Implementación de un SIEM para la auditoría de eventos de seguridad sobre cluster de Kubernetes en un entorno multicloud

José Ángel Fernández Ameijeiras
Máster de Seguridad de las tecnologías de la información y las comunicaciones
Seguridad Empresarial

Miguel Ángel Flores Terrón
Cristina Romero Tris

04/01/2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Implementación de un SIEM para la auditoría de eventos de seguridad sobre cluster de Kubernetes en un entorno multicloud*
Nombre del autor:	*José Ángel Fernández Ameijeiras*
Nombre del consultor/a:	*Miguel Ángel Flores Terrón*
Nombre del PRA:	*Cristina Romero Tris*
Fecha de entrega (mm/aaaa):	01/2022
Titulación:	*Máster de Seguridad de las tecnologías de la información y las comunicaciones*
Área del Trabajo Final:	*Seguridad Empresarial*
Idioma del trabajo:	*Castellano*
Palabras clave	*SIEM* *WAZUH* *KUBERNETES*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

En la actualidad cualquier organización que trabaje con sistemas y tecnologías de información está expuesta a múltiples amenazas de seguridad. El cada vez mayor número y heterogeneidad de estos sistemas hace indispensable que las organizaciones tengan capacidad de registrar, auditar y correlar todos los eventos de seguridad en un sistema centralizado con el objetivo de mantener bajo control la postura de seguridad y su nivel de riesgo tecnológico.

A través de este trabajo y empleando el caso ficticio de una organización para contextualizar el supuesto tecnológico de partida, se realiza un estudio de los SIEM de mercado, con una clara orientación hacia el uso de un modelo de software libre así como un diseño empleando los paradigmas del modelo cloud. Para ello se diseña e implementa un modelo de arquitectura desplegando una solución SIEM sobre un proveedor de cloud de mercado con capacidad de resiliencia y escalado.

Una vez alcanzado este objetivo, se diseña un juego de pruebas de concepto para validar la solución y se pone a prueba sobre un ecosistema distribuido de agentes remotos, corroborando la generación de eventos de auditoría de seguridad tanto desde sistemas On-premise como desde origenes en múltiples proveedores cloud.

Abstract (in English, 250 words or less):

Nowadays any organization that works with information systems and technologies is exposed to multiple security threats. The increasing number and heterogeneity of these environments makes it essential for these organizations to have the ability to audit and correlate all security events in a centralized system in order to keep the security posture and the level of risk under control.

Through this work and using a fictitious case of an organization to contextualize the starting technological assumption, a study of market SIEMs is carried out withclear orientation towards with opensource software and implementation in a multicloud model with scaling and resilience capabilities.

Then an architecture model is been designed and next it will the base for deployment a SIEM solution on a market cloud provider.

Once final system is designed and deployed, a set of proofs of concept is performed to validate the solution on distributed and remote agents, testing the generation of security events as from sources on-premises and multiple cloud providers.

Índice

Lista de figuras

Índice de tablas

1 Introducción

1.1 Contexto y justificación del Trabajo

En la actualidad estamos asistiendo a la entrada exponencial de incidentes de seguridad en multitud de ámbitos de nuestra vida cotidiana. Gobiernos, empresas y usuarios particulares nos vemos expuestos a una larga lista de riesgos de seguridad, con la continua exposición de nuestros datos en multitud de sistemas de información y el uso de decenas de dispositivos tanto en nuestro ámbito personal como profesional.
La situación de pandemia desencadenada en el año 2020 por la aparición del COVID-19 no ha hecho más que agravar estos riesgos, empujando a la deslocalización de los trabajadores de las empresas, que antaño basaban su seguridad en perímetros acotados, así como al fomento de la virtualización de las relaciones humanas y a la explosión definitiva del uso de la tecnología en cualquier franja de edad y condición social.

Si analizamos los datos que arrojan los grandes analistas del sector TI{url01}, el gasto en seguridad estimado para el cierre del año se prevé en más de 150 billones de dólares frente a los 120 del año anterior. Se estima además que el gasto pueda incrementarse hasta los 400 billones para el año 2027{url02}. Toda esta inversión no ha servido para reducir el número de incidentes pues se estima que, respecto a los años previos, el malware se ha incrementado en más de un 400%, el Ransomware en 435% así como un incremento de un 653% en la actividad cibercriminal, y por consiguiente el coste asociado también se ha disparado.{url03}

A nivel tecnológico, los últimos años también nos han traído nuevos paradigmas de computación y desarrollo. Así, nuevas tecnologías como la visualización en contenedores, el movimiento de las cargas de trabajo a entornos cloud de múltiples proveedores, las metodologías Devops y la orquestación de despliegues de infraestructura como código (IaaC) han propiciado un incremento de los riesgos de seguridad a los que se exponen las organizaciones, provocando un cambio continuo en la postura de seguridad de las mismas.

Por todo ello, cualquier empresa del sector tecnológico, bien sea para sus soluciones corporativas como para servicios a terceros, que pretenda ofertar un servicio seguro, se ve abocada en mayor o menor medida, a la implementación de un sistema que le permita correlar eventos de seguridad de múltiples orígenes de datos.

En este proyecto se plantea la implementación de SIEM corporativo empleando una solución de código abierto, que se diseñará sobre un ecosistema preexistente de tecnologías y con una arquitectura con soporte multicloud.

Para ello, en primer lugar se contextualiza un entorno inicial de trabajo partiendo de una empresa ficticia que sufre un problema de seguridad y que nos servirá para a definir una situación y ciertas premisas sobre un modelo de partida.

A continuación se realiza un estudio del mercado de soluciones SIEM de código abiertos, seleccionándose la herramienta que dará soporte a la solución que se implementará.
En las siguientes etapas se procede a realizar un análisis de los principales actores del mercado cloud y también se seleccionará el más adecuado para el despliegue de la solución. En base a esto, se diseña la arquitectura que dará soporte de ejecución del SIEM empleando tecnología de contenedores Kubernetes y formulando un modelo que incluya capacidad de soportar correlación de eventos tanto de un modelo de agentes On-premise como multicloud.

Finalmente se diseñarán unos casos de uso que nos servirán para verificar, a modo de prueba de concepto, las tecnologías y arquitecturas implementadas con este proyecto.

Esta solución permitirá así dotar a la empresa que nos ha servido como modelo de contextuaizacion de una solución para la gestión de eventos de seguridad escalable con un modelo de diseño y operación compatible con los actuales modelos y paradigmas de DevSecOps y cloud.

1.2 Contextualización, la Empresa *0MyG0d S.L.*

La empresa *0MyG0d S.L.* (*0MyG0d* en adelante) es una empresa del sector TI que con tan solo 10 años de existencia ya cuenta con un importante porfolio de clientes y una enorme proyección. Nacida como una spin-off en un vivero de empresas de la universidad y de la mano de 2 socios fundadores, actualmente cuenta ya con más de 100 empleados que trabajan tanto presencialmente desde las 2 oficinas que tienen en Madrid y Barcelona como en modalidad de teletrabajo.

El éxito de la empresa viene de la mano de un novedoso sistema de prevención de fraude y blanqueo de capitales, bautizado como KyCus, que los socios idearon y que actualmente la empresa mantiene, desarrolla y comercializa a

muchos de sus clientes en modelo de software como servicio (SaaS[1]). Este es el core principal de su negocio, pero la empresa ya ha afianzado muchas alianzas estratégicas con grandes clientes, y derivado de ello complementan su actividad principal con consultorías y desarrollos a medida de múltiples soluciones y adaptaciones de su producto estrella.
El producto *KyCus* comenzó sus andaduras en un pequeño centro de datos montado por los socios durante su inicio en el sótano de la oficina de Madrid. Actualmente, el datacenter de la oficina de Madrid mantiene unos cuantos servidores que la empresa usa como backoffice de algunos servicios, como son, el directorio corporativo, el software de gestión y ciertos entornos de explotación que se emplean para dar soporte al producto KyCus. Este centro de datos lo utilizan también como nudo central de comunicaciones entre las oficinas y los trabajadores que están en modalidad de teletrabajo y se conectan a través de VPN.
Debido a que gran parte de sus clientes ya tienen sus centros de datos de explotación en la nube, la empresa se ha visto empujada a desplegar las cargas de trabajo de su producto a distintos proveedores cloud. De este modo, como sus clientes empezaron a demandar el que se les facilitase acceso a la solución desde un entorno con menor latencia, *0MyG0d* no pudo preparar una estrategia de migración y adopción de cloud y tuvo que realizar una rápida transición a este modelo. Así, actualmente ofrece los servicios de su producto tanto desde el datacenter de Madrid como desde los proveedores cloud AWS y Azure.
Gran parte de los empleados, un 75% en la actualidad, forman parte de los equipos de desarrollo. Es un equipo de gente joven con gran conocimiento técnico y mucha inquietud por el empleo de las últimas tecnologías y los modelos de desarrollo y paradigmas más modernos. Por ello, a nivel corporativo hace tiempo que se adoptó principalmente el modelo de desarrollo ágil y se fomentó que los equipos de desarrollo también se encargasen de la operación de las plataformas.
Ya antes del salto a llevar sus cargas de trabajo al cloud, los equipos técnicos habían llevado a cabo un proyecto para la adopción de la tecnología de contenedores para el despliegue de las solución de su producto estrella. Ésta además tenía como objetivo el facilitar la posterior migración del servicio a estos nuevos modelos. El éxito de mismo fue tal que se adoptó ésta misma tecnología para el resto de soluciones tecnológicas de la empresa. Al principio se llevaban a cabo los despliegues de manera manual en contenedores individuales pero pronto se hizo patente la necesidad de un orquestador para dar soporte a toda aquella gestión. Aquí ya se encontraban en pleno proceso de adopción de cloud. En esta etapa, se dieron distintos silos separados de grupos de trabajo dentro del equipo de desarrollo, que llevaron diferentes líneas de trabajo; todo ello provocado porque cada uno de los líderes técnicos estaba enfocado al trabajo con un proveedor concreto de cloud y tenía mayor conocimiento de su tecnología y servicios. Así, un grupo se enfocó más en la gestión del datacenter de Madrid, otro grupo se enfocó al diseño de la nueva arquitectura de la solución para AWS, mientras que otro se centró en Azure. Esto hizo que cada línea de trabajo adoptase soluciones y orquestadores distintos en función de la oferta de servicios del proveedor en cuestión.

Para el día a día del trabajo todos los empleados cuentan con un equipo portátil maquetado con una imagen corporativa, mayoritariamente con el sistema operativo Microsoft Windows 10, las herramientas de Ofimática, los IDE's de desarrollo, la conexión VPN con la oficina central, etc. En el último año, uno de los equipos de desarrollo decidió que para su operación diaria necesitaban emplear un sistema operativo Linux, por lo que la empresa también adopto la posibilidad de uso de este sistema base en sus equipos de escritorio y encargo a su departamento de sistemas preparar una imagen corporativa con el sistema operativo Ubuntu 20.04.

Con este modelo de trabajo, equipos de escritorio y datacenters la empresa siguió operando con normalidad hasta un día del mes de abril del 2020. A raíz del confinamiento por la pandemia del COVID-19 todos los empleados se encontraban trabajando en remoto desde sus domicilios. El modelo de teletrabajo previo y la implantación cloud en gran parte de sus soluciones ayudo a que la empresa continuase trabajando con total normalidad al inicio de la pandemia.
Una noche, el CAU de soporte de la aplicación *KyCus* comenzó a recibir llamadas de varios de sus clientes. La aplicación desplegada en algunos de ellos comenzó a responder con lentitud y la mayor parte de las funcionalidades no respondían en tiempo, como en los días previos. Tras la revisión por parte del equipo de guardia no se detectó cual podía ser el problema de rendimiento de estas instancias. Tras un reinicio preventivo de alguno de los sistemas de los entornos afectados todo pareció responder de nuevo con normalidad. Varias noches más tarde, el problema se repitió, pero en este caso, al volver a revisarlo los técnicos pudieron comprobar que todas las instancias de clientes que estaban alojados en el proveedor AWS sufrían lentitud y tiempos de respuesta completamente anómalos. Tras toda la noche y subsiguientes días de revisión sin detectar nada, el problema fue en aumento. Las altas cargas de trabajo continuadas en el tiempo, provocaron que las instancias de la nube AWS empezasen a aprovisionar nuevos recursos automáticamente con lo que aparecieron las primeras alertas que avisaban de un aumento importante en la facturación, sin que los problemas de latencia mejorasen.
Esa misma tarde, exactamente los mismos problemas que estaban sufriendo los clientes de la aplicación desplegada en el cloud de AWS saltó al segundo entorno de cloud. El mismo comportamiento visto en días anteriores: se dispararon los consumos de recursos en los equipos desplegados en el cloud de Azure.
Con toda la infraestructura saturada y el servicio de la empresa completamente colapsado, los técnicos finalmente comenzaron a ver los primeros indicios del problema.
Desconociendo el origen inicial del problema, los técnicos comenzaron a ver que los contenedores donde se despliegan

1 https://es.wikipedia.org/wiki/Software_como_servicio

todos los componentes de la aplicación *KyCus* parece que no tenian exactamente el mismo software y componentes que el equipo había desarrollado. Algunos de ellos parecian estar enviando un alto volumen de datos por servicios que a priori no deberían estar usadno, otros contenedores parecian estar consumiendo CPU realizando millones de extrañas operaciones matemáticas, etc. En ese momento, los equipos de soporte comienzar a redesplegar versiones anteriores de las imágenes de sus contenedores, pensando en algún error en el software o en algún cambio o compilación incorrecta.
Finalmente, a la mañana siguiente, parece que el problema empieza a mejorar gracias a la estrategia de restauración de servicios con los backups de las imágenes anteriores. Aún así, ciertos problemas continúan.
En las horas siguientes, uno de sus grandes clientes contacta con ellos para indicarle que su SOC le ha notificado que han saltado numerosas alertas de seguridad en su SIEM que parecen provenir de la línea VPN que emplean para conectarse a la solución nube de *KyCus*, y que por lo tanto como medida preventiva, suspenden la interconexión de sus sistemas con la empresa.
Este es el momento, en el cual los técnicos de la empresa descubren finalmente el problema; existe un compromiso de seguridad generalizado en toda su infraestructura. A partir de aquí, contratan los servicios de una importante consultora de seguridad para que les ayude a enderezar la situación.
Tras varías semanas de trabajo, consiguen detectar y restablecer paulatinamente los servicios de la organización. Esto no ha impedido ya el descrédito y los daños irreparables en la empresa por la pérdida de confianza de sus clientes.
Tras varias semanas de análisis forense, *0MyG0d* descubrió que la causa raíz del problema se originó meses atrás en uno de los equipos de escritorio de uno de los desarrolladores que fue comprometido. De ahí, los asaltantes fueron moviéndose lateralmente a otros equipos empleando distintas técnicas, hasta que finalmente consiguieron llegar a comprometer un equipo que albergaba las credenciales de acceso a los portales de ambos proveedores cloud, acceso a repositorios de imágenes, bases de datos, etc.

1.3 Motivación

La principal motivación de este proyecto es que el este pueda servir como guía para que a través del estudio realizado, cualquier empresa que se encuentre en una situación tecnológica similar a la empresa "*0myG0d*" y tenga la necesidad de implementar un sistema de correlación de eventos de seguridad, pueda encontrar en este proyecto un análisis exhaustivo de una solución de código abierto de mercado como es el SIEM Wazuh, así como una guía detallada de diseño e implementación sobre una arquitectura en alta disponibilidad y escalable basada en un modelo cloud y con soporte tanto para gestionar los eventos de seguridad provinientes de un sistema de agentes distribuido y de un ambiente de cargas de trabajo tanto On-premise como de múltiples proveedores cloud.

Otra motivación es que este proyecto sirva como impulso para tratar de concienciar y dar visibilidad de la necesidad que existe actualmente de mejorar y proteger todos los sistemas de información que manejan nuestros datos, dejar patente cómo la implementación de un sistema de gestión de eventos de seguridad permite situarlos en una mejor posición de seguridad, y como la capacidad de correlar eventos de múltiples fuentes permite además dotar a los sistemas de información de mayor resiliencia.

1.4 Objetivos del Trabajo

El objetivo estratégico del presente proyecto es el diseño e implementación del modelo de arquitectura de un SIEM para la auditoría de eventos de seguridad que se generan desde entornos híbridos, tanto On-premise como multicloud, y desplegado sobre un clúster de contenedores Kubernetes que se ejecuta siguiendo el modelo cloud.

Los principales objetivos que se pretende conseguir son:

- Mejorar la postura de seguridad y resiliencia general de la empresa.
- Obtener un sistema en modalidad de coste por Uso (Pay-as-you-go).
- Desarrollar la capacidad de detección y respuesta frente a amenazas de seguridad .
- Ser capaz de generar inteligencia de seguridad avanzada.
- Mejorar el manejo del riesgo de la organización.
- Aportar mayor valor añadido de los servicios de la organización.
- Ventaja competitiva respecto a otros competidores del mercado.

1.5 Enfoque y metodología seguido

La planificación y el desarrollo del presente proyecto se basa en un modelo clásico en cascada o modelo Waterfall. Para ello, a continuación se detalla una planificación en base a tareas que comenzarán por las fases de análisis y diseño, e irán evolucionando hacia modelos de implementación, pruebas y finalmente la generación de casos de uso de ejemplo.
Para su desarrollo el proyecto se basa en ciertas premisas, ya detalladas en el modelo de contextualización y el estado del arte de una empresa ficticia. En base a ello, se parte de una situación y modelo tecnológico preexistente y trataremos de ir realizando análisis, adaptaciones de los sistemas subyacentes y diseño de nuevas arquitecturas de la empresa para la consecución de los objetivos detallados en el punto anterior.

Se estima que esta estrategia es la más adecuada por la situación actual de la empresa así como en tiempo y los recursos que se emplearán para el desarrollo del proyecto.

1.6 Planificación del Trabajo

A continuación se presenta una propuesta de la planificación del proyecto. Destacar en este apartado que para los cálculos de esfuerzo se han considerado jornadas de 8 horas de trabajo diarias de lunes a viernes y no se han computado los días festivos.

1.6.1 Desglose de tareas

A continuación se detalla un que desglose y descripción general de las tareas del proyecto.

- **Tarea:** Desarrollo de un plan de trabajo.
  Se desarrollará y detallará la justificación del proyecto, contextualización, objetivos y un plan de desarrollo y trabajo del mismo.
- **Tarea:** Análisis y estudio general de la solución.
  En esta fase inicial del proyecto se realizará una recopilación general de información sobre la solución, fabricantes y arquitecturas en soluciones de productos similares.
- **Tarea:** Estudio de los sistemas SIEM de marcado.
  Se realizará un estudio y comparación de los SIEM del mercado. Se analizará, contextualizarán y estudiarán y se probarán distintas soluciones.
- **Tarea:** Análisis y documentación del SIEM Wazuh.
  Se hará un estudio detallado del SIEM Wazuh. Se analizarán y presentarán modelos de despliegue y la arquitectura de la solución.
- **Tarea:** Análisis y documentación del modelo cloud para abortar esta solución.
  Se analizará la situación, modelo y proveedores para abordarlos como propuesta de solución tecnológica.
- **Tarea:** Análisis y documentación de las soluciones de orquestación de los proveedores Cloud.
  Se analizará la opciones de servicio y soluciones del proveedor seleccionado para abordarlo como sistema de orquestación de la solución en contenedores.
- **Tarea:** Diseño de la arquitectura de la solución.
  Se diseñará la arquitectura final de la solución. Se definirá el modelo de comunicaciones con los agentes y sistemas de eventos a conectar.
- **Tarea:** Implementación de la arquitectura de la solución.
  Se implementará y probará la solución de arquitectura.
- **Tarea:** Generación de ejemplos y prueba de concepto del modelo anterior.
  Se diseñara un modelo de pruebas formado por varios casos de uso que se probaran en la solución
- **Tarea:** Redacción y documentación final del proyecto.
  Se realizará la fase final de documentación y redacción del proyecto. Se realizarán las conclusión y siguientes pasos del proyecto
- **Tarea:** Realización de una presentación y video de guía sobre la misma.
  Se realizará una presentación del proyecto que servirá como guía e introducción al video de presentación.

1.6.2 Diagrama de Gantt

A continuación se detalla un diagrama de Gantt con la planificación del proyecto. Adicionalmente, incluimos mayor detalle de la planificación en los anexos del proyecto.

ID	TIPO	ASUNTO	ESTADO	FECHA DE INICIO	FECHA DE FINALIZACIÓN
36	MILESTONE	Inicio	New	15/09/2021	15/09/2021
48	TASK	Desarrollo de un plan de trabajo	New	15/09/2021	28/09/2021
37	MILESTONE	Entrega 1	New	28/09/2021	28/09/2021
49	TASK	Análisis y estudio de la solución Global	New	29/09/2021	06/10/2021
50	TASK	Estudio de soluciones SIEM del mercado	New	07/10/2021	12/10/2021
51	TASK	Análisis y documentacion SIEM Wazuh	New	12/10/2021	19/10/2021
43	MILESTONE	Entrega 2	New	26/10/2021	26/10/2021
52	TASK	Análisis y documentación modelo Cloud	New	19/10/2021	27/10/2021
53	TASK	Orquestación de contenedores en cloud	New	27/10/2021	02/11/2021
54	TASK	Diseño de la arquitectura de la solución	New	02/11/2021	23/11/2021
55	TASK	Implementación de la solución	New	10/11/2021	08/12/2021
59	MILESTONE	Entrega 3	New	23/11/2021	23/11/2021
56	TASK	Generación de los ejemplos y casos de u...	New	08/12/2021	17/12/2021
57	TASK	Redacción y documentación final del pro...	New	20/12/2021	27/12/2021
60	MILESTONE	Entrega 4	New	28/12/2021	28/12/2021
58	TASK	Presentación y video	New	28/12/2021	04/01/2022
61	MILESTONE	Presentación	New	04/01/2022	04/01/2022
62	MILESTONE	Defensa	New	14/01/2022	14/01/2022

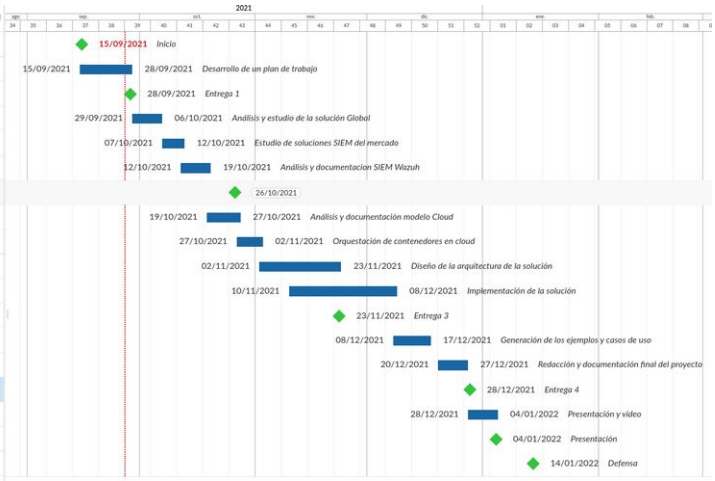


Figura 1: Diagrama Gantt de proyecto

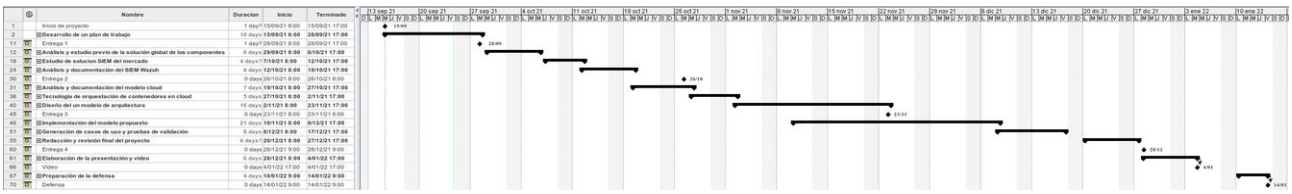


Figura 2: Diagrama Gantt con detalle de horas

		Nombre	Duracion	Inicio	Terminado
1		Inicio de proyecto	1 day?	15/09/21 8:00	15/09/21 17:00
2		**Desarrollo de un plan de trabajo**	10 days	**15/09/21 8:00**	**28/09/21 17:00**
11		Entrega 1	1 day?	28/09/21 8:00	28/09/21 17:00
12		**Análisis y estudio previo de la solución global de los componentes**	6 days	**29/09/21 8:00**	**6/10/21 17:00**
18		**Estudio de solucion SIEM del mercado**	4 days?	**7/10/21 8:00**	**12/10/21 17:00**
24		**Análisis y documentación del SIEM Wazuh**	6 days	**12/10/21 8:00**	**19/10/21 17:00**
30		Entrega 2	0 days	26/10/21 8:00	26/10/21 8:00
31		**Análisis y documentación del modelo cloud**	7 days	**19/10/21 8:00**	**27/10/21 17:00**
36		**Tecnología de orquestación de contenedores en cloud**	5 days	**27/10/21 8:00**	**2/11/21 17:00**
40		**Diseño del un modelo de arquitectura**	16 days	**2/11/21 8:00**	**23/11/21 17:00**
45		Entrega 3	0 days	23/11/21 8:00	23/11/21 8:00
46		**Implementación del modelo propuesto**	21 days	**10/11/21 8:00**	**8/12/21 17:00**
51		**Generación de casos de uso y pruebas de validación**	8 days	**8/12/21 8:00**	**17/12/21 17:00**
55		**Redacción y revisión final del proyecto**	6 days?	**20/12/21 8:00**	**27/12/21 17:00**
60		Entrega 4	0 days	28/12/21 9:00	28/12/21 9:00
61		**Elaboración de la presentación y video**	6 days	**28/12/21 8:00**	**4/01/22 17:00**
66		Video	0 days	4/01/22 17:00	4/01/22 17:00
67		**Preparación de la defensa**	4 days	**10/01/22 9:00**	**14/01/22 9:00**
70		Defensa	0 days	14/01/22 9:00	14/01/22 9:00

Figura 3: Resumen de tareas

1.7 Estado del arte

Como punto de situación previo, se realizará una breve introdución del estado de arte de los componentes clave que se van a tratar en el proyecto. Para ello, se tratan los sistemas SIEM, los sistemas de orquestación de contenedores (Kubernetes) y el modelo de proveedores cloud del mercado.
Es importante destacar que el estado de situación de cada una de estas tecnologías evoluciona muy rápidamente por lo que debería ser tenido en consideración y podría ser necesario reconsiderar su reevaluación.

