Rapport de tests de sécurité et vulnérabilités

DigitalBank

22 janvier 2026

1 Tests de sécurité et rapport de vulnérabilité

Cette section présente les résultats des tests de sécurité réalisés sur l'API REST de la plateforme *DigitalBank*. L'objectif est d'identifier d'éventuelles vulnérabilités, d'évaluer les mécanismes de protection existants et de proposer des recommandations d'amélioration.

1.1 Vulnérabilités identifiées

Plusieurs tests de sécurité ont été réalisés : tentative d'injection SQL, accès non authentifié, contournement des permissions RBAC et tentative de brute force sur le login.

- **Injection SQL :** Aucune vulnérabilité exploitable n'a été détectée. Les tentatives d'injection via les paramètres de requête ont échoué (erreur de typage), ce qui indique que les paramètres sont traités comme des valeurs et non comme du code SQL.
- **Accès sans authentification :** L'accès aux endpoints sensibles sans jeton JWT valide a été refusé par l'API (réponse de type *401 Unauthorized*).
- **Bypass RBAC :** Les tentatives d'exécution d'actions réservées à l'administrateur (ex. suppression de transaction) avec un jeton client ont été bloquées (réponse *403 Forbidden*), confirmant la mise en application des permissions.
- **Brute force sur le login :** Des tentatives répétées de connexion avec des identifiants incorrects n'ont jamais permis d'obtenir un jeton valide. Les messages d'erreur restent génériques (*invalid credentials*) et ne divulguent pas d'information sensible.

Conclusion : aucune vulnérabilité critique exploitable n'a été identifiée lors des tests réalisés.

1.2 Mesures de protection mises en place

Les mécanismes de sécurité suivants ont été identifiés et validés durant les tests :

- **Authentification JWT :** utilisation de jetons d'accès pour authentifier les requêtes.
- **Contrôle d'accès (RBAC) :** restrictions des actions selon les rôles (admin, client, analyst, customer_service).
- **Row Level Security (RLS) :** isolement des données afin qu'un client n'accède qu'à ses propres enregistrements.
- **Protection contre l'injection SQL :** requêtes paramétrées et typage strict des paramètres.

— **Clé API requise :** présence obligatoire d'une clé API pour accéder à l'API REST.

1.3 Recommandations d'amélioration

Bien que l'API présente un niveau de sécurité satisfaisant, plusieurs améliorations sont recommandées :

— Mettre en place un **rate limiting** explicite sur les endpoints sensibles (authentification, opérations d'écriture).
— Ajouter des **en-têtes de sécurité HTTP** (ex. Content-Security-Policy, X-Frame-Options, X-Content-Type-Options) si non présents.
— Renforcer la **journalisation** (audit logs) des actions sensibles (INSERT/UPDATE/DELETE) et mettre en place des alertes.
— Planifier des **scans réguliers** (OWASP ZAP) et des revues de configuration RLS/RBAC.
— Mettre en œuvre une **surveillance** des échecs d'authentification et comportements anormaux (monitoring).

Conclusion générale : les tests montrent que l'API est correctement sécurisée contre des attaques courantes (SQLi, accès non authentifié, bypass RBAC, tentative de brute force). Les recommandations ci-dessus visent à renforcer la défense en profondeur.