LABORATOIRE SECURITE

Sécuriser les applications web de l'entreprise

Mise en place de ModSecurity pour Apache

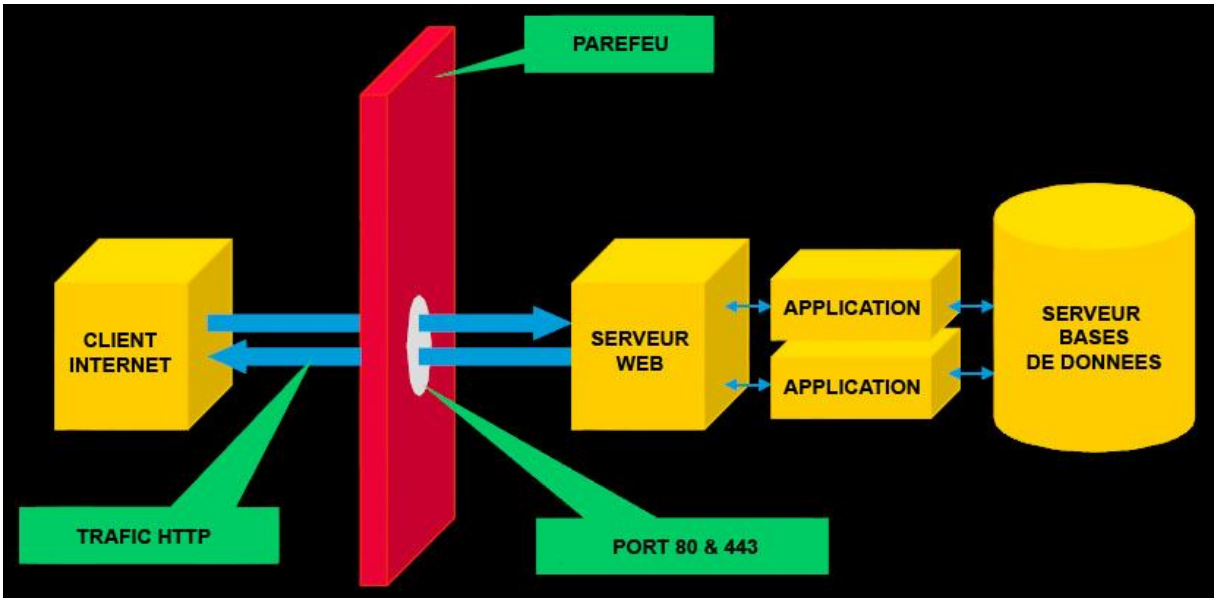
Julien SIMON - 61131

Sommaire

Présentation de la situation actuelle

Que ce soit l'activité en ligne d'une banque, un magasin en ligne, le portail d'un fournisseur, toutes les applications web sont accessibles, aussi bien à leurs clients qu'à des utilisateurs malveillants, de manière permanente à cause de la nature des services présents sur Internet qui ne s'arrête jamais.

Les attaques comme les injections SQL, le Cross-Site Scripting ou le piratage de sessions sont reconnus comme étant des vulnérabilités de l'application web en elle-même et ne dépendent pas du réseau. Pour cette raison, les équipements de sécurités traditionnels comme les pare-feux ou les IDS/IPS sont incapables de protéger contre ces types d'attaques ou ne permettent pas d'offrir une protection assez complète.

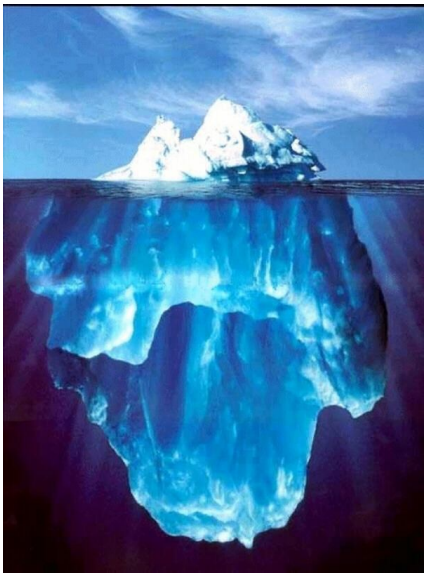


Le pare-feu bloque bien tous les ports d'accès au serveur mais le port 80 et 443 permettent de passer celui-ci laissant face à l'application et ses vulnérabilités qui en cas de défaillances peut permettre d'accéder au serveur de bases de données.