Un SIEM nos permite recopilar, analizar, normalizar y correlar eventos de seguridad. En la actualidad existen múltiples soluciones de mercado, cada una con unas debilidades y fortalezas en función de la solución específica y de las distintas modalidades de uso. En este aspecto, el estado del arte de las soluciones SIEM está íntimamente ligado al estado del arte de la seguridad informática. Las clásicas soluciones de log y auditoría de seguridad de los sistemas aislados o perimetrados se han quedado cortas desde el momento en que el número de sistemas de información escalan exponencialmente y se deslocaliza por múltiples zonas del mundo.
Adicionalmente los **sistemas SIEM** también se han convertido en el corazón de los SOCs y por lo tanto en el núcleo principal por donde comenzar el diseño de la seguridad. Entre otras, estas consideraciones han llevado el concepto de SIEM a estar ya presente en la actualidad en empresas de todos los tamaños, obviamente en cada una de ellas estrechamente relacionada con su tamaño y madurez tecnológica.
En cuento a soluciones SIEM de mercado, podemos encontrar claramente diferenciados aquellos casos que siguen un modelo comercial, con ejemplos como *Splunk*, *IBM QRadar*, *Elastic Security*, *Fortinet*, etc. con las soluciones de código abierto como *Wazuh*, *OSSEC*, etc.
Tanto la tecnología de contenedores como el modelo de despliegue cloud, actualmente forman parte una de las tecnologías de uso diario en los equipos de desarrollo e infraestructura de las empresas. Consideradas antaño emergentes, hoy en día sería imposible pensar en los tiempos de puesta en marcha de soluciones tecnológicas y el uso de metodologías ágiles sin la adopción de estas.
La adopción y uso de estas tecnologías para el despliegue de cargas de trabajo se pueden ver consolidadas tanto en empresas más emergentes y disruptivas como en las líneas de estrategia de las grandes organizaciones. Los estudios del sector indican que el uso del modelo cloud está ampliamente afianzado en las empresas y que continuará creciendo de manera exponencial en los próximos años.[2]
En cuanto a la **tecnología de contenedores**, el sistema más extendido que da soporte a la misma en las empresas, es la aplicación *Docker*. Para la gestión de las infraestructuras y cargas de trabajo de contenedores a nivel empresarial se hace siempre necesario el uso de un sistema de **orquestación de contenedores**. Existen múltiples alternativas en el mercado tanto con licencia comercial a través de solucion como *Redhat Openshift*, *Vmware Tanzu*, etc. como soluciones de código abierto com *OKD* o el despligue nativo de *Kubernetes*. Actualmente el mercado ya está marcando una

2 https://discoverthenew.ituser.es/hybrid-it/2020/11/covid19-ha-impulsado-la-adopcion-de-la-nube-segun-forrester

tendencia estable y prácticamente todas la soluciones se basan en la tecnología Kubernetes (K8S) convirtiendo en un estándar de facto.
Actualmente existe 3 grandes proveedores de referencia en el **sector cloud**, que ordenados por cuota de mercado resultan: AWS (Amazon), Azure (Microsoft) y GCP (Google). Esta tríada, se disputa prácticamente la totalidad del mercado incluyendo unos porfolios de servicios de mas de 200 soluciones y que creen mes a mes. Existen otros competidores más residuales en este mercado, como puede ser *IBM Cloud*[3] o *Alibaba Cloud*[4] pero actualmente estan considerados como proveedores más de nicho.
Como veremos a lo largo de este proyecto, se irán analizado, seleccionando y justificando las tecnologías que irán con formando la arquitectura de la solución final del proyecto. A priori, antes de profundizar en el análisis detallado del proyecto y con las premisas actuales, los componentes a desarrollar serán un SIEM de código abierto sobre un despliegue en contenedores orquestados desde un clúster de Kubernetes y desplegado en el cloud público de AWS o Azure.

1.8 Breve sumario de productos obtenidos

A raíz de este trabajo se generará una documentación completa y un análisis de una solución de arquitectura para el despliegue del SIEM Wazuh sobre una infraestructura de un proveedor cloud y desplegado con tecnología de contenedores. Para ello se desplegará el servidor Wazuh sobre uno de los principales hyperscalers del panorama cloud actual, pero además de diseñara una arquitectura completa para la implementación de una solución de recopilación de eventos de seguridad tanto desde entornos On-premise, entorno cloud de múltiples proveedores como a través de agentes remotos distribuidos.
Adicionalmente se definirán y diseñaran distintos ejemplos configuración y prueba del entorno para proceder a validarlo y demostrar sus capacidades.
Además de la documentación teórica generada, se facilitará con el proyecto, a través de un repositoro de código públicos, todos los ficheros de configuración y scripts de automatización que hayan podido ser necesarios para los desarrollo de los diseños, despliegues y configuración de los modelos de prueba.

3 https://www.ibm.com/uk-en/cloud
4 https://eu.alibabacloud.com/

2 Sistemas de gestión de eventos e información de seguridad

2.1 SIEM

En el pasado, la seguridad de las organizaciones se basaba en una protección del perímetro de red en la que se ofrecían los servicios corporativos, así como la seguridad individual de cada uno de los sistemas de información. Este modelo, a priori válido para otros escenarios, actualmente se considera insuficiente en la gran mayoría de los casos debido a la proliferación de múltiples sistemas de información heterogéneos donde la seguridad se encuentra fuertemente acoplada entre y dentro de ellos.
Aquí es donde nace la necesidad de interrelacionar o correlar eventos de seguridad que se generan desde decenas de sistemas individuales (switches, firewalls, equipos finales, etc.) en un sistema central donde se pueda, en primera instancia, almacenar durante todo el tiempo que puedan ser necesarios, así como su tratamiento y análisis posterior. Toda esta información en un mismo sistema, facilita tener una visión rápida del estado de la seguridad de la organización en cada momento, así como una mayor capacidad de detección y respuesta ante un incidente de seguridad.
Así es como surge el actual concepto de **sistema de gestión de eventos e información de seguridad** o **SIEM** (por sus siglas en inglés, **Security Information and Event Management**) como un sistema de información cuya función principal es centralizar el almacenamiento y los eventos relevantes de seguridad.
Los eventos, logs o información de seguridad que se registran en un SIEM son la información de seguridad (registros de acceso al sistema, auditoría de cambios que ha realizado un usuario en una plataforma, tráfico en un interfaz de red, etc.) que se generan y se registran en este sistema de información.

Las principales capacidades que un SIEM aporta son las siguientes:

- Correlación: Es capaz de analizar eventos de seguridad complejos que provienen de múltiples orígenes.
- Agregación de datos: Un SIEM es capaz de recolectar de forma consolidada eventos de múltiples sistemas de información distintos (servidor, equipos de usuario, bases de datos, elementos de red, otros agentes, etc.).
- Alertas y respuesta a incidentes: Es capaz de generar eventos de alerta y notificaciones a través de diversos canales (email, sms, etc.) en base a la información recolectada. Adicionalmente estes sistemas incorporan capacidad de respuesta a incidentes de seguridad, que como su propio nombre indica, permite tomar acciones automáticas basadas en las alertas.
- Almacenamiento y retención de los eventos de seguridad: Es capaz de almacenar toda la información de los eventos e inteligencia de seguridad generada así como de definir políticas de retención específicas para los datos que maneja.
- Redundancia y escalabilidad: El SIEM es capaz de proveer una solución de redundancia en caso de fallo y escalabilidad para adaptarse al procesado de la información que gestiona.
- Cumplimiento normativo: Este sistema permite automatizar y mejorar tareas de identificación y cumplimiento normativo.
- Consulta y presentación de datos e informes: Es capaz de realizar consultas de datos avanzadas y representaciones gráficas de los datos, a la vez que la generación de paneles e informes de mando avanzados.

{inve01}

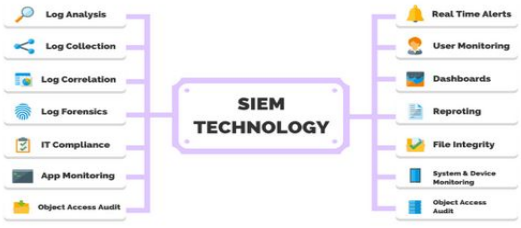


Figura 4: Detalle de capacidades de un SIEM [5]

En cuanto a auditoría y eventos de seguridad, además de los SIEM, siempre aparecen otros sistemas asociados a ellos. Estos son los IDS (sistema de detección de intrusos; en inglés, “Intrusion Detections System”) o los IPS (Intrusion Prevention System). Los IDS son sistemas cuyo propósito principal consiste en la detección de accesos no autorizados y la generación de alertas. Para ello monitorizan los sistemas donde se implementan, generalmente con técnicas basadas en las búsqueda de firmas o anomalías y comportamientos sospechosos. Actualmente existen múltiples tipos de IDS, catalogados generalmente en base a su sistema de detección o al punto y forma de recolección de los datos. En base a esto último, fundamentalmente suelen englobarse en dos tipos:

- HIDS: El sistema de detección de anomalías se instala a nivel del equipo host donde se recolectan.
- NIDS: El sistema de detección de anomalías se instala en un punto de la red sobre la que se recolectan los

5 Fuente: https://purplesec.us/

eventos.
A diferencia de los IDS, que solo detectan los accesos sospechosos y generan alertas, los sistemas IPS permiten adicionalmente actuar en base a los eventos detectados para tomar acciones que protejan el sistema.

De este manera, los sistemas IDS, IPS y SIEM son sistemas que se complementan para la gestión y el gobierno de los eventos de seguridad.

{url06}{url05}

2.2 Análisis y selección del SIEM

Una vez se ha introducido lo que es un SIEM, sus principales características y se han detallado otros conceptos base de estos sistemas, se procederá al análisis de las soluciones de mercado para la elección de la herramienta a implementar.

Figura 5: Cuadrante mágico Gartner

En primer lugar, partimos del análisis del denominado “cuadrante mágico” que elabora cada año la consultora Gartner con las tendencias de mercado de los sistemas de SIEM.
El grueso de las soluciones que se analizan dentro del informe de Gartner incluye sistemas propietarios, solo con opciones de código cerrado y de pago. Como premisa de este estudio, el SIEM a implementar se desarrollará íntegramente con software libre por lo que de este informe sólo se tendrán en consideración aquellas que presenten alguna opción compatible de uso libre. Fundamentalmente, dentro del marco de soluciones lideres, los actores elegibles se reducen a *Splunk Free* y la versión libre de *Elastic*, pero ambas opciones implican el sacrificio de múltiples funcionalidades que lo limitan como solución.

Por este motivo, se realiza un análisis más centrado en las soluciones SIEM de código abierto del mercado actual y analizamos otros trabajos académicos y páginas del sector donde ya se haya realizado un análisis de las características de varias soluciones.

En base a ello, se elabora el siguiente cuadro que nos ayudará a la selección de la herramienta:

Tabla 1: Características de los sistemas evaluados

	Elastic Security[6]
	✔ Interfaz de usuario avanzada con soporte para búsquedas, filtros y soporte para reportes a través del uso de la suite Kibana. ✔ Compatible con modelo de arquitectura de despliegue en cloud. ✔ Sistema fácilmente escalable y con capacidad de crear arquitectura de alta disponibilidad. ✔ Rendimiento excepcional en el indexado y análisis de los datos. ✖ Se puede contratar soporte adicional con el fabricante pero no en la versión gratuita. ✖ La versión gratuita tiene una limitación de funcionalidades a la hora de la correlación de datos y no incorpora un sistema de agentes para endpoint.
	Splunk Free[7]
	✔ Interfaz de usuario avanzada con soporte para búsquedas, filtros y soporte adicional para informes avanzados. ✔ Sistema de agentes para endpoint. ✖ Se puede contratar soporte adicional con el fabricante pero no en la versión gratuita. ✖ La versión gratuita tiene una limitación de funcionalidades muy importante y acota el uso máximo de indexado de información a 500MB diarios
	AlienVault OSSIM[8]
	✔ Interfaz de usuario avanzada con soporte para búsquedas y filtros. ✔ No compatible con modelo de arquitectura de despliegue en cloud ya que esta diseñado en base a una arquitectura de servidor centralizado. ✔ Implementación en un número amplio de arquitecturas y con una buena comunidad de desarrolladores y soporte. ✖ Se puede contratar soporte adicional con el fabricante pero no en la versión gratuita. ✖ La versión gratuita tiene limitación de funcionalidades.

6 https://www.elastic.co/
7 https://www.splunk.com/
8 https://cybersecurity.att.com/products/ossim

	Wazuh[9]
	✔ Se puede contratar soporte adicional con el fabricante. ✔ Interfaz de usuario avanzada con soporte para búsquedas, filtros y soporte extendido para reportes avanzados. ✔ Compatible con modelos de arquitectura de despliegue en nube. ✔ Capacidad de despliegue en múltiples sistemas operativos y con soporte de ejecución en modelo de contenedores. ✔ Documentación online con decenas de modelos de despliegue y con una comunidad de desarrollo muy importante y activa. ✔ Capacidad de orquestar y automatizar los despliegues con herramientas de IaC[10] y soporte nativo a Kubernetes. ✔ Incorpora un sistema de agentes para endpoint con EDR. ✔ Sistema fácilmente escalable y con capacidad de crear arquitectura de alta disponibilidad. ✔ Rendimiento excepcional en el indexado y análisis de los datos. ✔ Soporte para detección y respuesta a incidentes. ✔ Soporte para integración con herramientas de seguridad en la nube. ✔ Soporte para integración de un sistema de seguridad para entornos de contenedores. ✔ Soporte para la gestión y configuración de activos. ✔ Incluye solución de soporte para el cumplimiento de marcos normativos (PCI-DSS, GDPR, HIPAA, etc). ✖ Arquitectura basada en múltiples componentes lo que puede generar cierto grado de complejidad.

✔ Valora Positivamente
✖ Valora Negativamente

Tal y como se puede comprobar en el análisis anterior, de la lista de SIEM analizados, la opción Wazuh es la que implementa un mayor número de funcionalidades y características positivas. Como solución más completa se postula como mejor candidato. Para confirmarlo, se evalúa ya en base a las premisas que se marcaron al inicio del proyecto así como en base a los objetivos propuestos.

Una de las premisas principales que se planteaba se basaba en el uso de una solución gratuita de código abierto ya que se va a implementar en una organización con amplia experiencia en este ámbito y con un modelo de trabajo y operación ya consolidad en el uso de este tipo de herramientas. Se descartan fundamentalmente las opciones de Splunk Free y AlienVault OSSIM, ya que su versión gratuita se ve muy limitada en funcionalidades. La siguiente premisa clave es que se va a desplegar sobre un modelo de infraestructura y servicios cloud y empleando un sistema de orquestación y despliegue basado en contenedores. Así, de las últimas dos opciones nos decantamos por Wazuh en detrimento de Elastic, fundamentalmente porque éste tiene un sistema de agentes muy completo que resulta necesario para implementar el modelo completo de SIEM para esta solución, además de por que Wazuh dispone de compatibilidad para el despliegue en contenedores y soporte nativo del fabricante con Kubernetes.

Cabe destacar además que, Wazuh implementa un conjunto completo de funcionalidades muy superior a cualquiera de las alternativas analizadas.

Por los motivos expuestos, se selecciona Wazuh como solución SIEM para el desarrollo de este proyecto.

{url07}{tfm01}

9 https://wazuh.com/
10 https://es.wikipedia.org/wiki/Infraestructura_como_c%C3%B3digo

3 WAZUH

3.1 Introducción

Como ya hemos visto en apartados anteriores, Wazuh es una plataforma SIEM gratuita y de código abierto centrada en la detección de eventos y monitorización de seguridad y con capacidad de ofrecer funcionalidades para la gestión de la respuesta a incidentes de seguridad y herramientas de cumplimiento normativo. Es una solución tan versátil y completa que puede emplearse tanto para la monitorización de gran cantidad de endpoints como para la monitorización de la seguridad de servicios e infraestructura desplegada en entornos cloud. Además del soporte a los elementos anteriores, incluye soporte de monitorización de eventos de seguridad para aquellas organizaciones que estén desplegando cargas de trabajo empleando microservicios en contenedores además de otras muchas funcionalidades que se verán.

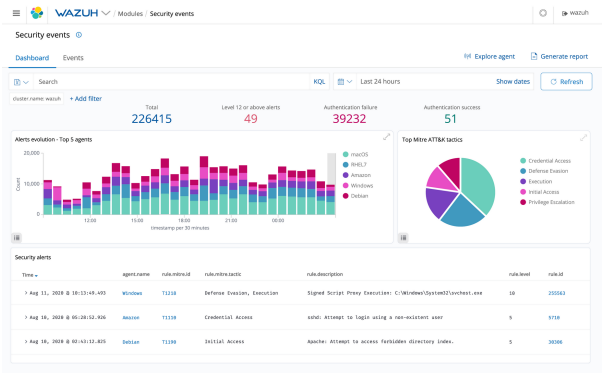


Figura 6: Wazuh Dashboard

Destaca la amplia documentación online que provee el fabricante a través de la web oficial[11], donde además de los habituales manuales y guías de instalación que suele incluir cualquier solución, podemos encontrar diferentes modelos de ejemplo de arquitecturas propuestas sobre distintos diseños de infraestructura, así como un análisis detallado de posibles casos de uso y como implementarlos con cada plataforma.
Su andadura comienza a mediados del ano 2015 como una bifurcación de la rama de desarrollo de otra solución de seguridad denominada OSSEC, pero en la actualidad Wazuh ya cuenta con una amplia comunidad de más de 100.000 desarrolladores que contribuyen a su código activamente y es cosiderado una solución robusta y madura.

{url04}

3.1.1 Modelos de uso e implementación

La plataforma Wazuh se ofrece fundamentalmente siguiendo tres modelos de uso o implementación. Por un lado, se puede desplegar como solución autónoma, empleando para ello cada cliente su propia infraestructura, permitiendo aprovechar los modelos de cargas de trabajo que cada uno posea e implementarlo como sistema sobre una base existente. En este modelo, el fabricante permite la posibilidad de contratar soporte adicional sobre el producto.
Por otro lado, también existe la modalidad de despliegue siguiendo un modelo de software como servicio (SaaS). En esta modalidad el fabricante ofrece ya implementada un solución Wazuh lista para ser usada con una instancia individual para cada cliente. Este modelo destaca por ser una solución en modalidad cloud completamente lista para comenzar a trabajar, donde el usuario puede empezar a usar la solución y conectar flujos de información y endpoints directamente tras realizar unos pasos de personalización y configuración básica.
Por último, existe una tercera forma de implementación, de reciente aparición y auspiciada por el auge de los *marketplace* de las plataformas cloud del mercado. Es una solución muy similar al primer modelo, pero aquí ya se parte de unos modelos de despliegue automático a través de una solución empaquetada sobre la infraestructura y los componentes del proveedor cloud Amazon AWS. Esta solución cuenta con el soporte de la empresa que lo empaqueta y mantiene.
Para este proyecto, como vimos en la contextualización del primer capítulo, partimos de una empresa con un modelo de infraestructura y servicios claramente definidos siguiendo el modelo cloud y con el conocimiento para mantener, operar y desplegar correctamente el sistema de despliegue autónomo. Además, este permite a la empresa una mayor flexibilidad y un modelo de coste más contenido, por lo que claramente será el modelo de uso a implementar.

3.1.2 Componentes y características

Dentro de las características de la plataforma Wazuh, podemos encontrar las siguientes funcionalidades:

11 https://documentation.wazuh.com/

- **Análisis de seguridad:** Este sistema tiene la capacidad de recopilar, agregar, indexar y analizar datos de seguridad de múltiples orígenes. De este modo, es capaz de proporcionar la inteligencia de seguridad necesaria para dar soporte a la detección de las amenazas de seguridad más novedosas y sofisticadas.
- **Detección de intrusiones:** A través del uso de los agentes instalados en los equipos, estos adquieren la funcionalidad de HIDS. Este agente es capaz de escanear el sistema anfitrión en busca de malware, rootkits y otros comportamientos sospechosos. Así, le confiere la capacidad de detectar nuevos archivos ocultos dentro del sistema, puertos de red a la escucha e invocaciones sospechosas de funciones del sistema. Para conseguirlo, el agente emplea en un sistema de detección de intrusiones basado en firmas e indicadores de compromiso.
- **Análisis de log de datos:** A través de los agentes, Wazuh es capaz de obtener información del sistema operativo del equipo, las aplicaciones y las entradas del registro en tiempo real y enviarla al sistema central para su procesado. A través de reglas aplicadas a estos registros se pueden detectar errores de configuración, violaciones de políticas y actividades sospechosas.
- **Monitorización de integridad de ficheros:** Wazuh, a través de los agentes, supervisa el sistema de archivos e identifica cambios en el contenido, permisos, propiedad y los atributos de los archivos que pueden suponer algún riesgo de seguridad. Además, identifica de forma nativa a los usuarios y las aplicaciones que utilizan los archivos de un equipo.
- **Detección de vulnerabilidades:** Los agentes Wazuh extraen el inventario de software del sistema y envían esta información al servidor central, donde se contrasta con las bases de datos CVE[12] (Common Vulnerabilities and Exposure) de MITRE ATT&CK[13] para identificar software vulnerable conocido. Esta evaluación permite encontrar los puntos débiles de estos sistemas y tomar las acciones correctivas necesarias desde una fase temprana.
- **Evaluación de configuración:** Wazuh supervisa los ajustes de configuración del sistema y las aplicaciones para asegurarse que cumplan con las políticas de seguridad, estándares y/o guías de bastionado de la organización. Los agentes realizan análisis periódicos para detectar aplicaciones vulnerables, que no están actualizadas o configuradas correctamente.
- **Respuesta a incidentes de seguridad:** Proporciona respuestas activas y contramedidas para abordar las amenazas de seguridad. Así, puede bloquear el acceso a un sistema desde la fuente de la amenaza cuando se cumplen ciertos criterios. Además, se puede utilizar para ejecutar de forma remota comandos o consultas sobre el sistema, identificando indicadores de compromiso de los sistemas y ayudando a realizar tareas de respuesta a incidentes y de análisis forense temprano.
- **Controles de cumplimiento normativo:** Wazuh proporciona algunos de los controles de seguridad necesarios para cumplir con los estándares y regulaciones del mercado. Estas características, combinadas con su escalabilidad y soporte multiplataforma, ayudan a las organizaciones al cumplimiento regulatorio.
- **Seguridad en la nube:** Wazuh ayuda a monitorizar la infraestructura cloud, utilizando módulos de integración que pueden extraer datos de la auditoria de seguridad de proveedores cloud, como Amazon AWS, Azure o Google Cloud. Además, proporciona reglas para evaluar la configuración y la postura de seguridad de estos entornos, detectando fácilmente debilidades de seguridad sobre los mismos.
- **Seguridad en contenedores:** Proporciona visibilidad de seguridad en contenedores basados en Docker, monitorizando su comportamiento y detectando amenazas, vulnerabilidades y anomalías. El agente de Wazuh incorpora integración nativa con el motor Docker, permitiendo monitorizar imágenes, volúmenes, configuraciones de red y contenedores en ejecución sobre el sistema donde esta instalado a través de la información que recibe del agente. Todo ello, recopilando y analizando continuamente información detallada en tiempo de ejecución para garantizar la seguridad y el control de las cargas de trabajo desplegadas en estos sistemas.

3.2 Arquitectura

La arquitectura de la plataforma Wazuh se basa en agentes ligeros multiplataforma que funcionan como HIDS dentro de los sistemas monitorizados y envían reportes a un servidor centralizado en el que se realiza el análisis y procesado de los datos recolectados. Como se puede ver en la siguiente figura, la arquitectura de la plataforma Wazuh se basa en 3 bloques claramente definidos de componentes que se intercomunican entre ellos, y cada uno está especializado en implementar una funcionalidad específica:
A continuación se detalla un diagrama de componentes de la misma:

12 https://cve.mitre.org/
13 https://attack.mitre.org/

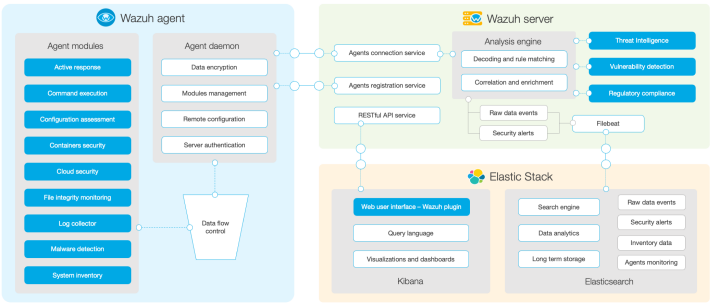


Figura 7: Arquitectura de componentes Wazuh [14]

3.2.1 Wazuh server

El componente Wazuh server se encarga de analizar los datos recibidos de los agentes, procesar los eventos a través de decodificadores y reglas, y utilizar la inteligencia de amenazas para buscar los indicadores de compromiso (IOC).
Para poder realizar estas funciones, internamente la arquitectura del servidor Wazuh se subdivide a su vez en diferentes servicios, cada uno con una funcionalidad específica. Estos son:

- **Wazuh RESTful API:** Este servicio proporciona la capa de API que permite a otros sistemas poder interactuar con el SIEM.
- **Analysis engine:** Este proceso se encarga de la parte analítica de los datos del servidor. Utiliza decodificadores que permiten identificar el tipo de información que se está procesando (por ejemplo, eventos de Windows, registros del servicio web, etc) para extraer los eventos con datos relevantes de los mensajes. Mediante el uso de reglas, se identifican patrones específicos en los eventos decodificados para, a continuación poder desencadenar alertas o contramedidas automatizadas si fuese necesario.
- **Agents connection service:** Este componente se encarga del proceso de control y gestión centralizada de los agentes de manera remota.
- **Agents registration service:** Este componente se encarga del proceso de registro de nuevos agentes gestionando toda la etapa de aprovisionamiento de nuevos sistemas y de la seguridad del proceso.
- **Wazuh cluster daemon:** Este proceso se encarga de la gestión del cluster del servicio Wazuh y, por tanto, es el que se encarga de proporcionar el soporte de la comunicación, control, alta disponibilidad y escalado de los posibles servidores Wazuh desplegados.
- **FileBeat (o Beats):** Este componente es el que se encarga del intercambio de información entre los procesos del servidor Wazuh (mediante el envío de eventos, alertas, información de control, etc) y el componente *Elasticsearch* que veremos en detalle un poco más delante.

3.2.2 Wazuh Agent

El sistema de agentes se basa en una solución de software multiplataforma (soportando la mayor parte de los sistemas operativos disponibles en la actualidad) que se instala en los equipos sobre los que se quiere realizar la monitorización con este sistema. Estos agentes se comunican directamente con el servicio central de Wazuh enviando toda la información sobre eventos de seguridad casi en tiempo real y empleando un canal de comunicación seguro que implementa medidas de autenticación y encriptación en tránsito.
Este agente está diseñado para monitorizar un amplio número de indicadores sin que afecte al rendimiento final del equipo y para que el consumo de recursos en el proceso sea mínimo, y aportando, entre otras, las capacidades de recolección de logs, ejecución de comandos, gestión de la configuración remota, detección de Malware, respuesta activa frente a incidentes e inventariado del sistema.

En el siguiente anexo se incluye un detalle del proceso de instalación de agente para los sistemas operativos Linux y Windows, seleccionados como ejemplo por ser dos de los sistemas operativos de servidor más habituales.

3.2.3 Elastic

Este sistema se encarga de proveer el indexado y almacenamiento de los eventos de seguridad que se generan a través del componente Wazuh server (y que a su vez provienen de los agentes). Además, es el sistema que aporta a Wazuh la interfaz de usuario desde donde se realiza la gestión y administracion del entorno y desde donde se puede realizar la analítica de los datos, así como la generación de los informes.
Elastic es una pila de subsistemas de código abierto, también conocida anteriormente por las siglas ELK porque integra los componentes Elasticsearch, Kibana, Beats y Logstash. Para la plataforma Wazuh solo se emplean los componentes Filebeat, ElasticSearch y Kibana.

