Facultad
de
Ciencias

INTEGRACIÓN DE UN CENTRO DE OPERACIONES DE SEGURIDAD CON COMPONENTES DE CÓDIGO ABIERTO

(Security Operations Center integration with Open Source components)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Alejandro Benítez Tocino

Director: Esteban Stafford Fernández

Diciembre - 2022

Tabla de contenido

Índice de figuras

Resumen

Este Trabajo de Fin de Grado ofrece una solución completa, organizada y práctica con la que poder aumentar la seguridad interna de una red de equipos. Se propone la implementación de un Centro de Operaciones de Seguridad ("*Security Operations Center*" en inglés) únicamente a partir de servicios de código abierto "Open Source".

Es una propuesta completa, ya que trata de abordar el proyecto simulando una situación real en la que este trabajo se presenta como mejora en una empresa. Esto implica cubrir una serie de aspectos que son necesarios para la evaluación y consideración de este. Pese a que no es de una magnitud muy grande, se ha intentado acercar lo máximo posible a esta idea incluyendo los siguientes desarrollos: un estudio y descomposición del producto, el despliegue del hardware y una estimación del gasto económico. Además, en el anexo se incluye una guía de implementación detallada paso a paso con su despliegue inmediato.

El campo de la ciberseguridad es muy extenso, y es necesario conocer los tipos de herramientas que se emplean en el trabajo de esta disciplina. Por ello, en primer lugar, se hace un repaso de los términos y conceptos que el lector debe conocer para el entendimiento del proyecto. Esto precede a una descripción de los sistemas que forman el SOC, con su definición y propósito. A continuación, se revisan los requisitos funcionales del sistema, para así dar pie a la muestra del flujo final de trabajo. Esto sirve para dar una idea superficial de la dinámica que sigue este centro de seguridad. Tras esto, se sigue con una explicación de cada uno de los softwares seleccionados, sus características y rol dentro del sistema.

Vista la parte teórica, se procede con la sección dedicada a la implementación física y evaluación de costes. Para el despliegue hardware se analizan primero los requisitos no funcionales, que se basan en las características de las instalaciones propuestas para el despliegue del SOC, los laboratorios de la Facultad de Ciencias. Con referencia al estudio de requisitos, se muestran dos opciones posibles para dicha instalación, que son el despliegue físico y el alquiler de servidores remotos o "*hosting*". Finalmente, como anotación o toma de contacto con el mundo real, se ve una estimación de los costes que puede tener la incorporación de un sistema de estas especificaciones.

Palabras clave: Ciberseguridad, centro de operaciones de seguridad, código abierto, administración de sistemas, implementación.

Abstract

This Final Degree Project shows a practical, organised and complete solution which upgrades any enterprise network internal security. It is proposed the implementation of a Security Operations Center, commonly called “SOC”, only made up of open source components.

It is a complete proposal, as it simulates a real scenario in which this work is presented as an improvement in a company. This implies covering a series of necessary aspects for the evaluation and consideration. Besides of the project’s magnitude which does not go that far, we have tried to get as close as possible to this “real-project” idea including the following researchings: a product’s study and decomposition, the hardware deployment, followed by a rough economic estimation. Furthermore, the appendix includes a complete installation guide, with step-by-step instructions to implement the SOC.

The field of cybersecurity is very extensive, so it is necessary to know the type of tools are used whitin this discipline. Therefore, first of all, the terms and concepts are reviewed so the reader can know and understand the project. This means a description of the systems that make up the SOC, along with their definition and purpose. Next, the functional requirements are reviewed, thus giving rise to the sample of the final workflow that assembles the system. This serves to give a superficial idea of the dynamics of this security center. Then, we make an explanation of each of the selected applications, their characteristics and role within the system.

Following the technical description of the SOC, we proceed with the section dedicated to the physical implementation and cost evaluation. For the hardware deployment, we first analyze the non-functional requirements, which are based on the characteristics of the facilities proposed for the deployment of the SOC, the laboratories of the Faculty of Science. Refering to the pre-requisite study, two possible options for such installation are shown, which are the physical deployment and hosting option. Finally, as a real annotation, I show an estimation of the costs of incorporating a system with these specifications.

Keywords: Cybersecurity, Security Operations Center, open source, system adminsitration, implementation.

CAPÍTULO I.

INTRODUCCIÓN

Este Trabajo de Fin de Grado se ha elaborado como continuación y mejora de las prácticas que se realizaron durante el 3º curso en la empresa Osane Consulting SL. Las prácticas consistieron en la creación de un sistema de monitorización de red para empresas del sector industrial. Tuvieron una ocupación de 6 créditos académicos, es decir, 150 horas de trabajo. La tarea encomendada era desbordante para el tiempo y conocimientos que se disponían, por lo que el proyecto no se consiguió terminar.

Tomando como base la memoria incluida en el ANEXO I, el proyecto mostrado en este TFG ha sido elaborado desde cero y forma el desarrollo final de aquella versión. Sin embargo, contiene diferencias y aborda el sistema desde una perspectiva diferente. La implementación que se sugiere aquí va dirigida a la monitorización de equipos finales en la Facultad de Ciencias y no considera algunos de los requisitos iniciales de las prácticas. Por ejemplo, se lleva a cabo en dos máquinas en vez de tres y Docker no se emplea. Además de contener la instalación y montado del sistema en base a unos requisitos ligeramente diferentes, se incluye un análisis más detallado sobre lo que supondría trasladar el desarrollo al mundo real.

1.1. Motivación

Durante los últimos 30 años, el uso de la informática ha crecido drásticamente. La gestión de cientos de tareas que antes las realizaba un humano, ahora son gestionadas por un programa informático. Incluso se puede dar el caso de que toda la información vital de una empresa esté almacenada en un pequeño disco duro. Esto ha generado que, a diario, con el conglomerado de tráfico sensible entre empresas, muchas personas vean oportunidades para sacar provecho o enriquecerse ilícitamente. Por ello, se deduce una conclusión: cualquier entidad, ya sea pequeña o grande, necesita de manera urgente medidas para garantizar la seguridad de sus datos.

Asimismo, han ido generándose diferentes maneras de efectuar la securización en los sistemas de información. Principalmente son dos: las medidas activas y las pasivas. Las primeras son las que tratan de evitar que los equipos informáticos se vean infectados por algún actor malicioso, como por ejemplo el uso de contraseñas seguras, antivirus, cortafuegos, la encriptación de datos o las auditorías de seguridad. Por otro lado, están las medidas pasivas, que toman parte una vez se abre una brecha a causa de un ataque. El haber implementado medidas de seguridad pasivas puede solucionar una parte del problema, ya que ofrecen soluciones para mitigar el efecto de los ataques. Son por ejemplo las copias de seguridad, el uso de la nube, ajustes del sistema operativo, características propias del hardware, etcétera.

Normalmente, las empresas cubren de manera exitosa las medidas pasivas, pero no es suficiente. Según los informes de CyberEdge sobre las amenazas cibernéticas, el 85% de las organizaciones sufrió un ataque exitoso el año pasado [1]. Es por eso que, las organizaciones necesitan mejorar su despliegue y reforzar sus medidas activas.

En cambio, la visión empresarial nunca elude el aspecto que más suele preocupar en una decisión: el coste. El coste de las mejoras es un gran obstáculo para desarrollar cualquier ampliación o proyecto. Por eso, los programas de código abierto "Open Source", son la clara solución para este problema de presupuestos al que tantas pequeñas y medianas empresas se enfrentan diariamente.

La fusión de estos dos motivos forma el embrión que da vida a este proyecto: proporcionar una medida de seguridad activa que vigila, previene y en caso excepcional, mitiga intrusiones. Todo esto de manera eficiente y gratuita. Este sistema de defensa activa se denomina Centro de Operaciones de Seguridad o SOC ("*Security Operations Center*"), y es el responsable de incorporar a cualquier sistema la posibilidad de monitorear fácilmente los equipos finales de una organización en busca de una actividad anómala y en caso necesario, efectuar un análisis o investigación de los indicios basada en información respaldada.

1.2. Objetivos del proyecto

El objetivo claro de este trabajo es mostrar un análisis exhaustivo y la implementación de una solución perfectamente viable y gratuita con la que cualquier organización pueda mejorar su despliegue e infraestructura de seguridad informática. Plantear una respuesta al problema planteado: la monitorización y securización de un despliegue de equipos para prevenir y mitigar ataques o actividad indeseada.

Para la solución de este problema, el trabajo se centra en la creación de un SOC enteramente con software libre. Este término, hace referencia al conjunto formado por las personas y herramientas que se emplean para mantener y securizar los sistemas de información (véase en más detalle más adelante). El SOC planteado en este TFG, debe cumplir los siguientes objetivos generales:

- Proporcionar mecanismos para la monitorización de registros de logs y de red en equipos finales, en busca de eventos que se tengan registrados en la base de datos del servidor como sospechosos o de notable importancia.
- Presentar un interfaz de usuario para la gestión de alertas.
- Proporcionar mecanismos para el análisis de datos referentes a alertas.
- Incorporar mecanismos para la respuesta activa a incidentes.

La idea trata de poner en marcha un sistema con el que podamos tener un flujo de trabajo "semi-automatizado" que abarque desde el registro de una alerta en alguno de los equipos, hasta la mitigación o eliminación de la alarma, pasando por su estudio y evaluación por parte de los analistas.

Para poner a prueba el sistema descrito, la implementación toma como referencia una parte de la red de equipos de los laboratorios docentes de la Facultad de Ciencias. Por lo

tanto, las dimensiones y requisitos no funcionales se basan en las cualidades y actividad que dichos laboratorios generarían en un caso real.

Por otro lado, este sistema requiere de una plataforma sobre la que ejecutarse, un despliegue de servidores físicos. Por ello, como detalle final, se presenta un análisis de prestaciones y económico de los equipos necesarios para su funcionamiento y así poder prestar servicio a todos los equipos finales de las aulas y laboratorios de la Facultad de Ciencias.

1.3. Estructura del documento

En primer lugar, este documento expone y desarrolla un pequeño glosario con los términos técnicos utilizados frecuentemente en este trabajo, para así poder introducir el diseño elegido en la implementación del SOC. En este diseño, se hace una revisión de los requisitos funcionales del sistema acompañado de una breve descripción del flujo de trabajo donde se ven todas las piezas en funcionamiento. Asimismo, se repasa más detenidamente cada uno de los componentes: la arquitectura de estos, sus propósitos y su motivo en el sistema.

A continuación, se hace un pequeño estudio del despliegue físico donde el sistema se ejecuta. En función de los requisitos no funcionales de nuestro sistema, y las recomendaciones de los desarrolladores de cada componente, se ve cómo pueden combinarse y qué especificaciones son necesarias para su implementación. Una vez establecido, se proponen modelos reales de servidores con el que se pueda comenzar a estimar costes y plantear la viabilidad económica del proyecto.

Tras esta revisión detallada sobre los componentes y su despliegue, se realiza un resumen de costes, comparando la opción de despliegue físico y el alquiler de servidores remotos. Se incluye además un balance general en el que se comparan las propuestas.

Finalmente, en el anexo se detalla el proceso de implementación total del SOC.

CAPÍTULO II.

ESTUDIO DE LOS CENTROS DE OPERACIONES DE SEGURIDAD

En el desarrollo de este documento, se hace referencia a múltiples términos y su comprensión es muy relevante. Por eso, se dispone un desarrollo sobre cada uno de ellos, para procurar el mejor entendimiento posible a lo largo de la explicación del proyecto. Los términos son: **SOC**, **SIEM**, **XDR**, **IRP** y **Sistemas de Detección de Intrusiones**.

2.1. SOC

Las organizaciones, durante mucho tiempo, han lidiado con el problema de la seguridad informática de diversas maneras: incorporando cortafuegos, software y herramientas antimalware, etcétera. Con esto, las pequeñas corporaciones en las que cuentan con poca logística o incluso un solo servidor, habrá sido suficiente. Sin embargo, las grandes empresas cuya magnitud y despliegue son tan grandes, no pueden arriesgarse a una simple barrera o programa antimalware. Surge así el concepto formal de Centro de Operaciones de Seguridad o SOC [2].

2.1.1. Definición

Un Centro de Operaciones de Seguridad, principalmente es un conjunto centralizado de personas, procesos y tecnología que trabajan para proteger los sistemas y redes de una organización a través de la monitorización, detección, prevención y análisis continuos de las amenazas cibernéticas [2]. Por eso, es normal encontrar referencias al término “SOC” de dos maneras diferentes:

- SOC, haciendo referencia a la **plataforma** que permite la supervisión y administración de la seguridad de un sistema de información a través de herramientas de recogida, correlación de eventos e intervención remota [3].
- SOC, con referencia al **equipo de personas** que operan con estas herramientas.

Hay muchos roles dentro de la organización de un equipo SOC, y aunque depende mucho de la magnitud y propósito de la organización que lo implemente, principalmente suelen estar formados por [2]:

- **Gerentes de SOC:** supervisan y administran el equipo, crean procesos y medidas; y evalúan la respuesta a incidentes y el aseguramiento de la calidad, las soluciones de seguridad presupuestarias.

- **Analistas de seguridad:** detectan y analizan amenazas y responden a ellas rápidamente. A menudo trabajan para implementar medidas de seguridad según las instrucciones de la gerencia.
- **Ingenieros de seguridad**, cuya función es manejar las soluciones y herramientas de seguridad que se utilizan: las mantienen, encuentran nuevas herramientas y crean y mantienen la arquitectura en los sistemas de seguridad.

2.1.2. Objetivos

El objetivo de un SOC es detectar, analizar y corregir incidentes de ciberseguridad utilizando soluciones tecnológicas y enfoques diferentes. Supervisan y analizan la actividad en redes, servidores, terminales, bases de datos, aplicaciones, sitios web y otros sistemas en busca de comportamientos anormales que puedan indicar un incidente de seguridad o un compromiso. Además, se debe garantizar que los posibles incidentes de seguridad sean identificados, analizados, defendidos, investigados y documentados adecuadamente [3].

Se puede concluir que los tres pilares para la implementación de un SOC son los siguientes [4]:

- **Gestión de eventos**, lo cual implica la administración de activos de seguridad, monitoreo de eventos, alertas y categorización de estos.
- **Respuesta** a incidentes, análisis y cotejo de información con distintas fuentes, análisis y determinación del estado de los sistemas críticos y recomendaciones para remediar.
- **Ciberdefensa**, trabajo de un equipo de expertos en ciberseguridad para la mitigación y resolución de los incidentes y cacería de posibles incidentes presentes o futuros.

2.1.3. Beneficios

Visto lo anterior, la implementación de un SOC en una organización es una clara ventaja, pero podemos resumirlo en los siguientes beneficios descritos [2]:

- **Monitorización continua del sistema de información y su actividad.** Esto provoca una respuesta a incidentes bastante mejorada, puesto que gestionar el sistema de manera ininterrumpida implica que el tiempo que pasa desde el momento de la detección a la respuesta sea mínimo.

- **Análisis eficiente.** Dicho de manera coloquial, “encontrar la aguja del pajar”. Los centros de operaciones de seguridad brindan una descripción general de todas las redes e infraestructuras de la organización, además de todas las debilidades potenciales que pueden provenir de partes de la superficie de ataque que normalmente no monitorizaría.

- **Capacidad de fijar prioridades.** Al introducir los datos de inteligencia sobre amenazas en sus herramientas de seguridad, los SOC pueden diferenciar entre

amenazas reales y no tan reales y, en función de eso, priorizar la estrategia y la respuesta.

- **Facilidad en la investigación de incidentes.** En el momento que llega el incidente, la investigación puede realizarse rápidamente gracias a la información detallada sobre los datos de seguridad que proporciona un SOC a diario.

- **Costes reducidos.** A largo plazo, es más asequible emplear un equipo de expertos, que brinda un control total de los sistemas y redes. Esto reducirá la gravedad de las propias filtraciones de datos y su coste.

2.2. SIEM

Dentro de un SOC, existen diversas herramientas empleadas para la gestión y análisis de alertas. Entre ellas, se pueden diferenciar varias categorías de software. En este caso, el SIEM concentra el software que tiene como objetivo otorgar a las organizaciones información útil sobre potenciales amenazas de seguridad de sus redes críticas de negocio, a través de la estandarización de datos y priorización de amenazas. Esto es posible mediante un análisis centralizado de datos de seguridad, obtenidos desde múltiples sistemas, que incluyen aplicaciones antivirus, firewalls y soluciones de prevención de intrusiones [5].

2.2.1. Definición

En un reporte de 2005 titulado "*Improve IT Security With Vulnerability Management*" [6], Gartner acuñó el término de **Gestión de Eventos e Información de Seguridad** o SIEM ("*Security Information and Event Management*" en inglés). El término reúne los conceptos de **Gestión de Eventos de Seguridad** o SEM ("*Security Event Management*" en inglés) con el de **Gestión de Información de Seguridad** o SIM ("*Security Information Management*" en inglés), para obtener lo mejor de ambos mundos [5].

- **SEM:** Cubre la monitorización y correlación de eventos en tiempo real, al mismo tiempo que alerta la configuración y vistas de consola relacionadas con esas actividades.
- **SIM:** Lleva estos datos a una siguiente fase que incluye el almacenamiento, análisis y generación de reportes de los resultados.

2.2.2. Funcionalidad

Partiendo de la premisa de que la cantidad de eventos y magnitud de la información que generan los sistemas de protección es muy alta, surge la dificultad de interpretar la información en su totalidad para poder reconocer los problemas reales. La cantidad de datos que pueden llegar a ser es muy alta, y para las organizaciones ya no es posible hacer este análisis en forma manual. Es aquí donde aparecen los software SIEM.

Con un SIEM, el equipo de seguridad cuenta con un método efectivo para automatizar sus procesos y centralizar la gestión de seguridad de una forma que ayude a simplificar la difícil tarea de proteger la infraestructura IT de la empresa. Un SIEM proporciona a los expertos una ventaja para comprender la diferencia entre una amenaza de bajo riesgo y una que puede ser determinante para su negocio [5]. La posibilidad de contar con datos presentados en una vista centralizada de su infraestructura es efectiva solo si esos datos pueden ser estandarizados. Esto significa que, a pesar de las miles o millones de entradas provenientes de los distintos sistemas y fuentes, todo puede ser colocado en un formato común, listo para que la solución SIEM pueda ejecutar su análisis y correlación. Esto reduce la carga de trabajo del personal y le permite aprovechar una vista optimizada de la actividad y las potenciales preocupaciones.

Una vez integrado, se cuenta con la capacidad de [5]:

- Centralizar la vista de potenciales amenazas.
- Determinar qué amenazas requieren resolución y cuáles no.
- Escalar temas a los analistas apropiados, para que puedan tomar una acción rápida.
- Incluir el contexto de los eventos para permitir resoluciones bien informadas.
- Documentar, en un registro de auditoría, los eventos detectados y cómo fueron resueltos.
- Cumplir con las regulaciones de la industria en un formato de reporte sencillo.

2.3. XDR

Las organizaciones, normalmente utilizan el SIEM para recopilar registros y alertas procedentes de múltiples soluciones. Si bien permiten a las empresas juntar mucha información procedente de múltiples lugares para conseguir una visibilidad más centralizada, esto hace que los eventos por segundo o EPS ("*Events per Second*" en inglés) crezcan de manera drástica. Los EPS son una medida que hace referencia al número de eventos log que se generan en un sistema por segundo. Por ejemplo, una compañía con 1000 empleados puede observar hasta 22000 eventos por segundo entrando en su SIEM. Eso hace un total de casi 2 millones de eventos al día [7].

Las alertas son difíciles de clasificar. La correlación y conexión de todos los registros de información para obtener una visibilidad de un contexto mayor resulta todo un desafío con una sola solución SIEM [8]. Además, esto provoca que el tiempo medio de detección y respuesta ("*Mean Time To Detect*" o MTTD, "*Mean Time To Respond*" o MTTR, respectivamente) aumenten significativamente, ya que las alertas no pueden ser atendidas de inmediato.

2.3.1. Definición

Un sistema de **Detección y Respuesta Extendida** o XDR ("*Extended Detection and Response*" en inglés) consiste en una tecnología que recopila y correlaciona automáticamente datos en múltiples capas de seguridad: correos electrónicos, equipos finales, servidores y redes, tanto locales como alojados en la nube. Esto permite una

detección más rápida de las amenazas y una mejor investigación y tiempos de respuesta mediante un análisis de seguridad.

2.3.2. Funcionalidad

XDR une toda una serie de actividades de baja fiabilidad en un evento de alta fiabilidad, creando menos alertas y priorizándolas para tomar medidas. Automatiza las investigaciones sobre amenazas eliminando los pasos manuales y proporciona valiosos datos y herramientas para un análisis que, de otro modo, hubiese sido imposible. Recopila datos y suministra esa información a un "*data lake*" para lograr un rastreo, búsqueda e investigación extendidas a través de las capas de seguridad. Un *data lake* es un repositorio centralizado diseñado para almacenar, procesar y proteger grandes cantidades de datos estructurados, semiestructurados o no estructurados. Mediante la aplicación de inteligencia artificial y análisis especializados, se enriquece el conjunto de datos y por ende, se producen menos alertas y con más contexto [8].

De igual manera, esto no reemplaza al SIEM, sino que lo mejora reduciendo el tiempo que los analistas de seguridad necesitan para evaluar alertas y registros relevantes, para así decidir qué es lo que necesita atención y merece mayor investigación.

2.4. IRP

En los Centros de Operaciones de Seguridad, los diferentes eventos o alarmas que se detectan contienen ciertos elementos, normalmente llamados "**observables**", con los que es posible elaborar hipótesis que tratan de responder a preguntas como: ¿Qué ha ocurrido? ¿Por qué ha ocurrido? ¿De dónde proviene el ataque? ¿Quién ha sido el atacante? Para responder, es necesario llevar a cabo un análisis exhaustivo sobre dichos elementos y contrastarlos con una información fiable. Tras una serie de análisis, es posible determinar si se trataba de una amenaza real o un falso positivo. Este es un suceso que se da cuando el agente, por error, identifica como malware un fichero que es legítimo e inofensivo.

2.4.1. Definición

El término **Plataforma de Respuesta a Incidentes** o IRP ("*Incident Response Platform*" en inglés) hace referencia a la práctica de investigar y corregir ataques cibernéticos en un sistema de información, normalmente mediante un software que guía, asiste y automatiza la respuesta a incidentes.