D'un point de vue technique le problème réside dans le fait que le web, et spécialement le protocole http, n'a pas été conçu pour les applications complexes que nous connaissons aujourd'hui.

Un nombre important de vulnérabilités puisent leur origine à cet endroit : par exemple, le protocole http établie des connexions sans en assurer le suivi (Stateful connections), les sessions ou la gestion du suivi des connexions doivent être implantées de façon sécuriser par l'application elle-même. Ces vulnérabilités sont augmentées par le degré de complexité des Frameworks et des dernières technologies devenues à la mode comme l'AJAX.

Le problème est encore trop souvent caché par les entreprises qui ne souhaitent en aucun cas effrayer leurs clients des éventuelles brèches dans leur système d'information.



La situation pourrait être décrite tel un iceberg. En effet on ne voit souvent qu'une petite partie de celui-ci flotter.

Dès lors il devient difficile depuis l'extérieur d'évaluer la sécurité des données clients entres les entreprises.

Nous allons voir à travers ce document comment aider l'entreprise à se protéger de ces nouveaux types d'attaques à l'aide des WAF, Web Application Firewall, présenter les attaques les plus répandues et procéder à l'installation d'une solution WAF : ModSecurity

Qu'est ce qu'un WAF ?

Un WAF est définit comme une solution de sécurité pour les applications web qui d'un point de vue technique ne dépend pas de l'application en elle-même par l'intermédiaire d'un code particulier. Il permet de se prémunir d'attaques qui n'auraient pas été détectées à temps durant le processus de test et de validation de l'application. De plus, le délai de mise à jour d'un WAF est extrêmement rapide comparé au temps de produire une nouvelle version de l'application corrigée lors de la découverte d'une vulnérabilité.

Les WAF peuvent se présenter comme une solution intégrée directement au serveur Web (Apache ou IIS) ou comme un équipement matériel à part entière se plaçant entre le pare-feu et le ou les serveurs Web.

Quelques solutions de WAF :

ModSecurity, www.modsecurity.org,
Projet Open Source développé par Breach

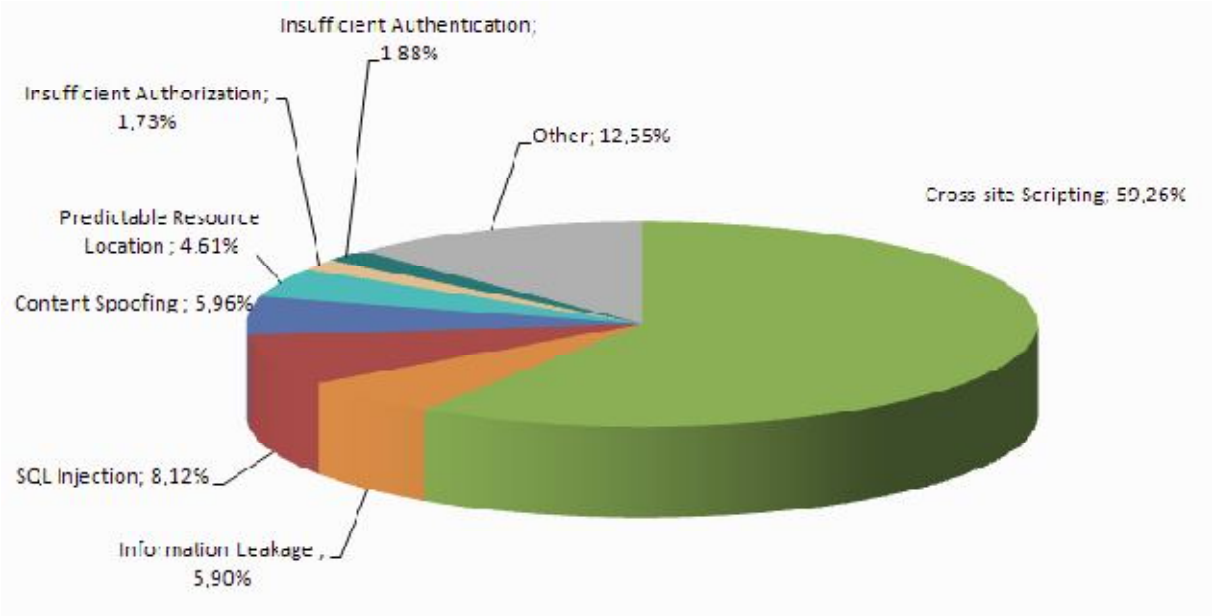
BinarySec, www.binarysec.fr,
Solution propriétaire

rWeb, www.denyall.com,
Solution propriétaire

Dans la suite du document, nous allons installer ModSecurity et sa console de gestion afin de voir à quoi ressemble dans la pratique un WAF.

