Protect your web assets with **Open Source WAF**

Thousands of website get hacked every day due to misconfiguration or vulnerable code. Web Application Firewall (WAF) is one of the best ways to protect your website from online threats.

If your website is available on the Internet then you can use online tools to scan a website for vulnerability to get an idea how secure your website is.

Don't worry if it's intranet website, you can use Nikto web scanner open source.

Commercial WAF can be expensive and if you are looking for the free solution to protect your website using WAF then the following open source Web Application Firewall can be helpful.

List of WAF

- 1. ModSecurity
- 2. IronBee
- 3. NAXSI
- 4. WebKnight
- 5. Shadow Daemon

# 1. ModSecurity

ModSecurity by TrustWave is one of the most popular web application firewalls and it supports Apache HTTP, Microsoft IIS & Nginx.

ModSecurity free rules will be helpful if you are looking for the following protection.

- Cross-site scripting
- Trojan
- Information leakage
- SQL injection
- Common web attacks
- Malicious activity

ModSecurity doesn't have a graphical interface and if you are looking for the one then you may consider using WAF-FLE. It let you store, search and view the event in a console.

# Description

The OWASP ModSecurity CRS provides protections in the following attack/threat categories:

- SQL Injection (SQLi)
- Cross Site Scripting (XSS)
- Local File Inclusion (LFI)
- Remote File Inclusion (RFI)
- Remote Code Execution (RCE)
- PHP Code Injection
- HTTP Protocol Violations
- Shellshock
- Session Fixation
- Scanner Detection
- Metadata/Error Leakages
- Project Honey Pot Blacklist
- GeoIP Country Blocking

# 2. IronBee

IronBee is a security framework to build your own WAF. IronBee is not available in the binary package yet so you got to compile from the source and tested on the following OS.

- CentOS
- Fedora
- Ubuntu
- OS X

It’s highly portable and very lightweight web security framework.

## 3. NAXSI

NAXSI is Nginx Anti-XSS & SQL Injection. So as you can guess this is only for Nginx web server and mainly target to protect from cross-site scripting & SQL injection attacks.

NAXSI filter only GET and PUT request and default configuration will act as a DROP-by-default firewall so you got to add the ACCEPT rule to work properly.

## 4. WebKnight

WebKnight WAF is for Microsoft IIS. It’s an ISAPI filter that secures your web server by blocking bad requests. WebKnight is good for securing from the following.

- Buffer overflow
- Directory transversal
- Character encoding
- SQL injection
- Blocking bad robots
- Hotlinking
- Brute force
- And much more…

In a default configuration, all blocked requests are logged and you can customize based on your needs. **WebKnight 3.0** got admin web interface where you can customize the rules and perform administration tasks including statistics.

## 5. Shadow Daemon

Shadow Daemon detect, record and prevent web attacks by filtering request from malicious parameters. It comes with an own interface where you can perform administration and manage this WAF. It supports PHP, Perl & Python language framework.

It can detect the following attacks.

- SQL injection
- XML injection
- Code injection
- Command injection
- XSS
- Backdoor access
- Local/remote file inclusion

Open source is free but you don’t get enterprise support means you need to rely on your expertise and community support. So if you are looking for the commercial WAF then you may refer the following one.

- CloudFlare (cloud-based)
- Incapsula (cloud-based)
- F5 ASM
- TrustWave ModSecurity commercial rules
- SUCURI (cloud-based)
- Akeeba Admin tools (for Joomla)

I hope this helps this helps you an idea about open source web application firewall for the various platform.

### mod Security

**modSecurity™** es un firewall de aplicaciones Web embebible que ejecuta como módulo del servidor web Apache, provee protección contra diversos ataques hacia aplicaciones Web y permite monitorizar tráfico HTTP, así como realizar análisis en tiempo real sin necesidad de hacer cambios a la infraestructura existente.[1]

modSecurity™ para Apache es un producto desarrollado por Breach Security. modSecurity™ está disponible como Software Libre bajo la licencia GNU General Public License, a su vez, se encuentra disponible bajo diversas licencias comerciales.

# Historia

- Ivan Ristic especialista en seguridad web, creó modSecurity™ en el año 2002. Abordó el desarrollo de esta aplicación después de haber utilizado durante un año y medio SNORT para monitorear tráfico Web y llegar a la conclusión de que necesitaba una herramienta que le permitiera especificar reglas más complejas y la capacidad de realizar acciones relacionadas específicamente con el tráfico HTTP.[2]

- En septiembre de 2006 Breach Security adquirió Thinking Stone y modSecurity™. [Para más información, ver Breach Acquires modSecurity Open Source Vendor Thinking Stone].

