PREMIER MINISTRE

Secrétariat général de la défense et de la sécurité nationale

Agence nationale de la sécurité des systèmes d'information

Rapport de certification ANSSI-CSPN-2010/05

ModSecurity v2.5.12

Paris, le 20 déc. 2010

Le directeur général de l'agence nationale de la sécurité des systèmes d'information

Patrick Pailloux

[ORIGINAL SIGNE]

Avertissement

Ce rapport est destiné à fournir aux commanditaires un document leur permettant d'attester du niveau de sécurité offert par le produit dans les conditions d'utilisation ou d'exploitation définies dans ce rapport pour la version qui a été évaluée. Il est destiné également à fournir à l'acquéreur potentiel du produit les conditions dans lesquelles il pourra exploiter ou utiliser le produit de manière à se trouver dans les conditions d'utilisation pour lesquelles le produit a été évalué et certifié. C'est pourquoi ce rapport de certification devrait être lu conjointement aux guides d'utilisation et d'administration évalués ainsi qu'à la cible de sécurité du produit qui décrit les menaces, les hypothèses sur l'environnement et les conditions d'emploi présupposées afin que l'utilisateur puisse juger de l'adéquation du produit à son besoin en termes d'objectifs de sécurité.

La certification ne constitue pas en soi une recommandation du produit par l'agence nationale de la sécurité des systèmes d'information (ANSSI), et ne garantit pas que le produit certifié soit totalement exempt de vulnérabilités exploitables.

Toute correspondance relative à ce rapport doit être adressée au :

Secrétariat général de la défense et de la sécurité nationale
Agence nationale de la sécurité des systèmes d'information
Centre de certification
51, boulevard de la Tour Maubourg
75700 Paris cedex 07 SP

certification.anssi@ssi.gouv.fr

Référence du rapport de certification

ANSSI-CSPN-2010/05

Nom du produit

ModSecurity v2.5.12

Référence/version du produit

Version 2.5.12

Critères d'évaluation et version

CERTIFICATION DE SECURITE DE PREMIER NIVEAU

(CSPN, Phase expérimentale)

Développeur

Breach Security

2141 Palomar Airport Road
Suite 200
Carlsbad, CA 92011
Etats-Unis

Commanditaire

Agence nationale de la sécurité des systèmes d'information

Secrétariat Général de la Défense et de la Sécurité Nationale
51, boulevard de la Tour Maubourg
75700 – Paris – 07 SP
France

Centre d'évaluation

SOGETI Infrastructure Services

6 - 8, Rue Duret, 75016 Paris, France

Tél : +33 (0)1 58 44 55 66, mél : edouard.jeanson@sogeti.com

Préface

La certification

La certification de la sécurité offerte par les produits et les systèmes des technologies de l'information est régie par le décret 2002-535 modifié du 18 avril 2002, publié au Journal officiel de la République française. Ce décret indique que :

- L'agence nationale de la sécurité des systèmes d'information élabore les **rapports de certification**. Ces rapports précisent les caractéristiques des objectifs de sécurité proposés. Ils peuvent comporter tout avertissement que ses rédacteurs estiment utile de mentionner pour des raisons de sécurité. Ils sont, au choix des commanditaires, communiqués ou non à des tiers ou rendus publics (article 7).
- Les **certificats** délivrés par le Premier ministre attestent que l'exemplaire des produits ou systèmes soumis à évaluation répond aux caractéristiques de sécurité spécifiées. Ils attestent également que les évaluations ont été conduites conformément aux règles et normes en vigueur, avec la compétence et l'impartialité requises (article 8).

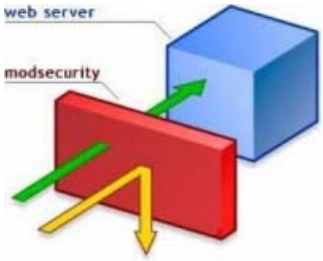
Les procédures de certification sont disponibles sur le site Internet www.ssi.gouv.fr.

Table des matières

1. Le produit

1.1. Présentation du produit

Le produit évalué « ModSecurity v2.5.12 » est un module qui s'intègre au serveur Web Apache et qui permet de mettre en place un pare-feu applicatif dédié aux applications Web. Il fournit un moteur permettant de détecter et de se prémunir des attaques avant qu'elles n'atteignent l'application Web protégée.



Fonctionnement global de ModSecurity

ModSecurity offre les mécanismes suivants :

- la journalisation du trafic HTTP ;
- la surveillance du trafic et détection en temps réel des attaques ;
- la prévention des attaques et adaptation des règles de détection en vue de corriger les vulnérabilités applicatives sans modifier réellement les applications Web.

Une fois le module ModSecurity installé sur un serveur Apache, celui-ci ne fournit que peu de protection par lui-même car il n'intègre pas de règles par défaut. Il faut donc ajouter et configurer les règles nécessaires pour sécuriser les applications Web.

Néanmoins, le projet « ***ModSecurity Core Rule Set*** » a été créé afin de fournir un jeu de règles fournissant une protection générique contre les attaques les plus employées à l'encontre des applications Web. Une version de ces règles est par ailleurs fournie dans l'archive de ModSecurity.

1.2. Description du produit évalué

La cible de sécurité [CDS] définit le produit évalué, ses fonctionnalités de sécurité évaluées et son environnement d'exploitation.

