Informe Técnico de Seguridad del Proyecto: Sistema de Reservas

1. Resumen técnico del proyecto

El proyecto **“Sistema de Reservas”** es una aplicación web en desarrollo orientada a la gestión de usuarios y reservas mediante un sistema de autenticación basado en **JSON Web Tokens (JWT)**. Su arquitectura se compone de un backend desarrollado en **Node.js** con el framework **Express.js**, y una base de datos **MongoDB** manejada a través de **Mongoose** como ORM.

El entorno de desarrollo empleado es **Windows 11**, con ejecución local de la base de datos y uso de **npm** para la gestión de dependencias. La aplicación está diseñada para proporcionar servicios de autenticación, registro, inicio de sesión y acceso a recursos protegidos mediante rutas seguras.

Las principales dependencias utilizadas en el backend son:

- **express** → Creación del servidor y definición de rutas RESTful.
- **mongoose** → Conexión y modelado de la base de datos MongoDB.
- **jsonwebtoken** → Generación y validación de tokens de autenticación.
- **bcryptjs** → Cifrado de contraseñas para proteger credenciales de usuario.
- **cookie-parser** y **morgan** → Manejo de cookies y registro de peticiones HTTP.

Actualmente, el proyecto se encuentra en una fase inicial de desarrollo, con funcionalidades básicas de autenticación implementadas y un enfoque principal en el backend. El frontend se desarrollará posteriormente utilizando **React.js**, permitiendo así la separación de responsabilidades y favoreciendo una arquitectura modular.

2. Identificación de vulnerabilidades OWASP Top 10 (2021)

Tras el análisis del código fuente del backend, se identificaron cinco vulnerabilidades relevantes basadas en el estándar OWASP Top 10 (versión 2021) que podrían afectar la seguridad del sistema:

A01: Broken Access Control

El proyecto implementa autenticación básica mediante un middleware (authRequired) que protege ciertas rutas. Sin embargo, no se aplica un control granular de acceso por roles (por ejemplo, administrador, usuario estándar), lo que podría permitir que cualquier usuario autenticado acceda a recursos o funcionalidades que no le corresponden.

A02: Cryptographic Failures

La clave secreta utilizada para firmar los tokens JWT se encuentra expuesta directamente en el código fuente (TOKEN_SECRET = 'some secret key'), lo que representa un riesgo alto. Si un atacante accede al repositorio, podría generar tokens válidos y comprometer la autenticación.

A03: Injection

Las entradas de usuario en endpoints como /register y /login no se validan ni sanitizan adecuadamente antes de almacenarse en la base de datos. Esto abre la posibilidad a ataques de inyección de código o datos maliciosos, como MongoDB injection o almacenamiento de payloads ejecutables.

A05: Security Misconfiguration

El servidor Express no cuenta con configuraciones de seguridad adicionales, como el uso de cabeceras seguras (por ejemplo, mediante helmet) o políticas CORS específicas. Además, no se ha configurado HTTPS, exponiendo las comunicaciones a riesgos de intercepción.

A07: Identification and Authentication Failures

Aunque se utiliza bcrypt para cifrar contraseñas, no se implementan mecanismos de bloqueo tras múltiples intentos fallidos, ni control de expiración de tokens JWT. Esto podría facilitar ataques de fuerza bruta o reutilización de tokens comprometidos.

3. Descripción del riesgo e impacto

La siguiente tabla resume los **riesgos asociados** y el **posible impacto** que cada vulnerabilidad podría tener sobre el sistema “Sistema de Reservas”:

Vulnerabilidad (OWASP 2021)	Riesgo principal	Posible impacto en el sistema
A01: Broken Access Control	Acceso no autorizado a recursos protegidos por falta de control de roles y permisos.	Usuarios autenticados podrían manipular datos de otros usuarios, acceder a información sensible o ejecutar acciones administrativas no permitidas.
A02: Cryptographic Failures	Exposición de la clave secreta de JWT directamente en el código fuente.	Un atacante con acceso al repositorio podría generar tokens válidos y obtener acceso completo al sistema, comprometiendo la confidencialidad e integridad de los datos.

A03: Injection	Ausencia de validación/sanitización en entradas de usuario.	Inyección de comandos o datos maliciosos, manipulación o borrado de información en la base de datos, y potencial escalamiento de privilegios.
A05: Security Misconfiguration	Configuraciones de seguridad incompletas o por defecto en el servidor.	Exposición de información sensible, cabeceras inseguras, interceptación de tráfico no cifrado y mayor superficie de ataque.
A07: Identification and Authentication Failures	Falta de políticas de expiración, bloqueo y monitoreo de sesiones.	Ataques de fuerza bruta exitosos, reutilización de tokens comprometidos y suplantación de identidad.

