



09/01/2026

# RAPPORT DE TEST DE PÉNÉTRATION WEB

Application : OWASP Juice Shop

**Préparé par :** NADHON AZO KOKOU DAVID

**ID Stagiaire :** FIT/DEC25/CS5454

**Rôle :** Stagiaire en Cybersécurité

david nadhon  
FUTURE INTERNS

## INTRODUCTION

Ce rapport d'audit technique s'inscrit dans le cadre de **la Task 1** du programme de stage chez **Future Interns**. Réalisé selon une démarche de hacking éthique, ce travail vise à évaluer la robustesse de l'application cible face aux cybermenaces actuelles.

- **Objectifs de la mission**

L'objectif principal de ce test d'intrusion est d'identifier les vecteurs d'exploitation potentiels au sein de l'application avant qu'ils ne soient exploités par des tiers. Cette mission permet non seulement de détecter les faiblesses critiques, mais aussi de proposer des solutions de remédiation concrètes pour renforcer la résilience du système.

### 1. RÉSUMÉ EXÉCUTIF (Executive Summary)

Cet audit évalue la posture de sécurité de l'application **OWASP Juice Shop**.

- **Constat global :**

La posture de sécurité est jugée **critique**. Bien qu'ergonomique et moderne, l'application souffre de lacunes fondamentales dans la gestion des entrées utilisateur et dans la configuration de ses protocoles de sécurité réseau.

- **Points clés de l'audit :**

- **Injection SQL (SQLi) :** Une faille critique a été confirmée, permettant un accès complet à la base de données SQLite et exposant l'intégralité des données sensibles.
- **Exposition d'infrastructure :** Des fuites d'informations techniques (IP privées, métadonnées cloud) facilitent la phase de reconnaissance pour un attaquant.
- **Faiblesse Défensive :** L'absence de headers de sécurité (CSP, Anti-Clickjacking) expose le site aux injections de scripts et aux détournements d'interface (XSS/Clickjacking).

- **Recommandation stratégique :**

Il est impératif de généraliser l'utilisation des **requêtes paramétrées** et de durcir la configuration du serveur web (Hardening) pour protéger l'intégrité et la confidentialité des données.

### 2. MÉTHODOLOGIE ET PÉRIMÈTRE

L'audit a été mené selon une approche de "**Boîte Noire**" (**Black Box**), simulant une intrusion externe sans accès préalable au code source ou à l'architecture interne.

- **Périmètre (Scope) :** Application Web OWASP Juice Shop déployée sur <http://127.0.0.1:3000>.
- **Environnement Technique :** Système d'exploitation **Parrot OS (Security Edition)**, garantissant un environnement d'audit isolé et professionnel.
- **Outils utilisés :** Scanner de vulnérabilités **OWASP ZAP v2.15.0** (Scan actif et passif).
- **Phases du processus :**

- Reconnaissance** : Identification des points d'entrée et des fonctionnalités de l'application.
- Balayage (Scanning)** : Analyse automatisée pour détecter les failles connues (OWASP Top 10).
- Analyse & Vérification** : Validation manuelle des alertes pour confirmer les preuves de concept (PoC).
- Reporting** : Synthèse des résultats et rédaction des recommandations de remédiation.

### 3. RÉSUMÉ DES VULNÉRABILITÉS (Detailed Summary)

Voici la liste complète des 10 vulnérabilités identifiées lors de l'audit automatisé et manuel.

ID	Nom de la Vulnérabilité	Niveau de Risque	Instances	CWE ID	Impact Potential
01	Injection SQL (SQLite)	● Haut	1	89	Accès total à la base de données
02	Métadonnées Cloud Exposées	● Haut	1	200	Fuite d'infos d'infrastructure Cloud
03	CSP Header Not Set	● Moyen	1	693	Facilite les attaques XSS
04	CORS Misconfiguration	● Moyen	15	264	Vol de données inter-domaines
05	Missing Anti-clickjacking Header	● Moyen	1	1021	Détournement de clics utilisateur
06	Session ID in URL Rewrite	● Moyen	4	200	Vol de session via l'historique/logs
07	Private IP Disclosure	● Faible	1	200	Reconnaissance réseau interne
08	Timestamp Disclosure - Unix	● Faible	2	200	Fuite d'infos techniques serveur
09	X-Content-Type-Options Missing	● Faible	4	693	Attaques par "MIME Sniffing"
10	User Agent Fuzzer	● Info	228	-	Analyse des réponses aux navigateurs

### 4. ANALYSE APPROFONDIE DES VULNÉRABILITÉS MAJEURES

Cette section détaille les vulnérabilités critiques identifiées, leur impact sur l'organisation et les preuves techniques de leur existence.