- A mediados de noviembre de 2006 se lanzó al mercado la versión 2.0 de modSecurity™.

# Funcionamiento

modSecurity™ es una herramienta para detección y prevención de intrusos para aplicaciones Web.

El módulo cuenta con diversas funcionalidades: Firewall de Aplicaciones con mod_Security – Características y Funcionalidades"

- Filtrado de Peticiones: los pedidos HTTP entrantes son analizados por el módulo mod_security antes de pasarlos al servidor Web Apache, a su vez, estos pedidos son comparados contra un conjunto de reglas predefinidas para realizar las acciones correspondientes. Para realizar este filtrado se pueden utilizar expresiones regulares, permitiendo que el proceso sea flexible.
- Técnicas antievasión: las rutas y los parámetros son normalizados antes del análisis para evitar técnicas de evasión.

    - Elimina múltiple barras (**//**)
    - Elimina directorios referenciados por si mismos (**./**)
    - Se trata de igual manera la **** y la **/** en Windows.
    - Decodificación de URL
    - Reemplazo de bytes nulos por espacios (**%00**)

- Comprensión del protocolo HTTP: al comprender el protocolo HTTP, ModSecurity™ puede realizar filtrados específicos y granulares.
- Análisis Post Payload: intercepta y analiza el contenido transmitido a través del método POST.
- Log de Auditoría: es posible dejar traza de auditoría para un posterior análisis forense.
- Filtrado HTTPS: al estar embebido como módulo, tiene acceso a los datos después de que estos hayan sido descifrados.
- Verificación de rango de Byte: permite detectar y bloquear shellcodes, limitando el rango de los bytes.

# Versión 2.x

Funcionalidades incorporadas en la versión 2.x:[3]

- Cinco fases de procesamiento, incluyendo: encabezados del pedido (*request headers*), cuerpo del pedido (*request body*), encabezados de respuesta (*response headers*), cuerpo de respuesta (*response body*) y almacenamiento en bitácora (*logging*).
- Opciones de transformación por regla.
- Variables transaccionales.
- Persistencia de datos (utilizado en seguimientos de direcciones IP, sesiones de aplicación, y usuarios de aplicación).
- Soporte para ranking de anomalías y correlación básica de eventos (los contadores pueden ser automáticamente decrementados con el paso del tiempo, las variables pueden expirar).
- Soporte para aplicaciones Web e IDs de sesión.
- Soporte para XML (parseo, validación, XPath).
- bloqueo de IP

# SEGURIDAD PERIMETRAL

¿Qué es la Seguridad Perimetral?

La **Seguridad Perimetral** hace referencia a la integración de elementos y sistemas tanto electrónicos como mécanicos para la protección de perímetros físicos, detección de tentativas de intrusión y/o disuasión en instalaciones sensibles. Algunos de los sistemas utilizados son radares, videosensores, vallas, cables sensores, barreras de microondas e infrarrojos.

Por ejemplo, el acceso a un Centro de Proceso de Datos (CPD) está dotado de muchas de las medidas de seguridad descritas.

Desde el punto de vista de Seguridad Lógica hace referencia al **Perímetro de la Red Corporativa**

- Cortafuegos (Filtrado de Tráfico, Filtrado de Contenido, Filtros AntiSpam, Control de la Navegación, Clasificación de Contenidos, Detección y Prevención de Intrusiones)
- Cortafuegos a Nivel de Aplicación (Control de Aplicaciones, Calidad de Servicio, Priorización de Tráfico)
- Cortafuegos de Aplicaciones Web (Protección de Aplicaciones Web, Control de Extremo a Extremo, Cumplimiento de Requerimientos Normativativos "PCI-DSS, LOPD, SOX")
- Ataques de Denegación de Servicios (DDoS)
- Correlación de Eventos (Gestión de Eventos de Aplicación, Seguridad y Sistemas)
- Balanceadores/Optimizadores de Tráfico
- Accesos VPN/SSL (Con/Sin Autenticación Fuerte)
- Accesos VPN/IPSec (Con/Sin Autenticación Fuerte)

The **OWASP ModSecurity Core Rule Set (CRS)** is a set of generic attack detection rules for use with ModSecurity or compatible web application firewalls. The CRS aims to protect web applications from a wide range of attacks, including the OWASP Top Ten, with a minimum of false alerts.

