Informe Técnico de Seguridad

Proyecto: Sistema Multiplataforma de Reservas
Grupo: 12
Integrantes: Boris Israel Rengel Japón, José Fermín Encalada Leiva
Fecha: 06/11/2025
Repositorio: https://github.com/FerminEncalada/Sistema-Reservas.git

1. Introducción

El presente documento tiene como finalidad analizar, describir y evidenciar las medidas de seguridad implementadas en el backend del Sistema Multiplataforma de Reservas, desarrollado bajo el entorno Node.js y el framework Express.
Dicho sistema gestiona usuarios, canchas y reservas, y constituye el componente central de la arquitectura multiplataforma, donde se prioriza la seguridad de la información y la protección de los datos personales de los usuarios.
La aplicación de medidas de seguridad está guiada por el estándar internacional OWASP Top 10 (2021), que recopila las vulnerabilidades más críticas en aplicaciones web.
Este informe evalúa cómo el sistema aborda cada uno de estos riesgos, detallando evidencias concretas, fragmentos de código, configuración del entorno y mecanismos de mitigación implementados.
Además, se analiza el uso de autenticación JWT (JSON Web Tokens), el cifrado de contraseñas con bcrypt, las validaciones de entrada y el manejo de errores bajo criterios de buenas prácticas de ingeniería de software segura

2. Alcance

El informe cubre las capas de seguridad del backend, específicamente:

- Control de autenticación y autorización.
- Validación de datos de entrada.
- Cifrado y protección de credenciales.
- Manejo de tokens y cookies seguras.
- Comunicación entre microservicios.
- Mitigación de vulnerabilidades OWASP.

No incluye la seguridad del frontend ni del entorno de despliegue, los cuales serán analizados en futuras fases del proyecto.

3. Tecnologías y dependencias

Lenguaje y entorno: Node.js 20.x
Framework: Express 5.1.0
Base de datos: MongoDB (Mongoose 8.19.1)
Principales librerías de seguridad:

- **bcryptjs** → Cifrado de contraseñas.
- **jsonwebtoken** → Generación de tokens JWT.
- **cookie-parser** → Manejo de cookies HTTP Only.

- **dotenv** → Gestión de variables de entorno.

4. Autenticación y Autorización

El sistema utiliza JWT (JSON Web Token) como mecanismo de autenticación.
Los tokens se generan al iniciar sesión y se envían al cliente dentro de una cookie con atributo httpOnly, evitando su acceso desde scripts del navegador.

Seguridad aplicada:

- El token expira en 24 horas (expiresIn: '1d').
- Se firma con un secreto almacenado en .env.
- Se guarda en una cookie httpOnly:

5. Cifrado y protección de credenciales

Durante el registro, las contraseñas se cifran antes de almacenarse en la base de datos usando bcrypt.

Buenas prácticas aplicadas:

- Cifrado irreversible con salt automático.
- Prohibición de contraseñas débiles mediante regex:
- Verificación de unicidad en correo, cédula y username.
- Exclusión del campo password al obtener usuarios:

6. Comunicación segura entre servicios

El sistema cuenta con un microservicio de notificaciones que utiliza nodemailer para enviar correos electrónicos de confirmación.

Buenas prácticas:

- Credenciales protegidas mediante variables de entorno .env.
- No se almacenan contraseñas de Gmail en el código fuente.
- Canales seguros con TLS.

Posibles correcciones:

- Reemplazar rejectUnauthorized: false por true en producción.
- Usar contraseñas de aplicación de Gmail (App Passwords).

7. Validación y Sanitización de datos

Se aplican múltiples validaciones de entrada en los controladores principales (auth.controller.js, cancha.controller.js, reserva.controller.js):

- Validación de campos vacíos.
- Validación de correo con formato Gmail.

- Validación de cédula ecuatoriana (10 dígitos).
- Validación de longitud y formato de nombre y acrónimo.
- Validación de horarios y conflictos de reservas.

Posible mejora:
Agregar middleware de sanitización para evitar inyecciones NoSQL con operadores como $ne, $gt.

8. Manejo de errores y respuesta controlada

Cada controlador incluye bloques try...catch con respuestas JSON estructuradas.
Sin embargo, algunos devuelven error.message, lo cual puede filtrar información sensible.

9. Principios OWASP Top 10 aplicados

Riesgo OWASP	Implementación	Estado
A01 – Broken Access Control	Middleware authRequired, isAdmin	Parcial
A02 - Cryptographic Failures	bcrypt + JWT + .env	Implementado
A03 - Injection	Validaciones y mongoose	Parcial
A04 - Insecure Design	Validación de reglas de negocio	Implementado
A05 - Security Misconfiguration	Cookies seguras, dotenv	Parcial
A06 - Vulnerable Components	Dependencias actualizadas en package.json	Implementado
A07 - Auth Failures	JWT + cifrado bcrypt	Implementado
A08 - Data Integrity Failures	Comunicación TLS (pendiente endurecer)	Parcial
A09 - Logging Failures	Captura básica en consola	Parcial
A10 - SSRF	Comunicación interna controlada	Implementado

10.Conclusiones

El backend del sistema de reservas demuestra una implementación sólida en seguridad básica, aplicando correctamente:

- Cifrado de contraseñas.
- Tokens JWT con expiración.
- Validaciones de entrada estrictas.
- Separación de credenciales mediante .env.

Sin embargo, aún existen aspectos por mejorar:

- Fortalecer la configuración de cookies y cabeceras HTTP.
- Implementar rate limiting y sanitización global.
- Estandarizar el manejo de errores en producción.

Con la aplicación de las mejoras recomendadas, el sistema alcanzará un nivel de seguridad alineado con las mejores prácticas OWASP.

11. Bibliografía

[1] OWASP Foundation. *OWASP Top 10: 2021 – The Ten Most Critical Web Application Security Risks.*

[2] Node.js Security Best Practices — https://nodejs.org/en/security

[3] OWASP Cheat Sheet Series — *Authentication & Cryptographic Storage*.