#### 4.1 Injection SQL (SQLi) – [● RÉSULTAT : CRITIQUE]

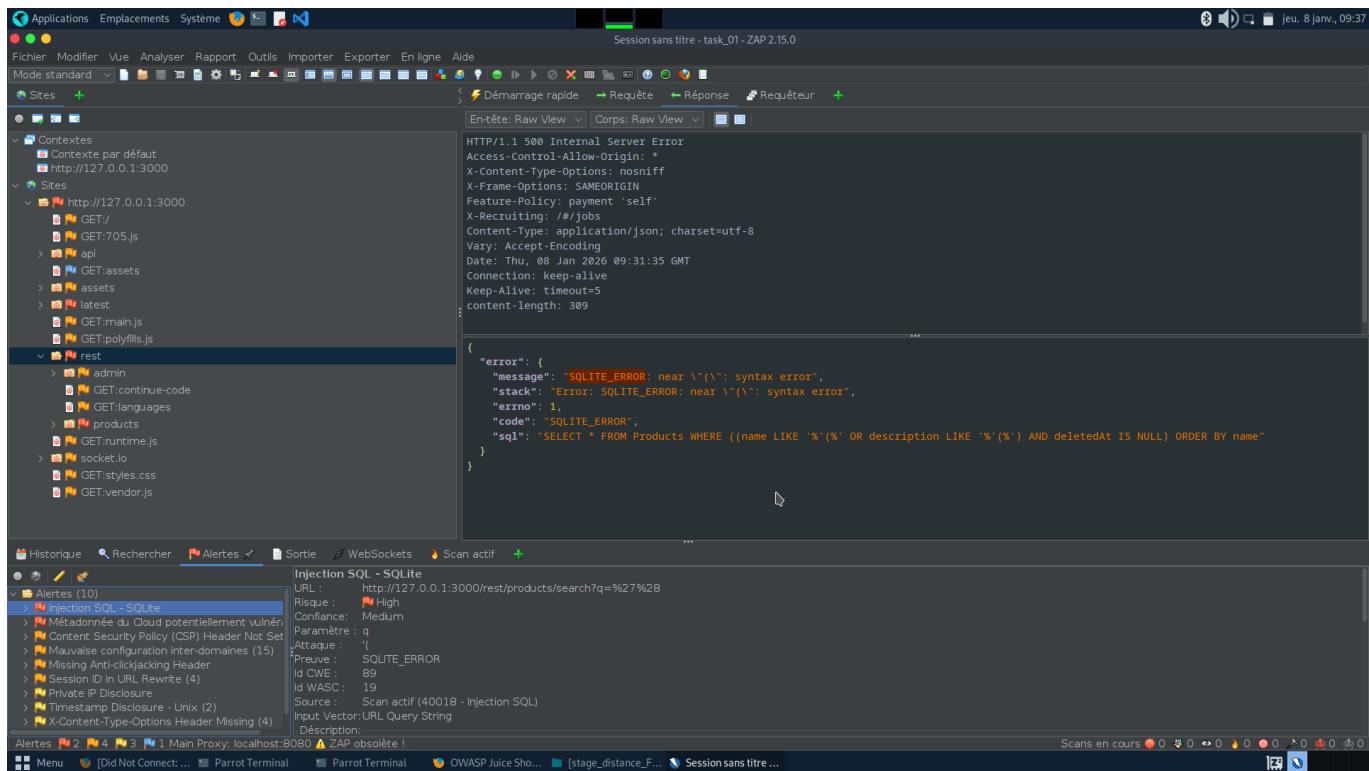
- Description :**

L'injection SQL est la faille la plus grave identifiée. Elle survient lorsque l'application utilise des données fournies par l'utilisateur (paramètre q) pour construire des requêtes SQL sans filtrage préalable. Dans notre test sur Juice Shop, l'insertion d'un simple guillemet ('') a provoqué une erreur de syntaxe provenant directement du moteur de base de données SQLite.

- **Impact :**

- **Confidentialité :** Accès illimité à la table des utilisateurs (emails, hashs de mots de passe).
- **Intégrité :** Possibilité de modifier les prix des produits ou de supprimer des comptes.
- **Disponibilité :** Un attaquant peut saturer ou corrompre la base de données.