Quels sont les attaques les plus répandues ?

Selon l'étude réalisée sur l'année 2007 par le WASC, Web Application Security Consortium, les attaques les plus fréquentes sont le Cross-site Scripting et les injections SQL :



Le Cross-site Scripting ou XSS, a pour principe d'injecter sur une page d'un site existant un code JavaScript que le client va exécuter sur sa machine en local qui va soit permettre une élévation de privilèges soit récupérer des informations d'identifications tels que des cookies permettant à l'attaquant de se connecter aux services web sous l'identité du client et se servir du compte sans son autorisation.

Cette attaque peut être contrée par l'utilisation de langage de programmation comme JAVA ou .NET avec les Taglibs et l'activation des filtres XSS. Le WAF détectera la tentative et la bloquera respectant l'objectif de sécurité d'intégrité des données entre le serveur et le client au détriment de la navigation en renvoyant une erreur bloquante au client.

L'injection SQL est une attaque qui consiste à exploiter les champs de saisis présent dans un formulaire sur un site web. La menace provient de la manière dont sont traités les champs au sein du code source de l'application. Un mauvais filtrage des caractères spéciaux permettant d'échapper certains caractères aura pour effet de construire une requête totalement différente de celle prévue par les développeurs. Le résultat permettra à l'attaquant de s'introduire dans le système d'information et de récupérer n'importe quelle information sur les clients.

Cette attaque peut être évitée par l'utilisation de requêtes paramétrées ou de procédures stockées directement au sein du code de l'application web. Le WAF détectera la tentative de passage de caractères non autorisés permettant de construire une nouvelle requête et bloquera la tentative.

Pourquoi choisir ModSecurity ?

ModSecurity est un projet issu du monde open source qui a débuté en 2002. Aujourd'hui disponible en version 2.5.x, il a acquit des performances honorables, tant en termes de stabilité et de traitements. Ce projet est accessible gratuitement et il est soutenu par Breach Security qui développe des solutions dédiés aux entreprises clef en main avec ses machines dédiés à la tâche.

La philosophie de ModSecurity est aussi très transparente. Ainsi rien n'est effectué implicitement puisque tout est accessible dans la configuration. Bref les utilisateurs contrôlent point par point le système et sans surprises.

Ces fonctionnalités sont :

- La génération de fichiers journaux du trafic http
- L'analyse et la détection en temps réel des attaques
- Un moteur de règles totalement personnalisables
- La prévention des attaques et la mise en jour juste à temps

ModSecurity propose en plus de son WAF, une console de gestion des incidents qui permet d'envoyer des rapports par mél pour permettre de minimiser les temps d'intervention à la détection d'une attaque ou d'une nouvelle vulnérabilité dans l'application web.

Passons maintenant à la mise en place de ModSecurity afin de tester ces fonctionnalités.

Installation de ModSecurity

L'installation de ModSecurity que nous allons réaliser a été effectué sur la distribution DEBIAN ETCH, DEBIAN LENNY, et UBUNTU 8.x. Cette procédure ne décrit pas l'installation d'un serveur Apache néanmoins elle conviendra aussi bien à une installation du serveur web Apache en version 1 ou 2.x.

Toutes les commandes données nécessitent d'être super utilisateur « root ».

Avant de pouvoir récupérer le service ModSecurity, un certain nombre de paquets doivent être installé pour permettre à ModSecurity de s'exécuter. Leur installation se lance à l'aide de la commande suivante :

```
aptitude install libxml2-dev liblua5.1-0 lua5.1 apache2-threaded-dev build-essential
```

Une fois installé, on télécharge la dernière version de ModSecurity directement sur le site. Pour accéder aux téléchargements, vous devez créer un compte gratuitement.