14 https://documentation.wazuh.com/

- **Elasticsearch:** Este componente aporta un motor de análisis, indexado y búsqueda para el consumo de los datos que se reciben a través del servidor central de Wazuh. Para trabajar, el sistema emplea habitualmente 3 índices, que se regeneran diariamente. Son los siguientes:
  - ***wazuh-alerts:*** En él se registran las alertas generadas por el servidor Wazuh, las cuales se crean cada vez que un evento dispara una regla predefinida en el sistema.
  - ***wazuh-events:*** En él se registran todos los eventos de auditoría recibidos de los agentes.
  - ***wazuh-monitoring:*** En este último índice se registran datos relacionados con el estado de los agentes a lo largo del tiempo.
- **Kibana:** Proporciona al sistema una interfaz web flexible e intuitiva para extraer, analizar y visualizar datos. La interfaz de usuario web de Wazuh se ha integrado y se ofrece desde *Kibana*, como un componente más del mismo. Incluye paneles listos para usar para eventos de seguridad, cumplimiento normativo (*PCI-DSS*, *GDPR*, *HIPAA*, *NIST 800-53*, etc.), aplicaciones vulnerables detectadas, datos de monitorización de integridad de archivos, resultados de evaluación de configuración, eventos de infraestructura en la nube, etc.
  El modelo de comunicación se basa en la recepción y consulta de los datos que ya están almacenados en el servicio de *Elasticsearch*, mientras que la comunicación de gestión y control con el servidor central se hace a través del API Wazuh.

4 Modelo tecnológico

Antes de continuar con el diseño y propuesta de la solución final de arquitectura introduciremos ciertas decisiones previas e información sobre el modelo tecnológico sobre el que vamos a apoyarnos para dar soporte a la solución.
Como objetivos principales del proyecto se marcó el despliegue de una solución empleando un modelo de cloud para dar soporte a las cargas de trabajo que necesita el proyecto, empleando un modelo de despliegue basado en contenedores orquestados empleando tecnología Kubernetes.

4.1 Cloud

Actualmente, existen cientos de proveedores que ofrecen en el mercado soluciones para ejecutar cargas de trabajo siguiendo un modelo de cloud público. Para este proyecto, solo hemos tenido en cuenta los 3 actores principales del mercado de cloud, según el último análisis de la consultora Gartner[15].

4.1.1 AWS

AWS[16],acrónimo de *Amazon Web Services*, es el nombre comercial de la solución de servicios de cloud ofrecida por el fabricante Amazon.com. AWS proporciona más de 200 servicios de computación cloud, que se ofrecen en múltiples modelos tanto software como servicio (SaaS), plataforma como servicio (PaaS) e infraestructura como servicio (IaaS). En conjunto ofrecen una plataforma completa para que cualquier tipo de compañía pueda llevar a la nube sus cargas de trabajo. Ofrece sus servicios en más de 18 regiones geográficas, y cada una de ellas con zonas de disponibilidad (que son los datacenter que ofrecen los servicios) redundadas para que los clientes puedan implementar arquitectura de alta disponibilidad y redundancia, permitiendo la construcción de soluciones con garantía de resiliencia.
Para el diseño de soluciones de arquitectura, este proveedor incorpora un modelo de diseño propio denominado *WELL-Architected Framework*, que ayuda a los arquitectos de soluciones en modelo nube a crear una infraestructura para aplicaciones y cargas de trabajo segura, de alto rendimiento, resistente y eficiente. Este marco, basado en cinco pilares (excelencia operativa, seguridad, fiabilidad, eficiencia de rendimiento y optimización de costos), ofrece un enfoque coherente para que los clientes evalúen las arquitecturas e implementen diseños robustos sobre los que se garanticen los objetivos dentro de estos 5 pilares.
AWS arranca su servicio en el año 2006, siendo desde el principio el actor principal de la tecnología cloud y proveedor líder de referencia. A través del siguiente enlace se puede consultar todo el catálogo de productos y servicios.

4.1.2 Microsoft Azure

Microsoft Azure[17] es el nombre de la solución cloud ofrecida por el proveedor Microsoft. Actualmente, ofrece una amplia gama de productos y servicios en modalidad nube para construir, desplegar y administrar aplicaciones y servicios mediante el uso de sus centros de datos. Proporciona software como servicio (SaaS), plataforma como servicio (PaaS) e infraestructura como servicio (IaaS) y es compatible con muchos lenguajes, herramientas y marcos de programación diferentes, incluidos software y sistemas específicos de Microsoft, pero también se incluye el de terceros. Dentro del conjunto de soluciones de mercado, Azure destaca por ofrecer capas de funcionalidad extra a soluciones y aplicaciones del propio fabricantes Microsoft y con amplio recorrido y cuota de mercado. Así, sobresale el servicio *Microsoft Azure AD* que funciona como una extensión en la nube y siguiendo el modelo de software como servicio de la más que conocida solución de directorio y gestión de identidades, *Microsoft Active Directory*.
Comenzó su andadura en el año 2010 y desde entonces, año a año ha sido el competidor principal frente a AWS por el liderazgo del nicho de mercado cloud. A través del siguiente enlace se puede consultar todo el catálogo de productos y servicios.

4.1.3 GCP (Google Cloud Platform)

Google Cloud Platform[18], conocido por sus siglas en ingles GCP, es el nombre comercial de la solución cloud ofrecida por el fabricante Google. Al igual que como vimos en los apartados anteriores también permite el uso de soluciones para la construcción de arquitecturas siguiendo el modelo cloud. Cabe destacar que este proveedor no tiene el mismo número de centros de datos, ni zonas geográficas, ni el mismo catálogo de servicios a nivel de despliegue de infraestructura que los otros dos proveedores anteriores. Su punto fuerte, donde sí supera a sus competidores, está en la entrega de servicios especializados de inteligencia artificial y aprendizaje automático. A través del siguiente enlace se puede consultar todo el catálogo de productos y servicios.

15 https://www.gartner.com/technology/media-products/reprints/AWS/1-271W1OTA-ESP.html
16 https://aws.amazon.com/es/what-is-aws/
17 https://azure.microsoft.com/
18 https://cloud.google.com/

Figura 8: Hyperscalers Cloud

Para este proyecto, se ha tomado la decisión de desplegar todos los componentes necesarios para ejecutar el sistema Wazuh dentro del cloud del proveedor AWS.
Se selecciona el proveedor AWS por ser líder del cuadrante Gartner durante los últimos años y porque, como introducimos en la contextualización, nuestra organización ya tiene conocimiento y despliega cargas de trabajo en este proveedor. Adicionalmente, destacar que tiene todos los servicios necesarios para ofrecer el modelo de orquestación y contenedores necesario, además de proveer servicios sobre un sistema escalable y con modalidad Pay-as-you-go, que nos permite un modelo de costes óptimo para la solución que buscamos.

4.2 Contenedores con Docker

Tal y como se introdujo como premisa del proyecto y durante la contextualización, para el modelo de despliegue y la entrega de las aplicaciones necesarias para la ejecución del SIEM Wazuh, se empleará el modelo de despliegue de cargas de trabajo empleando contenedores. No entra dentro de los objetivos de este proyecto introducir en detalle los componentes técnicos y arquitectura del modelo de despliegue en contenedores ni del sistema de orquestación que emplearemos, aunque si se introducirán más adelante los componentes básicos necesarios para entender el modelo desarrollado y poder realizar el despliegue básico.
Un contenedor es sencillamente un proceso para el sistema operativo que internamente contiene la aplicación que queremos ejecutar y todas las dependencias derivadas de la misma. Empaquetamos una aplicación en una unidad estandarizada para desempeñar un servicio que contiene lo necesario para funcionar como un todo (código, librerías, software, etc), envuelto bajo la analogía de un contenedor. De este modo, los componentes que hemos visto que requiere la solución Wazuh se desplegarán encapsulados en esta modalidad de sistema de ejecución.
Para dar soporte a este modelo de despliegue, la solución tecnológica subyacente sobre la que se apoya Kubernetes y que emplearemos en nuestra solución es ***Docker***[19]. En el siguiente enlace se puede consultar información detallada sobre esta tecnología, que como se ha introducido, no forma parte del objetivo de este trabajo.

4.3 Orquestación con Kubernetes

{url13}

Para la gestión del modelo de despliegue en contenedores es necesario un sistema de orquestación, que de soporte al modelo de gestión operativo y proporcione las herramientas necesarias para dar apoyo al trabajo diario con este modelo y a la automatización de todos los procesos involucrados.
Como solución de orquestación en este proyecto, usaremos ***Kubernetes (k8s)***[20]. A través de Kubernetes se administra un cluster de nodos de trabajo en forma de equipos de cómputo y se orquestan contenedores dentro del mismo, para que ejecuten un determinado trabajo. Los contenedores se ejecutan dentro de agrupaciones lógicas llamadas pods, que además posibilitan el ajuste de la escala en uno o más contenedores de aplicación.
En Kubernetes se utilizan objetos de API para describir la configuración en base a un estado deseado del cluster: qué aplicaciones y otras cargas de trabajo se quieren ejecutar, qué imágenes de contenedores usan, el número de réplicas, qué red y qué recursos de almacenamiento se quiere que tengan disponibles, etc. También se puede usar el API de Kubernetes directamente para interactuar con el cluster y especificar o modificar el estado deseado de cualquier despliegue, pero lo más habitual, por simplicidad y orden, es el empleo de ficheros con formato de marcado *yaml* con la definición de los objetos del API.
Una vez que se especifica el estado deseado, el Plano de Control de Kubernetes realizará las acciones necesarias para que el estado actual del clúster coincida con el estado deseado que se ha definido. La aplicación cliente que se emplea para esta interacción con los clusters de Kubernetes, se denomina *kubectl*[21].
En el siguiente enlace se puede consultar la información completa sobre esta tecnología que, como ya hemos dicho, su detalle no forma parte del objetivo de este trabajo, pues partimos de la premisa de que ya era un tecnología ampliamente conocida en la organización de referencia. De este modo, en este apartado solo se introducirá el detalle tecnológico de los objetos básicos de Kubernetes, que son necesarios para el entendimiento de los apartados de arquitectura e implementación que se necesitará para la puesta en marcha de la solución Wazuh.

19 https://www.docker.com/
20 https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/
21 https://kubernetes.io/docs/reference/kubectl/overview/

Los objetos básicos que podemos encontrar en Kubernetes y que se van a emplear en el despliegue del sistema Wazuh son los siguientes:

- **Pod:** Como hemos visto, es la unidad básica de ejecución de Kubernetes formada por un grupo de uno o más contenedores (contenedores *Docker* en esta implementación), con almacenamiento/red compartidos, y unas especificaciones de cómo ejecutar los contenedores. Los contenidos de un p*od* son siempre coubicados, preestablecidos y ejecutados en un contexto compartido. El concepto de pod también posibilita la aplicación una capa de aislamiento, abstracción y seguridad gracias al empleo de tecnologías de aislamiento que ofrece el sistema de contenedores.
- **Service:** Un servicio, o S*ervice*, es un objeto de Kubernetes que describe cómo se accede a las aplicaciones, como un conjunto de pods, y que se emplea para publicar una aplicación a través de exposición de puertos y el aprovisionamiento de balanceadores de carga. Además, los servicios incluyen una interfaz de descubrimiento para las distintas aplicaciones que se van desplegando dentro del cluster.
- **Volume:** El espacio de contenedores no es persistente, por lo que sólo existe mientras dure el ciclo de vida del contenedor. Los volúmenes son los objetos Kubernetes que se emplean en la definición de almacenamiento persistente dentro del espacio de contenedores.
- **Namespace:** Los espacios de nombres o *namespaces* proporcionan la capa de abstracción necesaria que proporciona aislamiento de componentes y recursos dentro de un cluster y sobre los que se aplica una política común.
- **Secrets:** Los secretos, o S*ecrets*, proporcionan un sistema para el almacenamiento de contraseñas, tokens o cualquier tipo de información confidencial que se quiera preservar en el entorno de objetos Kubernetes.
- **ConfigMap:** Es el objeto Kubernetes que se emplea para el almacenamiento de datos no sensibles en formato clave-valor y que permite compartir información de configuración dentro del sistema.

Kubernetes contiene además, abstracciones de nivel superior llamadas *Controladores (*Controlers*)*, que en base a los objetos básicos proporcionan funcionalidades adicionales sobre ellos. Entre ellos destacan:

- **ReplicaSet:** Este objeto se encarga de definir el número de replicas iguales de un pod que se despliegan dentro de una solución de aplicación y que deben de ejecutarse en todo momento.
- **Deployment:** Este objeto es un controlador de nivel superior al *ReplicaSet* y al *Pod*. Se trata de un objeto que permite definir de una manera declarativa el estado deseado de un pod y asociar al mismo un *ReplicaSet*.
- **StatefulSet:** Este objeto gestiona el despliegue y escalado de un conjunto de *Pods*, y garantiza el orden y unicidad de los mismos. Al igual que un *Deployment*, un *StatefulSet* gestiona *Pods* que se basan en una especificación idéntica de contenedor. A diferencia de un *Deployment*, un *StatefulSet* mantiene una identidad asociada y el control de los *Pods* que define.

5 Arquitectura y diseño de la solución

A continuación, una vez seleccionado el sistema SIEM que se va a desplegar, el proveedor donde se van a implementar las cargas de trabajo del core del sistema y comentado el modelo y marco tecnológico subyacente, se pasa a detallar la arquitectura de la solución y sus componentes. Como se verá seguidamente, la arquitectura está formada por el core de la solución Wazuh que se despliega sobre un cluster de Kubernetes implementado sobre el proveedor cloud Amazon AWS y con un diseño resiliente, donde destaca el soporte para escalado horizontal automático tanto a nivel de nodos de cómputo como a nivel de capacidad de escalado dentro de la aplicación. Como se plantea en los objetivos, se pretende una solución que permita el soporte para la monitorización de agentes tanto en entorno multicloud, On-premise como de manera distribuida sin estar asociados a ningún centro de datos o nodo principal de comunicaciones. Estos, se han representado en la arquitectura como clientes del SIEM dentro del bloque de cada uno de los centros de datos identificados como On-premise o Azure. Adicionalmente también se ha considerado la existencia de agentes en modelo "*roadwarrior*" conectados y enviando eventos de manera autónoma al sistema SIEM.

5.1 Arquitectura propuesta

En la siguiente figura se representa el modelo completo de arquitectura que se diseña para esta solución:

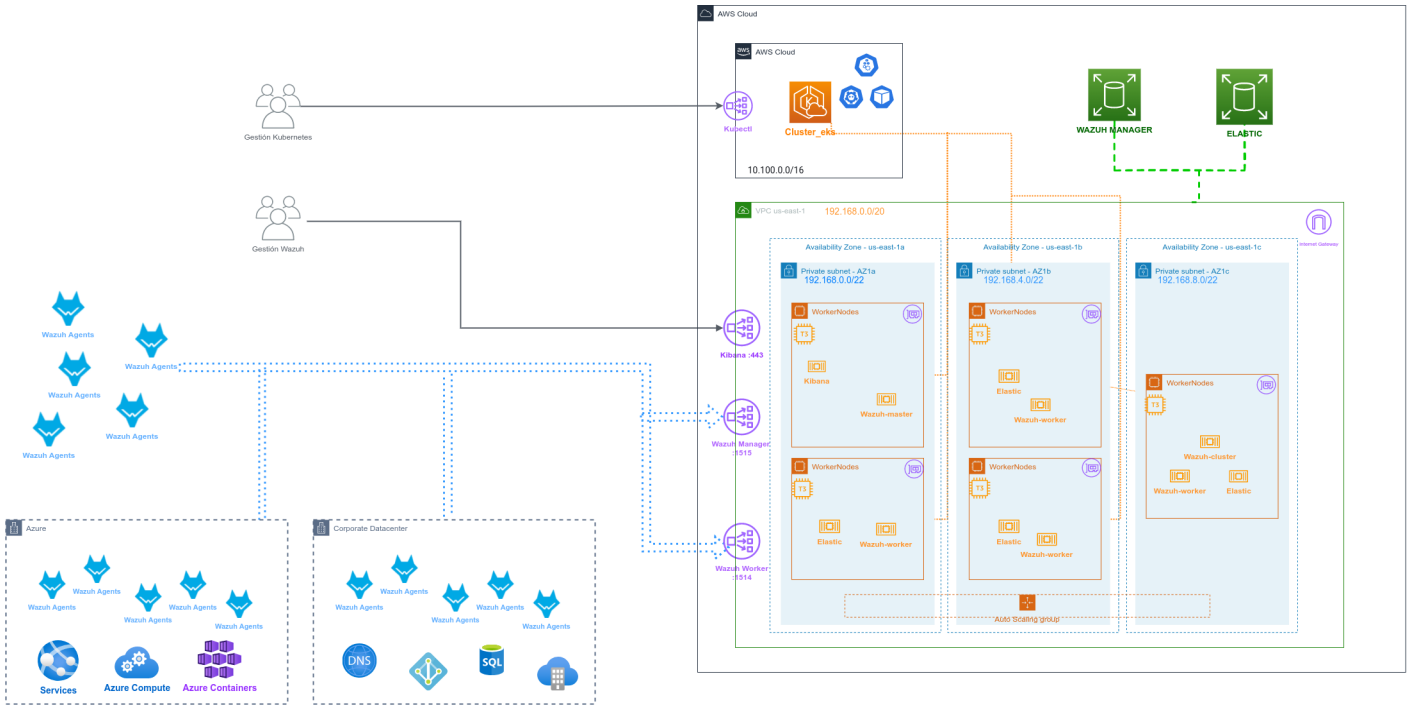


Figura 9: Diagrama de arquitectura

En el lado izquierdo se representan los clientes del sistema, mientras que en el lado derecho se representa el núcleo principal donde se despliega el servidor Wazuh. En primer lugar aparecen los agentes Wazuh en modalidad "roadwarrior", y a continuación una representación de un centro de datos en la nube de Microsoft Azure y un datacenter en modelo On-premise. Dentro de cada uno de estos centros de datos se ubican nuevos agentes de Wazuh, así como otras cargas de trabajo y servicios específicos. Como se aprecia, los agentes interactuarán con el *Wazuh Manager* a través del puerto 1515, tanto para registro de agentes como para tareas de la capa de control de nodos, y el *Wazuh Workers* (puerto 1514) para la recepción de los eventos de auditoria de seguridad.
También aparecen detallados los clientes que interactúan con la capa de administración, tanto de la plataforma Wazuh (*Kibana*) como del sistema de orquestación Kubernetes (*kubectl*).
Dentro del lado servidor, se ha diseñado la exposición de servicios para su consumo desde internet a través de sendos balanceadores de carga. Como solución de orquestación se ha empleado el servicio *Amazon EKS*, que proporciona la capa de control de Kubernetes desplegado en modalidad de servicio SaaS lo que permite simplificar enormemente las tareas de administración y operación de la solución. Para dar soporte a las cargas de trabajo dentro de Kubernetes se ha desplegado un enjambre de nodos de trabajo basados en el servicio de máquina virtual, *Amazon EC2*. El diseño general se implementa siguiendo el modelo de múltiples zonas de disponibilidad *(Multi-AZ*[22]*)*, que se detallará más adelante, y

22 https://aws.amazon.com/es/blogs/containers/amazon-eks-cluster-multi-zone-auto-scaling-groups/

que permite el despliegue de la solución sobre 3 datacenters (zonas *us-east1a, us-east1b y us-east1c*) ofreciendo un diseño de gran resiliencia. La arquitectura de red subyacente se ha implementado con la solución Amazon VPC, con un despliegue y segmentación en varias zonas en base a este modelo de segmentación en múltiples datacenters. A continuación se detallan los componentes empleados en la arquitectura de manera individual.

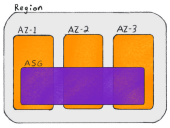
5.2 Componentes

{book5}

Como se puede ver en el diagrama general de arquitectura, para el diseño de esta solución se ha primado el uso de soluciones según el modelo SaaS que ofrece el proveedor, frente a otras alternativas en modelo IaaS que también habrían sido posibles. El motivo de esta decisión es que se ha optado por implementar una arquitectura donde se minimice el coste de mantenimiento y operación de los componentes, capitalizando el modelo de costes hacia un modelo OPEX[23] (Operational expenditures) y siguiendo la premisa del modelo de pago por servicios.
Se ha implementado un diseño resiliente, con capacidad de escalado horizontal y alta disponibilidad a través del sistema de datacenter, regiones y zonas de disponibilidad del fabricante Amazon AWS y siguiendo los modelos y diseños recomendados para cada una de las soluciones en base a su marco de buenas prácticas en arquitectura WELL-Architected Framework del fabricante.

{url12}

Los recursos de informática en la nube de Amazon se encuentran alojados en distintas ubicaciones por todo el mundo. Dichas ubicaciones se componen de regiones, zonas de disponibilidad y zonas locales. Cada región de AWS es un área geográfica independiente y cada región AWS contiene varias ubicaciones aisladas conocidas como zonas de disponibilidad o *multi-AZ*. En base a esto, si se distribuyen los equipos de cómputo y servicios en distintas zonas de disponibilidad dentro una misma región (centros de datos separados entre si por al menos 30Km) se implementa una solución capaz de persistir en funcionamiento pese a un fallo total de uno o varios de estos centros de datos (en función del número de zonas donde estén desplegados los servicios).
Para este proyecto, se ha elegido como región principal para el despliegue EEUU Este (Norte de Virginia), en adelante identificada como ***us-east-1*** y como se verá en detalle más adelante implementan todos los servicios en 3 zonas de disponibilidad, identificadas como *us-east-1a, us-east-1b y us-east-1c.* Se han definido estas 3 zonas y una sola región, porque consideramos 3 réplicas de los servicios en distintos centros de datos como suficiente garante para la continuidad del servicio en caso de un fallo. Como se verá en la fase de implementación, este diseño de arquitectura podría desplegarse en cualquier otra región (o en múltiples si se considerase y fuese necesario desplegar un entorno de mayor escala) de las 20 que existen actualmente repartidas por todo el mundo, o aumentar/disminuir/cambiar las zonas de disponibilidad simplemente realizando pequeños ajustes en la fase de implementación.



5.2.1 Red

Como componentes de la capa de red de la arquitectura propuesta se han empleado los siguientes servicios del cloud que se detallan a continuación:

5.2.1.1 VPC

{url11}

Para la capa de red que da soporte a toda la arquitectura, se ha empleado el servicio Amazon VPC. Este servicio permite crear una red definida por software (SDN) virtual sobre la que se despliegan los componentes y resto de servicios de la solución. Sobre este sistema se define una red privada identificada como VPC *us-east-1* y con direccionamiento 192.168.0.0/20.
Como se ha introducido al inicio de este capítulo, se ha diseñado un modelo sobre 3 datacenter donde se desplegarán servicios a través de zonas de disponibilidad, por lo que, para dar soporte al direccionamiento, se reparte la red principal a través de subnetting resultando 3 rangos: 192.168.0.0/22 para la zona de disponibilidad *us-east-1a*, 192.168.4.0/22 para la zona de disponibilidad *us-east-1b* y finalmente 192.168.8.0/22 para la zona de disponibilidad *us-east-1c*.
Sobre estos direccionamientos y zonas de red con carácter interno, se desplegarán los nodos que gestionarán la carga de trabajo de Kubernetes, los propios interfaces de red que se han empleados para proveer de conectividad para la comunicación individual de cada contenedor así como para la función de gateway o punto de conexión dentro de cada zona con el plano de gestión de cluster EKS y los balanceadores de tráfico.
El direccionamiento para esta arquitectura permite un total de 4096 direcciones ip asignables a equipos, reservando 1022 para cada zona de disponibilidad, lo cual hemos considerado óptimo para la solución particular sin generar gasto extra de direccionamiento. Es importante destacar en este punto, que al asignarse direcciones ip a nivel de contenedor el número de los mismos puede crecer rápidamente, por lo que en caso de que en el cluster de Kubernetes se desplegasen otras cargas de trabajo diferentes del sistema Wazuh, este número podría ser bajo y este cálculo necesitaría reajustarse.
Adicionalmente, al desplegar el servicio de cluster de Kubernetes EKS, automáticamente el proveedor despliega una red para dar soporte a los componentes internos de la capa de gestión y administración del cluster. Por defecto, el rango de

23 https://es.wikipedia.org/wiki/Opex

red empleado es 10.100.0.0./16 y la gestión de la misma recae sobre el propio proveedor al ser un servicio en modalidad SaaS y, por lo tanto, transparente para nosotros.

Tabla 2: Regiones, zonas de disponibilidad y direccionamiento

Región	Zona AZ	Red	Servicio
us-east-1	us-east-1a us-east-1b us-east-1c	10.100.0.0/16	Red de gestión del Cluster EKS
us-east-1	us-east-1a	192.168.0.0/22	Subred de nodos de trabajo de la zona us-east-1a
	us-east-1b	192.168.4.0/22	Subred de nodos de trabajo de la zona us-east-1b
	us-east-1c	192.168.8.0/22	Subred de nodos de trabajo de la zona us-east-1c

5.2.1.2 ELB

Como se ha visto en el apartado anterior, todas las redes empleadas en la arquitectura se definen como redes privadas lo que garantiza una mayor protección del entorno mejorando enormemente la seguridad del mismo al reducir el perímetro de exposición. Por este motivo, los servicios desplegados sobre el cluster de Kubernetes son únicamente accesibles a nivel interno dentro de la red general del VPC (192.168.0.0/20). Para aquellos servicios que se necesiten exponer públicamente desde internet, ha sido necesario implementar en el diseño una solución que nos dotase de esta capacidad. Para ello, se ha empleado el servicio de balanceador de carga *Amazon ELB* (acrónimo de *Elastic Load Balancing*) [24].
Dentro del proveedor cloud existen varios tipos distintos de balanceadores de carga que se pueden emplear para esta solución, donde la diferencia principal está en la capa de red en la que trabajan. En esta arquitectura se han empleado balanceadores ELB, que trabajan en capa 4 de red, que es la necesaria para exponer los servicios de Wazuh manager y los nodos de Wazuh managers y Wazuh workers (TCP/UDP). En el caso particular de los servicios del panel web de administración de W*azuh* y *Elastic* (kibana) y entorno de administración del API Kubernetes se podría considerar más recomendado el empleo de un balanceador *ALB* (*Application Load Balancing*) pero se ha decidido emplear igualmente ELB ya que también es válido como solución y permite simplificar y homogeneizar el diseño y la orquestación.
El balanceador de carga, además de emplearse para exponer públicamente un servicio, tiene otra funcionalidad muy importante en el diseño de esta arquitectura. Al ser una arquitectura escalable, donde los contenedores que dan servicios a ciertas funcionalidades pueden crecer dinámicamente o simplemente moverse a una nueva zona de disponibilidad, en caso de fallo, este servicio se encarga también de mantener, gestionar y chequear la salud de las aplicaciones (contenedores de un pod) a las que apunta, controla su número, distribución y direccionamiento interno.
Como se verá en la fase de implementación, el despliegue y creación de los balanceadores de carga se integra directamente dentro del control de cluster de EKS, por lo que su gestión se hará automáticamente desde los ficheros de definición del API de Kubernetes, a través de uso del driver "*service.beta.kubernetes.io/aws-load-balancer-backend-protocol*", y por tanto dentro de la misma orquestación del despliegue.

