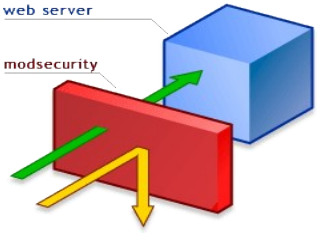
ISEC Lab #4

Firewall de Aplicación con Mod_Security



Christian Martorella
cmartorella@isecauditors.com

Índice

1.Introducción

Hoy en día las aplicaciones Web son el punto más vulnerable de la infraestructura de una empresa, cada día son más los ataques que se realizan en la capa HTTP: Sql Injection, XSS, Command Injection, Buffer Overflows, etc. La lista de vulnerabilidades va creciendo y las soluciones no. :(

Actualmente existen numerosas soluciones de Firewall para analizar y filtrar tráfico a nivel de red, pero a nivel de Aplicaciones, ¿qué opciones tenemos? Gratuitas y abiertas, muy pocas.

Si usted tiene una Aplicación Web pública y dinámica, las posibilidades de que sea vulnerable a alguno de estos ataques es muy alta. Quizá una simple comilla sea el comienzo de una pesadilla...

2.¿Qué es y para que sirve?

Mod_Security es un módulo de Apache que brinda detección y prevención de intrusos al servidor Web Apache; también a los productos que realizan estas tareas comúnmente se los llama "Firewall de Aplicaciones".

El funcionamiento es similar al de un Sistema de Detección de Intrusos (IDS) el cual es utilizado para analizar el tráfico de red con el fin de detectar anomalías, pero con la diferencia que este trabaja en el nivel de HTTP y lo comprende realmente muy bien.

Gracias a esto puede hacer cosas que son simples desde el punto de vista del protocolo HTTP, pero muy difícil para un IDS clásico.

Lo más importante es que además de detectar, tiene la funcionalidad de prevenir, si encuentra peticiones con carga maliciosa puede rechazar la petición basada en las reglas que incorpora el Mod_Security.

3.Ventajas

- Una de las mayores ventajas que tiene el Mod_Security es la posibilidad de proteger complejas aplicaciones donde la modificación del código fuente para poder securizarlas sea difícil, se necesite mucho tiempo , sea muy caro, o simplemente “no se pueda”.

- Se puede utilizar para proteger foros, blogs, wikis, portales de comentarios SPAM,etc.

- Si tenemos un servidor web o una aplicación que es vulnerable a cierto ataque/bug podemos protegerlo con el Mod_Security hasta que salga el parche o la actualización del mismo para solucionar el problema.

- Es OpenSource, y gratuito :)

- Fácil de configurar

- Eficiente

- Curva de aprendizaje muy rápida

- Podemos evitar un alto número de ataques con un pocas lineas de configuración.

- Se pueden crear reglas muy específicas, optimizando así el rendimiento del Mod_Security.

4.Características y Funcionalidades

El módulo cuenta con las siguientes funcionalidades:

-**Filtrado de Peticiones**: Las peticiones entrantes son analizadas antes de pasarlas al servidor web ó a cualquier otro módulo de Apache. Para realizar este filtrado podemos utilizar expresiones regulares, lo cual lo hace muy flexible.

-**Técnicas Anti-evasión**: Los paths y los parámetros son normalizados antes del análisis para evitar técnicas de evasión.

- Elimina múltiple barras (//)
- Elimina directorios referenciados por si mismos (./)
- Tratamiento de \ y / de manera igual (solo en Windows)
- Decodificación de URL
- Reemplazo de bytes nulos por espacios (%00)

-**Comprensión del protocolo HTTP**: Al comprender el protocolo HTTP, puede realizar filtrados muy específicos y granulares. Podemos analizar un campo específico de un formulario X de una página en particular.

-**Post Payload análisis**: Intercepta y analiza el contenido transmitido a través del método POST.

-**Audit Logging**: Es posible loguear con detalle (incluidas las peticiones POST) para un posterior análisis Forense.

-**HTTPS Filtering**: Al estar embebido como módulo tiene acceso a los datos despues de que estos hayan sido desencriptados.

