Normes de sécurité des API essentielles

Neo Financia

RÉFÉRENCE: API-SEC-STD-2025-V1.0
CLASSIFICATION: INTERNE
VERSION: 1.0
DATE D'APPROBATION: 21 avril 2025
APPROBATION: Comité de Sécurité (COSEC)
PROPRIÉTAIRE: RSSI
PÉRIODICITÉ DE RÉVISION: Semestrielle

Table des matières

---

1. Introduction et portée

1.1 Objectif

Le présent document définit les normes de sécurité obligatoires pour toutes les Interfaces de Programmation d'Applications (API) développées, déployées et exposées par Neo Financia. Ces normes visent à établir un cadre cohérent et robuste pour garantir la sécurité des services financiers interconnectés, tout en permettant l'innovation et l'agilité nécessaires au développement de l'entreprise.

1.2 Contexte

Neo Financia, en tant que néobanque européenne servant 2 millions de clients, dépend fortement de ses API pour:

- Fournir des services bancaires numériques à ses clients
- S'intégrer avec des partenaires fintech (Mangopay, Lemonway)
- Respecter les obligations réglementaires (Open Banking)
- Faciliter l'interconnexion entre ses propres systèmes

Avec un volume quotidien d'environ 150 000 appels API et un niveau de maturité actuel estimé à 3/10, l'amélioration de la sécurité des API constitue une priorité stratégique.

1.3 Champ d'application

Ces normes s'appliquent à:

- **Toutes les API** développées, opérées ou exposées par Neo Financia
- **Tous les environnements** (développement, test, production)
- **Toutes les infrastructures** supportant ces API (Azure 70%, OVHcloud 20%, AWS 10%)
- **Tous les acteurs** impliqués dans le cycle de vie des API (développeurs, architectes, opérateurs, partenaires)

1.4 Classification des API

Neo Financia catégorise ses API selon trois niveaux de criticité:

Catégorie	Description	Exemples	Niveau de protection
API Critiques	API exposant des fonctionnalités financières sensibles ou des données clients	APIs de paiement, APIs d'authentification, APIs de compte	Protection maximale
API Importantes	API fournissant des services nécessaires mais non critiques	APIs de gestion de profil, APIs de reporting	Protection élevée
API Standard	API à usage interne avec impact limité	APIs de logs, APIs de monitoring	Protection standard

1.5 Cadre réglementaire

Les APIs de Neo Financia doivent respecter les exigences réglementaires suivantes:

- **DSP2/PSD2**: Directive sur les Services de Paiement
- **RGPD/GDPR**: Règlement Général sur la Protection des Données
- **DORA**: Digital Operational Resilience Act
- **NIS2**: Directive Network and Information Security
- **RTS EBA**: Standards Techniques Réglementaires de l'Autorité Bancaire Européenne

2. Gouvernance des API

2.1 Organisation et responsabilités

2.1.1 Comité API

La gouvernance des API est assurée par le Comité API qui se réunit mensuellement. Ce comité comprend:

- Le Responsable API (président)
- Le Responsable Sécurité des API
- Les représentants des équipes de développement
- Un représentant de la DSI
- Un représentant du RSSI
- Un représentant des métiers

2.1.2 Matrice RACI

Activité	Business Owner	Architecte API	Développeur	Équipe Sécurité	Ops

Définition des besoins API	R/A	C	I	C	I
Conception de l'API	C	R/A	C	C	I
Implémentation des contrôles de sécurité	I	C	R	A	C
Validation sécurité	I	C	C	R/A	I
Déploiement	I	I	C	C	R/A
Surveillance	I	I	C	C	R/A

2.2 Cycle de vie des API

2.2.1 Processus de développement sécurisé

Le développement des API suit un processus sécurisé intégré au pipeline DevSecOps:

1. **Définition**: Spécification des besoins et documentation initiale (OpenAPI/Swagger)
2. **Conception**: Revue d'architecture et modélisation des menaces
3. **Développement**: Implémentation avec contrôles de sécurité intégrés
4. **Tests**: Tests de sécurité automatisés et manuels
5. **Déploiement**: Validation de conformité et déploiement contrôlé
6. **Opération**: Surveillance continue et gestion des vulnérabilités
7. **Décommissionnement**: Obsolescence planifiée et retrait sécurisé