2.4.2. Funcionalidad

Las plataformas de respuesta a incidentes son un software capaz de dotar al sistema con una facilidad para analizar cada uno de los eventos. Dentro de las IRP, se pueden aislar eventos que puedan estar relacionados y desviarlos a un caso de estudio o "*event case*". De esta manera, el trabajo sobre las alertas está mejor diferenciado. Además, entre

otras muchas funciones (dependientes del software), los eventos se pueden compartir con el resto de los analistas y luego dividirlos en tareas para su distribución en el equipo. Las plataformas IRP permiten además la automatización de respuestas en base a patrones preestablecidos o "*playbooks*". Funcionan de la siguiente manera: cuando se detecta un tipo determinado de evento, se desencadena un proceso de respuesta establecido que libera de trabajo al equipo de analistas del SOC [9].

En el flujo de trabajo de un SOC, es muy importante la automatización máxima de los procesos internos. Las plataformas IRP como TheHive, permiten la integración mediante APIs con otras instancias, de manera que la migración de las alertas entre servicios es instantánea.

2.5. Sistemas de Detección de Intrusiones

Como se describe anteriormente, el SIEM representa la visión global de la seguridad informática en una organización y tiene la principal función de detectar comportamientos anormales para prevenir la organización de amenazas informáticas. Pero dentro del mismo, existe una parte fundamental: la detección de intrusiones. Se repasa de manera más detallada ya que uno de los componentes del SOC del proyecto es un Sistema de Detección/Prevención de Intrusiones o IDS/IPS ("*Intrusion Detection/Prevention System*" en inglés), llamado Suricata.

2.5.1. IDS

El **Sistema de Detección de Intrusiones** o IDS ("*Intrusion Detection System*" en inglés) es una aplicación usada para detectar accesos no autorizados a un ordenador o a una red. Es decir, son sistemas que monitorizan el tráfico entrante y lo cotejan con una base de datos actualizada de firmas de ataque conocidas. Ante cualquier actividad sospechosa, emiten una alerta a los administradores del sistema quienes han de tomar las medidas oportunas. Estos accesos pueden ser ataques esporádicos realizados por usuarios malintencionados o repetidos cada cierto tiempo, lanzados con herramientas automáticas. Dichos sistemas sólo detectan los accesos sospechosos emitiendo alertas anticipatorias de posibles intrusiones, pero no toman acción ni mitigan la intrusión [10].

2.5.2. IPS

Un **Sistema de Prevención de Intrusiones** o IPS ("*Intrusion Prevention System*" en inglés) es un software que se utiliza para proteger a los sistemas de ataques e intrusiones. Su actuación es preventiva. Estos sistemas llevan a cabo un análisis en tiempo real de las conexiones y los protocolos para determinar si se está produciendo o se va a producir un incidente, identificando ataques según patrones, anomalías o comportamientos sospechosos y permitiendo el control de acceso a la red, implementando políticas que se basan en el contenido del tráfico monitorizado, es decir, el IPS además de lanzar alarmas, puede descartar paquetes y deshabilitar conexiones [10].

2.5.3. Sistemas basados en host

Un **IDS/IPS basado en host** (“*Host-based IDS/IPS*” en inglés) se implementa en un equipo final específico y está diseñado para protegerlo contra amenazas internas y externas. Este tipo de implementación puede tener la capacidad de supervisar el tráfico de red del equipo, tanto entrante como saliente, observar los procesos en ejecución e inspeccionar los registros del sistema. Un aspecto clave es que la visibilidad de un sistema basado en host se limita a su propio equipo, lo que disminuye el contexto disponible para la toma de decisiones. Igualmente, la visibilidad que tiene sobre sus componentes internos es muy profunda [11].

2.5.4. Sistemas basados en red

Un **IDS/IPS basado en la red** (“*Network-based IDS/IPS*” en inglés) es una solución que está diseñada para supervisar toda una red protegida. Tiene visibilidad de todo el tráfico que fluye a través de la red y toma determinaciones basadas en los metadatos y en los contenidos de los paquetes. Este punto de vista más amplio proporciona un mayor contexto y la capacidad de detectar amenazas generalizadas. Sin embargo, su capacidad es opuesta a los sistemas basados en host, que disponen de la información contenida en cada equipo [11].

CAPÍTULO III.

DISEÑO

En este capítulo, se ve de manera detallada los diferentes componentes software que forman el SOC. Primeramente, se hará un repaso sobre los requisitos funcionales. De esta manera, se clarifica al lector qué se está buscando o qué objetivos se consideran esenciales respecto al funcionamiento del centro de operaciones. Visto este apartado, se procede con una muestra del flujo de trabajo final que verían los analistas, describiendo las etapas y pasos que realizan en la solución de amenazas o evaluación del sistema. A continuación, para una comprensión mayor y más formalizada, se muestra el flujo anteriormente descrito con un diagrama UML de secuencia.

Vista esta breve introducción al funcionamiento, se entra en detalle con los componentes empleados y sus características, resaltando las que han sido necesarias de entender para implementar el SOC con éxito.

3.1. Requisitos funcionales

En la planificación de proyectos software, es de vital importancia el análisis y evaluación de los requisitos que directamente afectan al funcionamiento del sistema. Es necesario describir cualquier actividad que este deba realizar. Es decir, describir el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones.

En este proyecto de SOC, para el análisis de los requisitos funcionales, se valorarán los aspectos esenciales respecto a seguridad y funcionalidades de los componentes que lo integran.

- El sistema debe capturar y registrar constantemente la actividad de los equipos.
- El sistema debe capturar el tráfico de red entrante y saliente en cada equipo monitorizado.
- El sistema debe proveer las funcionalidades de SIEM y XDR detalladas a continuación:
    - Centralizar la visualización y gestión de las alertas generadas por los equipos.
    - Análisis de registros de log y clasificación de las alertas para generar un documento *JavaScript Object Notation* (JSON) de cada una de ellas, identificando todos los campos posibles recogidos en el registro de log.
    - Detección de procesos maliciosos o "*rootkits"*.
    - Evaluación de la configuración de los equipos en busca de fallos o posibles vulnerabilidades.

    - Detección de vulnerabilidades en las aplicaciones de cada equipo monitorizado.
    - Despliegue automático de medidas paliativas en respuesta a incidentes.
    - Monitorización de la integridad en sistemas de ficheros.
    - Escaneo periódico del inventario (hardware y software) del sistema para identificar cambios o eventos de sistema no esperados.
    - Cumplimiento de regulaciones y estándares.
    - Despliegue de agentes en los equipos que reporten toda su información al servidor.
- El sistema debe contar con una instancia IRP que reciba las alertas del SIEM y además cuente con las siguientes funcionalidades:
    - Centralización y visualización de alertas con la creación de paneles o *dashboards*.
    - Integración de herramientas que faciliten motores de análisis desde un mismo portal para la evaluación de los eventos.
    - Posibilidad de integración de motores de análisis para la evaluación de alertas y observables.
    - La utilización de los motores de análisis y todas las herramientas debe poder realizarse desde el mismo panel de control.
- El sistema debe integrar y automatizar la comunicación entre el SIEM y el IRP.

3.2. Flujo de trabajo

En este trabajo, se ha considerado que una explicación primeriza y superficial del funcionamiento del sistema final ayuda a comprender más rápidamente todo el conglomerado de piezas, componentes y propósitos de cada uno.

Se abarca el proceso completo desde la colecta de eventos por parte de los agentes del **SIEM** y el **IDS *Suricata***, pasando por su envío y clasificación al servidor SIEM con ***Wazuh***, que genera y clasifica las alertas, hasta el análisis de los observables considerados para corroborar con el segundo servidor, responsable de la solución **IRP**. Este es una combinación de las herramientas ***TheHive***, ***Cortex*** y ***MISP***.

3.2.1. Registro de eventos en el agente

Cada uno de los agentes desplegados por el SIEM **Wazuh**, dispone de una serie de procesos que, entre otros, se encarga del registro constante en los ficheros log y del reporte de éstos al servidor principal. La lista de los procesos que se registran es configurable, y puede tener tantos como la organización considere oportuno. En el caso de este proyecto, los eventos del IDS **Suricata** también son registrados y reportados al servidor. La integración de Wazuh con Suricata es una tarea sencilla. Véase el ANEXO III.

3.2.2. Análisis y consulta de eventos en el servidor Wazuh

La instancia SIEM, integra el programa Wazuh. En este despliegue, los agentes envían constantemente sus registros al servidor. Éste cuenta con un motor de análisis y un conjunto de reglas ya configuradas. Dichas reglas se agrupan en ficheros asociados a cada servicio del sistema. Por ejemplo, el fichero *0095-sshd-rules.xml* guarda las reglas asociadas al servicio SSH. Con ellas, se establecen los criterios para que un log proveniente de los agentes sea clasificado como una alerta o no. En la Figura 1 se ve que cada regla tiene un campo *match* el cual es comparado con el evento que envió el agente.

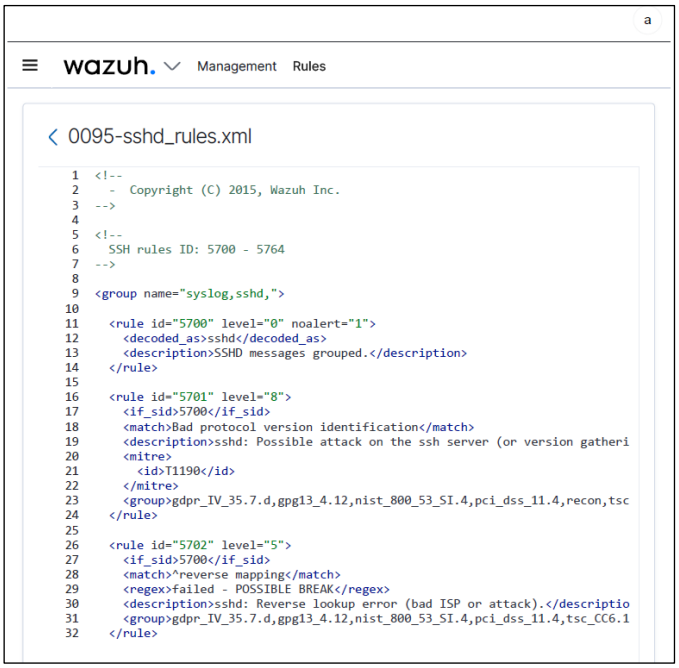

Figura 1. Vista parcial del fichero de reglas 0095-sshd_rules.xml en Wazuh.

Una vez se han generado las alertas, gracias al *Wazuh dashboard* pueden visualizarse de manera intuitiva (Figura 2). Es la interfaz principal de uso, donde se lleva a cabo la consulta y evaluación de las alertas. Además, como se ve en la Figura 3, podemos ver información más detallada de las mismas en función de su tipo y niveles, técnicas y tácticas de la alerta, etc. Las capacidades e información disponibles son muy altas.

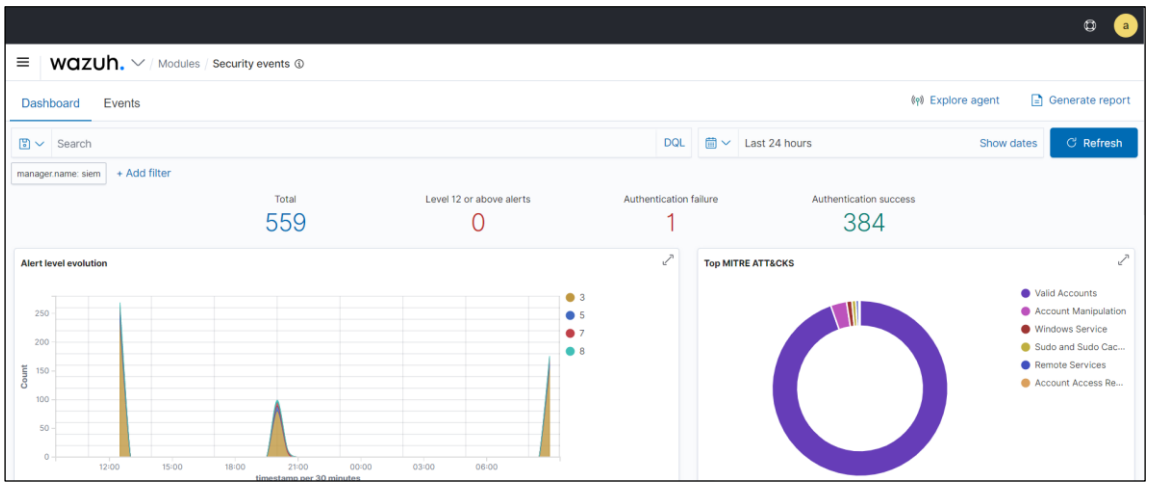

Figura 2. Vista de la interfaz con eventos en Wazuh.

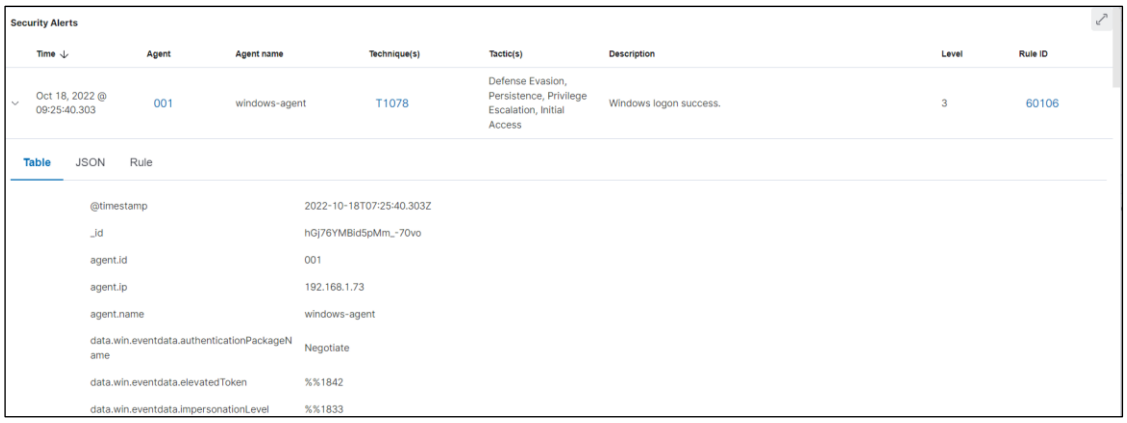

Figura 3. Vista parcial de la información aportada en una alerta de Wazuh.

3.2.3. Análisis de alertas en TheHive

Una vez las alertas están generadas, el servidor IRP con ***TheHive*** las recibe en su interfaz. De esta manera, podrá analizar las que el SOC necesite. Además, dentro del IRP están integradas las herramientas de análisis. En este ejemplo se cuenta con:

- **Cortex** como facilitador de los motores de análisis.
- **MISP** como motor de análisis para observables del tipo: *domain*, *IP*, *URL*, *FQDN*, *URI_path*, *user-agent*, *hash*, *mail*, *mail_subject*, *registry*, *regexp*, *other*, *filename.*
- **AbuseIPDB** como motor de análisis para observables de tipo: *IP*

Visto esto, considérese una situación como la siguiente:

1. El analista del SIEM detecta una alerta con un observable sospechoso.

2. En ese momento, el analista debe informar de esa alerta al analista IRP con la llave identificadora del evento.

3. El analista IRP selecciona la alerta gracias a la llave que le dio el compañero, para entonces generar un nuevo caso.

4. El analista IRP utiliza los analizadores disponibles (Figura 4) para poder evaluar si el observable es malicioso o no.

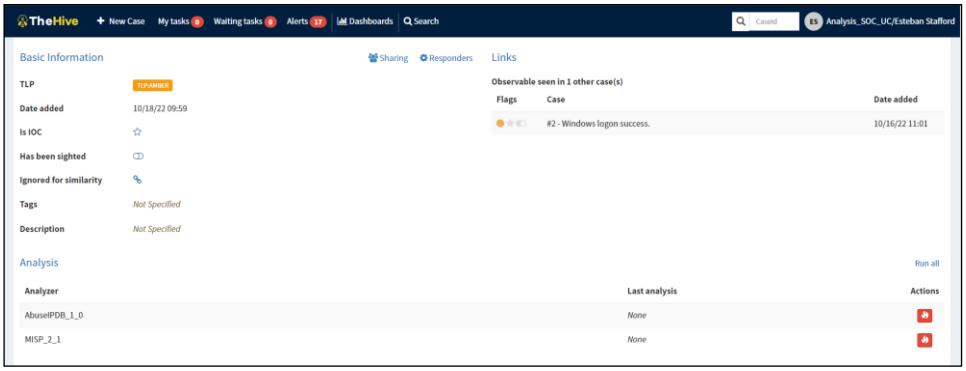

Figura 4. Vista de análisis en TheHive con analizadores AbuseIPDB y MISP disponibles.

5. El analista IRP obtiene el resultado del análisis (Figura 5) y poder resolver el caso.

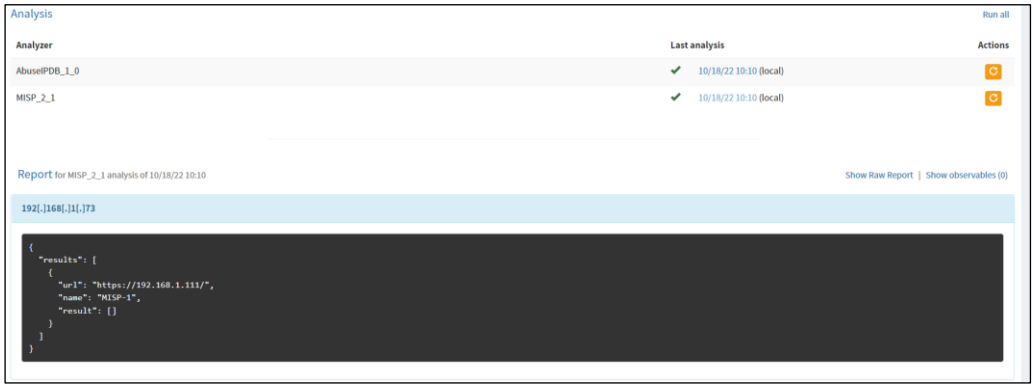

Figura 5. Vista del resultado de análisis con MISP sobre una dirección IP en TheHive.

6. El analista IRP cierra el caso con un resumen de lo obtenido.

3.2.4. Diagrama

A continuación, mediante un diagrama UML de secuencia, se resume de manera gráfica y formalizada, el flujo de funcionamiento general del SOC.

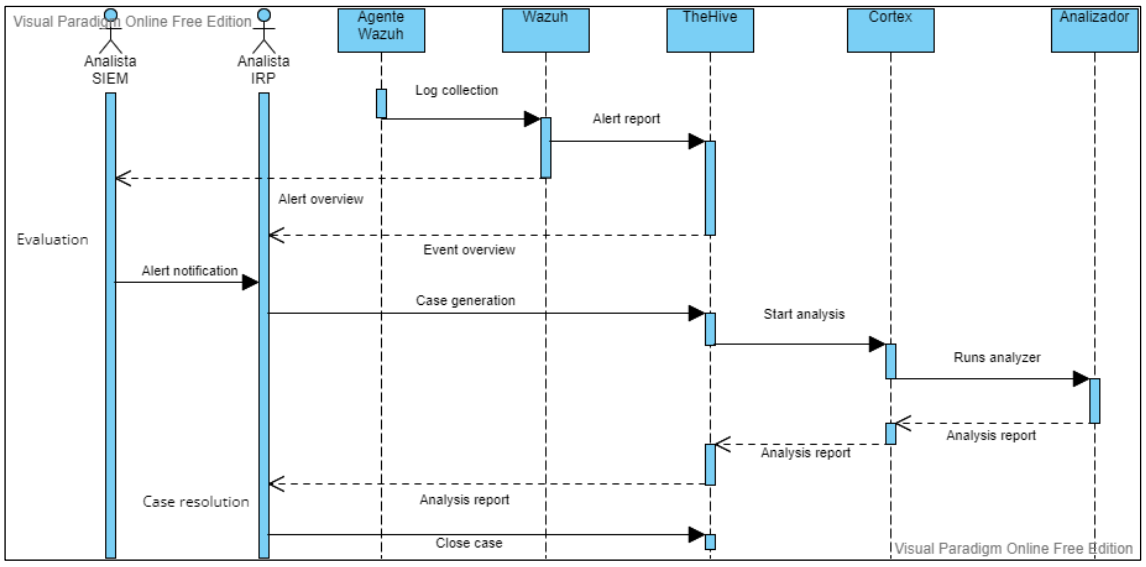

Figura 6. Diagrama de secuencia con flujo de trabajo del SOC.

En la Figura 6, se pueden ver todos los procesos descritos:

- **Log collection** (Agente Wazuh - Wazuh). Proceso por el cual el servidor recoge todos los logs de los agentes y los analiza.

- **Alert report** (Wazuh - TheHive). Proceso de reporte desde el servidor SIEM con Wazuh al servidor IRP con TheHive gracias al conector o *alert feeder thehive4py*.

- **Alert overview** (Wazuh – Analista SIEM). Proceso de por el que el servidor SIEM muestra el reporte de todos los eventos.

- **Event overview** (TheHive – Analista IRP). Proceso por el que el servidor IRP muestra la colección de alertas cogidas del SIEM.

- **Evaluation** (Analista SIEM). Proceso de investigación y análisis de alertas en busca de actividad anómala.

- **Alert notification** (Analista SIEM – Analista IRP). En caso de que el analista del SIEM vea un evento sospechoso que requiera un análisis, notifica al compañero para que lo efectúe desde el IRP.

- **Case generation** (Analista IRP - TheHive). El analista genera un caso a partir de las alertas que ha recogido el servidor.

- **Start Analysis** (TheHive - Cortex). El servidor, cuando va a efectuar un análisis, hace una llamada al servicio de Cortex para que ejecute el motor de análisis pertinente.

- **Runs Analyzer** (Cortex - Analizador). Proceso por el que Cortex ejecuta el analizador.

- **Analysis Report** (Analizador – Cortex – TheHive – Analista IRP). Proceso por el que el reporte del análisis realizado vuelve al dashboard del analista.

- **Case Resolution** (Analista IRP). Proceso de evaluación por parte del analista y solventación.

- **Close case** (Analista IRP - TheHive). Proceso de cierre con el resumen y conclusión del caso.

3.3. Wazuh

Dentro del proyecto SOC, la primera pieza necesaria para la securización del sistema es una instancia que recoja o identifique alertas para su posterior análisis y notificación de los trabajadores. Revisando los requisitos funcionales, se dice que una solución SIEM y XDR es requerida, puesto que implementa las funcionalidades descritas.

Wazuh es una plataforma de seguridad en código abierto que implementa soluciones de **SIEM** y **XDR** para equipos finales y cargas de trabajo en la nube. El sistema tiene cuatro componentes necesarios:

- **Agente** universal para la recolección de actividad en equipos finales.
- **Manager** de Wazuh, compuesto por:
    - Wazuh ***Server*** para el análisis de los datos.

