PROCÉDURE DE MISE EN PLACE DES CERTIFICATS SSL POUR WAF MODSECURITY

Guide Complet avec Explications

Document : Procedure_Certificat_SSL_Complete.md
Version : 1.0
Date : 11/12/2025
Auteur : Mouhamadou SALL

📚 TABLE DES MATIÈRES

---

1. OBJECTIF ET CONTEXTE

🎯 Objectif

Cette procédure décrit les étapes pour générer et configurer des certificats SSL/TLS auto-signés pour le WAF ModSecurity CRS sur Docker.

📖 Contexte et Explication

Qu'est-ce qu'un certificat SSL/TLS ?

- C'est comme une carte d'identité numérique pour votre serveur web

- Il permet de chiffrer les communications entre le navigateur et le serveur
- Il garantit que personne ne peut intercepter ou lire les données transmises

Pourquoi ajouter HTTPS au WAF ?

Sans SSL/TLS (HTTP) :

Navigateur → [données en clair] → WAF → Application
↑ Lisible par n'importe qui

Avec SSL/TLS (HTTPS) :

Navigateur → [données chiffrées] → WAF → Application
↑ Illisible sans la clé privée

Architecture actuelle VS future :

Avant	Après
HTTP seulement (port 8000)	HTTP (port 8000) + HTTPS (port 8443)
Données non chiffrées	Données chiffrées en HTTPS
Pas de certificat	Certificat SSL auto-signé

---

2. PRÉREQUIS

✅ Éléments nécessaires

- **Docker Desktop** : installé et fonctionnel
- **Accès administrateur** : au système Windows/Linux/Mac
- **docker-compose.yml** : du WAF déjà configuré
- **Connexion Internet** : pour télécharger l'image OpenSSL (première fois)

🔍 Vérification des prérequis

Ouvrir PowerShell ou Terminal et exécuter :

```
# Vérifier Docker
docker --version

# Vérifier Docker Compose
docker compose version

# Vérifier que le WAF est dans le bon répertoire
```

cd c:\Users\Utilisateur\Desktop\Formation\Docker\WAF
dir

Résultat attendu : Vous devez voir le fichier `docker-compose.yml`

---

3. GÉNÉRATION DES CERTIFICATS

📝 Vue d'ensemble

Cette étape crée deux fichiers essentiels :

- **server.key** : Clé privée (à garder secrète, comme un mot de passe)
- **server.crt** : Certificat public (peut être partagé)

Analogie : C'est comme créer une paire de clés pour votre maison :

- La clé privée (server.key) = votre clé physique (ne la donnez jamais)
- Le certificat (server.crt) = votre serrure (visible par tous)

ÉTAPE 3.1 : Ouvrir PowerShell en tant qu'administrateur

Windows :

1. Clic droit sur le menu Démarrer
2. Sélectionner "Windows PowerShell (Admin)" ou "Terminal (Admin)"
3. Accepter l'UAC (Contrôle de compte utilisateur)

Pourquoi administrateur ? Pour avoir les permissions d'écriture dans tous les dossiers.

ÉTAPE 3.2 : Se placer dans le répertoire du WAF

cd c:\Users\Utilisateur\Desktop\Formation\Docker\WAF

💡 **Astuce :** Adaptez ce chemin selon votre installation. Vous pouvez faire un clic droit dans l'explorateur de fichiers et choisir "Copier le chemin d'accès".

Vérification :

pwd # Affiche le répertoire actuel
dir # Liste les fichiers (vous devez voir docker-compose.yml)

ÉTAPE 3.3 : Générer le certificat et la clé

🖥️ Pour Windows (PowerShell)

```
docker run --rm -v "${PWD}:/certs" alpine/openssl req -x509 -nodes ^
  -days 365 -newkey rsa:2048 ^
  -keyout /certs/server.key ^
  -out /certs/server.crt ^
  -subj "/C=FR/ST=France/L=Paris/O=Formation/OU=WAF/CN=localhost"
```

🐧 Pour Linux/Mac (Terminal)

