



Rapport Exécutif de Sécurité

Audit de Sécurité Automatisé

acme/web-platform

Branche: main · Date: 2026-02-10



42

TOTAL

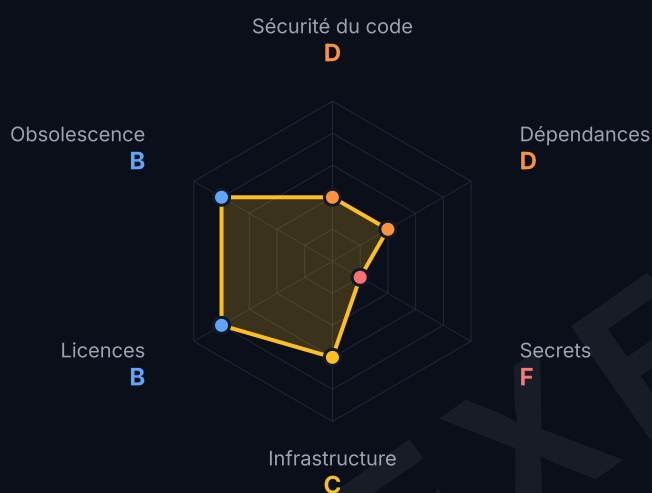
5

CRITIQUE

13

ÉLEVÉ

SCORE PAR CATÉGORIE



RÉSULTATS PAR SÉVÉRITÉ



Résumé de Santé Sécurité

Le projet acme/web-platform présente des risques de sécurité significatifs nécessitant une attention immédiate. Notre analyse automatisée a identifié 5 vulnérabilités critiques et 13 de sévérité élevée dans le codebase, incluant des vecteurs exploitables d'injection SQL et de cross-site scripting dans les endpoints API en production. La combinaison la plus alarmante est celle d'un traitement d'entrées utilisateur injectable et de secrets exposés — dont une clé secrète Stripe live committée dans le dépôt — qui pourrait permettre à un attaquant d'exfiltrer des données clients sensibles, d'exécuter des requêtes arbitraires sur la base de données et de compromettre le traitement des paiements. La chaîne de dépendances de l'application inclut 6 packages avec des CVE connues, dont deux permettant l'exécution de code à distance. Bien que la configuration d'infrastructure et la conformité des licences soient en meilleur état, les failles au niveau du code et de la gestion des secrets représentent un risque immédiat pour les opérations commerciales et la confiance des clients. Nous recommandons fortement de prioriser les vulnérabilités critiques avant le prochain déploiement.

Top 3 Risques Critiques

Injection SQL et exfiltration de données

Impact Business: Un attaquant peut extraire l'intégralité de la base utilisateurs — mots de passe hashés, emails et données personnelles — via l'endpoint de recherche non paramétré. Combiné au mot de passe de base de données exposé dans docker-compose.yml, cela peut mener à une compromission totale de la base, des obligations de notification RGPD et des amendes réglementaires potentielles.

Recommandation: Paramétrer immédiatement toutes les requêtes SQL en utilisant un ORM ou un query builder. Renouveler les identifiants de base de données et restreindre l'accès direct au réseau interne uniquement.

Exposition de clés secrètes et fraude aux paiements

Impact Business: Une clé secrète Stripe live (sk_live_) est committée dans l'historique du dépôt. Tout développeur, prestataire ou attaquant ayant accès au dépôt peut effectuer des transactions non autorisées, émettre des remboursements frauduleux ou accéder aux données de paiement des clients. L'exposition de la clé AWS risque également une utilisation non autorisée des ressources cloud.

Recommandation: Renouveler immédiatement la clé Stripe dans le dashboard Stripe, révoquer les credentials AWS dans la console IAM, supprimer tous les secrets du code source et implémenter une solution de gestion de secrets (variables d'environnement ou vault). Auditer les logs Stripe pour détecter des transactions non autorisées.

Exécution de code à distance via les dépendances

Impact Business: jsonwebtoken@8.5.1 (CVE-2022-23529) permet l'exécution de code à distance via des JWK forgés, et lodash@4.17.19 (CVE-2021-23337) permet l'injection de commandes via la fonction template(). Les deux packages sont directement importés en production. Un attaquant exploitant l'une de ces vulnérabilités obtient un accès complet au serveur.

Recommandation: Mettre à jour jsonwebtoken vers >=9.0.0 et lodash vers >=4.17.21. Implémenter un scan automatisé des dépendances dans le CI/CD. Lancer npm audit avant chaque déploiement.

Recommandations Prioritaires

1 Paramétrer toutes les requêtes en base de données et adopter un query builder ou ORM pour éliminer le risque d'injection SQL dans tout le codebase.

Effort: 2-3 jours Impact: Élimine tous les vecteurs d'injection SQL (3 découvertes)

1 Renouveler et révoquer tous les secrets exposés : clé API Stripe, credentials AWS, mot de passe de base de données et clé privée RSA. Implémenter une injection de secrets par variables d'environnement uniquement.

Effort: 2-4 heures Impact: Empêche l'accès non autorisé aux services de paiement, cloud et base de données

1 Mettre à jour toutes les dépendances vulnérables vers des versions patchées : jsonwebtoken $\geq 9.0.0$, lodash $\geq 4.17.21$, express $\geq 4.19.2$, axios $\geq 1.6.0$, pg $\geq 8.11.0$.

Effort: 1 jour Impact: Corrige 6 CVE connues dont 2 exécutions de code à distance

2 Implémenter la validation des entrées avec une bibliothèque de schémas (ex: Zod) et l'encodage des sorties sur tous les endpoints API pour éliminer les vecteurs XSS, SSRF et pollution de prototype.

Effort: 3-5 jours Impact: Élimine les risques d'injection et d'intégrité des données sur tous les endpoints

3 Durcir la configuration Docker : ajouter une directive USER non-root, définir un HEALTHCHECK, restreindre les ports exposés au réseau interne et mettre à jour l'image de base de node:16 vers node:20.

Effort: 0.5 jour Impact: Réduit le risque d'évasion de conteneur et corrige 12 CVE au niveau OS

Estimation de Remédiation

Estimation de 2-3 semaines pour un développeur unique afin de traiter toutes les vulnérabilités critiques et élevées. Nous recommandons de commencer par le renouvellement des secrets (2-4 heures) et la mise à jour des dépendances (1 jour) qui offrent la meilleure réduction de risque avec un effort minimal. Les corrections d'injection SQL (2-3 jours) doivent suivre immédiatement. Le durcissement de la validation des entrées peut être planifié pour le sprint suivant.

Conformité OWASP Top 10

● A01	Contrôle d'accès défaillant	● A02	Défaillances cryptographiques
● A03	Injection	● A04	Conception non sécurisée
● A05	Mauvaise configuration	● A06	Composants vulnérables
● A07	Défaillances d'authentification	● A08	Défaillances d'intégrité des données
● A09	Journalisation et surveillance	● A10	SSRF

RAPPORT EXEMPLE

RAPPORT EXEMPLE