DOSSIER DE CONTRÔLES

# Sécurité & robustesse

## KORA INO

Analyse dynamique, revue du code, résistance applicative, remédiations et chaîne de qualité.

DESTINATAIRE

INOVALLÉE

Référentiel de contrôles

Audits du 12.11 · Code et CI du 13.11

01 / LECTURE EXÉCUTIVE

# Ce que nous avons vérifié

Le processus associe la recherche de vulnérabilités en fonctionnement, l'examen automatisé du code, des essais de résistance et des contrôles de non-régression.

| 0 | 853 | 56 |
| --- | --- | --- |
| alerte de niveau High relevée dans les scans DAST disponibles | tests automatisés réussis sur la dernière CI consultée | fichiers de tests validés dans cette même exécution |

## UN CONTRÔLE EN PLUSIEURS TEMPS

12 NOV.

### SCAN DYNAMIQUE

Inspection du comportement HTTP et des surfaces accessibles.

12 NOV.

### REVUE STATIQUE

Recherche de défauts dans le code et les dépendances.

12 NOV.

### RETEST

Contre-vérification après une première série de modifications.

13 NOV.

### QUALITÉ CONTINUE

Exécution verte des vérifications de code, tests et compilation.

Lecture du résultat : les faiblesses repérées ont alimenté des corrections. Les preuves issues du dépôt et de la CI sont distinguées des vérifications devant être effectuées sur le site effectivement publié.

02 / PÉRIMÈTRE

# La surface examinée

Le scan dynamique a recensé des routes et services de l'environnement de développement ; il ne constitue pas, à lui seul, une couverture exhaustive de tous les scénarios métier.

## CARTOGRAPHIE DES CIBLES EXAMINÉES

356 points d'entrée recensés

**Application principale**

Frontend React / TanStack Start

339 points d'entrée recensés

**Service API / proxy interne**

Inspection initiale seulement

3 points d'entrée recensés

**Service Supabase local**

Inspection initiale seulement

1 point d'entrée recensé

**Service périphérique**

Authentification Basic observée

## MÉTHODES DE VÉRIFICATION

| | |
|---|---|
| DAST | OWASP ZAP 2.17.0 : exploration, inspection passive et tests actifs encadrés. |
| SAST | Semgrep OSS 1.185.0 : règles automatiques appliquées au dépôt applicatif. |
| RETEST | Comparaison des constats avant/après une première tentative de correction. |
| CI | TypeScript, ESLint, Vitest et compilation du projet sur GitHub Actions. |

03 / CONTRÔLE DYNAMIQUE

# Le comportement exposé

L'analyse dynamique cherche ce qu'un tiers pourrait observer ou solliciter depuis le réseau, et vérifie la réponse effective de l'application.

## FAMILLES DE CONTRÔLES RÉALISÉES

01 **En-têtes et navigateur**
CSP, anti-clickjacking, type MIME / nosniff, intégrité des ressources chargées.

02 **Transport et authentification**
Disponibilité HTTPS sur la cible, authentification Basic sur un service périphérique.

03 **Paramètres et jetons**
Inspection des valeurs transportées dans les URL et identification de leur rôle.

04 **Exposition de code**
Commentaires et métadonnées observables depuis un serveur de développement.

05 **Fuzzing et chemins**
Requêtes inattendues, paramètres atypiques, hypothèse de traversée de chemin.

06 **Erreurs de service**
Codes HTTP, temps de réponse, capacité du serveur à terminer le scan.

Résultat historique : les scans ZAP du 12 novembre n'ont remonté aucune alerte classée « High ». Plusieurs points de durcissement ont néanmoins été identifiés, puis orientés vers la remédiation. Le compte rendu détaillé est conservé dans le dossier technique d'audit.

04 / INSPECTION STATIQUE

# La sécurité du code

La revue statique recherche des défauts dans les mécanismes applicatifs sans dépendre de leur manifestation visible dans le navigateur.

## CONSTATS ET VÉRIFICATIONS CIBLÉS

### CHIFFREMENT DES CONNEXIONS

AES-GCM : longueur du tag d'authentification et refus des données tronquées.

Correction vérifiée dans le code actuel : longueur de 16 octets imposée au chiffrement comme au déchiffrement.

### INTÉGRITÉ DES DÉPENDANCES

Délai de carence appliqué aux versions nouvellement publiées.

Paramètre de 7 jours présent dans bunfig.toml ; l'outillage CI utilise encore Bun 1.2.19.

### ENTRÉES ET EXPRESSIONS RÉGULIÈRES

Recherche de constructions de regex susceptibles d'engendrer un déni de service.

Deux alertes mineures à investiguer avaient été relevées au retest Semgrep.

### JOURNAUX ET QUALITÉ DE PARSING

Chaînes de journalisation, anomalies de parsing, analyse de flux incomplète.

Cinq signalements informatifs et des limites d'analyse ont fait l'objet du suivi technique.

05 / RÉSILIENCE

# La résistance en situation

Un test de sécurité actif peut provoquer des requêtes inhabituelles et soutenues. Ce scénario a permis d'identifier un problème de capacité mémoire sur l'environnement de développement.

## CHRONOLOGIE DES ESSAIS DU 12 NOVEMBRE

09:53

### Premier passage

Le fuzzing actif entraîne l'épuisement de la mémoire JavaScript du serveur de développement Vite. Le conteneur s'arrête après environ 4 min 29 s.

APRÈS DIAGNOSTIC

### Correction infrastructure

L'allocation mémoire du conteneur est ajustée ; la campagne suivante peut être menée à son terme.