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
  -keyout server.key -out server.crt \
  -subj "/C=FR/ST=France/L=Paris/O=Formation/OU=WAF/CN=localhost"
```

📚 Explication détaillée de la commande

Paramètre	Signification	Explication
`docker run --rm`	Exécuter puis supprimer	Lance un conteneur temporaire qui se supprime après usage
`-v "${PWD}:/certs"`	Volume monté	Connecte votre dossier actuel au conteneur pour sauvegarder les fichiers
`alpine/openssl`	Image Docker	Petit système Linux avec OpenSSL préinstallé
`req`	Request	Commande pour créer une demande de certificat
`-x509`	Format X.509	Standard international pour les certificats numériques
`-nodes`	No DES	Pas de chiffrement de la clé privée (pas de mot de passe)
`-days 365`	Validité	Le certificat sera valide pendant 1 an (365 jours)
`-newkey rsa:2048`	Nouvelle clé RSA	Créer une clé de 2048 bits (niveau de sécurité standard)
`-keyout`	Fichier de sortie clé	Où sauvegarder la clé privée

`-out`	Fichier de sortie cert	Où sauvegarder le certificat
`-subj`	Subject	Informations d'identification du certificat

🏷️ Détail du Subject (-subj)

/C=FR → Country (Pays) : France
/ST=France → State (État/Région) : France
/L=Paris → Locality (Ville) : Paris
/O=Formation → Organization (Organisation) : Formation
/OU=WAF → Organizational Unit (Département) : WAF
/CN=localhost → Common Name (Nom du serveur) : localhost

⚠️ **Important :** Le `CN=localhost` doit correspondre au nom de domaine utilisé. Pour un serveur accessible via une IP ou un autre nom, modifiez cette valeur.

⏱️ Temps d'exécution

- Première fois : 30-60 secondes (téléchargement de l'image alpine/openssl)
- Fois suivantes : 2-5 secondes

ÉTAPE 3.4 : Vérifier la création des fichiers

Windows
dir *.crt, *.key

Linux/Mac
ls -lh *.crt *.key

✅ Résultat attendu :

server.crt (environ 1-2 Ko) - Certificat public
server.key (environ 1-2 Ko) - Clé privée

🔍 Vérification avancée :

Afficher les informations du certificat
```
docker run --rm -v "${PWD}:/certs" alpine/openssl x509 -in /certs/server.crt -text -noout
```

Vous devriez voir :

- La validité (Not Before / Not After)

- Le sujet (Subject: C=FR, ST=France, etc.)
- La clé publique (Public-Key: 2048 bit)

---

4. CONFIGURATION DU DOCKER-COMPOSE

📝 Vue d'ensemble

Nous allons modifier le fichier `docker-compose.yml` pour :

1. Ouvrir le port HTTPS (8443)
2. Monter les certificats dans le conteneur
3. Configurer les variables d'environnement SSL

🎯 Objectif

Permettre au conteneur WAF d'utiliser les certificats générés pour activer HTTPS.

ÉTAPE 4.1 : Ouvrir le fichier docker-compose.yml

Avec un éditeur de texte :

- Visual Studio Code (recommandé)
- Notepad++
- Bloc-notes Windows (éviter, problèmes d'encodage possibles)

```
# Ouvrir avec VS Code
code docker-compose.yml

# Ou avec le bloc-notes
notepad docker-compose.yml
```

ÉTAPE 4.2 : Ajouter le port HTTPS

🔍 **Localiser la section :** Cherchez la section `ports:` sous le service WAF.

❌ **Configuration actuelle :**

```
services:
  waf:
    image: owasp/modsecurity-crs:nginx
    ports:
      - "8000:8080"    # HTTP uniquement
```

✅ **Configuration modifiée :**

