Trabajo - Instalación, Administración y Uso de un servicio en Linux

<Firewall de Aplicaciones Web (WAF):

Apache ModSecurity y ModEvasive >

Servicios Telemáticos

Departamento de Ingeniería Telemática

<Felipe Bueno Carranza>

ÍNDICE

1. Objetivos y Alcance

1.1. Introducción

Las aplicaciones web, a diferencia del software de escritorio y de las apps para los dispositivos móviles, están desarrolladas (o al menos deben estarlo) para poderse ejecutar en cualquier navegador y en cualquier equipo del planeta. Esta universalidad las hace mucho más vulnerables y dispuestas a sufrir ataques. Si además se trata de aplicaciones relacionadas con el comercio electrónico o la banca online, entonces el nivel de amenaza se incrementa notablemente. Debido a ello los propietarios de dichas aplicaciones se ven en la necesidad de proteger su contenido ante esos ataques. Uno de los métodos más usados para la protección de las aplicaciones web son las herramientas Web Application Firewall (WAF), las cuales se interponen entre la aplicación y el usuario que realiza la petición a la misma, inspeccionando el tráfico entrante en la capa 7 del modelo OSI (capa de aplicación) en busca de patrones de ataque y bloqueando la llegada del tráfico supuestamente malicioso a la aplicación web. Los WAF, pueden ser dispositivos hardware o software, que protegen contra la mayoría de los ataques de la clasificación OWASP TOP TEN (Inyección, Autenticación dañada, Exposición de datos sensibles, Entidades externas XML (XXE), Control de acceso dañado, Configuraciones incorrectas de seguridad, Secuencias de comandos entre sitios (XSS), Deserialización insegura, Usar componentes con vulnerabilidades conocidas, Registro y monitoreo insuficientes) [1].

1.2. Motivación y funcionalidad del servicio

Los ciberataques en la red incrementan y evolucionan día a día, por lo que toda medida de seguridad es poca. Es una obligación para las empresas mantenerse al día con tendencias de hacking para reconocer y detectar vulnerabilidades sus sistemas.
Contar con un firewall para aplicaciones web WAF ofrece una capa de seguridad y tranquilidad que garantiza la disponibilidad de tu sitio web. Imagina: si tu sitio web es hackeado ¿cuántas horas vas a desperdiciar tratando de encontrar el problema y solucionarlo? ¿Cuánto dinero podrías perder por tener tu sitio fuera de servicio?
Los WAF brindan una respuesta inteligente, basada en la configuración de seguridad web, a posibles amenazas que pueden afectar su red.
De esta forma los WAF están diseñados para ayudar a proteger su red de posibles amenazas que aún no se han identificado.

1.3. Documentación bibliográfica

[1] https://hackwise.mx/las-10-principales-vulnerabilidades-owasp-2020/

[2] https://es.wikipedia.org/wiki/Web_application_firewall

[3] https://es.wikipedia.org/wiki/Inyecci%C3%B3n_SQL

[4] https://es.wikipedia.org/wiki/Cross-site_scripting

[5] https://www.ionos.es/digitalguide/servidores/know-how/servidor-web-definicion-historia-y-programas/

[6] https://www.euskadi.eus/navegadores-web/web01-a2wz/es/

[7] https://www.cisco.com/c/es_es/products/security/firewalls/what-is-a-firewall.html

[8] https://www.osi.es/es/actualidad/blog/2018/08/21/que-son-los-ataques-dos-y-ddos

[9] Práctica 04 de la asignatura

[10] https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto

[11] https://developer.mozilla.org/es/docs/Web/HTTP/Status

[12] https://modsecurity.org/

[13] https://es.wikipedia.org/wiki/Mod_Security

[14] https://github.com/SpiderLabs/ModSecurity/

[15] https://github.com/jzdziarski/mod_evasive

2. Base Teórica

Se explicarán conceptos necesarios para una mejor comprensión del problema y la solución final del trabajo.

2.1. Descripción del servicio y conceptos implicados

Servidor Web: Los servidores web sirven para almacenar contenidos de Internet y facilitar su disponibilidad de forma constante y segura. Cuando visitas una página web desde tu navegador, es en realidad un servidor web el que envía los componentes individuales de dicha página directamente a tu ordenador. Esto quiere decir que para que una página web sea accesible en cualquier momento, el servidor web debe estar permanentemente online. [5]

Navegador Web: Un navegador web es un programa que permite ver la información que contiene una página web. El navegador interpreta el código, HTML generalmente, en el que está escrita la página web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar. [6]

Firewall: Un firewall es un dispositivo de seguridad de la red que monitoriza el tráfico entrante y saliente y decide si debe permitir o bloquear un tráfico específico en función de un conjunto de restricciones de seguridad ya definidas. [7]

Firewall de Aplicaciones Web (WAF): Es un tipo de firewall que supervisa, filtra o bloquea el tráfico HTTP hacia y desde una aplicación web. Se diferencia de un firewall normal en que puede filtrar el contenido de aplicaciones web específicas, mientras que un firewall de red protege el tráfico entre los servidores. Al inspeccionar el tráfico HTTP un WAF protege a las aplicaciones web contra ataques como los de inyección SQL, XSS y falsificación de petición de sitios cruzados (CSRF). [2]

Inyección SQL: Es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar operaciones sobre una base de datos.[3]

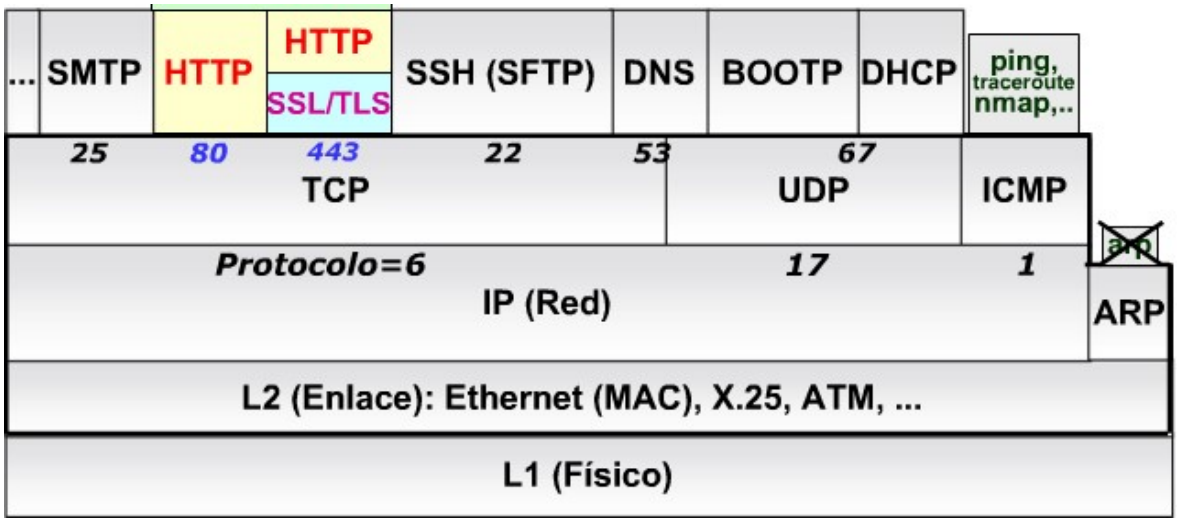
Cross-site scripting (XSS): Es un tipo de vulnerabilidad informática o agujero de seguridad típico de las aplicaciones Web, que puede permitir a una tercera persona inyectar en páginas web visitadas por el usuario código JavaScript o en otro lenguaje similar (ej: VBScript). Se puede evitar usando medidas como CSP, política del mismo origen, etcétera.[4]