-**Compressed content Filtering**: Al igual que la funcionalidad anterior, tiene acceso a los datos después de la descompresión.

-**Byte range verification:** Sirve para detectar y bloquear shellcodes, esto se puede lograr limitando el rango de los bytes.

Como podemos ver son muchas las opciones que tiene, y los controles que este puede efectuar.

5.Instalación

Ahora que conocemos las funcionalidades del Mod_Security, procederemos a su instalación.

El módulo se puede descargar a través de CVS, aquí se puede obtener la última versión disponible la cual está en etapa de desarrollo y puede resultar inestable; y la otra opción es descargarlo de la página web en sus 2 versiones, código fuente o binarios (los cuales solo están disponibles para windows).

En nuestro caso hemos utilizado la versión código fuente como recomienda el autor del módulo, el servidor en este caso es un Linux Ubuntu y Apache 2.0.52

Antes que nada debemos comprobar si tenemos instaladas las herramientas de desarrollo de Apache, las cuales nos proporcionan el programa apxs que será el encargado de compilar el módulo. En el caso de Ubuntu el paquete se llama “Apache-dev”.
Primero debemos bajar la última versión estable del Mod_Security disponible en "http://www.modsecurity.org/download/index.html"

Luego descomprimir la misma, y una vez que estemos dentro del directorio creado, nos cambiamos al directorio perteneciente a la rama de Apache que estemos utilizando, en este caso "Apache2" y una vez dentro de este directorio debemos ejecutar:

```
laramies@salvacion:~$ apxs -cia modsecurity.c
```

Este comando compilará el Mod_Security como módulo de Apache, y modificará la configuración del Apache para cargarlo cuando el servicio del Apache se inicie.

En este momento el módulo esta cargado en el Apache, pero al no tener una configuración no es de ninguna utilidad.

6.Configuración Básica

Para que comience a ejercer su función debemos agregar algunas lineas a la configuración del Apache, a continuación veremos un ejemplo simple:

```
<IfModule Mod_Security.c>
# Activamos el Mod_Security
SecFilterEngine On

# Escanear el contenido de la petición POST
SecFilterScanPOST On

# Escanear la respuesta de la petición (si se quiere evitar mostrar ciertos mensajes de error)
SecFilterScanOutput On

# Chequear codificación URL
SecFilterCheckURLEncoding On

# Chequear Codificación Unicode
SecFilterUnicodeEncoding On

#Esta opción debe estar activada solo si la Aplicación utiliza codificación Unicode.
#En cualquier otro caso puede interferir con la operatoria normal del sitio web.
SecFilterCheckUnicodeEncoding Off

#Permitir solo cierto valores de los bytes.
#Hay que tener en cuenta cuales son los caracteres que se utilizan en nuestro sistema.
#En este caso estamos permitiendo todos
SecFilterForceByteRange 1 255

#SQL Injection, controles muy básicos para esta vulnerabilidad.
SecFilter "delete[[:space:]]+from"
SecFilter "insert[[:space:]]+into"
SecFilter "select.+from"
SecFilter ".*['%;\"]+.*"


# Aquí evitaremos ataques simples de XSS
SecFilter "<(.|\n)+>"
SecFilter "<[[:space:]]*script"

#Controlo el Campo Nombre del Formulario para que no sea mayor a 6 caracteres, protección contra #Buffer
Overflow, si nuestro parámetro nombre es vulnerable.
SecFilterSelective ARG_nombre ".{6,}" "redirect:http://www.google.es"

#Nuestra aplicación es muy compleja y depende del RegisterGlobals, pero hay un formulario de autenticación que #se
puede saltar enviando la variable valido=1 o ok=1
SecFilterSelective "ARG_valido|ARG_ok" "1"

#Controlamos que se envíen los dos cabezeras (HTTP_USER_AGENT y HTTP_HOST) en las peticiones,
#generalmente los atacantes y algunas herramientas de caneo no enviran estas cabezeras.
SecFilterSelective "HTTP_USER_AGENT|HTTP_HOST" "^$"
```

```
# Prohibimos subir ficheros
SecFilterSelective "HTTP_CONTENT_TYPE" multipart/form-data)"

# Si alguien quiere acceder a /admin/administrar.php lo redirigimos a www.google.com
Secfilter “/admin/administrar.php” redirect:http://www.google.com

#Regla que controla la respuesta del servidor web, supongamos que hay un error en el código de la aplicación que
#nos da el famoso error del SQLServer exponiendo la consulta que ha fallado, pues con esta regla podemos hacer que
devuelva una página de error controlada.
SecFilterSelective OUTPUT "ODBC SQL Server
Driver”redirect:http://www.mypage.com/error.php

</IfModule>
```