```
services:
  waf:
    image: owasp/modsecurity-crs:nginx
    ports:
      - "8000:8080"    # HTTP
      - "8443:8443"    # HTTPS (nouveau)
```

📚 **Explication :**

- `8000:8080` signifie : port 8000 sur votre PC → port 8080 dans le conteneur
- `8443:8443` signifie : port 8443 sur votre PC → port 8443 dans le conteneur
- Le port 8443 est choisi car 443 (port HTTPS standard) peut nécessiter des droits administrateur

---

ÉTAPE 4.3 : Ajouter les volumes pour les certificats

🔍 **Localiser la section :** Cherchez la section `volumes:` ou créez-la si elle n'existe pas.

✅ **Configuration à ajouter :**

```
  volumes:
    - ./server.crt:/etc/nginx/certs/server.crt:ro
    - ./server.key:/etc/nginx/certs/server.key:ro
```

📚 **Explication détaillée :**

Partie	Signification
`./server.crt`	Fichier sur votre PC (répertoire actuel)
`:`	Séparateur
`/etc/nginx/certs/server.crt`	Chemin dans le conteneur Docker
`:ro`	Read-Only (lecture seule) - sécurité renforcée

Pourquoi `:ro` ?

- Empêche le conteneur de modifier les certificats

- Bonne pratique de sécurité
- Évite les modifications accidentelles

Analogie : C'est comme prêter un livre en disant "tu peux le lire mais pas écrire dedans".

ÉTAPE 4.4 : Ajouter les variables d'environnement SSL

🔍 **Localiser la section :** Cherchez la section `environment:` ou créez-la.

✅ **Configuration à ajouter :**

```
  environment:
    SSL_ENGINE: "on"
    SSL_PORT: 8443
    SSL_CERT: "/etc/nginx/certs/server.crt"
    SSL_CERT_KEY: "/etc/nginx/certs/server.key"
```

📚 **Explication de chaque variable :**

Variable	Valeur	Rôle
`SSL_ENGINE`	`"on"`	Active le moteur SSL dans Nginx
`SSL_PORT`	`8443`	Port d'écoute pour HTTPS
`SSL_CERT`	Chemin du certificat	Indique où se trouve le certificat public
`SSL_CERT_KEY`	Chemin de la clé	Indique où se trouve la clé privée

📄 Exemple de docker-compose.yml complet

```
version: '3.8'

services:
  waf:
    image: owasp/modsecurity-crs:nginx
    container_name: waf-modsecurity
    ports:
      - "8000:8080"    # HTTP
      - "8443:8443"    # HTTPS
    volumes:
      - ./server.crt:/etc/nginx/certs/server.crt:ro
      - ./server.key:/etc/nginx/certs/server.key:ro
```

```
    environment:
      BACKEND: "http://juiceshop:3000"
      SSL_ENGINE: "on"
      SSL_PORT: 8443
      SSL_CERT: "/etc/nginx/certs/server.crt"
      SSL_CERT_KEY: "/etc/nginx/certs/server.key"
    depends_on:
      - juiceshop

  juiceshop:
    image: bkimminich/juice-shop
    container_name: juice-shop
    ports:
      - "3000:3000"
```

💾 **Sauvegarder le fichier** : `Ctrl + S` (VS Code) ou Fichier → Enregistrer

---

5. APPLICATION DE LA CONFIGURATION

📝 Vue d'ensemble

Nous allons redémarrer le WAF pour qu'il prenne en compte les nouveaux certificats et la configuration HTTPS.

ÉTAPE 5.1 : Arrêter le conteneur WAF existant

docker compose down

📚 Que fait cette commande ?

- Arrête tous les conteneurs définis dans docker-compose.yml
- Supprime les conteneurs (mais pas les volumes ou images)
- Nettoie le réseau Docker créé

⏱️ **Temps d'exécution :** 5-10 secondes

✅ **Résultat attendu :**