Ataque por Denegación de Servicio (DoS): Se generan una cantidad masiva de peticiones al servicio desde una misma máquina o dirección IP, consumiendo así los recursos que ofrece el servicio hasta que llega un momento en que no tiene capacidad de respuesta y comienza a rechazar peticiones, esto es cuando se materializa la denegación del servicio. [8]

Ataque por Denegación de Servicio Distribuido (DDoS): se realizan peticiones o conexiones empleando un gran número de ordenadores o direcciones IP. Estas peticiones se realizan todas al mismo tiempo y hacia el mismo servicio objeto del ataque. Un ataque DDoS es más difícil de detectar, ya que el número de peticiones proviene desde diferentes IP´s y el administrador no puede bloquear la IP que está realizando las peticiones, como sí ocurre en el ataque DoS. [8]

2.2. Protocolos utilizados por el servicio

2.2.1. Protocolos Comunes



- HTTP-1.1 (RFC 7230-5), 2.0 (RFC 7540)
- SSL/TLS- (RFC 6101/RFC 5246)
- TCP- (RFC 793)
- IP- (RFC 791)

2.2.2. Protocolos Específicos del servicio

HTTP (HyperTextTransferProtocol), definido por el IETF, es el protocolo utilizado por los servidores y navegadores web para comunicarse entre sí. Su uso involucra habitualmente otros componentes web, tales como los descriptores de dirección URL o el lenguaje de marcado HTML (definido por el W3C).[9]

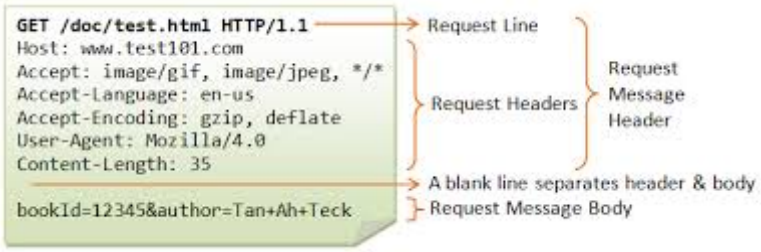
Aplicación	HTTP
Transporte	TCP
Red	IP

Mensaje Request

En la web, los clientes, como un navegador, por ejemplo, se comunican con los distintos servidores web con ayuda del protocolo HTTP, el cual regula cómo ha de formular sus peticiones el cliente y cómo ha de responder el servidor. El protocolo HTTP emplea varios métodos de petición diferentes, pero en este caso tan solo me centraré en los dos más comunes y conocidos.

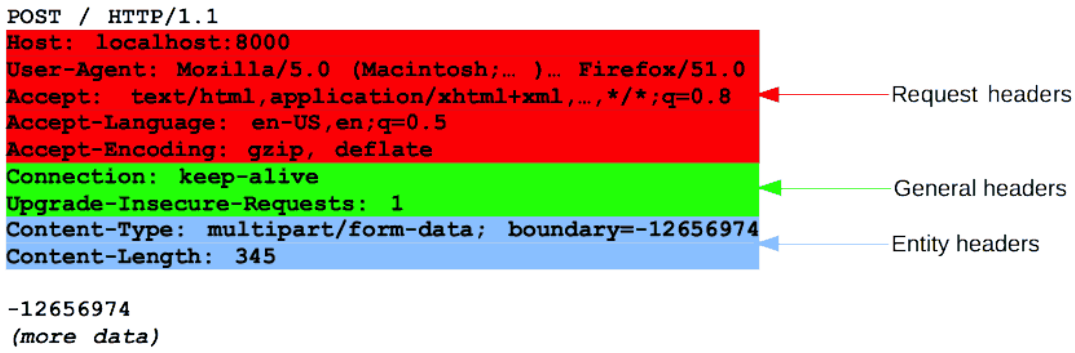
- Método GET

  El método GET solicita una representación del recurso especificado. Las solicitudes que usan GET solo deben recuperar datos y no deben tener ningún otro efecto. (Esto también es cierto para algunos otros métodos HTTP.) [10]



- Método POST

  RFC 2616. Envía datos para que sean procesados por el recurso identificado en la URI de la línea petición. Los datos se incluirán en el cuerpo de la petición. A nivel semántico está orientado a crear un nuevo recurso, cuya naturaleza vendrá especificada por la cabecera Content-Type. [10]



Mensaje Response

La respuesta o HTTP response es el mensaje que envía el servidor al cliente tras haber recibido una petición o HTTP request.

Los códigos de estado de respuesta HTTP indican si se ha completado satisfactoriamente una solicitud HTTP específica. Las respuestas se agrupan en cinco clases:

- Respuestas informativas (100–199),
- Respuestas satisfactorias (200–299),
- Redirecciones (300–399),
- Errores de los clientes (400–499),
- Errores de los servidores (500–599).

Los códigos de estado se definen en la sección 10 de RFC 2616. Puedes obtener las especificaciones actualizadas en RFC 7231. [11]

Diagrama de Secuencia de Mensajes de red

3. Evaluación de la implementación estudiada

Buscando soluciones al problema de seguridad en aplicaciones web expuesto anteriormente, en este apartado se detallarán aquellas más comunes además de las que serán estudiadas más adelante en este trabajo.

3.1. Descripción y características de la implementación de servicio estudiada

3.1.1. Breve Descripción de la solución adoptada

En este trabajo se adopta la solución de instalar los módulos *mod_security* y *mod_evasive* que ofrece el servidor web Apache.

Mod_Security

Funcionalidades	• Monitoreo de seguridad de aplicaciones en tiempo real y control de acceso • Registro de tráfico HTTP completo • Evaluación de seguridad pasiva continua • Fortalecimiento de aplicaciones web
Referencias	12,13,14
Autor	Ivan Ristic
Versión utilizada	2.9.2
Última versión	v3.0.4 (13/01/2020)

Mod_Evasive

Funcionalidad	Este módulo se encarga de gestionar una tabla hash con las IPs con accesos más frecuentes al servidor. Así, si detecta un número desmesuradamente alto de peticiones desde una misma IP, las anulará por un tiempo.
Referencias	15
Autor	Jonathan Zdziarski
Versión utilizada	1.10.1
Última versión	1.10.1 (17/01/2005)

Comentarios	Se procederá a la integración de las reglas OWASP-Core Rule Set (CRS) para proporcionar una protección base consistente ante el OWASP Top 10.

3.1.2. Equipamiento necesario

Hardware necesario.	Equipo cliente, Equipo servidor
Infraestructura de Red requerida.	Equipo servidor equipado con servidor web apache y módulos a trabajar. Equipo cliente con acceso a dicho servidor
Elementos Software requeridos para su instalación	Librerias: httpd-devel gcc make libxml2 pcre-devel libxml2-devel curl-devel git c++ flex bison yajl yajl-devel curl GeoIP-devel doxygen zlib-devel epel-release yum-utils libtool
S.O. concreto usado	Centos 7

3.1.3. Características y funcionalidades

Mod_Security

Funciona mediante la comparación del tráfico del servidor web contra un sistema de reglas. Dependiendo de su configuración, procederá al bloqueo o transformación del tráfico con contenido malicioso, generando un log de actividad (block mode), o únicamente detectando y generando un log del tráfico malicioso (detection mode). Su sistema de reglas se ha convertido en un estándar usado en diversas aplicaciones WAF comerciales.

