



Rapport Technique de Sécurité

Audit de Sécurité Automatisé — Analyse Détaillée

acme/web-platform

Branche: main · Date: 2026-02-10



42

TOTAL

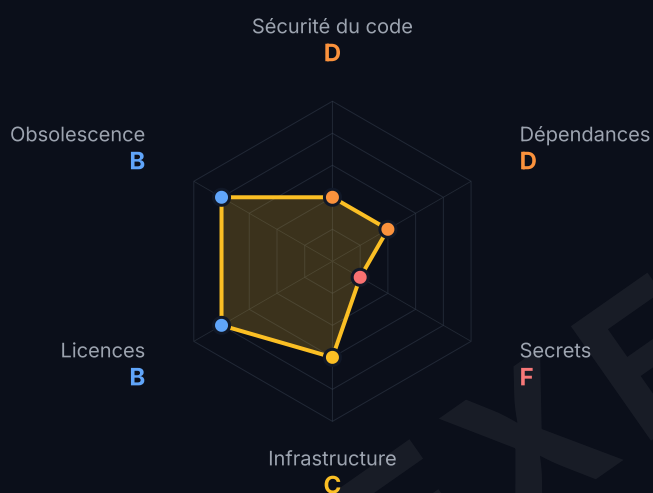
5

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

Détail des Vulnérabilités

● CRITICAL Injection SQL dans l'endpoint de recherche

src/routes/api/search.ts:47

Le paramètre de requête fourni par l'utilisateur est concaténé directement dans une chaîne SQL sans paramétrage ni assainissement. Un attaquant peut injecter du SQL arbitraire pour extraire, modifier ou supprimer des données de n'importe quelle table.

Impact Business: Compromission totale de la base de données. Un attaquant peut exfiltrer tous les enregistrements utilisateurs, données de paiement et identifiants administrateur en une seule requête.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
const results = await db.raw(
  `SELECT * FROM products WHERE name LIKE '%${req.query.q}%'`
);

// FIXED — use parameterized query:
const results = await db('products')
  .where('name', 'like', `%${req.query.q}%`);
```

<https://cwe.mitre.org/data/definitions/89.html>

https://owasp.org/Top10/A03_2021-Injection/

● CRITICAL XSS réfléchi via paramètre URL

src/routes/api/preview.ts:23

L'endpoint de prévisualisation injecte un paramètre URL directement dans la réponse HTML sans encodage. Un attaquant peut créer une URL qui exécute du JavaScript arbitraire dans le navigateur de la victime.

Impact Business: Vol de session, vol d'identifiants et attaques de phishing. Un attaquant peut voler les tokens d'authentification et usurper l'identité de tout utilisateur cliquant sur un lien malveillant.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
res.send(`<h1>Preview: ${req.query.title}</h1>`);

// FIXED — escape HTML entities:
import { escapeHtml } from '../utils/sanitize';
res.send(`<h1>Preview: ${escapeHtml(req.query.title)}</h1>`);
```

<https://cwe.mitre.org/data/definitions/79.html>

https://owasp.org/Top10/A03_2021-Injection/

● CRITICAL eval() avec entrée contrôlée par l'utilisateur

src/utils/template-engine.ts:89

Le moteur de templates utilise eval() pour traiter les chaînes de template fournies par l'utilisateur. Cela permet à un attaquant d'exécuter du JavaScript arbitraire sur le serveur via des variables de template.

Impact Business: Compromission totale du serveur. Un attaquant peut exécuter des commandes système, lire des fichiers, établir des reverse shells et pivoter vers l'infrastructure interne.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
function renderTemplate(template: string, data: object) {
  return eval(`\`${template}\``);
}