The Core Rule Set provides protection against many common attack categories, including:

| | |
|---|---|
| SQL Injection (SQLi) | HTTPoxy |
| Cross Site Scripting (XSS) | Shellshock |
| Local File Inclusion (LFI) | Session Fixation |
| Remote File Inclusion (RFI) | Scanner Detection |
| Remote Code Execution (RCE) | Metadata/Error Leakages |
| PHP Code Injection | Project Honey Pot Blacklist |
| HTTP Protocol Violations | GeoIP Country Blocking |

The Core Rule Set is free software, distributed under Apache Software License version 2.

CRS 3 includes many coverage improvements, plus the following new features:

- Over 90% reduction of false alerts in a default install
- A user-defined *Paranoia Level* to enable additional strict checks
- Application-specific exclusions for *WordPress Core* and *Drupal*
- *Sampling mode* runs the CRS on a user-defined percentage of traffic
- SQLi/XSS parsing using libinjection embedded in ModSecurity

For a full list of changes in this release, see the CHANGES document.

## Installation

CRS 3 requires an Apache/IIS/Nginx web server with ModSecurity 2.8.0 or higher.

Our GitHub repository is the preferred way to download and update CRS.

**HTTPS** `git clone https://github.com/SpiderLabs/owasp-modsecurity-crs.git`

**SSH** `git clone git@github.com:SpiderLabs/owasp-modsecurity-crs.git`

After download, copy `crs-setup.conf.example` to `crs-setup.conf`. Optionally edit this file to configure your CRS settings. Then include the files in your webserver configuration:

```
Include /.../crs-setup.conf
Include /.../rules/*.conf
```

For detailed installation instructions, see the INSTALL document. Also review the CHANGES and KNOWN_BUGS documents.

You can update the rule set using the included script `util/upgrade.py`.

### Handling False Positives and Advanced Features

Advanced features are explained in the `crs-setup.conf` and the rule files themselves. The `crs-setup.conf` file is generally a very good entry point to explore the features of the CRS.

We are trying hard to reduce the number of false positives (false alerts) in the default installation. But sooner or later, you may encounter false positives nevertheless.

Christian Folini's tutorials on installing ModSecurity, configuring the CRS and handling false positives provide in-depth information on these topics.

## Configurar ModSecurity con Apache en Ubuntu 14.04 y Debian 8

ModSecurity es un cortafuegos de aplicación web gratuita (WAF) que trabaja con Apache, Nginx y IIS. Soporta un motor de reglas flexibles para realizar operaciones simples y complejas y viene con una base regla Set (CRS) que tiene reglas para la inyección de SQL, cross site scripting, Troyanos, agentes de usuario mal, secuestro de sesión y un montón de otras hazañas. Para Apache, se carga como un módulo adicional que lo hace fácil de instalar y configurar.

# Paso 1 — Instalar ModSecurity

En este paso, vamos a instalar ModSecurity.
Primero, actualizar los archivos de índice de paquete.

- sudo apt-get update

Entonces, instalar ModSecurity.

- sudo apt-get install libapache2-mod-Real2 - y

Se puede verificar que el módulo de ModSecurity se cargó utilizando el siguiente comando.

- sudo apachectl - M | grep –color Real2

Si la salida de Lee `security2_module (shared)`, Esto indica que el módulo fue cargado.
Instalación de ModSecurity incluye un archivo de configuración recomendada que tiene que cambiar el nombre de.

- sudo mv /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf

Finalmente, recarga de Apache.

- sudo service apache2 reload

Se creará en el directorio de logs de Apache en un nuevo archivo de registro de ModSecurity `/var/log/apache2/modsec_audit.log`.

# Paso 2 — Configurar ModSecurity

Fuera de la caja, ModSecurity no hace nada porque necesita reglas para trabajar. En este paso, primero habilitamos algunas directivas de configuración.

Para encontrar y sustituir las directivas de configuración en este paso, vamos a usar `sed`, un editor de flujo. Usted puede leer el `sed` serie de tutoriales para aprender más acerca de la herramienta.

### Directivas básicas para activar

El archivo de configuración de ModSecurity predeterminado se establece en `DetectionOnly`, que registra las solicitudes de acuerdo a la norma los partidos y no bloquea nada. Esto se puede cambiar editando el `modsecurity.conf` archivo y

modificar el `SecRuleEngine` Directiva. Si usted se esta probando en un servidor de producción, cambiar esta Directiva sólo después de probar todas sus reglas.