1.2.1. Catégorie du produit

☐ 1 - détection d'intrusions
☐ 2 - anti-virus, protection contre les codes malicieux
☒ 3 – pare-feu
☐ 4 - effacement de données
☐ 5 - administration et supervision de la sécurité
☐ 6 - identification, authentification et contrôle d'accès
☐ 7 - communication sécurisée
☐ 8 - messagerie sécurisée
☐ 9 – stockage sécurisé
☐ 10 - matériel et logiciel embarqué
☐ 99- Autres

1.2.2. Identification du produit

La version est identifiable sur la première ligne du fichier « CHANGES » de l'archive du produit, ainsi que dans la documentation (fichier « index.html » par exemple).

1.2.3. Services de sécurité

La principale fonctionnalité de sécurité de ModSecurity est de protéger en disponibilité et en intégrité les applications et les données hébergées par un serveur Web contre des attaques réseau. Cette protection est assurée par les règles configurées dans le logiciel à travers l'utilisation des « *ModSecurity Core Rule Set* ».

Les fonctions de sécurité de ModSecurity sont donc :

1. **Mettre en œuvre les règles de sécurité telles que configurées dans l'application :**
   - détecter les événements selon le paramétrage du « *ModSecurity Core Rule Set* » :
     - analyse de la conformité protocolaire :
       - détection des anomalies dans le protocole HTTP ;
     - détection d'attaques :
       - détection d'outils de collecte d'information : scanneurs, robots d'indexation, robots, … ;
       - détection des tentatives de connexion des « *chevaux de Troie* » ou des « *portes dérobées* » déjà déployées au sein du système d'information ;
       - détection générique des attaques :
         - injections SQL diverses ;
         - *Cross Site Scripting* (XSS) ;
         - injection de commandes ;
         - injection de code ;
         - HTTP *Response Splitting* ;
   - réaliser des actions préventives (en vue de prévenir une attaque) :
     - protection du contenu XML ;
     - surveillance des accès aux sites Web ;
     - réécriture des messages d'erreur renvoyés par le serveur Web ;
2. **Bloquer les attaques suite à leur détection.**
3. **Journaliser les événements et les actions.**

La configuration par défaut de ModSecurity met en œuvre un filtrage par liste noire s'appuyant sur une liste de signatures ou motifs réputés dangereux.

1.2.4. Configuration évaluée

Le logiciel ModSecurity est évalué avec l'activation des règles par défaut de *ModSecurity Core Rule Set* version 2.0.5.

2. L'évaluation

2.1. Référentiels d'évaluation

L'évaluation est menée conformément au référentiel « Certification de Sécurité de Premier Niveau en phase expérimentale ». Les références des documents se trouvent en annexe 2.

2.2. Charge de travail prévue et durée de l'évaluation

L'évaluation de la version 2.5.12 de ModSecurity fait suite à l'évaluation de la précédente version 2.5.11 qui relevait la présence d'un certain nombre de non conformités er de vulnérabilités du produit. La charge de travail de cette évaluation est de 10 hommes x jours.

2.3. Travaux d'évaluation

Ce paragraphe apporte des compléments sur la cible de sécurité [CDS] fournie en entrée de l'évaluation. Ces précisions sont issues du Rapport Technique d'Evaluation [RTE] élaboré par l'évaluateur suite à ses travaux.

2.3.1. Fonctionnalités, environnement d'utilisation et de sécurité

2.3.1.1.Spécification de besoin du produit

Conforme à la cible de sécurité [CDS] (chapitre « Argumentaire »).

2.3.1.2.Biens sensibles manipulés par le produit

Conforme à la cible de sécurité [CDS] (chapitre « Description des biens sensibles que le produit doit protéger »).

2.3.1.3.Description des menaces contre lesquelles le produit apporte une protection

Conforme à la cible de sécurité [CDS] (Chapitre « Description des menaces »)

2.3.1.4.Fonctions de sécurité

Conforme à la cible de sécurité [CDS] (chapitre « Description des fonction de sécurité du produit »).

2.3.1.5.Utilisateurs typiques

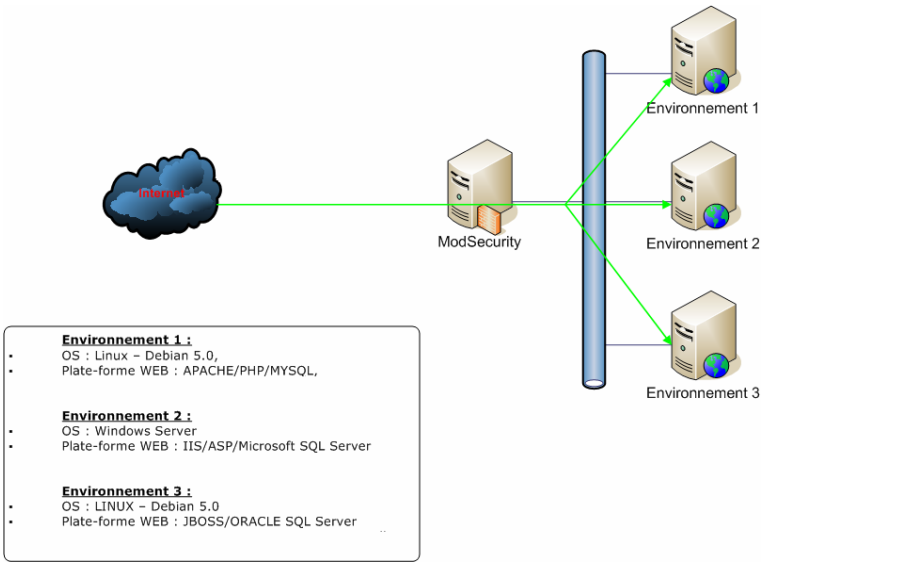
Conforme à la cible de sécurité [CDS] (chapitre « Argumentaire »).