- Wazuh ***Indexer*** para el almacenaje y clasificación de la información.
- Wazuh ***Dashboard*** para la correcta visualización del sistema y alertas.

Vistos los componentes, se dice que es un sistema con una arquitectura basada en agentes. Dichos agentes (dispuestos en los *endpoints* o equipos finales), son los que ejecutan el servicio de reenvío de datos de seguridad al servidor central. Además, Wazuh cuenta con la capacidad de integrar *agentless devices*, como lo son *firewalls*, *switches*, *routers*, etc. Esto es posible ya que el reenvío de datos también es realizable mediante Syslog, SSH o APIs.

Respecto a su arquitectura y despliegue, es destacable la "clusterización" por parte del servidor y el indexer. Esto afecta en gran medida al rendimiento obtenido, puesto que, en despliegues con un gran número de equipos, el almacenaje y procesamiento idealmente deben tener recursos dedicados.

A continuación, se muestra un diagrama con el despliegue de Wazuh. En él se pueden ver las piezas principales y la interacción de las mismas:

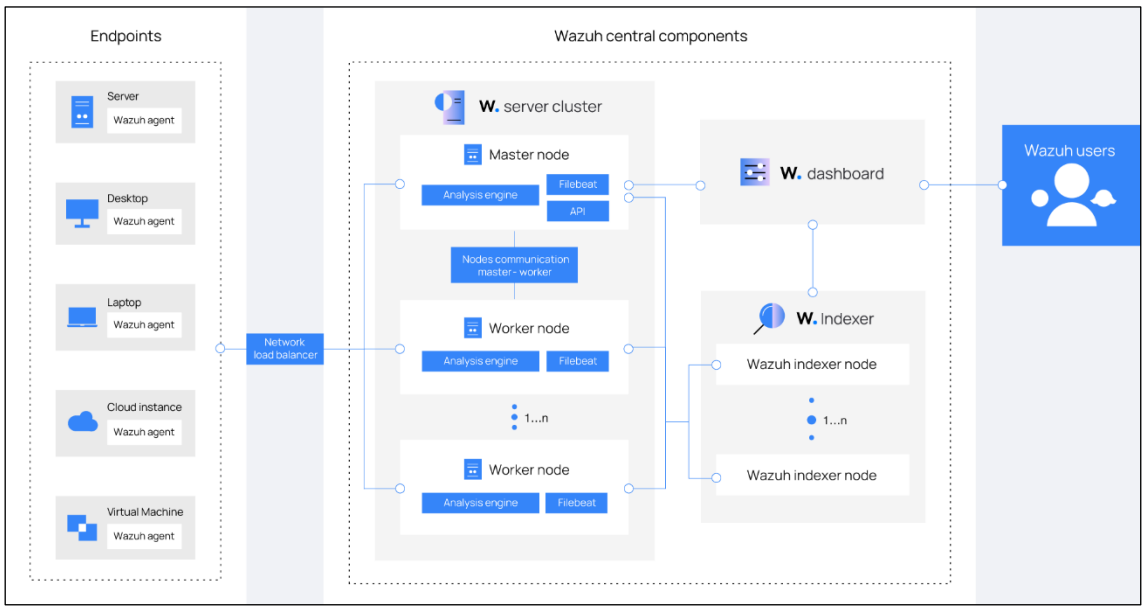

Figura 7. Diagrama general de la arquitectura de Wazuh [12].

Si analizamos la Figura 7, podemos ver que los endpoints ejecutan el servicio (***wazuh-agent.service***) correspondiente, que constantemente está reportando eventos al servidor. Dicho tráfico está regulado por un balanceador de carga para repartir los recursos del servidor entre todos los agentes. Una vez el tráfico llega al servidor, comienza el análisis.

En caso de tener un clúster para el Wazuh Server, el nodo maestro debe comunicar al resto de trabajadores la operativa. Entonces, cada nodo comenzará su análisis y evaluación de los eventos y así generar las alertas. Una vez realizado el análisis, se reporta la información clasificada y ordenada al Wazuh Indexer y Dashboard para su almacenaje y visualización. Hay que mencionar que, si el servidor principal y el indexer están en instancias diferentes, cada nodo del servidor emplea Filebeat para la comunicación segura con el indexer mediante TLS. Finalmente, los usuarios interactúan con la plataforma gracias al Wazuh dashboard.

Resumiendo, **¿por qué emplear Wazuh en un SOC?** En primer lugar, Wazuh es una plataforma con funcionalidades SIEM y XDR. Por lo tanto, cubre la parte referente al proceso de colecta de eventos en agentes, para luego notificar, detallar y resolver potenciales amenazas de seguridad. Además, Wazuh es un software gratuito. Es de código abierto y esto da lugar a muchas mejoras y personalizaciones posibles. Por último, como se ve en más adelante, es un sistema que requiere de pocos recursos para funcionar.

3.3.1. Wazuh server

Dentro de Wazuh, el servidor es el componente central más importante, ya que es el responsable del cometido principal: analizar y clasificar los eventos provenientes de los agentes y a partir de ellos, generar las alertas. De igual manera, pueden contemplarse los siguientes componentes que lo dotan de funcionalidad:

- ***Agent enrollment service.*** Es el servicio encargado de registrar los agentes dentro del sistema. Se realiza mediante la distribución de llaves únicas a cada uno. Es un proceso que se lleva a cabo cuando el agente se conecta por primera vez.

  Existen dos métodos para el registro de agentes: mediante el establecimiento de la IP del servidor o mediante la API del servidor. En este proyecto, se emplea el primero, pues es más rápido y sencillo. En la propia instalación del agente en el endpoint se establece la IP del servidor al que va a conectarse. Una vez se inicia el servicio, manda una petición al servidor solicitando la llave.

- ***Agent connection service.*** Es el servicio encargado de recibir los datos de los agentes. Para la comunicación, emplea las llaves asignadas durante el *enrollment*. De esta manera, puede verificar las identidades y encriptar la comunicación entre el servidor y el agente.

- ***Analysis engine.*** Pieza central de cada nodo del servidor. Es el componente que realiza el análisis de los datos recibidos. Esta evaluación se lleva a cabo mediante decodificadores y reglas, cuya misión es contrastar los datos en busca de IoCs conocidos. Un Indicador de Compromiso o IoC ("*Indicator of Compromise*" en inglés) es un dato que funciona como evidencia que permite detectar amenazas. Incluso puede servir como prueba de haber sido atacado con éxito. Con ello, es capaz de generar alertas y notificar al analista. Además, es el componente responsable de manejar, configurar y actualizar los agentes.

  Cabe destacar que, en el proceso de análisis y muestra al analista, Wazuh es capaz de enriquecer los datos gracias a ATT&CK, de MITRE [13]. ATT&CK es una base de datos facilitada por MITRE, globalmente accesible que concentra el conocimiento detallado de las tácticas y técnicas empleadas por criminales basadas en observaciones del mundo real. Dispone de varias matrices de técticas/tácticas, dependiendo de la instancia sobre la que se aplica: empresas, cloud, móvil, etc. Dentro de ATT&CK, considérense estas dos categorías:

  - **Táctica**: Representa el "***por qué***". Es decir, la razón por la que el atacante ha realizado esa acción. Por ejemplo, la táctica TA0043 está clasificada como

"*Reconnaissance"* y se describe como: "El adversario está intentando conseguir información para emplearla en operaciones futuras."

- **Técnica**: Representa el "***cómo***". Es decir, el método que ha empleado para cometer esa acción. Por ejemplo, dentro de la táctica de *Reconnaissance*, una de las técnicas es la T1595: "*Active Scanning*" o "Escaneo Activo". Dentro de ésta, tenemos además varias técnicas derivadas, como puede ser la T1595.001: "*Block IP Scanning*".

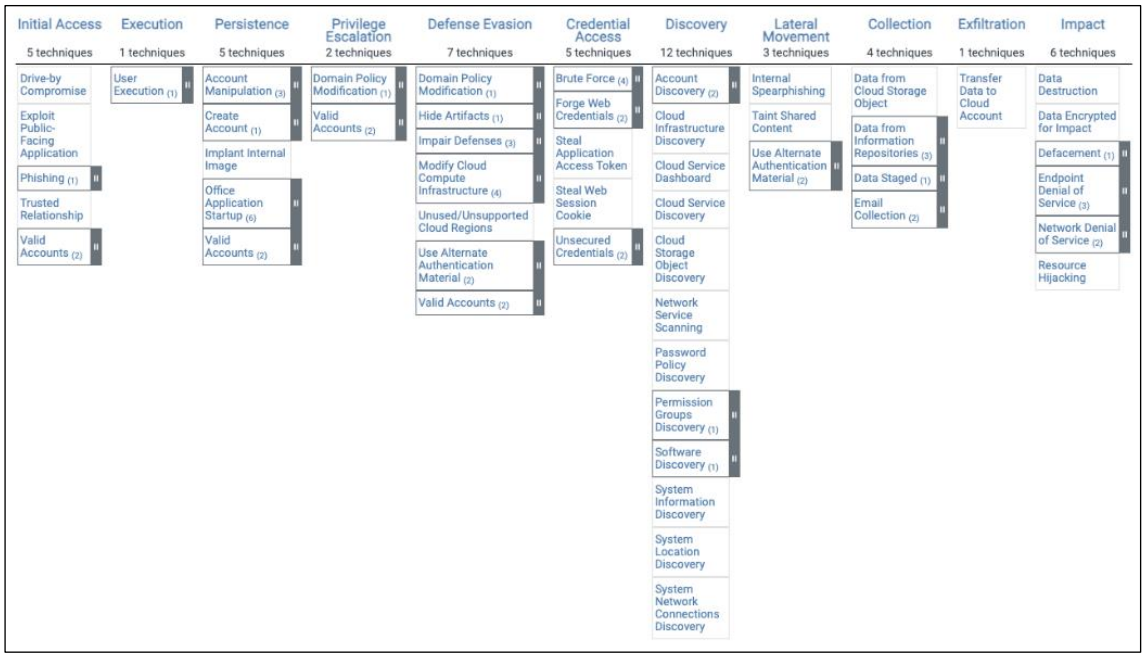

Figura 8. Matriz de la base de datos ATT&CK, de MITRE para instancias Cloud [13].

Como se puede ver en la Figura 8, cada táctica contiene una serie de técnicas derivadas. En conjunto, esto enriquece los datos mostrados por Wazuh. Dicha consulta indica, para cada alerta, tanto la táctica como la técnica empleada.

- **Wazuh RESTful API.** Este servicio proporciona una interfaz para interactuar con los componentes del servidor Wazuh. Por ejemplo, es empleado para modificar la configuración de los agentes, monitorear el estado de la infraestructura, editar los decodificadores y los conjuntos de reglas, etcétera.

- ***Wazuh cluster daemon.*** Es el servicio encargado de la gestión del servidor en caso de que esté dividido en clústeres. En este proyecto no se emplea puesto que el diseño será unificado en un mismo servidor.

- **Filebeat.** Es empleado para enviar los datos del servidor al indexer. Se encarga de leer las salidas del *analysis engine* y las reporta en tiempo real. Además, en este tráfico también incorpora un balanceo de carga en caso de clusterización. En este proyecto no tiene relevancia directa, pues no hay clusterización.

3.3.2. Wazuh indexer

El indexer de Wazuh es un motor de búsqueda de texto completo empleado para extraer información casi en tiempo real de la base de datos. Este componente elabora índices y almacena las alertas generadas por el servidor.

La base de datos se clasifica en índices o *indexes*. Cada index es un conjunto de documentos que están relacionados entre ellos. Cada uno de los documentos está almacenado en formato JSON y contiene una serie de llaves y propiedades con sus correspondientes valores: *strings*, números, booleanos, fechas, *arrays* o listas de valores, geolocalizaciones, entre otros. Véanse los cuatro índices empleados por Wazuh:

- ***wazuh-alerts:*** Almacena las alertas generadas por el servidor.
- ***wazuh-archives:*** Almacena todos los eventos recibidos por el servidor, ya sean alertas o no.
- ***wazuh-monitoring:*** Almacena todos los datos relacionados con los agentes. Es usado para representar los diferentes estados en los que los agentes pueden estar: *Active*, *Disconnected*, *Never Connected*.
- ***wazuh-statistics:*** Almacena todos los datos relacionados con el rendimiento del servidor.

3.3.2.1. Index management

Para evitar la sobrecarga de disco en el almacenamiento del servidor, existe la posibilidad de gestionar el ciclo de vida de los índices del Wazuh Indexer mediante políticas o "*Index Policies*". Esta funcionalidad es efectuada desde el servidor y es altamente configurable. Aunque por defecto, Wazuh no trae implementada ninguna de estas políticas.

El ciclo de vida de los índices funciona como las máquinas de estados: se define un estado inicial y en función de unas condiciones, se establece la transición al estado siguiente. Principalmente, existen 2 estados diferentes: "*hot state*" y "*cold state*". Además, dentro de cada estado, se definen las acciones que el analista decida. Por ejemplo, al llegar a *cold state*, eliminar el índice. La implementación y ejemplo práctico se ven en el ANEXO III.

3.3.3. Wazuh dashboard

El *dashboard* de Wazuh es la interfaz web que posibilita la correcta visualización de la información y procesos que maneja el servidor. Todos los procesos, como los análisis, la consulta de las alertas y eventos de seguridad, son monitoreados fácilmente desde este módulo. Un ejemplo del dashboard es la Figura 2, que muestra la vista de la interfaz principal con los eventos de seguridad emergentes.

3.3.4. Wazuh agents

Los agentes de Wazuh son las instancias desplegadas por el servidor en los equipos finales para recolectar toda la información pertinente y luego analizarla.

El servicio *wazuh-agent.service* tiene una arquitectura modular. Esto implica que el analista puede habilitar y deshabilitar cada una de las funcionalidades dependiendo de las preferencias de uso. Cada módulo está a cargo de sus propias tareas, las cuales se comentan a continuación. Además, los usuarios pueden gestionar los agentes mediante sus ficheros de configuración, adaptando la solución para sus casos de uso particulares. Las diferentes tareas de seguridad pueden dividirse en los siguientes módulos:

- ***Log collector.*** Este módulo se ocupa de la lectura de ficheros de log y eventos de Windows. También puede enriquecer los ficheros JSON con información adicional.

- ***Command execution.*** Los agentes pueden ejecutar comandos autorizados provenientes del servidor, para así obtener y agrupar nueva información que se requiera. Así, se envía de vuelta al Wazuh server para analizarla.

- ***File Integrity Monitoring* (FIM).** La funcionalidad de este módulo recae sobre el monitoreo de los sistemas de ficheros. Reporta cambios cuando nuevos ficheros son creados, modificados o eliminados. Además, mantiene el registro de los cambios en permisos, atributos y contenido. La puesta en marcha de esta funcionalidad es sencilla, ya que simplemente se indica en el fichero de configuración la ruta de los directorios que desean ser monitoreados.

- ***Security Configuration Assessment* (SCA).** Este módulo provee de continua evaluación sobre la configuración del sistema. Es decir, en base a los bancos de pruebas o "*benchmarks*" dispuestos por el CIS ("*Center for Internet Security"*), el agente comprueba posibles vulnerabilidades en la configuración del sistema.

- ***System inventory.*** Este módulo, periódicamente escanea el sistema, revisando el inventario, elementos como: la versión del sistema operativo, las interfaces de red y su actividad, los puertos abiertos, programas instalados, etcétera.

- ***Malware detection.*** Los agentes, disponen de un proceso de detección estática basada en firmas capaz de captar e identificar anomalías, procesos ocultos (caso de *rootkits*), ficheros y puertos durante llamadas al sistema. Este tipo de detección se caracteriza por examinar programas en busca de secuencias de código que deberían revelar el intento malicioso del programa. El objetivo es acceder al fragmento de código que representa el comportamiento del mismo.

- ***Active response.*** Este módulo se ejecuta de manera automática cuando las amenazas son identificadas. Tiene capacidad de bloquear conexiones de red, detener procesos o incluso eliminar ficheros maliciosos. Además, el usuario puede customizar las respuestas predefinidas e incluso crear nuevas.

- ***Container security monitoring.*** Este módulo está integrado con la API de Docker para así poder monitorizar contenedores.

- ***Cloud security monitoring.*** Este módulo permite la monitorización de instancias en la nube, como pueden ser Amazon AWS, Microsoft Azure o Google GCP.

El módulo de respuesta activa es el cual permite dotar al SOC de una solución capaz de tomar acción frente a situaciones de alerta. Dichas respuestas son scripts configurados para ejecutarse cuando un determinado evento sucede, cuando la alerta tiene un *alert_level* determinado, etc. En primer lugar, se debe definir cuándo va a ejecutarse la respuesta. Hay dos modos, *stateful*, que deshace la acción pasado un tiempo, o *stateless*, que se ejecuta una sola vez. A continuación, se determina dónde va a ejecutarse el script, que puede ser localmente, es decir, en el agente que saltó la alerta, en el servidor, en un agente determinado o en todos los equipos.

La creación es bastante sencilla, pues en primer lugar se debe crear el comando que ejecutará la respuesta activa, para después elaborar la misma. Su configuración se lleva a cabo en el fichero de configuración del manager.

3.4. Suricata

En los agentes de Wazuh, se han abordado diferentes funcionalidades que describen el trabajo desempeñado por el servicio *wazuh-agent.service*, pero existe un añadido posible a la infraestructura de detección de anomalías, en los pasos previos al análisis con TheHive.

Suricata es un programa con múltiples funciones, entre las cuales se destacan las soluciones IDS e IPS. Estas son capaces de monitorear la red en busca de amenazas o intrusiones. Para ello, este software dispone de un conjunto de reglas que se usan de la siguiente manera: si el tráfico registrado por el servicio encaja con alguna de las reglas programadas, es considerado una alerta. Cabe destacar que dichas normas son personalizables y de libre gestión, ya que se pueden implementar tantos criterios como la organización considere. Una vez se evalúa el tráfico, la salida se dispone en un apartado de logs e incluso ficheros de tipo JSON. Resumiendo, en el ámbito del TFG, Suricata cubre estas cualidades:

- Solución de Sistemas de Detección de Intrusiones o IDS.
- Monitorización del tráfico entrante y saliente en cada endpoint.
- Mejora de la efectividad añadiendo reglas adicionales, entre las que se encuentra el compendio de "*Emerging Threats*". Este proporciona un conjunto de reglas proporcionadas por la comunidad y se van actualizando a diario.

Resumiendo, **¿por qué emplear Suricata en un SOC?** En primer lugar, es un sistema bastante ligero que proporciona otro punto más en el que detectar amenazas a los equipos finales. Es una solución IDS/IPS, a diferencia de Wazuh, que es un SIEM y XDR. Son sistemas con funciones complementarias. Con Suricata, la cobertura de seguridad que ofrece el SOC es mucho mayor, ya que es una fuente más de la que obtener eventos. Además, el uso de Suricata es muy práctico y automático, puesto que los agentes de Wazuh recogen estos datos y por tanto el servidor reúne toda esta información en una misma instancia.

3.5. TheHive

Wazuh es una herramienta sumamente potente y es posible que por sí sola sea suficiente, ya que aporta información como para que el analista actúe y resuelva la mayoría de los problemas él mismo.

Sin embargo, puede darse el caso de recibirse una alerta cuya contramedida no pueda ser llevada a cabo con éxito manualmente por el administrador de sistemas y requiera de ayuda extra.

Considérese el siguiente ejemplo: Llega al sistema un evento que notifica el intento de un acceso por el puerto 80. Además, se observa que la dirección IP de origen no pertenece a la lista de direcciones conocidas. La consulta de si ese observable tiene origen malicioso o no es necesaria. Para ello, ***TheHive***, que sería el siguiente paso dentro de la escala de mitigación de amenazas en este SOC, nos proporciona la respuesta. Es una plataforma de seguridad en código abierto con solución IRP que facilita el análisis de observables recibidos o captados por la organización.

Primeramente, a nivel de despliegue y perfiles, este IRP se divide en **organizaciones** y **usuarios**. Las organizaciones son grupos de usuarios que facilitan la división del personal de seguridad según los criterios o campos de trabajo que abarque el SOC. Luego, dentro de cada organización, se podrán generar tantos usuarios asociados como se desee.

Seguidamente, TheHive proporciona estas capacidades:

- **Colección**. TheHive es una plataforma capaz de agrupar cientos de alertas y observables con el objetivo de someterlos a un análisis. Este proceso es posible gracias a una API de cliente llamada "***TheHive4py***". Esta API permite la captura de eventos y alertas provenientes de diversas fuentes. En nuestro caso de estudio, Wazuh recoge las alertas gracias a su despliegue de agentes, para luego reportarlas mediante la API *thehive4py* al IRP y posibilitar su análisis. La integración es un proceso que se desarrolla en el ANEXO III.

- **Colaboración**. TheHive permite la consulta en tiempo real de las alertas provenientes del SIEM. Además, se pueden clasificar y permitir que todo el equipo de trabajadores las consulte. Una misma instancia con TheHive, permite la creación de múltiples usuarios con distintos roles:
    - **Administrador.** Su abanico de acciones hace referencia a la configuración y gestión del funcionamiento del servidor, como puede ser la gestión de: plantillas de analizadores, organizaciones dentro del sistema, usuarios, estado de la plataforma, etc.
    - **Manager de organización.** Abarca la gestión y mantenimiento de una sola organización.
    - **Analista.** Trabajador cuya finalidad es la gestión inmediata de las alertas: creación de tareas y de casos de análisis, ejecución de analizadores, exportación de casos a otras instancias, etc.

- **Investigación.** TheHive recolecta cientos de alertas provenientes de distintas fuentes. Una vez se disponen en la plataforma, los analistas pueden generar casos de investigación o "***cases***" a partir de una o varias de ellas. Con el caso ya

generado, los analistas pueden dividirlo en diferentes tareas o "***tasks***". Esto facilita la división de trabajo entre el personal y su estructuración en fases.

- **Actuación.** Además de la creación de cases y tasks, TheHive permite la integración de motores de análisis para los cientos de observables y llevar a cabo una evaluación de la posible amenaza.

Resumiendo, **¿por qué emplear TheHive en un SOC?** Revisando los requisitos funcionales, TheHive es una plataforma de Respuesta a Incidentes de Seguridad (IRP), capaz de estructurar y llevar a cabo el estudio de las amenazas recogidas por el SIEM Wazuh. Además, permite el uso de múltiples motores de análisis mediante su integración con Cortex.

