

Protéger un site web avec un WAF

1. Installer OWASP Juice Shop via Docker

Contexte : qu'est-ce que Juice Shop / l'image Docker

- OWASP Shop est une application web open-source volontairement “insecure” — c'est-à-dire remplie de vulnérabilités connues (injections SQL, XSS, etc.) — dans le but de servir d'environnement d'apprentissage et d'entraînement à la sécurité web.
- L'image Docker “bkimminich/juice-shop” propose une version pré-configurée de cette application, prête à être lancée dans un conteneur, ce qui simplifie énormément le déploiement (pas besoin d'installer manuellement Node.js, dépendances, config, etc.).

Contenu et configuration de l'image

Quand vous tirez l'image “bkimminich/juice-shop” (ex: via `docker pull bkimminich/juice-shop`), vous obtenez :

- L'application Juice Shop complète, avec son code, ses dépendances, et tout ce qu'il faut pour la faire fonctionner.
- Un conteneur basé sur Linux (architecture typique `linux/amd64`). Un serveur web (Node.js + framework Express) prêt à écouter sur le port `3000/tcp` à l'intérieur du conteneur
- Par défaut, le répertoire de travail du conteneur est `/juice-shop`.
- Le point d'entrée (entrypoint) du conteneur est généralement le binaire Node.js, qui lancera le script principal de l'application (par exemple `/juice-shop/build/app.js`).

En résumé : l'image inclut tout ce qu'il faut pour exécuter Juice Shop — code + runtime + configuration — et rend l'application accessible via le port 3000 sans installation supplémentaire.

Utilisation : exécution simple via Docker

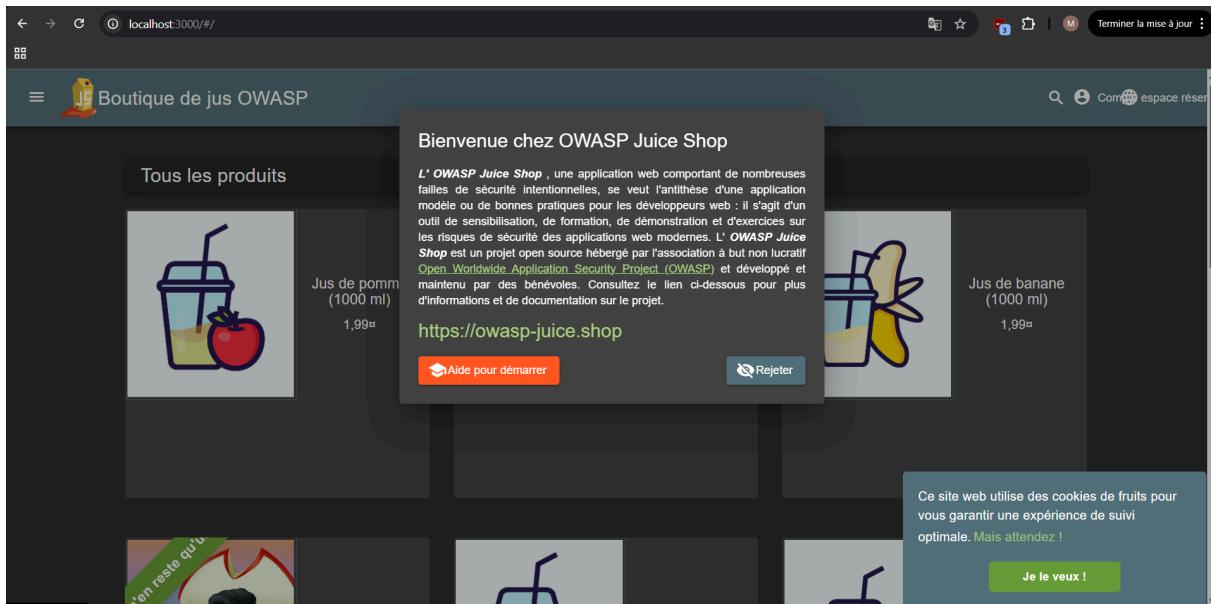
L'un des grands avantages de cette image est sa simplicité d'usage :

Après avoir installé Docker, une simple commande comme

```
docker pull bkimminich/juice-shop
```

```
docker run -d -p 3000:3000 bkimminich/juice-shop
```

- suffit pour lancer l'application.
- Ensuite, l'application est accessible dans un navigateur à l'adresse <http://localhost:3000>.



- Ceci rend le déploiement idéal pour des environnements de test, des ateliers de sécurité, des démonstrations, ou l'apprentissage de vulnérabilités web — sans la complexité d'une installation manuelle.

