Universidad Zaragoza

ORBE TELECOMUNICACIONES

Trabajo Fin de Grado

Prototipo de plataforma tipo SIEM para monitorización de alertas en un SOC

Prototype of a SIEM platform for monitoring alerts in a SOC

Miguel Yanes Fernández
683621

Grado en Ingeniería Informática

Directores:
Ricardo J. Rodríguez Fernández
Félix Serna Fortea
Daniel Sánchez Yubero

Septiembre 2020
Curso 2019/2020

Escuela Universitaria Politécnica de Teruel

RESUMEN

La ciberseguridad es un aspecto muy importante actualmente en todo tipo de organizaciones, y el poder estar protegido y alerta frente a posibles amenazas supone un aspecto fundamental. Para ello existen múltiples soluciones de ciberseguridad a nivel empresarial, pero no siempre es factible dedicar el tiempo necesario a monitorizar y gestionar esas alertas. Es por ello que están en auge las soluciones de SIEM (*Security Information and Event Management*), que permiten de manera centralizada la monitorización y gestión de dichas alertas.

Este proyecto se centra en el desarrollo de un prototipo de herramienta tipo SIEM que permita monitorizar eventos de ciberseguridad en tiempo real en múltiples organizaciones.

Toda la información recogida por el sistema se obtiene de una serie de herramientas de ciberseguridad comerciales del fabricante Cisco. Con estas herramientas, y mediante el uso de las APIs que disponen para recoger datos, el sistema recibe los logs, los parsea al formato deseado, y los muestra de manera gráfica de manera que un analista de ciberseguridad pueda revisar con facilidad toda la información relevante respecto a las alertas de ciberseguridad de una organización. El proyecto se ha realizado de manera conjunta con la empresa Orbe Telecomunicaciones. El prototipo pretende servir como posible herramienta de uso interno en Orbe para ser usado por su departamento de SOC (*Security Operations Center*).

El sistema sigue una arquitectura multi-tenant, que permite integrar múltiples organizaciones de manera independiente, de forma que un cliente pueda acceder al portal de su organización sin llegar a ver información de otras organizaciones, y permitiendo también al departamento de seguridad de Orbe, gestionar todas las soluciones de todos los clientes desde un mismo portal centralizado. Para ello se han probado varios sistemas gestores de bases de datos y soluciones SIEM disponibles en el mercado, optando finalmente por Elastic Stack.

Se ha logrado montar un sistema completamente funcional y automático, que permite monitorizar con facilidad toda la información generada por aquellas soluciones que se quieran integrar dentro del sistema.

ABSTRACT

Cybersecurity is a very important aspect nowadays in all kinds of organizations, and being able to be protected and alert to possible threats is a fundamental aspect. There are many cyber security solutions available at an enterprise level, but it is not always feasible to devote the time needed to monitor and manage these alerts. For this reason, SIEM (*Security Information and Event Management*) solutions are on the rise, allowing for centralized monitoring and management of these alerts.

This project is focused on the development of a prototype SIEM-type tool that allows monitoring of cyber security events in real time in multiple organizations.

All the information collected by the system is obtained from a series of commercial cybersecurity tools from the manufacturer Cisco. With these tools, and through the use of the APIs available to collect data, the system receives the logs, parses them into the desired format, and displays them graphically so that a cybersecurity analyst can easily review all relevant information regarding the cybersecurity alerts of an organization. The project has been carried out jointly with the company Orbe Telecomunicaciones. The prototype aims to serve as a possible tool for internal use in Orbe to be used by its SOC department (*Security Operations Center*).

The system follows a multi-tenant architecture, which allows the integration of multiple organizations in an independent way, so that a customer can access the portal of their organization without seeing information from other organizations, and also allowing Orbe's security department to manage all the solutions of all customers from the same centralized portal. For this purpose, several database management systems and SIEM solutions available in the market have been tested, finally choosing Elastic Stack.

It has been possible to assemble a fully functional and automatic system, which allows to easily monitor all the information generated by those solutions and other solutions that could be integrated in the future.

Índice general

1. Introducción **1**
1.1. Objetivo 1
1.2. Alcance 2
1.3. Estructura del documento 2

2. Conceptos preliminares **3**
2.1. Cisco 3
2.2. Gestores BBDD / Herramientas SIEM 5
2.3. Estándar UML 6
2.3.1. Diagrama de Casos de Uso 7
2.3.2. Diagrama de Clases 8
2.3.3. Diagrama de Componentes 8
2.3.4. Diagrama de Actividad 8

3. Plataforma SIEM para monitorización de alertas **9**
3.1. Perspectiva del producto 9
3.2. Evolución del sistema 10
3.3. Requisitos del sistema 11
3.4. Funcionalidad del producto 12
3.5. Sistema Base de Datos 13
3.6. RESTful APIs 14
3.7. Diseño 16
3.8. Diagramas del diseño 17

4. Puesta en marcha del sistema **21**
4.1. Servidor Sistema SIEM 21
4.2. Programa de gestión de logs 22
4.3. Puesta en marcha Elastic Stack 22
4.4. Sistema SIEM 23
4.5. Pruebas 24

5. Conclusiones y trabajo futuro **27**
5.1. Conclusiones obtenidas 27
5.2. Problemas encontrados 28
5.3. Trabajo futuro 28
5.3.1. Reportes automáticos 28
5.3.2. Configurar alertas personalizadas 28

Bibliografía **29**

A. Definiciones y acrónimos **33**

B. Horas de Trabajo **35**

Índice de tablas

2.1. Tabla comparativa gestores bases de datos/herramientas SIEM 7

B.1. Tabla horas dedicadas al proyecto 36

Índice de figuras

3.1. Esquema de la arquitectura de la aplicación 10
3.2. Diagrama UML casos de uso 13
3.3. Esquema de componentes 17
3.4. Diagrama de actividad - Gestión de Logs 17
3.5. Diagrama de clases - Gestión de Logs 18

4.1. Captura sistema inicio de sesión 23
4.2. Captura sistema selección de espacio 24
4.3. Captura sistema dashboard Cisco Umbrella 25

Capítulo 1

Introducción

Este trabajo final de grado se corresponde al análisis, estudio e implementación de un prototipo de un sistema para monitorizar y gestionar alertas de eventos de ciberseguridad a nivel empresarial. Se ha realizado en colaboración con Orbe Telecomunicaciones S.L., que ha colaborado con el proyecto con su conocimiento y experiencia en el sector de la ciberseguridad, y más concretamente con su conocimiento y estrecha relación con Cisco en lo relacionado a soluciones de ciberseguridad empresarial. Este primer capítulo sirve como introducción al proyecto, la motivación del mismo, objetivos y alcance deseado, así como también para indicar la estructura general del documento de la memoria [3, 16].

Se pretende por tanto desarrollar el prototipo de una herramienta para centralizar y monitorizar diferentes soluciones de ciberseguridad, de manera que pudiera ser empleada por un departamento de SOC (de las siglas en inglés *Security Operations Center*) para la monitorización de alertas en tiempo real. El sistema integrará diferentes fuentes de información de soluciones comerciales de ciberseguridad del fabricante Cisco, de manera que sean estas soluciones las que generen la información que luego el sistema recogerá para poder ser analizada [18].

La importancia del proyecto reside en la gran relevancia que tiene actualmente la ciberseguridad en todos los tipos de empresas, sin importar si son grandes o pequeñas. Cada día surgen ataques nuevos, y cada día se pueden leer noticias y artículos sobre organizaciones que han sido atacadas, que han sufrido pérdidas de datos o brechas de seguridad, u organizaciones víctimas de un *ransomware* y a las que se les pide un rescate a cambio de recuperar archivos confidenciales y esenciales en su organización. Aunque los titulares solo hablen de grandes organizaciones, todo tipo de empresas y organizaciones pueden ser víctimas de este tipo de ataques, y es por ello que la ciberseguridad es un elemento muy importante y en alza, que cada día hay que tener más en cuenta [10].

1.1. Objetivo

El objetivo final del proyecto es implementar un prototipo funcional con el que monitorizar las alertas de ciberseguridad generadas por diferentes soluciones comerciales disponibles en el mercado del fabricante Cisco. El sistema permitirá monitorizar y vi-

sualizar estas alertas para poder tomar medidas y detectar posibles ataques en tiempo real.

El sistema será accesible vía web, de manera que diferentes usuarios puedan acceder con su cuenta personal y gestionar o visualizar los datos de las organizaciones para las que tengan permisos. Será posible monitorizar alertas de ciberseguridad de manera centralizada y gráfica, de manera que un usuario no experto pueda comprender los datos mostrados y tomar las acciones necesarias desde el punto de vista de la seguridad.

1.2. Alcance

Se implementará un prototipo funcional de la aplicación con las funcionalidades básicas de monitorización, actualización y gestión de alertas. El alcance del proyecto a largo plazo es más amplio ya que se pretende que sea una aplicación que se use en producción, por lo que se deberá mantener y actualizar en el tiempo.

Inicialmente se pretende desarrollar el sistema para que tenga compatibilidad con las soluciones de ciberseguridad del fabricante Cisco, pero también para que sea sencillo adaptarlo a más herramientas, y de distintos fabricantes, en el futuro con la mayor facilidad posible.

El prototipo será principalmente de uso interno en Orbe Telecomunicaciones SL, pero se pretende desarrollar con la idea de que en un futuro se pueda dar acceso también a los clientes finales.

1.3. Estructura del documento

El capítulo 1 incluye un breve resumen del propósito y alcance del proyecto. En el capítulo 2 se analiza el estado actual de las soluciones de ciberseguridad disponibles en el mercado, tanto las que supongan competencia con el sistema a desarrollar, como aquellas soluciones que se integrarán dentro del mismo. Se estudian también los diferentes aspectos concretos del proyecto, como pueden ser los sistemas SIEM, las APIs que empleará el sistema, los gestores de bases de datos, o las herramientas comerciales de Cisco que se pretenden integrar [17]. El capítulo 3 se divide en el análisis de los casos de uso y los requisitos del proyecto. Se define también la fase de análisis del proyecto, así como el diseño a seguir para su puesta en marcha e implementación. El capítulo 4 incluye el proceso de implementación, puesta en marcha del sistema (servidor, componentes y software desarrollado), y de pruebas, aplicado al sistema prototipo. En el capítulo 5 se definen conclusiones extraídas del proceso de investigación, análisis, diseño, implementación, configuración y puesta en marcha del sistema. Además, se listan todos aquellos aspectos a desarrollar en el futuro y que se añadirán al sistema. Este apartado incluye también aquellos aspectos no desarrollados como parte del prototipo, pero que se han estudiado para poder añadidos al sistema en el futuro.