3.6. Cortex

Siguiendo el apartado anterior, Cortex es una herramienta que continúa el hilo de trabajo dentro del SOC. TheHive es una plataforma que admite la integración con múltiples motores de análisis para los observables. Cortex es la pieza que necesita TheHive para implementar dicha función. Se mejora el sistema principalmente de tres maneras:

- **Facilidad.** Cortex es una plataforma de seguridad que tiene acceso a múltiples (aproximadamente 100) motores de análisis. Algunos de ellos son: AbuseIPDB, MISP, VirusTotal o HybridAnalysis. Es una solución que permite agrupar la colección de herramientas de análisis que un SOC puede requerir para evaluar observables. Sin embargo, Cortex no los integra por defecto, pese a que es un proceso sencillo y relativamente rápido (véase el ANEXO III para más detalles). Esto es así porque para cada uno de ellos, existe un pequeño manual donde se detallan los parámetros o campos que permiten la habilitación de los motores en la plataforma. Por ejemplo, en el caso de AbuseIPDB, sólo se necesita una llave para habilitar la API del analizador dentro de Cortex.

- **Ejecución.** Una vez se habilitan los analizadores, se puede comenzar a emplearlos desde la interfaz web que dispone la plataforma, sin tener que instalar independientemente cada herramienta y provocar un desorden poco práctico dentro de la infraestructura de trabajo.

- **Integración.** TheHive, por defecto, dispone de un conector con una instancia Cortex para que desde la misma interfaz web, podamos ejecutar los analizadores que han sido habilitados en Cortex. Una vez se tenga un caso generado y los observables clasificados, aparecen los iconos de los analizadores para ver su ejecución y resultados (véase la Figura 4).

Resumiendo, **¿por qué emplear Cortex en un SOC?** TheHive y Cortex son dos herramientas que prácticamente van de la mano y la fusión de ambas, mejora en gran medida, la efectividad del SOC. Esto es así por la facilidad que tiene Cortex para agrupar todas las herramientas de trabajo que necesita el equipo. Además, es un programa muy ligero que apenas requiere de recursos y su practicidad es muy alta.

3.7. MISP

Analizando el servidor IRP, se puede concluir que TheHive es la pieza central del servidor. Esto es así ya que recoge todos los eventos provenientes de diversas fuentes. Igualmente, hay otro componente que permite el análisis de eventos y su intercambio con la comunidad.

MISP ("*Malware Information Sharing Platform*") es una plataforma de intercambio y análisis de amenazas capaz de compartir, almacenar y correlacionar IoCs provenientes de diversas fuentes: ataques, inteligencia de amenazas, información sobre fraudes financieros, vulnerabilidades e incluso medidas "anti-terroristas". Dicha información es aportada por las comunidades y organizaciones registradas dentro de la red. Es una plataforma que provee al SOC de cientos de casos evaluados con los cuales se puede correlacionar y comparar IoCs en busca de respuestas. Resumiendo, podemos agrupar sus funcionalidades principales en las siguientes:

- **Base de datos** que agrupa tanto datos técnicos como no-técnicos de IoCs referentes a malware, incidentes, actores maliciosos e inteligencia.

- **Correlación automática** buscando relación entre atributos e indicadores de eventos sobre malware, campañas de ataque o análisis.

- **Modelo de datos flexible** que permite expresar la información compleja (incidentes o la inteligencia de amenazas) mediante enlaces y relaciones fáciles de comprender.

- **Interfaz de usuario gráfica e intuitiva** para crear, actualizar y colaborar en eventos y sus respectivos atributos e indicadores.

- **Integrador de texto** capaz de importar datos en formato flexible o desestructurado. Por ejemplo, dígase de un usuario que tiene un fichero de texto plano que contiene URLs, IPs, dominios y usuarios que ha ido obteniendo durante sus investigaciones. En ese caso, MISP es capaz de clasificar todos esos datos de manera organizada.

- **Sistema de colaboración** donde los usuarios pueden colaborar aplicando cambios o sugerencias en los eventos y atributos.

- **Intercambio de datos automática** entre grupos, organizaciones y comunidades de la red.

- **Importar "*feeds*"** referentes a organizaciones que comparten sus estudios e inteligencia de amenazas.

- **API para integrar MISP** con otras herramientas que desee el equipo de trabajo o empresa. Cuenta con *PyMISP*, una librería en Python que permite actualizar, añadir y gestionar los atributos de los eventos.

- **Módulos de expansión** que permiten llevar MISP a otros servicios del sistema.

Resumiendo, **¿por qué emplear MISP en un SOC?** Primeramente, es una plataforma que por sí sola da un sustento y cantidad de información sumamente amplia. Cada evento compartido por las organizaciones contiene cientos de observables, gráficos de correlación con otros eventos, atributos diferenciadores, entidades emisoras, y demás. Es decir, forma una grandísima fuente de información fiable y asentada. En este caso de estudio, la intervención de MISP en el SOC es fundamentalmente trabajar como motor de análisis. Para ello, está enlazado con Cortex y así cotejar datos y analizar observables con la gran base de datos que dispone.

3.8. Otros componentes

Además de los mencionados anteriormente, en el proceso de instalación (véase ANEXO III) hay dos programas externos que son necesarios para la instalación de los componentes en el servidor IRP. Estos son: **Apache Cassandra** y **Elasticsearch**.

3.8.1. Apache Cassandra

Dentro de la instalación de TheHive, es necesario proveer una base de datos para el almacenaje de los casos y eventos importados de Wazuh. Para ello se emplea el software **NoSQL Apache Cassandra**, que es un sistema de gestión de bases de datos de código abierto para bases de datos muy grandes, pero estructuradas. En este caso, "NoSQL" debe interpretarse como "*Not Only SQL*" en vez de "No SQL". Esto es debido a que no sólo soporta el lenguaje SQL, sino que también incorpora el suyo propio, CQL ("*Cassandra Query Language*"). Es muy parecido a SQL, pero cuenta con adaptaciones específicas para el trabajo con Cassandra.

Cassandra se trata de un sistema verdaderamente distribuido, ya que no usa un maestro. Todos los nodos tienen el mismo rango y pueden procesar cualquier consulta de base de datos, lo que aumenta notablemente la capacidad de rendimiento. Los datos están distribuidos en los nodos. El sistema es fácilmente escalable debido a que se pueden añadir más nodos de manera muy sencilla. Una vez realizada la instalación, ya solo se tienen que distribuir los archivos de configuración entre los nuevos nodos. Cassandra ofrece herramientas propias para ello. Esto es lo que se denomina escalabilidad horizontal, que consiste en un rendimiento proporcional a la cantidad de nodos que implementa el sistema [14].

3.8.2. Elasticsearch

Para el funcionamiento de TheHive y Cortex, es necesario incluir un método que trabaje sobre la base de datos e indexe o clasifique la información, que por defecto viene desestructurada. Volviendo al Wazuh indexer, se vio lo que eran los índices: son una colección de documentos relacionados entre sí. Elasticsearch trabaja de la misma manera. De hecho, en versiones anteriores de Wazuh, el uso de este software era necesario.

Los datos sin procesar fluyen hacia Elasticsearch desde una variedad de fuentes, incluidos logs, métricas de sistema y aplicaciones web. La ingesta de datos es el proceso

mediante el cual estos datos son parseados, normalizados y enriquecidos antes de su indexación en Elasticsearch. Almacena datos como documentos JSON y en cada uno correlaciona un conjunto de claves (nombres de campos o propiedades) con sus valores correspondientes (textos, números, booleanos, fechas, variedades de valores, geolocalizaciones u otros tipos de datos). Usa una estructura de datos llamada "índice invertido", que está diseñado para permitir búsquedas de texto completo muy rápidas. Un índice invertido hace una lista de cada palabra única que aparece en cualquier documento e identifica todos los documentos en que ocurre cada palabra [15].

CAPÍTULO IV.

DESPLIEGUE HARDWARE

En este punto, los aspectos teóricos y razones de uso de cada componente del Centro de Operaciones están mostrados y desarrollados. A continuación, se introduce el apartado referente al despliegue de dichos componentes en las correspondientes máquinas físicas, especificando para cada una de ellas las especificaciones técnicas necesarias para un correcto funcionamiento del SOC. Este tipo de aspectos se cubren en el apartado de requisitos no funcionales. Además, se acompaña con un breve estudio de los motivos por los que se eligen dichos dispositivos.

Considérese que este estudio se basa en las instalaciones de la Facultad de Ciencias. La división y clasificación de los equipos finales tiene como referencia las aulas de laboratorio, los servidores desplegados, y los equipos docentes de cada aula.

Finalmente, en base a estos parámetros y a la documentación oficial de cada instancia del sistema, se realiza una estimación lo más realista posible de la infraestructura hardware necesaria para implementar el SOC.

4.1. Requisitos no funcionales

Como se ve en el capítulo anterior, la elaboración de un análisis sobre los requisitos funcionales del sistema es primordial, ya que define las características esenciales del SOC. En cambio, este apartado con los requisitos no funcionales aborda otros aspectos, aquellos que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos.

Cualquier SOC es un sistema que trabaja sobre numerosos equipos diferentes, que dependen en gran medida de la magnitud de la organización. Por ende, para los requisitos no funcionales, se tendrá en cuenta la cantidad de equipos que se quiere monitorizar. Además, dentro del despliegue total que se desea instalar, se ve el tipo de equipos que son y cuántos hay de cada uno: servidores, equipos personales, etcétera.

- El sistema debe soportar un despliegue simultáneo de 135 equipos, respetando un margen de ampliación de 10. Su desglose es el siguiente:
    - Laboratorio de Simulación LSC-1: Dispone de 42 equipos HP PRODESK 600 G3 Base MT.
    - Laboratorio de Simulación LSC-2: Dispone de 24 equipos:
        - 9 equipos Dell OptiPlex 5070.
        - 1 equipo HP PRODESK 600 G3 Base MT.
        - 3 equipos HP PRODESK 600 G4 Base MT.
        - 12 equipos Dell OptiPlex 5080.

    - Laboratorio de Simulación LSC-4: Dispone de 31 equipos Dell OptiPlex 5070.
    - Equipos de docencia: 20 aulas con 1 equipo cada una.
    - Equipos para el servicio "*UnicanLabs*": 12 equipos HP PRODESK 600 G3 Base MT.
    - Servidores. Dispone de 4 dispositivos:
        - Servidor "aulaserver": ProLiant DL585 G7.
        - Servidor "vm": HP Compaq 6300 Pro MT PC.
        - Servidor "selkie": HP Compaq 6200 Pro MT.
- El sistema debe soportar un acceso simultáneo de 2 usuarios.
- El sistema debe almacenar los eventos y alertas durante un periodo de 90 días. Tras este punto, se eliminan.

4.2. Criterios de despliegue

Una vez revisados los requisitos referentes al modo de operación, lo siguiente es responder a una sencilla pregunta: ¿Cuántos servidores hacen falta?

Entre los motivos tomados para decidir el despliegue, primeramente, se han considerado las tareas diferenciadas que se desempeñan dentro del SOC. Por tanto, dividirlo en 2 servidores: un servidor para las soluciones **SIEM/XDR** y otro para las soluciones de análisis e **IRP**.

Conectado con lo anterior, una razón por la que dividir según funcionalidades, es evitar la excesiva descentralización. Aparte de la asignación que ocupa a cada uno de los trabajadores, la gestión y mantenimiento de los equipos que soportan el sistema es un extra a su trabajo, que suele requiere mucha atención y tiempo. Por eso, incluir demasiados servidores (por ejemplo, uno por cada servicio) es muy poco práctico, ya que su mantenimiento no es algo sencillo. Con que cada administrador se ocupe de uno, es suficiente. Además, desde el punto de vista de la productividad del trabajador, es cauto recordar que cuanto más se pueda centrar en una misma actividad, más efectivo será en ella. El exceso de tareas o dedicaciones baja el rendimiento en todas ellas drásticamente. Como consecuencia, se ha asumido que el funcionamiento del Centro de Operaciones se lleva a cabo por dos trabajadores:

- Administrador SIEM. Es el responsable del mantenimiento, recogida, gestión y evaluación de los eventos que llegan al SOC. Su tarea corresponde al uso único de Suricata y Wazuh. Es el que procura una barrera de defensa contra las amenazas lo más densa posible y en caso de quiebra o amenaza del sistema, tomar medidas mediante las siguientes maneras:
    - La administración de los equipos finales para atender o mitigar cualquier problema emergente. Puede tratarse de una actualización de servicios, del sistema operativo, una eliminación de malware malicioso, el aislamiento de un equipo infectado, etcétera.
    - La notificación al analista IRP para que efectúe una investigación de los observables que considere sospechosos y respaldar las decisiones o evaluaciones que el administrador tome.

- Analista IRP. Encargado del análisis e investigación de los indicios o evidencias que generen los agentes para verificar las hipótesis que tengan sobre un actor malicioso. Así, se podrá concluir si la amenaza es un falso positivo o no.

Además de la consideración sobre el trabajo de cada uno de los miembros del equipo, esta división también procura un extra de seguridad, ya que se trata de máquinas diferentes y en caso de fallo o caída de uno de estos, no significará el fallo general de todo el sistema.

Una vez establecida la división por funciones, se aborda la siguiente cuestión: Dentro de cada servidor, ¿sería necesario separar las diferentes instancias que los componen**?**

La respuesta a ello se ve principalmente afectada por la magnitud del despliegue de agentes que se quiere monitorizar. En este caso es necesario comentar que este es un SOC de magnitud media. Cuenta con un despliegue de algo más de 100 equipos y los EPS generados no son tan altos como para saturar el servidor. Esta decisión se ha tomado en base a las especificaciones que se detallan en la documentación de cada plataforma.

4.3. Servidor SIEM/XDR

En el proceso de elección de los requisitos usados para el servidor SIEM, se han tenido en cuenta diferentes consideraciones. Principalmente, se ha acudido a la documentación oficial de Wazuh para obtener la estimación de recursos necesarios para ejecutar los servicios. Cabe destacar que dicha selección no ha sido tan fiel a la documentación como se podría esperar. A continuación, se explican todos los criterios empleados.

Primeramente, se muestra la Tabla 1, que considera que el servidor, indexer y dashboard se ejecutan en la misma máquina. Es una implementación sencilla que puede dar soporte a aproximadamente 100 agentes durante un periodo de 90 días.

Número de agentes	CPU	RAM	Almacenamiento (90 días)
1-25	4 vCPU	8 GiB	50 GB
25-50	8 vCPU	8 GiB	100 GB
50-100	8vCPU	8 GiB	200 GB

Tabla 1. Requerimientos hardware para una instancia completa de Wazuh [12].

Por otro lado, en la documentación también se dispone de los requisitos que necesitan sus componentes de manera individual, en caso de que el usuario desee desplegar el sistema en varios nodos. Han sido tenidos en cuenta ya que son bastante distantes de lo aportado en la Tabla 1.

En las características siguientes, recordar que el espacio necesario hace referencia a un solo nodo durante un periodo de 90 días. Y en función del tipo de dispositivo que sea, los EPS generados pueden variar. Esto hace que la instancia necesite más o menos espacio.

El **servidor** de Wazuh, que es instancia central de la plataforma, requiere estas características:

- Capacidad de procesamiento:
  - RAM (GB): 2 (mínimo), 4 (recomendado).
  - CPU (cores): 2 (mínimo), 8 (recomendado).
- Almacenamiento de disco:
  - Servidores: 0.25 EPS. Por tanto, 0.1 GB cada 90 días.
  - Equipos personales: 0.1 EPS. Por tanto, 0.04 GB cada 90 días.
  - Dispositivos de red: 0.5 EPS. Por tanto, 0.2 GB cada 90 días.

El **indexer** de Wazuh, instancia responsable de la base de datos, es la más exigente. Sus características son las siguientes:

- Capacidad de procesamiento:
  - RAM (GB): 4 (mínimo), 16 (recomendado).
  - CPU (cores): 2 (mínimo), 8 (recomendado).
- Almacenamiento de disco:
  - Servidores: 0.25 EPS. Por tanto, 3.7 GB cada 90 días.
  - Equipos personales: 0.1 EPS. Por tanto, 1.5 GB cada 90 días.
  - Dispositivos de red: 0.5 EPS. Por tanto, 7.4 GB cada 90 días.

El **dashboard** de Wazuh, es la menos exigente, ya que su función es ejecutar la interfaz. No requiere apenas de almacenamiento. De hecho, ni se considera:

- Capacidad de procesamiento:
  - RAM (GB): 4 (mínimo), 8 (recomendado).
  - CPU (cores): 2 (mínimo), 4 (recomendado).

Vista la documentación, si uno compara las especificaciones de la Tabla 1 con las recién mostradas, son bastante dispares. Por ejemplo, se recomienda 8GB de RAM para las 3 instancias juntas. Puede que sea suficiente, pero luego se mencionan: 4GB (Server) + 16GB (Indexer) + 8GB (Dashboard) = 28 GB. A causa de este "desequilibrio", se ha decidido ignorar las iniciales y dotar del servidor final con unas especificaciones algo más altas. En parte, porque el despliegue propuesto para la Facultad de Ciencias es de 135 agentes, algo más que la referencia de la Tabla 1. De igual manera, recordar que los equipos de la universidad no se usan el ni el 100% del tiempo, ni todos a la vez. Por tanto, tampoco es estrictamente necesario alejarse demasiado de las especificaciones iniciales.

A primera vista, no son unos requerimientos inabordables. Es por eso que incluir ciertos márgenes de seguridad aporta fiabilidad, ya que se tiene una garantía de rendimiento durante todo el tiempo de operación. Además, en caso de que la organización desee añadir equipos nuevos a la red, no supondría un problema por carencia de prestaciones. Con esto, la capacidad de procesamiento estimada para el servidor SIEM es la siguiente:

- **RAM (GB):** 32. La estimación exacta son 28GB, pero no existen dispositivos de memoria con un almacenamiento fuera de una potencia de 2. Aunque se pueden incorporar varios módulos de memoria y sumar 28, se ve algo innecesario.

- **CPUs (cores):** 16. Es un punto medio entre la suma total de los componentes por separado (8 + 8 + 4 = 20) y la recomendación inicial de Wazuh (8 CPUs).

Respecto al espacio de disco, se ha realizado un cálculo que tiene en cuenta el número de equipos que dispone la facultad y el tipo que son. Por lo tanto:

- **131 equipos personales.** En base a la información anterior, por cada nodo personal se requieren:

$$0.04\ GB\ server + 1.5\ GB\ indexer = 1.9\ GB\ por\ 90\ días$$

Entonces, si en el despliegue hay 131 equipos:

$$131\ nodos\ \times 1.9\ GB/nodo = 248.9 \approx 250\ GB\ totales$$

- **4 servidores.** En base a la información anterior, por cada nodo servidor se requieren:

$$0.1\ GB\ server + 3.7\ GB\ indexer = 3.8\ GB\ por\ 90\ días$$

Entonces, si en el despliegue hay 4 servidores:

$$4\ nodos \times 3.8\ GB/nodo = 15.2\ \approx 16GB\ totales$$

Con estos cálculos, podemos concluir que, estrictamente, la instancia SIEM/XDR requiere de 250 + 16 = 266 GB de espacio de disco para la instalación y correcto funcionamiento de Wazuh. Igualmente, debe tener más para la propia máquina en sí y otros servicios o programas que necesite implementar. Por eso se ha decidido ampliar este almacenamiento a 500GB. Finalmente, en la Tabla 2 se muestran las especificaciones definitivas para el servidor SIEM/XDR:

	SIEM
CPUs (cores)	16
RAM (GB)	32
Disk Space (GB)	500

Tabla 2. Especificaciones hardware de la instancia SIEM/XDR.

4.4. Servidor IRP

Esta segunda instancia dedicada al análisis de eventos, recordar que está formada por TheHive, Cortex y MISP. Por lo tanto, se han tenido en cuenta las especificaciones individuales de cada una de las fichas de documentación.

Para **TheHive**, los requisitos varían en función de la cantidad de usuarios que van a acceder a la plataforma de manera simultánea. Como en este SOC son 2 trabajadores, las especificaciones son las siguientes:

- Capacidad de procesamiento (< 3 usuarios):
    - RAM (GB): 4 - 8.
    - CPU (cores): 2.
- Almacenamiento de disco. Respecto a este apartado, la documentación oficial no brinda ninguna información. Esta plataforma implementa una base de datos con Cassandra y el motor de indexación Elasticsearch. Esto implica una clara necesidad de alto espacio de almacenamiento. De igual manera, TheHive por defecto recibe todas las alertas que se generan en Wazuh, a no ser que se regule el "*level threshold*" del colector *thehive4py* para que no envíe las alertas con un *alert_level* < *level_threshold*. Dicho parámetro establece, en base al nivel de urgencia que tiene cada alerta de Wazuh, cuáles se recogen en TheHive. Por lo tanto, se ha considerado que el almacenamiento debe ser como mínimo, el mismo que para Wazuh (250GB).

Cortex es el facilitador de motores de análisis para TheHive. A pesar de lo que indica la documentación, los recursos destinados a este servicio son bastante menores (véase más abajo). Igualmente, los requisitos expuestos en la documentación son los siguientes:

- Capacidad de procesamiento:
    - RAM (GB): 16.
    - CPU (cores): 8.
- Almacenamiento de disco. La documentación oficial no aporta nada en este aspecto.

MISP, a diferencia del resto, es un sistema con unos requisitos muy bajos. La mayor carga de trabajo se da al exportar casos los requisitos proporcionados son bastante sencillos:

- Capacidad de procesamiento:
    - RAM (GB): 4 - 8.
    - CPU (cores): 2.
- Almacenamiento de disco. El uso de almacenamiento que hace MISP es sobre el propio sistema de ficheros de la máquina. No necesita una base de datos, a diferencia de los anteriores. Por eso, con 10GB es suficiente.

La implementación que sugiere este trabajo se ha probado y desarrollado en un entorno virtual de las siguientes características: 2 CPUs, 16GB de memoria y 50GB de espacio de almacenamiento. Sospechando de las excesivas peticiones que tiene Cortex, se ha hecho una pequeña prueba en la máquina virtual consultado la memoria que está empleando cada proceso. Mediante el comando

```
ps -o pid,user,%mem,command ax | sort -b -k3 -r
```

se obtuvieron los siguientes resultados:

- Elasticsearch: 53.5%, es decir, **8.565 MB.**
- Cassandra: 27%, es decir, **4.322 MB.**
- TheHive: 5%, es decir, **80 MB.**
- Cortex: 3%, es decir, **48 MB.**

Con estos resultados, podemos concluir que el mayor gasto de memoria viene de parte de la base de datos Cassandra y del *index engine* Elasticsearch. Cierto es que son servicios que emplean tanto TheHive como Cortex, pero se ve claramente que las exigencias, al menos en este entorno de trabajo y caso de uso, son bastante menores. Cabe destacar que la prueba no ha sido bajo pleno rendimiento, pero el equipo host que se emplea para el desarrollo del trabajo no soporta dicha operación. Pese a todo esto, la respuesta de rendimiento será la misma, pero con más o menos carga de trabajo. Cassandra y Elasticsearch son los recursos más exigentes dentro de este sistema.