Pourquoi utiliser Juice Shop via Docker

Quelques raisons pour lesquelles cette image est très utilisée :

- **Praticité** : tout est pré-packagé, plus besoin d'installer manuellement Node.js, config, base de données, etc.
- **Reproductibilité** : l'image Docker garantit que tout le monde utilise la même version/configuration — utile pour ateliers, formations, CTF ou tests.
- **Séparation / isolation** : l'application tourne dans un conteneur isolé, ce qui limite les risques (utile pour tester des vulnérabilités ou attaques sans impacter le système hôte).
- **Accès rapide** : on peut lancer en quelques secondes (pull + run), plutôt que devoir cloner, installer dépendances, config, etc.

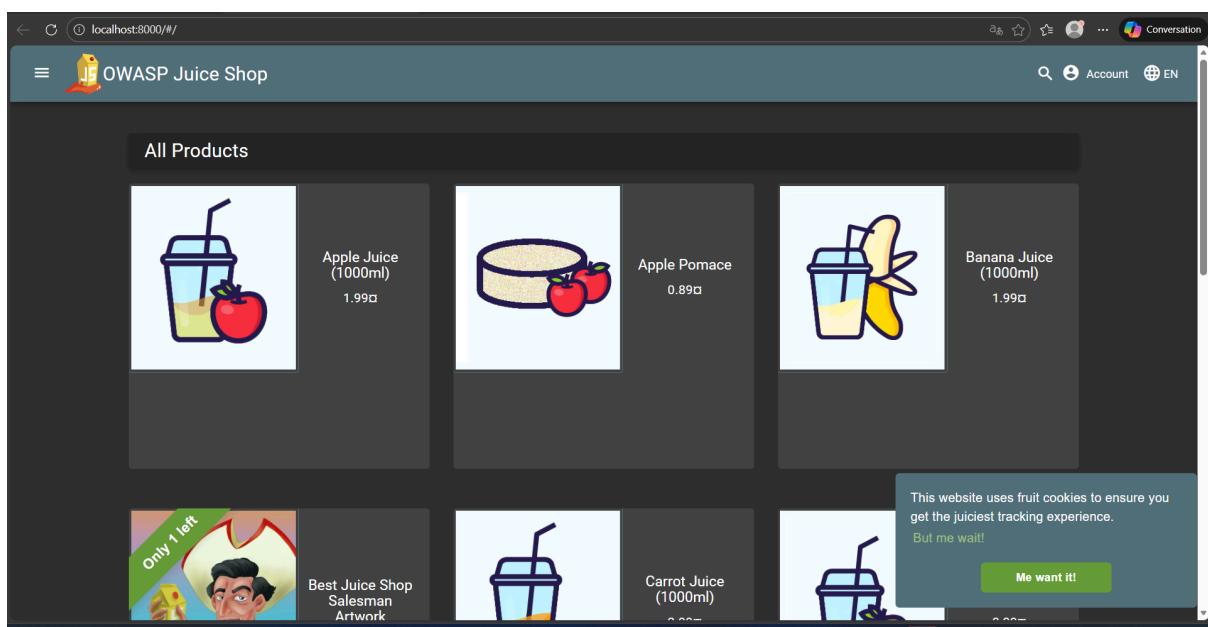
2. Configurer nginx pour agir comme un reverse proxy + WAF pour le site OWASP Juice Shop

En tant qu' administrateur Infrastructure sécurisé, tu dois mettre en place un WAF avec l'image docker : owasp/modsecurity-crs:nginx sur le port 8080 afin de ne pas exposer le url de notre site <http://juice-shop:3000>

Il faut créer un dossier WAF

docker-compose.yml a été créé. Maintenant, je vais lancer les conteneurs Docker : avec un docker-compose up -d

Dans le navigateur, il faut tapez localhost:8000



3. Tenter d'exploiter les mêmes vulnérabilités qu'au point

Cas pratique Injection SQL

A screenshot of a web browser showing the OWASP Juice Shop login page. The URL is localhost:8000/#/login. The page has a dark theme with a light gray header bar. The main content area has a title "Login" and a large error message box containing the following HTML code:

```
<html> <head><title>403 Forbidden</title></head>
<body> <center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center> </body> </html> <!-- a
padding to disable MSIE and Chrome friendly error
page --> <!-- a padding to disable MSIE and Chrome
friendly error page --> <!-- a padding to disable MSIE and Chrome
friendly error page --> <!-- a padding to disable MSIE and Chrome
friendly error page --> <!-- a padding to disable MSIE and Chrome
friendly error page --> <!-- a padding to disable MSIE and Chrome
friendly error page -->
```

