

RAPPORT TECHNIQUE D'ANALYSE DE SÉCURITÉ ANDROID

INFORMATIONS GÉNÉRALES

Application analysée : app.apk

Hash SHA256 : f01f08c8b6e2fe4612e81bc7e3a3ba9440dad0d7a962f4e67640390dd721d528

Date d'analyse : Septembre 2024

Consultant sécurité : Équipe d'audit sécurité

Note de sécurité MobSF : 49/100

RÉSUMÉ EXÉCUTIF

L'analyse de sécurité de l'application Android révèle plusieurs vulnérabilités critiques compromettant la confidentialité, l'intégrité et la sécurité des données utilisateurs. Les principales préoccupations incluent l'exposition de clés d'API sensibles, l'utilisation de méthodes de chiffrement obsolètes, et des vulnérabilités d'injection SQL.

MÉTHODOLOGIE

L'audit a été réalisé selon une approche structurée en 9 étapes :

- Vérification d'intégrité** - Validation du hash SHA256
- Analyse automatisée** - Scan MobSF pour identification rapide
- Extraction de composants** - Récupération des fichiers critiques
- Décompilation** - Analyse des ressources avec Apktool
- Analyse des chaînes** - Recherche de secrets dans strings.xml
- Analyse de code** - Décompilation du bytecode avec Jadx
- Recompilation** - Validation de la réversibilité
- Signature de l'APK** - Signature avec clés de test pour validation
- Test Firebase** - Vérification d'accès public à la base de données

FINDINGS ET VULNÉRABILITÉS

❑ CRITIQUE - Exposition de clés d'API

Localisation : strings.xml

Description : Clé API Google exposée en clair

```
<string name="google_api_key">AIzaSyBTgtvImsUfMWda41PCrDWAj7dmyIDhUg</string>
```

Impact : Utilisation malveillante de l'API Google par des tiers

Recommandation : Stocker les clés d'API côté serveur ou utiliser l'Android Keystore

❑ CRITIQUE - Exposition d'endpoints sensibles

Localisation : strings.xml

Description : URLs des services bancaires exposées

```
<string name="ACCOUNT_ENDPOINT">https://api.nickel.eu/customer-banking-api</string>
<string name="CUSTOMER_AUTHENTICATION_ENDPOINT">https://api.nickel.eu/customer-authentication-api</string>
```

Impact : Points d'entrée pour des attaques ciblées

Recommandation : Obfuscation des endpoints, validation côté serveur renforcée

❑ CRITIQUE - Injection SQL

Localisation : com/pushwoosh/inapp/e/b/b.java

Description : Requête SQL non échappée

```
sQLiteDatabase.rawQuery("Select * from inApps where code='" + str + "';", null);
```

Impact : Exécution de code arbitraire via injection SQL

Recommandation : Utiliser des requêtes préparées (PreparedStatement)

❑ ÉLEVÉ - Chiffrement faible

Localisation : CipherStorageKeystoreAESCBC.java

Description : Utilisation de CBC avec PKCS7Padding

Impact : Vulnérable aux attaques "padding oracle"

Recommandation : Migrer vers AES-GCM ou utiliser AEAD

❑ ÉLEVÉ - Communication non sécurisée

Description : Utilisation de ClearText détectée par MobSF

Impact : Interception des communications en clair

Recommandation : Forcer HTTPS et implémenter certificate pinning

❑ MOYEN - Permissions excessives

Description : Demande de permissions non justifiées

- READ_CONTACTS
- WRITE_EXTERNAL_STORAGE

Impact : Collecte de données non nécessaires

Recommandation : Appliquer le principe du moindre privilège

❑ MOYEN - Version Android obsolète

Description : Support d'Android 4.4 (API 19)

Impact : Exposition aux vulnérabilités système anciennes

Recommandation : Définir minSdkVersion ≥ 23 (Android 6.0)

❑ FAIBLE - Base Firebase sécurisée

Localisation : `https://application-client-nickel.firebaseio.com/`

Description : Test d'accès public à la base Firebase

Résultat : `{"error" : "Permission denied"}`

Impact : Accès correctement restreint

Recommandation : Maintenir les règles de sécurité Firebase actuelles

PREUVES TECHNIQUES

Hash de vérification

SHA256: `f01f08c8b6e2fe4612e81bc7e3a3ba9440dad0d7a962f4e67640390dd721d528`

Fichiers extraits

- `classes.dex` - Bytecode principal
- `AndroidManifest.xml` - Configuration et permissions
- `app_src/` - Code source décompilé (Apktool)

Recompilation et signature

- `app_unsigned.apk` - APK recompilé avec succès
- `apktool_build.log` - Log de construction sans erreurs
- `rebuilt_signed.apk` - APK signé avec clés de test
- `app_keystore.jks` - Keystore généré pour signature
- Validation signature avec `apksigner verify`

Tests de sécurité externes

- **Firebase** : Test d'accès public - `Permission denied` (sécurisé)
- **Endpoints API** : URLs bancaires exposées mais protection côté serveur présumée

RECOMMANDATIONS PRIORITAIRES

☐ Actions immédiates

1. **Retirer les clés d'API** du code source
2. **Corriger l'injection SQL** dans le module Pushwoosh
3. **Désactiver ClearText** trafic
4. **Revoir les permissions** demandées

☐ Plan de remédiation

Priorité	Action	Délai	Effort
P0	Suppression clés API	1 jour	Faible
P0	Fix injection SQL	2 jours	Moyen
P1	Migration chiffrement	1 semaine	Élevé
P1	HTTPS forcé	3 jours	Moyen
P2	Révision permissions	2 jours	Faible
P2	Mise à jour minSDK	1 semaine	Élevé

☐ Mesures préventives

- Mise en place d'une revue de code sécurisée
- Tests de sécurité automatisés dans la CI/CD
- Formation des développeurs aux bonnes pratiques
- Scanning périodique avec MobSF

CONCLUSION

L'application présente des vulnérabilités critiques nécessitant une action immédiate. La note de sécurité de 49/100 reflète ces problèmes majeurs. Avec les corrections recommandées, l'application pourrait atteindre un niveau de sécurité acceptable pour un environnement de production.

Recommandation finale : Bloquer le déploiement en production jusqu'à correction des vulnérabilités P0 et P1.

Rapport généré automatiquement - Validation manuelle requise