2.3.2. Installation du produit

2.3.2.1. Plate-forme de test

L'architecture nécessaire à la réalisation de l'évaluation est déployée à l'aide de machines virtuelles (**VMware Workstation** dans sa version **6.5.3 build-185404)**, dans un souci de reproductibilité des tests et de stockage des archives.

L'architecture mise en place est résumée sur le schéma suivant :



ModSecurity est déployé sur la distribution Linux Debian 5.0.3, installée sur une machine virtuelle. Il utilise le serveur Web Apache 2.2.9, configuré pour être utilisé en mode « **proxy inverse** » en utilisant le module Apache *mod_proxy*. Aucune application Web n'est installée sur ce serveur.

Cette première machine virtuelle en protège trois autres :

- **environnement 1 :** une machine virtuelle utilisant le système d'exploitation Linux Debian 5.0.3, sur laquelle le serveur Web Apache 2.2.9 et le serveur MySQL 5.0.51 fournissent l'application PHP « drupal », à protéger ;
- **environnement 2 :** une machine virtuelle utilisant le système d'exploitation Windows Server 2003, sur laquelle le serveur Web IIS 6.0 et le serveur SQL Server Express 2005 fournissent l'application ASP.NET « DotNetNuke », à protéger ;
- **environnement 3 :** une machine virtuelle utilisant le système d'exploitation Linux Debian 5.0.3, sur laquelle le serveur applicatif Apache Tomcat et le serveur SQL HSQLDB fournissent l'application J2EE « uPortal », à protéger. A l'origine, le serveur applicatif devait être JBoss accompagné du serveur SQL Oracle, mais l'application uPortal est plus difficile à installer sur ces serveurs.

Les environnements sont 32 bits uniquement (x86). Le système d'exploitation de toutes les machines virtuelles, ainsi que les services s'y exécutant, possèdent l'ensemble des mises à jour disponibles au début de l'évaluation.

La configuration du serveur Apache sur lequel ModSecurity est installé permet l'utilisation de règles différentes ainsi que la création de journaux distincts pour chaque application protégée.

2.3.2.2.Particularités de paramétrage de l'environnement

Aucun paramétrage supplémentaire n'est effectué.

2.3.2.3.Options d'installation retenues pour le produit

Le serveur Web Apache est installé par le gestionnaire de paquets de la distribution Linux Debian. ModSecurity est compilé depuis ses sources, téléchargées directement sur le site de l'éditeur. Aucune option particulière n'est précisée lors de la compilation ; il s'agit donc d'une installation standard par défaut du produit.

2.3.2.4.Description de l'installation et des non-conformités éventuelles

Téléchargement et extraction de l'archive

ModSecurity est installé sur le système via la compilation de ses sources, contenues dans l'archive « modsecurity-apache_2.5.12.tar.gz » préalablement téléchargée via le site officiel du logiciel sur le site *SourceForge*. Elle est signée numériquement en utilisant la clé PGP du développeur Brian Rectanus ; la signature est valide.

L'archive contient les dossiers suivants :

- **apache2** : ce dossier contient les sources ;
- **doc** : ce dossier contient la documentation anglaise aux formats PDF et HTML ;
- **rules** : ce dossier contient les ModSecurity Core Rule Set version 2.0.5 ;
- **tools** : ce dossier contient un outil pour mettre à jour les ModSecurity Core Rule Set.

Elle contient également les fichiers suivants : *changes*, *licence*, *modsecurity.conf-minimal*, *MODSECURITY_LICENSING_EXCEPTION* et *readme.txt*. Le fichier *changes* contient la liste des changements depuis la version 2.0.0 datée d'octobre 2006. Le fichier *modsecurity.conf-minimal* contient une configuration minimale pour activer ModSecurity sur le serveur Web Apache, une fois le module compilé.

Compilation

Les outils de compilation doivent être installés sur la machine. Lors de l'évaluation, ModSecurity est compilé en utilisant le compilateur GCC. Les bibliothèques suivantes sont requises pour la compilation du module :

Bibliothèque	*Version*	*Description*
libAPR	1.2.12-5	*Apache Portable Runtime*. Nécessaire au module Apache. Contient des fonctions et mécanismes (*apr_psprintf*, *apr_table_t*, etc.) utilisés par ModSecurity.
libAPR-util	1.2.12	*Apache Portable Runtime* - outils. Nécessaire au module Apache. Contient des fonctions utiles (*apr_md5*, *apr_sha1*, etc.) utilisées par ModSecurity.
libPCRE	7.6-2.1	Moteur d'expressions rationnelles. Nécessaire à la compilation des règles, majoritairement basées sur des expressions rationnelles.
libXML2	2.6.32	Moteur de traitement XML. Nécessaire à l'analyse du contenu XML.

libLUA	5.1.3-1	Interprétation du langage LUA. Nécessaire à l'exécution de scripts LUA en réponse à certaines règles.
libCURL3	7.18.2-8	Transfert par URL. Nécessaire à l'outil ModSecurity Audit Log Collector (mlogc).

