Universidad Zaragoza

Trabajo Fin de Grado

Sistema Integrado para la Gestión Centralizada de Logs

Integrated System for Centralized Log Management

Autor

Iván Moreno Sanz

Directores

Ricardo Julio Rodríguez Fernández

Víctor Pérez Roche

ESCUELA DE INGENIERÍA Y ARQUITECTURA
Septiembre 2023

RESUMEN

En el panorama actual de seguridad de la información es crucial contar con herramientas eficientes para la detección y mitigación de amenazas cibernéticas. El *threat hunting*, o caza de amenazas, se ha convertido en una práctica esencial para identificar y neutralizar proactivamente estas amenazas, pero requiere de sistemas de registro y análisis de *logs* que permitan una búsqueda y correlación eficiente de eventos. Un *log* se define como la información de nivel más bajo generada por los procesos que ocurren en sistemas o aplicaciones en respuesta a un evento, siendo un evento la ocurrencia identificada en el estado de un sistema, servicio o red que puede ser relevante para la seguridad.

El presente Trabajo de Fin de Grado se centra en la implementación y evaluación de una arquitectura de gestión centralizada de *logs* utilizando tecnologías de código abierto con el objetivo de dar cumplimiento a los requerimientos legales de retención de registros de actividad de usuario, detección de anomalías y potenciales amenazas de ciberseguridad necesarios en una Administración Pública.

La arquitectura propuesta se basa en el uso de `Syslog-ng`, que se encarga de recopilar y modificar los *logs* generados por los diferentes activos y dispositivos de una red para su posterior tratamiento. Estos *logs* son almacenados y procesados posteriormente en `Elasticsearch`, una herramienta de búsqueda y análisis de datos donde se realiza su parseo y normalización. La combinación de `Syslog-ng` y `Elasticsearch` junto con el resto de herramientas que se usan en este trabajo proporciona una solución eficaz para el *threat hunting*, facilitando la correlación de eventos y la detección de patrones de actividad sospechosa.

Además de la implementación de la arquitectura propuesta, el proyecto también incluye la evaluación de varios casos de uso que demuestran la eficacia de la solución. Estos casos de uso incluyen la detección de intentos de acceso no autorizados a sistemas y recursos, el seguimiento del tráfico generado por un usuario sospechoso y la detección de accesos a la red a través de cuentas privilegiadas.

Por último, este documento establece una serie de medidas de seguridad que deben ser implementadas por las organizaciones públicas para proteger su información y sus sistemas de información, según el Real Decreto 311/2022, de 3 de mayo, por el que se regula el Esquema Nacional de Seguridad (ENS). Entre estas medidas se encuentra la salvaguarda de los *logs* de actividad de usuario.

ABSTRACT

In the current information security landscape, it is crucial to have efficient tools for detecting and mitigating cyber threats. *Threat hunting* has become an essential practice for proactively identifying and neutralizing these threats, but it requires logging and log analysis systems that enable efficient search and correlation of events. A log is defined as the lowest-level information generated by processes occurring in systems or applications in response to an event, which is the identified occurrence in the state of a system, service, or network that can be relevant for security.

This Bachelor's Thesis focuses on the implementation and evaluation of a centralized log management architecture using open-source technologies with the goal of complying with legal requirements for retaining user activity records, detecting anomalies, and potential cybersecurity threats necessary in a Public Administration.

The proposed architecture is based on the use of `Syslog-ng`, which is responsible for collecting and modifying logs generated by different network assets and devices for subsequent processing. These logs are stored and processed in `Elasticsearch`, a data search and analysis tool, where parsing and normalization take place. The combination of `Syslog-ng` and `Elasticsearch`, along with the other tools used in this work, provides an effective solution for *threat hunting*, facilitating event correlation and the detection of patterns of suspicious activity.

In addition to implementing the proposed architecture, the project also includes various use cases that demonstrate the solution's effectiveness. These use cases include detecting unauthorized access attempts to systems and resources, tracking traffic generated by a suspicious user, and detecting network access through privileged accounts.

Finally, this document establishes a series of security measures that public organizations must implement to protect their information and information systems, according to the Royal Decree 311/2022, of May 3rd, which regulates the National Security Framework (ENS). Among these measures is the safeguarding of user activity logs.

Índice

Índice de Figuras

Capítulo 1

Introducción

1.1. Antecedentes y justificación del proyecto

Actualmente, la mayoría de los sistemas de información y comunicación generan *logs* (también llamados registros) que registran información detallada sobre su funcionamiento y su estado. En las organizaciones se encuentran *logs* como los del firewall, DNS, Windows, Nginx, bases de datos, etc. Los más habituales (como los *logs* de sistema, servidores web o servidores de bases de datos) son conocidos, pero también hay otros como los *logs* de Sistemas de Alimentación Ininterrumpida (SAI), de *Integrated Lights-Out* (ILO) o de *Integrated Dell Remote Access controller* (iDRAC) [2].

El manejo eficiente de *logs* se ha vuelto crucial para garantizar la seguridad y el correcto funcionamiento de los sistemas informáticos, ya que estos contienen información valiosa sobre eventos, acciones y comportamientos que ocurren en un sistema, convirtiéndolos en una fuente importante de datos para la monitorización, el análisis de incidentes y la detección de problemas en entornos informáticos. Desafortunadamente, a menudo se desaprovecha esta capacidad debido a la ausencia de un modelo de referencia estándar. Esto lleva a tener que diseñar arquitecturas propias, que no garantizan una definición común en términos de formato, método de transporte o almacenamiento de *logs*.