Además se incluyen dos anexos final de la memoria. El anexo A, indicando todos los términos, conceptos y abreviaturas que se mencionan en la memoria.Y el anexo B, que incluye del desglose de las horas dedicadas a la realización del trabajo.

Capítulo 2

Conceptos preliminares

En este capítulo se desarrollan los conocimientos previos necesarios para comprender el sistema a desarrollar. Por un lado están las soluciones de ciberseguridad de Cisco, su funcionamiento, y la manera de recoger los logs mediante sus APIs. Además, se define también los sistemas gestores de datos y sistemas SIEM estudiados como parte de la fase de investigación del proyecto. Por último, se define el estándar de diagramas UML, empleado para representar el diseño de la arquitectura del sistema, así como de la implementación necesaria para integrar todas las partes en un mismo sistema [20].

Existen múltiples soluciones en el mercado para proteger a las organizaciones y empresas frente a ataques cibernéticos. Estas soluciones pueden variar según el fabricante y el aspecto informático concreto que se pretende proteger. Como parte del proyecto se han estudiado las soluciones de ciberseguridad del fabricante Cisco, pero sin restringir el proyecto únicamente a este fabricante, de manera que sea fácilmente adaptable en el futuro en caso de necesidad. Se han analizado también una serie de soluciones de sistemas gestores de bases de datos que se pueden llegar a utilizar como parte del proyecto, así como otras herramientas SIEM disponibles en el mercado y que se puedan usar como referencia a la hora de analizar el alcance y funcionalidad del prototipo.

2.1. Cisco

Las soluciones de ciberseguridad que se van a utilizar como parte de este proyecto son del fabricante norteamericano Cisco. Cisco es una empresa centrada en el diseño y fabricación de equipos de telecomunicaciones, y hace una serie de años que también se ha enfocado en la protección de ciberseguridad para sus clientes. Desde firewalls (seguridad de red), hasta soluciones anti software malicioso (seguridad en los equipos, para prevenir malware como virus o troyanos), o soluciones de ciberseguridad basadas en la nube; en el portfolio de ciberseguridad de Cisco existen una veintena de diferentes soluciones para todas las empresas de todos los tamaños.

Al ser una gama de productos tan amplia, se ha optado por desarrollar la integración de 3 de las soluciones más populares, *Cisco Umbrella*, *Cisco AMP for Endpoints*, y *Cisco Cloud Email Security*. Además, todas estas soluciones se basan en la inteligencia de

Cisco Talos, el equipo de ciberseguridad de Cisco, y que es el equipo de investigadores de ciberseguridad privado más grande del mundo, colaborados habitual con organizaciones como la Interpol para detectar, estudiar, y prevenir nuevos ataques. A continuación se describen brevemente estas soluciones comerciales:

- *Cisco Umbrella* [8]: Umbrella es la protección a nivel DNS que ofrece Cisco. Basada en la nube, permite proteger el tráfico de red a nivel del servidor DNS de salida a Internet, de manera que todas aquellas peticiones que necesiten realizar una consulta DNS (es decir, todas las conexiones que no sean directas a IP y que necesiten resolver el nombre del dominio), pasarán por el servidor de Umbrella. Con esta información, y en base a la inteligencia de Talos, se puede identificar si un dominio puede ser malicioso. El cliente puede definir las políticas que considere necesarias y el nivel de protección que se aplicará a cada uno.

  En lo relativo a este proyecto, la información que ofrece Cisco Umbrella son todas las conexiones que ha resuelto el servidor DNS para la organización (tanto aquellas permitidas como las que no, habitualmente sumando varias millones de entradas en unas pocas semanas). A partir de esta información, la información concreta de seguridad (dominios maliciosos, o políticas concretas definidas), se podrá ver un listado de conexiones identificando el equipo desde el que se originan, usuario que ha realizado las peticiones DNS (siempre que la organización use *Windows Active Directory* y esté integrado con Umbrella), IPs de origen y destino, fecha y hora, dominio de la consulta, y categoría aplicada por Talos a ese dominio. Con esta información se puede obtener una inteligencia en tiempo real para detectar conexiones maliciosas e identificar su origen, y en el caso de Umbrella además sin necesidad de instalar un equipo físico en la organización y sin provocar un retraso en las conexiones, ya que todas las conexiones DNS de salida de internet, se tienen que realizar igualmente, sea con el servidor de Umbrella o bien con el DNS del IPS de la organización.

- *Cisco AMP for Endpoints* [4]: AMP (*Advanced Malware Protection*), es el anti-malware de nueva generación de Cisco. A diferencia de un antivirus tradicional, que solo comprobaría las firmas de los archivos contra una base de datos de malware conocido, AMP analiza también la ejecución de los procesos en tiempo real para analizar y detectar posible actividad maliciosa, incluso en procesos en principio legítimos. AMP funciona mediante un cliente que se instala en todos los equipos de la organización, y que según la política aplicada para cada equipo o grupo de equipos protege esos endpoints en diferente grado.

  Para el proyecto, se pretende obtener toda la información de procesos y ficheros maliciosos detectados por AMP. Por defecto, AMP muestra la información de todos los procesos en ejecución en todos los equipos de la organización en cualquier momento de los últimos 30 días. Esto claramente consiste en una gran cantidad de información difícil de manejar, y por eso la información a integrar será únicamente la relativa a las alertas de seguridad.

- *Cisco CES* [6]: *Cloud Email Security*, o CES, es la protección de Cisco para el correo electrónico. En el caso de CES esto se hace mediante un servidor virtual en la nube. Esta protección permite analizar todas las conexiones entrantes al servidor de correo, pudiendo así analizar la reputación del servidor de origen de la conexión, así como el contenido y los metadatos del correo. Dispone de diferentes motores para analizar spam, malware, un filtro de malware avanzado (el mismo motor que AMP), categorías grises (marketing, redes sociales y correos masivos), filtros manualmente definidos, y un último filtro de nuevas amenazas o amenazas emergentes, que pone en cuarentena correos sospechosos de una posible nueva amenaza hasta que se tenga más información sobre si puede suponer o no un problema.

  De toda esta información, se pretenden integrar en el sistema todos los datos de métricas de correos electrónicos maliciosos, y todos los correos detectados por cada motor. No se integrará por ejemplo todos los correos enviados a cuarentena, ya que la información de metadatos y contenido del propio correo, supondría una nivel de información que no es necesario para el proyecto.

Estas tres soluciones se han elegido por ser soluciones muy populares y que además se centran en aspectos de la seguridad empresarial muy diferentes (protección de red, protección de los equipos, y protección del correo electrónico, respectivamente). Para las 3 soluciones integradas, se obtendrá la información mediante las API REST públicas de cada solución. Estas APIs funcionan mediante unas credenciales definidas, y que siguiendo la documentación oficial de Cisco permiten obtener toda la información necesaria para incluir en la aplicación del proyecto. Se detalla más en concreto el funcionamiento y análisis realizado sobre estas APIs en la sección 3.6.

2.2. Gestores BBDD / Herramientas SIEM

Para el sistema de base de datos se han probado diferentes soluciones disponibles que están ya enfocadas al ámbito de la ciberseguridad, o bien que se podrían adaptar fácilmente al sistema a desarrollar.

Un sistema SIEM, de las siglas en inglés *Security Information and Event Management*, es un sistema que permite recoger información de los diferentes equipos de una organización para poder analizar posibles amenazas de ciberseguridad. Generalmente esto se realiza mediante la recolección de logs de los equipos, ya sea con un cliente específico instalado en cada uno, o bien mediante los logs recogidos por otra aplicación (que es el caso del presente proyecto). El objetivo del sistema es que un analista de seguridad pueda revisar los logs en busca de posibles amenazas, y en caso de necesidad realice las acciones necesarias para mitigar todo lo posible la amenaza o brecha de seguridad detectada.

Existen múltiples soluciones comerciales de aplicaciones SIEM para gestionar y monitorizar eventos de ciberseguridad. Se han probado varias opciones de las más populares disponibles en el mercado. El proceso para cada solución a probar ha sido el de comprobar la licencia de uso (comercial, no comercial, código abierto), comprobar las funcionalidades y facilidad de uso, y la facilidad de integración con las APIs que se usarán como parte

del proyecto. Existen diferentes soluciones comerciales y no comerciales que ofrecen capacidades similares a las que se busca con este proyecto. Es importante recalcar que todas estas soluciones requieren una parte de configuración y customización, ya que ninguna ofrece integración oficial con las soluciones de seguridad que se usarán como fuentes de información. A continuación se muestra un breve listado de las herramientas probadas:

1. *Splunk* [19]. Herramienta comercial, muy usada en el mercado, que tiene gran cantidad de documentación e incluso formación gratuita.

2. *Wazuh* [21]. Herramienta de código abierto para gestión de fuentes de información de soluciones de seguridad. Más centrada en el apartado SIEM, por tanto, tal vez no la mejor solución para integrar herramientas comerciales de ciberseguridad.

3. *Elastic Stack* [11]. Solución de código abierto con funcionalidades de pago, que ofrece un gran nivel de customización y escalado.

4. *Alienvault* [1]. Solución no comercial para 1 cliente, pero de pago para múltiples clientes. Solución ya configurada y con cliente del SIEM listo para desplegar a los equipos.

Mediante el estudio y pruebas de cada una de estas herramientas, se ha realizado una tabla comparativa de las diferentes funcionalidades, puntos fuertes y flaquezas de cada herramienta, listado en la tabla 2.1.

Finalmente se ha optado por desarrollar el sistema con la suite de herramientas de Elastic Stack, como se ve más en detalle en el apartado de análisis en el capítulo 3.5 Sistema Base de Datos. La solución elegida no es un SIEM como tal pero sí ofrece funcionalidades muy similares. De hecho uno de los plugins disponibles ofrece funcionalidades de SIEM, pero al ser un plugin de pago se ha decido no usarlo como parte del proyecto.

2.3. Estándar UML

Para la representación de los diagramas de diseño del proyecto, se ha seguido el estándar UML (*Unified Modeling Language*), ya que es uno de los más aceptados [20]. UML es un lenguaje de modelado estandarizado que consiste en un conjunto de diagramas, desarrollado para ayudar a los desarrolladores de sistemas y programas informáticos a especificar, visualizar, construir y documentar los elementos de los sistemas de software, así como para el modelado comercial y otros sistemas no informáticos.