2.2.2 Contrôles de sécurité par phase

Phase	Contrôles obligatoires
Définition	Analyse préliminaire de risque, classification de l'API
Conception	Modélisation des menaces (STRIDE), validation de l'architecture
Développement	Revues de code, SAST, contrôles d'entrée/sortie
Tests	DAST, tests de fuzzing, tests de pénétration
Déploiement	Scan de vulnérabilités, validation de configuration
Opération	Surveillance, WAF, détection d'anomalies
Décommissionnement	Plan de migration, notification préalable, révocation des credentials

2.3 Documentation et registre des API

2.3.1 Exigences de documentation

Chaque API doit être documentée selon les exigences suivantes:

- Spécification OpenAPI/Swagger obligatoire
- Documentation des contrôles de sécurité implémentés
- Informations sur le modèle d'authentification et d'autorisation
- Guide d'implémentation pour les consommateurs

- Catalogue des erreurs et codes de retour

2.3.2 Registre central des API

Neo Financia maintient un registre central des API contenant:

- Inventaire de toutes les API internes et externes
- Classification de sécurité et niveau de criticité
- Propriétaire technique et métier
- État du cycle de vie
- Résultats des derniers tests de sécurité
- Métriques d'utilisation et de performance

3. Architecture de sécurité

3.1 Principes d'architecture

L'architecture de sécurité des API de Neo Financia repose sur les principes fondamentaux suivants:

- **Défense en profondeur**: Multiples couches de protection complémentaires
- **Moindre privilège**: Limitation des droits d'accès au strict nécessaire
- **Sécurité par conception**: Intégration de la sécurité dès la conception
- **Zero Trust**: Validation continue de chaque requête, quel que soit son origine
- **Fail Secure**: En cas d'erreur, blocage par défaut des opérations

3.2 Architecture de référence

Toutes les API exposées doivent suivre l'architecture de référence à trois niveaux:

1. **Niveau Exposition**:
    - API Gateway centralisée
    - Terminaison TLS
    - Validation basique des requêtes
    - Authentification
    - Rate limiting
2. **Niveau Service**:
    - Logique métier
    - Autorisation fine
    - Validation approfondie
    - Transformation des données
3. **Niveau Données**:
    - Accès contrôlé aux données
    - Filtrage des données sensibles
    - Journalisation des accès

3.3 Zones de sécurité

Les API sont déployées dans des zones de sécurité distinctes selon leur classification:

Zone	Classification API	Contrôles spécifiques

Zone DMZ	APIs publiques	WAF, authentification renforcée, inspection approfondie
Zone Partenaires	APIs partenaires	Connexions dédiées, filtrage IP, authentification mutuelle
Zone Applicative	APIs internes	Segmentation, contrôles d'accès stricts
Zone Données	Aucune API directe	Accès indirect uniquement via les zones supérieures

3.4 API Gateway

3.4.1 Fonctionnalités de sécurité obligatoires

L'API Gateway doit implémenter les fonctionnalités de sécurité suivantes:

- Terminaison TLS (minimum TLS 1.2, recommandé TLS 1.3)
- Validation des schémas de requêtes (OpenAPI/Swagger)
- Application des politiques d'authentification
- Rate limiting et quotas
- Détection des comportements anormaux
- Journalisation complète
- Filtrage d'adresses IP/géolocalisation

3.4.2 Configuration de base sécurisée

- Désactivation des méthodes HTTP non utilisées
- En-têtes de sécurité HTTP (HSTS, X-Content-Type-Options, etc.)
- Suppression des informations de débogage et d'erreur détaillées
- Désactivation de CORS par défaut, configuration explicite si nécessaire
- Limitation de la taille des requêtes et des réponses

4. Authentification et autorisation

4.1 Modèle d'authentification

4.1.1 Méthodes d'authentification par type d'API

Type d'API	Méthode primaire	Méthode secondaire
APIs publiques	OAuth 2.0 avec OpenID Connect	JWT signé avec limitation de durée
APIs partenaires	mTLS + OAuth 2.0	Certificats X.509
APIs internes	OAuth 2.0	JWT ou HMAC

4.1.2 Configuration OAuth 2.0

- Utilisation obligatoire du flux Authorization Code avec PKCE pour les applications clientes
- Durée de validité des tokens d'accès: maximum 15 minutes
- Durée de validité des tokens de rafraîchissement: maximum 24 heures
- Rotation obligatoire des secrets clients tous les 90 jours
- Stockage sécurisé des secrets (coffre-fort, HSM)

