

Defensa Avanzada de Aplicaciones Web con ModSecurity

Christian Martorella (cmartorella@isecauditors.com),
Daniel Fernández Bleda (dfernandez@isecauditors.com)

Internet Security Auditors
www.isecauditors.com

Resumen. En este paper se presenta un módulo, bajo licencia GPL, de detección y prevención de ataques a aplicaciones web llamado `mod_security`. Éste componente, que funciona como un módulo de Apache, cuenta, entre sus funcionalidades la capacidad de detectar ataques muy documentados y que padecen habitualmente las aplicaciones como *SQL Injection*, *Cross Site Scripting*, etc. pero también puede ser utilizado para otras funciones como la detección de spam en blogs u otras aplicaciones de comunidades basadas en web, así como el hacer inútiles herramientas automáticas de identificación de recursos en una aplicación web. Además, se está trabajando, en conjunción con el creador principal del módulo para la inclusión de ciertas capacidades avanzadas que lo puedan equiparar, en prestaciones y capacidades a soluciones comerciales.

1 Introducción

ModSecurity es un módulo para Apache 1.3.x y Apache 2.0.x diseñado y creado con el objetivo de añadir una capa de seguridad a las aplicaciones web residentes en un sistema basado en dicho servidor web, pero con la capacidad de implementarse en entornos web heterogéneos, de forma transparente a las aplicaciones y servidores web que residan en una red, y que su uso, por tanto, no implique ningún cambio en la programación de estas.

El creador y mantenedor principal del proyecto es Ivan Ristic y la versión estable actual del proyecto es la 1.8.7, siendo la última publicada la 1.9dev3 (que es sobre la que se está trabajando para la inclusión de nuevas funcionalidades).

A parte de la versión que funciona como un módulo de Apache, también se está desarrollando una versión para Java según la especificación Servlet 2.3 (sobre esta versión no se ha realizado ninguna de las mejoras, pero se ha planteado su futura inclusión).

2 Situación actual de la seguridad web

La razón del por qué necesitan los creadores y usuarios de aplicaciones web un software como ModSecurity es sencilla: la mayor cantidad de ataques e intrusiones efectivas a los sistemas se realizan actualmente a través de vulnerabilidades las aplicaciones web en lugar de vulnerabilidades sobre servicios de red o Sistemas Operativos. Algunos sitúan esta proporción en 5 a 1, otros en 10 a 1.

Habitualmente, las aplicaciones web se desarrollan con un único objetivo: que funcionen correctamente y realicen las tareas que deben llevar a cabo de forma adecuada, incluso eficiente. El mayor problema es que esto no implica de ningún modo que todas estas tareas se realicen de forma segura, sobretodo cuando la seguridad se ve como un gasto y no un requerimiento.

Simplemente hay que analizar cuales son los últimos exploits publicados en puntos de referencia como Milw0rm para entender hasta donde llegan las implicaciones de la seguridad web: el 80% de los últimos exploits publicados explotan vulnerabilidades de aplicaciones web.

Otro de los problemas que habitualmente se encuentra es la falsa sensación de seguridad que pueden tener las organizaciones cuando han implantado sistemas de protección a nivel de red y estas ofrecen (o no) sistemas de firewall de aplicación. En la mayoría de casos, estos suelen ser muy primitivos o simples, no desempeñando correctamente la función real de firewall de aplicación.

Añadido a todas estas razones, además de ser la más importante para la mayoría de organizaciones y creadores, es el coste de las soluciones comerciales de Firewall de Aplicación. Esta es la razón por la que se decidió analizar algunas de las funcionalidades de soluciones comerciales y decidió añadir algunas de ellas en una solución sin coste de licencias, usable por todo aquel que lo desease.

3 Funcionalidades Actuales

Estas son algunas de las funcionalidades integradas dentro de ModSecurity actualmente:

3.1 Filtrado de Peticiones

Todas las peticiones que realizan los clientes web son filtradas antes de que sean procesadas por el servidor web, impidiendo que ataques o potenciales ataques lo alcancen, explotando, o intentado explotar vulnerabilidades existentes en la aplicación.