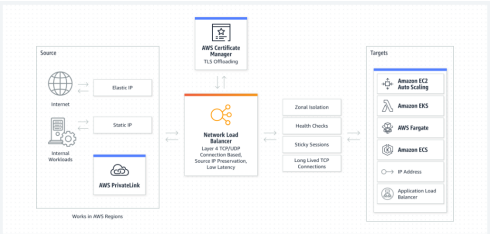


Figura 10: Balanceador de carga

Tabla 3: Servicios Wazuh publicados a través de un balanceador

Región	Puerto	Protocolo	Servicio Expuesto
us-east-1	443	https	Gestión del Cluster EKS (Kubernetes API)
us-east-1	443	https	Panel web de administración de wazuh y elastic (kibana)
us-east-1	1515,1516	tcp	Servicio Wazuh-Manager para el registro y control de los agentes
us-east-1	1514	tcp/udp	Servicio de nodos Wazuh-Worker para la conexión de los agentes

24 https://aws.amazon.com/es/elasticloadbalancing/

5.2.2 Almacenamiento

5.2.2.1 EBS

En un sistema de cargas de trabajo con contenedores, generalmente existen dos tipos distintos de almacenamiento: el almacenamiento volátil y el almacenamiento persistente. Por un lado, en el caso de los contenedores, los datos de los mismos sólo existen asociados su mismo ciclo de vida. De este modo, todos los datos que se generan en los contenedores desaparecen al eliminar el mismo. Este almacenamiento para los contenedores se consume del espacio de almacenamiento del nodo de trabajo donde se despliega.
Por otro lado, para el almacenamiento persistente, se emplean volúmenes de datos que se mapean asociándolos al contenedor donde se necesite que persistan. Para el despliegue del sistema Wazuh se necesitan dos volúmenes de datos persistentes. Uno para la persistencia de datos de Wazuh manager y otro para el sistema Elastic.
Para ello, como se desprende del mapa de la arquitectura general, se ha empleado el servicio *Amazon EBS* (acrónimo de *Elastic Block Store*). *EBS* es un servicio de volumen de almacenamiento en formato bloque (SAN), escalable y de alto rendimiento que se integra perfectamente con los nodos de computo EC2 y con el servicio de cluster EKS pudiendo ser directamente gestionados y desplegados de manera integrada desde la orquestación del despliegue de Kubernetes.
Como se verá en el detalle de implementación, el servicio EBS se integra directamente en los objetos del API de Kubernetes dentro del cluster de EKS, a través del driver de aprovisionamiento "*kubernetes.io/aws-ebs*", de modo que los volúmenes de almacenamiento se aprovisionarán automáticamente desde los ficheros de la orquestación.
Adicionalmente, gracias a emplear este servicio, se dispone de todas las ventajas y capacidades que ofrece el servicio de EBS en la nube de Amazaon, destacando el sistema de instantáneas y replicación automática de copias entre regiones así como un crecimiento de datos prácticamente ilimitado.

5.2.3 Carga de trabajo

5.2.3.1 EKS

{url10}

Amazon EKS (Acrónimos de Elastic Kubernetes Service) es el servicio de contenedores y orquestación de Kubernetes que ofrece el proveedor Amazon para ser empleado en la nube e integrado de manera nativa con el resto de servicios. Este servicio, cuenta con certificación de conformidad oficial de Kubernetes lo que garantiza compatibilidad completa con cualquier despliegue y migración de cargas desde otros sistemas.
El servicio Amazon EKS administra automáticamente la disponibilidad y escalabilidad de los nodos del plano de control de Kubernetes, que son responsables de iniciar, detener, ubicar, etc. contenedores en las máquinas virtuales que los forman y almacenar datos locales, entre otras tareas. Este servicio se despliega a través de las distintas regiones y zonas de disponibilidad dentro de AWS, lo que permite la implementación de manera prácticamente inmediata de una solución robusta para orquestación de contenedores. Además, EKS se encarga de toda la gestión de la capa de control de Kubernetes por lo que no es necesario preocuparse del escalado y dimensionamiento del servicio, incluyendo también la sustitución/reubicación automática de nodos o zonas con problemas, lo que garantiza gran capacidad de adaptación a fallos.
La parte principal de esta solución se basa en un conjunto de **nodos maestro** que, como hemos visto en esta solución, se encuentran completamente gestionados por el proveedor, y donde se ubican todos los componentes necesarios para la gestión y operación del cluster. Entre sus componente se encuentran:

- **kube-apiserver**: Es el componente que expone el API de Kubernetes. Es, por lo tanto, el punto de entrada para todos los comandos REST utilizados para el control y gestión del clúster.
- **kube-controller-manager**: Es el interfaz de administración del cluster que se encarga, entre otras funciones de la gestión de los posibles nodos maestros.
- **etcd**: Es un sistema de almacenamiento de los objetos compartidos en formato clave-valor, para la gestión del entorno Kubernetes.
- **scheduler**: Observa y gestiona la planificación de los nodos, servicios y contenedores, y se encarga de controlar su estado y despliegue.

EKS se despliega directamente sobre las zonas de red VPC y se integra con la solución de balanceadores del proveedor para crear el punto de entrada desde internet a los servicios desplegados en Kubernetes. Como ya hemos visto en el detalle de componentes previo, también se integra nativamente con el servicio de almacenamiento en bloques permitiendo a través de los volúmenes de Kubernetes dotar al sistema de almacenamiento de una solución de persistencia de datos.
Para completar la solución, adicionalmente EKS necesita incorporar **nodos de trabajo**, que son aquellas máquinas de cómputo donde se ejecutan los contenedores y que son controlados por el cluster de Kubernetes para la gestión de las cargas de trabajo. Para estos nodos se ha optado por la implementación con la solución de cómputo *Amazon EC2* que detallaremos más adelante.
El cluster EKS controla estos nodos de trabajo a través de la creación de un grupo de autoescalado, denominados "*Auto Scaling*". Este grupo, crea una asociación lógica de máquinas virtuales EC2 que forman los nodos de trabajo y permite gestionar de forma integrada la administración y escalado automático de los mismos. A través de este sistema, EKS

define una capacidad deseada de nodos sobre las distintas zonas de disponibilidad, y éstas se despliegan de manera automática hasta alcanzar la capacidad que se le ha establecido.
Dentro de EKS, se puede definir no solo un grupo de "*Auto Scaling*", sino que se pueden implementar todos los que sean necesarios e incluso combinar grupos de máquinas virtuales EC2 con distintas instancias.

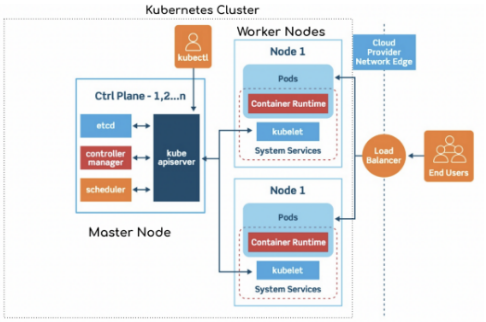


Figura 11: Arquitectura EKS

Se ha optado por esta solución, descartando otras posibles opciones disponibles en este proveedor, porque como hemos visto es una opción avalada y con soporte del propio fabricante Kubernetes. Además, es la solución que mejor encaja con el modelo y las premisas del proyecto, al ser el servicio de pago por uso, escalable y con la menor carga de administración y operativa disponible para esta funcionalidad.

5.2.3.2 EC2

Amazon EC2 (Acrónimo de *Elastic Compute Cloud*) es el sistema de despliegue de capacidad informática en nube sobre la infraestructura de Amazon y que se ha empleado para el aprovisionamiento de cómputo para los **nodos de trabajo del cluster** de Kubernetes.
Este servicio, al igual que el resto de componentes por los que se ha optado para la arquitectura de este diseño incluye capacidades de escalado automático y resiliencia.
Los servicios de EC2 se despliegan sobre red VPC privada que se ha descrito anteriormente y siguiendo el mismo modelo de despliegue sobre distintas zonas de disponibilidad que tambien hemos visto en componentes anteriores.
Dentro del conjunto de máquinas virtuales seleccionables, existen múltiples modelos de instancia, cada uno de ellos con unos requisitos y capacidades hardware específicas. De esta manera, existen instancias que incluyen GPU's de cómputo, procesadores de distintas arquitecturas (x64, Arm, etc), optimización para consumo extremo de CPU y memoria e incluso instancias de tipo físico (baremetal) dedicadas. Dentro de los tipos de máquina empleados en nuestro diseño para el despliegue de los nodos de trabajo para el cluster EKS, se han seleccionado instancias del tipo "*t3*"[25]. Este es un modelo de instancia de propósito general basado en procesadores Intel Xeon Platinum 8000 (Skylake-SP o Cascade Lake), de coste muy contenido y recomendada por el proveedor para soluciones que incluyen el despliegue de nodos de trabajo de EKS que van a contener aplicaciones de contenedores para aplicativos de propósito general. Por este motivo, se ha considerado como el modelo de instancia más adecuado para el cumplimiento de las premisas de esta funcionalidad.
Dentro de cada nodo de trabajo de EC2 y sobre cada máquina virtual EC2, el cluster EKS despliega automáticamente una serie de componentes, en modo de contenedores sobre el nodo, que forman parte del núcleo de servicios necesarios para dar soporte a la funcionalidad del nodo dentro del cluster. Estos componentes son:

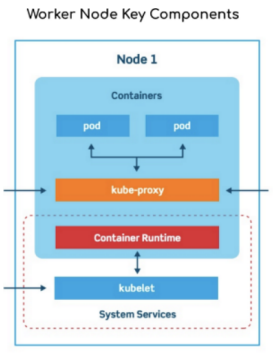


Figura 12: Componentes del nodo de trabajo

- **kube-proxy**: Este componente se encarga de la gestión de red individual de cada nodo de trabajo.
- **Container Runtime**: Este servicio se encarga de la gestión del backend de contenedores (Docker) de Kubernetes.
- **kubelet** : La función de k*ubelet* es ser el agente de control dentro de cada nodo que se encarge de la capa de administración y gestión del mismo dentro de ecosistema del cluster Kubernetes.

25https://aws.amazon.com/es/ec2/instance-types/t3/

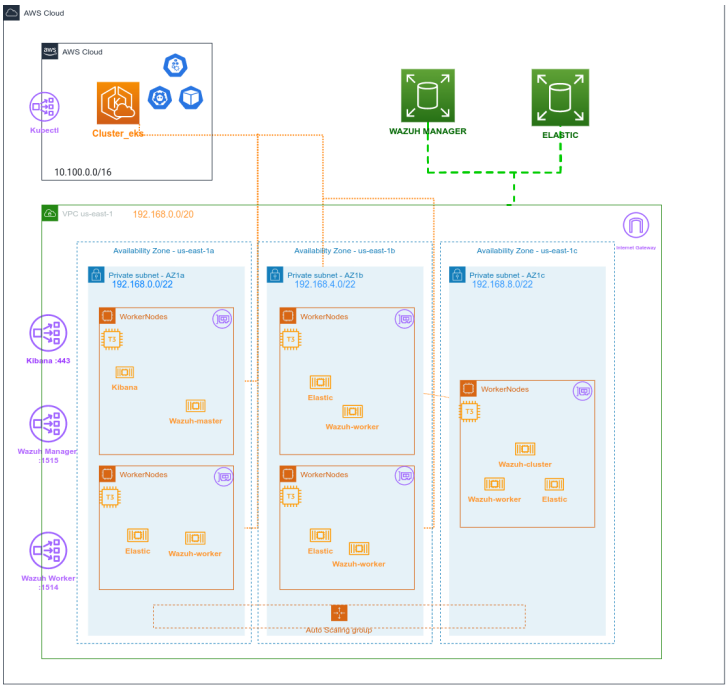


Figura 13: Detalle de componentes de la capa servidor

6 Implementación y despliegue

A continuación se detallarán los principales pasos para el despliegue de la arquitectura y el SIEM Wazuh. Se ha distribuido este punto en dos grandes bloques funcionales. En primer lugar, se implementarán los pasos para el despliegue de los componentes de la arquitectura sobre el entorno cloud del proveedor, para en un segundo punto detallar los pasos de implementación de los componentes del SIEM Wazuh sobre el cluster implementado en el paso anterior.

6.1 Implementación de la arquitectura en el cloud de Amazon

Para iniciar el despliegue de cualquier servicio sobre el cloud de Amazon se necesita disponer de una cuenta para este proposito. Para este particular, partimos de la premisa de la existencia de una cuenta previamente configurada, con un rol de administración y facturación activada. Se adjunta en el siguiente enlace la información detallada para la creación de una cuenta en caso de que fuese necesario.
El primer paso será la selección de la región de trabajo y se procederá a la **implementación de la red VPC** y las subredes para cada una de las multizonas detalladas en la arquitectura. Para ello:

1. Se accede al panel de la consola de Amazon a través del enlace https://console.aws.amazon.com/ .
2. Se selecciona la región EEUU Este (Norte de Virgina), us-east-1 como región por defecto para comentar a realizar el despliegue.
3. Se selecciona desde el panel de servicios VPC. Pulsamos directamente sobre la opción "*Crear VPC*" y directamente introducimos un nombre identificativo, "*RedEks*" en nuestro caso, y el bloque CIDR 192.168.0.0/20 (como se detalló previamente) para, en el siguiente paso, crear directamente la red que va a dar soporte a toda la infraestructura.
4. A continuación, se crean 3 subredes en las distintas zonas de disponibilidad donde se pretenden desplegar los servicios de infraestructura. Para ello, a través de la pestaña "*Subredes*", dentro del mismo servicio VPC, se selecciona "*Crear Subred*". Desde este punto se nos presenta un panel donde, en primera instancia, seleccionamos la red VPC principal, "*RedEks*", y desde aquí se van detallando las nuevas subredes a crear con su nombre identificativo, su propio bloque de direccionamiento CIDR, y se indica para cada una de ellas la correspondiente zona de disponibilidad. Se definen 3 zonas de disponibilidad *us-east-1a*, *us-east-1b* y *us-east-1c* sobre la región *us-east-1*.
5. Dentro de los nodos de trabajo y contenedores que se van a desplegar sobre estas subredes, se necesita que dispongan de salida a internet, principalmente para acceso a los registros de contenedores públicos que se emplearan. Para esta funcionalidad, es necesario implementar un componente adicional en la red VPC conocido como "*Internet Gateway*" o *IG*. Para ello, a través de la misma pestaña "*internet gateway*", seleccionamos crear un nuevo IG, y posteriormente se le asocia a cada una de las subredes que queremos dotar de salida a internet.

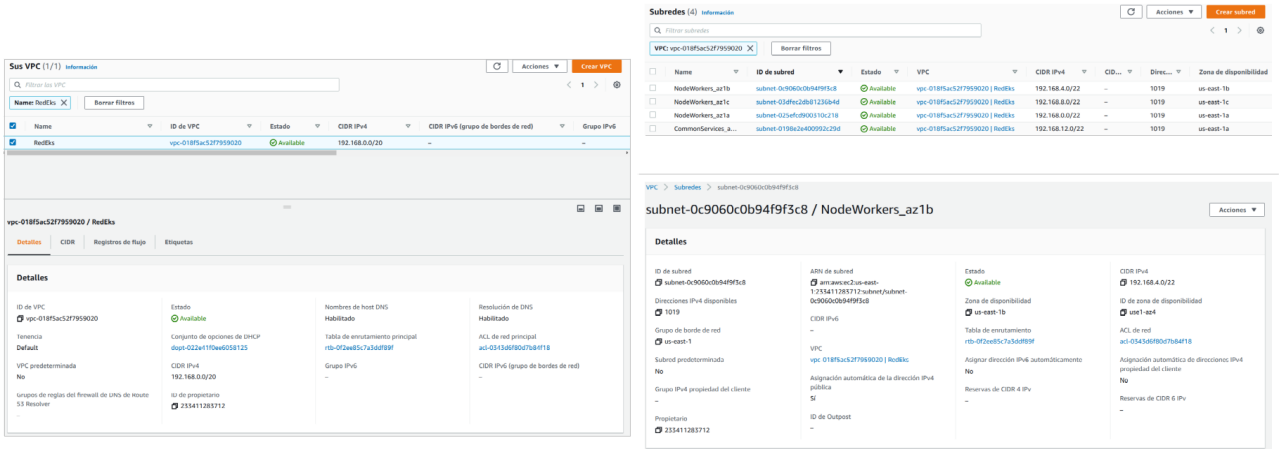


Figura 14: Detalle de red VPC

Una vez implementado el sistema de red subyacente, se procede al despliegue del **cluster de Kubernetes sobre EKS**. Para ello, se realizan los siguiente pasos:

1. Dentro del panel de servicios, se selecciona el servicio "*Elastic Kubernetes Service*". Desde la pantalla principal se marca, "*crear un nuevo cluster*".
2. Se introduce el nombre identificativo del cluster, "*cluster_eks*" en nuestro caso, y se selecciona la versión del software Kubernetes que se desea usar. Por defecto se optará siempre por las versiones más recientes.

3. Es necesario crear un rol de seguridad, que haremos siguiendo la guía de configuración del proveedor. Este rol, es un perfil de seguridad que se empleará para aplicar a las máquinas virtuales de los nodos de trabajo del cluster y a través del mismo, controlar y administrar el nodo desde el cluster.
4. Se selecciona la red VPC y las subredes donde se va a desplegar el plano de control de la conexión con la red de los nodos de trabajo. Se selecciona el VPC "*RedEks*" y las 3 subredes definidas, una por cada zona de disponibilidad.
5. Se define un grupo de seguridad, que será el sistema de ACL, que controla el acceso por red a los nodos de trabajo.
6. Se define el acceso a la capa de gestión del API de Kubernetes. Se selecciona acceso privado, para que se habilite el acceso de los nodos en las subredes internas al API y además se selecciona acceso público (limitando el acceso por IP de origen). Este acceso público se empleará para la gestión y la administración del cluster de Kubernetes de manera remota a través del cliente *kubectl*.
7. Se seleccionan las versiones de software de los distintos addons de control que se van a desplegar sobre el cluster de Kubernetes para la gestión de EKS. Por defecto, estos son *CNI*, como driver de gestión de la red de contenedores, *CoreDNS* y *kube-proxy*. Se seleccionan las últimas versiones disponibles de los sistemas.
8. Tras la última confirmación y pasados unos minutos se despliega completamente la solución de cluster de Kubernetes EKS.

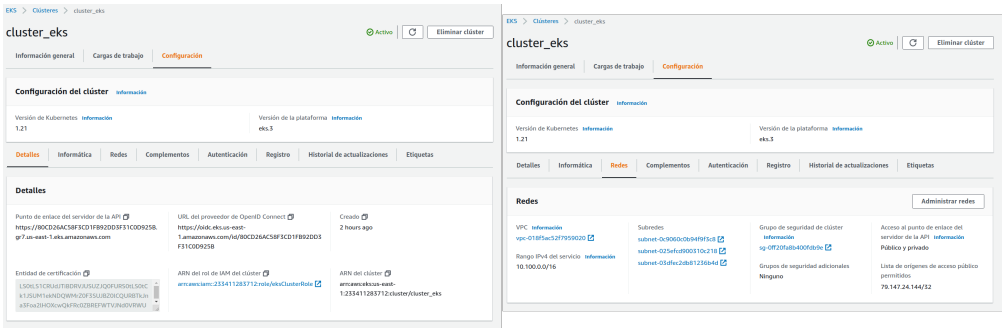


Figura 15: Cluster EKS

Una vez desplegado el cluster, el siguiente paso para completar la infraestructura de la solución, consiste en implementar el **conjunto de nodos de trabajo** y asociarlos al cluster creado en el paso anterior. Para ello, se procede creando el conjunto de nodos del siguiente modo:

1. Una vez se ha creado en cluster, se accede desde el panel del servicio EKS al detalle del cluster creado. A través de la pestaña configuración, se selecciona la opción "*Informática*", "*grupo de nodos*" y "*agregar nuevos nodos*". Desde este apartado se pueden añadir y eliminar los conjuntos agregados de nodos de trabajo que se desee desplegar sobre este cluster. Recordar en este punto, que para este despliegue se van a implementar hasta 3 nodos de trabajo, con instancias del tipo *t3.large* en base a un grupo de autoescalado con tamaño mínimo de 2 y máximo de 3 nodos.
2. Al seleccionar "*Agregar nodos*", se muestra un panel donde se introduce primeramente un nombre identificativo para este grupo de nodos. Se selecciona "*Workers*" como identificativo.
3. Es necesario seleccionar el role de control de acceso a los nodos que definimos previamente.
4. Tras confirmar, se accede al panel donde se definen las opciones de configuración hardware por cada nodo. Entre estas opciones, se define el tipo de instancias y el tamaño de disco de los nodos de trabajo. Como se había adelantado, se definen instancias "*t3.large*" (2 VCPU y 8GB de RAM) con 50 Gb de disco por nodo.
5. Se definen las opciones del grupo de escalado para este conjunto de instancias. En este particular se implementan 3 nodos como máximo, 2 como mínimo y 2 como tamaño deseado.
6. Tras confirmar, se accede al panel donde se selecciona la red VPC y las subredes donde se van a desplegar los nodos de trabajo. Se selecciona el VPC "*RedEks*" y las 3 subredes previamente definidas, una por cada zona de disponibilidad.
7. Se confirma la configuración anterior y en unos minutos, se despliega el conjunto de nodos y se conectan al cluster, listos para empezar a desplegar cargas de trabajo.

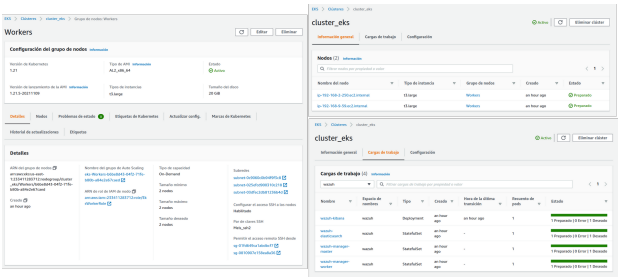


Figura 16: Nodos de trabajo EKS

6.2 Implementación de cliente de EKS

Una vez se ha desplegado y esta disponible la infraestructura necesaria para el despliegue de aplicaciones sobre el cluster implementado en la nube de Amazon, el siguiente hito consiste en configurar el cliente de gestión *kubectl* para poder proceder después al despliegue final de la aplicación Wazuh.
Para los siguientes pasos, se supone como equipo cliente desde donde se realiza la gestión y orquestación del entorno un equipo de escritorio Linux Ubuntu 20.04 LTS. Destacar que la adaptación de los comandos que vamos a presentar en esta implementación sobre otro sistema operativo es inmediata, simplemente será necesario adaptar la instalación de los ejecutables a la plataforma particular que se desee emplear.
En primer lugar, es necesario la instalación el software base del cliente de gestión *kubectl* adaptado para el uso con el cluster de EKS. En el siguiente anexo se detalla el proceso de instalación para la plataforma Linux Ubuntu 20.04 LTS.
Es segundo lugar, para trabajar con el servicio EKS también es necesaria la instalación del Framework de herramientas de gestión del proveedor Amazon, *AWS cli*, para poder interactuar con los servicios de EKS. En el siguiente anexo se detallan los pasos.
En tercer lugar, se necesita disponer de unas credenciales de API para conexión al portal de administración de Amazon AWS con un usuario que posea un rol con permisos suficientes para la administración de cluster EKS. Para ello, de manera resumida, los pasos serían los siguientes:

1. Iniciar sesión en la consola de AWS y acceder al sistema de gestión de identidades a través del enlace: https://console.aws.amazon.com/iamv2/home#/home
2. Seleccionar el usuario que se va a emplear para autenticarse en el sistema de API y seleccionar "*credenciales de seguridad*". A continuación se creará una nueva clave a través del botón "*Crear una clave de acceso*". Obtendremos un ID de clave[26] y una clave, que se empleará más adelante como token de autenticación.
3. Asociar a este mismo usuario la capacidad de gestionar el servicio EKS de Amazon AWS (este paso no sería necesario si estamos empleando un usuario con un perfil de Administrador). Para ello, a través de la pestaña permisos, "Añadir permisos" seleccionamos manualmente el rol predefinido por el fabricante identificado como "*AmazonEKSClusterPolicy*".

Llegados a este punto, se supone *kubectl* y el *AWS cli* de Amazon complemente instalado y listo para ser usado así como una credencial de usuario con permisos suficientes para administrar el servicio EKS.
Con estas premisas, en el siguiente paso se realizan las configuraciones y adaptaciones específicas para la conexión con la solución EKS y poder comenzar a gestionar el cluster. Para ello, en primer lugar se revisa el correcto funcionamiento del *AWS Cli*, invocando el comando que se muestra a continuación desde la consola:

```
$ aws --version
```

Si nos devuelve la versión de software de *AWS cli*, es señal que de está correctamente instalado, por lo que se procederá con el siguiente comando para inicializar la autenticación del cli con el panel del proveedor:

```
$ aws sts get-caller-identity
```

A continuación, nos solicitará las credenciales para proceder a autenticarnos. Se introducen en este paso las credenciales previamente generadas.
Si todo es correcto, se procede a la configuración del cliente *kubectl*, para trabajar con el cluster EKS que se ha creado previamente. Para ello, a través del siguiente comando, se introduce el nombre del cluster que queremos gestionar, mediante el argumento "*--name*" así como la ubicación de la región a través de "*--region*":

```
$ aws eks --region us-east-1 update-kubeconfig --name cluster_eks
```

Finalmente, si todo se ha llevado a cabo de forma correcta hasta este punto, ya se dispone de la configuración completa de la herramienta *kubectl* desde el equipo local, lista para la gestión del cluster EKS identificado como *cluster_eks* y desplegado en el entorno del cloud.

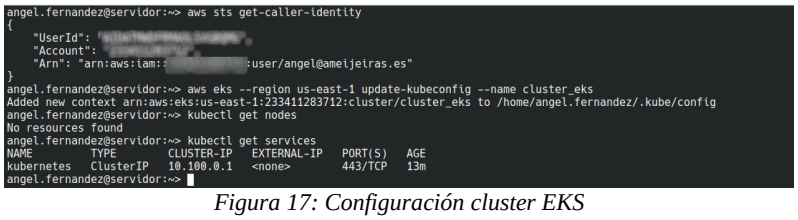


Figura 17: Configuración cluster EKS

A continuación, se puede proceder a una verificación rápida del estado del cluster, los nodos conectados y los pods desplegados a través de los siguiente comandos:

```
# Listar todos los servicios dentro de todos los namespaces de k8s
$ kubectl get svc --all-namespaces
# Listar todos los pods dentro de todos los namespaces de k8s
$ kubectl get pods –all-namespaces
```

26 Es necesario proteger de manera adecuada esta clave ya que exponerla supone un importante riesgo de seguridad

```
# Listar todos los nodos de trabajo del cluster k8s
$ kubectl get nodes
```

```
angel.fernandez@servidor:~> kubectl get nodes
NAME                          STATUS   ROLES    AGE    VERSION
ip-192-168-10-113.ec2.internal   Ready    <none>   4m3s   v1.21.5-eks-bc4871b
ip-192-168-2-106.ec2.internal    Ready    <none>   4m3s   v1.21.5-eks-bc4871b
ip-192-168-5-154.ec2.internal    Ready    <none>   4m3s   v1.21.5-eks-bc4871b
angel.fernandez@servidor:~>
```

Figura 18: Verificación de nodos worker

Una vez se ha llegado a esta etapa, el sistema está listo para el despliegue de cualquier solución que sea compatible con Kubernetes.

