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

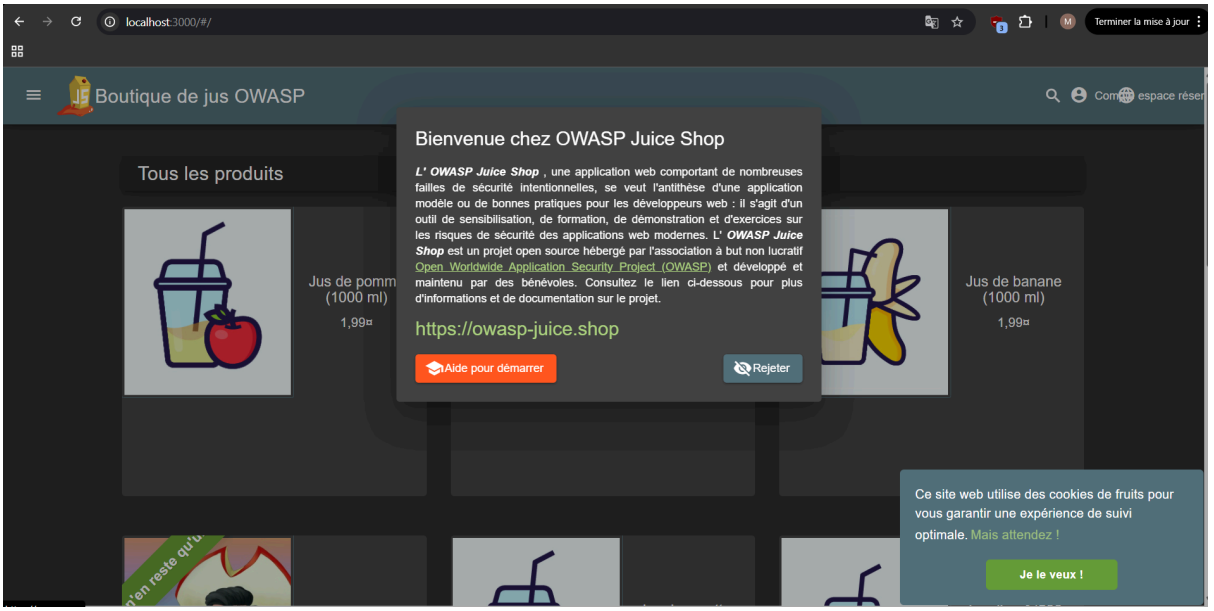
L'un des grands avantages de cette image est sa simplicité d'usage :

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

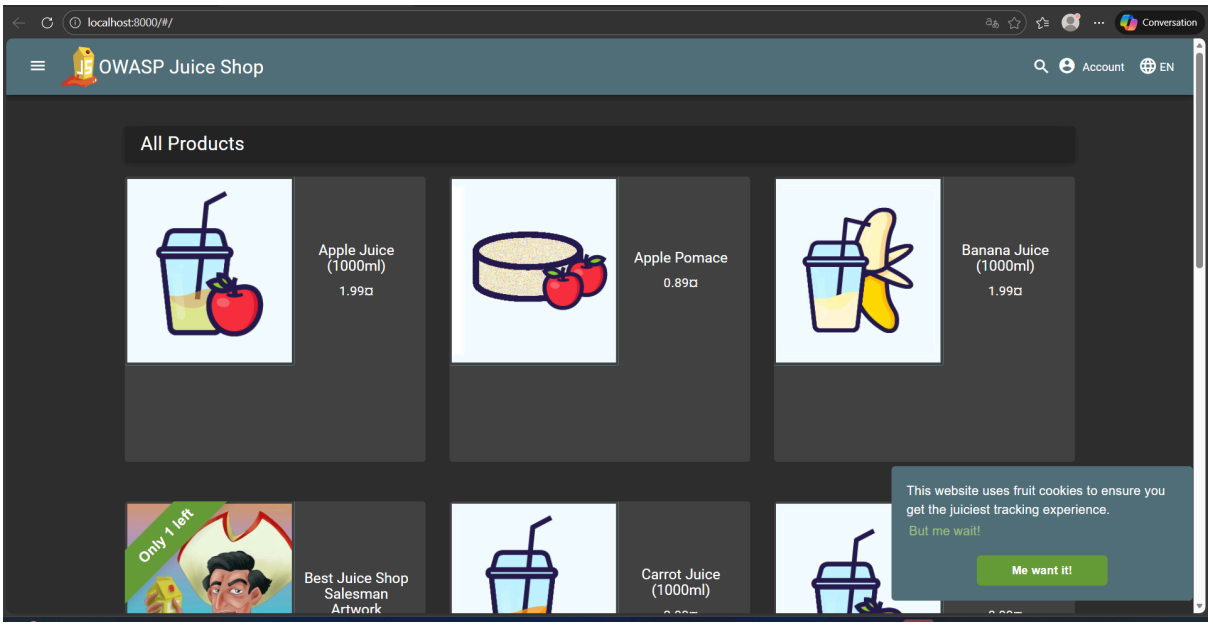
Quelques raisons pour lesquelles cette image est très utilisée :