La compilation s'effectue par l'enchainement des commandes suivantes :

```
$> ./configure
[…]
$> make
[…]
```

S'il manque une bibliothèque sur le système, la procédure remonte correctement l'erreur.

1) Installation

L'installation de ModSecurity se termine par la commande :

```
$> make install
[…]
```

Cette commande copie le module ModSecurity – mod_security2.so – nouvellement créé dans le répertoire des modules d'Apache, à savoir « /usr/lib/apache2/modules/ » sur Debian.

2) Configuration

L'activation de ModSecurity dans Apache s'effectue par l'ajout dans la configuration du serveur Web de la ligne suivante :

```
LoadModule   security2_module          /usr/lib/apache2/modules/mod_security2.so
```

Sur Debian, cette ligne doit être ajoutée dans un fichier créé dans le répertoire « mods-available » d'Apache et un lien symbolique pointant vers ce fichier doit être créé dans le répertoire « mods-enabled ». Ce détail n'est pas expliqué dans la documentation de ModSecurity car il s'agit d'une particularité de la configuration d'Apache sous Debian.

La configuration initiale se termine par l'activation des règles par défaut ModSecurity Core Rule Set, si souhaité. En considérant que les règles sont copiées dans le dossier « conf/mod_security2/ » d'Apache, cela se traduit par l'ajout des lignes suivantes dans la configuration de l'hôte virtuel à protéger (défini dans « sites-enabled » sous Debian) :

```
include         conf/mod_security2/*.conf
include         conf/mod_security2/base/*.conf
```

2.3.2.5.Durée de l'installation

En faisant abstraction des problèmes de dépendance aux bibliothèques tierces qui ne seraient pas installées, l'installation est immédiate.

2.3.2.6.Notes et remarques diverses

L'installation de ModSecurity peut aussi être effectuée via les paquets fournis par les distributions Linux. Il n'est pas toujours nécessaire de procéder à la compilation du module. L'installation et la compilation sous Windows sont aussi décrites dans la documentation.

2.3.3. Analyse de la documentation

La documentation est bien réalisée et à jour (voir [GUIDES]). L'évaluateur n'a pas détecté d'erreurs. Elle couvre l'intégralité du logiciel. La lecture du manuel de référence est nécessaire à la bonne compréhension du produit et à sa configuration (globale, écriture des règles, compréhension des journaux, etc.). La documentation est conforme au logiciel testé.
Une traduction française de la documentation a été réalisée par SOGETI/ESEC (voir [GUIDES]). Cette traduction concerne une partie du « ModSecurity Reference Manual » qui couvre la description, l'installation, la compilation et l'activation de ModSecurity. Les méthodes de fonctionnement générales du logiciel sont détaillées.

2.3.4. Revue du code source

Même si quelques erreurs au niveau du code source ont pu être décelées, la plus importante étant la non-vérification des valeurs de retour des allocations mémoires, le code source de ModSecurity est propre, clair, documenté et homogène.

2.3.5. Fonctionnalités testées

ModSecurity est testé avec un ensemble d'attaques classiques. Les tests sont réalisés avec le jeu de règles par défaut du ModSecurity Core Rule Set conformément aux environnements définis. Les règles sont particulièrement efficaces, et bloquent toutes les attaques de base.

Le tableau suivant résume les tests fonctionnels réalisés.

Résumé	*Résultat*
Détection des anomalies dans le protocole HTTP	**Réussite**
Détection et blocage des injections SQL	**Echec**
Détection et blocage des attaques de type Cross Site Scripting	**Réussite**
Détection et blocage des attaques de type HTTP Response Splitting	**Echec**
Détection et blocage des attaques de type injection de code PHP	**Réussite**
Détection et blocage des attaques de type inclusion de fichier	**Réussite**
Détection et blocage des attaques de type injection de commandes système	**Réussite**
Détection et blocage des attaques utilisant l'encodage des caractères	**Réussite**
Filtrage en sortie des exceptions JAVA	**Échec**
Filtrage en sortie des erreurs IIS	**Échec**
Filtrage en sortie des erreurs PHP	**Échec**

Détection d'une *backdoor* PHP envoyée sur une application Web	**Échec**
Analyse antivirale sur les fichiers envoyés au serveur	**Réussite**
Protection du contenu XML	**Réussite**
Journalisation des évènements et des actions	**Réussite** [1]

Détection d'outils de collecte d'information

Les règles destinées à détecter les robots, définies dans le fichier *modsecurity_crs_35_bad_robots.conf*, utilisent des signatures sur des éléments (`User-Agent`, champ « `Title` » des *Web Shell*, etc.) facilement modifiables par un attaquant sans incidence sur l'attaque. Elles ne sont présentes que pour nettoyer le trafic, comme indiqué en commentaire dans le fichier des règles en question.