- sudo sed -i “s/SecRuleEngine DetectionOnly/SecRuleEngine en /” /etc/ModSecurity/ModSecurity.conf

los `SecResponseBodyAccess` Directiva configura cuerpos de respuesta sean almacenadas (es decir. leído por ModSecurity). Esto sólo es necesario si se requiere protección y detección de fugas de datos. Por lo tanto, dejando en utilizará recursos de gota y también aumentar el tamaño de archivo de registro, por lo tanto apagará.

- sudo sed -i “s/SecResponseBodyAccess en/SecResponseBodyAccess de /” /etc/ModSecurity/ModSecurity.conf

Directivas opcionales para modificar

Existen otras directivas que desea personalizar editando `/etc/modsecurity/modsecurity.conf`. los `SecRequestBodyLimit` y `SecRequestBodyNoFilesLimit` directivas de limitan los datos máximos que pueden ser publicados en la aplicación web.

En particular, el `SecRequestBodyLimit` Directiva especifica el tamaño máximo de datos de correos. Si nada más es enviado por un cliente el servidor responderá con un 413 Entidad de solicitud demasiado grande error. Si la aplicación web no tiene ninguna carga de archivos, Este valor se puede dejar como está. El valor configurado previamente especificado en el archivo de configuración es 13107200 bytes (12,5 MB). Si desea cambiar este valor, Busque la línea siguiente `modsecurity.conf`:

Cambio Directiva opcional 'modsecurity.conf'

```
SecRequestBodyLimit 13107200
```

Del mismo modo, el `SecRequestBodyNoFilesLimit` límites el tamaño de los datos POST menos el archivo de ficheros. Este valor debe ajustarse al mínimo posible reducir la susceptibilidad a ataques denegación de servicio (DoS) cuando alguien está enviando solicitud cuerpos de tamaños muy grandes. El valor preconfigurado en el archivo de configuración es 131072 bytes (128KB). Si desea cambiar este valor, Busque la línea siguiente `modsecurity.conf`:

Cambio Directiva opcional 'modsecurity.conf'

```
SecRequestBodyNoFilesLimit 131072
```

Es una directiva que afecta al rendimiento del servidor `SecRequestBodyInMemoryLimit`. Esta directiva es bastante explica por sí mismo; Especifica la cantidad de “cuerpo de la solicitud” datos (publicado) deben conservarse en la memoria (RAM), nada que más se colocarán en el disco duro (como intercambio). Porque las gotas usan SSDs, Esto no es un problema. Sin embargo, Esto se puede cambiar si tienes RAM de sobra. El valor preconfigurado para esta directiva es 128 KB. Si desea cambiar este valor, Busque la línea siguiente `modsecurity.conf`:

Cambio Directiva opcional 'modsecurity.conf'

```
SecRequestBodyInMemoryLimit 131072
```

# Paso 3 — Pruebas de inyección SQL

Antes de configurar algunas reglas, crearemos un script en PHP que es vulnerable a inyección SQL para probar protección de ModSecurity.

Nota: este es un script de login PHP básico no saneamiento de manejo o forma de sesión. Se utiliza sólo como un ejemplo para probar la inyección de SQL y las reglas de ModSecurity. Se eliminarán antes del final del tutorial.

Primero, acceso al prompt de MySQL.

- MySQL -u root -p

Aquí, crear una base de datos MySQL llamado **muestra** y conectarse a ella.

- crear base de datos muestra;
- conectar muestra;

Crear una tabla con algunas credenciales, el nombre de usuario **Sammy** y la contraseña **contraseña**.

- crear table usuarios (usuario VARCHAR (100), contraseña VARCHAR(100));
- Inserte en la values('sammy','password') de los usuarios;

Finalmente, salir el prompt de MySQL.

- Crear usuario 'asterisk' @ '* localhost' identificado 'y0ur-passw0rd';

Siguiente, crear el script de inicio de sesión en raíz de Apache.