6.3 Implementación de Wazuh sobre cluster

Llegado a este punto, se procederá a la implementación del SIEM Wazuh sobre la arquitectura desplegada. Se supone, para los siguientes pasos, que ya se han implementado las configuraciones descritas en los apartador 6.1 y 6.2. De este modo, para realizar el despliegue de la solución se ejecutan los pasos que se indican a continuación:
En primer lugar se genera un directorio de trabajo, donde se va a proceder a descargar los ficheros de configuración de todos los objetos Kubernetes necesarios para implementar el sistema SIEM. A través de un cliente *git*, descargamos el repositorio donde se encuentran todos los ficheros necesarios para el despliegue. En caso de que no se disponga del cliente *git* instalado en el equipo cliente, también se puede descargar directamente el código manualmente desde un navegador accediendo al repositorio.

```
$ git clone https://github.com/ameijeiras/wazuh-uoc.git
```

A continuación se accede al directorio base desde el que se va a trabajar:

```
$ cd wazuh-uoc/wazuh/
```

Desde este punto, se ejecutan los siguientes comandos:

```
./certs/kibana_http/generate_certs.sh
./certs/odfe_cluster/generate_certs.sh
```

A través de estos comandos, se generan una serie de certificados auto-firmados temporales que van a ser usados para cifrar la comunicación sobre los servicios de gestión expuestos por el sistema. Es posible, antes de su ejecución, ajustar parámetros específicos de los mismos, como caducidad, *Common Names* y alias, unidades organizativas, etc. Finalmente, solo queda, a través del siguiente comando, lanzar el despliegue completo de todos los componentes necesarios para la ejecución de la solución:

```
$ kubectl apply -k .
```

A continuación se incluye un conjunto de capturas de pantalla, donde se documenta el proceso de despliegue así como los resultados de las distintas etapas:

```
angel.fernandez@servidor:~/codigo/git/wazuh> ls
base  certs  elastic_stack  kustomization.yml  secrets  wazuh_managers
angel.fernandez@servidor:~/codigo/git/wazuh> kubectl apply -k .
namespace/wazuh created
storageclass.storage.k8s.io/wazuh-storage created
configmap/elastic-odfe-conf-5b9hg68586 created
configmap/wazuh-conf-5bc7t8k9h7 created
secret/elastic-cred created
secret/kibana-certs-df9g92tt9h created
secret/odfe-ssl-certs-6b875d8b85 created
secret/wazuh-api-cred created
secret/wazuh-authd-pass created
secret/wazuh-cluster-key created
service/elasticsearch created
service/kibana created
service/wazuh created
service/wazuh-cluster created
service/wazuh-elasticsearch created
service/wazuh-workers created
deployment.apps/wazuh-kibana created
statefulset.apps/wazuh-elasticsearch created
statefulset.apps/wazuh-manager-master created
statefulset.apps/wazuh-manager-worker created
angel.fernandez@servidor:~/codigo/git/wazuh>
```

Figura 19: Despliegue

```
angel.fernandez@servidor:~/codigo/git/wazuh> kubectl get pods -n wazuh
NAME                        READY   STATUS    RESTARTS   AGE
wazuh-elasticsearch-0       1/1     Running   0          2m53s
wazuh-elasticsearch-1       1/1     Running   0          114s
wazuh-elasticsearch-2       1/1     Running   0          79s
wazuh-kibana-5cd658b78-nbf56   1/1     Running   0          2m54s
wazuh-manager-master-0      1/1     Running   0          2m53s
wazuh-manager-worker-0      1/1     Running   0          2m53s
wazuh-manager-worker-1      1/1     Running   0          2m53s
angel.fernandez@servidor:~/codigo/git/wazuh>
```

Figura 20: pods del sistema Wazuh

```
angel.fernandez@servidor:~/codigo/git/wazuh> kubectl get services -n wazuh
NAME                TYPE           CLUSTER-IP       EXTERNAL-IP                                                               PORT(S)                       AGE
elasticsearch       LoadBalancer   10.100.103.210   a4cb9e08297e44cd9b38e2233aafa64c-173568756.us-east-1.elb.amazonaws.com    9200:31601/TCP                21s
kibana              LoadBalancer   10.100.168.145   ac8a7fdd3148e4cea84794cb609db45f-2077561540.us-east-1.elb.amazonaws.com   443:30138/TCP                 20s
wazuh               LoadBalancer   10.100.72.219    a76a8a6dc0fc14b519d1d393ee4014de-183041565.us-east-1.elb.amazonaws.com    1515:31866/TCP,55000:30928/TCP   20s
wazuh-cluster       ClusterIP      None             <none>                                                                    1516/TCP                      20s
wazuh-elasticsearch ClusterIP      None             <none>                                                                    9300/TCP                      19s
wazuh-workers       LoadBalancer   10.100.244.204   aee34952a3a8440988d4156bff505aec-1026548912.us-east-1.elb.amazonaws.com   1514:30420/TCP                19s
angel.fernandez@servidor:~/codigo/git/wazuh>
```

Figura 21: Servicio Wazuh y balanceadores de entrada

6.3.1 Ajustes del despliegue y configuraciones

Dentro del directorio raíz de despliegue se encuentran todos los ficheros de objetos necesarios para la orquestación del

sistema Wazuh. A través de los mismos, se pueden establecer un gran número de ajustes, tanto de la configuración específica de los componentes de Wazuh como de la infraestructura; destacando en este apartado los recursos hardware (cpu, memoria, disco) y el número de instancias para cada uno de los subsistemas. Se detallan a continuación los directorios y los ficheros más relevantes:

- Fichero **kustomize.yaml:** Es el fichero principal desde donde se realiza el inicio del proceso de orquestado. Desde aquí se definen los secretos, ficheros de configuración base de wazuh que se copiarán a los pods y contenedores de wazuh, y la definición completa de los recursos. Dentro de estos, se instancian nuevos ficheros específicos donde se define el almacenamiento, los servicios de Kubernetes, los pods, etc.
- Directorio **certs:** Dentro de este directorio se encuentran los ficheros que definen los certificados que se emplearán para cifrar la comunicación entre todos los componentes internos del SIEM, así como de los clientes.
- Directorio **secrets:** Dentro de este directorio se encuentran todos los ficheros donde se definen los secretos y, por lo tanto, se pueden emplear para cambiar los ajustes iniciales de contraseñas y otros secretos del sistema wazuh.
- Directorio **elastik_stack:** Dentro de este directorio se encuentran los ficheros de configuración específicos de la solución *Elastiksearch* y *Kibana*. Además, se incluyen en el mismo, los ficheros de declaración de los contenedores, pods y servicios de este subsistema.
- Directorio **wazuh_managers:** Dentro de este directorio se encuentran los ficheros de configuración específicos de los componentes *wazuh*, tanto para la parte de nodos manager como para la de workers. Además, se incluyen en el mismo, los ficheros de declaración de los contenedores, pods y servicios del sistema wazuh.

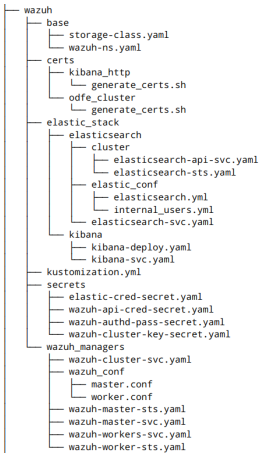


Figura 22: Ficheros de configuración

Destacar que los ficheros anteriores están comentados y practicamente todos los parámetros de ajuste son auto explicativos. No se va a realizar un detalle minucioso de los mismos ya que no es el objetivo del proyecto. Para más detalle, remitimos a la documentación del API Kubernetes aquí.
Una vez realizado cualquier ajuste sobre alguno de los objetos de los ficheros del despliegue, simplemente se trasladarían al entorno a través del comando:

```
$ kubectl apply -k .
```

6.3.2 Acceso al sistema

El acceso al sistema, se realiza a través de los servicios expuestos por medio de los balanceadores de carga que se detallan en la arquitectura. Para facilitar la configuración de los agentes y acceso al sistema, se han mapeado a un registro CNAME (kibana.*wazuh.ameijeiras.es, manager.wazuh.ameijeiras.es y wazuh.ameijeiras.es*), sobre un servidor DNS, los distintos balanceadores de los servicios expuestos en el sistema. Este paso no es imprescindible para el funcionamiento y, por ello, no entra en los objetivos de este proyecto el detalle del mismo. El acceso se podría realizar directamente a través de las direcciones que generan los balanceadores de carga, aunque sí es recomendable y una buena práctica, ya que los nombres de los balanceadores del proveedor pueden cambiar y, por ello, se debería tener en consideración ante la necesidad de reconfiguración de agentes que supondría. En las siguientes tablas se indican los distintos accesos al sistema:

Tabla 4: Datos acceso al sistema

Acceso web de gestión Wazuh (Kibana):	
Url	https://kibana.wazuh.ameijeiras.es (ac8a7fdd3148e4cea84794cb609db45f-2077561540.us-east-1.elb.amazonaws.com)
Usuario	admin
Contraseña	SecretPassword

Tabla 5: Datos acceso al sistema Wazuh Manager

Acceso a Wazuh Manager	
Dirección	manager.wazuh.ameijeiras.es (a4cb9e08297e44cd9b38e2233aafa64c-173568756.us-east-1.elb.amazonaws.com)
Puerto	1515

Tabla 6: Datos acceso al sistema Wazuh Worker

Acceso a Wazuh Worker (Configuración de los agentes)	
Dirección	wazuh.ameijeiras.es (aee34952a3a8440988d4156bff505aec-1026548912.us-east-1.elb.amazonaws.com.)
Puerto	1514

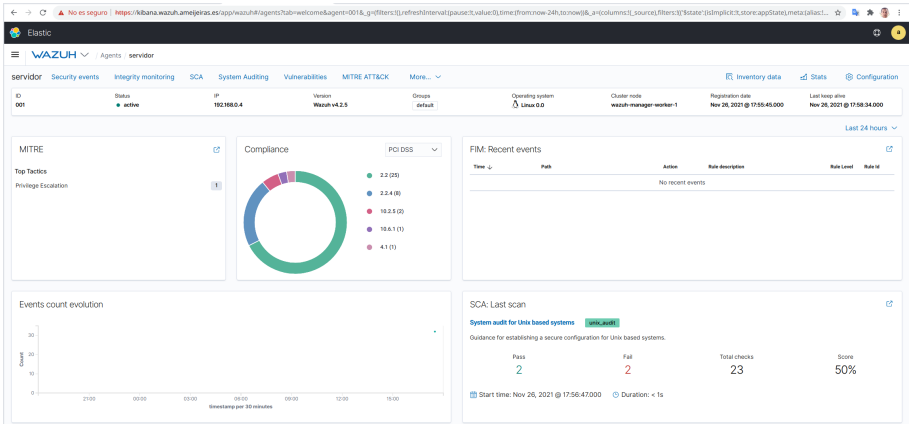


Figura 23: Interfaz de administración Wazuh

7 Casos de uso

Una vez se ha implementado el sistema Wazuh sobre la arquitectura diseñada empleando la nube de Amazon, se procederá a detallar un conjunto de casos de uso que nos servirán para validar la idoneidad de la solución y el diseño propuesto para cubrir los objetivos del proyecto.
Como se verá, hemos implementado un conjunto de casos tratando de incluir las distintas áreas de funcionalidad del SIEM Wazuh de modo que nos permitan validar las distintas capacidades de la solución donde hemos puesto el foco.
Para la elección de los casos a implementar, se ha tenido en consideración posibles escenarios de seguridad vistos en la contextualizacion de la empresa por nos ha servido para ejemplificar el proyecto.
En los escenarios que se presentara, se presupone que se han desplegado todos los pasos detallados en los apartados previos y por lo tanto se dispone de:

- Cluster EKS desplegado con sus correspondientes nodos de trabajo.
- Se dispone de una credencial de API de Amazon AWS previamente configurada.
- Sistema Wazuh completamente desplegado y con la configuración inicial desplegada para su uso.
- Se considera que se dispone de varios agentes desplegados que se emplearán para las pruebas de concepto sobre distintos sistemas operativos.

7.1 Monitorización de seguridad de los servicios del entorno cloud AWS

{url14}{url15}

Una vez se da el paso en cualquier organización de emplear un entorno para el despliegue de servicios en modalidad cloud, independientemente del modelo de servicio (SaaS, PaaS como IaaS), el principal riesgo de seguridad que aparece es el control y la gobernanza del plano gestión de los servicios en proveedor de cloud.
Tal y como se puede consultar en los estudios sobre los riesgos asociados a los entornos de cloud de INCIBE (“*Riesgos y amenazas del cloud computing*” {url16}) o de la consultora Gartner (“*Assessing the Security Risks of Cloud Computing*” {url17}) entre algunos de los principales riesgos que se detallan, están:

1. Incorrecta gestión de las identidades y accesos privilegiados.
2. Falta de visibilidad y gobierno de las cargas de trabajo y nuevos servicios en nube.
3. Deficiente control y gobierno del dato.
4. Incorrecta gestión de la configuración de los elementos del cloud.

Como la solución que se ha propuesto para el diseño y arquitectura de este proyecto, implica desplegar cargas de trabajo en nube, se ha considerado idóneo que uno de los primeros casos de uso a implementar sobre el sistema Wazuh se enfoque a la monitorización y protección de la capa de gestión del proveedor cloud, haciendo énfasis en las principales amenazas a las que se enfrentan estos entornos.
Por ello, se ha aprovechado la capacidad de integración que incorpora de forma nativa el sistema Wazuh con el proveedor de cloud Amazon AWS, a través del módulo *aws-s3*, para la monitorización de eventos de seguridad. En primer lugar se detallarán los pasos de configuración y puesta en marcha de la integración para posteriormente validar 4 ejemplos, a modo de prueba de concepto, para poder con ellos validar como se detectan las amenazas dentro del sistema implementado.

7.1.1 Configuración y puesta en marcha

A continuación se detallan los pasos necesarios para habilitar la integración del proveedor cloud con el sistema Wazuh.
La integración se realiza fundamentalmente en dos grandes etapas. En primer lugar, sera necesario realizar una configuración previa dentro del proveedor cloud para que se recolecten en primera instancia los eventos que posteriormente se trasladaran al SIEM Wazuh a través el módulo “*aws-s3*”. Este módulo, actualmente soporta integración con múltiples servicios del entorno AWS. Así, empleando distintas configuraciones sobre el mismo se puede integrar información de tráfico de red (VPC), servicio de auditoria Amazon CloudTrail, tráfico y eventos de los balanceadores (ALB, NLB), DLP a través de integración con el servicio Amazon Macie, etc. Para el detalle completo de los servicios soportados, así como los distintos pasos para su configuración pueden encontrarse en el siguiente enlace.
Para esta prueba de concepto se van a integrar los eventos de seguridad del servicio de auditoría, *Amazon CloudTrail*, ya que es el que incluye toda la información básica que necesitamos para la monitorización del plano de gestión del cloud. A medida que la adopción del modelo nube y por tanto el uso de servicios dentro del proveedor fuesen aumentando, se podría necesitar ir incorporando la integración adicional de alguno de los servicios detallados en el punto previo para aportar mas información y fuentes de datos para la generación de nueva inteligencia de seguridad.
La primera etapa para la configuración pasa, como hemos visto, por habilitar y configurar el servicio *Amazon CloudTrail*. Para habilitarlo se procede del siguiente modo:

1. Se accede al panel de la consola de Amazon a través del enlace https://console.aws.amazon.com/ .
2. Desde el panel de servicio de AWS, se accede al servicio *CloudTrail.*

3. Se procede a crear un trail, a través del botón “Create a Trail”
4. Se selecciona un nombre identificativo; en este particular hemos elegido “*AuditLog*”. Asociado al *trail* que se va a desplegar, el proveedor genera automáticamente un volumen de almacenamiento asociado (*bucket*), empleando el sistema de almacenamiento de objetos *Amazon S3*, donde se van a ir almacenando todos los eventos de auditoria a modo de histórico. En este ejemplo particular, se denominará: “*aws-cloudtrail-logs-233411283712-5108c4b6*”

A partir de este momento, todos los eventos de auditoría de seguridad del uso de la cuenta comenzarán a registrarse y guardarse en el almacenamiento asociado. La siguiente etapa de configuración, trasladará esta información al SIEM Wazuh para su procesado. Este paso se puede configurar para ser procesado desde cualquiera de los posibles agentes que se tengan implementados en el sistema. Por simplicidad se optado por configurar directamente la recolección desde un agente local del que se disponía para pruebas e implementado sobre un equipo Linux identificado como “*servidor*”. Para la configuración, se procede del siguiente modo:

1. En primer lugar, se necesita disponer de unas credenciales para acceso a API de Amazon AWS. Se considera para continuar los próximos pasos que se dispone previamente de una cuenta preparada para esta función. Al tratarse de un caso de uso para la validación de un entorno de pruebas, se empleará la credencial generada en el punto 6.2, pero en el caso de tratase de un despliegue real de producción, se recomendaría la creación de una cuenta específica para esta funcionalidad, proceso que se detalla en el siguiente enlace.
2. Se edita el fichero *ossec.conf*, en este caso particular ubicado en la ruta “*/var/ossec/etc/osssec.conf*” del agente, y se incluye el siguiente bloque de código, con las adaptaciones que se verán a continuación:

```
<wodle name="aws-s3">
  <disabled>no</disabled>
  <remove_from_bucket>no</remove_from_bucket>
  <interval>30m</interval>
  <run_on_start>yes</run_on_start>
  <skip_on_error>no</skip_on_error>
  <bucket type="cloudtrail">
      <name>aws-cloudtrail-logs-233411283712-fbbf0acb</name>
      <access_key>EXAMPLE-ACCESKEY</access_key>
      <secret_key>EXAMPLE-SECRETKEY</secret_key>
      <aws_account_id>112233445</aws_account_id>
      <aws_account_alias>prod-account</aws_account_alias>-->
  </bucket>
</wodle>
```

En el detalle del código anterior, se necesitarían ajustar los valores de las variables *access_key* y *secret_key*, con los datos de la credencial de acceso identificada en el paso anterior, así como el id de la cuenta AWS, *aws_account_id*, que puede verse en el perfil de usuario dentro del panel de proveedor. Además es necesario ajustar la variable “*name*”, con el valor del bucket S3 desde donde se importaran los datos de la auditoría (Para el ejemplo generado en el paso anterior, “*aws-cloudtrail-logs-233411283712-5108c4b6*”). Adicionalmente, se puede ajustar cierto comportamiento del proceso de importación, como el intervalo de carga, a través de otras variables como “*interval*” que, como se puede apreciar en el ejemplo de configuración son autoexplicativas. Para más detalle, se pueden consultar el conjunto completo de opciones en el siguiente enlace.

3. Se reinicia el servicio Wazuh en el agente, para que se aplique la nueva configuración:

```
$ systemctl restart wazuh-agent
```

4. Finalmente se activa la integración con Wazuh desde el panel de gestión de la consola del sistema. Se accede al portal de administración de kibana (*https://kibana.wazuh.ameijeiras.es/*), dentro de la pestaña “*Settings*”, opción “*Modules*” y se activa la opción “*Amazon AWS*”.

Una vez realizados los pasos anteriores, se despliega un nuevo panel en la sección “*Modules*” desde donde se ya puede acceder al tablero principal donde se muestra la información de seguridad recolectada del proveedor cloud. Como se mostró en los pasos previos, el sistema esta configurado para recolectar la información de seguridad del proveedor de forma recurrente cada 30 minutos.

A continuación, se detallan unas imágenes donde se ilustra el sistema desplegado y una muestra de los eventos de auditoría importados:

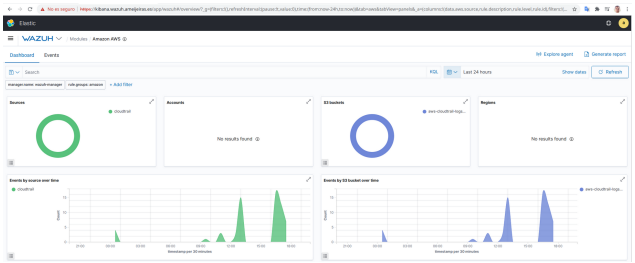


Figura 24: Tablero principal AWS

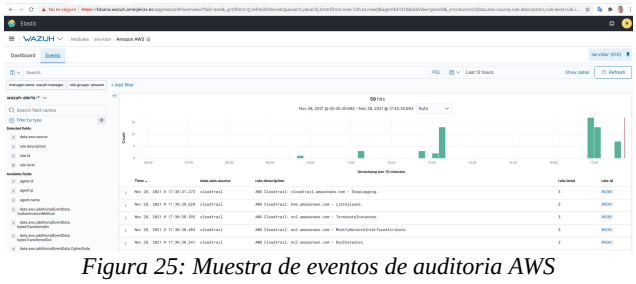


Figura 25: Muestra de eventos de auditoria AWS

7.1.2 Ejemplos de detección

A continuación se detalla un conjunto de ejemplos que se han diseñado para probar el sistema de detección de Wazuh asociado a distintos escenarios de riesgo del modelo cloud que se veían en la introducción.

7.1.2.1 Intento de ataque por fuerza bruta al panel de gestión

En este ejemplo, se simula un intento de ataque por fuerza bruta a la consola de administración de la cuenta monitorizada dentro del proveedor cloud.
Como se puede apreciar en la siguientes capturas que se han realizado para documentar la prueba, desde el panel de eventos de seguridad del sistema Wazuh se han detectado los distintos intentos de ataque al sistema, registrando información como el usuario que esta siendo atacado, la dirección ip del atacante, geoposicionamiento de la misma, número de intentos, etc. A través de esta información y eventos, se podrían generar alertas de seguridad más específicas, por ejemplo a un SOC, al personal de seguridad interno o incluso generar acciones de remediación.

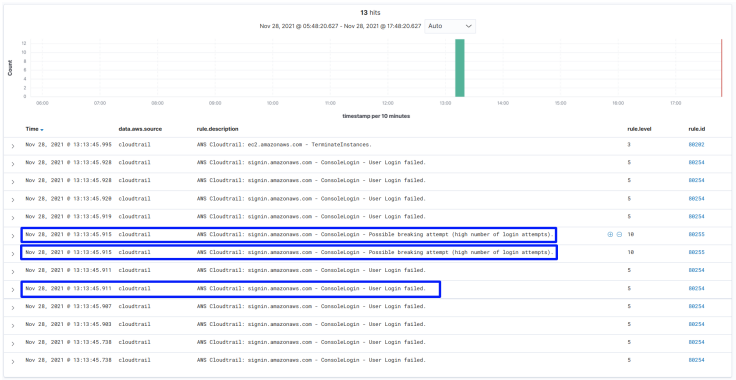


Figura 26: Detección de ataque

Figura 27: Intento de ataque de fuerza bruta

Figura 28: Detalle del ataque de fuerza bruta

7.1.2.2 Creación de un nuevo usuario y acceso al API de la plataforma

Para este ejemplo, se recrea la creación de una nueva cuenta de usuario con privilegios de administrador así como una credencial de uso de API dentro del panel de gestión del proveedor, a través de la simulación de una posible cuenta de usuario comprometida o a través del robo de una sesión a un usuario legítimo del sistema.
Como se puede ver en las siguientes capturas, a través de los eventos que se disparan en el sistema se puede detectar y generar unos indicadores de compromiso para esta casuística concreta, pudiendo de este modo generar las alertas o acciones de remediación que se considerasen necesarias.

Figura 29: Creación de un nuevo usuario

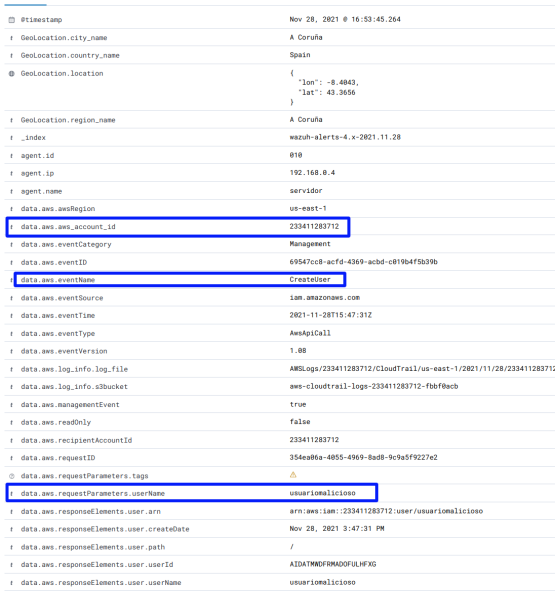


Figura 30: Detalle de la creación de un usuario

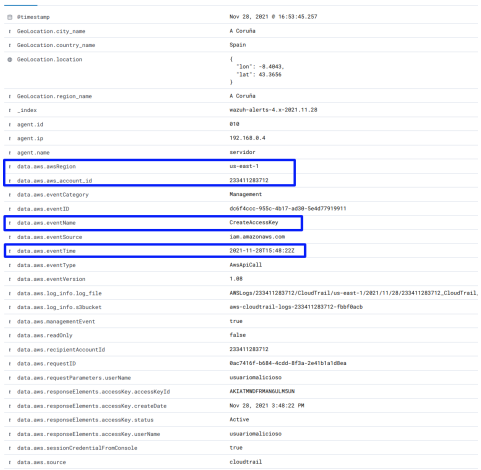


Figura 31: Generación de credenciales de API

7.1.2.3 Detección de nuevas cargas de trabajo

A través de la simulación de la existencia de una cuenta comprometida, se prueba la capacidad de identificación del sistema de auditoría de Wazuh para detectar el despliegue de cargas de trabajo o servicios no autorizados en el sistema. Al igual que en los casos anteriores, se puede comprobar que es posible detectar esta ocurrencia, así como el tipo de carga que se despliega, zona de disponibilidad, tipo de despliegue de red y exposición, ect. La generación de alertas, correlación con otros eventos o acciones de remediación es manifiesta, como en el caso de otros de los ejemplos propuestos.

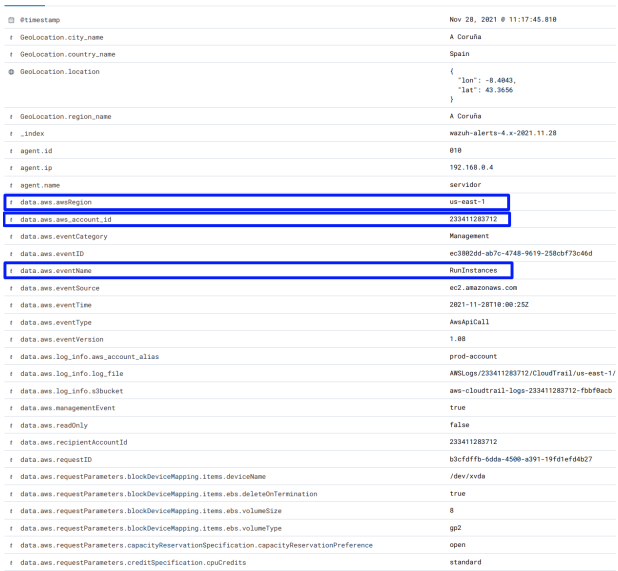


Figura 32: Detección de una nueva instancia

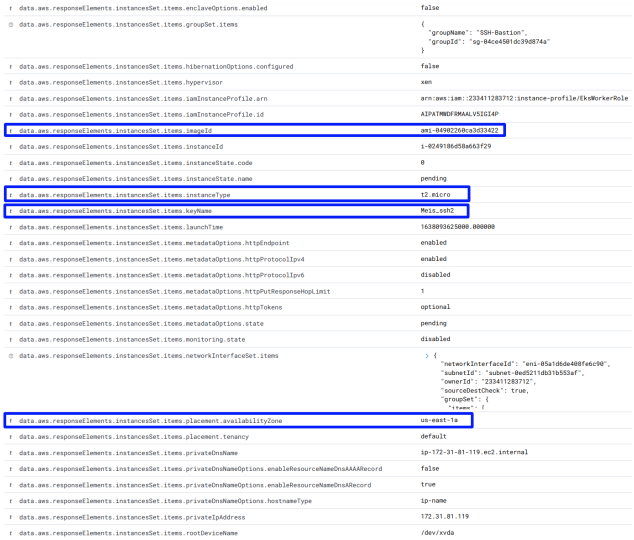


Figura 33: Detalle de la nueva instancia

7.1.2.4 Detección de cambios de configuración

De nuevo, en base a la suposición de la existencia de una cuenta comprometida, se prueba la capacidad de identificación del sistema para detectar cambios de configuración no autorizados sobre el mismo. Es este ejemplo, se prueba la detección de la exposición pública de un volumen de datos preexistente con información sensible, simulando una exfiltración a través de un cambio de configuración (intencionado o no). Al igual que en los casos anteriores, se puede comprobar que es posible detectar esta ocurrencia, así como el usuario que la provoca, el cambio de configuración y la

identificación del *bucket* de datos afectado. La generación de alertas, correlación con otros eventos o acciones de remediación vuelve a ser manifiesta.

