Rapport : Analyse de la surface d’attaque d’un site web local avec Nikto

1. Objectif du projet

Ce mini-projet a pour but de simuler une analyse de sécurité d’un site web local en utilisant Nikto, un outil open-source de scan de vulnérabilités.
L'objectif est d’observer les failles potentielles détectées automatiquement, de les vérifier manuellement, de les compléter par d'autres outils (WhatWeb, curl, tail log Apache), et de proposer quelques pistes de remédiation basique.

2. Environnement de test

- OS : Ubuntu 22.04 (VM locale)
- Serveur web : Apache2
- Outils utilisés :
    - Nikto
    - WhatWeb
    - Nmap
    - curl
    - tail /var/log/apache2/access.log
- Interface web : http://localhost
- Application web : DVWA installée en local

3. Mise en place de la cible

Installation et démarrage du serveur Apache :

```
sudo apt install apache2
sudo systemctl start apache2
```

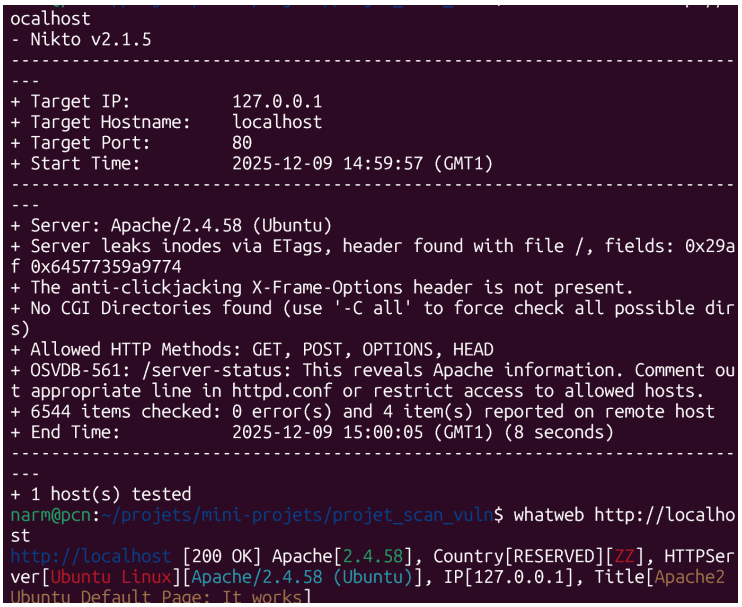
- Vérification de l'accès à `http://localhost`

4. Scan avec Nikto

Commande utilisée :

```
sudo nikto -h http://localhost
```

Résultats observés :



- **Informations serveur détectées** : Nikto a identifié le serveur comme étant Apache/2.4.58 (Ubuntu), ce qui révèle des données sur la technologie utilisée.

- **Fuite d'inodes via les ETags** : Le header `ETag` expose des identifiants de fichiers internes (inodes), ce qui peut aider un attaquant à détecter des changements de ressources ou à effectuer du fingerprinting de fichiers.

- **Header de sécurité manquant (X-Frame-Options)** : L'absence de ce header signifie que le site est vulnérable aux attaques de type clickjacking, où un utilisateur pourrait être trompé en interagissant avec un contenu malicieux masqué.

- **Page /server-status accessible** : Cette page expose des informations sensibles sur le fonctionnement d'Apache, comme les connexions actives, les requêtes en cours, et peut donc être exploitée par un attaquant. Elle devrait être restreinte ou désactivée.

- **Méthodes HTTP autorisées** : `GET, POST, OPTIONS, HEAD,` ces méthodes sont standards, mais aucune méthode dangereuse comme `TRACE` ou `PUT` n'est activée, ce qui est positif.

- **Nombre total de tests effectués** : Nikto a scanné 6544 items en 8 secondes, et a remonté 4 alertes sans générer d'erreurs.

Interprétation :
L'analyse Nikto a révélé plusieurs failles de configuration sur le serveur local, notamment l'absence de certains headers de sécurité et l'exposition de la page `/server-status`. Bien que les méthodes HTTP autorisées restent classiques, ces éléments peuvent être exploités pour du fingerprinting ou des attaques de type clickjacking.

5. Analyse complémentaire

5.1 WhatWeb

```
whatweb http://localhost
```

Résultat :

```
http://localhost [200 OK] Apache[2.4.58], Country[RESERVED][ZZ], HTTPSer
ver[Ubuntu Linux][Apache/2.4.58 (Ubuntu)], IP[127.0.0.1], Title[Apache2
Ubuntu Default Page: It works]
```

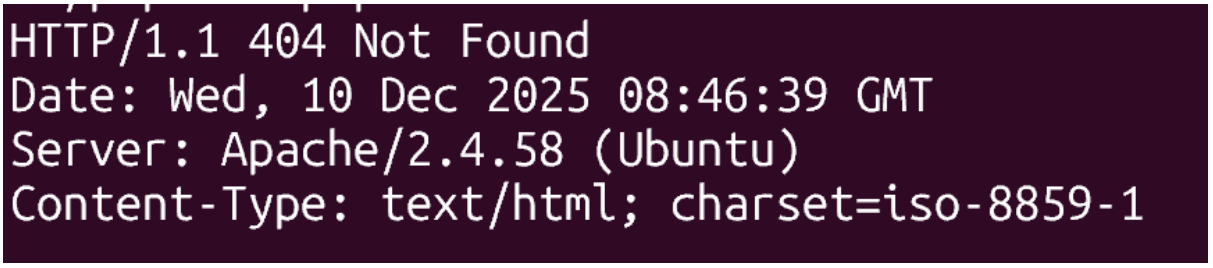
L'outil détecte que le serveur web utilisé est Apache 2.4.58, tournant sous Ubuntu Linux. Il identifie également l'adresse IP (`127.0.0.1`), la réponse HTTP (`200 OK`), et le titre de la page ("*Apache2 Ubuntu Default Page: It works*").