Por tanto, la estimación final de recursos destinados al servidor IRP concluye de la siguiente manera:

- **RAM (GB):** 32. Como se ha mencionado, el entorno de desarrollo del proyecto cuenta con 16 GB de memoria. Vista la prueba, que no es bajo una carga de trabajo alta, el uso de la memoria casi llega al límite. Además, está claro que cuando el servidor se encuentre bajo una demanda del trabajo normal en el SOC, esos 16GB no serán suficientes. Concluimos entonces con 32GB de memoria para dejar margen, además de no ser un requerimiento excesivamente alto en cuanto a servidores se refiere.
- **CPUs (cores):** 16. Es un servidor que se centra más en la cantidad de memoria para la gestión del almacenamiento y bases de datos, que en la propia potencia de procesamiento.

Respecto al almacenamiento, se ha considerado que 500GB son suficientes, teniendo en cuenta los razonamientos aportados para Wazuh: dejar espacio para los eventos, pero mantener un sobrante para la propia gestión de la máquina. Concluimos entonces en la Tabla 3 que las especificaciones hardware para la instancia IRP son las siguientes:

	IRP
CPUs (cores)	16
RAM (GB)	32
Disk Space (GB)	500

Tabla 3. Especificaciones hardware de la instancia IRP.

CAPÍTULO V.

COSTES Y BALANCE ECONÓMICO

Todas las empresas, cuando desean poner en marcha una nueva implementación o mejora de la infraestructura, elaboran un proyecto y deben definir cada uno de sus pasos para optimizar al máximo los recursos disponibles. Desde el inicio, este Trabajo de Fin de Grado ha querido acercar el proyecto a esta idea. Es decir, abordar los aspectos que un proyecto real debería de asumir y contemplar. Es por eso que, aparte de proporcionar una solución viable para la implementación de un SOC, se brinda un estudio sobre el coste estimado que tendría un sistema de estas dimensiones.

En este capítulo, se aportan diferentes alternativas para la implementación de servidores (tanto físicos como alquilados) que se ajusten a las especificaciones antes mencionadas, además de su precio y gasto energético.

5.1. Servidores físicos

Para la implementación de cualquier sistema de información, una opción es la compra directa de las máquinas que soportan todo el despliegue. Es una opción que poco a poco está cayendo en desuso debido al alto de entrada y poca escalabilidad. Por ejemplo, si se requieren más recursos de los disponibles en la máquina actual, es necesario ampliar con nuevo hardware, cuyo coste suele ser bastante alto, o incluso tener que migrar a una máquina totalmente nueva que se ajuste a las nuevas necesidades. Sin embargo, esto ofrece independencia y control sobre tu información, lo cual es interesante en caso de exigir estas dos premisas. Igualmente, aumenta el coste de mantenimiento y las dificultades técnicas que puedan surgir durante el uso de los servidores.

En base a la información del capítulo anterior (Tabla 2 y Tabla 3), se han consultado varios modelos, y el que mejor se ajustaba a los requerimientos y precio ha sido el equipo Dell EMC PowerEdge R450 Intel Xeon Silver 4314, cuyas características y precio son:

- Frecuencia de procesador: 2.4GHz – 3.4GHz (turbo).
- Núcleos: 16.
- Hilos: 32.
- Caché: 24MB.
- Memoria: 32GB DDR4-SDRAM.
- Almacenaje: 480GB SSD.
- Número de unidades SSD compatibles: 8.
- Fuente de alimentación: 800W
- Precio: 3500€ aprox.

5.1.1. Gasto energético

Cuando se elige una implementación física, un gasto adicional que no se puede obviar es el gasto energético, el coste de operación. El cálculo de dicha métrica no es un proceso sencillo y trivial, pero se ha encontrado una web [16] que permite calcular el consumo energético de un equipo en función de sus componentes. Los resultados no son los exactos debido a que el procesador que integra el servidor Dell EMC PowerEdge R450 no se encontraba entre las opciones disponibles. Igualmente, esto es una estimación donde un margen de 10€ no es significativo. Los parámetros elegidos han sido:

- Motherboard: Server
- Procesador: Intel Xeon D-1537
  - Núcleos: 8
  - Frecuencia: 2.4GHz
- CPU utilization: 90%
- Memoria: 32GB DDR4 Module
- Almacenamiento: SATA SSD
- Monitor: LED 15’
- Tiempo de utilización: 24h, 7 días a la semana.

Se ha obtenido el precio medio del kWh en España a noviembre de 2022. Actualmente, a día 15/11/2022, el precio está a 0.19€/kWh [17]. Por eso, una vez obtenido el gasto energético en vatios (W), podemos multiplicar y obtener el coste final en euros (Figura 9). Este cálculo también lo realiza la web.

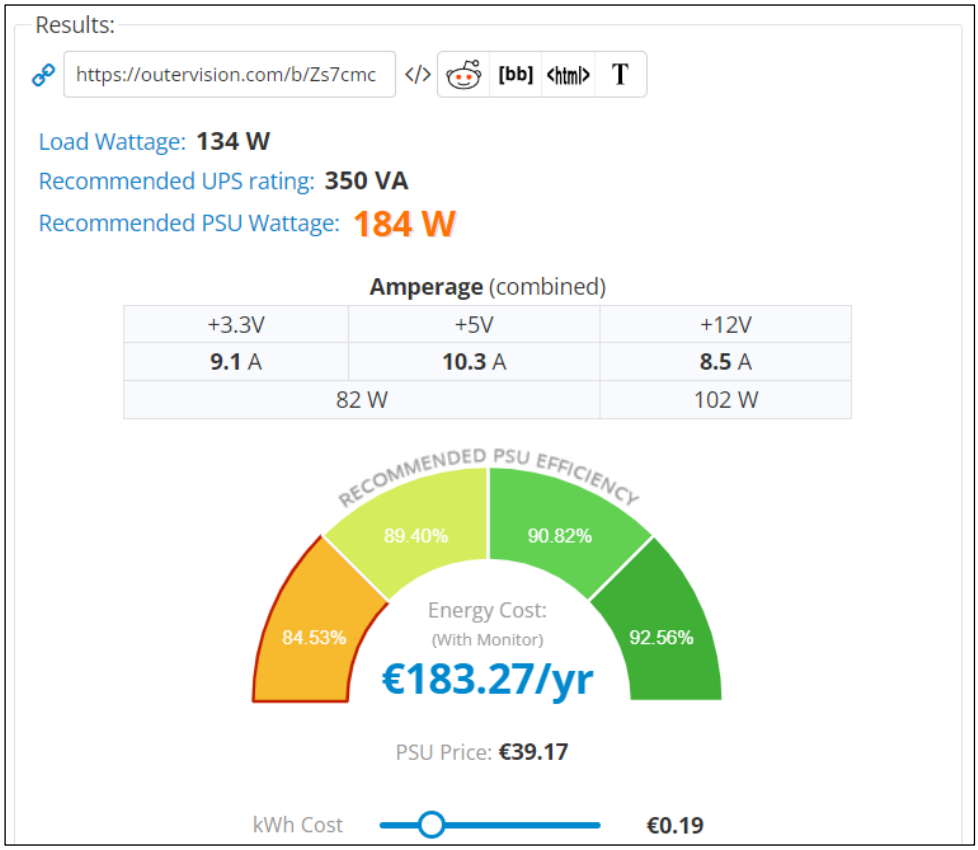

Figura 9. Coste estimado de operación por cada servidor del SOC.

5.2. Hosting

Otra de las opciones posibles para el despliegue del sistema es el alquiler de servidores remotos. Empresas se dedican a coleccionar y desplegar servidores para que clientes alquilen sus recursos en base a las necesidades que requieran. Actualmente, es un método mucho más empleado y viable económicamente que la instalación de los servidores físicos, por los siguientes motivos:

- **Costes**. El precio es más reducido debido a que no se necesita el espacio físico para los aparatos, además de que el gasto eléctrico cae en responsabilidad de la empresa de hosting que se contrate. Se incluye en el precio, pero pueden ofrecernos un coste menor porque al comprar mayores cantidades de energía, el precio del kW baja. Además, si los servidores se encuentran en un país con menor coste energético, el coste puede bajar más.

- **Mantenimiento**. Queda a cargo de la empresa host. Cualquier reparación o falla es responsabilidad suya.

- **Accesibilidad**. Al tener los servidores en remoto, el acceso es mucho más flexible.

- **Control**. Los servicios pueden ser contratados únicamente durante el tiempo requerido. En cambio, el despliegue físico, en caso de no necesitarlo, el gasto ya se hizo y puede que no sea rentable.

- **Seguridad**. El mantenimiento, cuidado y seguridad de estos servidores es mayor a la que una pequeña organización puede aportar a sus máquinas.

- **Almacenamiento**. Existe una mayor flexibilidad en el uso y necesidad del almacenamiento. En el despliegue físico, se necesita buscar el hardware necesario, comprarlo e instalarlo. En cambio, la sencillez de contratar el espacio justamente necesario a la empresa de hosting es más rápido y barato.

Vistas las ventajas, para este proyecto se ha considerado la empresa de hosting Clouding.io [18]. Ofrece instancias VPS configurables al gusto del cliente. Los servidores VPS son servidores virtualizados dentro de una máquina física más grande. En el caso de los servidores planteados para este proyecto, el coste por cada instancia es el siguiente:

	Cantidad	**Coste unidad (€)**	**Coste total (€)**
Licencia	Linux	0	0
RAM (GB)	32	1.25	40
CPUs (cores)	16	2.50	40
SSD NVMe	500	0.10	50
TOTAL (€)			**130.00€/mes**

Tabla 4. Desglose de costes por instancia alquilada en Clouding.io

5.3. Balance final

Ya se han evaluado las dos opciones posibles para el despliegue del SOC. Si hacemos un balance general, obtenemos que:

- La implementación **física** tiene un coste final de:

$$3500€ \times 2\ servidores = 7000€$$

  A esto, debemos añadirle, mensualmente el gasto energético con el precio del kWh. En base a lo calculado en el apartado 5.1.1, se obtuvo que el coste energético por cada uno de los servidores es de 185€/año aproximadamente. Finalmente:

$$Coste\ total = 7000€ + 185€/año$$

- El **alquiler de servidores** remotos mediante la empresa de hosting tiene un coste final de:

$$Coste\ total = 130€/mes \times 2\ servidores = 260€/mes$$

En comparación, el coste de la implementación física supone una barrera de entrada muy alta, a diferencia de la segunda. En cambio, de manera global, es más barata la primera de las opciones porque el alquiler, al cabo de los 27 meses, ya han supuesto 7000€.

$$\frac{7000€}{260€/mes} = 26.92\ meses \approx 27\ meses$$

Ambas tienen sus ventajas e inconvenientes. Pero como conclusión, se ha decidido recomendar la versión de hosting, ya que la barrera de entrada es menor y su retirada o no es más factible. Da posibilidad a corrección o reajuste de los sistemas. Igualmente, la intención de este estudio es mostrar las opciones disponibles y hacer una comparación entre las mismas, de forma que la organización destino disponga de una aproximación o al menos, más información para la toma de decisiones.

CAPÍTULO VI.

CONCLUSIONES Y TRABAJOS FUTUROS

Finalmente, tras el desarrollo completo del proyecto, se cierra el documento con los objetivos y conclusiones obtenidas. Además, se hace un repaso de las posibles líneas de trabajo futuras que mejoren el sistema aquí expuesto.

6.1. Objetivos conseguidos

En la introducción de este trabajo, se definen los objetivos que el proyecto debería de cumplir (sección 1.2). En este apartado, se hace un repaso de estos explicando brevemente las conclusiones obtenidas.

- **Proporcionar mecanismos para la monitorización de registro de logs y de red en equipos finales.**

  Para la consecución de este primer objetivo, la instancia SIEM/XDR es la que brinda una solución. Gracias a Wazuh y al IDS Suricata, el sistema es capaz de recopilar todos los registros y eventos ocurridos en los equipos finales. Concretamente, el indexer de Wazuh (ver 3.3.2) es el componente que permite la clasificación, organización y consulta de los datos.

- **Presentar un interfaz de usuario para la gestión de alertas.**

  El dashboard de Wazuh (ver 3.3.3) y la interfaz gráfica de TheHive en la instancia IRP, permiten un control total de las instancias y sus eventos.

- **Proporcionar mecanismos para el análisis de datos referentes a alertas.**

  Este es otra de las funcionalidades más importantes que necesita implementar un SOC. En la solución de este apartado, el servidor IRP manifiesta su propósito. TheHive se usa íntegramente para el análisis de eventos e inteligencia de amenazas. Junto a Cortex y MISP, forman la pieza que desempeña este trabajo.

- **Incorporar mecanismos para la respuesta activa a incidentes.**

  Este es un extra que dota al SOC de una autonomía para la defensa y respuesta a incidentes, fuera de la que llevan a cabo manualmente los miembros del equipo. Para ello, Wazuh incorpora un sistema en los agentes capaz de definir manualmente estas acciones (ver 3.3.4). Además, la configuración de Suricata de IPS también permite tomar acción frente al tráfico (ver 3.4).

6.2. Trabajos futuros

Durante la implementación del trabajo, muchas de las herramientas desplegadas han ido dando ideas y brindando una serie de aspectos inesperados, los cuales, en caso de añadir, dotarían al sistema de una mayor funcionalidad. Pero, debido a la propia duración y magnitud del TFG, muchas de estas ventajas de las herramientas que forman el SOC no se han podido dominar o considerar para el despliegue de los servidores. De igual manera, pueden dar lugar a vías de trabajo futuras con las que mejorar el proyecto aquí descrito. En este último apartado se hace un breve repaso de ellas para dar constancia y situar al lector sobre cómo podría evolucionar un sistema como este.

- **Integración directa de TheHive con MISP.**

  Recordando que MISP no sólo se trata de un motor de análisis, sino que también es una plataforma de intercambio de inteligencia de amenazas, actualmente el SOC no puede reportar fácilmente sus propios análisis con la comunidad. El proceso es manual y poco ágil. La posibilidad está en integrar MISP con TheHive para poder exportar directamente los casos a MISP y compartirlos con la comunidad de manera automática.

- **Estudio de las necesidades de la organización para la implementación de respuestas activas automáticas.**

  Recordando los módulos de Wazuh, la respuesta activa es una de las piezas más funcionales. Como se ha visto, su integración es un proceso relativamente sencillo, pero depende de muchos factores. El trabajo sería hacer un estudio progresivo de las necesidades del SOC, los tipos de ataques más comunes, las vulnerabilidades que presenta, etcétera. En base a ellos, ir incorporando los comandos y respuestas activas pertinentes para el fortalecimiento de la infraestructura.

- **Implementación de los servicios mediante contenedores de Docker.**

  Docker parte del concepto de que un programa para ejecutarse siempre necesita de dependencias. Por ejemplo, si queremos ejecutar un fichero Python debemos tener instalado Python. El hecho de que una aplicación sea “autocontenida”, implica que trae consigo todos los programas y dependencias necesarias para que funcione. La idea principal de Docker es poder correr una aplicación o proyecto siempre bajo un mismo entorno. Es decir, todas las dependencias que tenga nuestra aplicación van a estar dentro del contenedor. A su vez, este contenedor podrá moverse entre distintas máquinas, ya que va todo junto con sus dependencias.
  Por ello, incorporar esta mejora al SOC, lo dota de mucha portabilidad, la posibilidad de mover servicios entre máquinas sin perder configuraciones.

Referencias y biliografía

[1] «Cyberthreat Defense Report,» Cyber Edge Group, 2022. [En línea]. Available: https://cyber-edge.com/cdr/ [Último acceso: Octubre 2022].

[2] «¿QUÉ ES UN CENTRO DE OPERACIONES DE SEGURIDAD (SOC)?,» ciberseguridad, [En línea]. Available: https://ciberseguridad.com/guias/centro-operaciones-seguridad-soc/ [Último acceso: Octubre 2022].

[3] «¿Qué es un SOC?,» Oracle, [En línea]. Available: https://www.oracle.com/es/database/security/que-es-un-soc.html [Último acceso: Octubre 2022].

[4] C. Pachón, «Qué es un SOC: Funciones y objetivos principales,» nsit, [En línea]. Available: https://www.nsit.com.co/que-es-un-soc-funciones-y-objetivos-principales/ [Último acceso: Octubre 2022].

[5] «¿Qué es un SIEM?,» Fortra, 28 Mayo 2018. [En línea]. Available: https://www.fortra.com/es/blog/que-es-un-siem [Último acceso: Octubre 2022].

[6] A. T. Williams y M. Nicolett, «Improve IT Security With Vulnerability Management,» Gartner Research, 2 Mayo 2005. [En línea]. Available: https://www.gartner.com/en/documents/480703 [Último acceso: Octubre 2022].

[7] B. Hale, «Estimating Log Generation for Security Information Event and Log Management,» Solarwinds, [En línea]. Available: http://content.solarwinds.com/creative/pdf/Whitepapers/estimating_log_generation_white_paper.pdf [Último acceso: Octubre 2022].

[8] «¿Qué es XDR?,» TrendMicro, [En línea]. Available: https://www.trendmicro.com/es_es/what-is/xdr.html [Último acceso: Octubre 2022].

[9] «Incident Response Platform: The Road to Automating IR,» Cynet, [En línea]. Available: https://www.cynet.com/incident-response-services/incident-response-platform-the-road-to-automating-ir/ [Último acceso: Octubre 2022].

[10] INCIBE, «¿Qué son y para qué sirven los SIEM, IDS e IPS?,» INCIBE, 3 Agosto 2020. [En línea]. Available: https://www.incibe.es/protege-tu-empresa/blog/son-y-sirven-los-siem-ids-e-ips [Último acceso: Octubre 2022].

[11] «¿Qué es un sistema de detección de intrusos (IDS)?,» CheckPoint, [En línea]. Available: https://www.checkpoint.com/es/cyber-hub/what-is-an-intrusion-detection-system-ids/ [Último acceso: Octubre 2022].

[12] «Wazuh documentation,» Wazuh, [En línea]. Available: https://documentation.wazuh.com/current/index.html

[13] «MITRE ATT&CK,» MITRE, [En línea]. Available: https://attack.mitre.org/ [Último acceso: Octubre 2022].

[14] «Apache Cassandra: gestión distribuida de grandes bases de datos,» IONOS, 15 Octubre 2020. [En línea]. Available:

https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/apache-cassandra/ [Último acceso: Noviembre 2022].

[15] «¿Qué es Elasticsearch?,» Elastic, [En línea]. Available: https://www.elastic.co/es/what-is/elasticsearch [Último acceso: Noviembre 2022].

[16] «OuterVision® Power Supply Calculator,» OuterVision, [En línea]. Available: https://outervision.com/power-supply-calculator [Último acceso: Noviembre 2022].

[17] «¿Cuánto cuesta el kilovatio hora de luz (kWh) en España?,» TarifaLuzAhora, [En línea]. Available: https://tarifaluzhora.es/ [Último acceso: Noviembre 2022].

[18] Clouding.io, [En línea]. Available: https://clouding.io/ [Último acceso: Noviembre 2022].

[19] A. Benítez, «SOC-Implementation,» GitHub, [En línea]. Available: https://github.com/benited44/SOC-implementation.git

[20] Editorial Nordstern, «Ciberseguridad proactiva: las 10 funciones clave de un SOC,» Nordstern technologies, 13 Enero 2021. [En línea]. Available: https://www.nordsterntech.com/post/ciberseguridad-funciones-de-un-soc [Último acceso: Octubre 2022].

[21] «TheHive documentation,» TheHive, [En línea]. Available: https://docs.thehive-project.org/thehive/

[22] «Cortex documentation,» Cortex, [En línea]. Available: https://docs.thehive-project.org/cortex/

[23] «MISP documentation,» Malware Information Sharing Platform, [En línea]. Available: https://www.misp-project.org/documentation/

[24] AbuseIPDB, [En línea]. Available: https://www.abuseipdb.com/

ANEXO I

MEMORIA DE PRÁCTICAS

Creación SOC “low cost” de seguridad industrial.
“Low cost” industrial security SOC creation.

Información del estudiante

- Nombre y apellidos: Alejandro Benítez Tocino
- Titulación: Grado de Ingeniería Informática
- Curso Académico: 2020-2021
- Fecha de Presentación: 27-08-2021
- Entidad Colaboradora: Osane Consulting, SL.
- Lugar: Hermanos Lumiere, 11 Edificio Central, Miñano.
- Fecha inicio: 12-07-2021
- Fecha finalización: 6-08-2021

Tutores

- Tutor académico: Esteban Stafford Fernández.
- Tutor de la empresa: Lorenzo Díaz de Apodaca.

Resumen (castellano)

El contenido de estas prácticas se basa en la implementación de un SOC (Security Operations Center) de seguridad industrial. Su función es monitorizar la red de la empresa en la que se contrate, usando únicamente programas de código abierto. Permitirá a cualquier pequeña o mediana empresa el monitoreo de la red empleando un único analista, pues toda la información se centraliza dentro de la misma máquina, con un programa que se llama Wazuh.

El diseño ha sido facilitado por mi jefe de proyectos, y por ello sólo me tuve que ocupar de la creación en sí. Todo el trabajo lo he realizado de manera autónoma, pues no estaba en ningún equipo.

Ha sido una experiencia de puro aprendizaje. Pese a no haber acabado el proyecto, debido a su alta complejidad, he conseguido ver muchos detalles interesantes y sobre todo perspectiva. Sé qué programas se usan a nivel industrial, he conocido el ámbito de la ciberseguridad y sus programas, tipos de amenazas, he revisado mucho en las documentaciones, visto conferencias, etc. En resumen, he conocido mejor de primera mano lo que podría ser trabajar en una empresa, pues cuando tienes dudas no puedes ir directamente a preguntar para que te lo resuelvan, hay que investigar por uno mismo. Es decir, he trabajado como si estuviera en una empresa, sin ninguna ayuda de tipo académico como unos apuntes o un profesor. Además, creo que el crear un SOC es una

tarea muy interesante para hacer en unas prácticas, pues toca muchos temas y partes de la informática.

Resumen (inglés)

The purpose of this practice is the implementation of an industrial security SOC (Security Operations Center), which tries to monitorize the business network, using only Open Source programs. This will allow any small or medium business to control their network traffic and check if everything is working correctly. Also, this would be done by one analist, because all the information arrives to a single computer, using Wazuh.

The design was handed to me by my superior. So my occupation was only to implement the SOC. All has been done autonomously, as I did not belong to any team.

It has been an experience of intense learning. Despite of not having finished my job, I have learnt many details about working in an enterprise, but perspective above all. Now, I know which programs are used in the industrial context, I have seen much more about the cybersecurity world, their programs, types of threats, I have researched in official documentation and I have seen a lot of conferences. In summary, I have experienced how it could be to work in a real company. When you have a problem, you can not ask immediately to your superior, you have to learn by yourself.