- sudo nano /var/www/html/login.php

Pegue el siguiente script PHP en el archivo. Asegúrese de cambiar la contraseña de MySQL en el script a continuación a la que anterior colocar de modo que la secuencia de comandos se puede conectar a la base de datos:

/var/www/html/login.php

```
<html>
<body>
<?php
    if(isset($_POST['login']))
    {
        $username = $_POST['username'];
        $password = $_POST['password'];
        $con = mysqli_connect('localhost','root','your_mysql_password','sample');
        $result = mysqli_query($con, "SELECT * FROM `users` WHERE username='$username' AND password='$password'");
        if(mysqli_num_rows($result) == 0)
            echo 'Invalid username or password';
        else
            echo '<h1>Logged in</h1><p>This is text that should only be displayed when logged in with valid credentials.</p>';
    }
    else
    {
?>
        <form action="" method="post">
            Username: <input type="text" name="username"/><br />
            Password: <input type="password" name="password"/><br />
            <input type="submit" name="login" value="Login"/>
        </form>
<?php
    }
?>
</body>
</html>
```

Este script mostrará un formulario de inicio de sesión. Abra su navegador y vaya a `http://your_server_ip/login.php` para verlo. Si introduce el par correcto de credenciales, e. g. Sammy en el **Nombre de usuario** campo y la contraseña en el **Contraseña** campo, Usted verá el mensaje **Este es el texto que sólo obtiene muestra cuando ha iniciado sesión con credenciales válidas**. Si navega de regreso a la pantalla de login y use Credenciales incorrectas, Usted verá el mensaje **Nombre de usuario no válido o contraseña**.

El siguiente trabajo es intentar una inyección SQL para omitir la página de inicio de sesión. Introduzca lo siguiente para el campo Nombre de usuario.

Nombre de usuario SQL inyección

```
' or true --
```

Tenga en cuenta que debe haber un espacio después de `--` para esta inyección trabajar. Deje vacío el campo de contraseña y presione el botón de inicio de sesión. La secuencia de comandos muestra el mensaje para los usuarios autenticados. En el siguiente paso, evitaremos esto.

# Paso 4 — Establecimiento de normas

En este paso, vamos a establecer algunas reglas de ModSecurity.

## Permitiendo a la CRS

Para facilitar las cosas, hay un montón de reglas que ya están instalados junto con ModSecurity. Estos se llaman CRS (conjunto de reglas de base) y se encuentran en el `/usr/share/modsecurity-crs` directorio. Para cargar estas reglas, necesitamos configurar Apache para leer `.conf` archivos en estos directorios, tan abierta la `security2.conf` archivo para la edición.

```
sudo nano /etc/apache2/mods-enabled/security2.conf
```

Añadir las dos siguientes directivas, resaltado en rojo, interior antes de la última línea en el archivo (`</IfModule>`).

Security2.conf actualizado

```
        IncludeOptional /etc/modsecurity/*.conf
        IncludeOptional "/usr/share/modsecurity-crs/*.conf"
        IncludeOptional "/usr/share/modsecurity-crs/activated_rules/*.conf"
</IfModule>
```

Guarde y cierre el archivo.

## Exclusión de directorios/dominios (opcionales)

A veces tiene sentido excluir un directorio particular o un nombre de dominio si se está ejecutando una aplicación, como phpMyAdmin, como ModSecurity bloqueará consultas SQL. También es mejor excluir backends de admin de aplicaciones CMS como WordPress. Si estás siguiendo este tutorial en un servidor nuevo, puede omitir este paso.

Para desactivar ModSecurity para un VirtualHost completa, Coloque las siguientes directivas dentro de la `<VirtualHost>[...]</VirtualHost>` bloquear en el archivo de host virtual.

```
<IfModule security2_module>
    SecRuleEngine Off
</IfModule>
```

Para omitir un directorio particular (por ejemplo, `/var/www/wp-admin`):

```
<Directory "/var/www/wp-admin">
    <IfModule security2_module>
        SecRuleEngine Off
    </IfModule>
</Directory>
```

Si no desea deshabilitar totalmente ModSecurity en un directorio, utilizar la `SecRuleRemoveById` Directiva para eliminar una regla particular o una regla cadena especificando su identificación.

```
<LocationMatch "/wp-admin/update.php">
    <IfModule security2_module>
        SecRuleRemoveById 981173
    </IfModule>
</LocationMatch>
```

## Activar la regla de inyección SQL

Siguiente, activamos el archivo de reglas de inyección SQL. Los archivos de reglas requerido deben ser un enlace simbólico a `activated_rules` directorio, que es similar a la de Apache `mods-enabled` directorio. Cambiar a la `activated_rules` directorio.

