Procédure exploitation
MODSEC_RULE_ENGINES

Activer ModSecurity (surtout avec l'OWASP Core Rule Set) directement en mode "On" est suicidaire pour une application en production. Cette procédure repose sur deux concepts clés :

1. L'approche "Detection First" (Phases 1-3) :
   Au lieu de bloquer, on écoute. C'est comme installer une caméra de surveillance avant d'engager un vigile. On regarde qui entre et sort pour distinguer les clients légitimes des voleurs. Cela permet de "whitelister" (exclure) les comportements normaux de votre application qui pourraient être confondus avec des attaques (faux positifs).
2. L'approche "Entonnoir" (Phase 4) :
   C'est la partie la plus astucieuse de votre procédure. Au lieu de passer de "Rien ne bloque" à "Tout bloque", vous jouez sur le Score d'Anomalie (ANOMALY_INBOUND).
   - **Score 100 (Début) :** On ne bloque que les attaques très bruyantes et évidentes.
   - Score 5 (Fin) : On bloque dès qu'une petite règle stricte est enfreinte.
     Cela permet d'habituer le système progressivement.

📄 **Document Officiel : Procédure d'Exploitation**

Voici le document remis en forme pour une lisibilité maximale.

Procédure d'Exploitation : Activation Progressive ModSecurity

Méta-donnée	Valeur
Composant	MODSEC_RULE_ENGINE
Version	1.0
Date	11/12/2025
Auteur	Équipe Sécurité
Classification	🔒 Interne

---

1. Objectif

Cette procédure décrit la méthodologie standard pour activer le moteur de règles ModSecurity en production. L'objectif est d'atteindre un niveau de sécurité optimal tout en garantissant **zéro interruption de service** pour les utilisateurs légitimes (minimisation des faux positifs).

2. Prérequis

- [ ] Accès root ou sudo au serveur/conteneur WAF.
- [ ] Accès confirmé aux logs d'audit (/var/log/modsec_audit.log ou sortie Docker).
- [ ] Application backend stable.
- [ ] Fenêtre de maintenance ou période de faible trafic identifiée pour les bascules.
- [ ] **Plan de rollback** (Section 5) lu et compris.

---

3. Comprendre les Modes (Engine Modes)

Mode	Description Technique	Impact Métier
Off	Moteur éteint.	Aucune protection.
DetectionOnly	Traite les règles, écrit les logs, mais **ne bloque jamais** (retourne 200 OK).	Mode "Écoute". Permet d'apprendre le trafic.
On	Traite les règles et **bloque** (retourne 403 Forbidden) si le seuil est dépassé.	Mode "Protection".

---

4. Procédure d'Activation (5 Phases)

Phase 1 : Préparation (J-7)

Objectif : Sécuriser l'état initial avant tout changement.

Configuration Initiale (docker-compose.yml ou modsecurity.conf) :
YAML
environment:

```
MODSEC_RULE_ENGINE: "Off"
 MODSEC_AUDIT_ENGINE: "RelevantOnly"
 PARANOIA: 1
```

Action	Responsable
1.1 Snapshot/Backup de la configuration WAF actuelle.	Ops
1.2 Lister les flux critiques (ex: upload de fichiers, éditeurs HTML).	Dev
1.3 Vérifier que les logs remontent bien dans le puits de logs (ELK, Datadog, fichier).	Ops

Phase 2 : Mode Observation (J0 à J+7)

Objectif : Laisser tourner le moteur "à vide" pour voir ce qu'il *aurait* bloqué.

Action Configuration :
Passage en DetectionOnly. Les seuils d'anomalie restent bas pour tout logger.
YAML
environment:

```
MODSEC_RULE_ENGINE: "DetectionOnly"

  ANOMALY_INBOUND: 5

  ANOMALY_OUTBOUND: 4
```

Surveillance Quotidienne :
Utiliser cette commande pour repérer les fausses alertes :
Bash

```
docker logs waf-modsecurity 2>&1 | grep -E "(ModSecurity|id \"[0-9]+\")"
```

Phase 3 : Tuning et Exclusions (J+7 à J+14)

Objectif : Créer les exceptions pour le trafic légitime identifié en Phase 2.

Stratégie d'exclusion (Whitelist) :
Les règles doivent être ajoutées dans
REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf.
Exemple 1 : Ignorer l'injection SQL sur un champ de recherche spécifique
Apache
SecRuleUpdateTargetById 942100 "!ARGS:search_query"

Exemple 2 : Désactiver une règle précise sur une URL précise
Apache
SecRule REQUEST_URI "@beginsWith /api/upload" \

"id:1000,phase:1,pass,nolog,ctl:ruleRemoveById=920230"

-

Phase 4 : Activation Progressive (J+14 à J+21)

Objectif : Activer le blocage (On) mais avec une tolérance très élevée, que l'on réduit peu à peu.

Configuration de base :
YAML
environment:

MODSEC_RULE_ENGINE: "On"

PARANOIA: 1

-
- **Calendrier de réduction du seuil (ANOMALY_INBOUND)** :

Jour	**Seuil d'Anomalie**	**Signification**
J+14	100	Très permissif. Bloque uniquement les attaques massives.

J+16	50	Permissif.
J+18	20	Standard haut.
J+21	5	**Cible de Production**. Très strict.

⚠️ **Surveillance Critique** : À chaque baisse de seuil, surveillez les erreurs **403** dans les logs d'accès Nginx/Apache. Si > 1% d'erreur, remontez le seuil immédiatement.

Phase 5 : Production Stable (J+21+)

Le système est opérationnel.

Configuration Finale :
YAML
```
MODSEC_RULE_ENGINE: "On"

ANOMALY_INBOUND: 5

REPORTING_LEVEL: 2
```

-

---

5. Procédure de Rollback (Urgence)

Si l'application devient inutilisable ou bloque des clients critiques :

1. Action Immédiate : Repasser en mode détection.
   Modifier la variable d'environnement :
   `MODSEC_RULE_ENGINE: "DetectionOnly"`

Appliquer :
Bash
```
docker compose down && docker compose up -d
```

2.
3. **Analyser** : Ne repasser à "On" qu'après avoir identifié la règle coupable via les logs.

---

6. Annexes : Aide-mémoire

Règles souvent problématiques (Faux Positifs fréquents)

ID Règle	Nom	Contexte fréquent
920230	Multiple URL Encoding	Souvent déclenché par des tokens OAuth ou des URL complexes.
942100	SQL Injection	Déclenché par des cookies de session ou du JSON.
941100	XSS Detection	Déclenché par des éditeurs de texte riche (CMS, WordPress).
920420	Content-Type	Déclenché lors d'upload de fichiers sans extension claire.

Commandes de Debug

Voir les règles déclenchées (sans doublons) :

Bash

```
docker logs waf-modsecurity 2>&1 | grep "ModSecurity: Warning" | sed -E 's/.*id "([0-9]+)".*/\1/' | sort | uniq -c
```