UML representa una colección de las mejores prácticas de ingeniería que han demostrado ser exitosas en el modelado de sistemas grandes y complejos. UML es una parte muy importante del desarrollo de software orientado a objetos y del proceso de desarrollo de software. Se utiliza sobre todo notaciones gráficas para expresar el diseño de proyectos de software. Su uso ayuda a los equipos de proyecto a comunicarse, explorar diseños potenciales y validar el diseño arquitectónico del software.

Tabla comparativa herramientas probadas			
Herramienta	*Características Principales*	*Puntos fuertes*	*Flaquezas*
Splunk	Características de SIEM completo	Muy usada a nivel empresarial. Fácil de usar y de integrar con fuentes externas	Versión gratuita limitada, las funcionalidades avanzadas son de pago
Wazuh	Plataforma de gestión, monitorización y detección de eventos de seguridad.	Código abierto y licencia gratuita	No permite integrar fuentes de información externas
Elastic Stack	Solución de base de datos basada en texto, con portal web de visualización de los datos. Tiene algunas características de SIEM	Código abierto, licencia gratuita, muy configurable y adaptable	El plugin de SIEM es de pago
Alienvault OSSIM	Herramienta de SIEM completa ya desarrollada y lista para usar	Licencia gratuita para una organización, posibilidad de integrar fuentes externas	Solo tiene licencia gratuita con el primer cliente, de pago para funcionalidades multi-tenant

Tabla 2.1: Tabla comparativa gestores bases de datos/herramientas SIEM

Se definen a continuación los diferentes tipos de diagramas empleados en las fases de análisis y diseño del sistema a desarrollar.

2.3.1. Diagrama de Casos de Uso

El diagrama de casos de uso pretende representar los diferentes usos que se le pueden dar al sistema por parte de los usuarios finales. En un diagrama de casos de uso se pueden representar múltiples actores, que representan los diferentes tipos de usuarios que pueden interactuar con el sistema (administradores, usuarios finales, usuarios registrados/no registrados, ...). El rol de cada actor se entiende como único, aunque un mismo rol puede tener casos de uso en común con otro rol diferente. La manera de representar estos casos es mediante las actividades concretas que un actor puede realizar, relacionando a su vez a cada actividad, o caso de uso, con el actor o rol correspondiente.

2.3.2. Diagrama de Clases

El diagrama de clases en el estándar UML sirve para representar las diferentes clases de la implementación del código fuente de un programa informático. Se representan las clases u objetos a implementar, incluyendo detalles generales de la implementación como pueden ser los atributos y las funciones que incluirá cada clase, además de indicar gráficamente cómo se relacionan las diferentes clases entre ellas. En las relaciones entre clases se indica la multiplicidad, de manera que se indica gráficamente con un número sobre la línea de conexión del lado de cada clase, que indica el número de veces que se relaciona. Cuando la relación es 1, significa que esa clase u objeto solo se relaciona con una única instancia, mientras que si es N, *, o 1..*, significa que se relaciona un número N de veces que puede ser cualquiera de 0 a N, o en el caso 1..*, puede ser de 1 a N (es decir, que se relaciona obligatoriamente al menos una vez).

2.3.3. Diagrama de Componentes

El diagrama o esquema de componentes, a diferencia del diagrama de clases, permite representar los diferentes componentes de un sistema y la forma de relacionarse entre ellos. La diferencia consiste en que este tipo de diagrama permite representar componentes externos del sistema, como pueden ser secciones ya implementadas, código heredado, o como en el caso de este proyecto, componentes como la base de datos o las APIs, que ya están implementadas y con las que únicamente será necesario interactuar. Este diagrama, dividido en diferentes elementos (bloques generales de la arquitectura del sistema), representa todos los diferentes componentes y su relación entre ellos de una manera abstracta, sin entrar en detalles de la implementación o el tipo de relación entre los diferentes componentes del sistema.

2.3.4. Diagrama de Actividad

El último tipo de diagrama utilizado como parte del proyecto es el diagrama de actividad. Este tipo de diagrama, según el estándar UML, permite representar la lógica de ejecución o de uso de un sistema. Permite, por tanto, representar todas las acciones unitarias que un usuario o un componente del sistema debe realizar, indicando el orden y en un alto nivel la lógica de la ejecución o uso del sistema. Un diagrama de actividad representa bloques unitarios para cada actividad del sistema, con flechas indicativas de la lógica de ejecución o de uso, y en el caso de situaciones condicionales, representa también la lógica a seguir para cada uno de los casos posibles. Este diagrama es de alto nivel y no representa detalles concretos de la implementación del sistema, sino tan solo la lógica de una actividad concreta que puede realizar el mismo.

Capítulo 3

Plataforma SIEM para monitorización de alertas

En esta sección se introduce el análisis y diseño realizado para la implementación y puesta en marcha de la plataforma tipo SIEM para monitorización de alertas de ciberseguridad en una organización. Se estudian por tanto a continuación todos los aspectos relevantes al análisis y fase de investigación del proyecto, además del diseño seguido para la arquitectura del sistema, y la implementación del software a desarrollar.

Como se ha descrito anteriormente, el prototipo a desarrollar pretende servir como herramienta de gestión de logs de diferentes soluciones comerciales de ciberseguridad, de una manera centralizada y multi-tentant (múltiples clientes gestionados desde un mismo sitio). El esquema general del sistema es el que se muestra en la figura 3.1.

Por un lado están las APIs disponibles públicamente de las soluciones comerciales de Cisco que se han usado como parte del proyecto. Estas APIs se gestionarán desde un programa que se debe desarrollar y que gestione la sincronización de estos logs con la base de datos. Todos los datos se almacenarán en una base de datos, y se podrán visualizar en un portal Web o aplicación de visualización de los datos. Los usuarios finales solo tendrán visibilidad sobre la aplicación de visualización de los datos, pudiendo ver, para la organización que aplique, todos los gráficos y alertas recogidos y tratados por el sistema desarrollado.

3.1. Perspectiva del producto

El sistema que se va a proponer consiste en una aplicación para monitorización de logs para controlar alertas de seguridad según los datos recogidos de una serie de herramientas de ciberseguridad comerciales. El sistema estará compuesto por:

- *Base de Datos*: componente del sistema encargado de almacenar y gestionar los datos recogidos de los logs de las diferentes soluciones comerciales.
- *Aplicación visualización*: componente encargado de mostrar los datos recogidos en

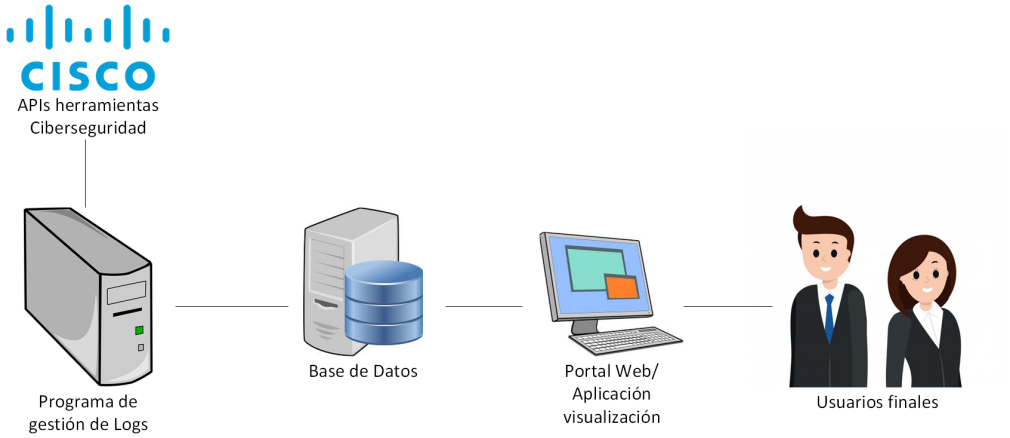

Figura 3.1: Esquema de la arquitectura de la aplicación

la base de datos. Deberá tratar los datos y mostrarlos de una manera gráfica y sencilla.

- *Gestión y tratamiento de Logs*: parte del sistema que se encargará de descargar, actualizar y tratar los logs en la base de datos según la última versión disponible.

El prototipo, como se ha comentado en la Sección 1.2 Alcance, pretende ser una herramienta principalmente de uso interno en Orbe Telecomunicaciones para la monitorización y gestión de alertas de ciberseguridad en los diferentes clientes del departamento de SOC, pero que además pueda ser adaptable en el futuro para poder ser usado también por los clientes finales. Por ello, el sistema a desarrollar deberá ser escalable y fácilmente mantenible, para poder seguir desarrollando sobre él y añadiendo nuevas funcionalidades en el futuro según se vayan necesitando.

El prototipo pretende semejar una herramienta SIEM, que se usa comercialmente en el ámbito de la ciberseguridad, y que permite monitorizar y gestionar eventos y alertas de seguridad que puedan surgir en una organización. La principal diferencia con el prototipo a desarrollar es que una heramienta SIEM completa permite, además de monitorizar, gestionar esas mismas alertas y tomar las medidas necesarias para solventar el problema. Por la naturaleza propia de las APIs a integrar, esto no es siempre posible, y por eso el prototipo solo permitirá la monitorización de alertas, lo que sería más similar a un SIM (*Security Information Management*).

3.2. Evolución del sistema

Para desarrollar el prototipo se ha buscado una solución de base de datos que permita almacenar los logs y que se pueda comunicar con facilidad con la aplicación de monitorización. Además, la gestión de logs se realizará de manera automática abstrayendo las APIs de las distintas aplicaciones de manera que se traten todas de la misma manera en la parte del programa de gestión de logs. Se deberá estudiar el funcionamiento

de las mismas y la manera de obtener los datos deseados para integrar en el sistema a desarrollar.

Todo esto se implementará sobre un servidor físico alojado en Orbe, de manera que tanto el almacenamiento de datos, como su recolección, como el software encargado de la visualización y monitorización, estén todos alojados en un mismo servidor alojado en el CPD de Orbe.

3.3. Requisitos del sistema

A partir de esta descripción del producto se definen dos categorías de requisitos, diferenciando los requisitos funcionales de los no funcionales. Estos requisitos los deberá cumplir el sistema, aunque algunos podrían no llegar a implementarse como parte del prototipo. Los requisitos funcionales son aquellos que se refieren a la funcionalidad del sistema, por ejemplo de cara a los distintos roles de usuarios finales. Estos requisitos se refieren a las acciones que se podrán realizar, y a las funcionalidades que deberá cumplir el sistema. Por otro lado, los requisitos no funcionales se refieren a todo aquello no relacionado con las funcionalidades finales del sistema, pero sí con características que deba cumplir, como pueden ser condiciones de disponibilidad o accesibilidad.

