Rapport de Sécurité

Analyse automatisée des vulnérabilités

Mon Projet Security Scan

Généré le 1 décembre 2025 à 21:46

ID du rapport : 69cc38b5-6328-426b-b2b7-1de249851763

SecureTech Solutions

Résumé Exécutif

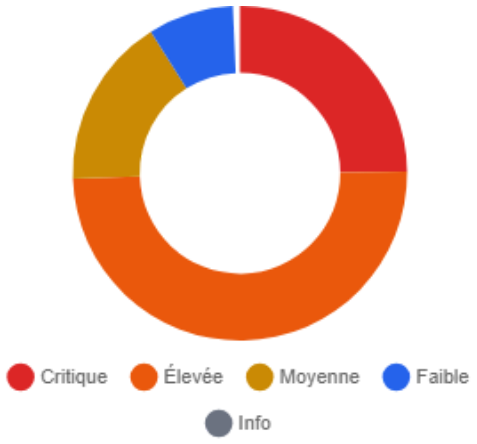
8
/100

Risque CRITIQUE

12	3	6	4
VULNÉRABILITÉS TOTALES	CRITIQUES	ÉLEVÉES	OUTILS UTILISÉS

3	6	2	1	0
CRITIQUE	ÉLEVÉE	MOYENNE	FAIBLE	INFO

Répartition par sévérité



Répartition par catégorie

Fichiers les plus affectés

FICHIER	VULNÉRABILITÉS
src/controllers/UserController.java	1
src/views/profile.jsp	1
src/config/DatabaseConfig.java	1
src/services/AuthService.java	1
package.json (lodash@4.17.15)	1
package.json (axios@0.21.0)	1
package.json (minimist@1.2.0)	1
config/aws-config.js	1
https://example.com/search?q=<script>alert(1)</script>	1
https://example.com/api/users?id=1' OR '1'='1	1

Recommandations Prioritaires

0. Corriger cette vulnérabilité selon les bonnes pratiques de sécurité.
1. Révoquer immédiatement ce secret, le retirer du code source et utiliser un gestionnaire de secrets sécurisé (HashiCorp Vault, AWS Secrets Manager, etc.)
2. Upgrade to lodash@4.17.21
3. Mettre à jour axios vers une version non vulnérable
4. Phase: Architecture and Design. Use a vetted library or framework that does not allow this weakness to occur.
5. Do not trust client side input. Use parameterized queries or stored procedures.
6. Mettre à jour minimist vers une version non vulnérable
7. Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers.
8. Whenever a cookie contains sensitive information or is a session token, then it should always be passed using an encrypted channel.

Détail des Vulnérabilités

CRITIQUE

SQL Injection vulnerability - User input is directly concatenated into SQL query

src/controllers/UserController.java Ligne 45 Qualité du code

SQL Injection vulnerability - User input is directly concatenated into SQL query

💡 Recommandation

Corriger cette vulnérabilité selon les bonnes pratiques de sécurité.

Détecté par :

SonarQube

CRITIQUE

Cross-Site Scripting (XSS) - User input is rendered without sanitization

src/views/profile.jsp Ligne 23 Qualité du code

Cross-Site Scripting (XSS) - User input is rendered without sanitization

💡 Recommandation

Corriger cette vulnérabilité selon les bonnes pratiques de sécurité.

Détecté par :

SonarQube

CRITIQUE

AWS exposé dans le code

CWE-798

config/aws-config.js | Ligne 5 | Secrets exposés

AWS Access Key ID exposed in configuration file. Ce secret a été vérifié comme étant actif.

Recommandation

Révoquer immédiatement ce secret, le retirer du code source et utiliser un gestionnaire de secrets sécurisé (HashiCorp Vault, AWS Secrets Manager, etc.)

Détecté par :

TruffleHog

ÉLEVÉE

Hardcoded password detected in source code

src/config/DatabaseConfig.java | Ligne 12 | Qualité du code

Hardcoded password detected in source code

Recommandation

Corriger cette vulnérabilité selon les bonnes pratiques de sécurité.

Détecté par :

SonarQube

ÉLEVÉE

Weak cryptographic algorithm MD5 used for password hashing

src/services/AuthService.java Ligne 78 Qualité du code

Weak cryptographic algorithm MD5 used for password hashing

Recommandation

Corriger cette vulnérabilité selon les bonnes pratiques de sécurité.

Détecté par :

SonarQube

ÉLEVÉE

Prototype Pollution in lodash in lodash@4.17.15

package.json (lodash@4.17.15) Dépendances vulnérables CVSS: 7.4

Versions of lodash prior to 4.17.19 are vulnerable to Prototype Pollution

Recommandation

Upgrade to lodash@4.17.21

Détecté par :

Snyk

ÉLEVÉE

Server-Side Request Forgery (SSRF) in axios@0.21.0

package.json (axios@0.21.0) Dépendances vulnérables CVSS: 7.5

axios before 0.21.1 allows attackers to perform SSRF attacks

Recommandation

Mettre à jour axios vers une version non vulnérable

Détecté par :

Snyk

ÉLEVÉE

Cross Site Scripting (Reflected)

CWE-79

https://example.com/search?q=<script>alert(1)</script> Cross-Site Scripting (XSS)

Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance.

Recommandation

Phase: Architecture and Design. Use a vetted library or framework that does not allow this weakness to occur.

Détecté par :

OWASP ZAP

ÉLEVÉE

SQL Injection

CWE-89

https://example.com/api/users?id=1' OR '1'='1 Injection

SQL injection may be possible. The page results were successfully manipulated using boolean conditions.

Recommandation

Do not trust client side input. Use parameterized queries or stored procedures.

Détecté par :

OWASP ZAP

MOYENNE

Prototype Pollution in minimist in minimist@1.2.0

package.json (minimist@1.2.0) Dépendances vulnérables CVSS: 5.6

minimist before 1.2.3 is vulnerable to Prototype Pollution

Recommandation

Mettre à jour minimist vers une version non vulnérable

Détecté par :

Snyk

MOYENNE **Missing Anti-clickjacking Header** CWE-1021

https://example.com/ Autre

The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.

💡 **Recommandation**

Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers.

Détecté par :

OWASP ZAP

FAIBLE **Cookie Without Secure Flag** CWE-614

https://example.com/login Autre

A cookie has been set without the secure flag, which means the cookie can be accessed via unencrypted connections.

💡 **Recommandation**

Whenever a cookie contains sensitive information or is a session token, then it should always be passed using an encrypted channel.

Détecté par :

OWASP ZAP

Informations du Scan

Date de début
01/12/2025 21:46:11

Durée
N/A

ID du rapport
69cc38b5-6328-426b-b2b7-1de249851763

Outils utilisés
Snyk, SonarQube, TruffleHog, OWASP ZAP

Rapport généré automatiquement par ReportGen Security Scanner
1 décembre 2025 à 21:46 • Mon Projet Security Scan