4.1.3 Authentification mutuelle TLS (mTLS)

Pour les APIs partenaires et critiques:

- Validation complète de la chaîne de certificats
- Utilisation d'une PKI dédiée ou autorité de certification reconnue
- Révocation immédiate possible des certificats compromis
- Durée de validité des certificats clients: maximum 12 mois
- Algorithmes minimums: RSA 2048 bits ou ECC P-256

4.2 Gestion des autorisations

4.2.1 Modèle d'autorisation

Neo Financia utilise un modèle d'autorisation basé sur:

- Les rôles (RBAC) pour les accès internes
- Les scopes OAuth 2.0 pour les API externes
- Les attributs (ABAC) pour les décisions complexes

4.2.2 Principes d'implémentation

- Autorisation systématique à chaque appel API
- Validation côté serveur obligatoire (jamais basée uniquement sur le token)
- Séparation claire authentication/authorization
- Granularité fine des permissions
- Contrôle temporel (restriction horaire si applicable)
- Évaluation contextuelle (appareil, localisation, comportement)

4.2.3 Règles de conception des scopes

Les scopes OAuth doivent suivre les conventions suivantes:

- Format: `ressource:action`
- Granularité adaptée (ni trop large, ni trop fine)
- Documentation explicite de chaque scope
- Attribution restrictive selon le principe du moindre privilège
- Révision régulière des scopes attribués

4.3 Gestion des sessions et tokens

4.3.1 Sécurisation des tokens

- Signature obligatoire (de préférence avec algorithmes asymétriques)
- Protection contre la réutilisation (nonce, jti)
- Validation de tous les claims (iss, aud, exp, nbf, etc.)
- Stockage sécurisé côté client
- Révocation possible via serveur d'autorisation

4.3.2 Rotation et renouvellement

- Rotation des clés de signature: tous les 6 mois minimum
- Renouvellement transparent des tokens expirés
- Mécanisme de révocation d'urgence

5. Protection des données

5.1 Classification des données API

Niveau	Description	Exemples	Mesures spécifiques

P3 - Critique	Données hautement sensibles	Authentification, autorisation de paiement	Chiffrement bout en bout, tokenisation
P2 - Confidentiel	Données client sensibles	Soldes, transactions	Chiffrement, masquage partiel
P1 - Interne	Données d'usage interne	Configurations, logs	Chiffrement en transit
P0 - Public	Données sans restrictions	Documentation publique	Intégrité

5.2 Chiffrement et protection

5.2.1 Exigences de chiffrement

- **En transit**: TLS 1.2+ obligatoire avec suites cryptographiques fortes
- **Requêtes sensibles**: Chiffrement additionnel au niveau message (JWE)
- **Réponses sensibles**: Chiffrement sélectif des données critiques
- **Clés**: Gestion via service dédié (Key Vault, HSM)

5.2.2 Algorithmes autorisés

Usage	Algorithmes autorisés	Taille minimale
Chiffrement symétrique	AES-GCM, ChaCha20-Poly1305	256 bits
Chiffrement asymétrique	RSA-OAEP, ECDH	RSA 2048, ECC P-256
Signature	RSA-PSS, ECDSA, Ed25519	RSA 2048, ECC P-256
Hachage	SHA-256, SHA-384, SHA-512	N/A

5.3 Validation des entrées et sorties

5.3.1 Validation des requêtes

- Validation complète de toutes les entrées utilisateur
- Utilisation des schémas OpenAPI/JSON Schema
- Sanitization des données avant traitement
- Vérification des types, formats et plages de valeurs
- Protection contre les injections (SQL, NoSQL, OS command, etc.)

5.3.2 Filtrage des réponses

- Principe de divulgation minimale des informations
- Filtrage basé sur les permissions de l'utilisateur
- Suppression des données sensibles non requises
- Masquage des informations partielles si nécessaire (PAN, etc.)
- Headers de sécurité systématiques

5.4 Protection contre les fuites de données

5.4.1 Contrôles préventifs

- Limitation du nombre d'objets retournés (pagination obligatoire)
- Détection des extractions massives de données
- Agrégation des données sensibles quand possible

- Journalisation des accès aux données sensibles

5.4.2 Anonymisation et pseudonymisation

Pour les API exposant des données à caractère personnel:

