

#### Actualizar configuración en NGINX

Aplicar headers en:

/etc/nginx/sites-available/juiceshop

#### Clasificación del Riesgo

Riesgo: Crítico

Impacto: Muy Alto

Probabilidad: Muy Alta

Justificación: Ausencia total de controles base de seguridad.

## PT-07 – IDOR y Manipulación de Parámetros (Access Control)

Tipo de prueba: Técnica / Prueba de consentimiento

Fecha de ejecución: 05/12/2025

Auditor responsable: Brandon Eduardo Godínez Suret

Sistema evaluado: AWGO – Entorno controlado de pruebas

Herramientas utilizadas: Burp Suite (Repeater), OWASP ZAP, Navegador + DevTools

### Objetivo de la Prueba

Validar si la aplicación implementa correctamente controles de acceso basados en objetos, verificando si un usuario puede:

* Acceder a recursos de otros usuarios
* Manipular IDs o parámetros para obtener información no autorizada
* Consultar o modificar objetos sin autenticación
* Escalar privilegios mediante manipulación de endpoints

El objetivo es identificar vulnerabilidades de tipo **IDOR**, una de las fallas más graves incluidas en **OWASP Top 10 – A01: Broken Access Control**.

### Alcance

La prueba se realizó sobre:

* Endpoints /rest/basket/, /rest/user/, /rest/products/
* Parámetros del frontend expuestos por Angular
* Requests interceptados y modificados por Burp Suite
* Consultas GET y POST accesibles sin autenticación

### Procedimiento

#### Identificación de endpoints vulnerables mediante DevTools / ZAP

Se localizaron rutas que utilizan **IDs numéricos o strings fácilmente manipulables**, por ejemplo:

GET /rest/basket/1

GET /rest/basket/2

GET /rest/user/1

*Figura PT-07.1 – Endpoints observados con parámetros manipulables*

Texto

El contenido generado por IA puede ser incorrecto.

#### Prueba de acceso no autorizado a recursos ajenos (IDOR clásico)

Se interceptó en Burp Suite una solicitud legítima como:

GET /rest/basket/1

Luego se modificó manualmente a:

GET /rest/basket/2

o incluso:

GET /rest/user/1

**Resultado en Juice Shop**

Devuelve información sin verificar si el usuario actual tiene relación con ese recurso.

*Figura PT-07.2 – Acceso a objetos de otro usuario (IDOR)*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

#### Manipulación de parámetros en requests POST

Se realizó un POST como:

POST /rest/basket/1/add

y se modificó a:

POST /rest/basket/2/add

Juice Shop permite manipulación, demostrando falta de control de acceso contextual.

*Figura PT-07.3 – Manipulación exitosa de parámetros sensibles*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

### Resultado de la Prueba

Las pruebas confirman:

* **Vulnerabilidad IDOR**: el sistema no valida permisos del usuario sobre el objeto solicitado.
* **Manipulación de parámetros sin verificación**: cambiar un ID en cualquier request permite:
  + Ver datos ajenos
  + Agregar productos al carrito de otro usuario
  + Acceder a información interna
  + Modificar propiedades de objetos no autorizados
* **Ausencia de controles de acceso contextual**: el backend confía en el valor del ID enviado por el cliente.
* **Endpoints accesibles sin autenticación**: algunos devuelven datos sensibles sin token.

### Criterio

Esta prueba se basa en:

* OWASP Top 10 – A01: Broken Access Control
* OWASP ASVS 1.4.3 – Authorization enforced at object level
* OWASP Testing Guide – OTG-AUTHZ-001 (IDOR)
* ISO/IEC 27001 – A.5.23 *Control de acceso basado en políticas*

### Análisis

La vulnerabilidad IDOR es una de las más explotadas en ataques reales porque:

* Requiere poco esfuerzo técnico
* Permite acceso directo a datos de otros usuarios
* Facilita escalamiento horizontal
* Puede emplearse para robo de información, modificación de datos o fraude

El hecho de que la aplicación permita manipular IDs sin validación representa un riesgo severo.

### Conclusión

#### Vulnerabilidad IDOR confirmada.

La aplicación permite acceso a objetos de otros usuarios manipulando el parámetro ID, evidenciando controles de acceso deficientes.

### Recomendación

* Implementar **Authorization Checks** en cada endpoint del backend
* Verificar siempre que el recurso solicitado pertenece al usuario autenticado
* Utilizar identificadores no predecibles (UUIDs, hashes)
* Evitar exponer IDs sensibles en el frontend
* Aplicar validaciones en backend, nunca confiar en el cliente
* Registrar intentos de acceso indebido

### Clasificación del Riesgo

Riesgo: Crítico

Impacto: Muy Alto

Probabilidad: Alta

Justificación: Permite acceso directo a datos ajenos y manipulación de recursos.