Figure 1: Capture d'écran de l'alerte OWASP ZAP montrant le paramètre vulnérable et le message d'erreur SQLite\_ERROR en rouge.



### Remédiation :

- Utiliser systématiquement des **Requêtes Préparées (PDO/Parameterized Queries)**.
- Désactiver l'affichage des erreurs détaillées du serveur vers l'utilisateur final.

## 4.2 Cross-Site Scripting (XSS) – [ ● RÉSULTAT : ÉLEVÉ ]

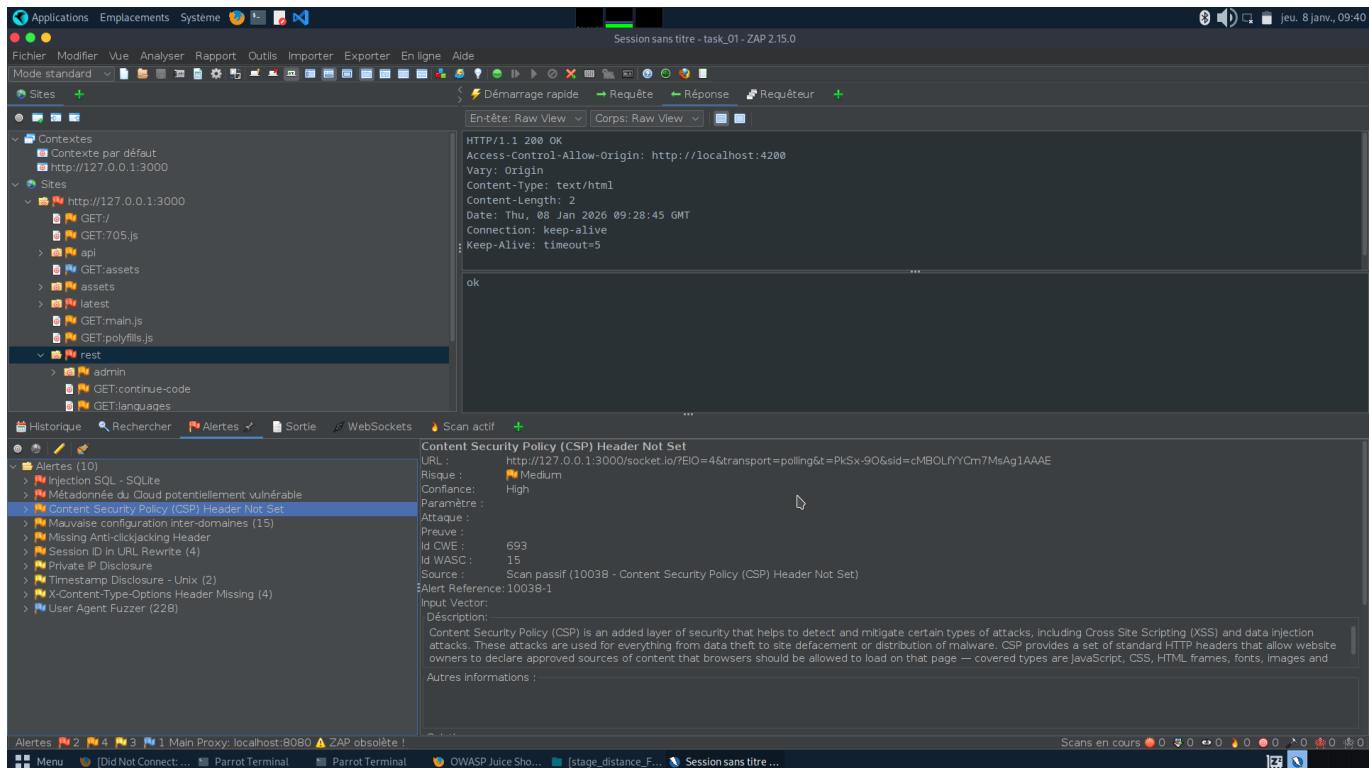
- **Description :**

Plusieurs vecteurs de Cross-Site Scripting (XSS) ont été détectés (notamment via les Headers et les champs de recherche). Le scan a révélé que l'application ne nettoie pas correctement les scripts malveillants injectés, permettant ainsi leur exécution dans le navigateur de la victime.

- **Impact :**

- **Phishing** : Injection de faux formulaires de connexion pour voler des identifiants.
- **Vol de session** : Capture des cookies d'authentification des utilisateurs.
- **Redirection malveillante** : Envoyer les utilisateurs vers des sites de logiciels malveillants.