- Pseudonymisation des identifiants directs
- Anonymisation des jeux de données pour API d'analyse
- Tokenisation des identifiants sensibles (PAN, IBAN)
- Mécanismes d'agrégation pour les données statistiques

6. Contrôles opérationnels

6.1 Rate limiting et quotas

6.1.1 Politique de limitation

Toutes les API doivent implémenter:

- Rate limiting par client/application
- Rate limiting par utilisateur final
- Rate limiting par ressource
- Quotas journaliers et mensuels

6.1.2 Valeurs de référence

Type d'API	Limitation par IP	Limitation par client	Limitation par utilisateur
APIs publiques	120/min	600/min	300/min
APIs partenaires	600/min	3000/min	1200/min
APIs internes	1200/min	6000/min	3000/min

Ces valeurs sont des références et doivent être ajustées selon les besoins spécifiques de chaque API.

6.1.3 Réponse aux dépassements

- Code HTTP 429 (Too Many Requests)
- En-tête Retry-After
- Documentation claire des limites
- Alertes automatiques en cas de dépassements répétés

6.2 Surveillance et détection

6.2.1 Journalisation

Exigences minimales de journalisation pour chaque appel API:

- Identifiant unique de la requête
- Horodatage précis
- Identité de l'appelant (client et utilisateur)
- Méthode HTTP et URI (sans paramètres sensibles)
- Code de statut de la réponse
- Temps de réponse
- Adresse IP source

- User-Agent

Les journaux doivent être:

- Centralisés dans le SIEM
- Protégés contre toute modification
- Conservés selon la politique de rétention (minimum 12 mois)

6.2.2 Détection des anomalies

Mise en place de mécanismes de détection pour:

- Patterns d'usage anormaux
- Tentatives d'attaques connues (injection, XSS, etc.)
- Comportements suspects (scan, fuzzing)
- Accès depuis des localisations inhabituelles
- Volumes de requêtes inhabituels

6.2.3 Alertes et réponse

Configuration des alertes pour:

- Violations de politique de sécurité
- Attaques détectées
- Dépassements significatifs de quotas
- Erreurs d'authentification répétées
- Accès à des ressources sensibles

6.3 Tests continus

6.3.1 Types de tests obligatoires

Type de test	Fréquence	Environnements
SAST (Static Application Security Testing)	À chaque build	Développement
DAST (Dynamic Application Security Testing)	Hebdomadaire	Pré-production
Tests de fuzzing API	Mensuel	Pré-production
Scan de vulnérabilités	Quotidien	Production
Tests de pénétration	Trimestriel	Production

6.3.2 Intégration CI/CD

Les pipelines CI/CD doivent intégrer:

- Validation automatique des standards de sécurité
- Security gates aux étapes clés
- Blocage du déploiement en cas d'échec des tests critiques
- Notification des vulnérabilités détectées
- Documentation des exceptions (avec justification et mesures compensatoires)

6.4 Gestion des versions et obsolescence

6.4.1 Stratégie de versionnement

- Versionnement sémantique (Major.Minor.Patch)
- Compatibilité descendante pour les modifications mineures
- Documentation claire des changements (changelog)

- Période de transition pour les changements majeurs (minimum 6 mois)

6.4.2 Processus de dépréciation

- Notification proactive aux consommateurs (minimum 3 mois à l'avance)
- Documentation des alternatives
- Période de coexistence des versions
- Monitoring dédié des versions dépréciées
- Plan de migration pour les clients

7. Sécurité des API partenaires

7.1 Processus d'onboarding partenaire

7.1.1 Évaluation préalable

Avant toute intégration, les partenaires API doivent être soumis à:

- Évaluation de risque documentée
- Due diligence sécurité
- Vérification de conformité réglementaire
- Test d'intégration en environnement de validation

7.1.2 Procédure d'accès

- Processus formel de demande d'accès
- Validation multipartite (métier, sécurité, juridique)
- Attribution restrictive des droits d'accès
- Documentation des flux de données
- Signature d'accords de confidentialité et de sécurité

7.2 Contrôles spécifiques pour les partenaires financiers

7.2.1 Exigences pour Mangopay et Lemonway

- Tunnels dédiés sécurisés
- Authentification mutuelle TLS obligatoire
- Journalisation renforcée des transactions
- Validation des montants et limites
- Monitoring spécifique des transactions
- Alertes sur transactions inhabituelles