The error message box is centered on the page. Below it is a form with two input fields: "Email*" and "Password*". The "Email" field contains the value "' or 1=1--". The "Password" field contains several dots and has a visibility icon. Below the form is a link "Forgot your password?". At the bottom are two buttons: "Log in" and "Remember me". To the right of the main content is a sidebar with the text "This website uses fruit cookies to ensure you get the juiciest tracking experience." and a "But me wait!" button. At the very bottom right is a green button with the text "Me want it!". The browser's address bar shows "localhost:8000" and the tab title "OWASP Juice Shop".

Cas pratique XSS

Dans la barre de recherche de navigateur : tu tape

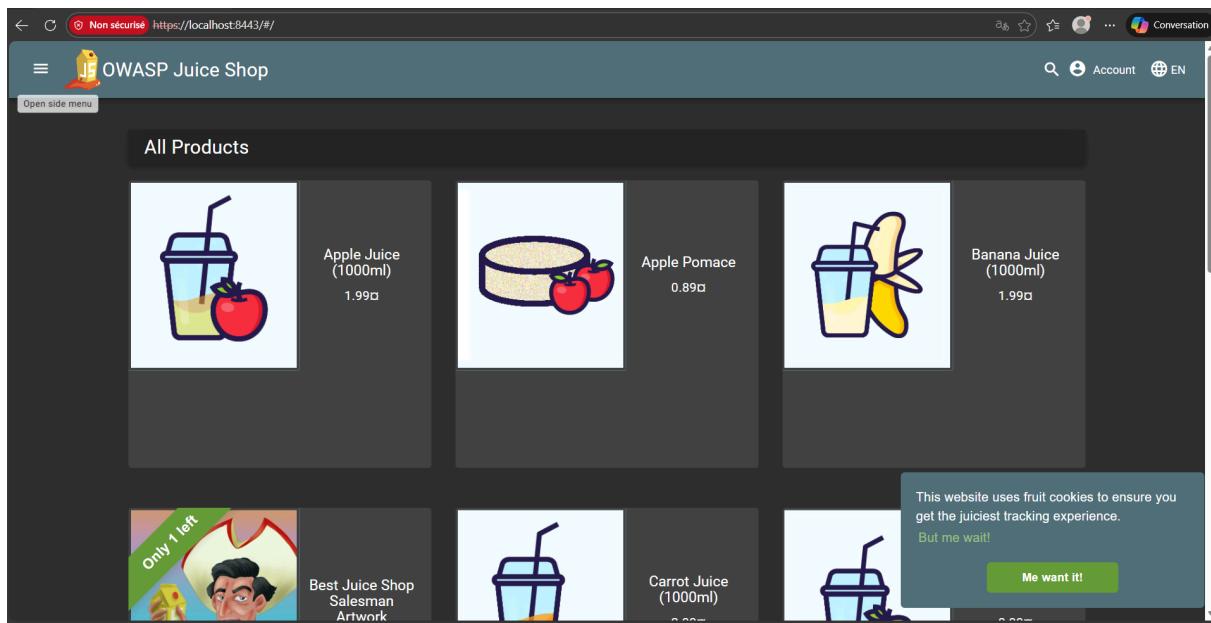
:[http://localhost:8000/#/search?q=<iframe src="javascript:alert\('xss'\)">](http://localhost:8000/#/search?q=<iframe%20src=%22javascript:alert(%27xss%27)%22>)

The screenshot shows a web browser with the URL `localhost:8000/#/search?q=<iframe%20src%3D'javascript:alert(%27%60xss%60)%27>`. The page title is "OWASP Juice Shop". A search bar at the top contains the injected XSS payload. Below the search bar, a dark header bar displays "All Products". On the right side of the header, there are buttons for "Items per page: 12", "0 of 0", and navigation arrows. At the bottom right, a green call-to-action button says "Me want it!". A light blue footer banner contains the text: "This website uses fruit cookies to ensure you get the juiciest tracking experience. But me wait!".