// FIXED – use a safe template library:
import Handlebars from 'handlebars';
function renderTemplate(template: string, data: object) {
  return Handlebars.compile(template)(data);
}
```

<https://cwe.mitre.org/data/definitions/95.html>

● HIGH Pollution de prototype via merge récursif

src/utils/deep-merge.ts:15

L'utilitaire de fusion profonde ne vérifie pas les propriétés __proto__, constructor ou prototype. Un attaquant peut injecter des propriétés dans Object.prototype, affectant tous les objets de l'application.

Impact Business: Dénier de service, contournement d'authentification ou exécution de code selon les patterns d'accès aux propriétés en aval.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
function deepMerge(target: any, source: any) {
  for (const key in source) {
    target[key] = typeof source[key] === 'object'
      ? deepMerge(target[key] || {}, source[key])
      : source[key];
  }
  return target;
}

// FIXED – block dangerous keys:
const BLOCKED = new Set(['__proto__', 'constructor', 'prototype']);
function deepMerge(target: any, source: any) {
  for (const key in source) {
    if (BLOCKED.has(key)) continue;
    target[key] = typeof source[key] === 'object'
      ? deepMerge(target[key] || {}, source[key])
      : source[key];
  }
  return target;
}
```

<https://cwe.mitre.org/data/definitions/1321.html>

● HIGH Secret JWT codé en dur dans le code source

src/config/auth.ts:12

Le secret de signature JWT est codé en dur comme littéral de chaîne dans le fichier de configuration d'authentification. Toute personne ayant accès au code source peut forger des tokens JWT valides pour n'importe quel utilisateur.

Impact Business: Contournement complet de l'authentification. Un attaquant peut créer des tokens JWT de niveau administrateur et accéder à n'importe quel compte sans identifiants.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
const JWT_SECRET = 'super-secret-jwt-key-2024';

// FIXED - use environment variable:
const JWT_SECRET = process.env.JWT_SECRET;
if (!JWT_SECRET) throw new Error('JWT_SECRET must be set');
```

<https://cwe.mitre.org/data/definitions/798.html>

● HIGH Redirection ouverte sans validation d'URL

src/routes/api/oauth/callback.ts:34

Le handler de callback OAuth redirige vers une URL fournie par l'utilisateur sans valider la destination. Un attaquant peut créer une URL de connexion qui redirige vers un site de phishing après l'authentification.

Impact Business: Attaques de phishing exploitant un domaine de confiance. Les utilisateurs sont redirigés vers des sites malveillants après s'être authentifiés.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
res.redirect(req.query.redirect_uri);

// FIXED - validate against allowlist:
const ALLOWED_ORIGINS = ['https://acme.com', 'https://app.acme.com'];
const url = new URL(req.query.redirect_uri);
if (!ALLOWED_ORIGINS.includes(url.origin)) {
  return res.status(400).json({ error: 'Invalid redirect' });
}
res.redirect(req.query.redirect_uri);
```

<https://cwe.mitre.org/data/definitions/601.html>

https://owasp.org/Top10/A01_2021-Broken_Access_Control/

● HIGH Traversée de chemin dans le téléchargement de fichiers

src/routes/api/files/download.ts:28

Le handler de téléchargement utilise l'entrée utilisateur pour construire les chemins de fichiers sans assainissement. Un attaquant peut utiliser des séquences ../ pour lire des fichiers arbitraires du serveur.

Impact Business: Exposition de fichiers sensibles du serveur incluant /etc/passwd, variables d'environnement, clés privées et code source.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
const filePath = path.join(UPLOADS_DIR, req.params.filename);
res.sendFile(filePath);

// FIXED – resolve and verify path stays within allowed directory:
const filePath = path.resolve(UPLOADS_DIR, req.params.filename);
if (!filePath.startsWith(path.resolve(UPLOADS_DIR))) {
  return res.status(403).json({ error: 'Access denied' });
}
res.sendFile(filePath);
```

<https://cwe.mitre.org/data/definitions/22.html>

● HIGH Vulnérabilité ReDoS dans la validation d'email

src/utils/validators.ts:67

La regex de validation d'email contient un pattern vulnérable au déni de service par expression régulière (ReDoS). Une entrée forgée cause un backtracking exponentiel, consommant le CPU.

Impact Business: Déni de service. Un attaquant peut envoyer une seule requête forgée qui bloque le serveur pendant plusieurs minutes.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
const EMAIL_REGEX = /^( [a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+\.( [a-zA-Z]{2,5})$)/;

// FIXED – use a non-backtracking approach:
import { z } from 'zod';
const emailSchema = z.string().email();
```

<https://cwe.mitre.org/data/definitions/1333.html>

● MEDIUM Protection CSRF manquante sur endpoint modifiant l'état

src/routes/api/settings.ts:15

L'endpoint de mise à jour des paramètres accepte des requêtes POST sans validation de token CSRF.