Este módulo ofrece las siguientes funcionalidades:

- **Monitoreo de seguridad de aplicaciones en tiempo real y control de acceso**: En esencia, ModSecurity le brinda acceso al flujo de tráfico HTTP, en tiempo real, junto con la capacidad de inspeccionarlo. Esto es suficiente para el monitoreo de seguridad en tiempo real. Existe una dimensión adicional de lo que es posible a través del mecanismo de almacenamiento persistente de ModSecurity, que le permite rastrear los elementos del sistema a lo largo del tiempo y realizar la correlación de eventos. Puede bloquear de manera confiable, si así lo desea, porque ModSecurity utiliza el almacenamiento en búfer completo de solicitudes y respuestas.

- **Registro de tráfico HTTP completo**: Los servidores web tradicionalmente hacen muy poco cuando se trata de iniciar sesión por motivos de seguridad. Registran muy poco de forma predeterminada, e incluso con muchos ajustes, no puede obtener todo lo que necesita. Todavía tengo que encontrar un servidor web que pueda registrar datos completos de transacciones. ModSecurity le brinda la capacidad de registrar todo lo que necesite, incluidos los datos de transacciones sin procesar, que son esenciales para la ciencia forense. Además, puede elegir qué transacciones se registran, qué partes de una transacción se registran y qué partes se desinfectan.

- **Evaluación de seguridad pasiva continua**: La evaluación de seguridad se considera en gran medida un evento programado activo, en el que se contrata a un equipo independiente para intentar realizar un ataque simulado. La evaluación de seguridad pasiva continua es una variación de la supervisión en tiempo real, donde, en lugar de centrarse en el comportamiento de las partes externas, se centra en el comportamiento del sistema en sí. Es un tipo de sistema de alerta temprana que puede detectar rastros de muchas anomalías y debilidades de seguridad antes de que sean explotadas.

- **Fortalecimiento de aplicaciones web**: Uno de mis usos favoritos de ModSecurity es la reducción de la superficie de ataque, en la que usted reduce selectivamente las funciones HTTP que está dispuesto a aceptar (por ejemplo, métodos de solicitud, encabezados de solicitud, tipos de contenido, etc.). ModSecurity puede ayudarlo a hacer cumplir muchas restricciones similares, ya sea directamente o mediante la colaboración con otros módulos de Apache. Todos caen bajo el endurecimiento de las aplicaciones web. Por ejemplo, es posible solucionar muchos problemas de administración de sesiones, así como vulnerabilidades de falsificación de solicitudes entre sitios.

Una de las alternativas para establecer un esquema de seguridad más completo del que ofrece por defecto este módulo es la implementación de del OWASP ModSecurity Core Rule Set (CRS), el cual provee una serie de reglas de detección de ataques para la protección de cualquier aplicativo ubicado en un servidor de Hosting.

El OWASP ModSecurity CRS brinda protección contra los tipos de ataques más comunes detectados contra sitios en Internet (OWASP Top 10).

Estas reglas están organizadas según el "nivel de protección" que el administrador desee, a estos niveles se les llama *nivel de paranoia (Paranoia Level, PL).*

Con cada aumento del nivel de paranoia, el CRS habilita reglas adicionales brindándole un mayor nivel de seguridad. Sin embargo, niveles más altos de paranoia también aumentan la posibilidad de bloquear parte del tráfico legítimo debido a falsas alarmas (también denominadas falsos positivos o FP). Si usa niveles de paranoia altos, es probable que deba agregar alguna regla de exclusión para ciertas solicitudes y aplicaciones que reciben entradas complejas.

Niveles de paranoia

El nivel de paranoia 1 es el predeterminado. En este nivel, la mayoría de las reglas básicas están habilitados. PL1 se recomienda para principiantes, para instalaciones cubriendo muchos sitios y aplicaciones diferentes, y para configuraciones con requisitos de seguridad estándar.

El nivel 2 de paranoia incluye muchas reglas adicionales, por ejemplo, habilitar muchas protecciones de inyección SQL y XSS basadas en expresiones regulares, y agregando palabras clave adicionales revisadas para inyecciones de código. Se aconseja PL2 para usuarios moderados a experimentados que desean una cobertura más completa y para instalaciones con requisitos de seguridad elevados.

El nivel 3 de paranoia permite más reglas y listas de palabras clave, y ajustes límite de caracteres especiales utilizados. PL3 está dirigido a usuarios experimentados en el manejo de falsos positivos y en instalaciones con alta seguridad

El nivel 4 de paranoia restringe aún más los caracteres especiales. Se recomienda el nivel más alto para usuarios experimentados que protegen instalaciones con requisitos de seguridad muy elevados. En PL4 es probable que se produzca un número muy elevado de falsos positivos.

Sintaxis de las reglas

Una SecRule es una directiva como cualquier otra entendida por ModSecurity. La diferencia es que esta directiva es mucho más poderosa en lo que es capaz de representar. Por lo general, una SecRule se compone de 4 partes:

- Variables: indica a ModSecurity dónde buscar (a veces llamados objetivos).
- Operadores: indica a ModSecurity cuándo activar una coincidencia.
- Transformaciones: indica a ModSecurity cómo debe normalizar los datos variables.
- Acciones: indica a ModSecurity qué hacer si una regla coincide.

La estructura de la regla es la siguiente:

- SecRule VARIABLES "OPERADOR" "TRANSFORMACIONES, ACCIONES"

Además, cada regla debe incluir dos variables de marcador de posición actualizadas por el servicio WAF al publicar la regla.

- id: {{id_1}}: este campo se actualiza con un ID de regla único generado mediante el servicio WAF y que identifica una SecRule.

- ctl:ruleEngine={{mode}}: acción que se debe realizar cuando se cumplan los criterios de la SecRule, ya sea OFF, DETECT o BLOCK.

El servicio WAF puede realizar una acción en una solicitud HTTP cuando se cumplen los criterios de una regla de protección personalizada.

- DETECT: registra la solicitud cuando se cumplen los criterios de la regla de protección personalizada.

- BLOCK: bloquea la solicitud cuando se cumplen los criterios de la regla de protección personalizada.

- OFF: la regla de protección personalizada se desactiva y no se realiza ninguna acción.

Ejemplo de un formato de regla de protección personalizada:

```
SecRule REQUEST_COOKIES "regex matching SQL injection - part 1/2" \
    "phase:2,                                                     \
    msg:'Detects chained SQL injection attempts 1/2.',            \
    id: {{id_1}},                                                 \
    ctl:ruleEngine={{mode}},                                      \
    deny"
    SecRule REQUEST_COOKIES "regex matching SQL injection - part 2/2" \
    "phase:2,                                                     \
    msg:'Detects chained SQL injection attempts 2/2.',            \
    id: {{id_2}},                                                 \
    ctl:ruleEngine={{mode}},                                      \
    deny"
```

Mod_Evasive

Es un módulo para Apache que proporciona acciones evasivas en caso de un ataque HTTP DoS o DDoS o un ataque de fuerza bruta.

La detección se realiza creando una tabla hash dinámica interna de direcciones IP y URI, donde se introduce cualquier dirección IP que realice alguna de estas acciones:

- Solicitar la misma página más de unas pocas veces por segundo
- Realizar más de 50 solicitudes simultáneas sobre el mismo equipo por segundo
- Hacer cualquier solicitud mientras está temporalmente en la lista negra (en una lista de bloqueo)

3.2. Comparativa de soluciones existentes en el mercado

Servicio	NAXSI	Aqtronix Webknight	Shadow Daemon
Versión	1.3	4.0	2.0.2
Funcionalidad	XSS y SQL Injection.	Modo bloqueo y modo detección	Interfaz propia
SSOO Soportados	UNIX	Windows	UNIX
Licencia	OpenSource	OpenSource	OpenSource
Lenguaje	C	XML	C++
Nivel de uso	SW nginx	SW IIS	Menos que los anteriores
Dependencias software	libpcre	-	-

3.3. Clientes para el servicio

3.3.1. Referencias y características del cliente adoptado

Tan solo necesitamos un cliente sencillo que pueda cursar peticiones al servidor web, por tanto, utilizaremos un Terminal del "Equipo Cliente".

