

Rapport d'audit de sécurité web

Analyse de
vulnérabilités
passive

Préparé par: Komlanvi

ANANIVI - Stagiaire

Cybersécurité

DATE: JANVIER 2026



Objectif

Identifier les failles de sécurité visibles sans impacter le fonctionnement du site.

Score global: Risque Moyen

Statistiques:

- Elevé: 1 (Vulnerable JS library)
- Moyen: 2 (Absence de jetons Anti-CSRF & Content security Policy (CSP) Header not set)
- Faible: 7 (En-têtes d'information, les cookies)

Alertes (15)

- Vulnerable JS Library
GET:https://ginandjuice.shop/resources/js/angular_1-7-7.js
- Absence de Jetons Anti-CSRF (2)
- Content Security Policy (CSP) Header Not Set (Systemic)
- Cookie No HttpOnly Flag (Systemic)
- Cookie Without Secure Flag (Systemic)
- Cookie with SameSite Attribute None (Systemic)
- Cookie without SameSite Attribute (Systemic)
- Server Leaks Version Information via "Server" HTTP Response Header
- Strict-Transport-Security Header Not Set (Systemic)
- X-Content-Type-Options Header Missing (Systemic)
- Information Disclosure - Suspicious Comments (4)
- Modern Web Application (Systemic)
- Re-examine Cache-control Directives (Systemic)
- Session Management Response Identified (27)
- User Controllable HTML Element Attribute (Potential XSS) (2)

Alertes 1 2 7 5 Main Proxy: localhost:8081

Analyse des services et composants

Vulnerable JS library

Composant détecté

Angular JS version 1.7.7

Pourquoi est-ce important ?

L'utilisateur de bibliothèques obsolètes permet à des attaquants d'exploiter des failles connues pour exécuter du code malveillant sur le navigateur des utilisateurs.

Vulnerable JS Library

URL : https://ginandjuice.shop/resources/js/angular_1-7-7.js

Risque :  High

Confiance: Medium

Paramètre :

Attaque :

Preuve : /*
AngularJS v1.7.7

Id CWE : 1395

Id WASC :

Source : Scan passif (10003 - Vulnerable JS Library (Powered by Ret

Input Vector:

Déscription:

The identified library appears to be vulnerable.

Problème

Cette version est obsolète et vulnérable

angularjs, version

Solution Pratique

Mettre à jour AngularJS vers une version supportée ou migrer vers un framework moderne (comme les dernières versions d'Angular, React ou Vue).

Quel est le problème ?

L'en-tête Content-Security-Policy est absent

Pourquoi est-ce important ?

Sans CSP, le navigateur fait confiance à n'importe quel script chargé par la page. Cela rend le site extrêmement vulnérable aux attaques XSS (Cross-Site Scripting).

Analyse des En-têtes de Sécurité (HTTP Headers)

Bloc 1: Absence de Content Security Policy (CSP)

Risque: Moyen

Cookie No HttpOnly Flag
URL : https://ginandjuice.shop/sitemap.xml
risque :  Low
Confiance : Medium
Paramètre : AWSALB
Attaque :
Preuve : Set-Cookie: AWSALB
CWE : 1004
WASC : 13
source : Scan passif (10010 - Cookie No HttpOnly Flag)
put Vector:
Déscription:
A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
Autres informations :
Solution :
Ensure that the HttpOnly flag is set for all cookies.
Référence :
https://owasp.org/www-community/HttpOnly

Solution pratique

Configurer le serveur pour envoyer un en-tête Content-Security-Policy qui définit les sources de confiance autorisées à exécuter des scripts.

Analyse des En-têtes de Sécurité (HTTP Headers)

Bloc 2: Absence de protection Anti-CSRF

Risque: Moyen 

Quel est le problème ?

Zap a détecté une “Absence de Jetons Anti-CSRF”

Pourquoi est-ce important ?

Une attaque CSRF force un utilisateur authentifié à exécuter des actions non désirées (changer un mot de passe, faire un achat) sans qu'il ne s'en rende compte.

Solution pratique

Implémenter des jetons de sécurité (tokens) uniques pour chaque session de formulaire afin de valider l'origine des requêtes.

Sécurité des cookies

Problème

Plusieurs cookies ne possèdent pas les attributs HttpOnly et Secure.