4. Medidas de mitigación implementadas o planificadas

A01 – Broken Access Control

Para mitigar los riesgos de control de acceso insuficiente, se planifica implementar un sistema de roles y permisos que distinga entre usuarios con diferentes privilegios (por ejemplo, administrador y usuario estándar). Esto se logrará mediante un middleware adicional que verifique el rol del usuario antes de permitir el acceso a rutas críticas. Además, se definirán políticas de autorización claras para cada endpoint, evitando así accesos no autorizados a recursos sensibles.

A02 – Cryptographic Failures

Con el fin de proteger la información sensible y evitar la exposición de claves, la clave secreta utilizada para firmar los tokens JWT será extraída del código fuente y almacenada de forma segura en un archivo .env, el cual no se versionará en el repositorio. Posteriormente, esta variable será accedida mediante process.env.TOKEN_SECRET. Esta práctica asegura que las credenciales no estén expuestas públicamente, reduciendo considerablemente el riesgo de generación maliciosa de tokens.

A03 – Injection

Para prevenir ataques de inyección, se integrará la librería express-validator con el fin de validar y sanitizar las entradas de usuario en los formularios de registro e inicio de sesión. Cada campo será sometido a controles de tipo, longitud y formato antes de ser procesado o almacenado en la base de datos. Esta medida evitará la inserción de datos maliciosos, mejorando la integridad de la información almacenada.

A05 – Security Misconfiguration

Se reforzará la configuración de seguridad del servidor mediante la incorporación de la librería helmet, que permite establecer cabeceras HTTP seguras por defecto, protegiendo contra ataques comunes como clickjacking o XSS reflejado. Además, se configurará CORS

de forma explícita para restringir los orígenes permitidos, y se planifica migrar las comunicaciones a HTTPS, garantizando la confidencialidad de los datos transmitidos entre cliente y servidor.

A07 – Identification and Authentication Failures

Para fortalecer la autenticación, se implementará una política de expiración de tokens JWT, de manera que las sesiones no puedan mantenerse indefinidamente. Asimismo, se añadirá un sistema de registro de intentos fallidos de autenticación y un bloqueo temporal de cuentas tras múltiples intentos infructuosos, mitigando ataques de fuerza bruta. Estas medidas incrementarán la robustez del proceso de autenticación y la trazabilidad de actividades sospechosas.

5. Evidencias de verificación

A continuación se presentan las medidas de mitigación planificadas en el proyecto Sistema de Reservas:

5.1 Protección de claves secretas (A02 – Cryptographic Failures)

Antes, la clave utilizada para firmar tokens JWT se encontraba directamente en el código, Ahora, se pretende trasladar la clave a un archivo .env, y acceder a ella mediante variables de entorno. Esto evitará la exposición directa de la clave en el repositorio y permitirá una gestión más segura de credenciales.

5.2 Validación y sanitización de entradas (A03 – Injection)

Se integrará express-validator para validar y sanitizar los datos en las rutas de registro y login. Con estas validaciones se prevé la inserción de código malicioso en la base de datos.

5.3 Configuración segura del servidor (A05 – Security Misconfiguration)

La agregación de la librería helmet y configuraciones de CORS permitirá mejorar la seguridad del backend. Esta configuración reduce la exposición de cabeceras inseguras y restringe el acceso a orígenes no autorizados.

5.4 Pruebas de autenticación JWT (A07 – Identification and Authentication Failures)

Mediante Postman, se realizaron pruebas de autenticación utilizando tokens válidos y expirados. En el caso de un token expirado, el servidor devuelve correctamente un error 401 Unauthorized. Esto demuestra que el control de expiración de tokens está funcionando como se espera.

5.5 Auditoría de dependencias (npm audit)

Se ejecutó la herramienta integrada de auditoría de npm

Estas evidencias muestran que el sistema se encuentra en un proceso activo de endurecimiento de su backend, aplicando prácticas recomendadas por el estándar OWASP Top 10.

6. Reflexión personal

Nuestro equipo de trabajo comprendió que la seguridad no es un componente accesorio del desarrollo de software, sino un aspecto esencial que debe integrarse desde las primeras etapas del ciclo de vida del proyecto. El análisis de nuestro backend a la luz del estándar OWASP Top 10 (2021) nos permitió identificar vulnerabilidades concretas y frecuentes en aplicaciones web, como controles de acceso insuficientes, validación deficiente de entradas y configuraciones inseguras del servidor.

Este análisis nos permitió proyectar mejoras futuras: implementar autenticación multifactor, usar escáneres de seguridad automatizados, y aplicar cifrado de datos en reposo, entre otras prácticas. Como grupo, asumimos el compromiso de continuar aplicando estándares de seguridad reconocidos, no solo para cumplir con requisitos académicos, sino para desarrollar sistemas éticamente responsables y resilientes ante amenazas reales.