```
[+] Running 3/3
 ✔ Container waf-modsecurity  Removed
 ✔ Container juice-shop       Removed
 ✔ Network waf_default        Removed
```

ÉTAPE 5.2 : Redémarrer avec la nouvelle configuration

docker compose up -d

📚 **Explication des options :**

- `up` : Démarre les services
- `-d` : Mode détaché (daemon) - s'exécute en arrière-plan

⏱️ **Temps d'exécution :** 10-20 secondes

✅ **Résultat attendu :**

[+] Running 3/3
✔ Network waf_default Created
✔ Container juice-shop Started
✔ Container waf-modsecurity Started

⚠️ **Si vous voyez des erreurs :**

- Vérifiez la syntaxe YAML (indentation !)
- Assurez-vous que les fichiers .crt et .key existent
- Consultez la section Dépannage

ÉTAPE 5.3 : Vérifier que le conteneur est en cours d'exécution

docker ps

✅ **Résultat attendu :**

CONTAINER ID IMAGE STATUS PORTS
abc123def456 owasp/modsecurity-crs:nginx Up 30 seconds 0.0.0.0:8000->8080/tcp, 0.0.0.0:8443->8443/tcp
xyz789ghi012 bkimminich/juice-shop Up 30 seconds 0.0.0.0:3000->3000/tcp

🔍 **Points à vérifier :**

- STATUS = "Up" (et non "Restarting" ou "Exited")
- Les deux ports 8000 et 8443 sont listés
- Le conteneur tourne depuis au moins quelques secondes

🔧 **Commandes utiles :**

Voir les logs du WAF
docker logs waf-modsecurity

Voir les logs en temps réel
docker logs -f waf-modsecurity

Voir les logs des 50 dernières lignes
docker logs --tail 50 waf-modsecurity

---

6. VÉRIFICATION

📝 Vue d'ensemble

Cette section vous guide pour vérifier que tout fonctionne correctement, étape par étape.

ÉTAPE 6.1 : Vérifier que les ports sont en écoute

Windows (PowerShell)
netstat -an | findstr "8000 8443"

Linux/Mac (Terminal)
```
netstat -an | grep "8000\|8443"
# ou
lsof -i :8000,8443
```

✅ Résultat attendu :

```
TCP    0.0.0.0:8000           0.0.0.0:0              LISTENING
TCP    0.0.0.0:8443           0.0.0.0:0              LISTENING
TCP    [::]:8000              [::]:0                 LISTENING
TCP    [::]:8443              [::]:0                 LISTENING
```

📚 Explication :

- `0.0.0.0` = écoute sur toutes les interfaces réseau
- `LISTENING` = le port attend des connexions
- Les lignes avec `[::]` sont pour IPv6

❌ Si rien ne s'affiche :

- Le conteneur ne s'est pas lancé correctement

- Vérifiez les logs : `docker logs waf-modsecurity`
- Assurez-vous qu'aucune autre application n'utilise ces ports

---

ÉTAPE 6.2 : Tester l'accès HTTP

Dans le navigateur

1. Ouvrez votre navigateur (Chrome, Firefox, Edge)
2. Tapez l'URL : `http://localhost:8000`
3. Appuyez sur Entrée

✅ **Résultat attendu :**

- La page d'accueil de Juice Shop s'affiche
- L'URL reste en `http://` (pas de redirection automatique vers HTTPS)
- Pas d'avertissement de sécurité

🎨 **Ce que vous devriez voir :**

- Logo Juice Shop
- Liste de produits (jus de fruits)
- Barre de recherche
- Panier d'achat

Test en ligne de commande (optionnel)