Impact Business: Prise de contrôle de compte via modification des paramètres.

CORRECTION RECOMMANDÉE

```
// Add CSRF middleware:
import csrf from 'csrf';
router.use(csrf({ cookie: true }));
```

<https://cwe.mitre.org/data/definitions/352.html>

● MEDIUM Générateur de nombres aléatoires non sécurisé pour les tokens

src/utils/tokens.ts:8

Math.random() est utilisé pour générer les tokens de réinitialisation de mot de passe et de vérification email. Math.random() n'est pas cryptographiquement sûr.

Impact Business: La prédiction de tokens permet la prise de contrôle de comptes.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
const token = Math.random().toString(36).slice(2);

// FIXED - use crypto.randomBytes:
import { randomBytes } from 'node:crypto';
const token = randomBytes(32).toString('hex');
```

<https://cwe.mitre.org/data/definitions/330.html>

● MEDIUM Rate limiting manquant sur les endpoints d'authentification

src/routes/api/auth/login.ts:12

Les endpoints de connexion et d'inscription n'ont aucun rate limiting configuré.

Impact Business: Compromission d'identifiants par force brute.

CORRECTION RECOMMANDÉE

```
// Add rate limiting middleware:
import rateLimit from 'express-rate-limit';
const authLimiter = rateLimit({
  windowMs: 15 * 60 * 1000,
  max: 5,
  message: 'Too many attempts'
});
router.post('/login', authLimiter, loginHandler);
```

<https://cwe.mitre.org/data/definitions/307.html>

● MEDIUM Messages d'erreur verbeux exposant des détails internes

src/middleware/error-handler.ts:22

Le handler d'erreurs envoie les stack traces complètes et les détails d'erreur internes dans les réponses API.

Impact Business: La divulgation d'informations facilite les attaques ciblées.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
res.status(500).json({
  error: err.message,
  stack: err.stack,
  query: err.query
});

// FIXED – generic error in production:
res.status(500).json({
  error: process.env.NODE_ENV === 'production'
    ? 'Internal server error'
    : err.message
});
```

<https://cwe.mitre.org/data/definitions/209.html>

● MEDIUM Cookie de session sans flag Secure

src/config/session.ts:18

Le cookie de session est configuré sans le flag Secure, permettant sa transmission sur des connexions HTTP non chiffrées.

Impact Business: Vol de session sur les connexions non-HTTPS.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
app.use(session({
  cookie: { httpOnly: true }
}));

// FIXED – add Secure and SameSite flags:
app.use(session({
  cookie: { httpOnly: true, secure: true, sameSite: 'strict' }
}));
```

<https://cwe.mitre.org/data/definitions/614.html>

● MEDIUM En-tête Content-Security-Policy manquant

src/middleware/headers.ts:5

L'application ne définit pas d'en-tête Content-Security-Policy, facilitant l'exploitation XSS.

Impact Business: Amplifie l'impact XSS en permettant le chargement de scripts externes.

CORRECTION RECOMMANDÉE

```
// Add CSP header:
app.use((req, res, next) => {
  res.setHeader('Content-Security-Policy',
    "default-src 'self'; script-src 'self'; style-src 'self' 'unsafe-inline'"
  );
  next();
});
```

<https://cwe.mitre.org/data/definitions/1021.html>

● MEDIUM Type de fichier uploadé non validé

src/routes/api/upload.ts:31

L'endpoint d'upload de fichiers accepte tout type de fichier sans validation.

Impact Business: Exécution de code à distance via des web shells uploadés.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
upload.single('file');

// FIXED – validate file type:
const ALLOWED_TYPES = ['image/jpeg', 'image/png', 'application/pdf'];
if (!ALLOWED_TYPES.includes(req.file.mimetype)) {
  return res.status(400).json({ error: 'Invalid file type' });
}
```

<https://cwe.mitre.org/data/definitions/434.html>

● MEDIUM Hachage de mot de passe faible (bcrypt cost=4)

src/services/user-service.ts:45

Les mots de passe sont hachés avec bcrypt à un facteur de coût de 4. Cela rend le cracking par force brute trivialement rapide.