3.3.2. Comparativa de clientes existentes en el mercado

No procede

4. Proceso de instalación y Uso del cliente

Para el uso del cliente y el cursado de peticiones sencillas se ha facilitado el uso de un script sencillo *usar_servicio_cliente.sh* ubicado en *Ficheros/Scripts_Instalacion,* así como el fichero *test.pl* ubicado en *Ficheros/Otros*.

4.1. Obtención del software del cliente

No procede

4.2. Instalación del cliente

No procede

4.2.1. Primera instalación del cliente

No procede

4.2.2. Desinstalación del cliente

No procede

4.3. Configuración del cliente

Será necesario modificar la variable $IP_SERVIDOR del script *usar_servicio_cliente.sh*, así como el campo correspondiente del fichero *test.pl.*

5. Proceso de instalación/administración del servidor

En este apartado desglosaremos el proceso de instalación seguido para el Servidor Web Apache, y sus módulos ModSecurity y ModEvasive (consultar versiones y dependencias en la tabla anteriormente expuesta).

Se proporciona el script *instalar_servicio.sh* ubicado en *Ficheros/Scripts_Instalacion* para facilitar la instalación de los servicios en el servidor.

5.1. Obtención del software del servidor

Todos los servicios por instalar son de licencia OpenSource. Los obtendremos de repositorios git o mediante la instalación por paquetes disponible en Centos 7.

- Servidor Web Apache: Su instalación será llevada a cabo mediante el instalador de paquetes "yum". URL: http://httpd.apache.org/
- Mod_Security: Su instalación se realizará por compilación del código fuente. URL: https://github.com/SpiderLabs/ModSecurity/archive/v2.9.2.tar.gz
    - OWASP-CRS: La descarga de dichas reglas se hará desde la URL: https://github.com/coreruleset/coreruleset/archive/v3.4/dev.zip
- Mod_Evasive: Su instalación será llevada a cabo mediante el instalador de paquetes "yum". URL: http://www.zdziarski.com/blog/?page_id=442

5.2. Instalación del servidor

Procedemos a explicar los pasos seguidos para la correcta instalación de los servicios en el equipo servidor. Recordemos que se instalará mediante instalador de paquetes el servidor web apache y su módulo mod_evasive, el módulo mod_security será instalado desde código fuente.

5.2.1. Primera instalación del servicio

5.2.1.1. Instalación desde código fuente (Compilación)

Como ya hemos mencionado, procederemos a la instalación del módulo mod_security, pero cabe destacar que la instalación de dicho módulo no puede llevarse a cabo si no tenemos instalado el servidor web apache, además de las dependencias expuestas en la tabla del apartado 3.1.1. Dichas dependencias las instalaremos con los comandos:

```
yum install gcc

yum install make

yum install libcurl

yum install libxml2

yum install pcre-devel

yum install libxml2-devel

yum install curl-devel

yum install git

yum install flex bison yajl yajl-devel curl GeoIP-devel doxygen zlib-devel

yum install epel-release yum-utils

yum install libtool
```

Ahora sí, procedemos a la instalación de mod_security:

Ingresamos en el directorio raíz de Apache con el comando `cd /etc/httpd` y descargamos el código fuente del módulo comprimido con `wget` https://github.com/SpiderLabs/ModSecurity/archive/v2.9.2.tar.gz . Descomprimimos el archivo descargado `tar xvzf v2.9.2.tar.gz` e ingresamos en el directorio ModSecurity-2.9.2 `cd ModSecurity-2.9.2/.`

Procedemos a la compilación del código fuente con los siguientes comandos:

```
./autogen.sh

./configure

gmake
```

```
gmake install
```

Llegados a este punto, copiaremos el fichero de configuración recomendado de mod_security en el directorio de ficheros de configuración de apache conf.d, además del fichero unicode.mapping, necesario al cargar apache con este módulo.

```
cp modsecurity.conf-recommended /etc/httpd/conf.d/mod_security.conf

cp /etc/httpd/ModSecurity-2.9.2/unicode.mapping /etc/httpd/conf.d
```

Para añadir las reglas OWASP-CRS a nuestro módulo, nos situamos en el directorio raíz de apache de nuevo y descargamos y descomprimimos el archivo que las contiene

```
cd ..

wget https://github.com/coreruleset/coreruleset/archive/v3.4/dev.zip

unzip dev.zip
```

Y copiamos el fichero de configuración de las crs en el directorio de mod_security, además de la carpeta con las reglas a incluir.

```
cp coreruleset-3.4-dev/crs-setup.conf.example /etc/httpd/ModSecurity-
2.9.2/crs-setup.conf

cp -r coreruleset-3.4-dev/rules /etc/httpd/ModSecurity-2.9.2
```

Añadimos en el fichero de configuración de apache conf/httpd.conf las siguientes líneas para incluir los ficheros de configuración del módulo añadido y de las reglas crs :

LoadModule security2_module modules/mod_security2.so

IncludeOptional ModSecurity-2.9.2/crs-setup.conf

IncludeOptional ModSecurity-2.9.2/rules/*.conf

Y, por último, reiniciamos apache con el comando:

```
service httpd restart
```

5.2.1.2. Instalación desde paquetes/repositorios

Instalaremos mediante paquetes tanto el servidor web apache como el módulo mod_evasive con los comandos:

```
yum install httpd
```

```
yum install mod_evasive
```

Cabe destacar que el orden en el que se ejecuten estos comandos es importante, debido a que si se ejecuta primero `yum install mod_evasive` también será instalado el paquete del servidor web apache como dependencia.

5.2.2. Actualización del servicio

5.2.2.1. Actualización desde código fuente

Para llevar a cabo la actualización del módulo mod_security bastaría con repetir los pasos anteriormente mencionados, habiendo descargado el archivo comprimido con la versión más actualizada del servicio tanto de cualquiera de las siguientes URLs:

https://github.com/SpiderLabs/ModSecurity/

https://modsecurity.org/

5.2.2.2. Actualización desde paquetes/repositorios

La actualización mediante paquetes es bastante más sencilla, bastaría con ejecutar los comandos siguientes para actualizar el servidor web apache y su módulo mod_evasive, respectivamente.

```
yum -y update httpd
```

```
yum -y update mod_evasive
```

5.2.3. Desinstalación del servicio

5.2.3.1. Desinstalación desde código fuente

Procedemos a desinstalar mod_security desde código fuente con los siguientes comandos:

```
cd /etc/httpd/ModSecurity-2.9.2
```

```
gmake uninstall
```

Tenemos que eliminar manualmente los archivos de configuración y el módulo cargado en la carpeta modules, así como la carpeta Modsecurity-2.9.2.

```
cd ..
```

```
rm -rf ModSecurity-2.9.2
```

```
cd modules

rm -rf mod_security2.so

cd ..

cd conf.d

rm -rf mod_security.conf
```

Es necesario borrar las siguientes líneas anteriormente incluidas en el fichero de configuración de apache /etc/httpd/httpd.conf para un correcto reinicio del servidor web apache.

LoadModule security2_module modules/mod_security2.so

IncludeOptional ModSecurity-2.9.2/crs-setup.conf

IncludeOptional ModSecurity-2.9.2/rules/*.conf

5.2.3.2. Desinstalación desde paquetes/repositorios

Tanto el servidor web apache como el módulo mod_evasive los desinstalaremos con los comandos:

```
yum remove mod_evasive

yum remove httpd
```

Cabe destacar que el orden en el que se ejecuten estos comandos es importante, debido a que si se ejecuta primero `yum remove httpd` también serán eliminados todos sus módulos, y, por tanto, mod_evasive.