- CD /usr/share/modsecurity-crs/activated_rules /

Luego cree un enlace simbólico de la `modsecurity_crs_41_sql_injection_attacks.conf` archivo.

- sudo ln -s... /base_rules/modsecurity_crs_41_sql_injection_attacks.conf .

Finalmente, recarga de Apache para las reglas para tomar efecto.

- sudo service apache2 reload

Ahora abra la página de inicio de sesión que creamos anteriormente y trate de usar la misma consulta de inyección de SQL en el campo de username. Porque hemos cambiado la `SecRuleEngine` Directiva para `On` en el paso 2, un **403 Prohibido** aparece error. (If `SecRuleEngine` se dejó a la `DetectionOnly` opción, la inyección será exitosa, pero el intento se registrarán el `modsec_audit.log` archivo.)

Debido a esta secuencia de comandos de inicio de sesión PHP pretende probar ModSecurity, se debe eliminar ya que el examen se realiza.

- sudo rm /var/www/html/login.php

# Paso 5 — Sus propias reglas de escritura

En esta sección, vamos a crear una cadena de la regla que bloquea la solicitud si se escriben algunas palabras comúnmente asociados con el spam en un formulario HTML.

Primero, vamos a crear un script PHP que recibe la entrada de un cuadro de texto y lo muestra al usuario de ejemplo. Abrir un archivo llamado `form.php` para la edición.

- sudo nano /var/www/html/form.php

Pegue el código siguiente:

/var/www/html/Form.php

```
<html>
    <body>
        <?php
            if(isset($_POST['data']))
                echo $_POST['data'];
            else
            {
        ?>
                <form method="post" action="">
                        Enter something here:<textarea name="data"></textarea>
                        <input type="submit"/>
                </form>
        <?php
            }
        ?>
    </body>
</html>
```

Reglas personalizadas pueden ser añadidas a cualquiera de los archivos de configuración o colocan en directorios de ModSecurity. Ponemos nuestras reglas en un nuevo archivo independiente denominado `modsecurity_custom_rules.conf`.

```
sudo nano /etc/modsecurity/modsecurity_custom_rules.conf
```

Pegar lo siguiente en este archivo. Son las dos palabras que nos estamos bloqueando **blockedword1** y **blockedword2**.

modsecurity_custom_rules.conf

```
SecRule REQUEST_FILENAME "form.php" "id:'400001',chain,deny,log,msg:'Spam detected'"
SecRule REQUEST_METHOD "POST" chain
SecRule REQUEST_BODY "@rx (?i:(blockedword1|blockedword2))"
```

La sintaxis para `SecRule` es `SecRule VARIABLES OPERATOR [ACTIONS]`. Aquí se utilizó la acción de la cadena para que coincida con las variables `REQUEST_FILENAME` con `form.php`, `REQUEST_METHOD` con `POST`, y `REQUEST_BODY` con la expresión regular `(@rx)` cadena `(blockedword1|blockedword2)`. los `?i:` hace un minúsculas. En un partido exitoso de estas tres reglas, el `ACTION` es negar y registro con el `msg "Spam detected."` La acción de cadena simula el AND lógico para que coincida con las tres reglas.

Guarde el archivo y volver a cargar Apache.

```
sudo service apache2 reload
```

Abierto `http://your_server_ip/form.php` en el navegador. Si introduce texto que contengan blockedword1 o blockedword2, Usted verá un 403 Página.

Porque este script de formulario PHP pretende probar ModSecurity, se debe eliminar ya que el examen se realiza.

- sudo rm /var/www/html/form.php

## Conclusión

En este tutorial, han aprendido cómo instalar y configurar ModSecurity, y añadir reglas personalizadas. Para aprender más, Usted puede comprobar fuera de la documentación oficial de ModSecurity.

## Seguridad en Apache: modSecurity

**modSecurity** es un firewall de aplicaciones Web embebible bajo licencia GNU que se ejecuta como módulo del servidor web Apache, provee protección contra diversos ataques hacia aplicaciones Web y permite monitorizar tráfico HTTP, así como realizar análisis en tiempo real sin necesidad de hacer cambios a la infraestructura existente. **modSecurity** filtra ataques por XSS, SQL Injection, comportamientos anómalos en protocolos, robots, troyanos, LFI ... incorporando además reglas específicas para algunos de los gestores de contenido más populares como Joomla o Wordpress. Soporta integración con ModProxy por lo que podemos proteger aplicaciones desplegadas en otros servidores gracias a esta integración. Además **modSecurity** cuenta con una consola de administración que permite recopilar registros de monitorización y alertas en tiempo real así como de opciones automatizadas de mantenimiento, entre otras características. (ModSecurity Console)

