Rapport d'Audit de Sécurité

Introduction

L'audit de sécurité a été réalisé sur un hôte Docker pour identifier des vulnérabilités potentielles concernant les services accessibles, les fichiers et répertoires sensibles, ainsi que des failles exploitables via des tests de sécurité applicative. Cet audit a été effectué depuis une machine locale, et notre application étant hébergée sur un serveur OVH, une approche agressive n'était pas recommandée. Les informations ci-dessous ne sont donc pas applicables à notre environnement de production (certaines failles ayant été corrigées sur ce dernier, notamment une configuration PostgreSQL inaccessibles publiquement).

Phases de Test Couvertes

Les tests ont couvert différentes phases :

- Énumération
- Bruteforce de répertoires
- Tests SQL
- Analyse de vulnérabilité avec Nessus
- Tests de Cross-Site Scripting (XSS)
- Bruteforce PostgreSQL

---

1. Énumération

Outil Utilisé : Nmap

Résumé

Trois ports ouverts ont été identifiés :

- **Port 4000** : Service HTTP proxy, version détectée : *ncat*
- **Port 5173** : Aucune information supplémentaire disponible
- **Port 5432** : Service PostgreSQL, version : PostgreSQL 9.6 ou plus récent

Détails

L'utilisation de Nmap a permis d'identifier des services clés sur l'hôte audité :

- **Port 5432** : La présence d'un service PostgreSQL pourrait permettre une attaque par énumération des bases de données et des utilisateurs. Aucune exploitation n'a été réalisée dans cet audit.
- **Port 4000** : Un service HTTP proxy via *ncat* pourrait être exploité pour des tentatives de contournement de réseau.
- **Port 5173** : Aucune version ou détail supplémentaire n'a été révélé, limitant les possibilités d'exploitation directe.

---

2. Bruteforce des Répertoires

Outils Utilisés : Gobuster, Dirb, Dirbuster, et un outil personnalisé en Python.

Résumé

L'énumération de répertoires a révélé les endpoints suivants :

- `/components`
- `/manifest`
- `/package`
- `/@vite`
- `/@vite/client`

Analyse

Ces endpoints peuvent potentiellement révéler des informations sensibles sur la configuration du serveur, comme le langage utilisé. La présence de *vite* indique l'utilisation probable de Vue.js pour le front-end. Des tests agressifs ont été effectués, et il est recommandé de bloquer les IP qui accèdent trop souvent à des URL non existantes.

---

3. Tests SQL

Outil Utilisé : SQLMap

Résumé

Une tentative d'injection SQL a été réalisée :

- **/api/users** : Liste des utilisateurs et leurs emails accessibles via une méthode GET (curl).
- Aucun point d'injection SQL exploitable détecté.

Détails

Les tests SQL ont permis d'identifier que certains points de l'application permettent l'envoi de requêtes POST, notamment pour soumettre un email utilisateur. Aucune des requêtes analysées ne s'est révélée vulnérable à une injection SQL.

---

4. Analyse de Vulnérabilité avec Nessus

Outil Utilisé : Nessus

Résumé

- **OpenSSL 3.1.1** : Vulnérabilités détectées :
    - **DDoS** : Vulnérabilité exploitée pour réaliser un déni de service.
    - **RCE après DDoS** : Sévérité haute et critique.
    - **CVE référencées** : CVE-2024-5535, CVE-2024-9143, CVE-2023-4807, CVE-2023-5363, CVE-2023-5678, CVE-2024-4741.

        - **Plan de Remédiation** : Mettre à jour OpenSSL vers la version 3.1.7 pour atténuer les vulnérabilités détectées.
- **Vulnérabilités XSS** : Identifiées dans plusieurs fichiers via des paramètres de query, mais aucune exploitation réussie.
    - **CVE référencées** : CVE-2002-1060, CVE-2002-1700, CVE-2003-1543, CVE-2005-2453, CVE-2006-1681, CVE-2012-3382
    - **Sévérité** : Moyenne

Détails

La version d'OpenSSL utilisée présente plusieurs vulnérabilités critiques, notamment des risques de déni de service et d'exécution de code à distance après un DDoS. La mise à jour à une version plus récente (3.1.7) est fortement recommandée.

---

5. Tests de Cross-Site Scripting (XSS)

Outil Utilisé : Test manuel

---

6. Bruteforce PostgreSQL

Utilisation du module `auxiliary/scanner/postgres/postgres_login` de Metasploit.

Résultat

Des informations d'identification PostgreSQL valides ont été trouvées.

Plan de Remédiation

- Créer un utilisateur avec un mot de passe complexe pour éviter le bruteforce.
- La base de données **time_manager** a été accessible, permettant d'obtenir les combinaisons email/nom d’utilisateur/mot de passe.

Conclusion

Des vulnérabilités XSS potentielles ont été identifiées dans certains paramètres de query, sans preuve d’exploitation directe (possiblement un faux positif). L'utilisation de *credentials* par défaut pour PostgreSQL expose les données sensibles, et des mesures correctives ont été prises sur le serveur de production. La vulnérabilité au DDoS et à l'exécution de code à distance (RCE) liée à OpenSSL nécessite une mise à jour.