Pour le télécharger directement sur le serveur dans le répertoire temporaire, vous pouvez utilisez les commandes suivantes, en ayant récupérer au préalable l'url sur l'archive :

```
cd /tmp

wget https://bsn.breach.com/downloads/t=92bc195870d2caf0bacd6e6e8c5aa76f/modsecurity-apache/modsecurity-apache_2.5.7.tar.gz
```

Une fois ModSecurity récupéré, il faut de le désarchiver à l'aide de la commande suivante

```
tar zxvf modsecurity-apache_2.5.x.tar.gz
```

On se place à l'intérieur du répertoire crée lors de l'extraction :

```
cd modsecurity-apache_2.5.x/apache2/
```

On lance le processus de compilation de ModSecurity et on demande que les fichiers soient installés automatiquement :

```
./configure && make && make install
```

Configuration de ModSecurity

Il faut maintenant configurer apache pour permettre au module ModSecurity d'être chargé. On crée donc un fichier de chargement :

```
nano  /etc/apache2/mods-available/mod-security2.load
```

Dans le fichier, on ajoute les lignes suivantes afin d'indiquer à Apache où se trouve les bibliothèques requises par ModSecurity

```
LoadFile /usr/lib/libxml2.so
LoadFile /usr/lib/liblua5.1.so.0
LoadModule security2_module /usr/lib/apache2/modules/mod_security2.so
```

On quitte le fichier et on active les modules ModSecurity et unique_id :

```
a2enmod mod-security2
a2enmod unique_id
```

On spécifie dans la configuration du module ModSecurity où se trouve les fichiers ses propres fichiers de configuration

```
nano /etc/apache2/conf.d/mod-security2.conf
```

On ajoute cette ligne puis on quitte :

```
Include /etc/modsecurity2/*.conf
```

Il faut maintenant créer les répertoires qui contiendront les fichiers journaux de ModSecurity

```
mkdir /etc/modsecurity2
mkdir /etc/modsecurity2/logs
touch /etc/modsecurity2/logs/modsec_audit.log
touch /etc/modsecurity2/logs/modsec_debug.log
```

On copie les règles de ModSecurity et ses fichiers de configurations dans le répertoire dédié :

```
cp /tmp/modsecurity-apache_2.5.x/rules/*.conf /etc/modsecurity2
```

On indique l'emplacement pour les fichiers journaux dans le fichier de configuration de ModSecurity

```
nano /etc/modsecurity2/modsecurity_crs_10_config.conf
```

Trouver la ligne SecDebugLog logs/modsec_debug.log
Remplacer par SecDebugLog /etc/modsecurity2/logs/modsec_debug.log
Trouver la ligne SecAuditLog logs/modsec_audit.log
Remplacer par SecAuditLog /etc/modsecurity2/logs/modsec_audit.log

Une fois la configuration terminée, quittez le fichier de configuration et lancez l'utilitaire de validation de configuration Apache pour vérifier les éventuelles erreurs :

```
apache2ctl configtest
```

Cette commande doit vous retourner « Syntax OK ». On peut maintenant redémarrer le serveur Apache pour activer les modifications que nous venons d'effectuer.

```
/etc/init.d/apache2 restart
```

Pour vérifier que ModSecurity est lancé vous pouvez taper la commande suivante :

```
cat /var/log/apache2/error.log | grep ModSecurity

[Sat Dec 20 17:38:12 2008] [notice] ModSecurity for Apache/2.5.7 (http://www.modsecurity.org/) configured.
```

Test de ModSecurity

Pour tester les règles de protection de ModSecurity, je vous propose d'écrire un petit script php (nécessite que PHP soit correctement installé et configuré sur votre serveur) afin de déclencher le système de blocage.

Ouvrer un bloc notes et copier coller le script ci-dessous afin de tester les protections. Le but de ce script est d'aller lire le fichier le fichier que l'on passe en paramètre via l'url.

```
<?php
$fichier = $_GET['fichier'];
$contenu = file ($fichier);
echo "Contenu du fichier $fichier ";
var_dump $contenu;
?>
```

Envoyez-le sur votre serveur, puis appelez-le avec votre navigateur internet de cette façon :

http://nomdedomaine.fr/test.php?fichier=/etc/passwd

Le résultat sans ModSecurity donnerai accès au contenu du fichier /etc/passwd qui contient tous les utilisateurs et les mots de passe du serveur.