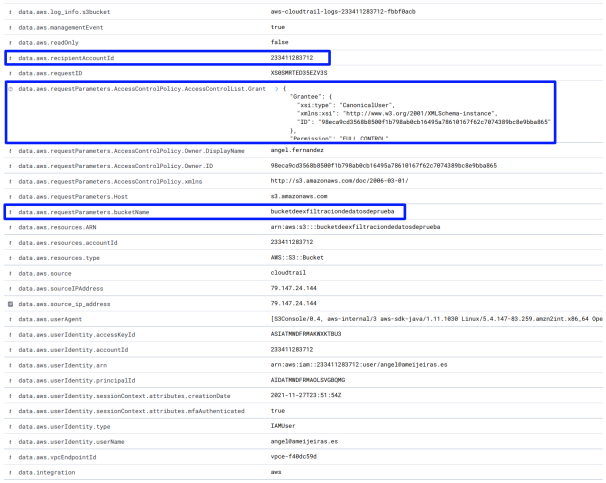


Figura 34: Detección de cambios de configuración no autorizados

7.2 Monitorización de seguridad de las cargas de trabajo en los nodos AWS

A través de la prueba de concepto implementada en el apartado anterior se valida el control de los eventos de auditoria del proveedor cloud gracias al sistema implementado. El siguente punto que se ha considerado clave para una prueba de concepto es integrar la capacidad de monitorizar la seguridad y los eventos de auditoria en las cargas de trabajo del entorno que, como se ha detallado en este proyecto, se basa en cargas a través de nodos de trabajo conectados a un cluster de Kubernetes y ejecutandose en contenedores sobre el sistema Docker.
Para ello, se implementará la integración de eventos recolectados del sistema de contenedores Docker, capacidad que ya incluye el sistema Wazuh de manera nativa.
Se definirán en primer lugar los pasos de configuración de esta integración para a continuación validar y documentar a través de 3 pruebas el funcionamiento completo de la integración.
Esta integración se puede realizar instalando y configurando el agente Wazuh en cualquier equipo donde se este ejecutando el sistema de contenedores Docker que queramos monitorizar. En este caso particular, para monitorizar las cargas de trabajo que dan soporte al proyecto tenemos previamente desplegados 3 equipos con el rol de nodos de trabajo del cluster EKS. Para complementar la seguridad de la solución vamos a realizar la implementación e integración con estos 3 equipos, que como hemos visto son 3 servidores virtuales Amazon EC2.
Consideraciones previas:
Previo a continuar con los pasos de configuración para la implementación del caso, se presupone que se han realizado los pasos detallados en los apartados previos y se asumen las siguientes consideraciones:

- Se dispone de un cluster EKS desplegado con sus correspondientes nodos de trabajo.
- Sistema Wazuh completamente desplegado con la configuración inicial desplegada y listo para su uso.
- Se dispone de tres máquinas virtuales que se emplean como nodos de trabajo del cluster *EKS*. Se trata de 3 servidores Amazon EC2 ejecutando el sistema operativo Amazon Linux en instancias *t3.large*. Los servidores se han configurado en el momento de despliegue del cluster de EKS para ser accesibles a su gestión a través de acceso administrativo SSH y con un sistema de claves asimétricas como mecanismo de auntication. Tenemos previamente disponible en acceso con un usuario de gestión administrativa (con capacidad de privilegios *root*) sobre estos agentes para su configuración.

7.2.1 Configuración y puesta en marcha

A continuación se detallan los pasos necesarios para habilitar la integración del sistema Wazuh con la monitorización de contenedores Docker en los nodos de trabajo de cluster Kubernetes.
Para ello, se procede individualmente dentro de cada uno de los 3 servidores realizando los siguienes pasos:

1. Se accede a cada servidor individualmente a través del acceso administrativo SSH con el usuario por defecto de este sistema “*ec2-user*”, para a continuación elevar privilegios y establecerse en el sistema como usuario root (“*sudo su*”). Se procede a continuación a instalar el agente Wazuh para el sistema Amazon Linux. El proceso y pasos de instalación en detalle se incluye en el siguiente anexo.

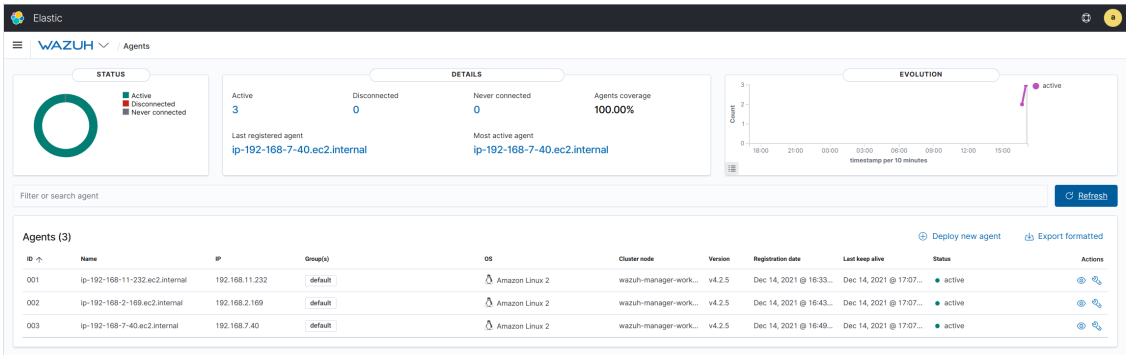


Figura 35: Agentes instalados en nodos de trabajo

2. Una vez se ha instalado el agente Wazuh en los nodos de trabajo, para el correcto funcionamiento de la integración es necesario instalar una librería del entorno *python* del sistema y realizar un reajuste de la versión de *python* por defecto en el sistema. Se procede con los siguientes comandos:

```
$ pip install docker
$ rm /usr/bin/python
$ln -s /usr/bin/python3 /usr/bin/python
```

3. Se edita el fichero *ossec.conf*, en este caso particular ubicado en la ruta "*/var/ossec/etc/osssec.conf*" dentro de cada agente, y se incluye el siguiente bloque de código, ajustando si se considera necesario el intervalo de actualizacion a través del parámetro *interval*:

```
<ossec_config>
  <wodle name="docker-listener">
    <interval>30m</interval>
    <attempts>5</attempts>
    <run_on_start>yes</run_on_start>
    <disabled>no</disabled>
  </wodle>
</ossec_config>
```

5. Se reinicia el servicio Wazuh en cada agente, para que se aplique la nueva configuración:

```
$ systemctl restart wazuh-agent
```

6. Finalmente se activa la integración desde el panel de gestión de la consola del sistema Wazuh. Se accede al portal de administración de kibana (*https://kibana.wazuh.ameijeiras.es/*), dentro de la pestaña "*Settings*", opción "*Modules*" y se activa la opción "*Docker Listener*". A partir de este instante, dentro del panel de cada agente donde se haya desplegado la configuración anterior, podremos consultar el cuadro de mando para Docker así como los eventos de seguridad especificos asociados a cada agente.

Se muestra en las siguientes imagenes, los cuadros de mando Docker de alguno de los agentes desplegados así como una muestra de la captura de los eventos de seguridad del entorno:

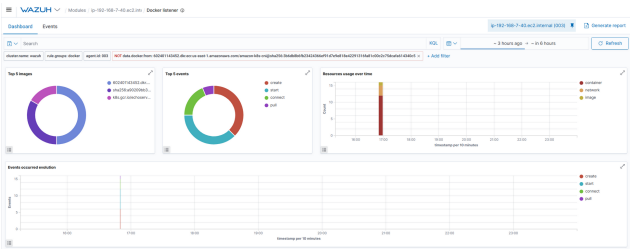


Figura 36: Detalle integración Docker Agente 1

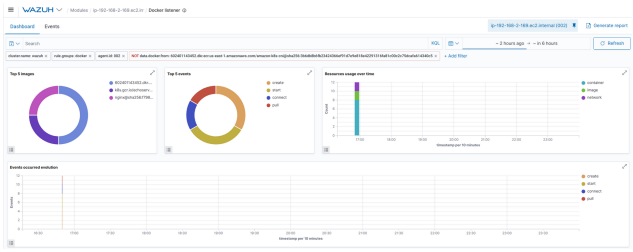


Figura 37: Detalle integración Docker Agente 2

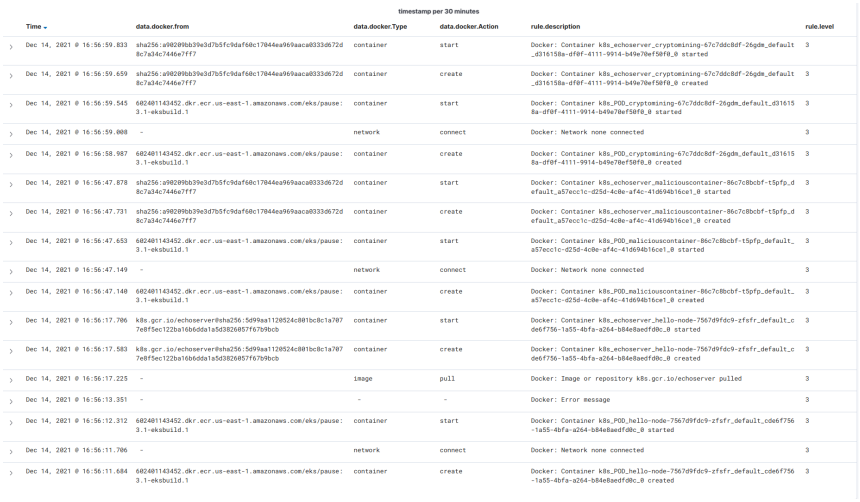


Figura 38: Detalle de eventos de seguridad en contenedores

7.2.2 Ejemplos de detección

A continuación se detalla un conjunto de ejemplos que se han diseñado para probar el sistema de detección Wazuh asociado a la detección de eventos en el entorno de contenedores:

7.2.2.1 Aparición de nuevas cargas de trabajo

En este ejemplo, se simula la creación y el arranque dentro del entorno de nodos de trabajo de varios contenedores nuevos desplegados dentro del entorno. En este ejemplo, se recrea la aparición y despliegue de un contenedor de criptominado "*criptomining*", otro contenedor malicioso "*maliciouscontainer*" y algún otro contenedor de ejemplo.

```
angel.fernandez@servidor:~/codigo/git/wazuh> kubectl get pods --all-namespaces
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
default       cryptomining-67c7ddc8df-26gdm         1/1     Running   0          9m12s
default       hello-node-7567d9fdc9-zfsfr           1/1     Running   0          9m59s
default       kali-845df89967-qt4hn                 1/1     Running   0          9m36s
default       maliciouscontainer-86c7c8bcbf-t5pfp   1/1     Running   0          9m24s
default       nginx-deployment-66b6c48dd5-6sbk8     1/1     Running   0          7m7s
default       nginx-deployment-66b6c48dd5-qqdwt     1/1     Running   0          7m7s
kube-system   aws-node-k7q5z                        1/1     Running   0          54m
kube-system   aws-node-mdh62                        1/1     Running   0          54m
kube-system   aws-node-wzwph                        1/1     Running   0          54m
kube-system   coredns-66cb55d4f4-gsh82              1/1     Running   0          67m
kube-system   coredns-66cb55d4f4-nnpj6              1/1     Running   0          67m
kube-system   kube-proxy-84vd8                      1/1     Running   0          54m
kube-system   kube-proxy-sznxg                      1/1     Running   0          54m
kube-system   kube-proxy-vfvtg                      1/1     Running   0          54m
wazuh         wazuh-elasticsearch-0                 1/1     Running   0          50m
wazuh         wazuh-elasticsearch-1                 1/1     Running   0          49m
wazuh         wazuh-elasticsearch-2                 1/1     Running   0          49m
wazuh         wazuh-kibana-5cd658b78-hhblh          1/1     Running   0          50m
wazuh         wazuh-manager-master-0                1/1     Running   0          50m
wazuh         wazuh-manager-worker-0                1/1     Running   0          50m
wazuh         wazuh-manager-worker-1                1/1     Running   0          50m
angel.fernandez@servidor:~/codigo/git/wazuh>
```

Figura 39: Detalle de los nuevos despligues de prueba

Como se puede apreciar en la siguientes capturas que se han realizado para documentar la prueba, desde el panel de seguridad del sistema Wazuh se han detectado los distintos eventos que se generan al arrancar, desplegar e iniciar nuevos contenedores dentro del sistema. Como se puede observar, a través de los eventos de Wazuh se puede ver el nombre del contenedor, la imagen desde la que se ha compilado, fecha y otra del despleigue, etc.
A través de esta información y actividad, se podrían generar alertas de seguridad más específicas en función de, por ejemplo, el despliegue de una imagen que no este en una lista de imagenes corporativas en cuyo caso de prodría generar una alerta a un SOC, al personal interno de seguridad o incluso generar acciones de remediación como parar o aislar el contenedor no autorizado.

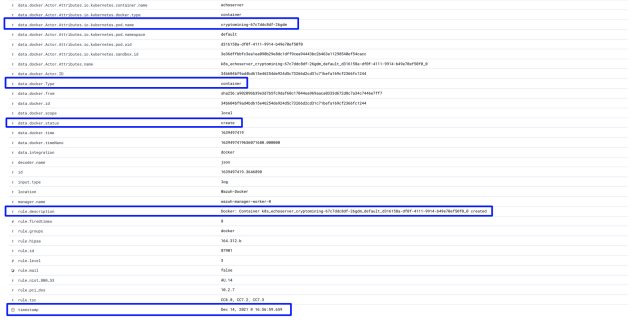


Figura 40: Despliegue de contenedor malicioso 1

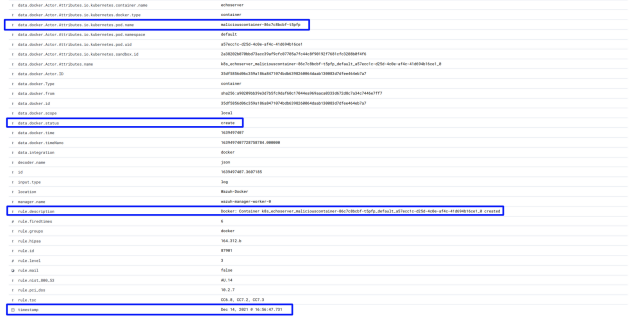


Figura 41: Despliegue de contenedor malicioso 2

7.2.2.2 Detección de nuevos volumenes de datos y montajes en contenedor

En este ejemplo, se simula la aparición de un nuevo volumen de datos permistentes en el sistema así como posible montaje de datos dentro de un contenedor. Se recrea un volumen que se empleará para una supuesta simulación de exfiltración de datos.
Como se puede apreciar en la siguientes capturas desde el panel de eventos de seguridad del SIEM se han detectado los distintos eventos que se generan al crear un nuevo volumen de almacenamiento y un posible montaje sobre un contendor en ejecución. A través de los eventos de Wazuh se puede ver el nombre del volumen, los puntos de montaje, fecha y otra del despleigue, etc.
De nuevo al igual que el ejemplo anterior, la generación de distintas alertas en base a estos eventos sería una opción trivial a través del potente sistema de reglas de alertas que incorpora el sistema.

Figura 42: Evento de creación de un volumen persistente

Figura 43: Evento de montaje en un volumen

7.2.2.3 Aparición y conexión de nuevos elementos de red

En este último ejemplo, se genera un nueva red sobre el sistema de los contenedores simulando de este modo una posible conexión a o creación de una zona de red no autorizada. Al igual que en los casos anteriores a través del sistema de eventos se detecta esta nuevo elemento de red, fecha del evento, nombre de la red y otros detalles.

7.3 Detección y mitigación de malware a través de la integración con VirusTotal

{url15}

Según uno de últimos informes hechos públicos por la empresa lider en seguridad M*cAfee*[27], "*quarterly-threats-apr-2021*" {inve02}, en el primer trimestre del año 2021 el crecimiento de malware representa un incremento del 53% respecto del año anterior. La continua evolución de las variantes y el aumento de la complejidad en la detección representan uno de los mayores retos a los que se enfrentan los sistemas de detección y los equipos de seguridad. Por este motivo, se ha considerado la necesidad de implementación de algún caso de prueba al respecto.
Se detallan a continuación los pasos necesarios para la implementación y configuración, que nos permitirá la detección automática de malware en el sistema desplegado, a través de la integración con el sistema VirusTotal[28] así como la definición de una acción de mitigación y respuesta para su tratamiento.
VirusTotal, es una empresa de servicios de seguridad que proporciona el servicio de análisis de archivos y páginas web sospechosos (actualmente incluye 55 antivirus y 65 motores de detección en línea) a través del envió de muestras automatizadas mediante un sistema de integración con un API propia que exponen. El sistema Wazuh, incorpora capacidad de integración nativa con el análisis de muestras de VirusTotal, de modo que se realizará la configuración necesaria para que se convierta en nuestro sistema de detección de malware. Adicionalmente, entre las funcionalidades del sistema Wazuh tambien se incluye un sistema de respuesta activa para la ejecución de acciones de remediación la cual tambien se empleará para el tratamiento en caso de darse un positivo.
La sistema de detonación del proceso de análisis se realiza a través del sistema FIM, File Integrity Monitoring, que incorpora el SIEM de forma nativa en los agentes. Se activará para esta configuración, la monitorización en tiempo real del directorio de trabajo de los usuarios (*home*) donde se realizan por defecto la descargas del navegador, ya que se ha considerado un potencial punto de entrada de ficheros peligrosos. Se implementa además, un script de remediación, que se ejecutará en los agentes de manera local en caso de detección de malware. A través de este script de remediación, una vez se confirma la identificación de un malware, se mueve el fichero a un directorio de cuarentena dentro del equipo donde se ha detectado y se ubica en una zona protegida, preservándolo para un posterior análisis pero eliminándolo del espacio de usuario.

Consideraciones previas:
Previo a continuar con los pasos de configuración para la implementación del caso, se presupone que se han realizado los pasos detallados en los apartados previos y se asumen las siguientes consideraciones:

- Se dispone de un cluster EKS desplegado con sus correspondientes nodos de trabajo.
- Sistema Wazuh completamente desplegado con la configuración inicial desplegada y listo para su uso.
- Se dispone de los ficheros de configuración e implementacion de Kubernetes empleados para el despliegue de la solución, acceso a su edición y capacidad de redesplegar de nuevo todo el entorno para poder aplicar cambios de configuración en los nodos que conforman la capa de gestión del sistema Wazuh.
- Para la realización de la prueba de integración se necesita disponer de una credencial de acceso para el uso del API de VirusTotal. Se presupone en este apartado que ya se dispone previamente de esta credencial. En caso de que no fuese así, sería necesario registrarse a través del portal de VirusTotal y proceder como se indica en el siguiente enlace.
- Como la credencial del API de VirusTotal de la que se dispone es de la versión gratuita, existe una limitación en el número de usos, por lo que se acota para la prueba a la configuración de detección dentro del agente para un único usuario local en el equipo (Usuario de prueba "*angel*" a afectos de configuración y capturas de pantalla/video).
- Se dispone de un agente previamente configurado y conectado al entorno Wazuh, que se empleará como equipo para probar la viabilidad de la solución. Este agente, se encuentra instalado en una máquina con sistema operativo Linux Ubuntu 20.04 y corresponde a un equipo de tipo cliente de escritorio con nombre "*angel-Standard-PC-Q35-ICH9-2009*".
- En el equipo agente se consideran instaladas las herramientas básicas para la ejecución de comandos del sistema Ubuntu, así como la herramienta *jq* del sistema que se empleará dentro del comando que realizará la acción de respuesta al malware detectado y que se emplea para el procesado del formato JSON de las respuestas.
- Se asume que existe un sistema automático externo a los agentes que diariamente recolecta los posibles eventos que se han generado en la cuarentena en los equipos (directorio */var/ossec/quarantine*), para archivarlos en una bóveda segura y posteriormente eliminarlo del equipo. El fin del archivado es que se puedan emplear para análisis de seguridad más detallados, para respuesta a incidentes de seguridad o con fines de auditoría.
- Algunos de los ajustes de configuración se realizan a nivel de agente por lo que sería necesario que se implementasen individualmente en cada uno de los agentes donde se necesitase implementar la funcionalidad. Para este particular, se han supuesto los pasos de configuración en un único agente por simplicidad, pero en

27 https://www.mcafee.com/
28 https://www.virustotal.com/

caso de tener que desplegarse en un parque más amplio, sería recomendado adaptar los pasos de configuración para implementar los cambios a través del sistema de despliegue de *configuración de agentes centralizado* [29].

- Es necesario tener en consideración que para realizar ajustes dentro de la configuración del sistema Wazuh (para cambiar la configuración del nodo *manager* o *master*), al ser desplegado en contenedores, para que los cambios sean persistentes es necesario realizarlos dentro del equipo desde donde se controla la orquestación y redesplegar el sistema para que los cambios se apliquen. Se denominará en adelante, equipo "*cliente de orquestación*" y "*directorio raíz de configuración de despliegue*" al equipo y directorio raíz donde se descarga el código fuente de la solución Wazuh detallado en el apartado 6.3 y que se emplea para gestionar el cluster Kubernetes y el sistema Wazuh.

7.3.1 Configuración y puesta en marcha

Para poder configurar este escenario, se necesitan editar ficheros de configuración, tanto dentro del nodo *master* del entorno Wazuh donde se gestiona la configuración de manera centralizada, como en cada uno de los agentes a los que se vaya a añadir esta funcionalidad.
Se configura y habilita en primer lugar la integración del sistema con el API de VirusTotal. Para ello, se realizan los siguientes pasos que veremos a continuación dentro del nodo *master* del sistema Wazuh:

1. Se parte del equipo "*cliente de orquestación*" y desde el "*directorio raíz de configuración de despliegue*". Se edita el fichero *wazuh-kubernetes/wazuh/wazuh_managers/wazuh_conf/master.conf*, que se corresponde con el fichero *ossec.conf* una vez se despliegue el nodo manager del sistema. Se adjunta al final del mismo el código que se detalla a continuación, ajustando la variable *api_key* con la credencial de API obtenida previamente:

```
<ossec_config>
  <integration>
    <name>virustotal</name>
    <api_key>AÑADIR_API_VIRUSTOTAL_AQUI</api_key>
    <rule_id>100200,100201</rule_id>
    <alert_format>json</alert_format>
  </integration>
</ossec_config>
```

2. A continuación, se aplica la nueva configuración en el entorno a través del siguiente comando:

```
$ kubectl apply -k .
```

Una vez realizado el proceso anterior, se ha completado el proceso de integración con VirusTotal y ya se dispone de un sistema que permite el envío de cualquier fichero para su análisis a través se su API cuando se disparen las reglas 100200 o 100201.
A continuación, se procede a realizar los cambios necesarios dentro del nodo manager, de modo que una vez se detecte a través del sistema FIM (Monitorización de integridad de ficheros) la acción de añadir o modificar un fichero dentro del directorio objetivo de la monitorización (*sid=550* y *sid=554* respectivamente), se disparen las reglas definidas en el apartado anterior de envío para su análisis (*rule_id=100200* y *rule_id100201*). Para ello se procede mediante los siguientes pasos:

1. Se parte desde el equipo "*cliente de orquestación*" y desde el "*directorio raíz de configuración de despliegue*". Se edita el fichero *wazuh-kubernetes/wazuh/wazuh_managers/wazuh_conf/rules/local_rules.xml* (Este fichero se monta y mapea dentro de nodo manager y es el que nos permite persistir la configuración del fichero *local_rules.xml* estándar de Wazuh). Se añade al final del mismo el siguiente bloque de código:

```
<group name="syscheck,pci_dss_11.5,nist_800_53_SI.7,">
    <rule id="100200" level="7">
        <if_sid>550</if_sid>
        <field name="file">/home/angel/Descargas</field>
        <description>File modified in directory.</description>
    </rule>
        <rule id="100201" level="7">
        <if_sid>554</if_sid>
        <field name="file">/home/angel/Descargas</field>
        <description>File added to directory.</description>
    </rule>
</group>
```

2. A continuación, se aplica la configuración a través del comando:

```
$ kubectl apply -k .
```

29 https://documentation.wazuh.com/current/user-manual/reference/centralized-configuration.html

El siguiente paso se realiza dentro del nodo agente, que como hemos visto en las consideraciones previas, es un equipo con sistema operativo Linux Ubuntu 20.04 que se ya encuentra previamente disponible y conectado al sistemaWazuh.
En primer lugar, se configura el módulo FIM para que integre en su monitorización el directorio sobre el que se ha considerado realizar una monitorización en tiempo real. Para ello, se edita el fichero */var/ossec/etc/ossec.conf* del agente y se añade el siguiente bloque de código

```
<syscheck>
    <directories whodata="yes">/home/angel/Descargas</directories>
</syscheck>
```

A continuación se procede a preparar y configurar los componentes necesarios para controlar el comportamiento de respuesta activa que se detonará en caso de detección de malware. Se exponen a continuación los pasos:

1. Se crea un script en lenguaje BASH dentro de la ruta */var/ossec/active-response/bin/* con el nombre *delete_and_quarantine.sh* y con en siguiente código:

```
#!/bin/bash


LOCAL=`dirname $0`;
cd $LOCAL
cd ../
PWD=`pwd`
read INPUT_JSON
FILENAME=$(echo $INPUT_JSON | jq -r .parameters.alert.data.virustotal.source.file)
COMMAND=$(echo $INPUT_JSON | jq -r .command)
LOG_FILE="${PWD}/../logs/active-responses.log"
PASSWORD="SomePassword"
QUARANTINE="/var/ossec/quarantine"
if [ ${COMMAND} = "add" ]
then
  # Send control message to execd
 printf '{"version":1,"origin":{"name":"remove-threat","module":"active-response"},"command":"check_keys", "parameters":{"keys":[]}}\n'
  read RESPONSE
  COMMAND2=$(echo $RESPONSE | jq -r .command)
  if [ ${COMMAND2} != "continue" ]
  then
    echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Moviendo amenaza a cuarentena" >> ${LOG_FILE}
  exit 0;
  fi
fi
# Se mueve la amenaza al directorio de cuarentena que despues se recogera para analisis en honeypod y se elimina
NOW=$(date '+%Y%m%d_%H%M%S')
#echo "##DEBUG--> /usr/bin/zip -q --password $PASSWORD $QUARANTINE/${FILENAME##*/}_$NOW.zip $FILENAME" >> ${LOG_FILE}
/usr/bin/zip -q --password $PASSWORD $QUARANTINE/${FILENAME##*/}_$NOW.zip $FILENAME && rm -f $FILENAME && rm -f $FILENAME
if [ $? -eq 0 ]; then
  echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Se ha trasladado la amenaza a cuarentena correctamente" >> ${LOG_FILE}
else
  echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Error moviendo amenaza a la cuarentena" >> ${LOG_FILE}
fi
exit 0;
```