5.3. Configuración del servidor

Fichero de configuración del Servidor Web Apache: /etc/httpd/conf/httpd.conf: Es necesario incluir manualmente las líneas explicadas anteriormente para una correcta integración de mod_security.

Fichero de configuración de mod_security: /etc/httpd/conf.d/mod_security.conf: Para incluir las reglas OWASP-CRS y hacer que el módulo actúe en modo filtrado o bloqueo de petición. También se pueden configurar otras opciones como dónde se arrojan los logs.

Fichero de configuración de mod_evasive: /etc/httpd/conf.d/mod_evasive.conf: Distintos parámetros configurables para controlar los ataques DoS y DDoS.

6. Puesta en funcionamiento del servicio

Se proporciona el script *usar_servicio_servidor.sh* ubicado en *Ficheros/Scripts_Instalacion* para facilitar la gestión de los servicios en el servidor.

6.1. Arranque del servicio con el sistema

Arrancaremos el servidor web apache con el comando `service httpd start`.

6.2. Administración y monitorización del funcionamiento

6.2.1. Configuración y Uso de los ficheros de registro

Los ficheros de logs están ubicados en /etc/httpd/logs donde encontraremos:

Error_log: Lugar donde Apache httpd enviará información de diagnóstico y registrará los errores que encuentre en el procesamiento de solicitudes. Es el primer lugar que mirar cuando se produce un problema con el inicio del servidor o con el funcionamiento del servidor, ya que a menudo contendrá detalles de lo que salió mal y cómo solucionarlo.

Access_log: El registro de acceso al servidor registra todas las solicitudes procesadas por el servidor.

6.2.2. Arranque del servicio en modo detallado (verbose)

No aplica

6.2.3. Seguridad del servicio

No aplica

6.3. Tests y Pruebas del correcto arranque del servicio

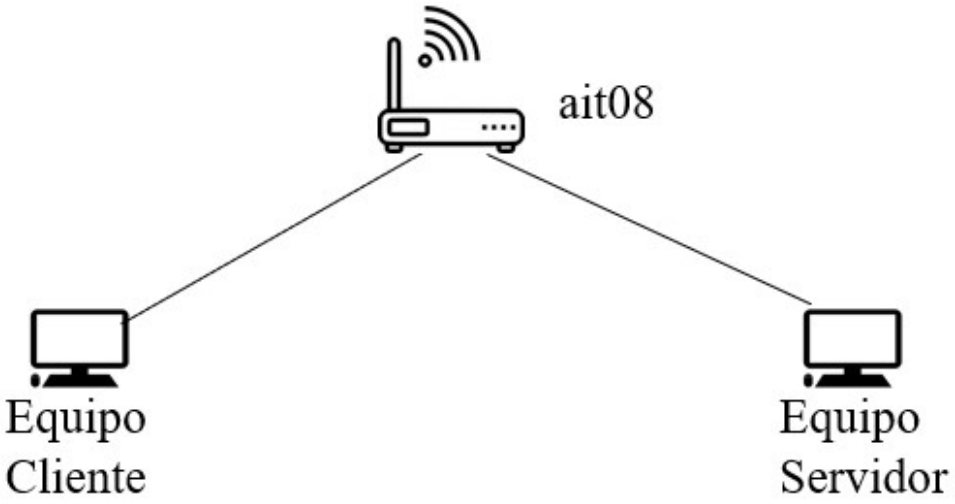
Para comprobar la carga adecuada de ambos módulos por apache, ejecutaremos el comando httpd -M y comprobamos la presencia de los módulos resaltados:

7. Diseño de los escenarios de prueba

7.1. Escenario de Defensa 1: <Escenario con ModSecurity>

En este primer escenario utilizaremos el servidor web Apache en el “Equipo Servidor” equipado con el módulo mod_security. Desde el “Equipo Cliente” realizaremos una serie de ataques básicos para comprobar la eficacia del módulo, así como sus principales funcionalidades.

7.1.1. Escenario 1: Esquema de la red



7.1.2. Escenario 1: Configuración del servidor

Para la preparación del servidor y su adecuada configuración inicial para el uso del módulo, se deben seguir en orden los pasos detallados en *5.2 Instalación del servicio.*

Para la activación del módulo en modo *bloqueo* o *filtrado,* modificaremos el parámetro *SecRuleEngine* del fichero /etc/httpd/conf.d/mod_security.conf a *On* o *DetectionOnly*, respectivamente.

Para cambiar entre niveles de paranoia (PL) 1 y 2, tendremos que cambiar el parámetro *setvar:tx.paranoia_level=x* de la regla *SecAction,* cambiando la "x" por un 1 o un 2 en función del nivel deseado.

```
SecAction \
  "id:900000,\
   phase:1,\
   nolog,\
   pass,\
   t:none,\
   setvar:tx.paranoia_level=x"
```

Para personalizar la regla *REQUEST-949-BLOCKING-EVALUATION,* cambiaremos la regla de la figura siguiente de PL1 a PL2 (simplemente cortando y pegando la regla en el PL indicado).

```
SecRule TX:ANOMALY_SCORE "@ge %{tx.inbound_anomaly_score_threshold}" \
    "id:949110,\
    phase:2,\
    deny,\
    t:none,\
    msg:'Inbound Anomaly Score Exceeded (Total Score: %{TX.ANOMALY_SCORE})',\
    tag:'application-multi',\
    tag:'language-multi',\
    tag:'platform-multi',\
    tag:'attack-generic',\
    ver:'OWASP_CRS/3.3.0',\
    severity:'CRITICAL',\
    setvar:'tx.inbound_anomaly_score=%{tx.anomaly_score}'"
```

Además, se añadirá en PL2 una regla muy simple hecha por mi, que será marcada si en la petición se contiene la palabra "test" en el argumento *testparam.*

```
SecRule ARGS:testparam "@contains test" "id:1234,deny,status:403,msg:'Regla de prueba hecha por mi'"
```

Nota: Todas estas operaciones anteriormente mencionadas se realizarán mediante el script *usar_servicio_servidor.sh,* donde cada vez que se ha de modificar un fichero, simplemente se borra el archivo a modificar y se sustituye por uno ya modificado ubicado en /Ficheros/Ficheros_Configuracion

7.1.3. Escenario 1: Tests y Pruebas del Escenario

Se realizarán 4 tipos de ataques sencillos:

a) Ataque Remote Code Exectucion: http://$IP_SERVIDOR/index.html?exec=/bin/bash

b) Ataque Cross Site Scripting (XSS): http://$IP_SERVIDOR/index.html'<script>alert</script>'

c) Ataque DoS: Mediante el comando `perl test.pl`. Se ejecuta el fichero *test.pl* proporcionado por el fabricante (previamente hay que modificar la dirección IP del servidor en dicho fichero).

d) Ataque personalizado: http://$IP_SERVIDOR/index.html?testparam=test.

Consideraremos las siguientes situaciones:

1) Ejecución de los 4 ataques con ModSecurity activo en modo filtrado (DetectionOnly) y PL1.

   Todas las peticiones generarán un Warning en los logs de ModSecurity, pero ninguna será bloqueada debido al modo en el que se encuentra el módulo y todas serán respondidas con un HTTP 200 OK, y por tanto correctamente atendidas.

2) Ejecución de los 4 ataques con ModSecurity activo en modo bloqueo (On) y PL1.