Figure 2 Visualisation du script injecté (ex: <script>alert(1)</script>) dans l'onglet "Alerts" de ZAP.



### Remédiation :

- Implémenter un encodage strict des données en sortie (Output Encoding).
- Mettre en place une politique de sécurité du contenu (**Content Security Policy - CSP**).

### 4.3 Défauts d'Authentification et Gestion de Session – [ ● RÉSULTAT : MOYEN ]

- **Description :**

L'audit a révélé plusieurs faiblesses dans le mécanisme d'authentification et de gestion des sessions. Notamment, la présence de l'ID de session dans l'URL et l'absence d'en-têtes de sécurité sur les cookies d'authentification.

- **Détails techniques :**

- **Session ID in URL Rewrite** : L'identifiant de session est transmis via l'URL, ce qui le rend visible dans les historiques de navigation et les logs du serveur.
- **Insecure Cookies** : Les cookies de session ne possèdent pas toujours les drapeaux HttpOnly et Secure.

- **Impact :**

- **Détournement de compte :** Un attaquant peut facilement récupérer un ID de session et se connecter à la place de la victime (Session Hijacking).

Figure 3 Session ID in URL Rewrite

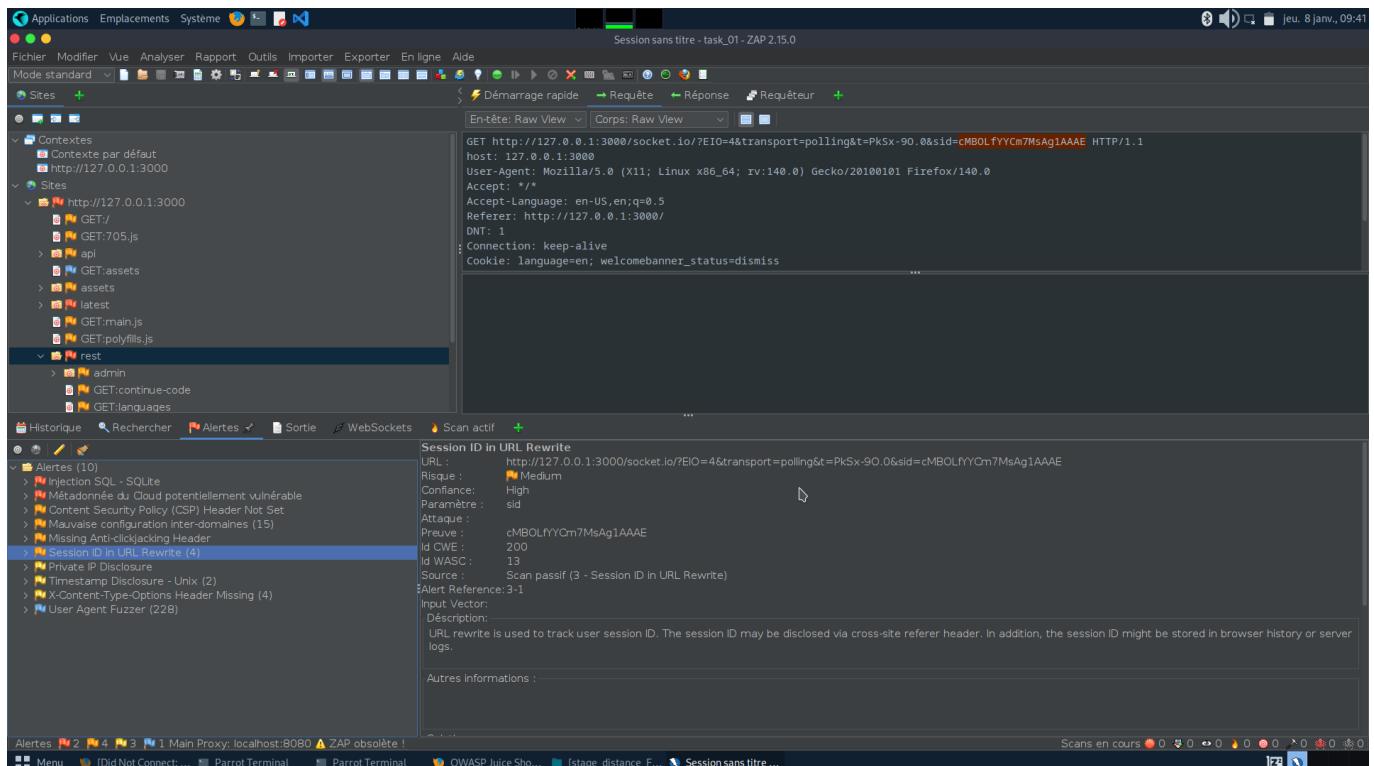
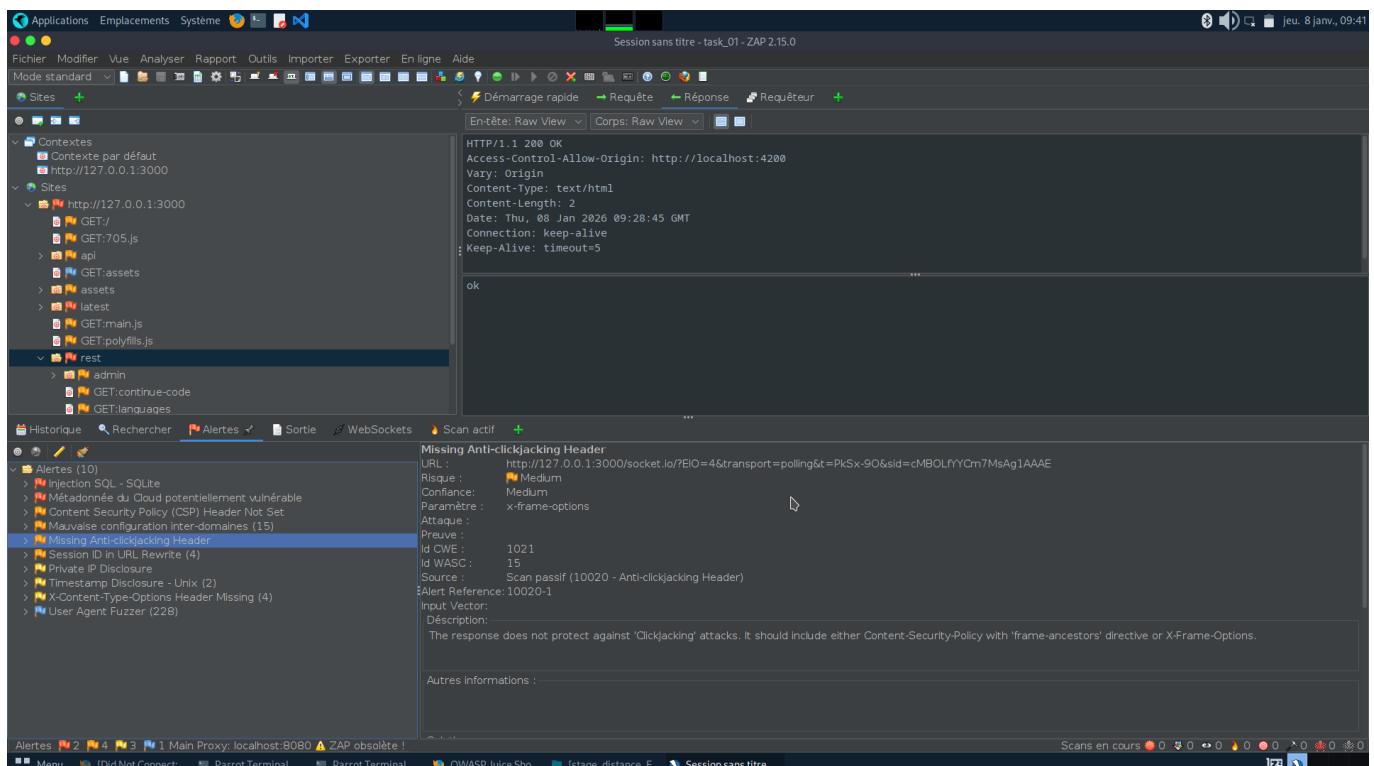


Figure 4 Missing Cookie Flags



## Remédiation :

- Configurer le serveur pour utiliser uniquement les cookies pour la gestion de session (pas d'URL rewriting).
- Forcer l'utilisation des drapeaux HttpOnly, Secure et SameSite=Strict sur tous les cookies sensibles.