Avec ModSecurity, une erreur est retournée :

```
Method Not Implemented

GET to /test.php not supported.

Apache/2.2.0 (Fedora) Server at debian Port 80
```

Si l'on analyse les fichiers journaux la tentative de lecture est enregistrée comme ceci :

```
cat /etc/modsecurity2/logs/modsec_audit.log

--d83f296c-A--
[20/Dec/2008:17:59:42 +0100] SU0kfn8AAQEAACqThKgAAAAA 192.168.1.10 58826 192.168.1.17 80
--d83f296c-B--
GET /test.php?fichier=/etc/passwd HTTP/1.1
User-Agent: Opera/9.62 (Windows NT 6.0; U; fr) Presto/2.1.1
Host: ubuntu
Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif,
image/x-xbitmap, */*;q=0.1
Accept-Language: fr-FR,fr;q=0.9,en;q=0.8
Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
Connection: Keep-Alive, TE
TE: deflate, gzip, chunked, identity, trailers

--d83f296c-F--
```

```
HTTP/1.1 501 Method Not Implemented
Allow: TRACE
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 226
Connection: close
Content-Type: text/html; charset=iso-8859-1

--d83f296c-H--
Message: Access denied with code 501 (phase 2). Pattern match
"(?:\b(?:\.(?:ht(?:access|passwd|group)|www_?acl)|global\.asa|httpd\.conf|boot\.ini)\b|\/etc\/)"
at ARGS:fichier. [file "/etc/modsecurity2/modsecurity_crs_40_generic_attacks.conf"] [line "114"] [id
"950005"] [msg "Remote File Access Attempt"] [data "/etc/"] [severity "CRITICAL"] [tag
"WEB_ATTACK/FILE_INJECTION"]
Action: Intercepted (phase 2)
Stopwatch: 1229792382006796 3614 (891 2417 -)
Producer: ModSecurity for Apache/2.5.7 (http://www.modsecurity.org/); core ruleset/1.6.1.
Server: Apache/2.2.9 (Ubuntu) PHP/5.2.6-2ubuntu4 with Suhosin-Patch
```

Le fichier nous permet de savoir exactement ce qu'il s'est produit, quelle était l'adresse ip de l'attaquant et quelle mesure a été prise par ModSecurity.

Attention toutefois l'activation de ModSecurity peut provoquer des faux positifs en détectant certaines de vos applications comme étant corrompues ou essayant d'utiliser des méthodes non autorisées.

Pour résoudre ces problèmes, il faut analyser les fichiers journaux d'Apache et de ModSecurity afin d'affiner la détection et le blocage soit en modifiant (recommandé) le code de l'application web soit en ajoutant une exception. Une exception désactivera l'analyse sur un fichier donné fragilisant la sécurité et l'efficacité du WAF.

Pour définir une exception sur une page particulière, il vous faut récupérer l'id de l'erreur dans les fichiers journaux d'Apache (/var/log/apache2/error.log) puis définir dans la configuration (/etc/apache2/apache2.conf) les directives suivantes :

```
<LocationMatch "/chemin/vers/le/script.php">

        SecRuleRemoveById 1234567

</LocationMatch>
```

Exemple avec notre script PHP :

```
[error] [client 192.168.1.10] ModSecurity: Access denied with code 501 (phase 2). Pattern match "(?:\\b(?:\\.(?:ht(?:access|passwd|group)|www_?acl)|global\\.asa|httpd\\.conf|boot\\.ini)\\b|\\/etc\\/)" at ARGS:fichier. [file "/etc/modsecurity2/modsecurity_crs_40_generic_attacks.conf"] [line "114"] [id "950005"] [msg "Remote File Access Attempt"] [data "/etc/"] [severity "CRITICAL"] [tag "WEB_ATTACK/FILE_INJECTION"] [hostname "ubuntu"] [uri "/test.php"] [unique_id "SU0kfn8AAQEAACqThKgAAAAA"]
```

Dans cet exemple, les champs intéressant (colorés en rouge) sont :

- client 192.168.1.10
  - Adresse IP de l'attaquant
- file "/etc/modsecurity2/modsecurity_crs_40_generic_attacks.conf"
  - Le fichier de règles utilisé
- id "950005"
  - l'identifiant de l'erreur générée
- msg "Remote File Access Attempt"
  - Le nom de l'attaque bloquée
- uri "/test.php"
  - le fichier incriminé

Votre serveur web est maintenant protégé contre les attaques les plus répandues sur la toile actuellement mais attention tout de même certaines failles restent toujours accessible et nécessite donc de continuer à analyser le code de l'application afin de limiter les risques d'attaques et de piratages des informations.

WEBOGRAPHIE

- http://www.modsecurity.com/
- http://www.webappsec.org/projects/wafec/
- http://binarysec.com/
- http://en.wikipedia.org/
- http://www.owasp.org/index.php/Main_Page