# Directives de Sécurité des Applications Mobiles pour les Solutions de Paiement

**Neo Financia**
**Version:** 1.0
**Date:** 21 avril 2025
**Classification:** Confidentiel (P2)
**Propriétaire:** RSSI
**Approbation:** Comité de Sécurité (COSEC)

## Table des matières



## 1. Introduction et objectifs

### 1.1 Contexte et importance

Neo Financia, en tant que néobanque européenne opérant principalement à travers des canaux digitaux, reconnaît l'importance critique de la sécurité de ses applications mobiles. Avec 80% de notre clientèle active utilisant notre application mobile, la sécurité de cette plateforme constitue un pilier fondamental de notre stratégie de cybersécurité globale.

Les applications mobiles de paiement représentent une cible privilégiée pour les attaquants en raison de la sensibilité des données financières traitées. Une compromission pourrait entraîner des conséquences graves en termes financiers, réglementaires et réputationnels.

### 1.2 Portée des directives

Ces directives s'appliquent à :

- L'application mobile Neo Financia sur iOS et Android
- Toutes les fonctionnalités de paiement (virements, paiements instantanés, NFC)
- Les intégrations avec les solutions tierces (Apple Pay, Google Pay, Samsung Pay)
- L'ensemble du cycle de vie de l'application, de la conception à la maintenance
- Toutes les équipes impliquées dans le développement, le test et la maintenance de l'application

### 1.3 Relation avec la PSSI globale

Ces directives constituent une extension spécifique de la Politique de Sécurité des Systèmes d'Information (PSSI) de Neo Financia. Elles précisent et complètent les exigences générales de la PSSI dans le contexte spécifique des applications mobiles de paiement.

En cas de contradiction apparente entre ce document et la PSSI, la règle la plus restrictive s'applique. Toute dérogation doit suivre le processus d'exception défini dans la PSSI.

# 2. Principes fondamentaux

## 2.1 Security by Design et by Default

- La sécurité est intégrée dès la phase de conception, et non ajoutée après coup
- Les configurations par défaut doivent être les plus restrictives possibles
- Toute fonctionnalité sensible doit être désactivée par défaut et activée explicitement
- Le principe de moindre surprise pour l'utilisateur doit être appliqué pour les fonctionnalités de sécurité

## 2.2 Defense in Depth

- Multiples couches de sécurité complémentaires pour protéger les données sensibles
- Aucune mesure de sécurité ne doit être considérée comme infaillible
- Les contrôles doivent être redondants pour les fonctionnalités critiques
- Les défaillances d'un mécanisme de sécurité ne doivent pas compromettre l'ensemble du système

## 2.3 Zero Trust

- Aucune confiance implicite, que ce soit pour les composants internes ou externes
- Vérification systématique à chaque étape d'une transaction
- Validation continue du contexte d'utilisation
- Authentification et autorisation pour chaque accès aux ressources sensibles

## 2.4 Least Privilege et Data Minimization

- Limitation des accès au strict nécessaire pour chaque fonctionnalité
- Collecte uniquement des données strictement nécessaires
- Conservation des données pendant la durée minimale requise
- Cloisonnement strict entre les différentes fonctionnalités

# 3. Cycle de développement sécurisé

## 3.1 Exigences et modélisation des menaces

### 3.1.1 Analyse des risques spécifiques

- Réaliser une modélisation des menaces (STRIDE) pour chaque fonctionnalité de paiement
- Documenter les scénarios d'attaque potentiels et leurs contre-mesures
- Évaluer l'impact potentiel de chaque menace identifiée
- Mettre à jour l'analyse à chaque évolution significative de l'application

### 3.1.2 Exigences de sécurité

- Intégrer les exigences de sécurité dès le cahier des charges
- Respecter les standards OWASP MASVS (Mobile Application Security Verification Standard) niveau L2 minimum
- Pour les fonctionnalités de paiement critiques, appliquer le niveau MASVS-L3
- Documenter les contrôles de sécurité pour chaque exigence fonctionnelle

### 3.2 Développement sécurisé

#### 3.2.1 Standards de code sécurisé

- Appliquer les règles de codage sécurisé spécifiques à chaque plateforme (iOS, Android)
- Utiliser les bibliothèques de sécurité standard et maintenues à jour
- Éviter le développement de fonctions cryptographiques propriétaires
- Respecter les recommandations OWASP Mobile Top 10

#### 3.2.2 Revue de code

- Effectuer des revues de code systématiques par des pairs
- Automatiser l'analyse statique du code (SAST) dans le pipeline CI/CD
- Réaliser des revues manuelles approfondies pour le code critique lié aux paiements
- Documenter et suivre la résolution des problèmes identifiés