   Las peticiones realizadas por los ataques a) y b) serán bloqueadas con un HTTP 403 FORBIDDEN mientras que las realizadas tanto por el ataque c) como por el d) serán satisfactoriamente procesadas con un HTTP 200 OK.

3) Ejecución de los ataques a) y d) con ModSecurity activo en modo bloqueo (On), PL1 y la regla REQUEST-949-BLOCKING-EVALUATION personalizada.

   Las peticiones realizadas por los ataques a) y d) serán respondidas correctamente con un HTTP 200 OK. En el caso del ataque a) se debe a que hemos cambiado la regla que bloqueaba dicha petición en 2) de PL1 a PL2, continuando ModSecurity en PL1.

4) Ejecución de los ataques a) y d) con ModSecurity activo en modo bloqueo (On), PL2 y la regla REQUEST-949-BLOCKING-EVALUATION personalizada.

   Caso análogo al anterior, pero con ambas peticiones bloqueadas con un HTTP 403 FORBIDDEN debido al cambio de PL1 a PL2 en ModSecurity.

7.1.4. Escenario 1: Análisis del intercambio real de mensajes de red

Analizaremos los mensajes recibidos en Cliente y Servidor tras cada ataque (a-d) para cada situación (1-4). Para cada uno de los ataques se han considerado los mensajes más relevantes de las herramientas adecuadas.

1)

Equipo Cliente

a)

```
Realizando ataque Remote Code Execution
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="inicio.css" />
    <title>Web de prueba Trabajo STA</title>
  </head>
  <body>
  <div id="cabecera">
    <img src="objetos/logo.gif" alt="Logo de la página" width="15%"
    <h1>Idex.html</h1>
  </div>
  <div id="navegacion">
    <div class="menu"><a href="index.html">Inicio</a></div>
    <div class="menu"><a href="listas.html">Listas</a></div>
    <div class="menu"><a href="tablas.html">Tablas</a></div>
    <div class="menu"><a href="formulario.html">Formulario</a></div>
                <div class="vacio"></div>
 </div>
    <div class="vacio"></div>
  </div>
  <div id="pie">
    <div class="col_pie">
      <h4>Sobre este sitio</h4>
      <p>Propiedad de : Felipe Bueno</p>
    </div>
    <div class="vacio"></div>
  </div>
  </body>
</html>
```

b) Análogo a)

c)

No.	Time	Source	Destination	Protoc	Lengt	Info
744	11.07696110	192.168.1.65	192.168.1.66	HTTP	107	GET /?58 HTTP/1.0
746	11.07880174	192.168.1.66	192.168.1.65	HTTP	1134	HTTP/1.1 200 OK (text/html)
755	11.07980924	192.168.1.65	192.168.1.66	HTTP	107	GET /?59 HTTP/1.0
757	11.08170016	192.168.1.66	192.168.1.65	HTTP	1134	HTTP/1.1 200 OK (text/html)
765	11.08580123	192.168.1.65	192.168.1.66	HTTP	107	GET /?60 HTTP/1.0
767	11.08911282	192.168.1.66	192.168.1.65	HTTP	1134	HTTP/1.1 200 OK (text/html)
775	11.08999749	192.168.1.65	192.168.1.66	HTTP	107	GET /?61 HTTP/1.0
777	11.09187367	192.168.1.66	192.168.1.65	HTTP	1134	HTTP/1.1 200 OK (text/html)
785	11.09293155	192.168.1.65	192.168.1.66	HTTP	107	GET /?62 HTTP/1.0
787	11.09488154	192.168.1.66	192.168.1.65	HTTP	1134	HTTP/1.1 200 OK (text/html)
795	11.09602546	192.168.1.65	192.168.1.66	HTTP	107	GET /?63 HTTP/1.0
797	11.09811981	192.168.1.66	192.168.1.65	HTTP	1134	HTTP/1.1 200 OK (text/html)
805	11.09904358	192.168.1.65	192.168.1.66	HTTP	107	GET /?64 HTTP/1.0
807	11.10091809	192.168.1.66	192.168.1.65	HTTP	1134	HTTP/1.1 200 OK (text/html)
815	11.10195854	192.168.1.65	192.168.1.66	HTTP	107	GET /?65 HTTP/1.0

d)

```
Realizando ataque de Inyección personalizado
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="inicio.css" />
    <title>Web de prueba Trabajo STA</title>
  </head>
  <body>
  <div id="cabecera">
    <img src="objetos/logo.gif" alt="Logo de la página" width="15%" />
    <h1>Idex.html</h1>
  </div>
  <div id="navegacion">
    <div class="menu"><a href="index.html">Inicio</a></div>
    <div class="menu"><a href="listas.html">Listas</a></div>
    <div class="menu"><a href="tablas.html">Tablas</a></div>
    <div class="menu"><a href="formulario.html">Formulario</a></div>
                <div class="vacio"></div>
 </div>
    <div class="vacio"></div>
  </div>
  <div id="pie">
    <div class="col_pie">
      <h4>Sobre este sitio</h4>
      <p>Propiedad de : Felipe Bueno</p>
    </div>
    <div class="vacio"></div>
  </div>
  </body>
</html>
```

Equipo Servidor

En todos los casos ModSecurity enviará varios Warning como éste:

```
[Wed Jan 13 18:22:08.931180 2021] [:error] [pid 54516] [client 192.168.1.65:3744
5] [client 192.168.1.65] ModSecurity: Warning. Operator GE matched 5 at TX:anoma
ly_score. [file "/etc/httpd/ModSecurity-2.9.2/rules/REQUEST-949-BLOCKING-EVALUAT
ION.conf"] [line "150"] [id "949110"] [msg "Inbound Anomaly Score Exceeded (Tota
l Score: 8)"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.0"] [tag "application-mu
lti"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [host
name "192.168.1.66"] [uri "/index.html"] [unique_id "X-8sQOybWO1b5RRQzog9xQAAAAA
"]
```

2)

Equipo Cliente

a)

```
Realizando ataque Remote Code Execution
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /index.html
on this server.</p>
</body></html>
```

b) Análogo a)

c) Análogo 1.c)

d) Análogo 1.d)

Equipo Servidor

a)

```
[Wed Jan 13 18:24:25.176576 2021] [:error] [pid 54556] [client 192.168.1.65:3744
6] [client 192.168.1.65] ModSecurity: Access denied with code 403 (phase 2). Ope
rator GE matched 5 at TX:anomaly_score. [file "/etc/httpd/ModSecurity-2.9.2/rule
s/REQUEST-949-BLOCKING-EVALUATION.conf"] [line "150"] [id "949110"] [msg "Inboun
d Anomaly Score Exceeded (Total Score: 8)"] [severity "CRITICAL"] [ver "OWASP_CR
S/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"
] [tag "attack-generic"] [hostname "192.168.1.66"] [uri "/index.html"] [unique_i
d "X-8syZr-YqJcosxOTh2lkQAAAAA"]
```

b) Análogo a)

c) Análogo 1.c)

d) Análogo a Warning de Equipo Servidor en 1)

3)

Equipo Cliente

a) Análogo 1.a)

d) Análogo 1.d)

Equipo Servidor

a) Análogo a Warning de Equipo Servidor en 1)

d) Análogo a Warning de Equipo Servidor en 1)

4)

Equipo Cliente

a) Análogo 2.a)

d)

```
Realizando ataque de Inyección personalizado
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /index.html
on this server.</p>
</body></html>
```

Equipo Servidor

a) Análogo 2.a)