Impact Business: Cracking de la base de mots de passe en quelques heures en cas de compromission.

CORRECTION RECOMMANDÉE

```
// VULNERABLE:
const hash = await bcrypt.hash(password, 4);

// FIXED — use cost factor of 12 (or switch to argon2id):
const hash = await bcrypt.hash(password, 12);
// Or better: use argon2id
import argon2 from 'argon2';
const hash = await argon2.hash(password);
```

<https://cwe.mitre.org/data/definitions/916.html>

● CRITICAL lodash@4.17.19 — Injection de commandes via template()

package.json

Les versions de lodash antérieures à 4.17.21 sont vulnérables à l'injection de commandes via la fonction template() avec une entrée contrôlée par l'utilisateur.

Impact Business: Exécution de code à distance. Un attaquant peut exécuter des commandes système arbitraires.

CORRECTION RECOMMANDÉE

```
Update lodash to >=4.17.21:
npm install lodash@latest
```

<https://nvd.nist.gov/vuln/detail/CVE-2021-23337>

<https://github.com/lodash/lodash/pull/5085>

● CRITICAL jsonwebtoken@8.5.1 — Récupération de clé non sécurisée (RCE)

package.json

Les versions de jsonwebtoken antérieures à 9.0.0 ne valident pas correctement le paramètre secretOrPublicKey, permettant l'exécution de code via un JWK forgé.

Impact Business: Exécution de code à distance via des tokens JWT forgés.

CORRECTION RECOMMANDÉE

```
Update jsonwebtoken to >=9.0.0:
npm install jsonwebtoken@latest
```

<https://nvd.nist.gov/vuln/detail/CVE-2022-23529>

<https://github.com/auth0/node-jwebtoken/releases/tag/v9.0.0>

● **HIGH** **express@4.17.1 — Redirection ouverte dans res.redirect()**

package.json

Les versions d'Express antérieures à 4.19.2 ont une vulnérabilité de redirection ouverte dans res.redirect().

Impact Business: Attaques de phishing utilisant des URLs de confiance.

CORRECTION RECOMMANDÉE

Update express to >=4.19.2:
npm install express@latest

<https://nvd.nist.gov/vuln/detail/CVE-2024-29041>

● **HIGH** **axios@0.21.1 — SSRF via contournement d'intercepteur**

package.json

Les versions d'axios antérieures à 1.6.0 sont vulnérables au SSRF, permettant de contourner les intercepteurs de requêtes.

Impact Business: Accès aux services internes et aux métadonnées cloud.

CORRECTION RECOMMANDÉE

Update axios to >=1.6.0:
npm install axios@latest

<https://nvd.nist.gov/vuln/detail/CVE-2023-45857>

● **HIGH** **pg@8.7.1 — Injection SQL via paramètres de connexion**

package.json

Les versions de pg antérieures à 8.11.0 sont vulnérables à l'injection SQL via des paramètres de connexion forgés.

Impact Business: Compromission de la base de données via injection au niveau connexion.

CORRECTION RECOMMANDÉE

Update pg to >=8.11.0:
npm install pg@latest

<https://nvd.nist.gov/vuln/detail/CVE-2024-21511>

● HIGH node:16-alpine — 12 CVE dont 2 critiques (OpenSSL)

Dockerfile

L'image Docker de base contient 12 vulnérabilités connues dans les packages système. Node.js 16 a atteint sa fin de vie.

Impact Business: Contournement de la sécurité TLS et potentielles attaques man-in-the-middle.

CORRECTION RECOMMANDÉE

Update the base image to node:20-alpine in the Dockerfile:
FROM node:20-alpine

<https://endoflife.date/nodejs>

● HIGH Clé secrète AWS exposée dans .env.example

.env.example:12

Une vraie AWS Secret Access Key est présente dans .env.example, qui est suivi par git.

Impact Business: Accès non autorisé aux ressources AWS.

CORRECTION RECOMMANDÉE

Replace real values in .env.example with placeholders:
AWS_SECRET_ACCESS_KEY=your-secret-key-here

Then revoke the exposed key in IAM and generate a new one.