### 3.3 Tests de sécurité

#### 3.3.1 Tests automatisés

- Intégrer des tests de sécurité automatisés dans le pipeline CI/CD
- Analyser les dépendances tierces (SCA) pour identifier les vulnérabilités
- Appliquer l'analyse dynamique (DAST) sur l'application compilée
- Mettre en œuvre des tests de fuzzing sur les interfaces sensibles

#### 3.3.2 Tests manuels

- Réaliser des tests d'intrusion approfondis par des spécialistes en sécurité mobile
- Effectuer des tests avant chaque mise à jour majeure
- Tester spécifiquement les flux de paiement de bout en bout
- Vérifier la résistance aux techniques avancées (hooking, instrumentation)

### 3.4 Validation et déploiement

#### 3.4.1 Critères de validation

- Définir des critères de validation de sécurité formels ("security gates")
- Obtenir l'approbation du RSSI pour les fonctionnalités de paiement critiques
- Vérifier la conformité aux exigences réglementaires (DSP2/PSD2, SCA)
- Documenter les résultats des tests de sécurité

#### 3.4.2 Processus de déploiement

- Signer numériquement toutes les versions de l'application
- Vérifier l'intégrité des packages avant leur publication
- Mettre en place un processus de déploiement progressif (rollout)
- Surveiller activement les indicateurs de sécurité post-déploiement

## 4. Protection du code et de l'application

### 4.1 Obfuscation et protection du code

- Appliquer des techniques d'obfuscation adaptées à chaque plateforme
- Renommer les classes, méthodes et variables critiques
- Masquer les chaînes de caractères sensibles dans le code
- Implémenter la protection contre la décompilation et le reverse engineering

### 4.2 Anti-tampering et vérification d'intégrité

- Implémenter des mécanismes de détection de modification de l'application
- Vérifier l'intégrité des ressources critiques au démarrage et périodiquement
- Mettre en place des réponses graduées en cas de détection de falsification
- Signaler les tentatives de manipulation au serveur pour analyse

### 4.3 Détection des environnements compromis

- Détecter les appareils rootés (Android) ou jailbreakés (iOS)
- Appliquer une politique stricte d'interdiction d'utilisation sur appareils compromis
- Mettre en œuvre des contrôles secondaires en cas de contournement de la détection primaire
- Documenter clairement cette politique dans les conditions d'utilisation

### 4.4 Protection contre le debugging

- Implémenter des mesures anti-debugging
- Détecter les tentatives d'attachement de debuggers
- Bloquer les outils d'analyse dynamique comme Frida
- Appliquer des contre-mesures en cas de détection (dégradation fonctionnelle, alertes)

## 5. Authentification et gestion des sessions

### 5.1 Méthodes d'authentification

#### 5.1.1 Options d'authentification

- Proposer la biométrie (empreinte digitale, reconnaissance faciale) comme option principale
- Maintenir le code PIN comme alternative obligatoire
- Appliquer des règles de complexité pour les codes PIN (min. 6 chiffres, pas de séquences)
- Permettre à l'utilisateur de choisir sa méthode préférée

#### 5.1.2 Authentification initiale

- Exiger une authentification forte lors de l'enrôlement de l'appareil
- Établir un canal sécurisé pour l'échange des credentials
- Lier de manière cryptographique l'identité de l'utilisateur à l'appareil
- Mettre en place un processus de ré-authentification périodique

### 5.2 Authentification adaptative

- Adapter le niveau d'authentification en fonction du risque évalué
- Prendre en compte le comportement de l'utilisateur, la localisation, le montant
- Exiger une authentification renforcée pour les opérations sensibles
- Détecter et réagir aux comportements anormaux

### 5.3 Gestion des sessions

- Mettre en place des tokens de session à durée limitée (15 minutes maximum)

- Renouveler les tokens de manière sécurisée
- Implémenter un délai d'inactivité avec déconnexion automatique (5 minutes par défaut)
- Permettre à l'utilisateur de déconnecter toutes ses sessions actives

### 5.4 Authentification pour les paiements

- Appliquer l'authentification forte du client (SCA) conforme à DSP2/PSD2
- Exiger systématiquement une authentification pour les paiements, quelle que soit la méthode
- Implémenter un mécanisme de validation hors bande pour les transactions à haut risque
- Appliquer des mesures anti-rejeu pour prévenir la réutilisation des authentifications

## 6. Stockage sécurisé des données

### 6.1 Minimisation des données stockées

- Limiter les données stockées localement au strict minimum nécessaire
- Éviter de stocker les données de carte complètes sur l'appareil
- Privilégier l'utilisation de tokens de paiement plutôt que les données réelles
- Mettre en œuvre une politique d'effacement automatique des données temporaires