***Réalisation de tests fonctionnels sur l'environnement 1 - Application* Drupal**

Cinq *exploits* publics ciblant *Drupal* sont téléchargés sur le site *milw0rm*[2], et testés sans aucune modification. Tous les *exploits* sont bloqués. La réalisation de ces tests est pertinente car quatre de ces attaques utilisent des méthodes d'exploitation différentes.

Versions vulnérables	***Type***	***Résultat***
Drupal <= 4.5.3 et <= 4.6.1	Injection de code PHP dans le contenu d'une requête POST	**Bloqué**
Drupal <= 4.7	Envoi d'un fichier malveillant par un formulaire d'upload	**Bloqué**
Drupal < 5.1	Injection de code PHP dans le contenu d'une requête POST	**Bloqué**
Drupal < 4.7.6	Injection de code PHP dans le contenu d'une requête POST	**Bloqué**
Drupal <= 5.2	Injection de code PHP dans les paramètres de l'URL. Mauvaise gestion des *hashs* dans PHP Zend.	**Bloqué**

***Réalisation de tests fonctionnels sur l'environnement 2 - Application* DotNetNuke**

Les vulnérabilités de *DotNetNuke* utilisées dans ces tests fonctionnels proviennent du site officiel de l'éditeur. Les tests qui échouent correspondent à des divulgations d'informations (version du serveur, de la base de données, etc.) non filtrées par ModSecurity.

[1] Par défaut, ModSecurity n'enregistre pas les requêtes légitimes.
[2] http://www.milw0rm.com/search.php?dong=drupal

Version vulnérable	*Type*	*Résultat*
DotNetNuke <= 4.8 et <= 5.1.4	Injection de code HTML et JavaScript	**Bloqué**
DotNetNuke <= 4.0 et <= 5.1.4	Divulgation d'informations	**Non bloqué**
DotNetNuke <= 4.9.3	Injection de code HTML et JavaScript	**Bloqué**
DotNetNuke <= 4.9.3	Divulgation d'informations par les pages d'erreurs	**Non bloqué**

***Réalisation de tests fonctionnels sur l'environnement 3 – Application* HacmeBooks**

L'application *HacmeBooks*[3] est utilisée à la place d'*uPortal* car cette dernière ne possède pas de vulnérabilités publiques intéressantes. *HacmeBooks* est une application de test de vulnérabilités sur plate-forme J2EE qui représente des scénarii réels d'attaques. Les vulnérabilités proviennent du guide « User and Solution » de *HacmeBooks*.

Type de vulnérabilité	*Type*	*Résultat*
Filtrage des exceptions JAVA	Déclenchement d'une exception Java	**Non bloqué**
Injection SQL 1	`SELECT * FROM <field> WHERE <data> LIKE ''; SHUTDOWN;--`	**Non Bloqué**
Injection SQL 2	`<data>', 1); insert into products (title,...);`	**Non Bloqué**
Injection SQL 3	`<data>'; DROP TABLE products;--`	**Non Bloqué**
***Cross Site Scripting* (XSS)**	`<script> location="http://attaquant.com/" </script>`	**Bloqué**
***Cross Site Request Forgery* (CSRF)**	`<img src="http://cible.com/...">`	**Bloqué**

2.3.6. Fonctionnalités non testées

Conformément à la cible, les seules fonctionnalités qui ne sont pas testées sont l'analyse des documents PDF et l'administration.

2.3.7. Synthèse des fonctionnalités testées et des non-conformités

Deux non-conformités sont identifiées par rapport à la cible de sécurité.

La première concerne le filtrage en sortie ; ModSecurity, dans sa configuration par défaut, se contente de journaliser les sorties d'erreurs générées par les diverses applications Web et non de les bloquer. Il suffit de modifier cette configuration pour que ModSecurity bloque ces flux sortants.

[3] http:/www.foundstone.com/us/resources/proddesc/hacmebooks.htm

La deuxième est due à ModSecurity Core Rule Set 2.0.5 qui contient deux failles entraînant le non-blocage de certaines attaques de type HTTP Response Splitting et injections SQL.
Le Core Rule Set 2.0.5 comprend une erreur dans le fichier "modsecurity_crs_41_sql_injection_attacks.conf" de gestion des injections SQL. Dans un premier temps, les éléments de la requête sont bien comparés au contenu du fichier .data en effectuant les différentes transformations classiques : none, urlDecodeUri, htmlEntityDecode, lowercase, replaceComments, compressWhiteSpace. Ceci est donc correct.
Cependant, dans un deuxième temps, les éléments des requêtes sont testés par rapport à des expressions rationnelles et dans ce cas, aucune transformation n'est utilisée. Ainsi, il suffit pour un attaquant de mettre, par exemple, un de ces mots en majuscules pour que l'expression régulière ne soit pas validée.