   Como se puede apreciar en el detalle del código anterior, el código extrae el nombre del fichero y el resultado del proceso de análisis del malware. En caso de ser positivo, se comprime el fichero en el directorio de cuarentena, protegiendolo con un password previamente establecido y se elimina del espacio de usuario.
2. Se crea el directorio de cuarentena:

```
$ mkdir /var/ossec/quarantine
```

3. Se reajusta el propietario y los permisos, tanto para el directorio de cuarentena como para que el script de remediación pueda ser ejecutado correctamente:

```
$ chmod 750 /var/ossec/quarantine
$ chmod root:ossec /var/ossec/quarantine
$ chmod 750 /var/ossec/active-response/bin/delete_and_quarantine.sh
$ chown root:ossec /var/ossec/active-response/bin/delete_and_quarantine.sh
```

4. Finalmente se reinicia el servicio Wazuh sobre el agente:

```
$ systemctl restart wazuh-agent.service
```

En este punto, ya se han desarrollado todos los pasos de configuración necesarios dentro de agente, por lo que solo resta terminar de configurar el proceso de respuesta activa en la configuración del nodo manager Wazuh. Para ello, se describen los pasos a realizar en mismo:

1. Desde el equipo "cliente de orquestación" y desde el "directorio raíz de configuración de despliegue". Se edita el fichero *wazuh-kubernetes/wazuh/wazuh_managers/wazuh_conf/master.conf*, que ya hemos visto que se corresponde con el fichero *ossec.conf* una vez se despliegue el nodo manager en un contenedor. Para poder emplear el script que se ha definido en el apartado anterior, es necesario previamente declararlo y configurarlo para que se ejecute de manera local en cada agente. Para ello, se añade al final del fichero de configuración el siguiente código:

```
<ossec_config>
    <command>
        <name>quarantine-threat</name>
        <executable>delete_and_quarantine.sh</executable>
        <timeout_allowed>no</timeout_allowed>
    </command>
    <active-response>
        <disabled>no</disabled>
        <command>quarantine-threat</command>
        <location>local</location>
        <rules_id>87105</rules_id>
    </active-response>
</ossec_config>
```

2. Desde el "*cliente de orquestación*" y "*directorio raíz de configuración de despliegue*", se edita el fichero *wazuh-kubernetes/wazuh/wazuh_managers/wazuh_conf/rules/local_rules.xml* para incluir el detalle de configuración que gestiona que se detone respuesta en función del resultado del análisis de VirusTotal:

```
<group name="virustotal,">
  <rule id="100092" level="12">
    <if_sid>657</if_sid>
    <match>Successfully quarantine threat</match>
                <description>$(parameters.program)    quarantine    threat    located    at    $
(parameters.alert.data.virustotal.source.file)</description>
  </rule>
  <rule id="100093" level="12">
    <if_sid>657</if_sid>
    <match>Error moving threat to quarantine</match>
     <description>Error moving threat located at $(parameters.alert.data.virustotal.source.file) to
quarantine</description>
  </rule>
</group>
```

3. Desde la misma ruta que en el paso anterior, se edita el fichero *wazuh-kubernetes/wazuh/wazuh_managers/wazuh_conf/decoders/local_decoder.xml* para configurar un nuevo *decoder* que se empleará para la extracción de información que se empleará para registrar el detalle del procesado del malware a la cuarentena. Se añade en siguiente bloque de configuración:

```
<decoder name="ar_log_fields">
    <parent>ar_log</parent>
    <regex offset="after_parent">^(\S+) Quarantined threat located at (\S+)</regex>
    <order>script_name, path</order>
</decoder>
```

4. Finalmente se despliega de nuevo la configuración para aplicar todos los cambios sobre el nodo manager (desde "*cliente de orquestación*" y "*directorio raíz de configuración de despliegue*"):

```
$ kubectl apply -k .
```

En este punto, ya se dispone del proceso de implementación al completo, por lo que para terminar de documentar el detalle del proceso, se incluyen unas capturas de pantalla del funcionamiento del sistema.

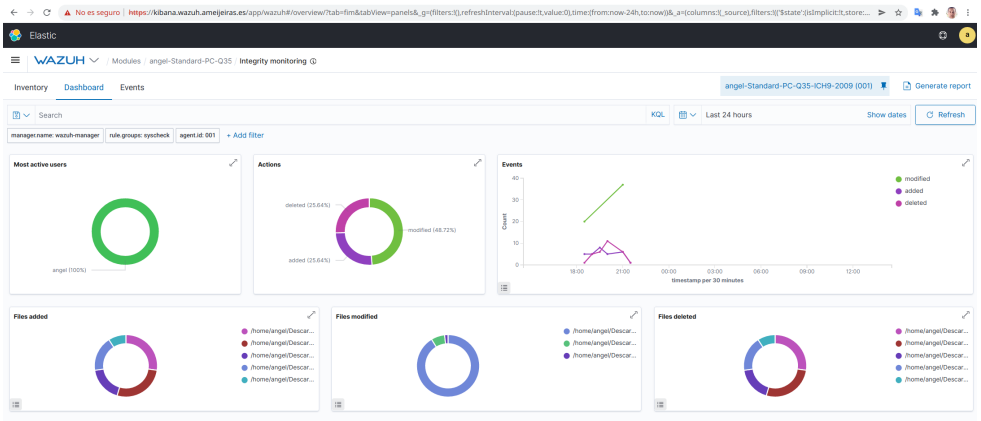


Figura 44: Detalle FIM en agente

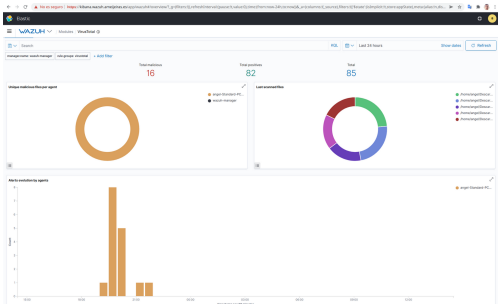


Figura 45: Integración VirusTotal

Figura 46: Integración con respuesta activa

7.3.2 Ejemplo de detección

A continuación se detalla un conjunto de ejemplos que se han diseñado para probar el sistema de detección y las distintas integraciones que se ha implementado.

7.3.2.1 Detección de descarga de malware y remediación

Para probar la implementación, se va a simular el comportamiento tipo de un usuario trabajando con un equipo de escritorio,donde se ha instalado el agente y configurado con la solución que se indica en la sección anterior. El usuario descargará varios ficheros de internet, algunos de los cuales están infectados con malware.
Los ficheros que el usuario descarga, son enviados para su revisión y análisis automático a través de la integración con el sistema de análisis web de VirusTotal. Si se detecta infección en alguno de los ficheros, se ejecutará en el equipo una primera acción automática de remediación que trasladará cada fichero infectado a un sistema de cuarentena y se eliminará del espació de usuario.
Para la simulación del malware se han empleado dos tipos de ficheros de prueba, inocuos para el sistema, pero que nos permiten realizar el conjunto de pruebas completo:

- *EICAR*[30], descargando un fichero de pruebas especialmente preparado para ser detectado como un malware.
- *Fransom*{code01}. Se ha compilado y creado un ejecutable en base a este proyecto de desarrollo, a través del cual se genera una herramienta que permite realizar distintas pruebas, en base a simular ciertos indicadores de compromiso que se suelen ver en el comportamiento habitual del ransomware.

Se documenta, a través de las siguientes capturas de pantalla como el comportamiento del sistema es el esperado y se detectan y remedian, dos amenazas. En el proceso, se descargan en el equipo 3 ficheros:

- putty.exe, fichero inocuo.
- MalwareDescargado.exe, fichero ejecutable compilado del proyecto *Fransom.*
- *eicarcom2.zip*, fichero *EICAR.*

Inicialmente, se accede al panel de eventos de seguridad del agente que se ha utilizado para la prueba y ya se puede ver que se están ejecutando reglas de activación de VirusTotal (Rosa) y acciones de remediación (Amarillo claro.)

30 https://www.eicar.org/

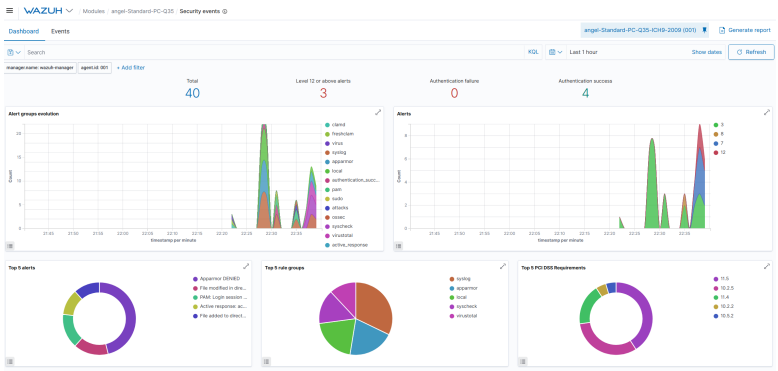


Figura 47: Cuadro de mando de seguridad del agente

Como se puede apreciar, desde el módulo de integración con VirusTotal en el panel de Wazuh, se puede observar el detalle de los ficheros descargados y analizados, así como el resultado y el informe detallado de cada proceso de análisis:

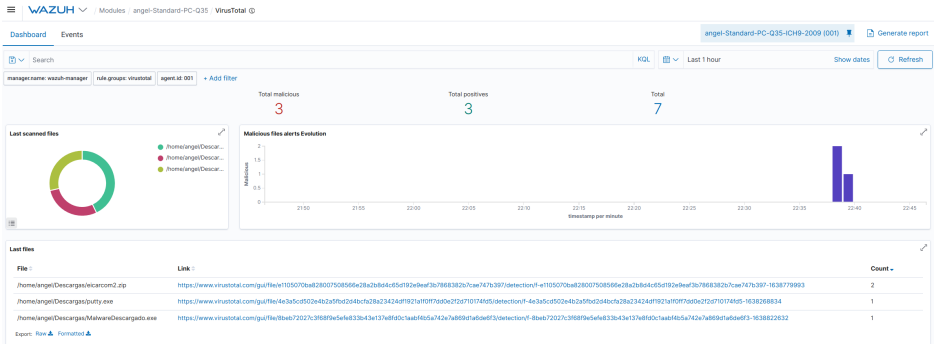


Figura 48: Análisis de ficheros descargados

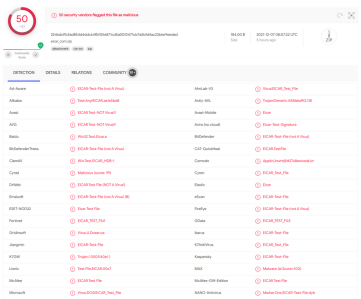


Figura 49: Informe VirusTotal EICAR

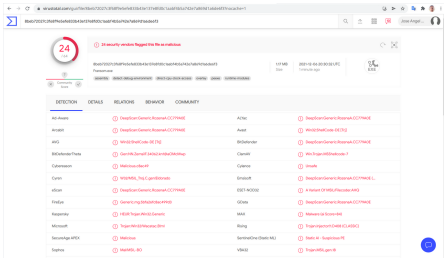


Figura 50: Informe VirusTotal Fransom

A continuación se incluye un detalle las alertas de seguridad identificadas por Wazuh y las acciones que detona cada regla:

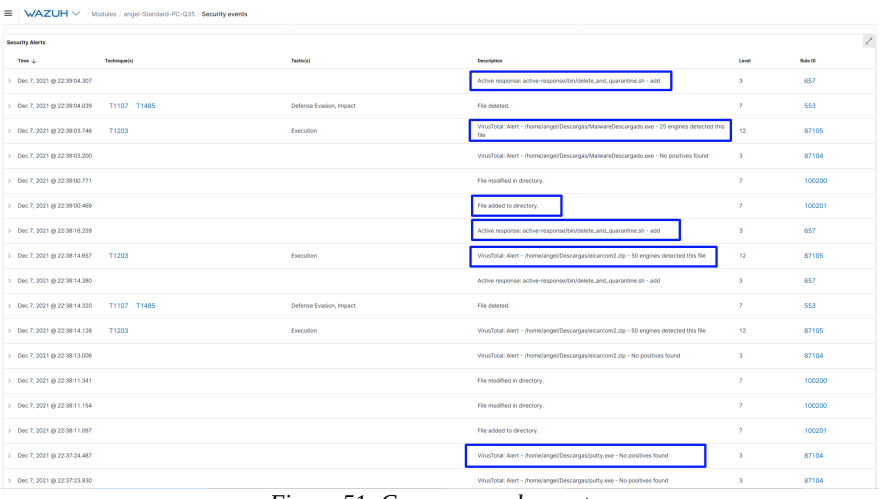


Figura 51: Cronograma de eventos

Finalmente, se incluye el detalle del proceso que ha pasado cada uno de los ficheros descargados y como, al resultar un caso positivo, se ejecuta la respuesta en el agente y donde se puede ver el resultado de su acción.

Figura 52: Proceso de detección y respuesta

Figura 53: Detalle de la regla de respuesta

8 Conclusiones

8.1 Estimación costes

En este apartado se realiza una aproximación de los posibles costes de implantación de la solución propuesta. Se han dividido los costes en 2 bloques diferenciados. Por un lado se estima el coste a nivel de infraestructura y servicios cloud que son necesarios para abordar la solución técnica presentada. Este se ha calculado en base al precio de los servicios en la región de EE.UU. Este en North Virgina,incluyendo:

- El servicio de cluster EKS administrado por el proveedor y desplegado en 3 zonas de disponibilidad.
- 3 servidores EC2 del tipo t3.large, con disco local SSD de 50Gb que se emplean como nodos de trabajo para el cluster del servicio Amazon EKS.
- 2 volumenes de almacenamiento EBS para la persistencia de datos, con una estimación de 2 TB de almacenamiento.
- 3 balanceadores de carga necesarios para la publicación de servicios, con una estimación de 2 TB de tráfico mensual.

Por otro lado, se estima el coste de implementación inicial del sistema y la operación mensual de mantenimiento y evolución básica que podría conllevar. Para la realización del calculo, se ha supuesto la ejecución del trabajo empleando un perfil *Cloud Specialist Senior* y su coste/hora se ha estimado en base a un estudio salarial de la consultora de recursos humanos MichaelPage, de este mismo año. {est01}
A continuación, se detalla el desglose de ambos bloques:

Tabla 7: Coste estimado

Servicio	Descripción	Región	Coste/mes
Amazon EKS	Número de clústeres de EKS (1)	EE. UU. Este (Norte de Virginia)	64,43 €
Amazon EC2	Sistema operativo (Linux), Cantidad (3), Estrategia de precios (EC2 Instance Savings Plans 1 año Sin pagos iniciales), Cantidad de almacenamiento (50 GB), Tipo de instancia (t3.large)	EE. UU. Este (Norte de Virginia)	114,14 €
Amazon EBS	Número de volúmenes (2), Duración promedio de la ejecución de cada instancia (730 horas al mes), Cantidad de almacenamiento Por volumen (1 TB),	EE. UU. Este (Norte de Virginia)	81,20 €
Network Load Balancer	Número de balanceadores de carga de red (3), Bytes procesados por NLB para TCP (2 TB por mes)	EE. UU. Este (Norte de Virginia)	76,04 €
			335,81 €

Descripción	Perfil	Coste / hora	Horas	Total
Implementación y configuración inicial	Cloud Specialist Senior	33,00 €	40	1.320,00 €
Coste mensual de Operación	Cloud Specialist Senior	33,00 €	8	264,00 €

Como apoyo a la estimación del coste de infraestructura presentados en la tabla anterior, se ha empleado la herramienta *calculadora de costes*[31] que facilita el proveedor Amazon, detalle de la cual se incluye a continuación:

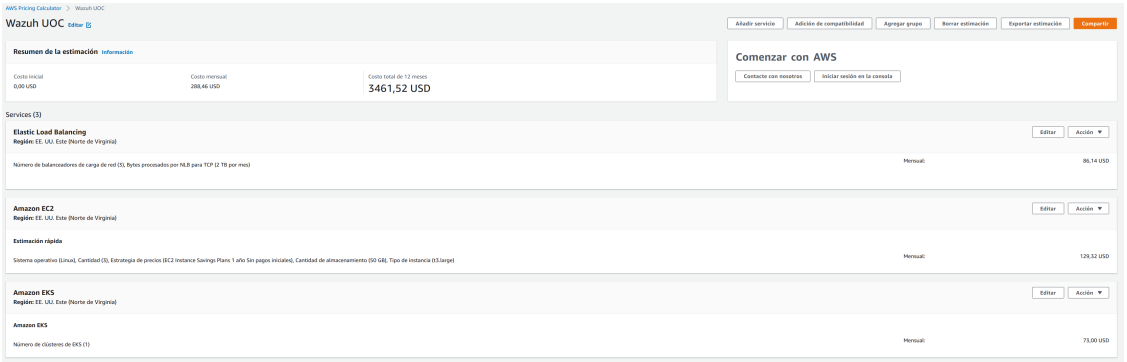


Figura 54: Calculadora de costes AWS de la solución

8.2 Repositorio de recursos

Para la gestión y orquestación de los recursos y desplegables de este proyecto se ha habilitado un repositorio público donde se puede encontrar tanto la documentación de la arquitectura como todo el código fuente y configuraciones

31 https://calculator.aws/#/

necesarias para el despliegue completo de la solución.

Se puden consultar y descargar todos los recursos del siguiente repositorio público:
https://github.com/ameijeiras/wazuh-uoc

8.3 Líneas de mejora y próximos pasos

En este apartado se enumeran de manera resumida una serie de lineas de mejora por donde se podría continuar evolucionando el proyecto. Al ser un proyecto donde se incluyen todas las etapas de la puesta en marcha de un SIEM, partiendo desde el diseño de la arquitectura base en modalidad cloud, desplegando la solución a través de un orquestador de contenedores para finalmente implementar y definir unos casos de uso a modo de prueba de concepto, existen múltiples lineas de trabajo por donde se podría continuar evolucionando la solución. Haciendo foco en el área de la seguridad, se consideran relevantes las siguiente lineas de trabajo:

8.3.1 Implementación de IDS de red

Para este proyecto se ha implementado un sistema de SIEM para la recolección de eventos de seguridad a través de un sistema de agentes (HIDS). Otro punto donde se puede considerar crucial la recolección de información es a través de la red que interconecta y comunica todos los elementos de una infraestructura. En un entorno actual, el numero de componentes que pueden generar eventos de seguridad es muy heterogéneo y su perímetro a veces difuso por lo que esta solución se complementaría con la integración de un sistema de IDS de red que aportase mayor inteligencia de seguridad gracias a la correlación con eventos y flujos de los datos que circulan por la red.
Para ello, una posible linea de continuación del proyecto se podría enfocar en la implementación de una integración del sistema Wazuh con un IDS de red, añadiendo sobre de la arquitectura propuesta algún sistema como *Suricata*[32] o *OwlH*[33].

8.3.2 Monitorización y auditoria de seguridad en Kubernetes

Como se ha visto, se ha implementado un sistema que es capaz de proteger cargas de trabajo en nube, identificando y generando alertas de seguridad en base a la monitorización de eventos. Un linea de mejora podría ir enfocada a integrar la capacidad de monitorización del plano de gestión del sistema de despliegue nativo de Kubernetes. Dado que este tipo de sistemas de orquestación difuminan enormemente la línea entre infraestructura y desarrollo de aplicaciones, se considera la importancia de poder obtener inteligencia de seguridad y control del entorno a través del monitorización de los eventos de seguridad que se generan dentro del orquestador, como pueden ser, la creación de nuevos volúmenes de datos, creación de secretos, mapas de configuración, exposición de puertos y servicios, por citar algunos de ellos.

8.3.3 Solución Wazuh en cloud multi-proveedor y en entorno distribuido

La arquitectura y solución propuesta se ha diseñado de manera que ofrezca una gran capacidad de resiliencia y escalabilidad. El diseño de la aplicación Wazuh sobre el cloud de Amazon AWS se ha pensado desde un inicio para que todas las cargas de trabajo necesarias para dar soporte a la aplicación se encuentren dentro del mismo proveedor. Además, se ha asumido que todos los agentes que se conectan de manera distribuida al entorno SIEM lo hagan individualmente y de manera autónoma.
Existe una limitación en este modelo que es la dependencia del proveedor o "*vendor lock-in*". Un fallo a gran escala en el proveedor o en un servicio crítico interno, podría suponer un problema en la disponibilidad para toda la solución.
Si además se dispone de otros entornos cloud en otros hyperscalers, es posible que aumente exponencialmente el uso y el número de agentes dentro de la misma infraestructura. Si se dan este tipo de casuísticas, el modelo de agentes distribuido no sería el modelo más optimo ya que el consumo de recursos podría también dispararse al estar los nodos worker distribuidos solo en un mismo proveedor.
Por ello, una posible linea de continuación en el proyecto se podría enfocar en solventar estas limitaciones. Para ello, se podrían explorar la viabilidad de extender la funcionalidad del cluster *Kubernetes* de *Amazon EKS* a otros entorno cloud de otros proveedores a través del uso de una solución como *Amazon EKS Everywhere*[34].

8.4 Conclusiones

A lo largo de este proyecto se han diseñado y validado todas la etapas necesarias para la implementación del SIEM Wazuh, desplegado a través de un sistema de contenedores orquestados desde Kubernetes y sobre uno de los hyperscalers cloud más destacado.
Como se ha podido verificar a lo largo de los apartados previos, se han ido desarrollando de forma exitosa las distintas

32 https://suricata.io/
33 https://www.owlh.net/
34 https://aws.amazon.com/es/eks/eks-anywhere/

tareas e hitos planificados inicialmente sin apenas desviaciones reseñables.

En primer lugar, se han analizado las distintas soluciónes disponibles en el mercado para la implementación de un SIEM. Con los objetivos presentados en un inicio, se opto por la elección del sistema de código abierto Wazuh. Tras ello, se analizó y detallo la solución validando su viabilidad.
A continuación, con las mismas premisas inciales se analizaron los tres proveedores de soluciones nube principales y se selecciono Amazon AWS como provedor para el despliegue de las cargas de trabajo de la capa servidores del sistema Wazuh. En base a las soluciones del proveedor, se diseño la arquitectura y los componentes necesarios para abordar los objetivos del proyecto, para en segunda instancia documentar y configurar todo lo necesario para su puesta en marcha. Finalmente, se definieron una serie de casos de uso para validar la solución a nivel funcional a través de la simulación de ejemplos para verificar el sistema.

Como se ha evidenciado, ha sido posible desarrollar un diseño resiliente y escalable que se implemente en modelo nube, con un coste contenido y que permite el aprovechamiento de todas las ventajas de este modelo.
Se ha demostrado que la implementación del sistema permite mejorar la postura de seguridad y el manejo del riesgo de cualquier organización en donde se plante instalar, gracias a la capacidad de detección y respuesta que aporta. Además, hemos demostrado la capacidad de generar inteligencia de seguridad avanzada a través de las distintas integraciónes de incluye la solución de manera nativa.

En conclusión, se han podido validar por completo los objetivos que se presentaban al inicio del trabajo, gracias a la solución diseñada.

9 Glosario

Correlar	Establecer una relación (o correlación) o correspondencia entre dos o más cosas.
On-premise	El término On-premise o en local se refiere al tipo de instalación de una solución de software. Es el modo de instalación en equipos locales a diferencia del modelo cloud.
SIEM	Gestión de Eventos e Información de Seguridad (del inglés, Security Information and Event Management, SIEM) es un sistema que centraliza la interpretación y el almacenamiento de los eventos deferentes a la seguridad.
EDR	Acrónimo del inglés, Endpoint Detection and Response, es un sistema que proporciona monitorización y análisis continuo del endpoint y la red.
Endpoint	Son todos los puntos de acceso a la empresa desde dispositivos que se conectan virtualmente a su sistema.
Pay-as-you-go	Es la modalidad de pago de uso de un determinado servicio.
Road-Warrior	Usuario o agente con acceso remoto y conexión itinerante.
Hyperscaler	Las empresas que facilitan estos servicios de infraestructura, plataforma y computación de escala masiva en la nube.
OPEX	Acrónimo del inglés "Operational expenditures", es un costo permanente para el funcionamiento de un producto o servicio.
Multi-AZ	Uso de zonas de disponibilidad del proveedor Amazon AWS.
EKS	Servicio de cluster de Kubernetes, Amazon Elastic Kubernetes Service.
EBS	Servicio de almacenamiento de datos, Amazon Elastic Block Storage.
VPC	Servicio de red, Amazon Virtual Private Connection.
SDN	Redes definidas por software.
EC2	Servicio de máquinas virtuales, Amazon Elastic Compute.
CNAME	Tipo de registro del servicio DNS.
IOC	Acrónimo del inglés, Indicator of Compromise, es información relevante que describe cualquier incidente de seguridad, actividad y/o artefacto malicioso, mediante el análisis de sus patrones de comportamiento.
FIM	Servicio de monitorización de integridad de ficheros que incluye Wazuh.

10 Bibliografía

url01: Louis Columbus, CyberSecurityWaste, 2020, https://www.forbes.com/sites/louiscolumbus/2020/08/09/cybersecurity-spending-to-reach-123b-in-2020/?sh=4e804b26705f
url02: Gartner, Gartner Forecasts Worldwide Security, 2021, https://www.gartner.com/en/newsroom/press-releases/2021-05-17-gartner-forecasts-worldwide-security-and-risk-managem
url03: Fortinet, Cybersecurity Statistics, 2020, https://www.fortinet.com/resources/cyberglossary/cybersecurity-statistics
inve01: Incibe, Diseño y Configuración de IPS,IDS y SIEM en Sistemasde Control Industrial, 2017
url06: Wikepedia, Sistemas de detección de intrusos, 2021, https://es.wikipedia.org/wiki/Sistema_de_detección_de_intrusos
url05: A2secure, SIEM, 2019, https://www.a2secure.com/blog/ids-ips-hids-nips-siem-que-es-esto/
url07: dnsstuff, free-siem-tools, 2019, https://www.dnsstuff.com/free-siem-tools
tfm01: Javier Polo Cózar, Implementación de Wazuh en una organización pública, 2020
url04: Wazuh, Wazuh Documentation, 2021, https://documentation.wazuh.com/current/index.html
url13: Kubernetes, Kubernetes Documentation, 2021,
book5: Ed Robinson, Kubernetes on AWS, 2019
url12: Amazon AWS, AWS Well-Architected, 2021, https://aws.amazon.com/es/architecture/well-architected/?wa-lens-whitepapers.sort-by=item.additionalFields.sortDate&wa-lens-whitepapers.sort-order=desc
url11: Amazon AWS, AWS VPC, 2021, https://aws.amazon.com/es/vpc/
url10: Amazon AWS, Aws EKS, 2021, https://aws.amazon.com/es/eks/
url14: Wazuh, Wazuh POC Examples, 2021, https://documentation.wazuh.com/current/proof-of-concept-guide/aws-infrastructure-monitoring.html#poc-aws-monitoring
url15: Wazuh, Wazuh POC Examples, 2021, https://documentation.wazuh.com/current/proof-of-concept-guide/index.html
url16: INCIBE, Riesgos y amenazas en entornos cloud computing, 2015, https://www.incibe.es/extfrontinteco/img/File/intecocert/EstudiosInformes/cert_inf_riesgos_y_amenazas_en_cloud_computing.pdf
url17: Gartner, Inc., Assessing the Security Risks of Cloud Computing, 2008, https://www.gartner.com/en/documents/685308/assessing-the-security-risks-of-cloud-computing
inve02: mcafee.com, Informe sobre amenas 2021, 2021, https://www.mcafee.com/enterprise/es-mx/assets/reports/rp-quarterly-threats-apr-2021.pdf
code01: Fraktal Cyber, , 2021, https://github.com/fraktalcyber/Fransom
est01: MichaelPage, Estudio de remuneración tecnológica 2021, 2021, https://www.michaelpage.es/sites/michaelpage.es/files/estudio_remuneracion_tecnologia_2021.pdf
url18: Wazuh, Wazuh POC Active Response Example, 2021, https://documentation.wazuh.com/current/proof-of-concept-guide/detect-remove-malware-virustotal.html#poc-detect-remove-malware-virustotal

11 Anexos

11.1 Instalación Agentes

11.1.1 Windows

En este apartado se detalla el proceso de instalación del agente Wazuh sobre el sistema operativo Windows.