14:25

### Scan complet

La campagne produit un ensemble d'observations exploitables, sans nouvelle interruption du service pendant le scan.

17:14

### Retest

Une seconde lecture du comportement applicatif mesure l'effet de la première série de changements.

## OBSERVATIONS DU SCAN COMPLET

3 % de réponses HTTP 5xx selon les statistiques du scan ; des réponses lentes sont également observées sous sollicitation. Ces données décrivent la campagne ZAP : elles ne remplacent pas un benchmark de charge ni une mesure de disponibilité en production.

06 / DÉFENSE APPLICATIVE

# Protections intégrées

Le dépôt et sa documentation décrivent les garde-fous côté serveur. Ce panorama explique les mécanismes en place, distincts des findings automatiques ZAP et Semgrep.

### IDENTITÉ & SESSIONS

Vérification des jetons Supabase Auth sur les routes concernées ; comptes autorisés et invitations.

### DROITS & RÔLES

Contrôle des autorisations au niveau serveur avant les actions de modification.

### ANTI-CSRF

En-tête de requête exigé pour les écritures ; cookies de téléchargement restreints.

### VALIDATION DES DONNÉES

Schémas d'entrée, bornes sur les tailles de fichiers et sur les requêtes.

### APPELS VERS LE WEB

Filtrage SSRF, validation des URL et redirections ; maîtrise des contenus non fiables.

### TRAÇABILITÉ & CHIFFREMENT

Journal des actions métier ; secrets côté serveur et chiffrement des clés de connexion.

Point de méthode : ces dispositifs sont décrits par le code et la documentation du projet. Leur présence ne signifie pas que chaque scénario d'attaque a fait l'objet d'un test d'intrusion manuel dédié.

07 / STATUT VÉRIFIABLE

# Correctifs & état de preuve

Les vérifications ci-dessous portent sur la branche principale consultée le 14 novembre et sur les rapports d'audit datés du 12 novembre.

ÉTAT VISIBLE DANS LES ÉLÉMENTS CONSULTÉS

### CHIFFREMENT AES-GCM

Correction présente dans le code : tag 16 octets imposé.

VÉRIFIÉ DANS LE CODE

### X-CONTENT-TYPE-OPTIONS

Ajout de nosniff dans les réponses serveur et pour les fichiers statiques.

VÉRIFIÉ DANS LE CODE

### POLITIQUE CSP

Présente en Report-Only ; le blocage effectif n'est pas encore activé sur main.

PARTIEL / OBSERVATION

### DEPENDANCES BUN

Délai de 7 jours configuré ; la CI épingle toujours Bun 1.2.19.

CONFIGURÉ / NON OPÉRANT

### HTTPS ET HÉBERGEMENT FINAL

Le scan historique visait un environnement de développement HTTP.

RETEST RUNTIME À DOCUMENTER

### SERVICES PÉRIPHÉRIQUES

Le dernier retest DAST ne couvrait pas les services annexes.

RETEST À DOCUMENTER

08 / ASSURANCE QUALITÉ

# Des vérifications reproductibles

La dernière exécution GitHub Actions accessible sur la branche principale est entièrement réussie. Elle fournit une preuve datée d'une batterie de contrôles automatisés.

EXÉCUTION DU 13 NOVEMBRE 2026

853

**TESTS RÉUSSIS**

56 fichiers de tests validés. Résultat issu des journaux de la CI exécutée sur le commit cd67d9b.

## CHAÎNE DE VÉRIFICATION

| | | |
|---|---|---|
| 01 | **Installation figée** | Dépendances installées avec verrouillage du lockfile. |
| 02 | **Contrôle TypeScript** | Absence d'erreurs de typage détectées par la commande CI. |
| 03 | **Analyse ESLint** | Conformité des règles de qualité de code configurées. |
| 04 | **Suite Vitest** | 853 tests verts répartis dans 56 fichiers. |
| 05 | **Compilation** | Build de production généré par la chaîne CI. |
| 06 | **Intégrité du diff** | Vérification git diff --check réussie. |

Référence : Exécution GitHub Actions 43464735069. Ce contrôle valide le commit ; la mise en ligne relève d'une validation d'environnement distincte.

09 / REMISE ET TRAÇABILITÉ

# Une démarche documentée

La valeur du contrôle réside dans sa traçabilité : une observation datée, un correctif identifié, puis une preuve de fonctionnement sur une version déterminée.

## CHAÎNE DE PREUVE CONSERVÉE

01

### AUDIT DYNAMIQUE

Rapports OWASP ZAP du 12 novembre 2026, scan complet et retest.

02

### AUDIT STATIQUE

Rapport Semgrep OSS initial et retest, avec constats et limites d'analyse.

03

### SUIVI DES CORRECTIFS

Rapport KOREV Evidence consolidé du 12 novembre et lecture du code actualisé.

04

### QUALITÉ LOGICIELLE

CI GitHub Actions du 13 novembre 2026 sur le commit cd67d9b.

## VALIDATION SUR L'ENVIRONNEMENT LIVRÉ

Pour rattacher définitivement les constats au service remis à Inovallée, la version déployée doit être identifiée, puis contrôlée sur son hébergement effectif : transport HTTPS, en-têtes HTTP, accès aux services périphériques et contre-scan des règles de sécurité concernées.

**CONTRÔLER. CORRIGER. REVÉRIFIER.**

KOREV AI · Application KORA INO · Dossier de contrôles de sécurité et de robustesse