Ce problème est facilement corrigeable puisqu'il suffit d'ajouter les transformations pour chacune des règles où celles-ci sont manquantes. Par exemple, pour régler le problème pour le mot clef « shutdown », il suffit d'effectuer la modification suivante :

```
# Version originale
        SecRule REQUEST_FILENAME|ARGS_NAMES|ARGS|XML:/* "\;\W*?\bshutdown\b" \

"phase:2,rev:'2.0.5',capture,t:none,ctl:auditLogParts=+E,pass,nolog,auditlog,msg:'S
QL Injection Attack',id:'959002',tag:'WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-
19',tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',logdata:'%{TX.
0}',severity:'2',setvar:'tx.msg=%{rule.msg}',setvar:tx.sql_injection_score=+%{tx.cr
itical_anomaly_score},setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:
tx.%{rule.id}-WEB_ATTACK/SQL_INJECTION-%{matched_var_name}=%{tx.0}"
```

```
# Version modifiée
        SecRule REQUEST_FILENAME|ARGS_NAMES|ARGS|XML:/* "\;\W*?\bshutdown\b" \

"phase:2,rev:'2.0.5',capture,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:lowercase,t
:replaceComments,t:compressWhiteSpace,ctl:auditLogParts=+E,pass,nolog,auditlog,msg:
'SQL Injection Attack',id:'959002',tag:'WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-
19',tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',logdata:'%{TX.
0}',severity:'2',setvar:'tx.msg=%{rule.msg}',setvar:tx.sql_injection_score=+%{tx.cr
itical_anomaly_score},setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:
tx.%{rule.id}-WEB_ATTACK/SQL_INJECTION-%{matched_var_name}=%{tx.0}"
```

2.3.8. Avis d'expert sur le produit

La documentation est claire et complète. Elle est mise à jour à chaque nouvelle version de ModSecurity.
Le code source est propre, clair, et homogène. La lecture est par ailleurs facilitée par les commentaires présents pour la plupart des fonctions. Malgré quelques erreurs, la sécurité semble avoir été prise en compte dès la conception du projet.
Les tests effectués montrent que ModSecurity bloque efficacement les attaques Web classiques. Comme indiqué au paragraphe 2.3.7, les non-conformités identifiées peuvent être facilement corrigées en modifiant la configuration de ModSecurity et certaines règles du ModSecurity Core Rule Set.

L'exploitation[4] efficace des journaux nécessite l'utilisation d'un outil externe à cause de leur niveau de verbosité.

2.3.9. Analyse de la résistance des mécanismes et des fonctions

2.3.9.1.Liste des fonctions et des mécanismes testés - résistance

Les fonctions de sécurité de ModSecurity s'articulent autour de trois axes :

- la mise en œuvre des règles de sécurité telles que configurées dans l'application ;
- le blocage des attaques suite à leur détection ;
- la journalisation des évènements et des actions.

Plusieurs attaques permettant de contourner le système de détection de ModSecurity ont été découvertes.

Contournement des règles en sortie

Les règles par défaut de ModSecurity examinent le contenu des réponses envoyées par le serveur, dans le but de détecter si des informations sensibles[5] sont retournées par le serveur, ou si la page Web retournée par le serveur référence un site malveillant connu.

Dans la configuration par défaut, les requêtes provoquant une erreur de l'application Web protégée sont journalisées, mais les réponses associées ne sont pas bloquées, ce qui donne une fausse impression de sécurité.

Il est par ailleurs trivial de contourner l'examen des réponses (filtrage, journalisation, etc.). Comme spécifié dans la documentation, la directive `SecResponseBodyMimeType` liste les types MIME pour lesquels la réponse est examinée (par défaut : (null), `text/html`, `text/plain`, `text/xml`). Un attaquant peut simplement demander au serveur que la réponse soit encodée dans un format non spécifié par la directive `SecResponseBodyMimeType` ; le contenu de la réponse ne sera alors pas examiné par ModSecurity.

Contournement de certaines règles spécifiques

Le filtrage des injections et des commandes par liste noire mis en œuvre par défaut est intrinsèquement limité. Une approche par liste blanche permettrait de réaliser un filtrage adapté à une application Web.

- Contournement du filtrage d'injections SQL.

Les injections SQL sont bloquées par les règles contenues dans le fichier *modsecurity_crs_41_sql_injection_attack.conf*. Ces règles filtrent un ensemble de mots clés susceptibles d'être présents dans une injection SQL.

L'exemple suivant montre le contournement d'une authentification à un site Web sans être bloqué par ModSecurity. L'apostrophe avant le second 1 contourne la règle :

[4] Lors de l'évaluation, un script Python basique a été développé afin de faciliter l'étude des journaux.
[5] Code source, erreurs Java, etc….

```
/vulns/output.html?login=admin' or 1='1
```

Même si ModSecurity est configuré pour bloquer les injections SQL, il est à priori toujours possible de modifier des requêtes SQL ne pouvant être bloquées par ModSecurity.

- Contournement du filtrage de XSS.

Les injections de type *Cross Site Scripting* sont bloquées par les règles du fichier *modsecurity_crs_41_xss_attacks.conf*. Ces règles filtrent un ensemble de mots clés susceptibles d'être présents dans une attaque XSS contenant du code JavaScript ou de balises HTML par une requête GET ou POST.

Il est cependant possible d'injecter des balises `script` par une requête POST, en divisant le nom de celles-ci par le caractère nul, et en utilisant des techniques de polymorphisme pour ne pas déclencher les filtres sur le code JavaScript.

- Contournement du filtrage de commandes système.

Un ensemble de règles bloque les commandes système classiques (*modsecurity_crs_40_generic_attacks.conf*, section « Prequalify Request Matches »). Du fait de la diversité des commandes existantes, il est possible de contourner les mots clés spécifiés par ces règles (dans le fichier *modsecurity_40_generic_attacks.data*) et ainsi d'exécuter une commande arbitraire. Seules des attaques génériques sont bloquées.