- **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

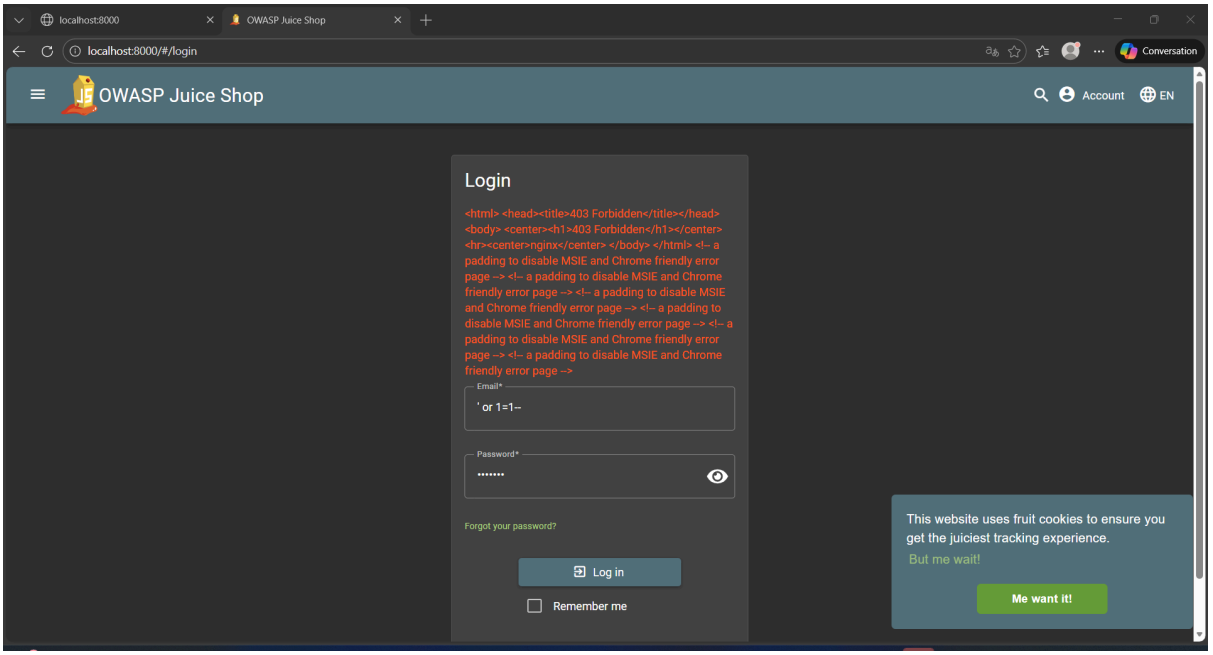
En tant qu' administrateur Infrastructure sécurisé, tu dois mettre en place un WAF avec l'image docker : owasp/modsecurity-crs:nginx sur le port 8080 afin de ne pas exposer le url de notre site http://juice-shop:3000