# Instalación de modSecurity

A continuación vamos a describir el proceso de instalación de modSecurity 2.7 en Debian, en Ubuntu está disponible en los repositorios. Las descargas necesarias podemos realizar desde la web oficial del proyecto http://www.modsecurity.org/ **1.- Descargamos las fuentes y descomprimimos:**

```
wget https://www.modsecurity.org/tarball/2.7.5/modsecurity-apache_2.7.5.tar.gz
tar -zxvf modsecurity-apache_2.7.5.tar.gz
```

**2.- Instalamos algunas dependencias necesarias:**

```
apt-get install apache2-threaded-dev libxml2-dev libcurl4-gnutls-dev -y
```

**3- Compilamos e instalamos:**

```
cd modsecurity-apache_2.7.5
./configure
make
make install
```

**4.- Cargamos el módulo:** Creamos el siguiente fichero

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

con el siguiente contenido:

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

**5.- Activamos modSecurity:** Antes de activar modSecurity debemos activar unique_id

```
a2enmod unique_id
```

y activamos modSecurity ahora:

```
a2enmod mod-security2
```

**6.-Configuramos las reglas:** Una vez hemos instalado modSecurity es necesario establecer las reglas que establecerán las condiciones del filtrado de métodos. Descargamos las reglas de filtrado en /etc/apache2:

```
cd /etc/apache2
apt-get install git
git clone https://github.com/SpiderLabs/owasp-modsecurity-crs
```

Una vez descargado veremos varios directorios además de diversas utilidades. Para una primera aproximación renombraremos modsecurity_crs_10_config.conf.example a modsecurity_crs_10_config.conf:

```
cp modsecurity_crs_10_config.conf.example modsecurity_crs_10_config.conf
```

Para finalizar editamos apache2.conf con el siguiente contenido: vim /etc/apache2/apache2.conf

```
   SecRuleEngine On
   SecRequestBodyAccess On
   SecResponseBodyAccess Off
   SecDebugLog /var/log/apache2/modsec_debug.log
   SecDebugLogLevel 1
   SecAuditEngine RelevantOnly
   SecAuditLogRelevantStatus ^5
   SecAuditLogParts ABIFHZ
   SecAuditLogType Serial
   SecAuditLog /var/log/apache2/modsec_audit.log
   SecRequestBodyLimit 131072
   SecRequestBodyInMemoryLimit 131072
   SecResponseBodyLimit 524288
   SecDataDir  /tmp/
   SecUploadDir /tmp/
   SecTmpDir /tmp/
   Include owasp-modsecurity-crs/modsecurity_crs_10_config.conf
   Include owasp-modsecurity-crs/base_rules/*.conf
   Include owasp-modsecurity-crs/optional_rules/*.conf
   Include owasp-modsecurity-crs/slr_rules/*.conf
```

Hay que ser cuidadoso con SecDataDir, SecUploadDir y SecTmpDir ya que las aplicaciones que suban ficheros y usen temporales no van a funcionar. De forma similar, hay que prestar atención a posibles configuraciones de los atributos SecRequestBodyLimit, SecRequestBodyInMemoryLimit, SecResponseBodyLimit que se suelen definir en el archivo apache2.conf. Estas configuraciones nos va determinar las características de subida de archivos a través de nuestras aplicaciones web. Si todo ha ido bien, pronto comenzaremos a ver en nuestro archivo de logs de Apache, decenas por no decir centenares de logs de este tipo:

```
[client 192.168.1.154] ModSecurity: Access denied with code 403 (phase 2).
Pattern match "^[\\\\d.:]+$" at REQUEST_HEADERS:Host.
[file "owasp-modsecurity-
crsbase_rules/modsecurity_crs_21_protocol_anomalies.conf"]
[line "98"] [id "960017"]
```

Si queremos ampliar el potencial de filtrado de nuestro módulo podemos acudir a http://www.gotroot.com/mod_security+rules y descargarnos libremente sus colecciones de reglas propias. Espero que esta primera introducción a modSecurity les sirva para reforzar la seguridad de sus servidores web Apache, tarea primordial en estos tiempos. Como siempre, gracias por leernos, y le invitamos a dejar un comentario si tiene alguna pregunta.