Also, I think making a SOC has been the perfect task, because it has a little of everything: software, cybersecurity, administration of information systems, etc.

Descripción tareas, trabajos desarrollados y departamentos asignados

Como planteamiento inicial, mostraré lo que se me otorgó como guía inicial para llevar a cabo la tarea. Son dos esquemas: la Figura 10 muestra los programas empleados correspondientes a cada fase de la detección y solvencia de las amenazas, y la Figura 10 es un diagrama de flujo mostrando el funcionamiento que debía tener el SOC.

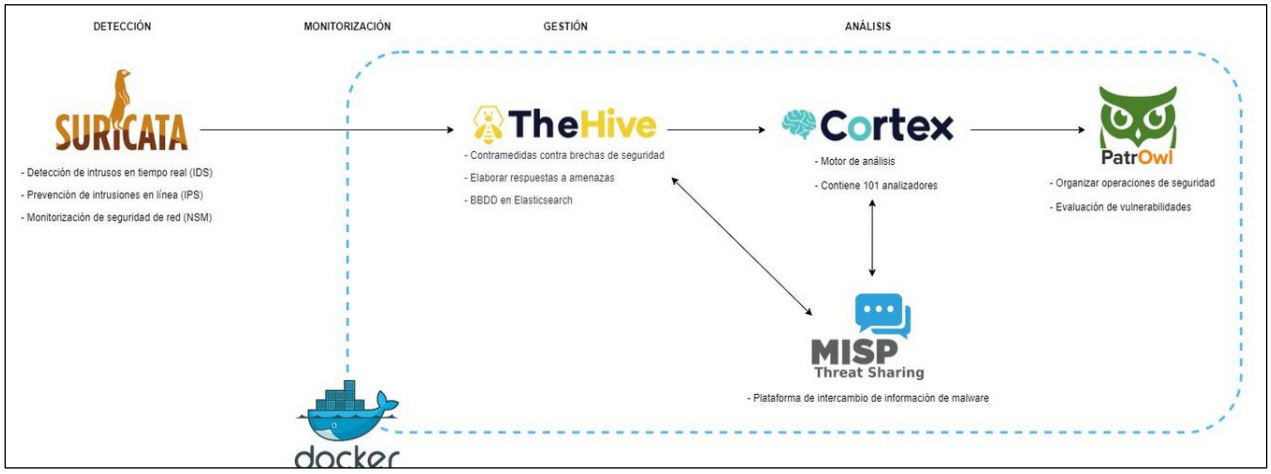

Figura 10. Esquema de los componentes en el SOC.

El funcionamiento general es: Suricata detecta las anomalías de la red, ya que cuenta con un conjunto de reglas (modificables) que juzgan qué actividad debe ser detectada o no. Estos logs son enviados a Wazuh y a su vez enviados a TheHive. TheHive lo que hace es procesar estos IOC (Indicator of Compromise). Para ello se apoya en MISP y Cortex: con MISP se comparte y consulta la comunidad de desarrolladores para ver si ese problema ha sido resuelto con anterioridad y así aprovecharse, y con Cortex se analiza dicha amenaza, ya que Cortex emplea casi cien analizadores, como por ejemplo VirusTotal. Finalmente, PatrOwl es un último analizador que emplea hiperautomatización (Inteligencia Artificial).

De manera esquemática y algo simplificada, el funcionamiento y coordinación de las máquinas que forman este sistema es el siguiente:

- Wazuh tiene el rol de director general y administrador de la información. Todo lo que ocurra en la red (monitorización) será reflejado por él, para que el analista tenga una única fuente desde la que controlar todo. Tiene 2 bloques, el servidor y el almacenamiento del mismo. Esto lo he repartido en 2 MVs, pero únicamente por temas de rendimiento, no es estrictamente necesario.

- Los encargados de captar la información, intrusiones, actividades anómalas o sospechosas para luego procesarla, enviar y recibir reportes, etc. serán todos los contenedores de Docker. Cada uno tiene una misión que en conjunto con el resto, forman un sistema sólido de monitorización de red. Según vayan captando la información, logs, etc. serán enviados al servidor de Wazuh.

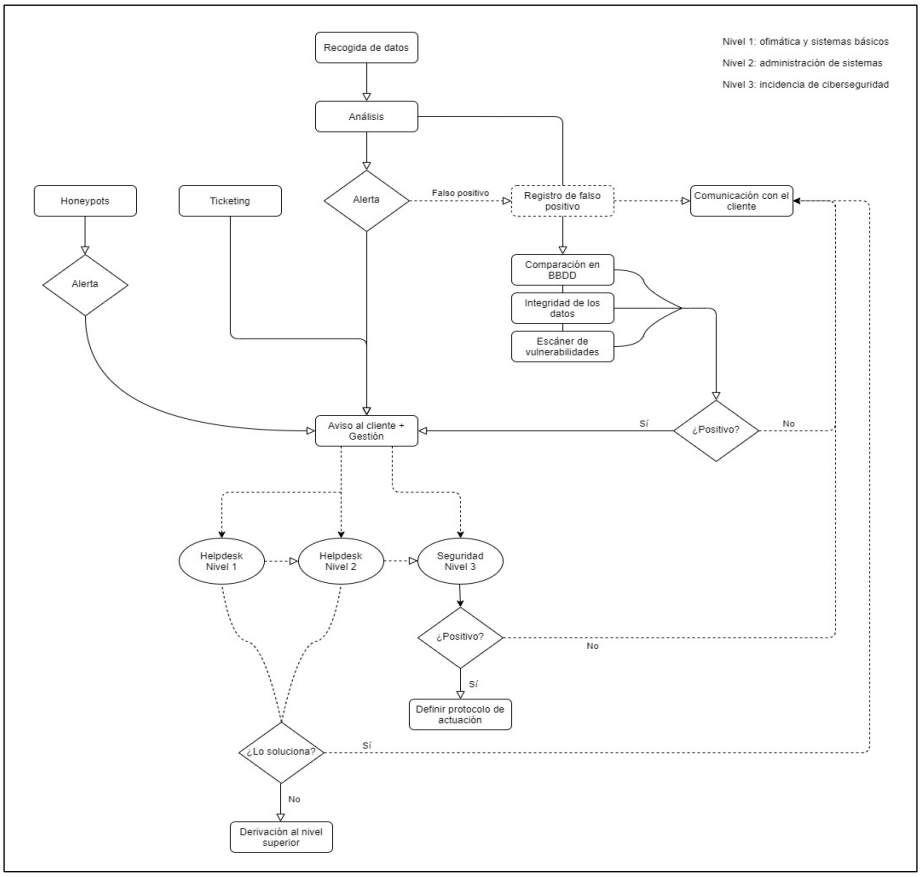

Figura 11. Diagrama de flujo para el funcionamiento del SOC.

Una vez con esta pequeña guía inicial, procedí con la formación teórica sobre la materia: saber qué es un SOC, sus partes, método de trabajo, número de personas que lo forman, hasta qué ámbito alcanza, etc. Para ello vi varias conferencias en Internet y poco a poco fui captando la idea y viendo las similitudes con mi diseño a realizar, identificando partes y viendo qué formas había de hacerlo.

Además, el resto de los programas del esquema también me eran desconocidos. Por lo tanto, tuve que formarme en ellos teóricamente, de la misma forma que hice con el SOC. Algunos fueron más sencillos de aprender y encontrar, pero por ejemplo Docker, es muy denso y tiene una cantidad de contenido abrumadora. Da para hacer un curso entero.

Efectivamente comencé por Docker, ya que la vi la más general y por alguna tenía que empezar. Docker parte del principio de que un programa para ejecutarse, suele necesitar otros muchos para funcionar, como por ejemplo Java, Python, etc. Y en este caso, los contenedores de Docker, son como "paquetes" que llevan todo lo necesario para que un programa se ejecute correctamente en cualquier máquina, sin tener que irse a una máquina virtual que es algo mucho más pesado y complejo de mover. Es una herramienta muy empleada en la informática ya que da mucha portabilidad a los programas.

Para cada herramienta hice esto mismo. Además, documenté toda la información que iba acumulando. Tengo un fichero para cada una de ellas, explicando en qué consiste, qué hace, y demás notas interesantes. También he documentado todos los términos o conceptos que iba aprendiendo. Respecto al proceso de creación absoluto del SOC, también está documentado. En él, detallo paso por paso cada etapa de creación del mismo. Incluso hice un diario de trabajo, donde fui redactando la experiencia en las prácticas: horario seguido, problemas que iban surgiendo, ideas, etc.

La primera semana consistió completamente en la formación teórica, y una vez me sentí cómodo y con conocimiento de cada una de las cosas importantes, comencé la implementación real del sistema. Es un sistema formado por 3 máquinas virtuales:

- **MV1**. Wazuh-Server: monitorización general del sistema. Con ella podremos acceder, desde el navegador del host, al servidor web de Wazuh donde recibiremos toda la información recaudada por Suricata, MISP, Cortex, TheHive, etc. Dicho servidor, una vez configurado, tendrá una dirección IP la cual será accesible desde cualquier navegador. Será con lo que trabaje el analista.

- **MV2**. ElasticSearch con Kibana. El servidor de Wazuh necesita un soporte de almacenamiento, una base de datos. Por temas de rendimiento, es mucho más eficiente que dicho sistema esté en una máquina aparte. Kibana será la herramienta con la que podremos visualizar de manera cómoda la información de ElasticSearch.

- **MV3**. Contenedores Docker. Instalaremos Docker, crearemos una red donde estarán todos los contenedores, es decir, las herramientas TheHive, MISP, Cortex, PatrOwl, ticketing y honeypots.

Una vez introducido el esquema general y funcionamiento, indico y comento los pasos que seguí para la creación absoluta del sistema.

1. **Crear MVs.** Para este paso empleé VirtualBox, ya que VMWare me pareció menos cómodo y más lioso. Al estar acostumbrado a VBox, me es una opción más sencilla. Además de que es algo que no influye en el producto final.

2. **Instalación SSOO.** Para cada MV lo mejor es instalar *Ubuntu minimal 18.04 LTS*, ya que es ligera y no necesitamos el entorno gráfico para nada. Comentar que para cada una de las máquinas virtuales, he habilitado el uso de SSH para poder trabajar desde la consola de Windows.

3. **Configuración de red.** Tuve en cuenta que para que las máquinas estén dentro de la misma red debemos configurarlas como una red NAT.

4. **Instalación Wazuh-Server (MV1).** El procedimiento seguido fue el que indican ellos desde su página web.

5. **Instalación Filebeat (MV1).** Envía los logs al motor de ElasticSearch.

6. **Instalación ElasticSearch (MV2).** Empleando la documentación oficial.

7. **Instalación Kibana (MV2).** Empleando la documentación oficial.

8. **Instalación Docker (MV3).** Documentación oficial.

9. **Configuración previa de los contenedores.** Los contenedores de Docker generan unos volúmenes internos que una vez se apaga la máquina, son borrados, y por tanto, los cambios que les hagamos también. Una solución para esto es crear nuestros propios volúmenes (almacenados en el host) con la configuración que deseamos y cuando vayamos a correr el contenedor, montarle dichos volúmenes.

10. **Creación del fichero *docker-compose.yml***. Para correr todas las herramientas a la vez, y no tener que iniciarlas una a una cada vez que iniciamos el sistema, lo más coherente es emplear docker-compose, que automatiza el inicio de todos los contenedores con la configuración deseada. Depende de un fichero que determina el arranque de cada contenedor (comandos adicionales, configuración, direcciones IP, etc.). Tenemos que crearlo y determinar ahí el arranque de todos los contenedores del sistema.

11. **Configuración de las herramientas.** Una vez tenemos el docker-compose.yml creado y correcto y los contenedores son visibles entre ellos, debemos orquestarlo todo con Wazuh. La clave está en saber que cada contenedor tiene un fichero de configuración completo tipo YAML donde se detalla cada aspecto de su funcionamiento. Se debe pues acceder a cada uno de ellos y determinar que la salida de sus datos sea el servidor de Wazuh que creamos anteriormente. Así, todo será visible desde un mismo lugar.

12. **Puesta en marcha del SOC.** Arranque de todos los contenedores, acceso desde el navegador a Wazuh, y comenzar a monitorear la red.

Comentar que los tres últimos pasos no he sido capaz de completarlos. Aparte de por tiempo, cada vez los problemas que surgían eran más complejos y notaba que superaban con creces mi nivel.

Valoración de las tareas y conocimientos adquiridos

Mi experiencia durante las prácticas me ha parecido muy interesante. Ya que como no tenía la necesidad obligada de terminar el proyecto, pude disfrutar del aprendizaje y apreciar los detalles que iba observando.

Pese a no haber trabajado en equipo o terminado el SOC, creo que lo que me llevo es lo que más vale. En la informática se es muy autodidacta y el tener que lidiar con todos los problemas que te van surgiendo son el pan de cada día. Eso es lo que he tenido yo, cientos de problemas a diario que tienes que ir lidiando para poder avanzar. También he visto qué tipo de programas se usan en la ciberseguridad, he tenido que pegarme mirando en documentaciones oficiales, mirar logs del sistema para detectar errores que a primera vista son irracionales, valorar si el problema es debido al rendimiento de mi máquina, o si está relacionado con la comunicación entre los programas, etc.

Además, mi tutor ya sabía que esta tarea era muy compleja y que seguramente no la llegase a acabar. Simplemente querían que realizase un primer intento de prueba de concepto, indagase sobre el funcionamiento de las diferentes herramientas y documentase mi experiencia lo mejor posible para que mi esfuerzo pudiera ser de provecho para la empresa.

Relación de los problemas planteados y el procedimiento seguido

Dependiendo del problema planteado se seguía una solución u otra, pero lo que más he tenido que hacer ha sido investigar por las documentaciones de los programas, foros y los logs. No sabía que los logs eran una cosa tan útil, usada y densa para obtener informes de problemas y con ello, resolverlos. Es una tarea a veces muy costosa y traumatizante. Algunos de los problemas más relevantes han sido:

- Al reinstalar tantas veces las MVs, a veces daba errores al conectarse por SSH pese haber seguido todos los pasos. Es debido a que el proceso de instalación de SSH crea un nuevo par de llaves cada vez. El cliente de SSH recuerda la llave pública de cada servidor al que se conecta y avisa con un error si esta cambia. En el mensaje de error aparece lo que hay que hacer: irse al fichero `.ssh` con la ruta que indica, y borrar la línea que te dice que falla.

- Conexión por SSH a las MVs. El reenvío de puertos que se configura dentro de VBox era algo nuevo para mí y tuve que ahondar bastante y entender bien cómo funcionaban las direcciones IP en ese apartado. Pero al final conseguí solucionarlo.

- Comunicación entre las MVs. El poder conectarme por SSH y a la vez poder hacer un ping entre las máquinas fue un proceso muy tedioso. Hay múltiples modos de conexión en VBox y al final conseguí dar con el correcto: red NAT.

- El servicio *elasticsearch.service* de la MV2 conseguía que arrancase. Pero si reiniciaba la máquina, el servicio no arrancaba. Era algo demasiado extraño y dentro de los logs no encontraba nada coherente. Al final mi jefe me dijo que le quería echar un vistazo. Y hasta día de hoy sigo sin saber qué problema era.

Identificación de las aportaciones que en materia de aprendizaje que han supuesto las prácticas

Como digo, en la creación del SOC he tocado muchos de los ámbitos y varias de las asignaturas cursadas durante la carrera:

- Redes para que las MVs se puedan comunicar entre ellas y a la vez con Internet, organizarlas en una red NAT para que a la vez pueda trabajar desde la terminal de Windows con SSH.

- Redes para la configuración y estructura de los contenedores de Docker en los que se almacenaba cada una de las herramientas del SOC (Suricata, TheHive, ElasticSearch, MISP, etc.).

- Sistemas informáticos para toda la instalación de los SSOO y la gestión total del trabajo, pues sólo he empleado una distribución mini de Ubuntu 18.04 LTS sin entorno gráfico.

- Software para entender y usar Docker, una herramienta muy útil para la portabilidad de programas entre máquinas, sin llegar a ser tan pesadas como una máquina virtual.

- He creado un fichero para cada programa que he necesitado donde tomaba apuntes y documentaba todos los aspectos que iba aprendiendo. Además, tengo otro en el que he anotado todos los términos o conceptos interesantes que antes desconocía.

- He descubierto y tanteado con los programas Wazuh, VMWare, Suricata, ElasticSearch, TheHive, MISP, Cortex, PatrOwl, Docker, T-POT, Kibana y programas de sistemas de tickets. Para cada uno de ellos tuve que formarme desde cero para poder comenzar a usarlos y saber cómo encajarlos dentro del esquema de funcionamiento del SOC.

ANEXO II

GLOSARIO DE TÉRMINOS

- **Observable:** Cadena de caracteres de la que podemos tener sospecha acerca de comportamiento o actividad maliciosa: direcciones IP, hashes, direcciones de correo electrónico, nombres de dominio, URL, etc.

- **Falso positivo:** Suceso que se da cuando el agente, por error, identifica como malware un fichero que es legítimo e inofensivo.

- **IoC (“*Indicator of Compromise*”):** Dato que funciona como evidencia que nos permite confirmar que nuestro equipo fue atacado con éxito, pero también es información que puede ser utilizada para prevenir futuros ataques. Dicho esto, un indicador de compromiso puede venir dado bajo la forma de un nombre de archivo, el nombre de un proceso en el administrador de tareas, una URL o dirección IP, comportamiento anómalo en el tráfico web, intentos de inicio de sesión fallidos, entre otros tantos más.

- **Rootkit:** Conjunto de software que permite un acceso de privilegio continuo a un ordenador pero que mantiene su presencia activamente oculta al control de los administradores al corromper el funcionamiento normal del sistema operativo o de otras aplicaciones.

- **EPS (“*Events per Second*”):** Es una medida que hace referencia al número de eventos log que se generan en un sistema por segundo.

$$EPS = \frac{\#\ eventos\ del\ sistema}{Periodo\ de\ tiempo\ en\ Segundos}$$

- **ATT&CK MITRE (“*MITRE Adversarial Tactics, Techniques, and Common Knowledge*”):** Base de datos facilitada por MITRE, globalmente accesible que concentra el conocimiento detallado de las tácticas y técnicas empleadas por criminales basadas en observaciones del mundo real.

ANEXO III

GUÍA DE IMPLEMENTACIÓN

Tabla de contenido

Índice de figuras

Introducción

Como se explica en el desarrollo del documento, la instalación mostrada en este anexo está hecha para 2 servidores diferentes, los cuales se referenciarán como:

- Servidor 1: **Server_SIEM**
- Servidor 2: **Server_IRP**

Las direcciones IP de cada uno de ellos son libres, puesto que dependerá de la subred en la que se encuentren. Igualmente se referirá a ellas como ***Server_SIEM_IP*** y ***Server_IRP_IP***. No necesitan ser accesibles desde internet, ya que su gestión y uso se lleva a cabo desde una intranet. Son direcciones privadas y ambos servidores deben verse entre ellos, ya que el SIEM envía las alertas a TheHive.

Notas

Como detalles a la hora de implementar el SOC, tendremos en cuenta que:

- Todos los comandos son ejecutados con permisos de administrador.
- Los ficheros de referenciados en esta guía, necesarios para el proceso de implementación, se encuentran en el repositorio de GitHub [19].

Tabla de usuarios

Durante el proceso de instalación, hay varios pasos de creación de usuarios. Se encuentran resumidos a continuación, en la Tabla 5:

	Rol	**Usuario/login**	**Contraseña**
Wazuh	Default	admin	
TheHive	Default	admin@thehive.local	secret
	Organisation	Analysis_SOC_UC	
	Org-admin	admin@unican.es	
	analyst	analyst-1@unican.es	
Cortex	superadmin	admin@unican.es	
	Organisation	UC_Analyst	
	read, analyze, org-admin	org-admin1@unican.es	
MISP	Default	admin@admin.test	admin
	admin	admin@unican.es	
	User	analyst@unican.es	

Tabla 5. Usuarios empleados para durante el desarrollo del SOC.

Instalación y configuración del SIEM: Componentes centrales de Wazuh (SIEM server)

1.1 Configuración de dirección IP estática

Debido a que es un servidor, el acceso al mismo debe ser rápido y de referencia única. Además, los agentes deben tener configurada la dirección del servidor al que enviar la información.

Primero, editamos la configuración del servicio *netplan* y añadimos el siguiente fichero `/etc/netplan/01-netcfg.yaml`. Lo editamos según el Código 1.

```
➔ vim /etc/netplan/01-netcfg.yaml
```

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      addresses:
        - <Server_SIEM_IP>/24
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
      routes:
        - to: default
          via: <IP_GATEWAY>
```

Código 1. Configuración netplan para IP estática.

```
➔ netplan apply
```

1.2 Instalación de componentes

Antes que nada, debemos descargar y ejecutar el asistente de Wazuh.

```
➔ curl -sO https://packages.wazuh.com/4.3/wazuh-install.sh
  && sudo bash ./wazuh-install.sh -a
```

A continuación, guardamos las credenciales de acceso obtenidas para posterior uso.

.
.
INFO: --- Summary ---
INFO: You can access the web interface https://<wazuh-dashboard-ip>
User: admin
Password: <ADMIN_PASSWORD>
INFO: Installation finished.

Finalmente, ya podemos acceder al servidor desde nuestro navegador e iniciar sesión con las credenciales anteriores.

https://<Server_SIEM_IP>

1.3 Configuración del *manager* para permitir el registro de nuevos agentes con autenticación

Ejecutamos los siguientes comandos para la gestión de los permisos.

```
➔ grep "<use_password>" -B7 -A8 /var/ossec/etc/ossec.conf
➔ sed -i 's/<use_password>no/<use_password>yes/'
  /var/ossec/etc/ossec.conf
➔ grep "<use_password>" -B7 -A8 /var/ossec/etc/ossec.conf
➔ echo "please123" > /var/ossec/etc/authd.pass
```

Tras esto, reiniciamos el servicio.

```
➔ systemctl restart wazuh-manager
```

Vemos que funciona comprobando la escucha de los puertos.

```
➔ netstat -natp | egrep "(:1514|:1515)"
```

Instalación y configuración del servidor de análisis #1: TheHive (IRP server)

2.1 Configuración de IP estática

Ver paso 1.1

2.2 Instalación de Java Virtual Machine

```
➔ apt install -y openjdk-8-jre-headless
➔ echo JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64" >>
  /etc/environment
➔ export JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
```

2.3 Instalación de ElasticSearch

ElasticSearch es un motor de búsqueda y analítica distribuido, gratuito y abierto para todos los tipos de datos, incluidos textuales, numéricos, geoespaciales, estructurados y no estructurados. En nuestro sistema, cumplirá la búsqueda sobre la base de datos implementada con Cassandra.

Primero debemos añadir el repositorio al sistema.