### 6.2 Sécurisation des données sensibles

- Utiliser le stockage sécurisé natif de la plateforme (Keychain iOS, Keystore Android)
- Chiffrer toutes les données sensibles au repos avec des algorithmes robustes (AES-256)
- Isoler les données de paiement dans un conteneur sécurisé
- Mettre en œuvre une protection supplémentaire pour les appareils anciens

### 6.3 Gestion des clés et secrets

- Stocker les clés cryptographiques dans les éléments sécurisés quand disponibles
- Dériver les clés de chiffrement de manière sécurisée
- Ne jamais stocker de clés en dur dans le code (hardcoding)
- Mettre en place une rotation périodique des clés et secrets

### 6.4 Protection contre la fuite de données

- Désactiver les captures d'écran pendant l'affichage d'informations sensibles
- Bloquer le copier-coller des données critiques
- Nettoyer la mémoire après utilisation de données sensibles
- Implémenter des contrôles pour prévenir les fuites via les logs ou le presse-papier

## 7. Communications sécurisées

### 7.1 Sécurisation des connexions

- Utiliser exclusivement TLS 1.3 pour toutes les communications
- Implémenter le certificate pinning pour prévenir les attaques MitM
- Refuser les connexions en cas d'échec de validation du certificat
- Vérifier la révocation des certificats

### 7.2 Protection des API

- Authentifier chaque appel API avec des tokens signés à courte durée de vie
- Protéger les paramètres sensibles avec un chiffrement supplémentaire
- Implémenter des nonces pour prévenir les attaques par rejeu
- Valider systématiquement toutes les entrées côté serveur

### 7.3 Sécurité NFC

- Respecter les standards EMV pour les paiements NFC
- Limiter les données transmises au strict nécessaire
- Utiliser la tokenisation pour les paiements sans contact
- Mettre en œuvre des contrôles supplémentaires pour les transactions à risque

### 7.4 Détection des attaques réseau

- Détecter les tentatives d'interception SSL/TLS
- Identifier les proxys de débogage et outils d'analyse réseau
- Alerter en cas de configuration réseau suspecte
- Implémenter des contre-mesures en cas de détection d'attaque

## 8. Sécurité des paiements

### 8.1 Tokenisation des données de paiement

- Utiliser la tokenisation pour toutes les opérations de paiement
- Ne jamais stocker les PAN (Primary Account Number) complets sur l'appareil
- Mettre en œuvre des tokens à usage unique ou limité pour les paiements
- Utiliser les standards du secteur (EMV Payment Tokenisation)

### 8.2 Intégration des solutions tierces

- Intégrer Apple Pay, Google Pay et Samsung Pay conformément aux exigences de sécurité des fournisseurs
- Implémenter les contrôles complémentaires requis par Neo Financia
- Valider la sécurité de chaque intégration avant déploiement
- Maintenir une veille sur les vulnérabilités des SDK tiers

### 8.3 Validation des transactions

- Mettre en œuvre une validation multi-niveaux pour les transactions
- Vérifier la cohérence des données de transaction de bout en bout
- Appliquer des contrôles anti-fraude avant validation finale
- Implémenter une signature cryptographique des transactions

### 8.4 Conformité PCI-DSS

- Respecter les exigences PCI-DSS applicables au canal mobile
- Réaliser des audits de conformité réguliers
- Former les équipes de développement aux exigences PCI
- Documenter les mesures mises en place pour assurer la conformité

## 9. Détection et prévention des fraudes

### 9.1 Analyse comportementale

- Mettre en place une analyse comportementale des utilisateurs
- Établir des profils d'utilisation typiques
- Détecter les déviations significatives par rapport aux comportements habituels

- Adapter les contrôles en fonction du niveau de risque détecté

### 9.2 Contrôles transactionnels

- Appliquer des contrôles renforcés pour les montants élevés
- Mettre en place une gradation des contrôles en fonction du risque
- Implémenter des délais de sécurité pour certaines opérations sensibles
- Définir des seuils d'alerte paramétrables par l'utilisateur

### 9.3 Surveillance des appareils

- Surveiller les changements d'appareil ou de comportement
- Détecter les tentatives d'automatisation ou de scripting
- Identifier les patterns suspects (vitesse de frappe, navigation)
- Alerter sur les connexions depuis des localisations inhabituelles

### 9.4 Alertes et notifications

- Notifier l'utilisateur en temps réel pour les transactions sensibles
- Proposer des options de confirmation ou de rejet immédiats
- Adapter le canal de notification selon la criticité
- Permettre à l'utilisateur de configurer ses préférences de notification

## 10. Gestion des incidents et vulnérabilités

### 10.1 Mises à jour de sécurité

- Mettre en place un processus de déploiement rapide des correctifs de sécurité
- Implémenter un canal de mise à jour prioritaire pour les vulnérabilités critiques
- Forcer la mise à jour des applications présentant des vulnérabilités graves
- Communiquer de manière transparente sur les correctifs de sécurité