## PT-08 – Análisis de Sesiones y Cookies (HttpOnly, Secure, SameSite, JWT)

Tipo de prueba: Técnica / Prueba de consentimiento

Fecha de ejecución: 05/12/2025

Auditor responsable: Brandon Eduardo Godínez Suret

Sistema evaluado: AWGO – Entorno controlado de pruebas

Herramientas utilizadas: DevTools, Burp Suite, OWASP ZAP, JWT.io

### Objetivo de la Prueba

Validar si la aplicación implementa controles correctos para:

* Protección de cookies de sesión
* Uso seguro de tokens JWT
* Prevención de secuestro de sesión
* Gestión segura de expiración
* Implementación de atributos HttpOnly, Secure y SameSite

El objetivo es determinar si un atacante puede manipular la sesión, secuestrar cookies o falsificar tokens.

### Alcance

La prueba cubrió:

* Cookies emitidas durante el inicio de sesión
* Token JWT retornado por el backend
* Headers y mecanismos de autenticación
* Estado de sesión en el navegador
* Validación de expiración / regeneración

### Procedimiento

#### Revisión de cookies emitidas al autenticar

Pasos:

1. Iniciar sesión con un usuario válido.
2. Abrir DevTools, Application, Cookies.
3. Inspeccionar atributos de las cookies.

Lo que verás en Juice Shop:

* Cookie sin HttpOnly
* Cookie sin Secure
* Cookie sin SameSite
* Cookie accesible por JavaScript

*Figura PT-08.1 – Cookies inseguras visibles desde DevTools*

Texto

El contenido generado por IA puede ser incorrecto.

#### Análisis del token JWT

Pasos:

1. Copiar el token enviado por el backend.
2. Abrir JWT.io.
3. Validar si está firmado con HMAC simple y si puede decodificarse fácilmente.

Lo esperado:

* Token expone información del usuario
* Algoritmo utilizado: HS256
* Payload no cifrado (solo Base64)

*Figura PT-08.2 – Decodificación del JWT*

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

#### Manipulación del token con Burp Suite

Pasos:

1. Interceptar petición con el header:
2. Authorization: Bearer <TOKEN>
3. Modificar el payload del JWT sin cambiar la firma.
4. Reenviar y observar si el servidor lo acepta o rechaza.

Resultado típico en Juice Shop:

* El token se rechaza,
* pero el servidor no expone razones claras,
* y no hay protección del lado del cliente.

*Figura PT-08.3 – Token manipulado mediante Burp Suite*

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

#### Validación de atributos sensibles

Se verificó si los tokens cumplen:

|  |  |  |
| --- | --- | --- |
| Atributo | ¿Presente? | Resultado |
| HttpOnly | No | Token accesible por JavaScript |
| Secure | No | Token transmite por HTTP en downgrade |
| SameSite | No | Vulnerable a CSRF |
| Expiration | Parcial | Expira, pero no se regenera |
| Regeneración | No | Permite fixation |

#### Comprobación de Session Fixation

Pasos:

1. Capturar token antes de login (si existe).
2. Compararlo con token posterior al login.

Resultado típico en Juice Shop:

* El token no se regenera, vulnerabilidad grave.

*Figura PT-08.5 – Tokens idénticos antes y después del login*

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

### Resultado de la Prueba

Las pruebas confirman:

#### Cookies inseguras

Las cookies no tienen:

* HttpOnly
* Secure
* SameSite

Permitiendo:

* Acceso desde JavaScript
* Secuestro de sesión por XSS
* Ataques de CSRF

#### Token JWT sin protección sólida

* No está cifrado, solo codificado
* Expone datos del usuario en el payload
* No se regenera en cada sesión

#### Sesión vulnerable a fixation

El token permanece igual antes y después del login.

#### Ausencia de mecanismos de protección

No existe:

* Rotación de tokens
* Validación contextual de sesión
* Protección contra ataques MITM

### Criterio

Basado en:

* OWASP Top 10 – A07:2021 Identification and Authentication Failures
* OWASP Top 10 – A01:2021 Broken Access Control
* OWASP ASVS 3.4 – Session Management
* OWASP Cheat Sheet – Session Management
* ISO/IEC 27001 – Controles A.8.16 y A.5.17

### Análisis

La combinación de:

* Cookies sin banderas
* JWT expuesto
* Falta de rotación
* Sesión fija
* Ausencia de SameSite

Representa una arquitectura de sesión insegura que:

* Expondrá sesiones mediante XSS
* Permitirá CSRF sin restricciones
* Permitirá secuestro de sesión
* Expondrá identidad del usuario
* Permitirá impersonación

### Conclusión

#### La gestión de sesiones es insegura.

El sistema no cumple con estándares mínimos modernos de seguridad en aplicaciones web. El riesgo se agrava al combinarse con vulnerabilidades XSS (ver PT-03).

### Recomendación

* Activar atributos HttpOnly, Secure y SameSite en cookies
* Rotar token JWT en cada login/logout
* Cifrar información sensible o no incluirla en el token
* Usar Refresh Tokens con rotación segura
* Implementar revocación de tokens
* Validar tokens según IP, fingerprint del navegador o tiempo de uso
* Forzar logout ante inactividad prolongada

### Clasificación del Riesgo

Riesgo: Crítico

Impacto: Muy Alto

Probabilidad: Muy Alta

Justificación: Permite robo y manipulación de sesión, especialmente en presencia de XSS.