```
➔ curl -fsSL https://artifacts.elastic.co/GPG-KEY-
  elasticsearch | sudo apt-key add -
➔ echo "deb https://artifacts.elastic.co/packages/7.x/apt
  stable main" | sudo tee -a
  /etc/apt/sources.list.d/elastic-7.x.list
```

Actualizamos repositorios con los nuevos paquetes disponibles

```
➔ apt update && apt upgrade
```

Instalamos Elasticsearch

```
➔ apt install elasticsearch
```

Habilitamos y probamos que está escuchando en el puerto correspondiente 9200:

```
➔ systemctl start elasticsearch
➔ systemctl enable elasticsearch
➔ netstat -an | grep 9200
```

2.4 Instalación y configuración de Apache Cassandra

Apache Cassandra es un **sistema de gestión de bases de datos** (DBMS) de código abierto para bases de datos muy grandes, pero estructuradas.

Primero debemos añadir el repositorio al sistema.

```
➔ curl -fsSL https://www.apache.org/dist/cassandra/KEYS |
  sudo apt-key add –
➔ echo "deb http://www.apache.org/dist/cassandra/debian 311x
  main" | sudo tee -a
  /etc/apt/sources.list.d/cassandra.sources.list
```

Actualizamos repositorios con los nuevos paquetes disponibles

```
➔ apt update && apt upgrade
```

Instalamos Cassandra

```
➔ apt install cassandra
```

Mediante una consulta CQL, modificamos el nombre del *cluster* y la *key*

```
➔ cqlsh localhost 9042
➔ UPDATE system.local SET cluster_name = 'thp' where
  key='local';
➔ exit;
```

Aplicamos cambios

```
➔ nodetool flush
```

Configuramos Cassandra modificando en base a los parámetros mostrados en el Código 2.

```
➔ vim /etc/cassandra/cassandra.yaml
```

```
cluster_name: 'thp'
listen_address: 'localhost' # address for nodes
rpc_address: 'localhost' # address for clients
seed_provider:
      - class_name: org.apache.cassandra.locator.SimpleSeedProvider
parameters:
    # Ex: "<ip1>,<ip2>,<ip3>"
      - seeds: '127.0.0.1' # self for the first node
data_file_directories:
    - '/var/lib/cassandra/data'
commitlog_directory: '/var/lib/cassandra/commitlog'
saved_caches_directory: '/var/lib/cassandra/saved_caches'
hints_directory:
      - '/var/lib/cassandra/hints'
```

Código 2. Configuración de Apache Cassandra para TheHive.

```
➔ systemctl start cassandra
➔ systemctl enable cassandra
```

Comprobamos que funciona viendo los puertos de escucha

```
➔ netstat -an | grep 7000
```

2.5 Instalación y configuración de TheHive

Primero debemos añadir el repositorio al sistema.

```
➔ curl https://raw.githubusercontent.com/TheHive-
  Project/TheHive/master/PGP-PUBLIC-KEY | sudo apt-key add –
➔ echo 'deb https://deb.thehive-project.org release main' |
  sudo tee -a /etc/apt/sources.list.d/thehive-project.list
```

Actualizamos repositorios con los nuevos paquetes disponibles

```
➔ apt update && apt upgrade
```

Instalamos TheHive

```
➔ apt install thehive4
```

Configuramos editando el correspondiente fichero de en base al Código 3.

```
➔ vim /etc/thehive/application.conf
```

```
http.address=<Server_IRP_IP>
http.port=9000

## Database configuration
db.janusgraph {
  storage {
    ## Cassandra configuration
    # More information at
https://docs.janusgraph.org/basics/configuration-
reference/#storagecql
    backend: cql
    hostname: ["127.0.0.1"]
    # Cassandra authentication (if configured)
    username: "thehive"
    password: "password"
    cql {
      cluster-name: thp
      keyspace: thehive
      local-datacenter: datacenter1
      read-consistency-level: ONE
      write-consistency-level: ONE
    }
```

```
index.search {
    #backend: lucene
    #directory: /opt/thp/thehive/index
    # If TheHive is in cluster ElasticSearch must be used:
    backend: elasticsearch
    hostname: ["127.0.0.1"]
    index-name: thehive
  }

  ## For test only !
  # Comment the two lines below before enable Cassandra database
  #storage.backend: berkeleyje
  #storage.directory: /opt/thp/thehive/database
  // berkeleyje.freeDisk: 200 # disk usage threshold
}

## Attachment storage configuration
storage {
  ## Local filesystem
  provider: localfs
  localfs.location: /opt/thp/thehive/files

  ## Hadoop filesystem (HDFS)
  #// provider: hdfs
  #// hdfs {
  #//   root: "hdfs://localhost:10000" # namenode server hostname
  #//   location: "/thehive"           # location inside HDFS
  #//   username: thehive              # file owner
  #// }
}
.
.
.
# Define maximum size of attachments (default 10MB)
play.http.parser.maxDiskBuffer: 1GB
```

Código 3. Configuración en TheHive.

Cambiamos permisos de usuario sobre los ficheros

```
➔ chown thehive:thehive -R /opt/thp/thehive/index
➔ chown thehive:thehive -R /opt/thp/thehive/files
```

Reiniciamos servicio y comprobamos escucha

```
➔ systemctl start thehive
➔ systemctl enable thehive
➔ netstat -an | grep 9000
```

Nos conectamos al servidor desde el navegador con las siguientes credenciales predeterminadas:

https://<Server_IRP_IP>:9000/
USUARIO: admin@thehive.local
CONTRASEÑA: secret

2.6 Set up de la consola de TheHive

El acceso que hemos realizado es un usuario de administración por defecto. Por lo tanto, no viable para la gestión correcta y diaria del sistema. Tenemos que crear la nueva organización que queramos emplear y luego definir en ella la cantidad de usuarios que queramos en el sistema.

En la interfaz de TheHive, seleccionaremos la pestaña de *Admin* > *Organisations* (véase la Figura 12).

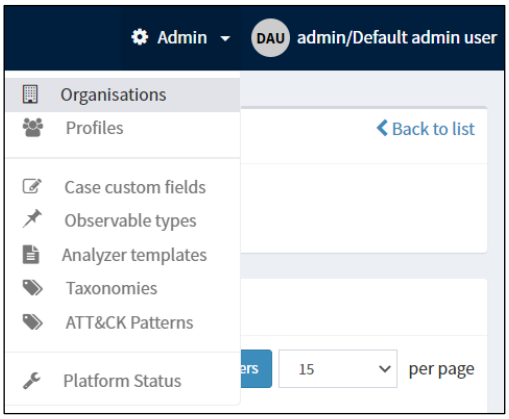

Figura 12. Vista del menú de administrador en TheHive.

A continuación, pincharemos en *"+ New Organisation"* y rellenaremos la información requerida:

- Nombre: Analysis_SOC_UC
- Descripción: Analyst organisation for the UC.

Luego, una vez dentro de la organización, añadiremos un par de usuarios nuevos. Uno encargado de la administración general del sistema, y otro destinado únicamente al análisis.

Usuario 1:

- Login: admin@unican.es
- Name: <Nombre_del_administrador>
- Profile: org-admin

Usuario 2:

- Login: analyst-1@unican.es
- Name: <Nombre_del_analista>
- Profile: analyst

Tras esto, podemos establecer una nueva contraseña en cada usuario, dándole a *"New password"*. Así, finalmente salimos de la sesión e iniciamos con el perfil que se desee.

Integración de Wazuh con TheHive (SIEM and IRP server)

3.1 Generación de API key en TheHive (IRP server)

Primero vamos a la lista de usuarios de nuestra organización. Seleccionamos el usuario analista y pinchamos la opción "*Generate API key*". Una vez mostrado, la guardamos para posterior uso.

3.2 Configuración del Wazuh Manager (SIEM server)

Ahora es turno de configurar Wazuh. En el Wazuh manager, lo primero es instalar el módulo de Python "*thehive4py*" para TheHive.

➔ `sudo /var/ossec/framework/python/bin/pip3 install thehive4py==1.8.1`

Ahora, añadimos un programa para la integración de Wazuh con TheHive llamado `/var/ossec/integrations/custom-w2thive.py`

(Ver /etc/ossec/integrations/custom-w2thive.py)

Seguidamente, creamos un script `custom-w2thive.py` que ejecutará debidamente el código Python del paso anterior

(Ver /var/ossec/integrations/custom-w2thive)

Cambiamos los permisos de ejecución y de usuario

➔ `chmod 755 /var/ossec/integrations/custom-w2thive.py`
➔ `chmod 755 /var/ossec/integrations/custom-w2thive`
➔ `chown root:wazuh /var/ossec/integrations/custom-w2thive.py`
➔ `chown root:wazuh /var/ossec/integrations/custom-w2thive`

Ejecutamos el script

➔ `cd /var/ossec/integrations/`
➔ `sed -i -e 's/\r$//' custom-w2thive.sh`
➔ `./custom-w2thive`

Ahora, añadimos la siguiente sección a la configuración de Wazuh e integrar la *API key* del usuario integrado

➔ `vim /var/ossec/etc/ossec.conf`

```
<ossec_config>
…
  <integration>
    <name>custom-w2thive</name>
    <hook_url>http://Server_IRP_IP:9000</hook_url>
    <api_key><USER_KEY></api_key>
    <alert_format>json</alert_format>
  </integration>
…
</ossec_config>
```

Código 4. Configuración de Wazuh para la integración con TheHive.

Reiniciamos el servicio para aplicar los cambios

➔ `systemctl restart wazuh-manager`

Ya deberíamos poder acceder a TheHive con el usuario integrado. Como se puede ver en la Figura 13, este usuario recibirá las alertas captadas por el *alert feeder* de TheHive *thehive4py*.

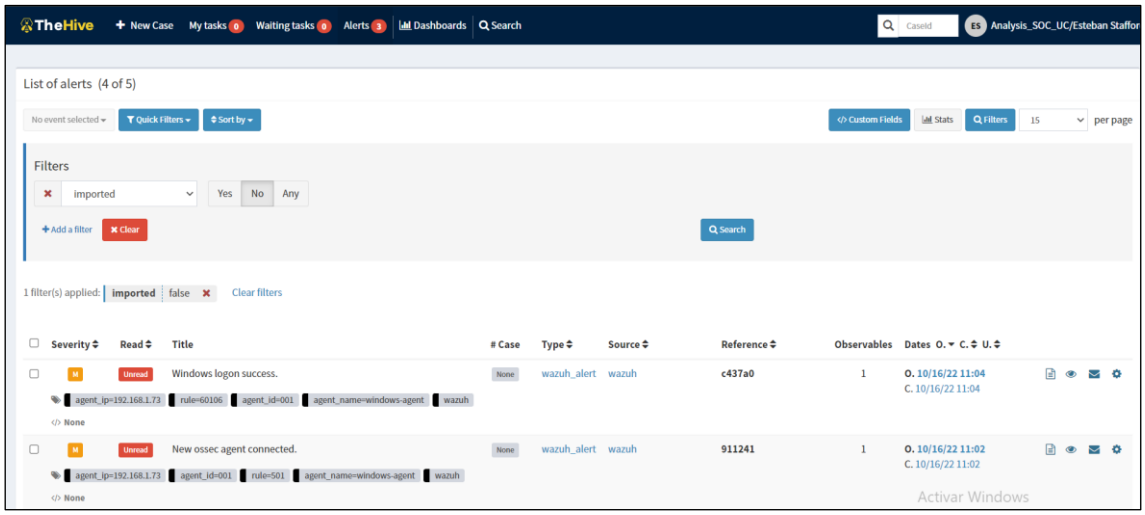

Figura 13. Vista de TheHive con alertas de Wazuh.

Instalación y configuración del servidor de análisis #2: Cortex (IRP server)

4.1 Configuración de Elasticsearch

Cortex también emplea Elasticsearch como motor de búsqueda para la base de datos. Por lo tanto, sólo deberemos ampliar su configuración para su efecto sobre Cortex.

Vamos a la configuración y editamos los campos referenciados en el Código 5.

➔ `vim /etc/elasticsearch/elasticsearch.yml`

```
# ---------------------------------- Cluster -----------------------
#
cluster.name: hive
#
# ------------------------------------ Node ------------------------
node.name: node1
#
# ---------------------------------- Paths -----------------------
#
# Path to directory where to store the data (separate multiple
locations by comma):
#
path.data: /var/lib/elasticsearch
#
# Path to log files:
#
path.logs: /var/log/elasticsearch
#
.
.
.
# ---------------------------------- Network -----------------------
#
network.host: <Server_IRP_IP>
#
http.port: 9200
http.host: <Server_IRP_IP>
#
# --------------------------------- Discovery -----------------------
#
discovery.seed_hosts: ["127.0.0.1"]
#
.
.
.

thread_pool.search.queue_size: 100000
xpack.security.enabled: false
discovery.type: single-node
ingest.geoip.downloader.enabled: false
```

Código 5. Configuración de Elasticsearch para Cortex

Reiniciamos servicio para aplicar los cambios

```
➔ systemctl restart elasticsearch
```

4.2 Cambios en la configuración de TheHive

En el paso 3.3, al instalar TheHive, la configuración de Elasticsearch no la hemos modificado. Por defecto, se configura para que la escucha sea en 127.0.0.1:9000. Pero con la configuración que acabamos de hacer para Cortex, TheHive ya no escucha en la dirección correcta, la tiene mal configurada. Debemos cambiar la IP del host que sostiene el servicio.

```
➔ vim /etc/thehive/application.conf
```

```
.
.
.
index.search {
    #backend: lucene
    #directory: /opt/thp/thehive/index
    # If TheHive is in cluster ElasticSearch must be used:
    backend: elasticsearch
    hostname: ["<Server_IRP_IP>"]
    index-name: thehive
  }
```

Código 6. Configuración de TheHive compatible con Cortex.

```
➔ systemctl restart thehive
```

4.3 Instalación de Cortex

```
➔ curl https://raw.githubusercontent.com/TheHive-
  Project/TheHive/master/PGP-PUBLIC-KEY | sudo apt-key add -
➔ echo 'deb https://deb.thehive-project.org release main' |
  sudo tee -a /etc/apt/sources.list.d/thehive-project.list
➔ sudo apt-get update
➔ apt install cortex
```

4.4 Configuración de Cortex

Debemos instertar la secret key dentro de la configuración. Para ello:

```
➔ cd /etc/cortex
➔ (cat << _EOF_
➔ # Secret key
  # ~~~~~
  # The secret key is used to secure cryptographics
  functions.
```

```
# If you deploy your application to several instances be
sure to use the same key!
play.http.secret.key="$(cat /dev/urandom | tr -dc 'a-zA-
Z0-9' | fold -w 64 | head -n 1)"
_EOF_
) | sudo tee -a /etc/cortex/application.conf
```

Ahora debemos tocar la configuración de Cortex para que pueda localizar la instancia de Elasticsearch

➔ `vim /etc/cortex/application.conf`

```
## ElasticSearch
search {
  # Name of the index
  index = cortex
  # ElasticSearch instance address.
      uri = "http://<Server_IRP_IP>:9200"
.
.
.
```

Código 7. Configuración de Cortex para integrar Elasticsearch.

Reiniciamos servicio para aplicar los cambios

➔ `systemctl restart cortex`

4.5 Primer acceso al servidor

Entramos al navegador e insertamos la URL de la instancia Cortex. Como se ve en la Figura 14, en este primer acceso se debe actualizar la base de datos. Pinchamos en el botón de "*Update Database*".

http://<Server_IRP_IP>:9001/

Figura 14. Vista del primer acceso a la instancia de Cortex.

A continuación, aparece un formulario para la creación del primer usuario administrador. Es necesario para poder comenzar a utilizar la herramienta. Tras esto, el proceso es igual que en TheHive (ver paso 2.6). Es cuestión de crear una nueva

organización para después, crear usuarios. De hecho, la interfaz tiene el mismo diseño y funciones. Véase la Tabla 5 para ello.

Super-Administrador inicial:

- Login: admin@unican.es
- Name: <Nombre del administrador>
- Roles: superadmin

Organización:

- Name: UC_Analyst
- Descripción: Organization for UC analysis

Usuario 1:

- Login: org-admin@unican.es
- Name: <Nombre del org-admin>
- Roles: read, analyze, org-admin

4.6 Instalación y configuración de *analyzers* y *responders*

Los analizadores son la herramienta principal que facilita Cortex. Es un integrador de herramientas para emplearlas en los análisis de TheHive.

```
➔ apt-get install -y --no-install-recommends python3-pip
  python3-dev ssdeep libfuzzy-dev libfuzzy2 libimage-
  exiftool-perl libmagic1 build-essential git libssl-dev
➔ sudo pip3 install -U pip setuptools
```

Cada analizador o "*responder*", tiene una serie de requerimientos para habilitarlo dentro de Cortex que dependen de su desarrollador. Por ejemplo, el analizador de direcciones IP "*AbuseIPDB*" sólo exige una *API key* del usuario registrado en su web "*abuseipdb.com*". En cambio, MISP, requiere una *API key*, una url con la instancia que ejecuta el motor, y una validación de certificados.

Dentro de `/opt/cortex` clonamos el repositorio e instalamos los requerimientos de cada uno:

```
➔ git clone https://github.com/TheHive-Project/Cortex-
  Analyzers
➔ for I in $(find Cortex-Analyzers -name
  'requirements.txt'); do sudo -H pip3 install -r $I ||
  true; done
```

Por último, debemos editar la configuración de Cortex para establecer dónde va a encontrar los analizadores dentro del sistema:

```
➔ vim /etc/cortex/application.conf
```

```
analyzer {
  # analyzer location
  # url can be point to:
  # - directory where analyzers are installed
  # - json file containing the list of analyzer descriptions
  urls = [
    #"https://download.thehive-project.org/analyzers.json"
    "/opt/cortex/Cortex-Analyzers/analyzers/"
  ]
      .
      .
      .
responder {
  # responder location (same format as analyzer.urls)
  urls = [
    #"https://download.thehive-project.org/responders.json"
    "/opt/cortex/Cortex-Analyzers/responders/"
  ]
```

Código 8. Configuración de Cortex para la habilitación de analyzers y responders.