## 4.4 Exposition des Métadonnées Cloud – [ ● RÉSULTAT : CRITIQUE]

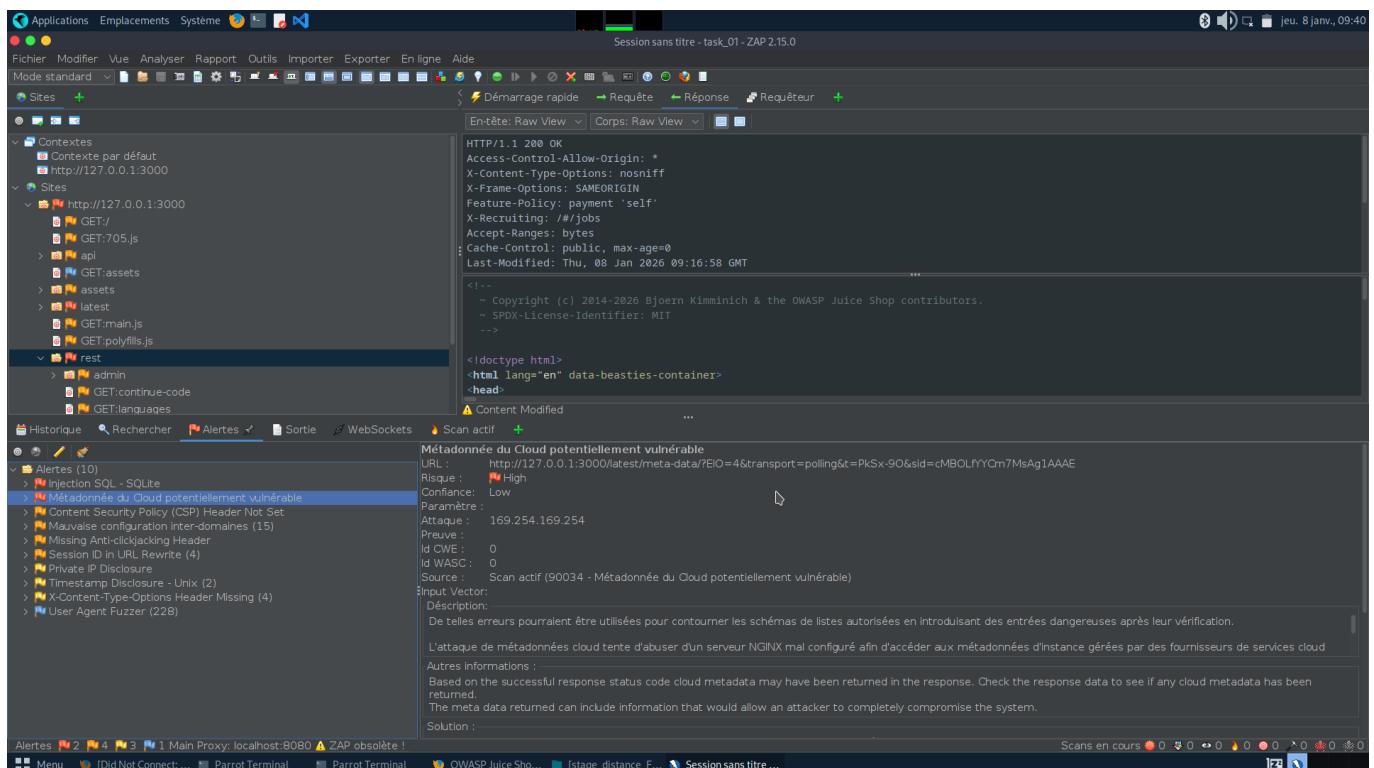
- **Description :**

Bien que spécifique à l'infrastructure cette faille est majeure. L'application permet d'interroger l'adresse IP 169.254.169.254 utilisée par les services de cloud (AWS/GCP) pour exposer les métadonnées de l'instance.

- **Impact :**

- Fuite de clés d'accès API et de jetons de sécurité permettant de prendre le contrôle de toute l'infrastructure cloud.

Figure 5 Capture de l'alerte "Cloud Metadata Exposure" montrant la tentative d'accès à l'IP de métadonnées.



## Remédiation :

- Bloquer l'accès à l'IP 169.254.169.254 au niveau du pare-feu ou du proxy inverse.

## 5. CONCLUSION ET RECOMMANDATIONS

L'audit de l'application **OWASP Juice Shop** a mis en évidence des lacunes de sécurité sévères. La présence d'une injection SQL à elle seule représente un risque de compromission totale.

- **Feuille de route recommandée :**
  1. **Immédiat** : Corriger la faille d'injection SQL et bloquer l'accès aux métadonnées cloud.
  2. **Court terme** : Configurer les headers de sécurité manquants (CSP, HSTS, X-Frame-Options).
  3. **Long terme** : Intégrer des tests de sécurité automatisés (DAST) dans le cycle de développement.