Se definen los requisitos correspondientes a la funcionalidad final del sistema, enumerando los distintos aspecto que éste deba satisfacer:

RF 1. Cada usuario podrá iniciar sesión introduciendo su nombre de usuario único y contraseña.

RF 2. Al iniciar sesión, a cada usuario se le permitirá elegir un espacio entre los que tenga disponible.

RF 3. Habrá dos tipos de usuario, dependiendo de si es un usuario del SOC, o un usuario cliente.

RF 3. Los usuarios cliente solo podrán acceder al espacio de su organización.

RF 4. Los usuarios cliente podrán visualizar los datos de su organización, pero no podrán modificar la visualización de los mismos.

RF 5. Los usuarios cliente podrán gestionar los usuarios con acceso al espacio de su organización.

RF 6. Los usuarios cliente no podrán modificar su espacio ni los datos que se muestran en él.

RF 7. Los usuarios del SOC podrán realizar todas las tareas de los usuarios clientes.

RF 8. Los usuarios del SOC podrán modificar los espacios de los clientes.

RF 9. Los usuarios del SOC podrán gestionar los usuarios de los diferentes espacios, así como a otros usuarios del SOC.

RF 10. Los usuarios del SOC podrán generar los informes de los espacios de los clientes.

Se exponen a continuación los requisitos relativos a necesidades, condiciones o exigencias del sistema:

RNF 1. La aplicación será accesible vía web.

RNF 2. La aplicación solamente será accesible de manera local a la red de Orbe.

RNF 3. La aplicación deberá permitir un mínimo de 5 usuarios concurrentes.

3.4. Funcionalidad del producto

En este apartado se definen una serie de casos de uso que se darán en la aplicación de visualización. Estos casos corresponden tanto a un usuario del SOC de Orbe (usuario administrador), como a un usuario final de una organización cliente (usuario cliente). Todos los usuarios administradores heredan de los clientes, de manera que puedan realizar sus mismas tareas además de las tareas concretas de administración. Se muestra el diagrama de casos de uso en la figura 3.2.

1. Caso *Gestionar Espacios*: los usuarios administradores podrán gestionar los distintos espacios disponibles, pudiendo crear o modificar espacios existentes, así como configurar cada espacio y los usuarios que puedan acceder a él.

2. Caso *Eliminar espacio*: los usuarios administradores podrán eliminar espacios existentes.

3. Caso *Gestionar Usuarios SOC*: los usuarios administradores podrán gestionar al resto de usuarios del SOC, definiendo el acceso a espacios para cada usuario.

4. Caso *Generar Informe*: los usuarios administradores podrán generar el informe de una organización.

5. Caso *Visualizar Datos*: los usuarios cliente pueden visualizar los datos de su propia organización, para poder monitorizar y revisar alertas de seguridad.

6. Caso *Configurar Alertas*: los usuarios cliente pueden configurar las alertas que quieren recibir en tiempo real.

7. Caso *Gestionar Usuarios Espacio*: los usuarios cliente pueden gestionar al resto de usuarios que tengan acceso al espacio de su organización.

8. Caso *Acceder al Portal*: Todos los usuarios podrán acceder al portal web para acceder a la aplicación.

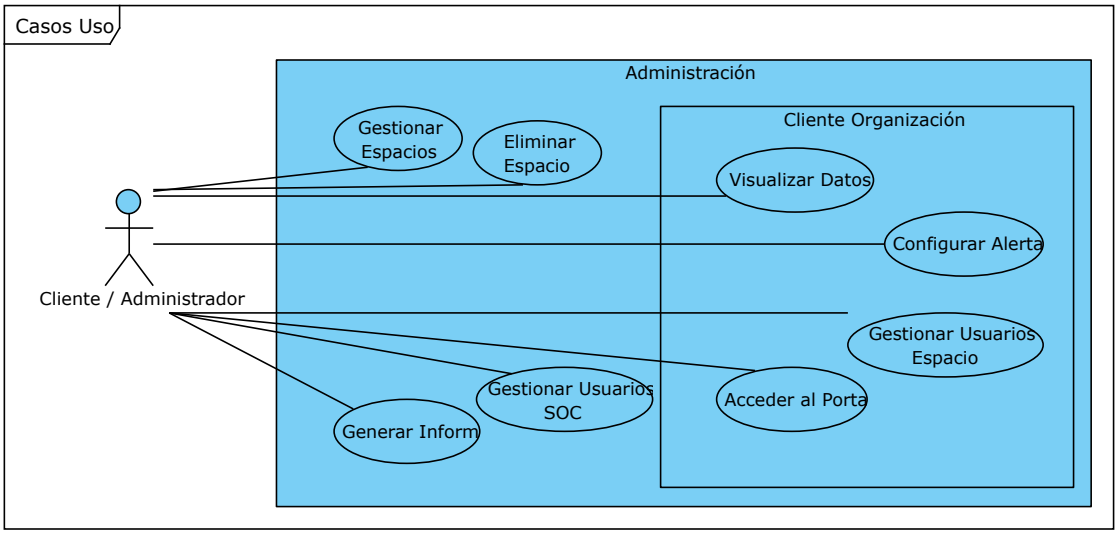

Figura 3.2: Diagrama UML casos de uso

3.5. Sistema Base de Datos

De todas las soluciones analizadas, se ha optado por usar Elastic Stack para la implementación ya que es la herramienta probada que tiene más opciones de configuración y customización, siendo además de uso no comercial en todas las funcionalidades requeridas para el proyecto.

Por tanto, Elastic Stack es un conjunto de herramientas de código abierto que incluye:

1. *Elasticsearch* [13]: base de datos no relacional basada en texto, diseñada para poder realizar búsquedas de forma muy rápida incluso con grandes cantidades de datos. Se deberá crear un índice para cada fuente de datos de cada cliente. Dentro de dicho índice se almacenarán todos los datos relativos a una API de un cliente, de manera que un cliente pueda tener múltiples índices diferentes.

2. *Kibana* [14]: interfaz de usuario vía web para la visualización de datos y gráficos. Está integrado directamente con Elasticsearch (definiendo en la instalación la dirección del servidor de Elasticsearch), de manera que Kibana podrá analizar un índice de la base de datos para reconocer el tipo de datos y poder visualizarlos. Será necesario realizar la configuración manualmente, así como definir la visualización concreta que se desea de los datos según el tipo de fuente de datos a emplear.

3. *Beats* [2]: plugin empleado para enviar datos desde diferentes máquinas. Permite recoger datos de diferentes fuentes, ya sean sistemas externos o plugins de monitorización de equipos, de manera que automáticamente esos datos se carguen en Elasticsearch y se visualicen en Kibana. Este plugin no se usará en el proyecto ya que no ofrece integración oficial con las herramientas a utilizar. Por ello, todo el

tratamiento y carga de datos se realizará con un software específico desarrollado para el proyecto.

4. *Logstash* [15]: plugin de recopilación y tratado de logs, que permite realizar el proceso automáticamente según las reglas que se definan. Al igual que con Beats, Logstash no se usará como parte del proyecto ya que el tratamiento de los logs se realizará por el software a desarrollar para la gestión de logs.

Todo el Stack está diseñado para que las 4 partes funcionen de manera conjunta, facilitando así en gran medida la conexión entre base de datos y plataforma de visualización. De cara al proyecto, es necesario definir en Elasticsearch un índice de datos donde se almacenarán los logs de cada solución de ciberseguridad de cada cliente (varios índices por cliente). Además, para poder visualizar los datos correctamente, también es necesario definir en Kibana el formato de dichos datos, de manera que por ejemplo los campos de fecha se reconozcan como tal y se puedan filtrar resultados por rango de tiempo.

3.6. RESTful APIs

Toda la información recolectada por el sistema, y que proviene de las soluciones de Cisco anteriormente mencionadas, será recolectada a través de una serie de APIs. Una API (*Application Programming Interface*, interfaz de programación de aplicaciones, en español), es un conjunto de funciones que permite interactuar con un sistema con un cierto nivel de abstracción, de manera que no sea necesario conocer la implementación del mismo. Las APIs REST, o RESTful APIs, son un tipo de APIs accesibles vía protocolos existentes. En concreto, REST (*Representational State Transfer*) puede ser usado sobre casi cualquier protocolo, pero concretamente para APIs web, como es el caso que se está tratando, generalmente usa el protocolo HTTP.

Estas APIs, permitirán por tanto recibir a través del protocolo HTTP, información de los logs almacenados por las soluciones de ciberseguridad que se van a integrar. Estos datos se recibirán en formato JSON (*JavaScript Object Notation*), y deberán ser parseados y tratados para poder ser cargados al servidor.

Según la herramienta concreta a integrar, la API o APIs varían, por tanto es necesario estudiar la documentación de cada una de estas herramientas para poder lograr obtener los datos deseados. Para las tres herramientas a integrar, sus respectivas APIs funcionan de la siguiente manera:

1. *Umbrella*: Para Umbrella existen 4 APIs diferentes, que serían [9]:
   - *Network Devices*: para la integración de Umbrella con equipos de red, como pueden ser routers, firewall, o puntos de acceso wifi.
   - *Legacy Network Devices*: igual que la API de *Network Devices*, para integración de equipos, en este caso para equipos legados como *Cisco Wireless Lan Controllers* o *Cisco Integrated Services Routers (ISE) serie 4000*.
   - *Reporting*: API para realizar consultas de eventos de seguridad y destinos (dominios e IPs) concretos.

- *Management*: para la gestión de organizaciones, redes, clientes y políticas de seguridad.

Según estas 4 APIs disponibles, la que mejor se adapta al alcance del proyecto es la API de Reporting, ya que permite obtener todas las conexiones detectadas relacionadas con eventos de seguridad. Según la documentación, un comando `curl` base para realizar una consulta sobre las últimas 24 horas tendría el siguiente formato:

```
curl -X GET --url 'https://reports.api.umbrella.com/v1/
organizations/org-id/security-activity' --header
'Authorization: Basic clave-api-base64'
```

Las únicas variables serían el ID de la organización (org-id), y la clave de la API en formato Base64 y uniendo clave y secreto separado por ':' con el formato clave:secreto [9].