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html

● HIGH Clé secrète Stripe live committée dans le code

src/config/payments.ts:8

Une clé secrète Stripe live (sk_live_) est codée en dur dans le fichier de configuration des paiements.

Impact Business: Fraude financière et exposition des données de paiement.

CORRECTION RECOMMANDÉE

Move to environment variable:
const STRIPE_KEY = process.env.STRIPE_SECRET_KEY;

Rotate the key immediately in the Stripe dashboard.

<https://stripe.com/docs/keys>

● HIGH Clé privée RSA committée dans le dépôt

certs/server.pem:1

Un fichier de clé privée RSA encodé en PEM est committé dans le dépôt.

Impact Business: Usurpation TLS et déchiffrement du trafic.

CORRECTION RECOMMANDÉE

Remove the private key from the repository:

```
git rm certs/server.pem
```

Add *.pem to .gitignore. Generate a new key pair and store securely.

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/04-Testing_for_Weak_Encryption

● HIGH Mot de passe de base de données dans docker-compose

docker-compose.yml:24

Le mot de passe PostgreSQL est codé en dur dans docker-compose.yml, committé dans le dépôt.

Impact Business: Accès direct à la base de données de production.

CORRECTION RECOMMANDÉE

Use environment variables or Docker secrets:

environment:

```
POSTGRES_PASSWORD: ${DB_PASSWORD}
```

Store the actual password in a .env file excluded from git.

<https://docs.docker.com/compose/environment-variables/>

● MEDIUM Dockerfile s'exécute en tant que root

Dockerfile:1

Le Dockerfile n'inclut pas de directive USER, l'application s'exécute en root dans le conteneur.

Impact Business: Compromission étendue du conteneur. L'accès root facilite l'évasion de conteneur.

CORRECTION RECOMMANDÉE

Add a non-root user:

```
RUN addgroup -S appuser && adduser -S appuser -G appuser
```

```
USER appuser
```

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#user

● MEDIUM Port PostgreSQL 5432 exposé publiquement

docker-compose.yml:14

docker-compose.yml mappe le port PostgreSQL sur toutes les interfaces (0.0.0.0:5432).

Impact Business: Accès direct à la base de données depuis Internet.

CORRECTION RECOMMANDÉE

```
Bind to localhost only:
ports:
  - "127.0.0.1:5432:5432"
```

<https://docs.docker.com/compose/networking/>

● MEDIUM Pas de HEALTHCHECK défini dans le Dockerfile

Dockerfile:22

Le Dockerfile ne définit pas d'instruction HEALTHCHECK.

Impact Business: Temps d'arrêt prolongé en cas de défaillance sans récupération automatique.

CORRECTION RECOMMANDÉE

```
Add a health check:
HEALTHCHECK --interval=30s --timeout=5s --retries=3 \
  CMD wget -q0- http://localhost:3000/health || exit 1
```

<https://docs.docker.com/engine/reference/builder/#healthcheck>

● LOW Dépendance GPL-3.0 dans un projet propriétaire

package-lock.json

La dépendance node-forge@1.3.1 est sous licence GPL-3.0, incompatible avec un projet propriétaire.

Impact Business: Risque de conformité juridique.

CORRECTION RECOMMANDÉE

```
Replace node-forge with an MIT/ISC-licensed alternative:
npm uninstall node-forge
npm install @peculiar/webcrypto # MIT licensed
```

<https://www.gnu.org/licenses/gpl-3.0.en.html>

● INFO **React 17 in use — React 19 available**

The project uses React 17. React 19 offers improved performance, automatic batching, and new hooks. Consider upgrading.

Impact Business: No immediate security impact. Missing performance improvements and new features.

CORRECTION RECOMMANDÉE

```
npm install react@latest react-dom@latest
```

● INFO **TypeScript 4.9 — TypeScript 5.x available**

TypeScript 4.9 is in use. Version 5.x includes improved type checking, decorators support, and faster compilation.

Impact Business: No security impact. Improved developer experience available.

CORRECTION RECOMMANDÉE

```
npm install typescript@latest
```

● INFO **No .nvmrc or .node-version file**

No Node.js version pinning file found. Different developers may run different Node.js versions, causing inconsistencies.