```
➔ systemctl restart cortex
```

4.7 Habilitación del analizador AbuseIPDB

La habilitación de *analyzers* y *responders* suele ser un proceso rápido y sin mucha complicación. Pero cabe destacar, que depende del sistema que se quiera integrar, los requisitos o campos son diferentes. En este caso habilitaremos uno sencillo: **AbuseIPDB**. Servirá como ejemplo para la integración de cualquier otro analizador.

Primero, vamos a la pestaña de "*Organization* > *Analyzers*":

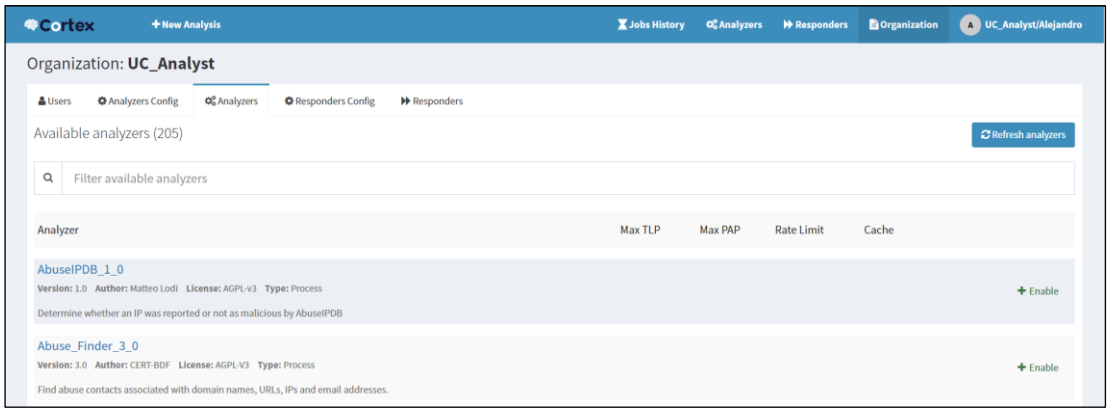

Figura 15. Vista de Cortex con analizadores disponibles.

Véase la Figura 15. En este momento tenemos una lista con todos los analizadores disponibles por defecto en Cortex (veremos más adelante que se pueden hacer integraciones personalizadas). Primero habilitaremos AbuseIPDB.

AbuseIPDB tiene la función básica de analizar direcciones IP. Dicho análisis re realiza comparando si anteriormente esa dirección fue reportada o no. Su misión es crear una "lista negra" o *blacklist* con aquellas direcciones asociadas con actividad maliciosa.

Es bastante sencillo de habilitar en Cortex. Sólo debemos:

1. Ingresar en su página web: https://www.abuseipdb.com/.
2. Crear una cuenta.
3. Ir al apartado "*API*" para generar una llave nueva.
4. Guardarla para integrarla en Cortex.

Una vez en Cortex, pinchamos en el botón de "+*Enable*". En ese momento, nos aparecerá un formulario (véase la Figura 16):

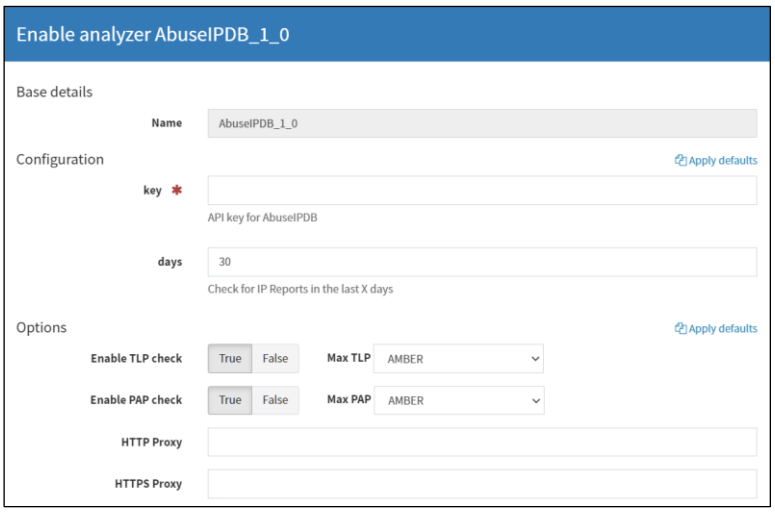

Figura 16. Vista de formulario para AbuseIPDB en Cortex.

Vemos que hay muchos campos, pero sólo es necesario introducir la *API key* generada en el paso anterior. Le damos a guardar y ya tenemos el analizador listo. El resto de los campos son opciones adicionales para la configuración del analizador, que va en función de las preferencias del usuario.

4.8 Primer análisis con AbuseIPDB

Como ejercicio de prueba, vamos a generar un análisis para probar el funcionamiento del sistema. Podemos elegir cualquier dirección de nuestra preferencia. En este caso, usaremos la conocida 8.8.8.8.

Primero, vamos a la pestaña de "*Analyzers*". Seleccionamos AbuseIPDB, y le damos a "*run*". Aparecerá entonces un formulario para ejecutar el análisis para ingresar los datos del mismo. Véase la Figura 17 para rellenar los campos.

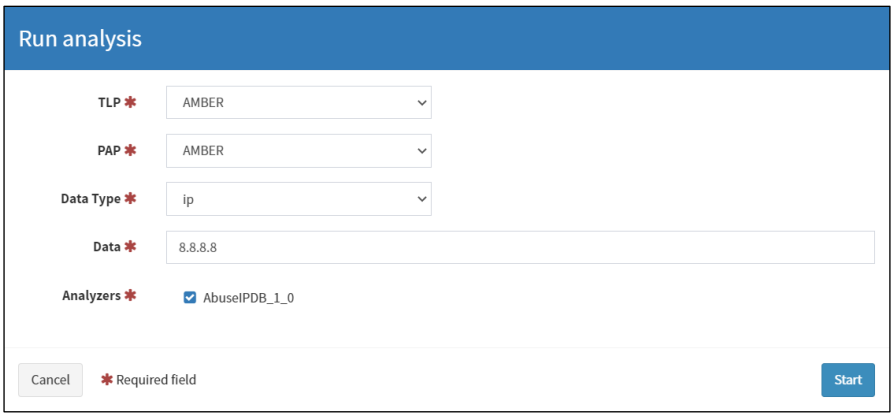

Figura 17. Vista del formulario de análisis para AbuseIPDB en Cortex.

Le damos a "*Start*", y en unos segundos, veremos que el análisis ha terminado con éxito. Con ello, se ha generado un reporte completo del análisis. Empleamos el botón "*View*" para ver el reporte.

Instalación y configuración del servidor de análisis #3: MISP (IRP server)

5.1 Instalación de MISP

Primero, descargamos la fuente.

➔ `wget -O /tmp/INSTALL.sh https://raw.githubusercontent.com/MISP/MISP/2.4/INSTALL/INSTALL.sh`

Ejecutamos la instalación como usuario normal. Este comando no puede ejecutarse como *root*. Puede durar varios minutos.

➔ `bash /tmp/INSTALL.sh -A`

Cuando termina la instalación, en la Figura 18 vemos la información reportada. Es necesaria para el primer acceso al servidor.

Figura 18. Vista del reporte tras la instalación de MISP.

5.2 Primer acceso y configuración

Accedemos a la instancia desde el navegador

https://<Server_IRP_IP>/

Introducimos las credenciales por defecto:

- Login: admin@admin.test
- Contraseña: admin

Inmediatamente pedirá que introduzcamos una nueva contraseña. La configuramos.

A continuación, habilitaremos las "*feeds*" principales de la que obtener eventos. Vamos a la pestaña "*Sync Actions* > *List feeds*", seleccionamos las dos y pulsamos el botón "*Enabled selected*" (véase la Figura 19).

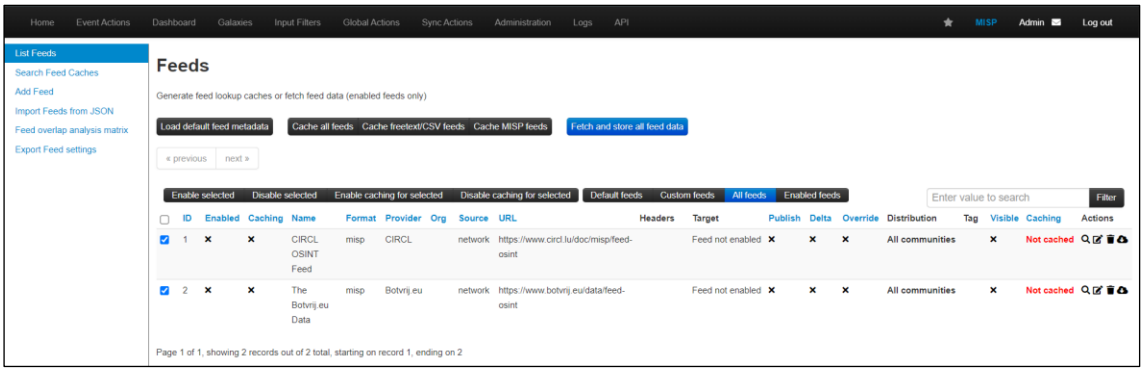

Figura 19. Vista de MISP para la habilitación de feeds.

Una vez habilitadas, debemos pulsar el botón pequeño de la derecha que dice "*Fetch all events*". Esto comenzará un proceso nuevo que podemos ver en "*Administration* > *Jobs*" donde se traen todos los eventos reportados por dichas organizaciones a la interfaz principal. Una vez termine, en el apartado "*Home*", tendremos los eventos listos para su consulta (ver Figura 20).

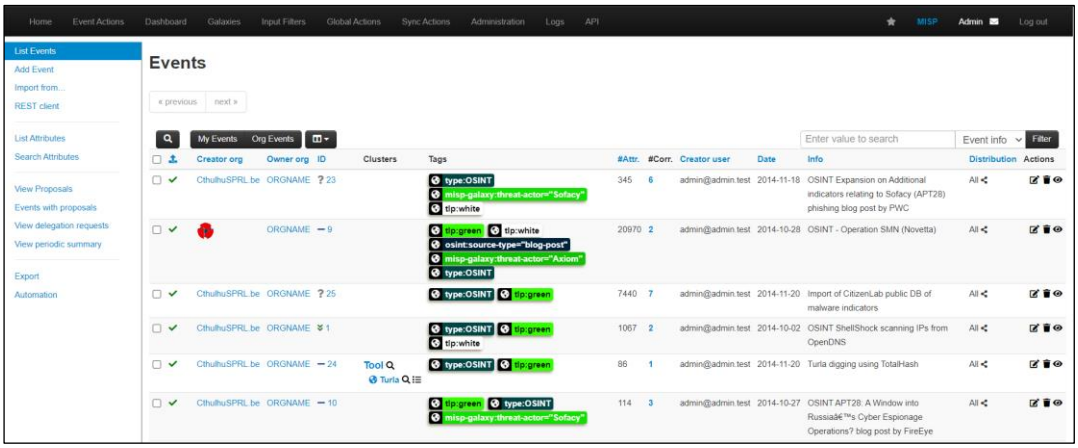

Figura 20. Vista de la interfaz de eventos en MISP.

Ahora, con el usuario por defecto, crearemos dos nuevos usuarios para la gestión diaria del servidor: un administrador y un usuario. Así diferenciaremos las tareas de gestión y las de usuario.

Administrador:

- Login: admin@unican.es
- Rol: admin

Usuario 1:

- Login: analyst@unican.es
- Rol: user

Iremos al apartado "*Administration* > *Add User*" y configuramos según la Figura 21:

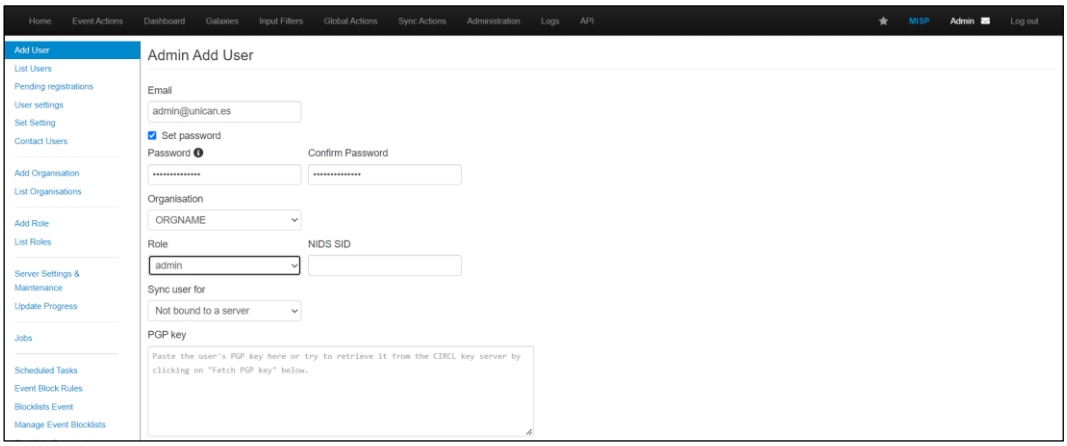

Figura 21. Vista de la creación de usuario en MISP.

Deberemos hacer lo mismo para el usuario normal. En vez de rol *admin*, pondremos *user*.

5.3 Habilitación de MISP en Cortex

Primero, debemos crear la *API key* dentro de MISP. Para ello, iniciaremos sesión como adminstrador ya que es el único que puede crear las llaves. Vamos al apartado de "*Administration* > *List Auth Keys*".

Seleccionaremos el botón de *"+ Add Authentication Key*". Sólo podemos crear llaves para el usuario actual. Por tanto, la key deberá ser sí o sí de nuestro administrador. Además, como se ve en la Figura 22, podemos editar qué direcciones IP específicas podrán hacer uso de esta llave. Introduciremos la IP de nuestro servidor.

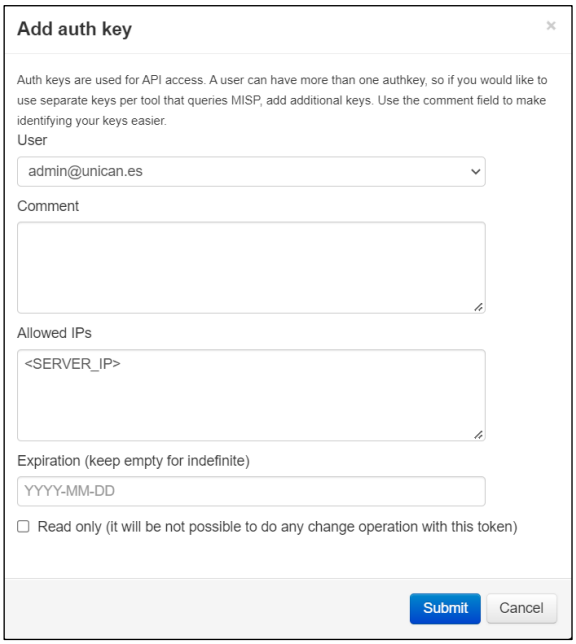

Figura 22. Vista de formulario para generar una API key en MISP.

Tener en cuenta que la *API key* sólo será visible una vez, en el momento de su creación. Por tanto, debemos guardarla aparte para su posterior uso en Cortex.

Como último paso, deberemos ir a Cortex y realizar de nuevo el proceso de habilitación de un analizador. Seleccionamos MISP y rellenamos el formulario como aparece en la Figura 23.

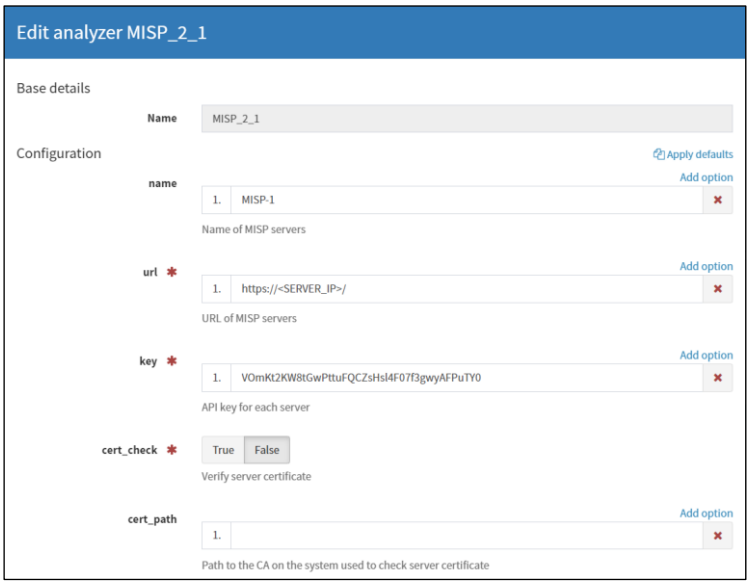

Figura 23. Vista del formulario para habilitar MISP en Cortex.

Con esto, ya podremos realizar análisis sobre observables de todos los tipos que cubre MISP_2_1 (ver Figura 24).

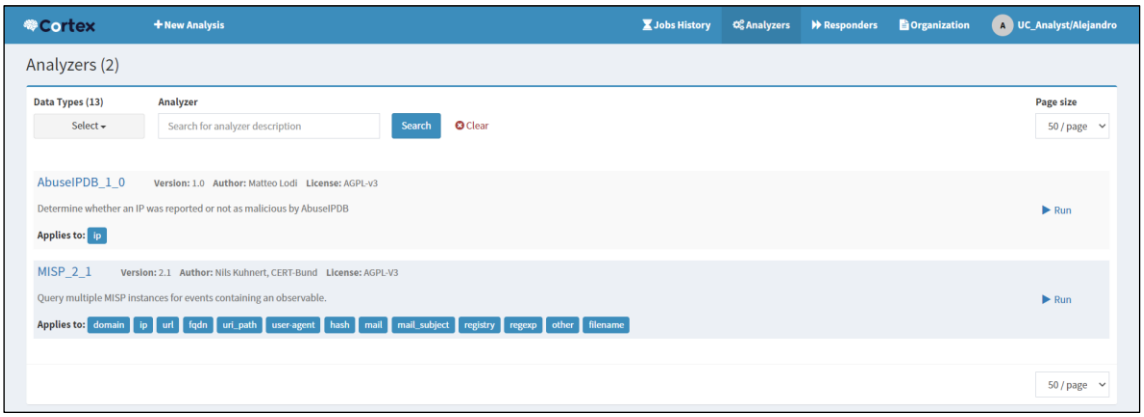

Figura 24. Vista de Cortex con analizadores habilitados.

Habilitación genérica de analizadores en Cortex (IRP server)

Puede darse el caso de que, por diferentes cuestiones o circunstancias, la organización necesite integrar nuevos analizadores. En caso de querer llevar a cabo esta decisión, podemos considerar dos situaciones diferentes, que dependen del tipo de analizador que queramos incorporar. Serán los analizadores predeterminados disponibles en Cortex, o uno nuevo hecho por nosotros mismos.

6.1 Analizadores predeterminados

Como hemos visto, cada analizador o "responder" tiene una serie de requerimientos diferentes. Para saber cómo habilitar cada uno de ellos deberemos ir a la documentación oficial del proyecto y consultar el que queramos [20].

6.2 Analizadores personalizados

Además, al tratarse de una herramienta de código abierto, podemos implementar nuestros propios analizadores en Python2 o Python3 e incorporarlos dentro del servidor. Para ello, consultamos la documentación [21].

Integración de TheHive con Cortex (IRP server)

El paso final para completar el puzle del SOC es habilitar el análisis de observables, facilitado por Cortex, para la plataforma de respuesta a incidentes TheHive.

De esta manera, podremos tomar acción directamente desde TheHive cuando queramos hacer un análisis. En cambio, si esta integración no se realiza, por cada observable que queramos analizar, el procedimiento sería manual y tedioso, como vimos en el paso 4.8 usando AbuseIPDB.

7.1 Generación API key en Cortex

Primero debemos acceder a Cortex con un usuario que tenga permisos de administrador. Así podremos gestionar los usuarios y la creación de *API keys*. Elegiremos el administrador que creamos en el paso 4.5. Por ello, seleccionaremos el usuario y con ello el botón "*Create API key*" y la guardamos a salvo.

7.2 Integración en TheHive

Debemos modificar el fichero de configuración de TheHive según el Código 9:

➔ `vim /etc/thehive/application.conf`

```
## CORTEX configuration
# More information at https://github.com/TheHive-
Project/TheHiveDocs/TheHive4/Administration/Connectors.md
# Enable Cortex connector
 play.modules.enabled += org.thp.thehive.connector.cortex.CortexModule
 cortex {
  servers: [
    {
      Name = "local"                        # Cortex name
      url = "http://<SERVER_IP>:9001" # URL of Cortex instance
      auth {
        type = "bearer"
        key = "<CORTEX_API_KEY>"    # Cortex API key
      }
      wsConfig {}
    includedTheHiveOrganisations = ["*"]
    includedTheHiveOrganisations = []
    }
  ]
  refreshDelay = 5 seconds
  maxRetryOnError = 3
  statusCheckInterval = 1 minute
 }
```

Código 9. Configuración de TheHive para la integración con Cortex.

➔ `systemctl restart thehive`

Implementación de los agentes en los equipos de laboratorio (SIEM server)

8.1 Creación de los grupos (Wazuh manager)

Wazuh tiene la posibilidad de juntar los agentes en grupos, para que su gestión y manejo sean más fácil e intuitivo. El criterio que emplearemos para la asignación de grupos será clasificar los equipos en función del aula o grupo al que pertenecen.

En nuestro caso, como hay 7 grupos de equipos diferenciados:

- Lab-1
- Lab-2
- Lab-4
- Lab-5
- Docencia
- UnicanLabs
- Suricata

La creación del grupo adicional "Suricata" sirve para facilitar la gestión de reglas y criterios que afectan a los equipos que ejecutan dicho servicio. En vez de configurar de uno en uno, empleamos la funcionalidad de configuración compartida que ofrece el *manager*.

Ejecutando desde el *manager*, la creación de cada grupo sería:

➔ `/var/ossec/bin/agent_groups -a -g <GROUP_NAME> -q`

8.2 Agregado de los agentes (Wazuh agent)

Una vez creados los grupos, debemos instalar en cada equipo destino el agente de Wazuh. Es tan sencillo como, dentro de la interfaz, ir al apartado "*Agents* > + *Deploy New Agent*".

Esto nos llevará a una página donde podemos modificar los campos del agente destino y en función de la selección, nos brinda un comando listo para ejecutar e instalar el servicio en el equipo destino.

Por ejemplo, para un equipo con las siguientes características:

- Sistema Operativo: Debian/Ubuntu
- Arquitectura: x86_64
- Dirección IP del servidor: 192.168.1.110
- Grupo/s de agentes: "default"

El comando para ejecutar en el equipo destino sería:

```
➔ curl -so wazuh-agent-4.3.9.deb
  https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-
  agent/wazuh-agent_4.3.9-1_amd64.deb && sudo
  WAZUH_MANAGER='192.168.1.110'
  WAZUH_REGISTRATION_PASSWORD='please123'
  WAZUH_AGENT_GROUP='default' dpkg -i ./wazuh-agent-
  4.3.9.deb
```

En nuestro caso, el campo "Grupo de agentes" será, claramente, el aula a la que queremos agregar dicho equipo. Sólo faltaría reiniciar el agente y el servicio:

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

8.3 Integración de agente de Wazuh con NIDS Suricata (Wazuh agent)

Para que tengamos un control sobre la red en la que los equipos están, debemos instalar un NIDS. Cada equipo integrará este servicio. Es una mejora considerable en la recogida de alertas.

8.3.1 Instalación Suricata

Primero debemos agregar el repositorio para instalar Suricata

```
➔ add-apt-repository ppa:oisf/suricata-stable
➔ apt update
➔ apt install suricata
```

Configuramos la línea 589, cambiamos el parámetro `interface` por `enp0s3` o la que queramos que Suricata analice

```
➔ vim /etc/suricata/suricata.yaml
```

Reiniciamos el servicio para aplicar los cambios

```
➔ systemctl restart suricata
```

8.3.2 Descarga e integración de las reglas *Emerging Threats*

"***Emerging Threats***" es un proyecto colaborativo que va añadiendo reglas al IDS Suricata. Es una mejora considerable para la protección que va a llevar a cabo en nuestro sistema.

```
➔ suricata-update
```

El fichero se descarga en `/var/lib/suricata/rules/suricata.rules`. Por tanto, para que esas reglas tomen efecto, debemos modificar la ruta por defecto al conjunto de reglas desde la configuración de Suricata (ver Código 10).

➔ `vim /etc/suricata/suricata.yaml`

```
default-rule-path:
/var/lib/suricata/rules/

rule-files:
-     suricata.rules
```

Código 10. Configuración de Suricata para recoger las reglas de Emerging Threats.

➔ `systemctl restart suricata`

8.3.3 Integración con Wazuh

De manera predeterminada, Suricata escribe las alertas en `/var/log/suricata/eve.json`. Este fichero no es monitoreado por defecto en Wazuh. Por tanto, debemos decirle que para todos los agentes que pertenecen al grupo "Suricata" (ver creación en el paso 8.1), analicen ese fichero para el registro de alertas y las manden al *manager*. Para ello editamos la configuración *shared* del *manager* y añadimos una sección como esta:

➔ `vim /var/ossec/etc/shared/Suricata/agent.conf`

```
<agent_config>
    <localfile>
        <log_format>json</log_format>
        <location>/var/log/suricata/eve.json</location>
    </localfile>
</agent_config>
```

Código 11. Configuración shared de Wazuh para el grupo "Suricata".

➔ `systemctl restart wazuh-manager`

Comprobamos que la configuración es válida:

➔ `/var/ossec/bin/verify-agent-conf`

Index management del Wazuh indexer (SIEM server)

Mediante las políticas de índices, se realiza el "reciclaje" de eventos y cubriremos el requisito no funcional de mantener los ficheros durante 90 días.

Desde la interfaz web, vamos al apartado de *Index Management*. A continuación, daremos a "*Create policy*", seleccionando la opción del editor visual. Crearemos una política para cada índice, ya que con el tiempo es posible querer depurar estos criterios para un determinado índice sin modificar el resto. Asimismo, el proceso es igual para cada uno de ellos. En total serán 4:

- *alerts-policy*
- *archives-policy*
- *monitoring-policy*
- *statistics-policy*

1. Asignamos un identificador acompañado de una pequeña descripción.

2. Estado inicial: *hot*
    a. Orden: *add before cold*
    b. Acciones: *none*
    c. Transición:
        i. Estado destino: *cold*
        ii. Condición: *Minimum index age*
        iii. Minimum index age: 90d

3. Estado final: *cold*
    a. Orden: *add after hot*
    b. Acciones: *delete*
    c. Transición: *none*

Una vez creadas estas políticas, deben añadirse a los índices deseados. Por ejemplo, asignaremos la política *alerts-policy* a todos los índices "*wazuh-alerts-…*". En la pestaña *Indices* seleccionamos los índices deseados y pulsamos el botón superior "*Apply policy*". Seleccionamos la política deseada.

Este proceso de asignar políticas a los índices es manual y debe efectuarse periódicamente, para así poder efectuar este reciclaje.