7.2.2 Chiffrement des données sensibles

- Double chiffrement des données de paiement
- Tokenisation des références client
- Séparation des clés de chiffrement
- Rotation des clés tous les 3 mois

7.3 Surveillance dédiée

7.3.1 Monitoring spécifique

- Tableau de bord dédié par partenaire
- Alertes personnalisées selon profil d'usage
- Détection des anomalies comportementales
- Rapports réguliers d'activité et de conformité

7.3.2 Revue périodique

- Audit des accès tous les 3 mois

- Revue des incidents et anomalies
- Actualisation de l'évaluation des risques
- Test de pénétration annuel des interfaces partenaires

8. Conformité Open Banking

8.1 Exigences DSP2/PSD2

8.1.1 Mesures de sécurité obligatoires

- Authentification forte du client (SCA) pour toutes les opérations sensibles
- Gestion des exemptions à la SCA selon le référentiel RTS
- APIs dédiées conformes aux standards du marché
- Mécanisme de secours (fallback) documenté
- Monitoring de la disponibilité (objectif: >99.98%)

8.1.2 Interfaces dédiées TPP

- Enregistrement sécurisé des TPP
- Vérification systématique des licences auprès des autorités nationales
- Limitation des accès selon les autorisations déclarées
- Surveillance spécifique des activités TPP
- Reporting aux autorités réglementaires

8.2 Standards d'implémentation

8.2.1 Conformité aux standards de marché

Les APIs Open Banking doivent être conformes aux standards:

- STET pour les opérations en France
- Berlin Group NextGenPSD2 pour l'interopérabilité européenne
- Open Banking UK pour les opérations britanniques

8.2.2 Sécurité spécifique

- eIDAS QWAC pour l'identification des TPP
- Validation des certificats en temps réel
- Vérification des rôles dans les certificats
- Conservation des preuves d'accès (minimum 18 mois)

8.3 Gestion du consentement

8.3.1 Recueil du consentement

- Interface explicite et claire pour les clients
- Granularité fine des consentements
- Limitation temporelle obligatoire
- Documentation des finalités d'usage
- Possibilité de révocation immédiate

8.3.2 Suivi et renouvellement

- Traçabilité complète des consentements actifs
- Notification avant expiration
- Processus de renouvellement explicite
- Tableau de bord client pour la gestion des consentements
- Historique des accès réalisés sur la base du consentement

9. Gestion des incidents

9.1 Détection et qualification des incidents API

9.1.1 Types d'incidents spécifiques

- Violations d'authentification ou d'autorisation
- Exploitation de vulnérabilités API
- Fuites de données via API
- Attaques par déni de service
- Utilisation abusive des API
- Défaillances techniques critiques

9.1.2 Critères de gravité

Niveau	Description	Exemples	Temps de réponse
P1 - Critique	Impact majeur sur les opérations	Fuite de données, compromission d'authentification	15 minutes
P2 - Majeur	Impact significatif	Indisponibilité d'une API critique	1 heure
P3 - Modéré	Impact limité	Dégradation de performance	4 heures
P4 - Mineur	Impact minimal	Erreurs isolées	24 heures

9.2 Procédures de réponse

9.2.1 Procédure d'escalade

1. Détection de l'incident (automatique ou signalement)
2. Qualification initiale par l'équipe de surveillance
3. Notification aux responsables selon la gravité
4. Formation de l'équipe de réponse adaptée
5. Containment et investigation
6. Communication aux parties prenantes
7. Résolution et rétablissement
8. Post-mortem et documentation

9.2.2 Actions immédiates

Selon le type d'incident:

- Blocage d'accès temporaire
- Révocation des credentials compromis
- Limitation de trafic
- Activation des contrôles d'urgence
- Mise en place de règles de filtrage temporaires

9.3 Communication et notification

9.3.1 Communication interne

- Matrice de notification selon gravité
- Canaux dédiés pour la gestion de crise
- Points de situation réguliers
- Coordination entre équipes (sécurité, IT, métiers, communication)

9.3.2 Communication externe

- Notification aux clients impactés
- Information des partenaires concernés
- Déclaration aux autorités réglementaires si applicable
- Communication publique si nécessaire (coordonnée avec Direction Communication)

9.4 Analyse post-incident

9.4.1 Revue post-incident

Pour chaque incident significatif (P1, P2):