2. *AMP*: En AMP solo existe una API, aunque existen dos versiones de la misma [5]:

   - *Versión 0*: versión original de la API, solo permite obtener información de los equipos, eventos ocurridos en esos equipos, y gestión de los grupos definidos.
   - *Versión 1*: permite, además de todo lo permitido en la versión original, consultar información de eventos según el tipo de evento ocurrido, modificar políticas, grupos y configuración de equipos, consultar información de ficheros y procesos, indicadores de compromiso, y software vulnerable detectado. Por ello, esta nueva versión, también permite realizar la gestión del sistema a nivel de API.

   Al ser la versión 1 la más completa, se usará esta versión como la que se integrará en el sistema. Sin embargo, tan solo se empleará la funcionalidad de realizar consultas avanzadas sobre la información recopilada.

   Un ejemplo de consulta de los eventos producidos en las últimas 24 horas sería:

   ```
   curl --location --request GET
   'https://api-pair@api.eu.amp.cisco.com/v1/events'
   ```

   Las únicas variables en esta API será el par de *api-pair*, que se corresponderá a la pareja clave:secreto de los datos de la API generada [5].

3. *CES*: En la solución de correo electrónico Cisco CES, existen dos portales de gestión diferentes. El primero es ESA (*Email Security Appliance*), que se emplea para toda la configuración de los servidores de corre electrónico, relays de entrega, rutas SMTP, políticas y reglas definidas. Por otro lado, el SMA (*Security Management*

Appliance) es el que se emplea para consultar la información de los correos, pudiendo ver el tracking de cada uno de los correos entrantes, así como información de los diferentes motores de bloqueo para cada correo recibido [7].

Al ser dos portales diferentes y con utilidades distintas, cada uno dispone de su propia API. Como parte del proyecto, solo se integrará la API del portal de SMA, es decir, del portal de monitorización que permite consultar la información de seguridad de los correos.

El esquema de una consulta básica de esta API es similar a las dos anteriores estudiadas para Umbrella y AMP. El comando `curl` para realizar una consulta sobre los eventos de las últimas 24 horas sería:

```
curl --location --request GET
'https://dhXXX-sma1.eu.iphmx.com/sma/api/v2.0/'
--header 'Authorization: Basic clave-api-base64'
```

Las variables de esta consulta serían la dirección del appliance (dhXXX-sma1), cambiando XXX por el número concreto del appliance en uso proporcionado por Cisco, y por último el par clave:secreto en formato base64 [7].

Las 3 APIs de las 3 soluciones integradas se consultan mediante protocolo HTTPS, de manera que la conexión y descarga de los datos se realiza de manera segura y cifrada. Además, los datos descargados se vuelcan en un fichero temporal que solo se almacenará hasta que el sistema cargue los datos, de manera que la única versión de los logs descargados estará en la base de datos y no en ficheros del servidor desde donde se realice la descarga y actualización de los logs.

3.7. Diseño

El diseño del prototipo se compone de tres bloques principales, siendo uno de ellos el bloque de Elastic Stack y por tanto de todo el lado de servidor encargado de la BBDD y portal Web; por otro lado estará el bloque de la gestión de logs, que se encargará de actualizar los logs con las APIs y de subir esos logs al servidor de Elasticsearch. Por último estará el bloque de Cisco, que es donde se encuentran alojadas las 3 APIs que se van a utilizar en el proyecto. Será necesario conectarse a los servidores de Cisco para las 3 soluciones de ciberseguridad y descargar desde estos servidores la última versión de los logs según la API deseada.

El usuario final tan solo interactuará con Kibana (el portal web de Elastic Stack), por lo que no tendrá control directo sobre la base de datos ni las APIs, como se muestra en el esquema de la figura 3.3. En esta figura se representan tres bloques, en el bloque del servidor web están los componentes de Elastic Stack, es decir Elasticsearch y Kibana, que se comunican entre ellos según la dirección de red en que se encuentre cada componente.

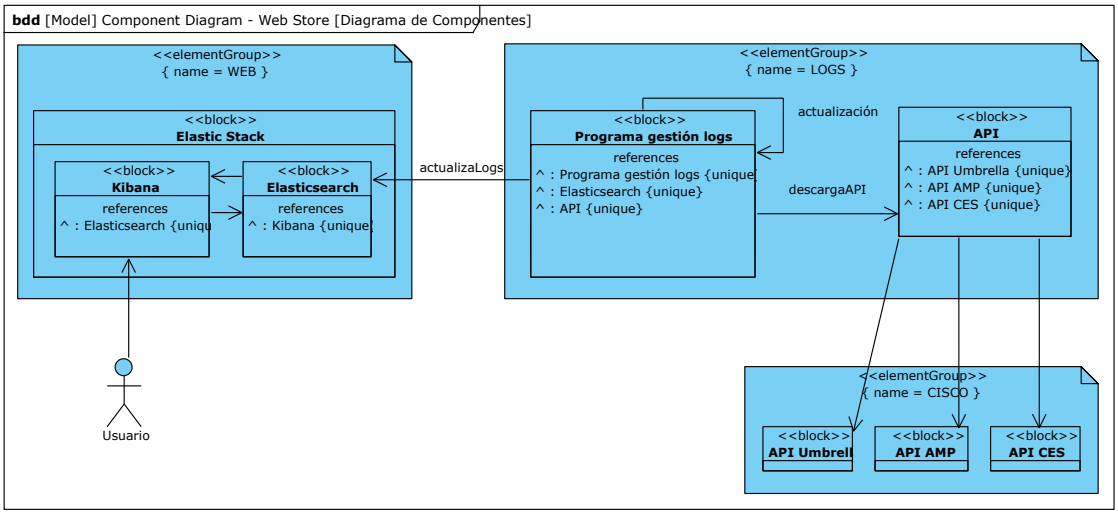

Figura 3.3: Esquema de componentes

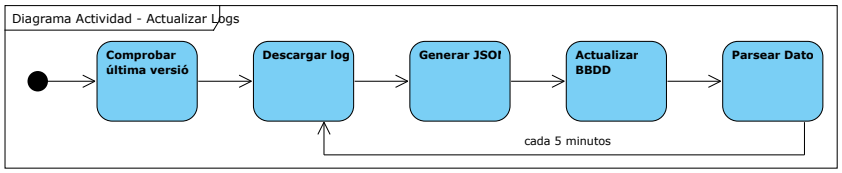

Figura 3.4: Diagrama de actividad - Gestión de Logs

Además, el usuario final accede directamente a Kibana que, recogiendo los datos de la base de datos de Elasticsearch, los muestra según la visualización que se haya configurado.

Otro componente es el programa de gestión de logs. Este se comunica con el bloque del servidor web, de manera que actualizará los logs que se encuentran en Elasticsearch. Para ello, se comunica con las APIs de cada solución de manera genérica, el sistema no tiene por qué conocer el funcionamiento concreto de cada API. El componente de la API representa una API genérica, que en el caso concreto de este proyecto es una de las tres APIs que se listan en el último bloque, que es el bloque de Cisco. Estas tres soluciones, como ya se ha mencionado, son Umbrella, AMP y CES, quienes a través de sus APIs se comunican con el programa de gestión de logs.

3.8. Diagramas del diseño

El diseño general de la implementación de la aplicación es como el diagrama de actividad mostrado en la figura 3.4. Este diagrama afecta principalmente al programa de gestión de logs, quien conectándose a las respectivas APIs (abstraídas como un componente genérico), generar un JSON y actualizar la base de datos de Elasticsearch.

El esquema de la figura 3.4, inicia con la ejecución de un usuario solicitando al programa de gestión de logs para que actualice a la última versión (proceso que por defecto

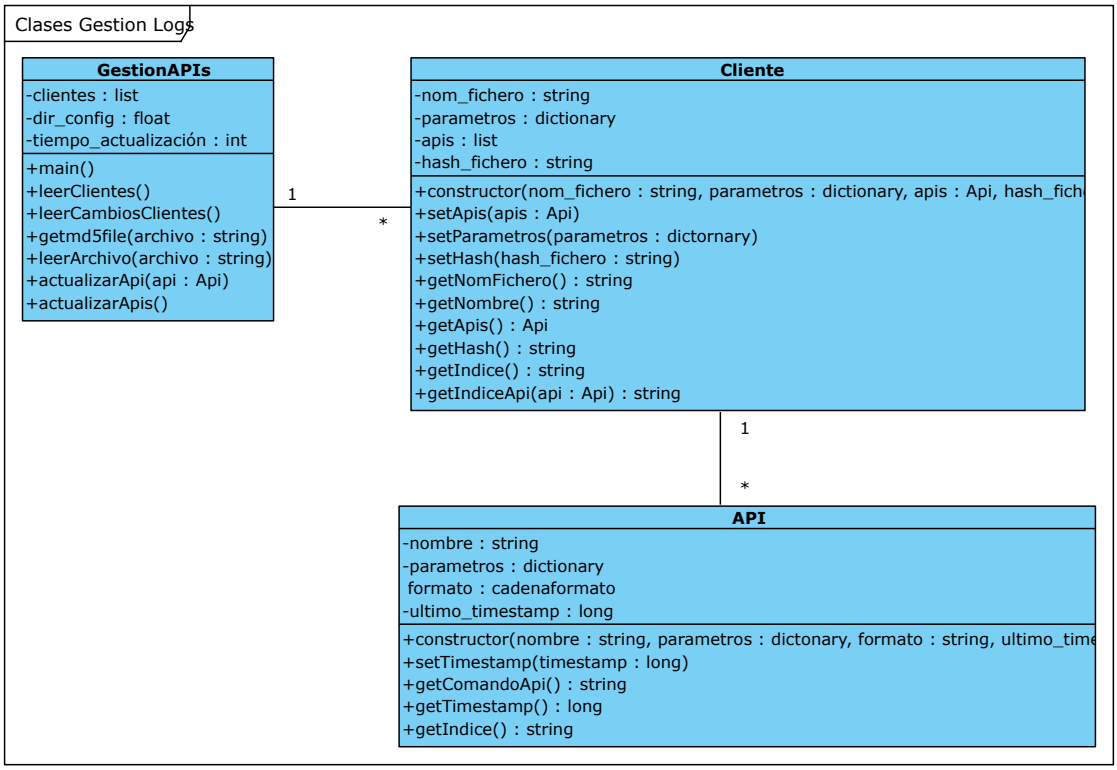

Figura 3.5: Diagrama de clases - Gestión de Logs

se realiza automáticamente cada cierto rango de tiempo a definir). Para ello, necesitará conectarse a la base de datos para comprobar la última versión. Entonces, se conecta a las APIs para descargar esa versión (la última para los últimos 5 minutos, o para el rango leído al comprobar la última versión). Una vez descargados genera el JSON con el formato correspondiente para poder ser leído por Elasticsearch, y lo sube al servidor para actualizar la base de datos. Por último, Elasticsearch parsea los datos según el formato definido para el espacio del cliente correspondiente (según el índice de elasticsearch que se hubiera definido), de manera que estos datos se puedan visualizar en Kibana.

