# Rapport d'analyse de sécurité du *Client Android*

## Analyse fonctionnelle et technique

*Projet : Client Android*

**Auteur :**

BOUSSALEF El Mokhtar

Date : 17 novembre 2025

Ce document présente une analyse de la sécurité du projet de « Client Android » couvrant la sécurité fonctionnelle (usages, modération, protection des usagers) et la sécurité technique (architecture, données, résilience).

# Table des matières

# 1 Introduction

Le projet DemPerm fournit une application mobile Android développée en React, permettant d'interagir avec le serveur Social et le serveur Vote. Le présent document décrit la conception fonctionnelle et les exigences de sécurité du **client Android**, en couvrant l'accès aux fonctionnalités sociales, de vote, de messagerie privée, ainsi que la gestion d'identité et de confidentialité.

# 2 Description fonctionnelle du client Android

## 2.1 Parcours utilisateurs principaux

**Authentification et accès**. L'utilisateur se connecte, puis l'application récupère son profil via `GET /users/me`. Une fois authentifié, l'accès est donné aux sections Votes, Élus, Débats, Messages privés et Profil.

**Consultation et gestion des votes**. L'application permet :

- voter via `POST /votes`,
- retirer un vote via `DELETE /votes/{domain}`,
- consulter ses votes via `GET /votes/by-voter/{voterId}`,
- visualiser les élus (`GET /elected`) et les résultats (`GET /results`),
- gérer le statut public/anonyme (`GET /publication/{userId}`, `PUT /publication/{userId}`).

**Interaction sociale**. L'utilisateur peut :

- consulter et éditer son profil (`GET /users/me`, `PATCH /users/me`),
- consulter un autre profil (`GET /users/{id}`),
- suivre / ne plus suivre (`POST /friends/{id}`, `DELETE /friends/{id}/delete`),
- naviguer dans les forums (`GET /topics/`),
- créer un forum (`POST /topics/create`),
- consulter des posts (`GET /topics/{id}`),
- s'abonner / se désabonner d'un thème (`POST /subscriptions/topics/{id}`, `DELETE /subscriptions/topics/{id}/unsubscribe`),
- publier un message ou répondre (`POST /topics/{id}/comments/add`),
- consulter des commentaires (`GET /topics/{id}/comments`),
- liker / disliker (`POST /comments/{id}/like`, `POST /comments/{id}/dislike`),
- annuler like/dislike (`DELETE /comments/{id}/delete`).

**Messagerie privée**. Envoi de messages privés via `POST /messages/send/{receiverId}` et consultation via `GET /messages/{userId}`.

**Blocage d'utilisateurs (prévu)**. Blocage via `POST /users/{id}/block` et déblocage via `DELETE /users/{id}/unblock`.

**Recherche**. Recherche de contenus sociaux ou utilisateurs via `/search` (à préciser côté backend).

### 2.2 Synthèse des endpoints utilisés

Le client Android fait appel à :

— Endpoints Social : gestion de profil, abonnements, forums, posts, commentaires, MP, blocage.

— Endpoints Vote : vote, retrait, élus, tendances, statut public/anonyme.

Ces endpoints sont mappés à chaque écran fonctionnel selon les interactions décrites ci-dessus.

# 3 Analyse des risques côté client Android

## 3.1 Authentification et gestion de session

**Risques identifiés** :

— vol de session en cas de stockage non sécurisé du token,

— absence de mécanisme clair d'expiration de session,

— fuite de tokens dans les logs ou dans le stockage local.

**Exigences** :

— stockage sécurisé (Keystore / mécanisme adapté React – Android),

— suppression complète du token lors de la déconnexion,

— redirection vers l'écran de connexion si token expiré.

## 3.2 Confidentialité du vote et statut public/anonyme

**Risques** :

— confusion utilisateur entre mode public et mode anonyme,

— affichage accidentel d'informations permettant de déduire l'identité des votants,

— incohérences entre backend et client concernant les données "publiques".

**Exigences** :

— indication explicite de l'état actuel (public/anonyme),

— confirmation lors du changement de statut,

— aucune fuite d'identité lorsqu'un vote est anonyme.

## 3.3 Vie privée sociale et messagerie privée

**Risques** :

— exposition des messages dans les notifications,

— stockage durable de messages privés sur le téléphone,

— fuite dans les logs de debug.

**Exigences** :

— notifications non sensibles,

— absence de stockage local hors cache strictement nécessaire,

— pas de logs contenant du contenu sensible.

### 3.4 Forums, commentaires et interactions

**Risques** :

— usurpation visuelle d'un espace privé comme public,

— usage abusif de like/dislike,

— incohérences UI entre actions possibles et autorisations serveur.

**Exigences** :

— distinction visuelle claire des espaces,

— cohérence complète like/dislike/suppression,

— revalidation côté serveur même si UI désactive certaines actions.

### 3.5 Blocage d'utilisateurs

**Risques** :

— blocage incomplet (accès résiduel à des écrans),

— confusion sur l'état bloqué/débloqué.

**Exigences** :

— état de blocage respecté dans tous les écrans,

— affichage explicatif clair lors du blocage/déblocage.

### 3.6 Recherche

**Risques** :

— exposition dans les résultats de données d'utilisateurs souhaitant rester anonymes,

— possibilité d'énumérer des profils (user enumeration).

**Exigences** :

— respect strict du statut public/anonyme,

— limitation des métadonnées renvoyées par la recherche.

# 4 Exigences de sécurité techniques

## 4.1 Communication réseau

— utilisation exclusive du HTTPS vers les serveurs Social et Vote,

— rejet des certificats non valides en production,

— absence d'endpoints de test ou non sécurisés dans le client final.

## 4.2 Stockage local et tokens

— aucun stockage en clair de mots de passe ou tokens,

— cache minimal, effaçable, sans données sensibles,

— déconnexion effaçant toutes les données identifiantes.

## 4.3 Permissions Android

— permissions limitées au strict nécessaire,

— absence de `android:debuggable=true` en production,

— pas de logs ou traces en production contenant des données personnelles.

## 4.4 UX de sécurité

— indications visuelles explicites pour les actions sensibles,

— messages d'erreur génériques côté UI,

— confirmations pour changements de statut, votes, blocage.

# 5 Conclusion

La sécurité du client Android repose sur :

— une gestion solide des tokens et de l'authentification,

— une interface cohérente pour distinguer Public / Privé / Anonyme,

— une communication réseau strictement sécurisée,

— la protection des messages privés et des données sociales.

**Priorités immédiates** :

1. sécurisation du stockage des tokens et des flux d'authentification,
2. clarification UX du statut de publication,
3. durcissement des notifications et logs,
4. vérification systématique de l'usage des endpoints sensibles.