3.2 Filtrado de respuestas

Uno de los problemas más comunes de las aplicaciones es no disponer de una correcta gestión de errores, hecho que suele implicar la emisión al usuario (o atacante) de información sobre el sistema o sistemas implicados en la aplicación, toda esta información puede ser eliminada antes de alcanzar al cliente web.

3.3 Conocimiento del protocolo HTTP

Gracias a que ModSecurity “entiende” el protocolo HTTP, y no simplemente lo trata como un conjunto de caracteres ASCII, permite una granularidad de afinamiento en el filtrado mucho mayor que un simple control por patrones de ataque.

3.4 Detección de técnicas antievasión

Ataques comunes como retroceso de directorios, codificación de parámetros, URLs, etc. ya no son efectivos, dado que una de las primeras acciones que lleva a cabo ModSecurity es el filtrado de este tipo de contenidos, normalizando las peticiones que alcanzan el servidor web.

3.5 Análisis del contenido de peticiones POST

De igual forma, ModSecurity es capaz de analizar el *payload* de peticiones POST al servidor en búsqueda de ataques incluidos en este.

Estas son sólo algunas de las funcionalidades incorporadas actualmente en ModSecurity, de algunas de ellas se realizarán demostraciones reales en la presentación que se lleve a cabo.

4 Funcionalidades en Desarrollo

Como se ha comentado inicialmente, los autores de este *paper*, analizamos algunas de las soluciones comerciales existentes en el mercado con el fin de evaluar posibles mejoras sobre ModSecurity con el fin de equipararlo a éstas, siguiendo siempre la premisa de conseguir que Ivan Ristic las considerase lo suficientemente valiosas como para incluirlas en el código creado y mantenido por él como “*Built-in Features*”.

Estas son algunas de estas nuevas funcionalidades, más o menos finalizadas o en fase de testeo y debug, sobre las que se está trabajando:

4.1 Eliminación de comentarios

Una de las consecuencias habituales de modificaciones en el código fuente de las aplicaciones es que código antiguo, comentarios de sus desarrolladores, o más grave incluso, información de *debug*, no se elimine del código que acaba llegando al cliente web.

Una de las funcionalidades nuevas es la eliminación del código de comentarios HTML. Para conseguir esto de una forma más eficiente se ha empleado la librería libxml2 para el parseado del fichero HTML y de esta forma identificar de forma rápida este tipo de contenido, manteniendo código Javascript (que por compatibilidad se suele incluir entre los tags de comentarios HTML “`<!-- -->`”).

4.2 Firma de *cookies*

Algunos de los ataques que se pueden realizar a las aplicaciones es mediante la alteración de las *cookies* que se emplean en esta. Mediante la librería criptográfica cryptlib32, se ha implementado la incorporación de una firma criptográfica al valor de la *cookie* que permite conocer si esta ha sido alterada y de qué forma.

4.3 Firma de enlaces HTTP

Otro de los ataques comunes a las aplicaciones web es el escaneo para la identificación de recursos web mediante técnicas de diccionario o de fuerza bruta. Esto puede permitir descubrir, por ejemplo, directorios de administración incorrectamente protegidos o recursos abandonados en la aplicación. Mediante cryptlib32 se calcula y añade una firma criptográfica de aquellos links a los que puede navegarse de forma legítima, impidiendo en acceso a cualquier recurso cuya petición no incluya esta firma.

Dado que el uso de criptografía fuerte implica una penalización de rendimiento importante, se está decidiendo dónde es más eficiente implementar una firma en lugar de una criptografía de datos que deben recuperarse. El primer reduce significativamente la penalización dado que los algoritmos empleados pueden ser más simples, no perdiendo por ellos seguridad en el resultado.

Referencias

1. [ModSecurity: http://www.modsecurity.org](http://www.modsecurity.org)
2. [Cryptlib: http://www.cs.auckland.ac.nz/~pgut001/cryptlib](http://www.cs.auckland.ac.nz/~pgut001/cryptlib)
3. [Libxml2: http://www.xmlsoft.org](http://www.xmlsoft.org)