- Contournement du filtrage de code PHP.

Un ensemble de règles traite l'injection de code PHP (*modsecurity_crs_40_generic_attacks.conf*, section « PHP injection »). Cependant, différentes techniques de polymorphisme utilisant des mots clés interdits encodés en base 64 permettent de contourner les expressions rationnelles destinées à décrire du code PHP. Cette vulnérabilité concerne le *ModSecurity Core Rule Set.* Notons que l'injection de code PHP serait rendue légèrement plus complexe si le mot clé _REQUEST et les fonctions de gestion des fonctions étaient filtrés.

2.3.9.2.Liste des fonctions et des mécanismes non testés - résistance

Sans objet.

2.3.9.3.Avis d'expert sur la résistance des mécanismes

Les techniques d'évasion montrent les limites de tout système de filtrage. En effet, il est impossible de bloquer toutes les attaques sans obtenir de faux-positifs. Il est donc nécessaire de trouver un juste milieu pour n'être vulnérable qu'à des attaques plus complexes tout en ne nuisant pas au fonctionnement de l'application Web protégée. ModSecurity remplit pleinement cet objectif pour les injections de code PHP et de XSS. Cependant, les règles concernant l'injection de commandes système et les injections SQL restent à améliorer.

Les règles par défaut ne bloquent pas les réponses du serveur Web et génèrent seulement des entrées dans les journaux, contrairement aux attentes premières d'un administrateur. Une lecture attentive du guide utilisateur permet de modifier aisément ces règles pour inverser ce comportement.

2.3.10. Analyse des vulnérabilités (conception, construction…)

2.3.10.1. Liste des vulnérabilités connues

Il n'y a pas de vulnérabilité connue concernant la version 2.5.12 de ModSecurity.

2.3.10.2. Liste des vulnérabilités découvertes lors de l'évaluation et avis d'expert

Trois vulnérabilités sont découvertes lors de l'évaluation.

Déni de service sur les expressions rationnelles
Le traitement de chaînes de caractères très longues prend un temps important. En effet, dans la configuration par défaut, le traitement d'une requête POST contenant un argument d'une longueur de 70 000 octets prends 75 secondes. Lors de la réception d'une telle requête, ModSecurity applique successivement chacune des règles, et l'accumulation du temps passé pour vérifier chaque expression régulière de chaque règle provoque un déni de service. Au final, ModSecurity ne bloque pas la requête.

Avis d'expert
Les options de configuration permettent d'atténuer les risques de déni de service sous réserve que la taille des requêtes inspectées soit raisonnable. La directive `SecRequestBodyNoFilesLimit` peut être utilisée pour limiter la taille des requêtes POST (hors envoi de fichiers). En effet, la plupart des applications Web n'ont en général pas besoin de traiter des arguments de longueur supérieure à 1 Ko (entrée de formulaire, par exemple).

L'utilisation du paramètre `SecPcreMatchLimit` permet de limiter le temps d'exécution d'une expression rationnelle donnée.

Déni de service sur la journalisation
Lorsqu'il n'y a plus de place sur le disque, ModSecurity continue de fonctionner normalement. Cependant, il n'enregistre plus d'information dans ses journaux. Les requêtes non légitimes continuent à être bloquées.

Avis d'expert
Il est recommandé de configurer le niveau des journaux de debug à une valeur inférieure à 3, et de déporter le stockage des journaux sur un serveur distant.

Vérification des valeurs de retour des fonctions d'allocation de mémoire
Les valeurs de retour des fonctions d'allocation de mémoire ne sont quasiment jamais vérifiées. Un attaquant peut faire planter le processus Apache en allouant toute la mémoire disponible du système, de façon à ce qu'une fonction d'allocation de mémoire dont la valeur de retour n'est pas vérifiée échoue. Le processus Apache plantera suite au déréférencement du pointeur nul retourné par la fonction d'allocation de mémoire.

Avis d'expert
Bien que la qualité du code source soit globalement bonne, les valeurs de retour de toutes les fonctions d'allocation de mémoire devraient être vérifiées pour éviter les crashs du système en cas de manque de mémoire. Cette vulnérabilité est difficilement exploitable par un attaquant.

2.3.11. Accès aux développeurs

Le code source complet est disponible depuis le site de ModSecurity (http://www.modsecurity.org). L'évaluateur a eu des échanges avec les développeurs de ModSecurity au cours de l'évaluation de la version 2.5.11, qui ont conduit au développement de la version 2.5.12.

2.3.12. Analyse de la facilité d'emploi et préconisations

2.3.12.1. Cas où la sécurité est remise en cause

Il n'est signalé nulle part dans la documentation que la modification ou l'ajout d'une règle nécessite le rechargement du serveur Web.
Dans sa configuration par défaut, *ModSecurity* n'est pas configuré pour bloquer une attaque en sortie, et se contente de journaliser l'information.