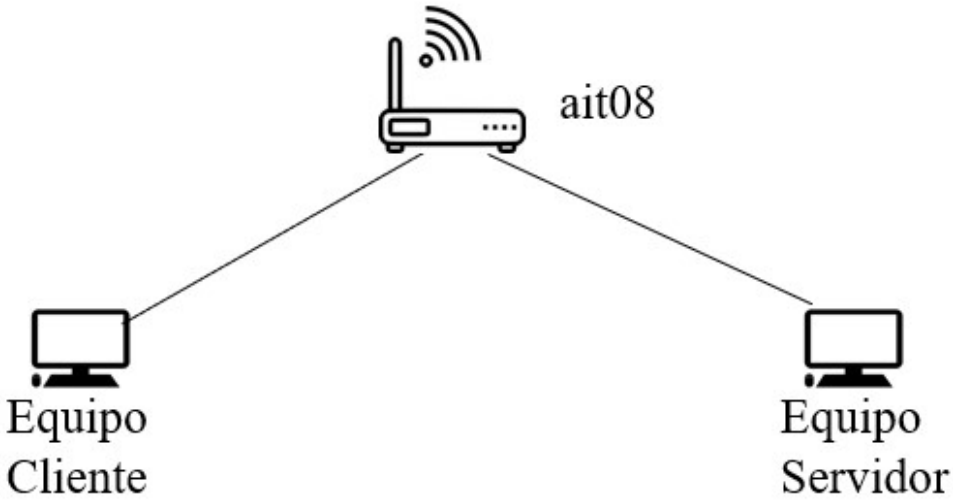
d)

```
[Wed Jan 13 18:34:13.172687 2021] [:error] [pid 54644] [client 192.168.1.65:37451] [c
lient 192.168.1.65] ModSecurity: Access denied with code 403 (phase 2). String match
"test" at ARGS:testparam. [file "/etc/httpd/ModSecurity-2.9.2/rules/REQUEST-949-BLOCK
ING-EVALUATION.conf"] [line "170"] [id "1234"] [msg "Regla de prueba hecha por mi"] [
hostname "192.168.1.66"] [uri "/index.html"] [unique_id "X-8vFdqrOvhZQtoAKDZK5AAAAAE"
]
```

7.2. Escenario de Defensa 2: <Escenario con ModEvasive>

En esta ocasión el módulo instalado (en el "Equipo Servidor") será mod_evasive, habiendo desinstalado previamente el módulo mod_security (o desactivado). Desde el "Equipo Cliente" se procederá a la realización de un ataque DoS mediante el fichero *test.pl.*

7.2.1. Escenario 2: Esquema de la red



7.2.2. Escenario 2: Configuración del servidor

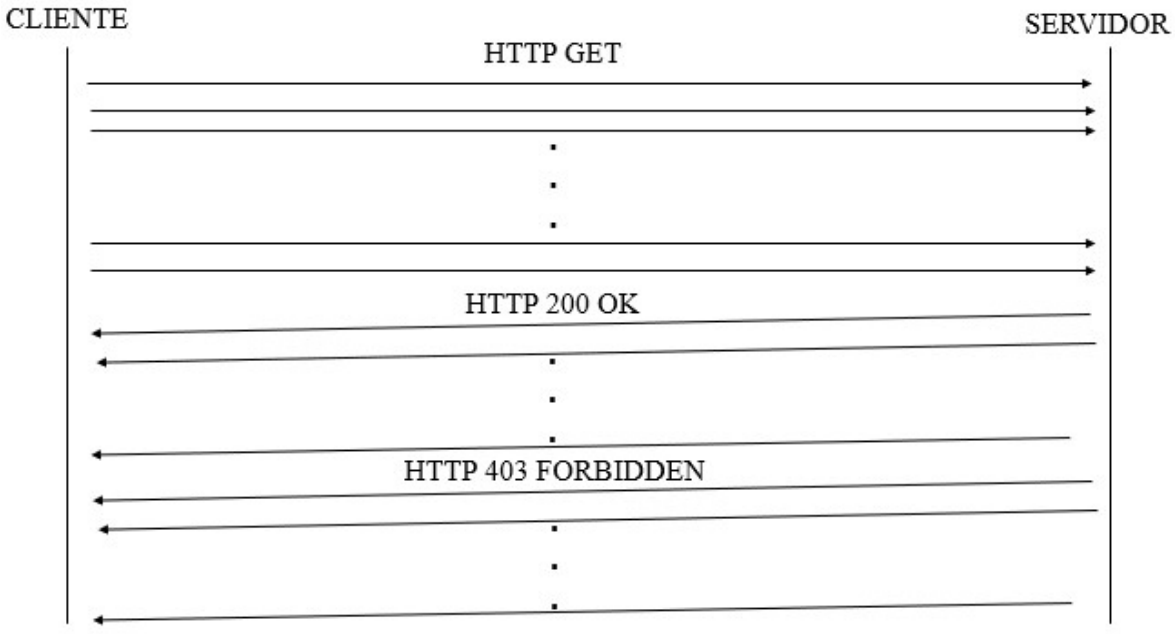
Instalaremos el módulo mod_evasive en el equipo servidor con el comando `yum install mod_evasive`.

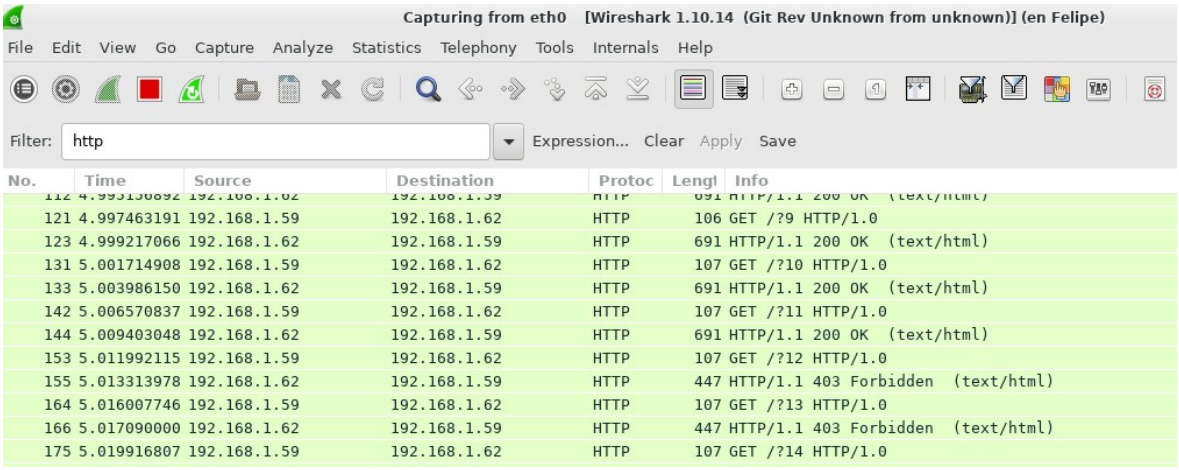
7.2.3. Escenario 2: Tests y Pruebas del Escenario

Ejecutando en el cliente el fichero test.pl con el comando `perl test.pl` (opción 3 en usar_servicio_cliente.sh) realizaremos un ataque DoS a nuestro servidor web con dirección IP indicada en dicho fichero (modificar antes de realizar el ataque).

Tras el ataque, comprobaremos como las primeras peticiones serán respondidas de manera correcta con el mensaje HTTP 200 OK, llegando a una petición concreta en la que el servidor nos bloqueará el acceso devolviéndonos un HTTP 403 FORBIDDEN en todas las peticiones sucesivas a causa del bloqueo efectuado por mod_evasive.