Este proceso de actualización de logs se repite automáticamente cada 5 minutos, con la única excepción representada en el diagrama en que un usuario (administrador) solicite la actualización inmediata.

Para el sistema de gestión de logs, se usará una serie de objetos para gestionar los distintos clientes, APIs, y ficheros de configuración de cada cliente, según el diseño de la figura 3.5.

Según este diseño, la implementación del programa de gestión de logs constará de 3 clases, que son:

- *GestionAPIs*: la clase GestionAPIs es la clase principal y que incluye el código principal. En esta clase se almacenan los objetos Cliente, que corresponden a los diferentes clientes que forman parte del sistema. Tiene además como atributos

el directorio de los ficheros de configuración, y el tiempo por defecto entre cada actualización de los logs.

Las funciones a implementar en esta clase son:

- *leerClientes()*: esta función se encarga de enumerar todos los clientes disponibles. Para ello, en el directorio de configuración, revisa todos los ficheros disponibles, y para cada uno de ellos llama a la función *leerArchivo()*, que se encargará de procesar el contenido del fichero y devolver el objeto Cliente correspondiente. Una vez devuelto, esta función lo añade a la lista de clientes del atributo de la clase.
- *leerCambiosClientes()*: esta función revisa si ha habido cambios en algún cliente. Para ello, primero revisa el listado de clientes guardados, para comprobar si son los mismos que los existentes en el directorio de configuración (si se ha borrado uno, lo borra del listado de clientes, y si hay alguno nuevo llama a *leerArchivo()* y lo añade al listado). Para cada cliente ya existente, comprueba el hash del fichero con el almacenado en el objeto Cliente correspondiente. Si el hash hubiera cambiado, indica que el contenido del fichero también, por lo que lo vuelve a leer y modifica los atributos correspondientes. Si el hash se corresponde con el ya almacenado, no se modifica el cliente.
- *getmd5file()*: función que devolverá el hash de un fichero según el algoritmo MD5.
- *leerArchivo()*: lee el contenido del archivo que se pasa por parámetro, y según ese contenido crea las Apis correspondientes y el objeto Cliente. Devuelve el Cliente creado según el contenido del archivo.
- *actualizarApi()*: actualiza una API de un cliente. Se pasa la referencia al objeto Api correspondiente. Lee el formato del comando de la Api según la cadena de formato, ejecuta el comando para descargar el JSON con los logs de la API, y por último carga ese log al servidor Elasticsearch.
- *actualizarLogs()*: actualiza todos los logs de todas las APIs de todos los clientes. Realiza esta actualización según el tiempo indicado en el atributo correspondiente de la clase. Además, también comprueba posibles cambios en los ficheros de clientes mediante la función *leerCambiosClientes()*. Esta función *actualizarLogs()* es la que ejecuta siempre el programa para actualizar todos los logs, realizar la espera, y comprobar los cambios de clientes.

- *Cliente*: la clase Cliente almacenará todos aquellos datos relacionados con cada cliente que forme parte del sistema. Esto incluye, primero, el nombre del fichero de configuración relativo a ese cliente, además de los parámetros de dicho fichero en formato diccionario, el listado de objetos API relativos a dicho cliente, y por último el hash del fichero de configuración de ese cliente.

  Todos los métodos implementados son los getters y setters de los atributos de la clase. El único caso especial es la función *getNombre()*, que devuelve el nombre almacenado en el diccionario del objeto (atributo *parametros*).

- *API*: la clase API almacena los datos concretos de la API de un cliente, siendo estos datos el nombre de la API (nombre de la herramienta de la que se obtienen datos), los parámetros de la API en formato diccionario, una cadena de formato que indique el formato de una consulta a la API en función de los parámetros almacenados en el diccionario, y por último el último timestamp de la última actualización de los logs.

  Al igual que el objeto Cliente, el objeto Api solo implementa las funciones de getters y setters. La función *getComandoApi()* hace uso de la cadena de formato para devolver el comando de la API en función de los parámetros almacenados. La función *getIndice()* devuelve el campo ïndice"del diccionario de los parámetros del objeto.

Con esta implementación se almacena un fichero de configuración para cada cliente. Este archivo tiene todos los datos de un cliente, incluyendo los datos de las APIs de las diferentes soluciones de ciberseguridad de dicho cliente. Con estos datos, y según el tiempo de actualización de 5 minutos, el programa comprueba los ficheros de configuración y si hay algún cambio (algún fichero nuevo, alguno desaparece, o alguno se modifica), actualiza en el objeto correspondiente los datos de ese cliente. Una vez actualizados los datos, se descargan uno por uno logs de cada solución de cada cliente, se genera un archivo JSON temporal y se cargan los datos al servidor. Una vez cargados, al haber definido en Kibana el formato de los mismos y la manera de representarlos, instantáneamente los datos ya se pueden visualizar. Estos datos se descargan solo para los últimos 5 minutos (guardando el *timestamp*, que es la firma de tiempo, de la última actualización), y si en ese rango de tiempo no hubiera datos (la API no devuelve nada), no se carga nada (como puede ser el caso por ejemplo de madrugada). Por último, este programa de gestión de logs está siempre en ejecución, y todos los cambios en los clientes se hacen en los ficheros de configuración para que el programa se actualice en caliente (no se detiene la ejecución).

Capítulo 4

Puesta en marcha del sistema

Este capítulo introduce todos los aspectos necesarios para la implementación y puesta en marcha del sistema. Esto incluye todos los componentes como puede ser el servidor en que se alojará la base de datos, el programa de gestión de logs o la aplicación de visualización.

4.1. Servidor Sistema SIEM

Todo el sistema se va a implementar en un único servidor, por motivos de simplicidad a la hora de realizar la comunicación entre los diferentes componentes. Para el sistema operativo se ha optado por la distribución CentOS por ser la alternativa gratuita al popular sistema operativo RedHat, ampliamente usado a nivel corporativo a la hora de elegir una distribución para un servidor Linux. Además, la solución antimalware usada en Orbe (que es donde se aloja el servidor), y que es AMP, es compatible únicamente con las distribuciones Linux de RedHat y CentOS. Esto permitirá también ofrecer una capa de seguridad al servidor pudiendo securizarlo con una herramienta antimalware que ayude también a proteger los datos alojados en el propio servidor.

Este sistema inicialmente se había montado en un equipo en Orbe que tenía una capacidad de procesamiento limitada. Esto causó problemas de rendimiento y cuelgues, por lo que finalmente se ha migrado todo el sistema a una máquina virtual en un servidor VMWare alojado también en Orbe. Según las pruebas realizadas en la primera iteración funcional del sistema, se ha calculado que será necesario al menos 16GB de memoria RAM para un funcionamiento fluido de la descarga y carga de logs, tratamiento de los datos, comunicación con Kibana, y visualización por parte del usuario final. Además, según lo especificado en los requisitos del sistema, debe soportar al menos 3 usuarios simultáneos realizando consultas al servidor. Se ha calculado también 1TB de almacenamiento para los logs, que en el futuro deberá ser ampliado, y 4 núcleos de procesador para poder realizar las diferentes tareas de manera simultánea y sin retrasos. Este sistema, al estar alojado internamente en Orbe, solo es accesible a través de una IP interna de la organización (el prototipo solo es de uso interno, si se publicara para clientes finales sería necesario definir un dominio público que apuntara a la IP del servidor).

4.2. Programa de gestión de logs

El programa de gestión de logs se ha implementado en Python, y su funcionalidad principal consiste en descargar los logs de las diferentes APIs y subir esos logs en formato JSON a la base de datos de Elasticsearch. Se ha elegido Python como lenguaje de programación por la facilidad de uso, permitiendo también una ejecución rápida y eficiente que no afecte al rendimiento del sistema. Según lo analizado en la Sección 3.8 Diagramas de Diseño, se han implementado cuatro clases en cuatro ficheros independientes.

Para almacenar los datos de cada organización, se usan ficheros de configuración, de manera que cada organización tiene un fichero. El programa de gestión de logs lee todos estos ficheros y gestiona en variables locales su contenido. Además, almacena también el hash de cada fichero, de manera que en cada iteración puede comprobar si el hash ha cambiado y por tanto ha cambiado su contenido. De esta manera, únicamente en la primera lectura del fichero, y luego en cada modificación, el sistema leerá el contenido y actualizará las variables locales. Si no hubiera cambios, el sistema no lee el fichero ya que se entiende que las variables locales están actualizadas.

Para cada cliente leído, se almacenan N objetos API para almacenar los datos de la API en concreto. Por un lado se almacenan todos los parámetros de la misma, y que varían de una API a otra. Para solventar el problema de que cada API es diferente, también se indica una cadena de formato que representa el formato concreto de la API en relación con las variables almacenadas en el fichero de configuración. De esta manera, el programa de gestión de logs solamente necesita leer el formato de la API y completarlo con las variables que se indican y que están almacenadas en el objeto correspondiente. Esta solución permite que el sistema pueda realizar conexiones a cualquier tipo de API de manera abstracta y sin conocer el formato de la misma.

Esto, junto con la abstracción de los clientes y los ficheros de configuración, permite que el sistema sea lo más flexible posible de cara al futuro. Idealmente, el programa de gestión de logs está siempre corriendo (no se debería detener nunca), y por ello la implementación tampoco se puede modificar. De esta manera, abstrayendo todo lo posible el sistema, se logra que con la misma implementación, el sistema se pueda adaptar a nuevos formatos de datos o nuevas APIs de nuevas soluciones que se quieran integrar en el sistema.

4.3. Puesta en marcha Elastic Stack

Elastic Stack requiere montar múltiples servidores para las aplicaciones que componen el stack (Elasticsearch, Kibana, Beat, y Logstash). Por simplicidad se ha decido montar todas las aplicaciones en un mismo servidor. En concreto, el esquema del servidor de Elastic Stack queda como:

- Elasticsearch, base de datos basada en texto que almacena los logs de todas las aplicaciones de todos los clientes que se gestionen. Este servicio está funcionando en el puerto 9200 de la máquina.