Tengamos en cuenta que estas reglas son solo un ejemplo, cada aplicación tendrá sus propias reglas acorde a la tecnología que utilice, servidor web en la cual esta instalada, etc.

Como podemos ver es muy simple agregar reglas, y proteger recursos específicos, podemos protegernos desde Xss, Bufer Overflows, Sql Injection hasta de evitar que un campo de un formulario tenga el valor X.

También hay scripts que traducen las reglas del snort a reglas de Mod_Security, con esta transformación tendríamos cubiertos la mayoría de los ataques web, pero serian muy genéricas y seguramente estaríamos perdiendo rendimiento, pero no esta demás probarlo para ver cuales genera y de que manera, y luego podemos utilizar las que mejor se adapten a nuestra Aplicación.

Es muy importante tener en cuenta que cuanto más reglas y más genéricas sean estas, tendremos mayor impacto en la rendimiento del sistema.

7.Mi propio Firewall de aplicación por 0€

Hasta aquí hemos visto como proteger el mismo servidor Apache donde instalábamos el Mod_Security, ahora veremos que podemos hacer si tenemos otro tipo de servidor como por ejemplo un IIS?, o si tenemos 2 ó más servidores web distintos? todavía es posible protegerlos con Mod_Security.

Supongamos que tenemos dos servidores web, un Iplanet y un IIS, lo que vamos a hacer para proteger a ambos, es montar un Apache +Proxy Inverso + Mod_Security.

Con esta combinación logramos obtener lo que sería nuestro propio Firewall de Aplicaciones. El mismo se ubicaría de cara a internet y sería el que atendería las peticiones, las analizaría con el Mod_Security y luego el módulo de proxy enviaría las peticiones válidas a nuestros servidores Iplanet e IIS.

Las ventajas que presenta montar este sistema son las siguientes:

- Un solo punto de ingreso por lo consiguiente un solo punto de control.
- Ocultamiento de topología de red, solo es visible el servidor proxy desde el exterior.
- Mayor rendimiento, descargamos al servidor web de la tarea de analizar las peticiones, así como también es posible finalizar el canal SSL en nuestro proxy, y enviar las peticiones en claro a nuestros servidores web.
- Aislamiento de la red: Con nuestro proxy añadimos otra capa de firewall, en lugar de tener varios Servidores web y sistemas operativos expuestos a exterior, solo tendremos uno, nuestro Firewall de Aplicaciones.
- Nuevamente, el coste :)

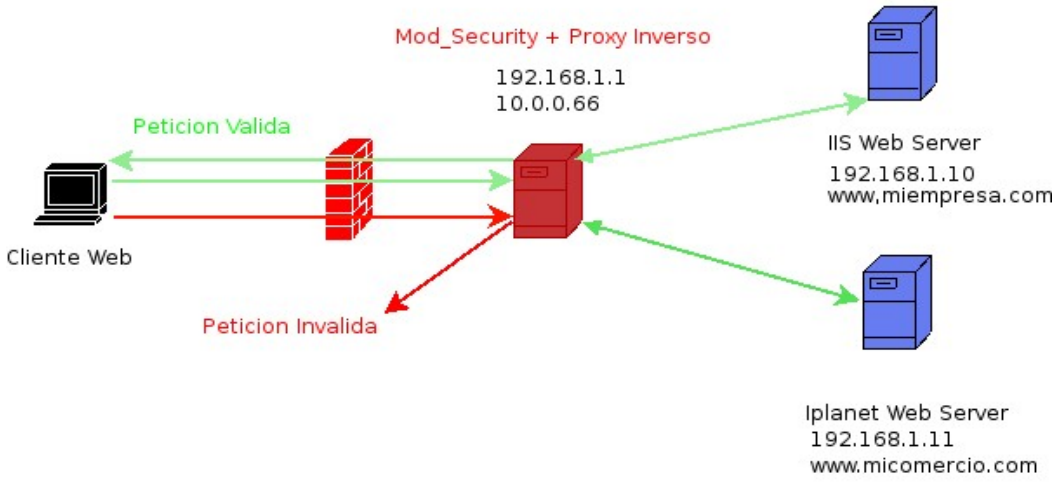
Pero también tenemos algunas desventajas:

- Punto único de fallo, si falla nuestro firewall de aplicación se anulan todos los servicios que este protege. Igualmente esto podemos solventarlo con un cluster de varios Firewall de Aplicaciones, así cuando falla uno otro toma el control.
- Añade complejidad a nuestro sistema.

Los requerimientos para montar nuestro Firewall de aplicación son mínimos:

- Servidor Web Apache
- Módulo mod_proxy y mod_proxy_http
- Módulo Mod_Security
- Módulo mod_ssl, en caso de trabajar con HTTPS.

Ejemplo de nuestra red:



La configuración básica del Apache de nuestro Firewall de Aplicaciones tendría que tener las siguientes lineas:

```
<VirtualHost www.miempresa.com>
ServerName www.miempresa.com
#Rechazamos peticiones de proxy, para que no sea un proxy abierto
ProxyRequests Off
ProxyPass / http://192.168.1.10/
ProxyPassReverse / http://192.168.1.10/
</VirtualHost>

<VirtualHost www.micomercio.com>
ServerName www.micomercio.com
ProxyRequests Off
ProxyPass / http://192.168.1.11/
ProxyPassReverse / http://192.168.1.11/
</VirtualHost>
```

Con esta simple configuración logramos que nuestro Firewall de Aplicación acepte peticiones para www.miempresa.com y www.micomercio.com y las reenvíe al servidor correspondiente tras analizarlas con el Mod_Security. En cuanto a la configuración del Mod_Security para estos VirtualHost, estamos heredando las reglas que definimos anteriormente. Igualmente es posible evitar que se hereden y definir así reglas específicas para cada servidor; la directiva para evitar heredar la configuración es:

```
SecFilterInheritance Off
```

Nota: Con esta configuración, al utilizar proxy inverso, la directiva para cambiar la identificación del Servicio `SecServerSignature "Microsoft II$/1.0"`, no va a funcionar debido a la naturaleza del módulo mod_proxy, el cual devolverá la verdadera versión de los servidores Web que el mod_Security protege.

Igualmente esto se puede solucionar utilizando el módulo mod_headers; y agregando la siguiente directiva a la configuración del principal del Apache:

```
Header set Server “Microsoft II$/1.0”
```

8.Conclusiones

Como pudimos ver en este artículo, la configuración del Mod_Security es muy simple y sus aporte a nuestra infraestructura de seguridad es muy alta.

Es muy ambicioso compararlo con dispositivos específicamente desarrollados para esta tarea, pero realmente la función básica de proteger una aplicación la cumple, y además de hacerlo con coste 0 realmente lo hace muy bien.

El proyecto está activo, y en cada versión se agregan funciones nuevas y se mejoran otras, es importante ir controlando las nuevas versiones.

Cada vez más gente esta utilizando esta solución para proteger sus Aplicaciones Web.

9.Referencias:

Web Security Appliance With Apache and mod_security, Ivan Ristic, SecurityFocus: http://www.securityfocus.com/infocus/1739

Mod Security Manual, Ivan Ristic: http://www.modsecurity.org/documentation/modsecurity-manual.pdf

Defending Web Services using ModSecurity, Shreeraj Shah, InfosecWriters: http://www.infosecwriters.com/texts.php?op=display&id=255