```
# Test simple
curl http://localhost:8000

# Test avec en-têtes
curl -I http://localhost:8000
```

✅ **Résultat attendu :** Code HTTP 200 OK

ÉTAPE 6.3 : Tester l'accès HTTPS

Dans le navigateur

1. Ouvrez votre navigateur
2. Tapez l'URL : `https://localhost:8443`
3. Appuyez sur Entrée

⚠️ **ATTENTION - Avertissement de sécurité attendu :**

Vous allez voir un écran d'avertissement. C'est **NORMAL** et **Attendu** car le certificat est auto-signé.

Chrome / Edge

Votre connexion n'est pas privée
NET::ERR_CERT_AUTHORITY_INVALID

Actions à effectuer :

1. Cliquez sur **"Avancé"** (ou "Advanced")
2. Cliquez sur **"Continuer vers localhost (non sécurisé)"**

Firefox

Attention : risque probable de sécurité

Actions à effectuer :

1. Cliquez sur **"Avancé..."**
2. Cliquez sur **"Accepter le risque et poursuivre"**

Safari (Mac)

Ce site pourrait ne pas être sécurisé

Actions à effectuer :

1. Cliquez sur **"Afficher les détails"**
2. Cliquez sur **"visiter ce site web"**
3. Confirmez avec votre mot de passe si demandé

📚 Pourquoi cet avertissement ?

Analogie de la carte d'identité :

Type de certificat	Équivalent réel
Certificat d'une CA reconnue	Passeport officiel délivré par l'État
Certificat auto-signé	Carte d'identité que vous vous êtes faite vous-même

Le chiffrement fonctionne parfaitement dans les deux cas, mais :

- Un certificat auto-signé n'a pas été vérifié par une autorité de confiance
- Le navigateur ne peut pas confirmer l'identité du serveur
- C'est acceptable pour le développement/test, mais pas pour la production

✅ **Après avoir accepté l'exception :**

- La page Juice Shop s'affiche en HTTPS
- Un cadenas barré ou un triangle d'avertissement apparaît dans la barre d'adresse
- Les données sont quand même chiffrées !

Test en ligne de commande (optionnel)

Test en ignorant la vérification du certificat
```
curl -k https://localhost:8443
```

Test avec détails SSL
```
curl -kv https://localhost:8443
```

Option `-k` : Ignore la vérification du certificat (comme "accepter le risque" dans le navigateur)

ÉTAPE 6.4 : Vérifier le chiffrement SSL (optionnel mais instructif)

Voir les détails du certificat dans le navigateur

Chrome / Edge :

1. Cliquez sur le cadenas (ou triangle) dans la barre d'adresse
2. Cliquez sur "Le certificat n'est pas valide"
3. Onglet "Détails" : vous verrez toutes les informations

Firefox :

1. Cliquez sur le cadenas
2. Connexion non sécurisée → Plus d'informations
3. Onglet "Sécurité" → Voir le certificat

Ce que vous devriez voir :

- Émis pour : localhost
- Émis par : localhost (auto-signé)
- Valide du : date d'aujourd'hui
- Valide jusqu'au : dans 365 jours
- Clé publique : RSA 2048 bits

Test SSL avec OpenSSL (avancé)

```
docker run --rm alpine/openssl s_client -connect localhost:8443 -showcerts
```

À chercher dans la sortie :

- `Verify return code: 18 (self signed certificate)` ← Normal !
- `Protocol : TLSv1.2` ou `TLSv1.3`
- `Cipher : ...` (algorithme de chiffrement utilisé)

---

7. URLS D'ACCÈS

📋 Tableau récapitulatif

Protocole	URL	Usage	Certificat	WAF
HTTP	http://localhost:8000	Accès standard sans chiffrement	❌ Aucun	✅ Activé
HTTPS	https://localhost:8443	Accès sécurisé avec chiffrement	✅ Auto-signé	✅ Activé
HTTP Direct	http://localhost:3000	Juice Shop sans protection WAF	❌ Aucun	❌ Désactivé

🔄 Flux des requêtes

Accès via HTTP (port 8000)

Navigateur → [HTTP non chiffré] → WAF (analyse + filtre) → Juice Shop
Port 8000

Accès via HTTPS (port 8443)

Navigateur → [HTTPS chiffré] → WAF (déchiffre → analyse + filtre → rechiffre) → Juice Shop
Port 8443 ↑ Couche de sécurité

Accès direct (port 3000)

Navigateur → [HTTP non chiffré] → Juice Shop directement
Port 3000 ↑ AUCUNE PROTECTION

💡 Quand utiliser quelle URL ?

Situation	URL recommandée	Raison
Tests de base	http://localhost:8000	Plus simple, pas d'avertissement

Tests de sécurité HTTPS	https://localhost:8443	Simule un environnement sécurisé
Tests de WAF	http://localhost:8000 ou https://localhost:8443	Les deux passent par le WAF
Tests sans WAF	http://localhost:3000	Pour comparer avec/sans protection
Démonstration à d'autres	https://localhost:8443	Plus professionnel

🌐 Accès depuis d'autres machines (réseau local)

Si vous voulez accéder au WAF depuis un autre ordinateur sur votre réseau :

1. **Trouver votre adresse IP locale :**