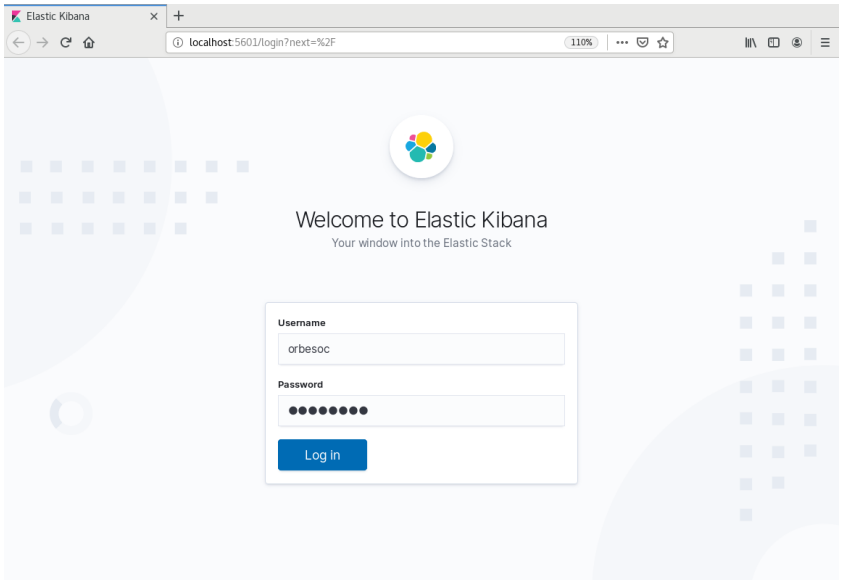

Figura 4.1: Captura sistema inicio de sesión

- Kibana, aplicación web a la que acceden los usuarios finales. Está funcionando en el puerto 5601 del servidor.
- Beats, aplicación para la recogida de logs. No se usará como parte del proyecto.
- Logstash, aplicación para tratamiento de logs (en principio no se usará como parte del proyecto pero se deja montado para estudiar su uso más adelante una vez se integren más fuentes de información diferentes a las mencionadas).

El acceso a las instancias de Elasticsearch y Kibana se ha securizado, de manera que sea necesario introducir un usuario y contraseña para poder leer o escribir en ambos. Para Kibana, se crearán cuentas concretas para todos los posibles usuarios finales.

4.4. Sistema SIEM

Una vez implementado el sistema, se ha configurado el portal web para definir los usuarios concretos que requieren acceso, el nivel de permisos de cada usuario, y las visualizaciones de los datos cargados en el sistema. La captura de la figura 4.1, muestra el portal de inicio de sesión del portal, en el caso de la captura, con un usuario genérico llamado *orbesoc*.

Una vez se inicia sesión, según los permisos de cada usuario, se deberá seleccionar el espacio sobre el que trabajar. Estos espacios son únicos por cliente, además de un espacio general que incluye a todos los clientes y que solo es accesibles por usuarios del SOC de Orbe. En la figura 4.2, se pueden ver los dos espacios disponibles: espacio General, y el espacio de Orbe. Cada espacio de cada cliente tiene el logo de la organización para poder

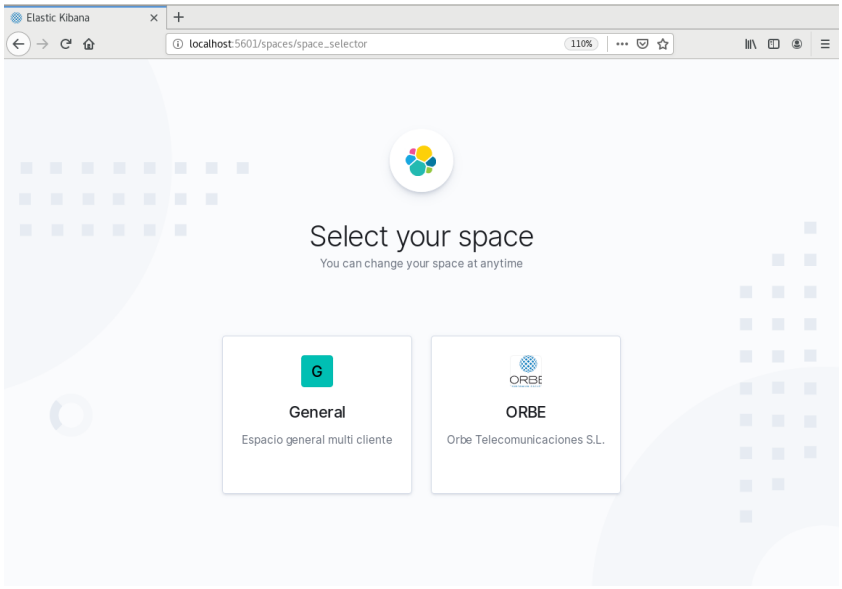

Figura 4.2: Captura sistema selección de espacio

identificar fácilmente cada espacio. Una vez seleccionado el espacio de trabajo, también se puede cambiar y moverse a otro espacio. Si un usuario solo tiene acceso a un espacio, como es el caso de un usuario cliente, no se preguntará el espacio y automáticamente se inicia su único espacio de trabajo.

Una vez en el espacio, están disponibles los diferentes dashboards que se han definido para ese cliente. Estos dashboards dependen de las visualizaciones de los datos que se han definido, así como de las soluciones integradas. En el ejemplo de la figura 4.3, se puede observar un dashboard con los datos de Cisco Umbrella, donde se muestra primero un mapa de calor con las principales categorías de seguridad detectadas, un gráfico con los usuarios o equipos de origen con más conexiones maliciosas, seguido de una tabla con la información detallada de dichas conexiones. Estos gráficos son completamente configurables y se pueden adaptar a las necesidades concretas de cada cliente.

4.5. Pruebas

Las pruebas de uso realizadas se corresponden a la comprobación del correcto funcionamiento del sistema. Se debe probar que el sistema responda correctamente, sea capaz de actualizar los datos de manera automática sin interacción por parte del usuario, y que también sea capaz de mostrar estos datos de forma gráfica y según los *dashboards* y visualizaciones que se han definido en Kibana.

Se ha probado el proceso completo de añadir un nuevo cliente, que sería como sigue:

- **Definir fichero configuración del cliente**. Se crea el fichero en el directorio de configuración del programa de gestión de logs, que contiene los datos del cliente y

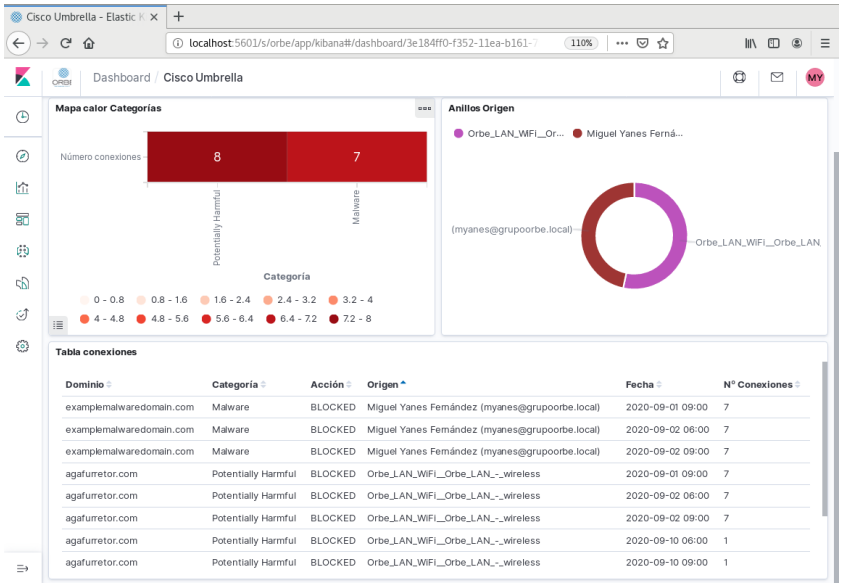

Figura 4.3: Captura sistema dashboard Cisco Umbrella

al menos una API a integrar.

- **Crear dashboard en Kibana**. En cuanto el programa de gestión de logs lo lee, crea el índice y actualiza los datos. Con el índice creado, hay que crear un *index pattern* en Kibana que permita la visualización de los datos de dicho índice. Debería reconocer automáticamente el formato de los datos, y sino será necesario definirlo manualmente.

- **Crear visualización en Kibana**. Una vez creado el *index pattern*, se pueden crear las visualizaciones que se quieran para esos datos (gráficas, tablas...).

- **Dar acceso a un usuario o crear uno nuevo**. Con un usuario existente, o uno nuevo que se cree, definir los permisos de acceso al nuevo dashboard creado (al menos permiso de lectura para la visualización y el dashboard).

- **Visualizar los datos**. Comprobar que ese cliente realmente puede visualizar correctamente los datos cargados.

Se ha realizado este proceso completo para comprobar el correcto funcionamiento del sistema. Ha sido necesario retocar algunos aspectos del mismo, como la manera de crear índices en el programa de gestión de logs, y se han probado variaciones como por ejemplo eliminar un cliente existente, o modificarlo. El sistema ha respondido correctamente y ha funcionado según lo deseado.

Capítulo 5

Conclusiones y trabajo futuro

Este último capítulo de la memoria del proyecto analiza las conclusiones obtenidas de todo el proceso realizado, así como del prototipo final que se ha implementado. Se indican además los principales problemas encontrados durante el desarrollo del prototipo, y cómo se han solucionado. Por último, se indican una serie de aspectos que no se han integrado como parte del sistema pero que sí que se han investigado como parte de la fase de análisis, y que por tanto se podrían llegar a añadir al sistema en el futuro.

5.1. Conclusiones obtenidas

El sistema final implementado consiste en una herramienta de monitorización de alertas que puede llegar a resultar muy útil como parte de un departamento de SOC. El sistema es ágil y ligero de cara al usuario, pero es importante tener en cuenta que los requisitos y carga del sistema se irán incrementando con el tiempo al añadir nuevos clientes, y al haber almacenado datos de varios meses en vez de tan solo unos días como se ha implementado como parte del prototipo.