7.2.4. Escenario 2: Análisis del intercambio real de mensajes de red





8. Interfaz gráfica de administración del servidor

No procede, ninguno de los módulos de Apache tratados en este trabajo posee una interfaz gráfica para la configuración de este.

9. Deficiencias del servicio

La carencia principal detectada en la implementación del servicio es la falta de una interfaz gráfica al menos para la gestión de ModSecurity, a ello se debe la implementación del script *usar_servicio_servidor.sh.* Tareas como modificación de reglas, cambios de nivel de paranoia y modo de actuación del módulo, etc, hacen que la configuración de este sea bastante desagradable.

En cuanto a ModEvasive, me parece un módulo quizás demasiado sencillo (pocos parámetros configurables) para el ataque tan importante del que protege (recordemos que los ataques DoS y DDoS llevan años en el OWASP Top 10).

Otro defecto importante en mi opinión es la dificultosa visualización de los logs sobre todo de ModSecurity. En numerosas ocasiones son de vital importancia estos logs y se hace muy complicado analizarlos con detalle.

10. Ampliaciones/mejoras del servicio

Al hilo de lo comentado en el apartado anterior, la ampliación principal que necesita sobre todo el módulo ModSecurity es la integración de una interfaz gráfica para su administración.

Incluiría también algún mecanismo intuitivo y sencillo de utilizar para probar la eficacia de las reglas, debido a que en mi caso me ha resultado complicado explorar los límites del módulo ModSecurity, ya que carezco de un servidor web con la envergadura necesaria para realizar las pruebas, así como de conocimientos sobre ataques más severos que pongan en jaque a estas.

11. Incidencias y principales problemas detectados

El principal problema encontrado ha sido buscar maneras de estudiar las funcionalidades de ModSecurity de manera práctica en vista de los motivos explicados en el apartado previo.

También me resultó complicada la instalación por código fuente del módulo ModSecurity, ya que en función de la versión descargada del repositorio web se integraba de manera

correcta o no con la versión de Apache disponible mediante la instalación por paquetes de Centos 7.

12. Resumen y Conclusiones

A pesar de los inconvenientes comentados en apartados anteriores, tanto ModSecurity como ModEvasive son soluciones inteligentes y eficaces de cara a proteger nuestro servidor web.

El principal punto fuerte de ambas es que son de licencia OpenSource, lo que les hace situarse un paso por delante de la competencia WAF del mercado sin tener nada que envidiar en cuanto a funcionalidad ofrecida.

Son módulos altamente complementarios, ya que cada uno cubre las carencias del otro. ModSecurity es mucho más completo y abarca un rango mucho más amplio de ataques, y ModEvasive se especializa en ataques DoS y DDoS, permitiendo configurar de manera exhaustiva nuestro cortafuegos para este tipo de amenazas en concreto.

ANEXO A: Parámetros de configuración y comandos de gestión del servidor

Fichero de configuración de ModSecurity

Ubicación: /etc/httpd/conf.d/mod_security.conf

- Rule engine initialization: mediante la directiva SecRuleEngine On | Off DetectionOnly, se determina la inicialización del motor de reglas de ModSecurity; en una primera fase de pruebas es conveniente configurar esta directiva en modo DetectionOnly con el objeto de ver que las reglas están funcionando correctamente (analizando los registros de log), sin que se llegue a bloquear ninguna petición.

- Request body handling: mediante la directiva SecRequestBodyAccess On | Off, determina si ModSecurity tiene acceso al cuerpo de las peticiones (fase 2); es importante que esté activada, ya que de lo contrario no podrán analizarse los parámetros POST.

- Response body handling: en esta sección nos encontramos con cuatro directivas:

    - SecResponseBodyAccess On | Off: lo activaremos en el caso de tener reglas que deban acceder al cuerpo de la respuesta (fase 4). El objetivo de las reglas en la fase 4 es evitar que el servidor pueda responder con algún tipo de contenido inadecuado (fuga de información).

    - SecResponseBodyMimeType: determina el tipo de contenido del cuerpo de la respuesta que debe inspeccionarse: es posible definir la inspección de archivos de texto, html, etc, evitando la inspección de, por ejemplo, imágenes.

    - SecResponseBodyLimit: determina el tamaño máximo hasta el cual se va a inspeccionar el cuerpo de la respuesta.

    - SecResponseBodyLimitAction: determina la acción que debe llevarse a cabo si se supera el tamaño máximo a procesar del cuerpo de la respuesta.

- Debug log configuration: hay dos directivas en esta sección:

    - SecDebugLog: especifica la ruta y el archivo en el que se van a guardar los registros del log.

  - SecDebugLogLevel: especifica el nivel de detalle con que se va a llevar a cabo el log (desde 0 hasta 9).

- AuditLogConfiguration: las siguientes directivas se aplican en esta sección:

  - SecAuditEngine RelevantOnly | On | Off: por defecto únicamente se van a registrar las transacciones que estén marcadas por una regla, o aquellas que produzcan un código de error relevante, determinado por la directiva SecAuditLogRelevantStatus.

  - SecAuditLogRelevantStatus: determina el tipo de error producido por el servidor web que va a ser considerado relevante y, por lo tanto, registrado en el log. Por defecto, suelen registrarse los errores de tipo 5xx y 4xx (excluyendo el error de tipo 404 – prohibido).

  - SecAuditLogParts: especifica qué partes de la transacción serán registradas. La descripción de cada una de las partes de la transacción se detalla en la tabla 5.

  - SecAuditLogType Serial | Concurrent: indica qué tipo de registro de log será llevado a cabo. En modo serial, todas las transacciones se registran en un único archivo. En modo concurrent, registra cada transacción en un fichero.

  - SecAuditLog: especifica el archivo en el que se van a registrar las transacciones del log, si éste está configurado en modo serial.

  - SecAuditLogStorageDir: especifica la ubicación en la que se van a guardar los archivos log generados, si está en modo concurrent.

Fichero de configuración de ModEvasive

Ubicación: /etc/httpd/conf.d/mod_evasive.conf

- DOSHashTableSize = Tamaño de la tabla hash utilizada para el seguimiento de la actividad en cada dirección IP. Al aumentar este número, será más rápido detectar los sitios que el cliente ha visitado en el pasado, pero puede afectar (consumo alto de memoria) el rendimiento global del servicio. Es recomendable aumentar este parámetro en los servidores web de carga pesada.

- DOSPageCount = Número máximo de peticiones posible de una misma IP a la misma página, en el tiempo (segundos) definido en "DOSPageInterval". Una vez que se alcanza este número de peticiones, la IP será bloqueada.

- DOSSiteCount = Número máximo de peticiones posible de una misma IP a cualquier página del site, en el tiempo (segundos) definido en "DOSSiteInterval". Una vez que se alcanza este número de peticiones, la IP será bloqueada.

- DOSPageInterval = Cantidad de tiempo (segundos) para el parámetro umbral DOSPageCount.

- DOSSiteInterval = Cantidad de tiempo (segundos) para el parámetro umbral DOSSiteCount.

- DOSBlockingPeriod = Cantidad de tiempo (en segundos) que el atacante (IP) será bloqueada. Durante este tiempo, todas las solicitudes posteriores del atacante (IP) se traducirá en un error 403 (Forbidden).

- DOSWhitelist = IP o rango de IP que no será excluido de las reglas de mod_evasive.

- DOSEmailNotify = Dirección de correo electrónico que recibirá información sobre el ataque.

- DOSSystemCommand = Comando o instrucción en bash que será ejecutada cuando el atacante (IP) sea bloqueada

- DOSLogDir = Ruta (path) para el almacenamiento de log's

ANEXO B: Parámetros de configuración y comandos de gestión del cliente

No procede, utilizado como cliente tan solo un terminal.