Risk Mitigation Strategies

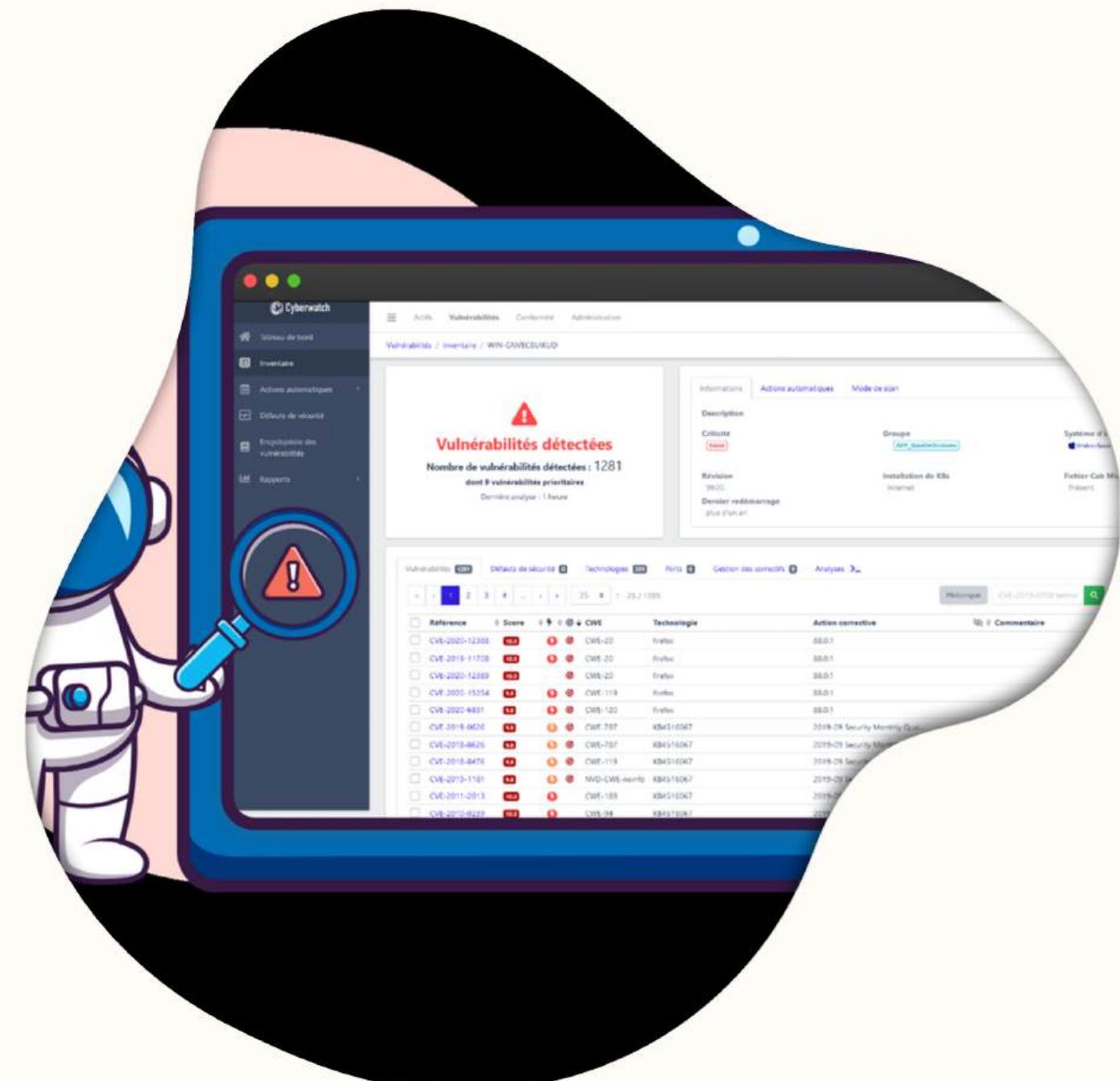
Faible (Low) 

Solution pratique

Modifier la configuration de l'application pour que tous les cookies de session incluent les drapeaux ; HttpOnly; Secure.

Pourquoi est-ce important ?

- Sans HttpOnly, un script malveillant peut voler le cookie de session de l'utilisateur.
- Sans Secure, le cookie peut être intercepté si l'utilisateur se connecte sur un réseau Wi-Fi public non sécurisé.



Conclusion

Cette analyse en lecture seule, effectuée à l'aide de l'outil OWASP ZAP, a permis d'évaluer la posture de sécurité de surface de l'application.

Bien que l'infrastructure actuelle présente des mécanismes de défense de base, l'audit a révélé des vulnérabilités critiques liées à l'utilisation de composants obsolètes (AngularJS) et à l'absence de directives de sécurité modernes dans les en-têtes HTTP (CSP, Anti-CSRF).

Principales recommandations:

1. Priorité Haute: Mettre à jour les bibliothèques JavaScripts vers des versions supportées pour éliminer les failles publiques connues (CVE).
2. Priorité Moyenne: Durcir la configuration du serveur en implémentant une politique de sécurité du contenu (CSP) et des jetons anti-CSRF pour protéger les utilisateurs finaux.
3. Hygiène de sécurité: Configurer les attributs de sécurité (HttpOnly, Secure, SameSite) sur l'ensemble des cookies de session.

En conclusion, la correction de ces points permettra de réduire considérablement la surface d'attaque et de protéger l'intégrité des données des utilisateurs face aux menaces web les plus courantes.