4. Les certificats ssl pour le WAF en utilisant les volumes de la section "Add TLS server certificate and key"

Ci joint le PROCÉDURE DE MISE EN PLACE DES CERTIFICATS SSL POUR WAF MODSECURITY dans dossier WAF-Rendu

RÉSULTAT



Visionneuse de certificats : localhost

Général Détails

Émis pour

Nom commun (CN)	localhost
Organisation (O)	MyOrg
Unité d'organisation (UO)	MyUnit

Émis par

Nom commun (CN)	localhost
Organisation (O)	MyOrg
Unité d'organisation (UO)	MyUnit

Période de validité

Date d'émission	jeudi 11 décembre 2025 à 14:14:56
Date d'expiration	vendredi 11 décembre 2026 à 14:14:56

Empreintes digitales SHA-256

Certificat	6a85badb81d0cef9fd04c290f03aa4c4f9c24db106d6eb26baabe986dee46154
Clé publique	7be2d63a97b5f56591e0f241644c82499b66c31f232ce4aa08499bfc841caadb

Bonus

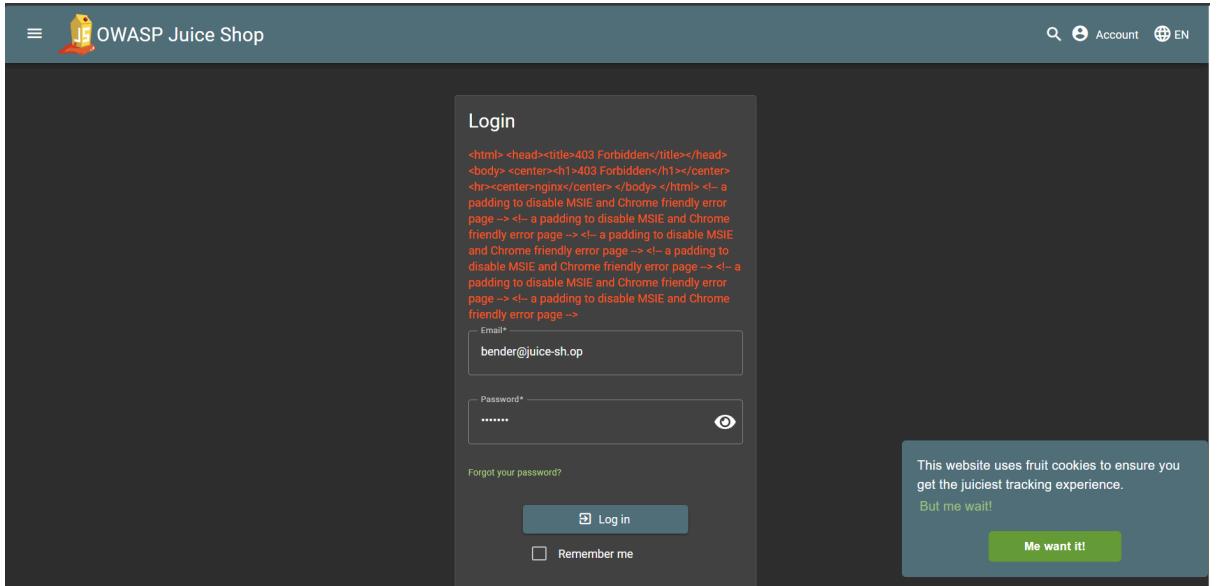
1. Créez une règle WAF empêchant un utilisateur particulier de se connecter

Dans le dossier WAF, créez un fichier et nommez le
REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf

Dans ce fichier, je crée une règle pour bloquer l'e mail bender@juice-sh.op

```
REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf
1  # =====
2  # REGLES WAF PERSONNALISEES - BLOCAGE UTILISATEUR
3  # =====
4  # Ce fichier contient des règles ModSecurity personnalisées
5  # À placer AVANT les règles CRS
6
7  #
8  # REGLE 1 : Bloquer l'utilisateur Bender
9  # -----
0  ✓ SecRule ARGS:email "@streq bender@juice-sh.op" \
1    "id:100001, \
2     phase:2, \
3     deny, \
4     status:403, \
5     log, \
6     msg:'Acces refuse - Utilisateur Bender bloque par politique de securite', \
7     tag:'custom-rule', \
8     tag:'user-block'"
9
0  #
1  # REGLE 2 : Bloquer aussi dans le corps JSON (API login)
2  # -----
3  ✓ SecRule REQUEST_BODY "@contains bender@juice-sh.op" \
4    "id:100002, \
5     phase:2, \
6     deny, \
7     status:403, \
8     log, \
9     msg:'Acces refuse - Utilisateur Bender bloque (API)', \
0     tag:'custom-rule', \
1     tag:'user-block'"
2
```

Résultat



2. Monter un volume pour le fichier auditlog (à configurer) de mod_security et l'importer dans wazuh via l'agent