Ha sido necesario dedicar más tiempo del planificado a los apartado de investigación y análisis, ya que existen múltiples soluciones disponibles que o bien se podrían emplear, o bien pueden servir como referencia a la hora de diseñar el sistema y sus funcionalidades. El sistema final es relativamente similar a otras soluciones SIEM disponibles comercialmente, con la ventaja de que se ha logrado montar el sistema mediante soluciones sin coste, o bien desarrollando manualmente las partes que fueran necesarias para conectar las diferentes partes. Por ello, se ha conseguido que el coste de mantenimiento y el coste de producción del sistema sea muy bajo. En caso de añadir, modificar o eliminar clientes, tan solo se necesitaría modificar el fichero de configuración del mismo y, si fuera necesario, definir la configuración en Kibana de ese cliente (visualizaciones y dashboards disponibles, o usuarios ya sean de Orbe o externos con acceso al portal del cliente).

5.2. Problemas encontrados

Durante las fases de investigación y puesta en marcha del sistema, se han ido encontrando una serie de problemas que se han debido solucionar. El principal problema es debido al servidor en el que se estaba desarrollando el sistema, dado que no disponía de una capacidad a nivel de hardware suficiente. El equipo empleado originalmente como servidor disponía de un procesador antiguo y tan solo 4GB de RAM, lo cual causó problemas de rendimiento en algunas fases de pruebas. Es por ello que finalmente, y una vez se ha podido probar la carga de procesamiento y memoria que tiene el sistema, se ha migrado a un servidor virtualizado con más capacidad de procesamiento y memoria.

5.3. Trabajo futuro

A continuación se listan todos los aspectos estudiados que no se han llegado a integrar en el sistema. Como ya se han estudiado, para cada aspecto se indica brevemente también el proceso necesario a alto nivel para poder adaptar esa funcionalidad.

5.3.1. Reportes automáticos

Un aspecto interesante del prototipo es la posibilidad de generar informes de ciberseguridad de manera automática. Estos informes contendrían, principalmente, un resumen de la información obtenida por el sistema para todas las soluciones de un cliente, y durante un rango de tiempo definido. Esto se puede realizar mediante la funcionalidad de reporting que forma parte del plugin *X-PACK*. Este plugin tiene diferentes funcionalidades, algunas de ellas como parte de la versión de pago. Otra alternativa es usar funcionalidades sueltas (por ejemplo la securización del acceso a los servidores de Elasticsearch y Kibana se realiza con una funcionalidad de X-PACK), pero que deben ser activadas y configuradas manualmente mediante una serie de archivos de configuración. Para activar la funcionalidad de reporting es necesario modificar el fichero de configuración de Kibana y reiniciar el servidor. Para poder ser usado por un usuario final, es necesario que este usuario tenga permisos sobre el rol de *reporting_user* dentro de Kibana. [12]

Por último, una manera de mejorar y customizar estos reportes sería extrayendo toda la información generada e integrándola en un documento PDF customizado con la imagen corporativa de Orbe.

5.3.2. Configurar alertas personalizadas

Otro aspecto interesante del sistema sería poder definir alertas automáticas, de manera que el sistema pudiera notificar a un administrador o analista de seguridad sobre una posible amenaza en curso en alguno de estos clientes. Para definir estas alertas, sería necesario revisar en tiempo real toda la información cargada a la base de datos. Para ello, se podría implementar un módulo extra que cada vez que se cargasen los datos de un cliente a Elasticsearch, pudiese comparar y analizar estos datos con datos históricos ya disponibles para ese cliente, pudiendo por ejemplo detectar anomalías en tiempo real.

Otro modo más sencillo sería definir las alertas según tipo de eventos, ya que las tres herramientas integradas definen el tipo de evento, y en el caso de AMP incluso la gravedad del mismo. Así, con tan solo definir el tipo de eventos o gravedad, se podrían configurar alertas que se generasen automáticamente con cada carga de datos al servidor.

Bibliografía

[1] Alienvault.
https://cybersecurity.att.com/.

[2] Beats.
https://www.elastic.co/es/beats/.

[3] Cisco.
https://www.cisco.com/c/es_es/about.html.

[4] Cisco AMP.
https://www.cisco.com/c/es_es/products/security/advanced-malware-protection/index.html.

[5] Cisco AMP API Documentation.
https://api-docs.amp.cisco.com/api_resources?api_host=api.eu.amp.cisco.com&api_version=v1.

[6] Cisco CES.
https://www.cisco.com/c/es_es/products/security/email-security/index.html.

[7] Cisco CES API Documentation.
https://docs.ces.cisco.com/docs/api.

[8] Cisco Umbrella.
https://www.cisco.com/c/dam/global/es_mx/solutions/security/pdf/cisco-only-umbrella-mx.pdf.

[9] Cisco Umbrella API Documentation.
https://docs.umbrella.com/umbrella-api/docs/reporting_api_overview.

[10] ECCouncil Cybersecurity significance.
https://blog.eccouncil.org/the-evolving-significance-of-cybersecurity/.

[11] Elastic Stack.
https://www.elastic.co/es/elastic-stack.

[12] Elastic X-PACK Documentation.
https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-xpack.html.

[13] Elasticsearch.
https://www.elastic.co/es/what-is/elasticsearch.

[14] Kibana.
https://www.elastic.co/es/kibana.

[15] Logstash.
https://www.elastic.co/es/logstash.

[16] Orbe Telecomunicaciones S.L.
https://www.orbe.es/.

[17] Security Information and Event Management.
https://blogs.imf-formacion.com/blog/tecnologia/que-significa-siem-y-como-funciona-201808/.

[18] Security Operation Center.
https://opendatasecurity.io/es/soc-que-es-y-por-que-es-interesante-para-tu-empresa

[19] Splunk.
https://www.splunk.com/.

[20] Unified Modelling Language.
https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml.

[21] Wazuh.
https://documentation.wazuh.com/3.13/index.html.

Apéndice A

Definiciones y acrónimos

A continuación se enumeran y definen los distintos términos técnicos empleados en el presente documento:

- **SOC**: *Security Operations Center*. Departamento de Ciberseguridad de una organización centrado en monitorizar y gestionar alertas.
- **SIEM**: *Security Information and Event Management*. Herramienta de monitorización y gestión de eventos de ciberseguridad generalmente utilizada dentro de un SOC.
- **SIM**: *Security Information Management*. Similar a un SIEM, un SIM permite únicamente monitorizar eventos de ciberseguridad, pero no permite gestionar esos eventos ni realizar acciones para mitigarlos.
- **DNS**: *Domain Name Server*, o servidor de nombres de dominio, es un servidor encargado de "traducir" un dominio de un servidor a su IP correspondiente, de manera que un ordenador sepa a que IP realizar la conexión.
- **ISP**: *Internet Service Provider*, es el proveedor de internet de una organización, y que habitualmente también dispone de un servidor DNS propio.
- **API**: *Application Programming Interface*, es una funcionalidad de un sistema que permite a componentes externos realizar funciones en ese sistema, como puede ser modificar parámetros o pedir que devuelva una serie de datos almacenados o recogidos por ese sistema.
- **JSON**: *JavaScript Object Notation*, es un formato de almacenamiento de datos ampliamente usado, y que se usa en el proyecto para almacenar temporalmente los logs recogidos por el sistema.
- **Umbrella**: Cisco Umbrella. Solución de protección DNS para prevenir conexiones de malware, Command & Control, phishing o cryptomining.

- **AMP**: Cisco Advanced Malware Protection. Solución antimalware para los equipos finales de usuarios y servidores.
- **CES**: Cisco Cloud Email Security. Solución de seguridad del correo electrónico.
- **Multi-tenant**: una arquitectura multitenant o de *tenencia múltiple* es aquella que se ejecuta en un único servidor pero que es accesible por diferentes clientes u organizaciones, de manera que cada organización tenga su propia instancia de la aplicación.
- **UML**: *Unified Modelling Language*, es un estándar a seguir para la documentación del análisis y diseño de un proyecto informático.
- **RF**: Requisitos funcionales.
- **RNF**: Requisitos no funcionales.
- **Parsear**: proceso de analizar un texto para comprobar y actualizar el formato del mismo.
- **BBDD**: Bases de Datos
- **CURL**: herramienta de terminal para realizar peticiones a protocolos web

Apéndice B

Horas de Trabajo

Este último apéndice de la memoria del proyecto lista el listado de las horas invertidas en el proyecto a lo largo de los últimos meses. Se dividen estas horas entre las diferentes fases del proyecto, pudiendo solapar algunas de esas fases en momentos puntuales. Las fases definidas son:

- *Investigación*: fase de investigación del proyecto. Incluye desde la definición del alcance del proyecto, como también todas las pruebas realizadas con diferentes sistemas para estudiar su uso y su posibilidad de implementarlo en el proyecto. Por último, incluye toda la parte de documentación e investigación de los aspectos concretos del proyecto, listados todos ellos en el capítulo 2. Se ha dedicado un total de 110 horas a lo largo de los primeros 3 meses del proyecto.

- *Análisis y diseño*: esta fase se corresponde al análisis del sistema y sus funcionalidades, así como del diseño que debe seguir el mismo. Esta fase ha supuesto un total de 35 horas a lo largo de cerca de 2 meses.

- *Puesta en marcha, implementación y configuración*: puesta en marcha del sistema, incluyendo el servidor CentOS, servidor con todas las herramientas de Elastic Stack, implementación del programa de gestión de logs, y configuración de todas las partes para que pudieran funcionar correctamente entre ellas. Esta fase ha durado 87 horas y cerca de un total de 4 meses.

- *Pruebas*: la fase de pruebas ha consistido en testear el correcto funcionamiento del sistema (tanto base de datos, como portal web, como programa de gestión de logs), y ha supuesto 12 horas de tiempo.

- *Documentación*: la fase de documentación ha consistido en el desarrollo de esta memoria, incluyendo su redacción y edición. Ha supuesto un total de 82 horas a lo largo de 2 meses.

- *Reuniones*: se han realizado un total de 17 reuniones a lo largo de 6 meses, en las últimas fases del proyecto siendo estas semanales. Estas reuniones han tenido una

duración de aproximadamente media hora, lo cual suma un tiempo contado de 10 horas.

Tabla horas dedicadas al proyecto		
Fase	*Horas dedicadas*	*Duración*
Investigación	110 horas	3 meses
Análisis y diseño	35 horas	2 meses
Puesta en marcha, implementación y configuración	87 horas	4 meses
Pruebas	12 horas	1 mes
Documentación	82 horas	2 meses
Reuniones	10 horas	6 meses
Total	336 horas	6 meses

Tabla B.1: Tabla horas dedicadas al proyecto