*Santiago Lorenzatti*

*Diseño de Sistemas de Información | Universidad Tecnológica Nacional FRSFCO*

*Gestor de tareas domésticas*

Este informe analiza posibles problemas de seguridad presentes en la aplicación Gestor de Tareas Domésticas y detalla las contramedidas necesarias para mitigarlos. Se describen riesgos, impactos y medidas de implementación específicas en los apartados siguientes.

**1. Almacenamiento inseguro en localStorage:**

Descripción técnica del riesgo: El uso de localStorage para guardar tokens o credenciales es inseguro porque cualquier script puede acceder a ellos. Esto expone al sistema a ataques de tipo XSS o al acceso físico del dispositivo.

Impacto en la aplicación: Un atacante podría robar sesiones de usuarios, manipular tareas y puntajes, o eliminar registros.

Contramedida técnica: La solución consiste en no almacenar tokens en localStorage. En su lugar, usar cookies seguras con las banderas HttpOnly, Secure y SameSite, de forma que no sean accesibles por JavaScript.

Implementación detallada: En el backend (Express + JWT) se deben emitir cookies seguras con el token. En el frontend, eliminar el uso de localStorage y configurar fetch/axios con credentials: 'include'.

**2. Falta de hash en contraseñas almacenadas:**

Descripción técnica del riesgo: Guardar contraseñas en texto plano es un error crítico, ya que una fuga de la base de datos expone las credenciales directamente. Esto se agrava porque los usuarios suelen reutilizar contraseñas.

Impacto en la aplicación: La exposición de credenciales permite a un atacante tomar el control del sistema y posiblemente de otros servicios externos usados por los usuarios.

Contramedida técnica: Las contraseñas deben almacenarse con hash y salt utilizando algoritmos resistentes como bcrypt, Argon2 o scrypt.

Implementación detallada: En el registro de usuarios, aplicar bcrypt.hash(password, 10) antes de guardar. En el login, usar bcrypt.compare para verificar la contraseña.

**3. Gestión insegura de sesiones (JWT sin protección suficiente):**

Descripción técnica del riesgo: Un JWT robado permite acceso total mientras sea válido. Si no expira o tiene tiempos demasiado largos, el atacante conserva acceso indefinido.

Impacto en la aplicación: El atacante podría manipular información de usuarios, modificar tareas o alterar rankings.

Contramedida técnica: Usar tokens con expiración corta (ej. 15 minutos) y refresh tokens almacenados en cookies seguras. Revocar tokens comprometidos desde el backend.

Implementación detallada: En backend, generar JWT con expiresIn: '15m'. Gestionar refresh tokens en base de datos y cookies. En frontend, no almacenar tokens manualmente, confiar en cookies.

**4. CORS mal configurado:**

Descripción técnica del riesgo: Si el servidor permite cualquier origen con Access-Control-Allow-Origin: \*, un atacante puede disparar peticiones maliciosas usando las cookies del usuario legítimo.

Impacto en la aplicación: Esto permite ataques CSRF que borren tareas, alteren datos o manipulen puntajes.

Contramedida técnica: Configurar CORS para aceptar solo dominios de confianza.

Implementación detallada: En Express, usar cors({ origin: ['http://localhost:5173','https://miapp.com'], credentials: true }).

**5. XSS (Cross-Site Scripting) en formularios / inputs**

Descripción técnica del riesgo: Los ataques XSS permiten inyectar scripts en campos de texto. Si se almacenan y luego renderizan sin sanitización, se ejecutan en el navegador de otro usuario.

Impacto en la aplicación: El atacante podría robar cookies, tokens, credenciales o manipular la interfaz para phishing.

Contramedida técnica: Validar y sanitizar entradas en backend y escapar contenido en frontend. Usar librerías como validator y DOMPurify.

Implementación detallada: En backend, validar campos con validator.isAlphanumeric o similares. En frontend, usar DOMPurify.sanitize antes de mostrar HTML dinámico.