Audit de sécurité - Client Web

Auditeur : YOUYOU Khatir

1 décembre 2025

Cible : `src/client/web/`

1 Gestion des Secrets et Configuration

Risque Identifié : Clés d'API en dur

Le fichier de configuration Firebase contient des clés d'API et des identifiants de projet codés en dur directement dans le code source. Bien que les clés Firebase soient souvent considérées comme publiques, il est recommandé de ne pas les exposer dans le dépôt Git pour éviter les abus de quota ou le ciblage par des bots.

- **Fichier concerné :** `src/client/web/src/firebase-config.ts`
- **Code problématique :**

```
const firebaseConfig = {
  apiKey: "AIzaSyDvFfs-_AqsIn2UHvGUpoJUvhomsKLRu7E",
  authDomain: "demperm-153aa.firebaseapp.com",
  // ...
};
```

- **Recommandation :** Utilisez des variables d'environnement (fichiers `.env`) via Vite (`import.meta.env.VITE`

2 Authentification et Gestion de Session

Risque Potentiel : Stockage des Tokens

Dans le composant de connexion, le token d'identité (ID Token) est récupéré et passé à une fonction `setCredentials`.

- **Fichier concerné :** `src/client/web/src/app/auth/login.tsx`
- **Code observé :**

```
const idToken = await user.getIdToken();
setCredentials(user.email ?? username, idToken);
```

- **Analyse :** Si `setCredentials` stocke ce token dans le `localStorage` (pratique courante mais risquée), l'application est vulnérable aux attaques XSS (Cross-Site Scripting). Tout script malveillant injecté dans la page pourrait exfiltrer ce token.
- **Recommandation :** Privilégiez le stockage des tokens sensibles en mémoire ou via des cookies `HttpOnly` sécurisés gérés par le backend, si possible.

3 Isolation Environnement de Développement

Point Positif : Isolation du Mock Service Worker (MSW)

L'initialisation des mocks d'API est correctement conditionnée à l'environnement de développement. Cela empêche le code de test et les handlers de mock de fuiter en production.

- **Fichier concerné :** `src/client/web/src/main.jsx`
- **Code validé :**

```
// Only load MSW in development (never in production)
if (import.meta.env.DEV) {
  import('./tests/mock')
    .then(({ worker }) => worker.start())
}
```

4 Qualité du Code et Linting

Amélioration Possible : Règles de Sécurité ESLint

La configuration ESLint actuelle est standard pour React mais n'inclut pas de plugins spécifiques à la sécurité.

- **Fichier concerné :** `src/client/web/eslint.config.js`
- **Recommandation :** Ajouter `eslint-plugin-security` ou `eslint-plugin-no-unsanitized` pour détecter automatiquement des patterns dangereux (comme `dangerouslySetInnerHTML` ou des regex vulnérables au ReDoS).

5 Protection XSS (Cross-Site Scripting)

Point Positif : Utilisation standard de React

Les composants UI inspectés utilisent le rendu standard de React (JSX), qui échappe automatiquement les variables, protégeant contre la plupart des injections XSS basiques.

- **Exemple :** `src/client/web/src/components/ui/Avatar.jsx`

```
{src ? (
  <img src={src} alt={alt} className="h-full w-full object-cover" />
) : (
  // ...
)}
```

Note : Il faut tout de même s'assurer que les URLs passées dans `src` (pour les images ou les liens) sont validées pour éviter les schémas `javascript:`.

6 Résumé

L'application présente une structure saine avec une bonne séparation Dev/Prod. La priorité immédiate doit être de **sortir les secrets du code source** (`src/client/web/src/firebase-config.ts`) vers des variables d'environnement.