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

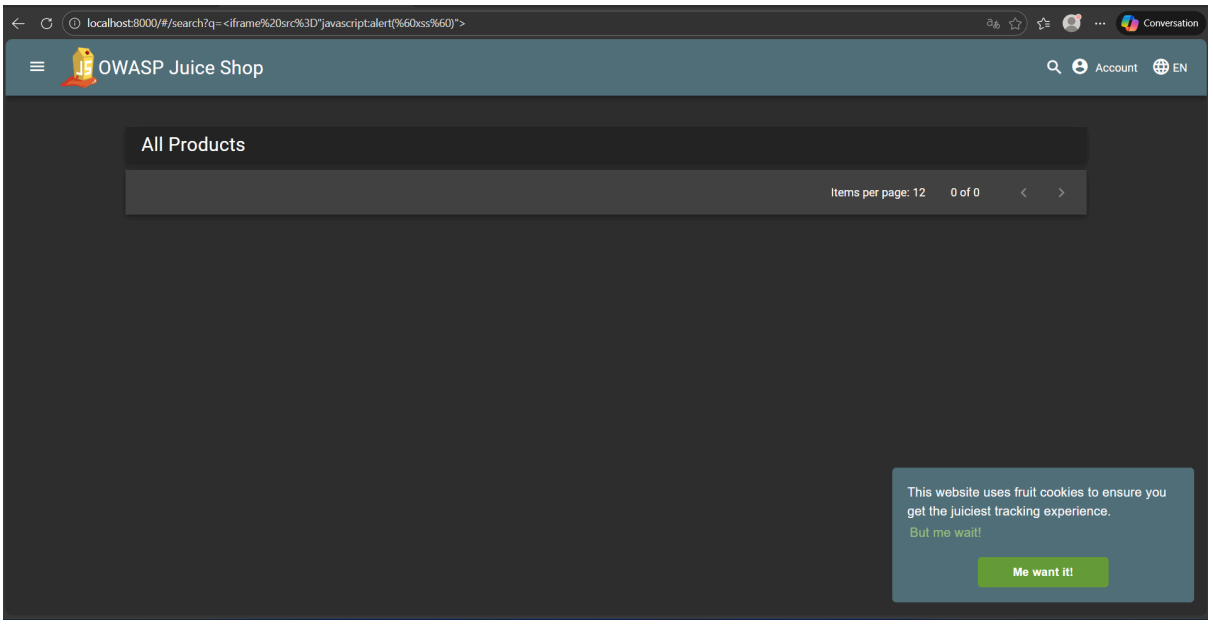
Cas pratique Injection SQL



FR;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6"}},"response":{"body":"<html>\r\n<head><title>403 Forbidden</title></head>\r\n<body>\r\n <h1>403 Forbidden</h1> \r\n<hr> nginx \r\n</body>\r\n</html>\r\n<!-- a padding to disable MSIE and Chrome friendly error page -->\r\n<!-- a padding to disable MSIE and Chrome friendly error page -->\r\n<!-- a padding to disable MSIE and Chrome friendly error page -->\r\n<!-- a padding to disable MSIE and Chrome friendly error page -->\r\n<!-- a padding to disable MSIE and Chrome friendly error page -->\r\n<!-- a padding to disable MSIE and Chrome friendly error page -->\r\n","http_code":403,"headers":{"Server":"nginx","Date":"Thu, 11 Dec 2025 12:32:58 GMT","Content-Length":"548","Content-Type":"text/html","Access-Control-Allow-Origin":"*","Connection":"keep-alive","Access-Control-Max-Age":"3600","Access-Control-Allow-Methods":"GET, POST, PUT, DELETE, OPTIONS","Access-Control-Allow-Headers":"*"}},"producer":{"modsecurity":"ModSecurity v3.0.14 (Linux)","connector":"ModSecurity-nginx

Cas pratique XSS

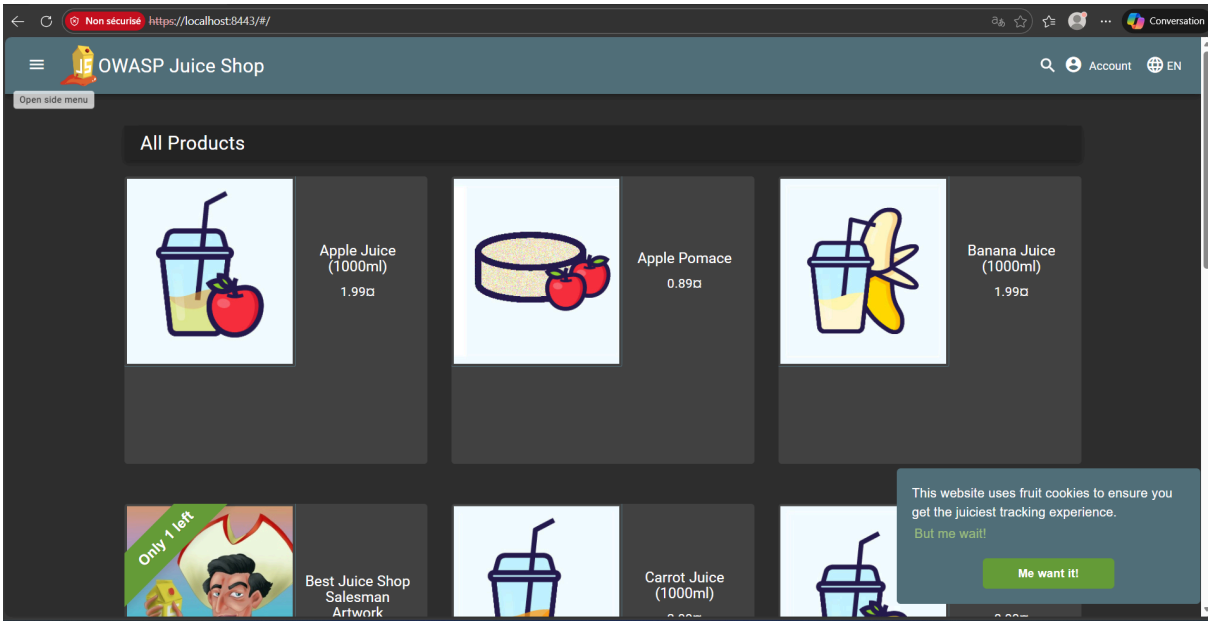
Dans la barre de recherche de navigateur : tu tape :http://localhost:8000/#/search?q=<iframe src="javascript:alert(`xss`)">