### 10.2 Surveillance et détection

- Mettre en œuvre une surveillance en temps réel des comportements anormaux
- Collecter les logs pertinents pour analyse (dans le respect du RGPD)
- Détecter les tentatives d'exploitation de vulnérabilités connues
- Alerter l'équipe de sécurité en cas d'incident potentiel

### 10.3 Réponse aux incidents

- Définir un plan de réponse spécifique aux incidents de sécurité mobile
- Mettre en place des capacités de blocage à distance des fonctionnalités compromises
- Prévoir des procédures de communication vers les utilisateurs affectés
- Coordonner la réponse avec les équipes backend et les partenaires de paiement

### 10.4 Reporting et amélioration continue

- Documenter tous les incidents de sécurité affectant l'application mobile
- Réaliser une analyse post-incident systématique
- Intégrer les enseignements dans le cycle de développement
- Mettre à jour les contrôles en fonction des nouvelles menaces identifiées

## 11. Exigences spécifiques par plateforme

### 11.1 iOS

- Utiliser les API de sécurité Apple (Keychain, Secure Enclave, App Transport Security)
- Implémenter l'App Attest pour renforcer la sécurité des applications
- Respecter les directives de sécurité de l'App Store
- Mettre à profit les fonctionnalités de confidentialité d'iOS

### 11.2 Android

- Appliquer les recommandations de sécurité Android de Google
- Utiliser les API de sécurité Android (Keystore, Strongbox, SafetyNet)
- Implémenter le Play Integrity API pour la vérification de l'intégrité
- Adapter les protections aux différentes versions d'Android

### 11.3 Exigences communes

- Maintenir la compatibilité avec les dernières versions du système d'exploitation
- Cesser le support des versions obsolètes ne recevant plus de mises à jour de sécurité
- Tester sur un éventail représentatif d'appareils
- Optimiser les contrôles selon les capacités matérielles de l'appareil

## 12. Annexes et références

### 12.1 Checklist de validation de sécurité

Liste de contrôles à vérifier avant toute mise en production :

1. Vérification des contrôles de protection du code
2. Tests de pénétration réalisés et problèmes critiques corrigés
3. Analyse des vulnérabilités des dépendances
4. Revue de code sécurité complétée
5. Validation de la conformité réglementaire (DSP2, PCI-DSS)
6. Tests d'authentification et de gestion des sessions
7. Vérification des mécanismes cryptographiques
8. Tests des contrôles anti-fraude
9. Validation des communications sécurisées
10. Tests d'intégration avec les systèmes de paiement tiers

### 12.2 Références réglementaires

- DSP2 (Directive sur les Services de Paiement 2)
- Règlement délégué (UE) 2018/389 sur l'authentification forte (SCA)
- PCI DSS (Payment Card Industry Data Security Standard)
- RGPD (Règlement Général sur la Protection des Données)
- UK GDPR et réglementations britanniques post-Brexit
- Exigences de la FCA (Financial Conduct Authority) pour le Royaume-Uni
- Exigences de l'ACPR pour la France

### 12.3 Standards et bonnes pratiques

- OWASP Mobile Application Security Verification Standard (MASVS)
- OWASP Mobile Top 10
- NIST Guidelines for Managing the Security of Mobile Devices
- Recommandations de l'ANSSI sur la sécurité mobile
- Guides de sécurité des plateformes (Apple, Google)
- Bonnes pratiques EMVCo pour les paiements mobiles

### 12.4 Glossaire

- **API (Application Programming Interface)** : Interface permettant à deux applications de communiquer entre elles
- **Certificate Pinning** : Technique liant une application à un certificat spécifique pour prévenir les attaques MitM
- **DSP2/PSD2** : Directive sur les Services de Paiement 2, cadre réglementaire européen pour les paiements
- **EMV** : Standard international pour les cartes à puce et les terminaux de paiement
- **Jailbreak/Root** : Processus permettant d'obtenir des privilèges élevés sur un appareil, contournant les restrictions du fabricant
- **MitM (Man in the Middle)** : Attaque où l'attaquant s'interpose dans une communication entre deux parties
- **NFC (Near Field Communication)** : Technologie de communication sans fil à courte portée utilisée pour les paiements sans contact
- **OWASP** : Open Web Application Security Project, communauté travaillant sur la sécurité des applications
- **PAN (Primary Account Number)** : Numéro de carte bancaire à 16 chiffres
- **SCA (Strong Customer Authentication)** : Authentification forte du client, exigée par DSP2
- **Tokenisation** : Processus de remplacement des données sensibles par des équivalents non sensibles (tokens)

---

**Document approuvé par :** [Signature RSSI]
**Date d'approbation :** 21 avril 2025
**Prochaine révision prévue :** 21 avril 2026