1. El proceso de instalación se inicia a partir de la descarga del software, en formato Windows installer.
2. A continuación se ejecuta el instalador y se van completando los distintos pasos del mismo.
3. Una vez instalado, se inicia el proceso de configuración a través de una pantalla de configuración.
4. En este punto, es necesario configurar en la dirección "*Manager IP*" la dirección del balanceador de los nodos worker del sistema Wazuh implementado.

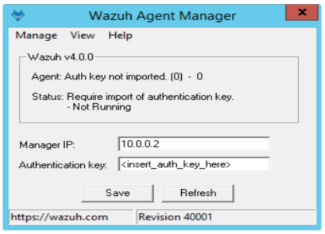


Figura 55: Agente Wazuh Windows

5. Un vez finalizada esta configuración, el agente wazuh se encuentra completamente instalado en el sistema y listo para su ejecución. El sistema de instala como un servicio Windows y se configura automáticamente para arrancar con sistema operativo.

Se puede encontrar el detalle completo de la instalación y la documentación completa a través del siguiente enlace.

11.1.2 Linux Ubuntu 20.04

En este apartado se detallará el proceso de instalación del agente Wazuh sobre el sistema operativo Linux Ubuntu 20.04 LTS.

1. El proceso de instalación se inicia a partir de la importación de la clave GPG del fabricante en el sistema:

```
$ curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | apt-key add -
```

2. A continuación se añade el repositorio de software al sistema y se actualizan los paquetes:

```
$ echo "deb https://packages.wazuh.com/4.x/apt/ stable main" | tee -a
/etc/apt/sources.list.d/wazuh.list
$ apt-get update
```

3. Para finalmente instalar el servicio de aplicación wazuh a través del siguiente comando, ajustado en la variable "WAZUH_MANAGER" la dirección del balanceador de los nodos worker del sistema Wazuh implementado.

```
$ WAZUH_MANAGER="wazuh.manager.ameijeiras.es" apt-get install wazuh-agent
```

4. A partir de este punto el agente Wazuh Linux se encuentra complemente instalado. Finalizaremos el proceso, configurando el servicio y programando su inicio automático con el sistema:

```
$ systemctl daemon-reload
$ systemctl enable wazuh-agent
$ systemctl start wazuh-agent
```

Se puede encontrar el detalle completo del proceso de instalación y la documentación completa a través del siguiente enlace.

11.1.3 Amazon Linux

En este apartado se detalla el proceso de instalación del agente Wazuh sobre el sistema operativo Amazon Linux que se emplea dentro de los agentes de los nodos de trabajo del cluster Kubernetes donde se despliegua la solución Wazuh y que en han empleado dentro de los casos de uso.

1. El proceso de instalación se inicia a partir de la importación de la clave GPG del fabricante en el sistema:

```
$ rpm --import https://packages.wazuh.com/key/GPG-KEY-WAZUH
```

2. A continuación se añade el repositorio de software al sistema y se actualizan los paquetes:

```
$ cat > /etc/yum.repos.d/wazuh.repo << EOF
[wazuh]
```

```
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
enabled=1
name=EL-\$releasever - Wazuh
baseurl=https://packages.wazuh.com/4.x/yum/
protect=1
EOF
```

3. Para finalmente instalar el servicio de aplicación wazuh a través del siguiente comando, ajustado en la variable "WAZUH_MANAGER" la dirección del balanceador de los nodos worker del sistema Wazuh implementado.

```
$ WAZUH_MANAGER="wazuh.manager.ameijeiras.es" yum install wazuh-agent
```

4. A partir de este punto el agente Wazuh Linux se encuentra complemente instalado. Finalizaremos el proceso, configurando el servicio y programando su inicio automático con el sistema:

```
$ systemctl daemon-reload
$ systemctl enable wazuh-agent
$ systemctl start wazuh-agent
```

Se puede encontrar el detalle completo del proceso de instalación y la documentación completa a través del siguiente enlace.

11.1.4 Otras plataformas

Se puede encontrar el detalle completo de la instalación del los agentes sobre múltiples plataformas a través del siguiente enlace.

11.2 Instalación del cliente kubectl

11.2.1 Windows

En este apartado se detallarán los pasos para la instalación del cliente Kubernetes *kubectl* sobre el sistema operativo Windows.
Los pasos a realizar para desplegar este cliente son los siguientes:

1. Descargar la versión v1.22.0 para Windows a través del siguiente enlace.
2. Añadir el ejecutable descargado a las rutas del sistema a través de la variable de entorno Windows "*PATH*".
3. En este instante el cliente Kubernetes esta listo para su uso.

Se puede encontrar el detalle completo de la instalación y la documentación completa a través del siguiente enlace.

11.2.2 Linux

En este apartado se detallarán los pasos para la instalación del cliente Kubernetes *kubectl* sobre un sistema operativo Linux.
Los pasos a realizar para desplegar este cliente sobre el sistema operativo Ubuntu 20.04 LTS, son los siguientes:

1. Se prepara el sistema de repositorios para la instalación del software.

```
$ sudo apt-get update
$ sudo apt-get install -y apt-transport-https ca-certificates curl
```

2. Se añade la clave GPG de Kubernetes.

```
$ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

3. Se añade el repositorio al sistema.

```
$ echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

4. Se instala el paquete de la última versión del software.

```
$ sudo apt-get update
$ sudo apt-get install -y kubectl
```

5. En este instante el cliente Kubernetes ya esta listo para su uso.

Se puede encontrar el detalle completo de la instalación y la documentación completa a través del siguiente enlace.

11.3 Instalación de las herramientas CLI de Amazon AWS

11.3.1 Windows

En este apartado se detallarán los pasos para la instalación del cliente de administración de Amazon AWS, *AWS cli*, sobre el sistema operativo Windows.
Se puede encontrar el detalle completo de la instalación y la documentación completa a través del siguiente enlace.

11.3.2 Linux

En este apartado se detallarán los pasos para la instalación del cliente de administración de Amazon AWS, *AWS cli*, sobre el sistema operativo Windows.
Se puede encontrar el detalle completo de la instalación y la documentación completa a través del siguiente enlace.

11.4 Puertos

A continuación se detalla una tabla con la información de los puertos de cada uno de los componentes de la arquitectura de la plataforma Wazuh, así como su propósito:

Component	Software	Port	Protocol	Purpose
Wazuh server	Wazuh manager	1514	TCP (default)	Agents connection service
		1514	UDP	Agents connection service
		1515	TCP	Agents registration service
		1516	TCP	Wazuh cluster daemon
		514	UDP (default)	Wazuh syslog collector (disabled by default)
		514	TCP	Wazuh syslog collector (disabled by default)
	Wazuh API	55000	TCP	Wazuh RESTful API
Elastic Stack	Elasticsearch	9200	TCP	Elasticsearch RESTful API
		9300-9400	TCP	Elasticsearch cluster communication
	Kibana	443	TCP	Kibana web interface

11.5 Código desarrollado para respuesta activa (*delete_and_quarantine.sh*)

Se detalla el código empleado en el fichero */var/ossec/active-response/bin/delete_and_quarantine.sh* para la gestión de respuesta activa que se ha desarrollado en el proyecto. El código, esta basado una plantilla de ejemplo que ofrece Wazuh, y sobre el mismo se han desarrollado los cambios necesarios para que se adaptase a la nueva función que se quería implementar{url18}. También se ha incluido en el repositorio de ficheros de código asociado a mismo (https://github.com/ameijeiras/wazuh-uoc).

```
#!/bin/bash

LOCAL=`dirname $0`;
cd $LOCAL
cd ../


PWD=`pwd`

read INPUT_JSON
FILENAME=$(echo $INPUT_JSON | jq -r .parameters.alert.data.virustotal.source.file)
COMMAND=$(echo $INPUT_JSON | jq -r .command)
LOG_FILE="${PWD}/../logs/active-responses.log"
PASSWORD="SomePassword"
QUARANTINE="/var/ossec/quarantine"


if [ ${COMMAND} = "add" ]
then
 # Send control message to execd
```

```
 printf '{"version":1,"origin":{"name":"remove-threat","module":"active-
response"},"command":"check_keys", "parameters":{"keys":[]}}\n'

 read RESPONSE
 COMMAND2=$(echo $RESPONSE | jq -r .command)
 if [ ${COMMAND2} != "continue" ]
 then
  echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Moviendo amenaza a cuarentena" >> ${LOG_FILE}
  exit 0;
 fi
fi

# Se mueve la amenaza al directorio de cuarentena que despues se recogera para analisis en honeypod
y se elimina
NOW=$(date '+%Y%m%d_%H%M%S')
#echo "##DEBUG--> /usr/bin/zip -q --password $PASSWORD $QUARANTINE/${FILENAME##*/}_$NOW.zip
$FILENAME" >> ${LOG_FILE}
/usr/bin/zip -q --password $PASSWORD $QUARANTINE/${FILENAME##*/}_$NOW.zip $FILENAME && rm -f
$FILENAME && rm -f $FILENAME
if [ $? -eq 0 ]; then
 echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Se ha trasladado la amenaza a cuarentena
correctamente" >> ${LOG_FILE}
else
 echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Error moviendo amenaza a la cuarentena" >> $
{LOG_FILE}
fi

exit 0;
```

11.6 Detalle de características de las instancias “T3”

Mac T4g T3 T3a T2 M6g M6i M5 M5a M5n M5zn M4 A1

Las instancias T3 son de última generación, ampliables y de uso general y proporcionan un nivel básico de rendimiento de la CPU con posibilidad de ampliar el uso de la CPU en cualquier momento durante el tiempo que sea necesario. Las instancias T3 ofrecen un equilibrio entre recursos informáticos, de memoria y de red, y están diseñadas para aplicaciones con un uso moderado de la CPU que experimentan picos temporales de uso.

Las instancias T3 acumulan créditos de CPU cuando una carga de trabajo opera por debajo del umbral base. Cada crédito de CPU permite a la instancia T3 la oportunidad de aumentar el rendimiento del núcleo de una CPU durante un minuto cuando sea necesario. Las instancias de T3 pueden ampliarse en cualquier momento durante el tiempo que sea necesario en el modo Ilimitado.

Características:

- CPU ampliable, que se rige por créditos de CPU y rendimiento base constante
- Modo ilimitado de forma predeterminada para garantizar el rendimiento durante los picos de uso y la opción de modo estándar para obtener un costo mensual predecible
- Con tecnología del sistema Nitro de AWS, una combinación de hardware dedicado e hipervisor ligero
- El sistema Nitro de AWS y los procesadores escalables Intel Xeon de alta frecuencia dan como resultado una mejora de la relación entre el precio y el rendimiento de hasta el 30 % en comparación con las instancias T2

Instancia	CPU virtual*	Créditos por hora de CPU	Memoria (GiB)	Almacenamiento	Rendimiento de red (Gbps)***
t3.nano	2	6	0,5	Solo EBS	Hasta 5
t3.micro	2	12	1	Solo EBS	Hasta 5
t3.small	2	24	2	Solo EBS	Hasta 5
t3.medium	2	24	4	Solo EBS	Hasta 5
t3.large	2	36	8	Solo EBS	Hasta 5
t3.xlarge	4	96	16	Solo EBS	Hasta 5
t3.2xlarge	8	192	32	Solo EBS	Hasta 5

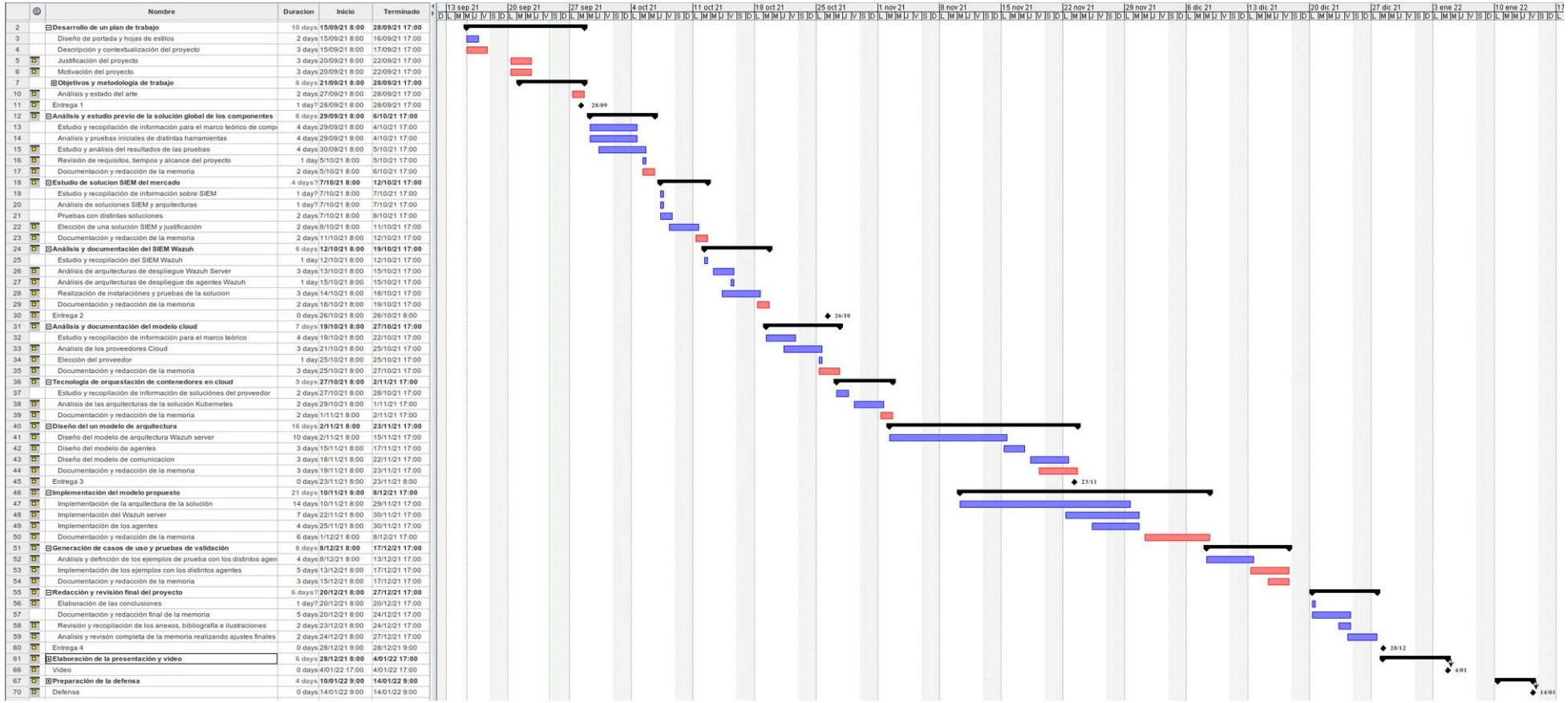
Todas las instancias tienen las siguientes especificaciones:

- Procesador Intel Xeon Platinum de hasta 3,1 GHz
- Intel AVX†, Intel AVX2†, Intel Turbo
- Optimizadas para EBS
- Redes mejoradas†

Casos de uso:

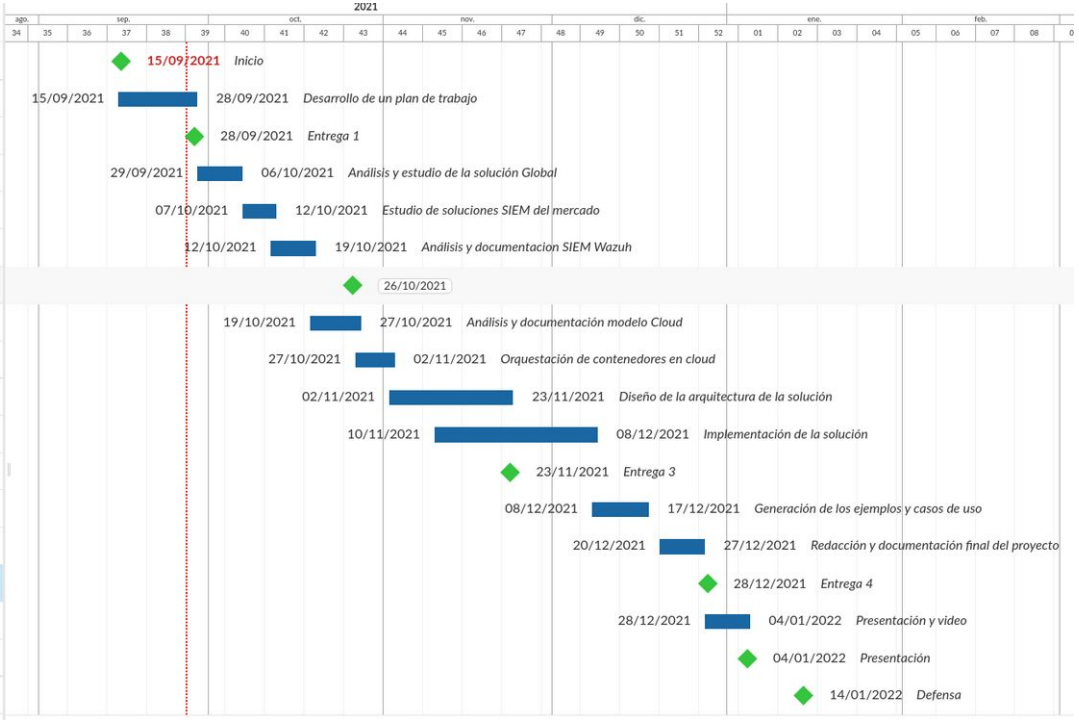
Microservicios, aplicaciones interactivas con baja latencia, bases de datos de tamaño pequeño y mediano, escritorios virtuales, entornos de desarrollo, repositorios de código y aplicaciones críticas para la empresa

11.7 Diagramas de planificación del proyecto



	Nombre	Duracion	Inicio	Terminado
2	**Desarrollo de un plan de trabajo**	10 days	**15/09/21 8:00**	**28/09/21 17:00**
3	Diseño de portada y hojas de estilos	2 days	15/09/21 8:00	16/09/21 17:00
4	Descripción y contextualización del proyecto	3 days	15/09/21 8:00	17/09/21 17:00
5	Justificación del proyecto	3 days	20/09/21 8:00	22/09/21 17:00
6	Motivación del proyecto	3 days	20/09/21 8:00	22/09/21 17:00
7	**Objetivos y metodología de trabajo**	6 days	**21/09/21 8:00**	**28/09/21 17:00**
10	Análisis y estado del arte	2 days	27/09/21 8:00	28/09/21 17:00
11	Entrega 1	1 day?	28/09/21 8:00	28/09/21 17:00
12	**Análisis y estudio previo de la solución global de los componentes**	6 days	**29/09/21 8:00**	**6/10/21 17:00**
13	Estudio y recopilación de información para el marco teórico de compo	4 days	29/09/21 8:00	4/10/21 17:00
14	Analisis y pruebas iniciales de distintas harramientas	4 days	29/09/21 8:00	4/10/21 17:00
15	Estudio y análisis del resultados de las pruebas	4 days	30/09/21 8:00	5/10/21 17:00
16	Revisión de requisitos, tiempos y alcance del proyecto	1 day	5/10/21 8:00	5/10/21 17:00
17	Documentación y redacción de la memoria	2 days	5/10/21 8:00	6/10/21 17:00
18	**Estudio de solucion SIEM del mercado**	4 days?	**7/10/21 8:00**	**12/10/21 17:00**
19	Estudio y recopilación de información sobre SIEM	1 day?	7/10/21 8:00	7/10/21 17:00
20	Análisis de soluciones SIEM y arquitecturas	1 day?	7/10/21 8:00	7/10/21 17:00
21	Pruebas con distintas soluciones	2 days	7/10/21 8:00	8/10/21 17:00
22	Elección de una solución SIEM y justificación	2 days	8/10/21 8:00	11/10/21 17:00
23	Documentación y redacción de la memoria	2 days	11/10/21 8:00	12/10/21 17:00
24	**Análisis y documentación del SIEM Wazuh**	6 days	**12/10/21 8:00**	**19/10/21 17:00**
25	Estudio y recopilación del SIEM Wazuh	1 day	12/10/21 8:00	12/10/21 17:00
26	Análisis de arquitecturas de despliegue Wazuh Server	3 days	13/10/21 8:00	15/10/21 17:00
27	Análisis de arquitecturas de despliegue de agentes Wazuh	1 day	15/10/21 8:00	15/10/21 17:00
28	Realización de instalaciónes y pruebas de la solucion	3 days	14/10/21 8:00	18/10/21 17:00
29	Documentación y redacción de la memoria	2 days	18/10/21 8:00	19/10/21 17:00
30	Entrega 2	0 days	26/10/21 8:00	26/10/21 8:00
31	**Análisis y documentación del modelo cloud**	7 days	**19/10/21 8:00**	**27/10/21 17:00**
32	Estudio y recopilación de información para el marco teórico	4 days	19/10/21 8:00	22/10/21 17:00
33	Analisis de los proveedores Cloud	3 days	21/10/21 8:00	25/10/21 17:00
34	Elección del proveedor	1 day	25/10/21 8:00	25/10/21 17:00
35	Documentación y redacción de la memoria	3 days	25/10/21 8:00	27/10/21 17:00
36	**Tecnología de orquestación de contenedores en cloud**	5 days	**27/10/21 8:00**	**2/11/21 17:00**
37	Estudio y recopilación de información de soluciónes del proveedor	2 days	27/10/21 8:00	28/10/21 17:00
38	Análisis de las arquitecturas de la solución Kubernetes	2 days	29/10/21 8:00	1/11/21 17:00
39	Documentación y redacción de la memoria	2 days	1/11/21 8:00	2/11/21 17:00
40	**Diseño del un modelo de arquitectura**	16 days	**2/11/21 8:00**	**23/11/21 17:00**
41	Diseño del modelo de arquitectura Wazuh server	10 days	2/11/21 8:00	15/11/21 17:00
42	Diseño del modelo de agentes	3 days	15/11/21 8:00	17/11/21 17:00
43	Diseño del modelo de comunicacion	3 days	18/11/21 8:00	22/11/21 17:00
44	Documentación y redacción de la memoria	3 days	19/11/21 8:00	23/11/21 17:00
45	Entrega 3	0 days	23/11/21 8:00	23/11/21 8:00
46	**Implementación del modelo propuesto**	21 days	**10/11/21 8:00**	**8/12/21 17:00**
47	Implementación de la arquitectura de la solución	14 days	10/11/21 8:00	29/11/21 17:00
48	Implementación del Wazuh server	7 days	22/11/21 8:00	30/11/21 17:00
49	Implementación de los agentes	4 days	25/11/21 8:00	30/11/21 17:00
50	Documentación y redacción de la memoria	6 days	1/12/21 8:00	8/12/21 17:00
51	**Generación de casos de uso y pruebas de validación**	8 days	**8/12/21 8:00**	**17/12/21 17:00**
52	Análisis y definción de los ejemplos de prueba con los distintos agen	4 days	8/12/21 8:00	13/12/21 17:00
53	Implementación de los ejemplos con los distintos agentes	5 days	13/12/21 8:00	17/12/21 17:00
54	Documentación y redacción de la memoria	3 days	15/12/21 8:00	17/12/21 17:00
55	**Redacción y revisión final del proyecto**	6 days?	**20/12/21 8:00**	**27/12/21 17:00**
56	Elaboración de las conclusiones	1 day?	20/12/21 8:00	20/12/21 17:00
57	Documentación y redacción final de la memoria	5 days	20/12/21 8:00	24/12/21 17:00
58	Revisión y recopilación de los anexos, bibliografía e ilustraciones	2 days	23/12/21 8:00	24/12/21 17:00
59	Analisis y revisón completa de la memoria realizando ajustes finales	2 days	24/12/21 8:00	27/12/21 17:00
60	Entrega 4	0 days	28/12/21 9:00	28/12/21 9:00
61	**Elaboración de la presentación y video**	6 days	**28/12/21 8:00**	**4/01/22 17:00**
66	Video	0 days	4/01/22 17:00	4/01/22 17:00
67	**Preparación de la defensa**	4 days	**10/01/22 9:00**	**14/01/22 9:00**
70	Defensa	0 days	14/01/22 9:00	14/01/22 9:00

ID	TIPO	ASUNTO	ESTADO	FECHA DE INICIO	FECHA DE FINALIZACIÓN
36	MILESTONE	Inicio	New	15/09/2021	15/09/2021
48	TASK	Desarrollo de un plan de trabajo	New	15/09/2021	28/09/2021
37	MILESTONE	Entrega 1	New	28/09/2021	28/09/2021
49	TASK	Análisis y estudio de la solución Global	New	29/09/2021	06/10/2021
50	TASK	Estudio de soluciones SIEM del mercado	New	07/10/2021	12/10/2021
51	TASK	Análisis y documentacion SIEM Wazuh	New	12/10/2021	19/10/2021
43	MILESTONE	Entrega 2	New	26/10/2021	26/10/2021
52	TASK	Análisis y documentación modelo Cloud	New	19/10/2021	27/10/2021
53	TASK	Orquestación de contenedores en cloud	New	27/10/2021	02/11/2021
54	TASK	Diseño de la arquitectura de la solución	New	02/11/2021	23/11/2021
55	TASK	Implementación de la solución	New	10/11/2021	08/12/2021
59	MILESTONE	Entrega 3	New	23/11/2021	23/11/2021
56	TASK	Generación de los ejemplos y casos de u...	New	08/12/2021	17/12/2021
57	TASK	Redacción y documentación final del pro...	New	20/12/2021	27/12/2021
60	MILESTONE	Entrega 4	New	28/12/2021	28/12/2021
58	TASK	Presentación y video	New	28/12/2021	04/01/2022
61	MILESTONE	Presentación	New	04/01/2022	04/01/2022
62	MILESTONE	Defensa	New	14/01/2022	14/01/2022



11.8 Diagrama de arquitectura