2.3.12.2. Recommandations pour une utilisation sûre du produit

- Lire le manuel de référence : ce manuel est nécessaire à la bonne compréhension du produit et à l'établissement d'une configuration correcte.
- Désactiver les journaux de debug : une fois les règles en place vis-à-vis des applications à protéger, il n'est plus utile de conserver les journaux de debug, trop verbeux pour une utilisation courante.
- Déporter les journaux d'audit sur un serveur de journaux dédié : cette précaution simple évite de saturer le serveur sur lequel est installé ModSecurity, ce qui provoquerait un ralentissement sur toute la plate-forme.
- Créer des exceptions plutôt que de désactiver des règles entières : en effet, il est souvent nécessaire de désactiver des règles pour que les applications protégées ne génèrent pas de faux-positifs. Au lieu de désactiver totalement une règle sans doute utile, il est plus judicieux de créer une exception à cette règle sur l'application qui pose problème.
- Tester l'installation du *ModSecurity Core Rule Set* avant de l'utiliser dans un environnement de production.
- Configurer ModSecurity et le *ModSecurity Core Rule Set* pour que les attaques en sortie soient bloquées.
- Modifier le *ModSecurity Core Rule Set* pour bloquer les attaques de type HTTP *Response Splitting* et les injections SQL.

2.3.12.3. Avis d'expert sur la facilité d'emploi

La qualité des règles du *ModSecurity Core Rule Set* est hétérogène. Certaines règles, plus globales, remplissent bien leur rôle alors que d'autres ne semblent pas réellement utiles tant elles sont spécifiques (filtre sur le nom d'une fonction PHP sans importance pour un attaquant, filtre pour une application tierce, etc.).

La création de règles n'est pas compliquée en soi et un administrateur peut facilement s'aider des règles existantes et du manuel de référence. Cependant, écrire des règles **efficaces** et **génériques** demande une excellente compréhension du produit.

L'analyse des journaux semble difficile dans la version actuelle du produit et peut être source d'erreurs. Cette lecture peut être simplifiée par le produit ModSecurity Console et l'*appliance*, commerciale, de Breach Security.

2.3.12.4. Notes et remarques diverses

Recommandation pour la mise en œuvre d'un relai inverse

Deux politiques de filtrage peuvent être distinguées :
– le filtrage par liste noire s'appuie sur une liste de signatures ou motifs réputés dangereux. Ces listes doivent faire l'objet de mises à jour rationnelles.
– le filtrage par liste blanche : les variables doivent être confrontées à des suites d'expressions rationnelles, caractérisant précisément en nature (caractères alphabétiques, numériques, alphanumériques, entiers, flottants,etc.) et en taille, le paramètre transmis.

La mise en œuvre d'une solution de filtrage au niveau d'un relais inverse configuré en liste blanche peut être progressive :

- mise en œuvre d'un filtrage des extensions autorisées ;
- mise en œuvre d'un filtrage des ressources autorisées, URI par URI ;
- mise en œuvre d'un filtrage de paramètres sur les ressources autorisées, avec une politique d'alerte, et non une politique de filtrage, afin d'évaluer les dysfonctionnements pouvant être engendrés par une telle solution de filtrage tierce ;
- mise en œuvre d'un filtrage de paramètres strict, avec filtrage, journalisation des requêtes interdites et lancement d'alertes, dans le cadre d'une détection d'intrusion opérationnelle.

Finalement, on notera que les approches « liste noire » et « liste blanche » sont généralement complémentaires, notamment au niveau du filtrage de champs complexes (champs de texte libre), par exemple afin de se prémunir des attaques de type XSS (Cross-Site Scripting).

2.4. Analyse de la résistance des mécanismes cryptographiques

Sans objet.

2.5. Analyse du générateur d'aléas

Sans objet.

3. La certification

3.1. Conclusion

L'évaluation a été conduite conformément aux règles en vigueur, avec la compétence et l'impartialité requise pour un centre d'évaluation agréé.

Ce certificat atteste que le produit « ModSecurity v2.5.12 » soumis à l'évaluation répond aux caractéristiques de sécurité spécifiées dans sa cible de sécurité [ST], aux limites près indiquées dans le présent rapport.

3.2. Restrictions d'usage

Ce certificat porte sur le produit spécifié au chapitre 1.2 du présent rapport de certification.

L'utilisateur du produit certifié devra s'assurer du respect des objectifs de sécurité sur l'environnement d'exploitation spécifiés dans la cible de sécurité [ST] et suivre les recommandations énoncées dans le présent rapport de certification.

Annexe 1. Références documentaires du produit évalué

[CDS]	*Cible de sécurité CSPN v1.2* ;
[RTE]	*Rapport Technique d'Evaluation* – K08-CD-PRE-738-2009 Version 1.2
[GUIDES]	ModSecurity 2.5 Guide utilisateur Version 0.81 ModSecurity® Reference Manual, Breach Security Version 2.5.11 Mod Security Core Rule Set, OWASP

Annexe 2. Références liées à la certification

Décret 2002-535 du 18 avril 2002 modifié relatif à l'évaluation et à la certification de la sécurité offerte par les produits et les systèmes des technologies de l'information.	
[CSPN]	Certification de sécurité de premier niveau (CSPN) des technologies de l'information, version 2. 4, phase expérimentale, n° 915/SGDN/DCSSI/SDR/CCN du 25 avril 2008. Critères pour l'évaluation de sécurité de premier niveau des technologies de l'information, phase expérimentale, version 1. 4. Méthodologie d'évaluation en vue de la CSPN et contenu attendu du RTE, phase expérimentale, version 1. 3. Documents disponibles sur www. ssi. gouv. fr