- Analyse des causes racines
- Évaluation de l'efficacité de la réponse
- Identification des axes d'amélioration
- Mise à jour de la cartographie des risques
- Documentation complète de l'incident

9.4.2 Actions correctrices

- Plan d'action formalisé
- Mesures pour prévenir la récurrence
- Renforcement des contrôles défaillants
- Mise à jour des procédures si nécessaire
- Test de validation des corrections

10. Annexes techniques

10.1 Configurations de référence

10.1.1 API Gateway

```
# Configuration sécurisée de l'API Gateway
security:
  # Configuration TLS
  tls:
    minimum_version: "TLS1.2"
    preferred_ciphers:
      - "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384"
      - "TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256"
    certificate_validation: "full"

  # Headers de sécurité
  headers:
    X-Content-Type-Options: "nosniff"
    Strict-Transport-Security: "max-age=31536000; includeSubDomains"
    Content-Security-Policy: "default-src 'self'"
    X-Frame-Options: "DENY"

  # Rate limiting
  rate_limiting:
    default:
      requests_per_second: 10
      burst: 20
    paths:
```

```
      "/api/v1/payments":
        requests_per_second: 5
        burst: 10

  # Validation des requêtes
  request_validation:
    enforce_content_type: true
    max_body_size: "10KB"
    schema_validation: true
```

10.1.2 OAuth 2.0

```
{
  "issuer": "https://auth.neofinancia.eu",
  "authorization_endpoint": "https://auth.neofinancia.eu/authorize",
  "token_endpoint": "https://auth.neofinancia.eu/token",
  "jwks_uri": "https://auth.neofinancia.eu/.well-known/jwks.json",
  "response_types_supported": ["code"],
  "grant_types_supported": ["authorization_code", "refresh_token"],
  "subject_types_supported": ["public"],
  "id_token_signing_alg_values_supported": ["RS256", "ES256"],
  "scopes_supported": ["openid", "profile", "email", "accounts:read",
"payments:write"],
  "token_endpoint_auth_methods_supported": ["client_secret_post", "private_key_jwt"],
  "claims_supported": ["sub", "iss", "auth_time", "acr", "name", "given_name",
"family_name", "email"],
  "code_challenge_methods_supported": ["S256"]
}
```

10.2 Exemples de modèle de menaces API

10.2.1 STRIDE appliqué aux API

Menace	Description	Exemples dans le contexte API	Contrôles
Spoofing	Usurpation d'identité	Utilisation de credentials volés, session hijacking	OAuth 2.0, JWT signé, mTLS
Tampering	Altération des données	Modification des requêtes/réponses, injection	Signature des messages, validation des entrées
Repudiation	Déni d'actions effectuées	Contestation d'une transaction initiée	Journalisation sécurisée, signatures non-répudiables
Information Disclosure	Divulgation d'informations	Exposition excessive de données, IDOR	Filtrage des réponses, contrôle d'accès
Denial of Service	Déni de service	Saturation des API, ressources épuisées	Rate limiting, quotas, timeout
Elevation	Élévation de	Contournement des	Validation stricte des

of Privilege	privilèges	contrôles d'accès	droits, principe du moindre privilège

10.3 Procédures d'audit de sécurité API

10.3.1 Checklist d'audit

- ☐ Validation de la spécification OpenAPI/Swagger
- ☐ Vérification des mécanismes d'authentification
- ☐ Vérification des contrôles d'autorisation
- ☐ Test de séparation multi-tenant
- ☐ Validation des contrôles d'entrée/sortie
- ☐ Vérification du chiffrement en transit
- ☐ Test des mécanismes de rate limiting
- ☐ Vérification de la journalisation
- ☐ Test de robustesse (fuzzing)
- ☐ Vérification de la gestion des erreurs
- ☐ Test des en-têtes de sécurité
- ☐ Vérification de la configuration TLS

10.4 Ressources et références

10.4.1 Standards de sécurité API

- OWASP API Security Top 10
- NIST SP 800-204 (Security Strategies for Microservices)
- PCI DSS API Guidelines
- OpenAPI Security Guidelines

10.4.2 Outils recommandés

- **Conception**: Swagger Editor, Postman
- **Tests**: OWASP ZAP API Scan, Burp Suite Professional
- **Surveillance**: API Security Gateway, ELK Stack
- **Documentation**: Swagger UI, ReDoc

Document approuvé par le RSSI le 21 avril 2025

Fin du document