Cela confirme que le serveur fonctionne, mais ne révèle pas d'autres technologies web (comme PHP ou CMS), car la page par défaut ne les utilise pas.

5.2 curl

```
curl -I http://localhost/phpinfo.php
```

Résultat :



La commande retourne un code *404 Not Found*, indiquant que le fichier phpinfo.php n'existe pas ou n'est pas déployé sur le serveur.

Cependant, on observe dans les en-têtes :

```
Server: Apache/2.4.58 (Ubuntu)
Content-Type: text/html; charset=iso-8859-1
```

Certains headers de sécurité essentiels comme `X-Frame-Options` ou `X-Content-Type-Options` sont absents, ce qui représente un risque potentiel de clickjacking ou d'interprétation MIME non sécurisée si des pages sensibles étaient hébergées.

Certains headers importants pour la sécurité sont absents : X-Frame-Options, X-Content-Type-Options...

6. Vérification manuelle de vulnérabilités

À partir des résultats Nikto, les chemins suivants ont été testés :

- `http://localhost/phpinfo.php` → page très sensible affichant les infos PHP
- `http://localhost/test/` → fichier accessible sans authentification

Ces fichiers devraient être **supprimés ou restreints**.

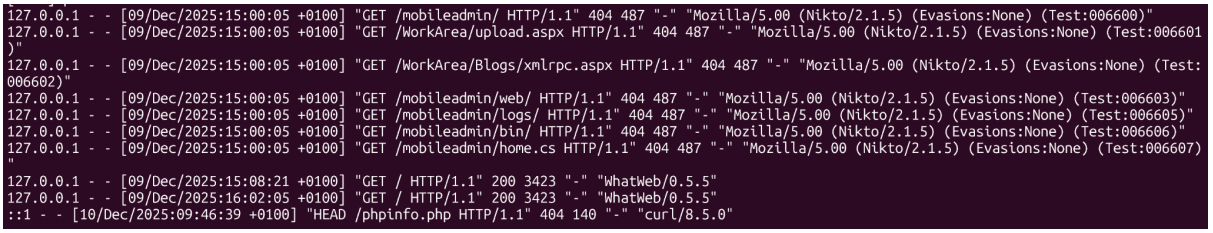
7. Surveillance réseau et journalisation

Commande :

```
sudo tail -f /var/log/apache2/access.log
```

Cela permet de **voir en temps réel** les requêtes faites par Nikto ou les tests manuels.

Résultat :



Ce qui est renvoyé montre la trace des requêtes effectuées par les scanners. On observe que Nikto a testé automatiquement plusieurs chemins sensibles (`/mobileadmin/`, `/WorkArea/`, `/logs/`, `/bin/`…), renvoyant des codes 404, signe que ces ressources n'existent pas, ce qui est plutôt rassurant car aucune interface d'administration exposée n'a été trouvée.

On voit également des identifications claires dans l'User-Agent :

`Mozilla/5.00 (Nikto/2.1.5)`

`WhatWeb/0.5.5`

`curl/8.5.0`

Cela confirme que le serveur **ne filtre pas ou n'obscurcit pas l'origine des requêtes**, ce qui faciliterait le travail d'un attaquant automatisé. Les lignes `200` montrent que la page d'accueil répond normalement, tandis que `404` indique que certaines potentielles failles recherchées par Nikto ne sont pas présentes.

9. Résumé des vulnérabilités détectées

Élément détecté	Gravité	Suggestion
`phpinfo()` exposé	Moyenne	Supprimer ou restreindre l'accès
Headers de sécurité absents	Moyenne	Ajouter `X-Frame-Options`, etc.
TRACE activé *(si détecté)*	Moyenne	Désactiver via la conf Apache
Fichiers de test accessibles	Faible	Supprimer les fichiers inutiles

10. Conclusion

Ce mini-projet a permis de :

- Découvrir et utiliser **Nikto** pour identifier des erreurs de configuration serveur.
- Compléter le scan avec **WhatWeb**, **curl**, et l'analyse des **logs Apache**.
- Visualiser concrètement **ce qu'un outil automatisé analyse** sur un serveur web.
- Comprendre l'importance des **headers HTTP** et des **bonnes pratiques de configuration Apache**.

Il constitue une bonne première étape dans un audit de sécurité web simple. Pour aller plus loin, on pourrait intégrer des tests plus poussés avec Burp Suite, OWASP ZAP, ou des outils de vulnérabilité authentifiée comme OpenVAS ou Nessus.