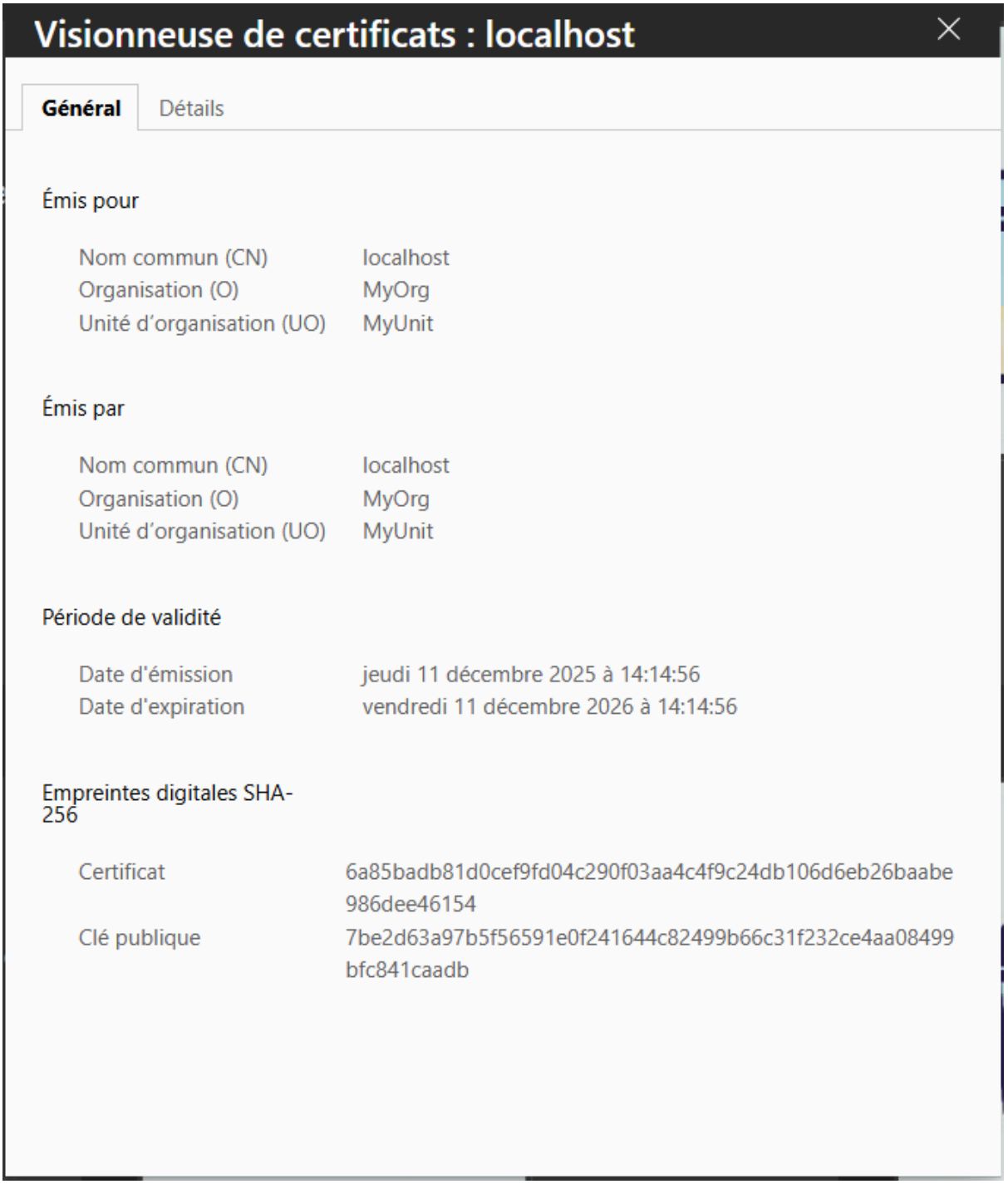


4. Les certificats ssl pour le WAF en utilisant les volumes de la section "Add TLS server certificate and key"

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

RÉSULTAT





Bonus

1. **Créer 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

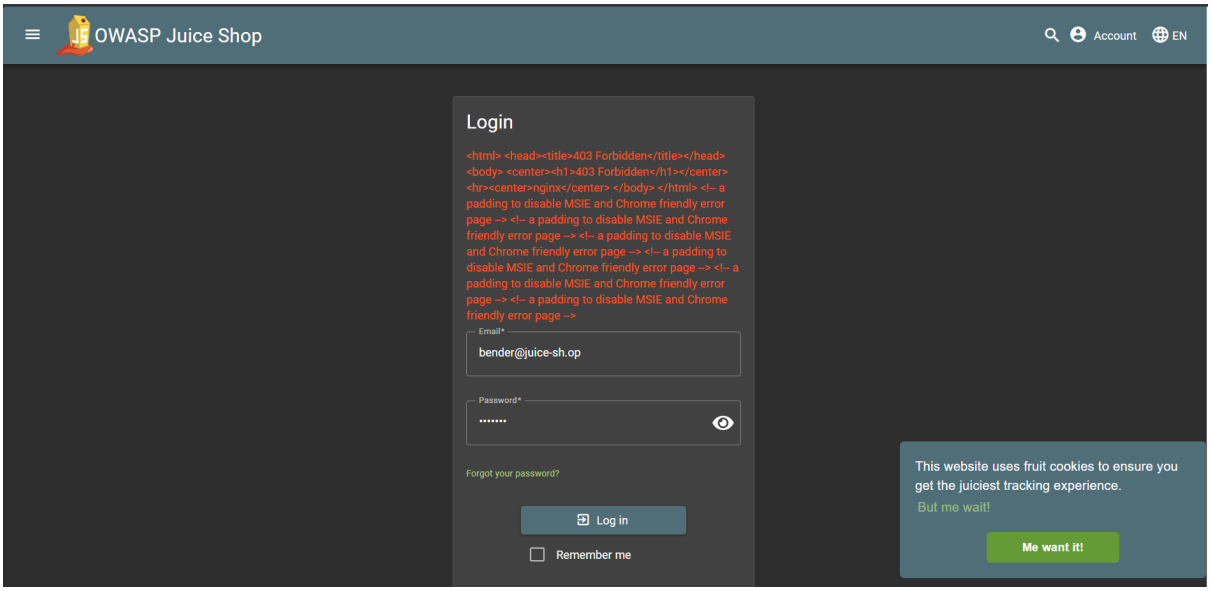
REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf

```
# ============================================================================
# REGLES WAF PERSONNALISEES - BLOCAGE UTILISATEUR
# ============================================================================
# Ce fichier contient des règles ModSecurity personnalisées
# À placer AVANT les règles CRS

# ----------------------------------------------------------------------------
# REGLE 1 : Bloquer l'utilisateur Bender
# ----------------------------------------------------------------------------
SecRule ARGS:email "@streq bender@juice-sh.op" \
    "id:100001,\
    phase:2,\
    deny,\
    status:403,\
    log,\
    msg:'Acces refuse - Utilisateur Bender bloque par politique de securite',\
    tag:'custom-rule',\
    tag:'user-block'"

# ----------------------------------------------------------------------------
# REGLE 2 : Bloquer aussi dans le corps JSON (API login)
# ----------------------------------------------------------------------------
SecRule REQUEST_BODY "@contains bender@juice-sh.op" \
    "id:100002,\
    phase:2,\
    deny,\
    status:403,\
    log,\
    msg:'Acces refuse - Utilisateur Bender bloque (API)',\
    tag:'custom-rule',\
    tag:'user-block'"
```


Résultat



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