Impact Business: Potential build inconsistencies across environments.

CORRECTION RECOMMANDÉE

```
echo "20" > .nvmrc
```

● INFO **No lockfile integrity check in CI**

The CI pipeline does not use `--frozen-lockfile` or equivalent, allowing dependency resolution differences between environments.

Impact Business: Supply chain risk from inconsistent dependency resolution.

CORRECTION RECOMMANDÉE

Use `npm ci` instead of `npm install` in CI pipelines.

● INFO **console.log statements in production code**

23 `console.log` statements found in production source files. These may leak sensitive information in production logs.

Impact Business: Potential information disclosure via server logs.

CORRECTION RECOMMANDÉE

Replace `console.log` with a structured logger (e.g., `pino`, `winston`).

● INFO **No CORS configuration detected**

No explicit CORS configuration found. The API may accept requests from any origin by default.

Impact Business: Cross-origin data access if sensitive endpoints are exposed.

CORRECTION RECOMMANDÉE

Configure CORS with explicit allowed origins.

● INFO **Missing X-Content-Type-Options header**

The X-Content-Type-Options: nosniff header is not set. Browsers may MIME-sniff responses, potentially treating non-executable content as scripts.

Impact Business: Minor XSS amplification risk.

CORRECTION RECOMMANDÉE

Add helmet middleware: `app.use(helmet())`

● INFO **No Referrer-Policy header**

The Referrer-Policy header is not configured. Browser may send full URL in the Referer header to external sites.

Impact Business: URL-based information leakage.

CORRECTION RECOMMANDÉE

Set Referrer-Policy: `strict-origin-when-cross-origin`

● INFO **webpack-dev-server dependency detected**

webpack-dev-server is listed in production dependencies instead of devDependencies.

Impact Business: Larger production bundle and potential dev-only code in production.

CORRECTION RECOMMANDÉE

Move to devDependencies in package.json.

● INFO **No automated security scanning in CI/CD**

No security scanning tools (SAST, SCA, secret detection) are configured in the CI/CD pipeline.

Impact Business: Vulnerabilities may be introduced without detection.

CORRECTION RECOMMANDÉE

Add Shieldify scan to your CI pipeline for automated security scanning.

● INFO **Potentially unused dependencies detected**

4 packages in dependencies appear unused in the codebase: moment, bluebird, request, underscore.

Impact Business: Increased attack surface from unnecessary packages.

CORRECTION RECOMMANDÉE

Remove unused dependencies: `npm uninstall moment bluebird request underscore`

● INFO **Database migrations not version-controlled**

No database migration files found in the repository. Schema changes may be applied manually.

Impact Business: Deployment inconsistencies and difficult rollbacks.

CORRECTION RECOMMANDÉE

Adopt a migration tool like Prisma Migrate or knex migrations.

Arbre de Dépendances Vulnérables

PACKAGE	VERSION	VULNÉRABILITÉS	UTILISÉ PAR
lodash	4.17.19	CVE-2021-23337 (COMMAND INJECTION)	express-validator@6.14.0, webpack@5.75.0
jsonwebtoken	8.5.1	CVE-2022-23529 (RCE)	passport-jwt@4.0.1
express	4.17.1	CVE-2024-29041 (OPEN REDIRECT)	
axios	0.21.1	CVE-2023-45857 (SSRF)	stripe@11.0.0
pg	8.7.1	CVE-2024-21511 (SQL INJECTION)	knex@2.4.0
node-forge	1.3.1	GPL-3.0 LICENSE (INCOMPATIBLE)	selfsigned@2.1.1, webpack-dev-server@4.11.0

Détails d'Exécution des Scanners

SCANNER	CATÉGORIE	RÉSULTATS	DURÉE	STATUT
Semgrep	SAST	16	45.2s	OK
Trivy (SCA)	Dependency Audit	6	12.8s	OK
Trivy (IaC)	Infrastructure	3	3.4s	OK
Gitleaks	Secrets	4	8.9s	OK
license-checker	Licenses	1	2.1s	OK
npm audit	SCA (npm)	4	5.6s	OK

RAPPORT EXEMPLE