```
# Windows
ipconfig

# Linux/Mac
ifconfig
# ou
ip addr
```

Cherchez une adresse comme `192.168.x.x` ou `10.0.x.x`

2. **Utiliser cette IP sur l'autre machine :**

```
http://192.168.1.100:8000    (remplacez par votre IP)
https://192.168.1.100:8443
```

⚠️ **Important :**

- Le certificat étant créé pour "localhost", vous aurez un avertissement supplémentaire
- Vérifiez que votre pare-feu Windows autorise les connexions sur ces ports

---

8. DÉPANNAGE

🔧 Problèmes courants et solutions

PROBLÈME 1 : Le conteneur ne démarre pas

Symptôme :

docker ps
Le conteneur n'apparaît pas ou STATUS = "Exited"

🔍 **Diagnostic :**

Voir les logs d'erreur
docker logs waf-modsecurity

Voir si le conteneur a existé
docker ps -a

✅ **Solutions possibles :**

A. Erreur de syntaxe YAML

Vérifier l'indentation :

❌ INCORRECT (mauvaise indentation)
services:
waf:
  image: owasp/modsecurity-crs:nginx

✅ CORRECT (indentation de 2 espaces)
services:
  waf:
    image: owasp/modsecurity-crs:nginx

Outil de validation : https://www.yamllint.com/

B. Fichiers certificats manquants

Vérifier la présence des fichiers
dir server.crt, server.key

Si absents, régénérer (voir étape 3.3)
docker run --rm -v "${PWD}:/certs" alpine/openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /certs/server.key -out /certs/server.crt -subj "/C=FR/ST=France/L=Paris/O=Formation/OU=WAF/CN=localhost"

C. Ports déjà utilisés

Vérifier si les ports sont occupés :

netstat -ano | findstr "8000 8443"

Si occupés, libérer ou changer les ports :

```
# Changer 8000 en 8001 et 8443 en 8444 par exemple
ports:
  - "8001:8080"
  - "8444:8443"
```

PROBLÈME 2 : Erreur "connection refused" sur le port 8443

Symptôme :

ERR_CONNECTION_REFUSED
ou
Connection refused

🔍 **Diagnostic :**

```
# Vérifier que le port est en écoute
netstat -an | findstr "8443"

# Vérifier les logs
docker logs waf-modsecurity | findstr "SSL"
```

✅ **Solutions :**

A. SSL_ENGINE n'est pas activé

Vérifier dans docker-compose.yml :

```
environment:
  SSL_ENGINE: "on"    # ← DOIT être "on" (avec guillemets)
```

Redémarrer après modification :

```
docker compose down
docker compose up -d
```

B. Mauvais chemin vers les certificats

Dans docker-compose.yml, vérifier :

```
environment:
  SSL_CERT: "/etc/nginx/certs/server.crt"      # ← Chemin exact
  SSL_CERT_KEY: "/etc/nginx/certs/server.key"  # ← Chemin exact
```

volumes:
  - ./server.crt:/etc/nginx/certs/server.crt:ro   # ← Cohérence
  - ./server.key:/etc/nginx/certs/server.key:ro

C. Nginx n'a pas redémarré correctement

Forcer le redémarrage
docker compose restart waf

Ou reconstruire complètement
docker compose down
docker compose up -d --force-recreate

PROBLÈME 3 : Erreur de certificat persistante dans le navigateur

Symptôme :

- Même après avoir accepté l'exception, l'erreur persiste
- ERR_CERT_COMMON_NAME_INVALID
- ERR_CERT_DATE