Aunque existen diversas aplicaciones comerciales completamente orientadas a la gestión de *logs*, su enfoque se centra en grandes empresas capaces de asumir los elevados costes asociados a su implementación y soporte. Por esta razón, el objetivo de este trabajo es implementar una solución integral basada en código abierto asumible para pequeñas y medianas empresas.

En el caso particular de la Universidad de Zaragoza (UNIZAR), una institución educativa con una infraestructura considerable, se manejan numerosos sistemas y aplicaciones que generan *logs* de manera continua. Sin embargo, actualmente la organización y gestión de *logs* que se lleva a cabo no es adecuada, lo que dificulta

el acceso y la utilización de esta información.

En este Trabajo de Fin de Grado se propone la implementación de un sistema integrado para la gestión centralizada de *logs* en UNIZAR, siguiendo la normativa descrita para el almacenamiento y tratamiento de estos definida según el Real Decreto 311/2022, de 3 de mayo, por el que se regula el Esquema Nacional de Seguridad (ENS). El resultado de este proyecto permitirá centralizar y gestionar los *logs* que generan los diferentes sistemas y aplicaciones, facilitando la búsqueda y filtrado de la información relevante. Con este enfoque, los analistas podrán acceder rápidamente a la información necesaria para la detección de incidentes de seguridad y la monitorización del rendimiento de los sistemas, entre otros objetivos.

1.2. Objetivos y alcance del trabajo

El objetivo general de este proyecto es desarrollar un sistema integrado mediante el uso de herramientas de software libre que permita la gestión centralizada de *logs*, enfocándose en la observación, organización y análisis de eventos generados por diversos sistemas y aplicaciones. Este Trabajo de Fin de Grado se ha realizado durante unas prácticas en el Servicio de Informática Central de la Universidad de Zaragoza (SICUZ), por lo que el proceso de diseño e implementación del sistema se guiará por el cumplimiento de UNIZAR con las leyes y regulaciones aplicables para el almacenamiento y manejo de registros.

Se definen también una serie de objetivos intermedios como la correcta organización de los *logs* en el servidor central y la detección de incidencias en tiempo real, así como la posterior notificación mediante los medios oportunos.

1.3. Estructura de la memoria

Esta memoria está organizada en cinco capítulos. El Capítulo 1 contiene la introducción, los objetivos y el alcance del trabajo a realizar. El Capítulo 2 se centra en explicar los conceptos necesarios. El Capítulo 3 trata sobre el desarrollo y la implementación del sistema centralizado propuesto. El Capítulo 4 comenta las pruebas llevadas a cabo y los resultados obtenidos y por último, en el Capítulo 5 se exponen las conclusiones y posibles líneas futuras de este trabajo.

Además, se incluye un apéndice A que contiene la planificación temporal del trabajo desarrollado.

Capítulo 2

Conocimientos previos

En este capítulo se comentan los fundamentos teóricos relacionados con la gestión de *logs* y se aborda el software que será usado y las funciones que este implementa para entender el desarrollo del proyecto.

2.1. Gestión de logs y sistemas de registro

Dependiendo del origen o tipo del que se trate, un *log* podrá ser auto generado o programado para que se generen cuando ocurra cierta acción. Los *logs* provienen de muchas fuentes, como por ejemplo:

- Sistemas Linux y Windows.
- Aplicaciones.
- Routers.
- Switches.
- Sistemas de seguridad: firewalls, IPS, IDS...
- Puntos de acceso Wi-Fi.
- Controladores de puntos de acceso.

Estas fuentes generadoras de *logs* pueden clasificarse en dos categorías: fuentes *push-based* y fuentes *pull-based* [3]. En las fuentes *push-based*, el dispositivo o aplicación envía un *log* al disco local o a través de la red cada vez que ocurre un evento. Estas fuentes generan *logs* a medida que ocurren y el receptor los recoge de manera pasiva, sin necesidad de solicitarlos. Al contrario, en la categoría *pull-based* las fuentes no se encargan de enviar los *logs* al receptor, si no que este pide explícitamente su envío

(es decir, el receptor envía solicitudes de obtención de *logs* de manera periódica o programada y la fuente responde proporcionando los *logs* pertinentes).

Antes de analizar los *logs*, es necesario realizar un filtrado y normalización. La normalización es la acción de recoger los *logs* enviados por las distintas aplicaciones, que tendrán un formato diferente, y convertirlos a un formato común. De esta manera el análisis y manejo de *logs* se simplifica [4]. El filtrado de *logs* se puede realizar en cualquiera de los equipos, tanto en el sistema generador como en el sistema centralizador o en los equipos intermediarios. En este trabajo se utilizan distintas herramientas de línea de comandos (como `grep`, `awk` y `sed`), así como las aplicaciones `Filebeat` y `Logstash`, descritas posteriormente.

También es necesario implementar procesos de autenticación, confidencialidad e integridad en el envío de *logs*, tanto para los sistemas generadores como para el servidor central. La autenticación garantiza que solo los usuarios autorizados puedan acceder a los *logs*, la confidencialidad protege los *logs* de ser leídos por usuarios no autorizados y la integridad consigue que la modificación de *logs* por usuarios no autorizados sea detectable [3].

2.2. Tecnologías y herramientas para la gestión centralizada de logs

En esta sección se habla de las herramientas comúnmente utilizadas para la administración de *logs*, que facilitan la búsqueda, visualización, correlación y análisis de *logs*.

2.2.1. Syslog

`Syslog` es un protocolo estándar para la recolección, el procesamiento y el transporte de mensajes de registro del sistema [5]. Los equipos y servicios con soporte para envío de *logs* a este sistema pueden enviar cualquier tipo de mensaje. Los registros más habituales se refieren a eventos de seguridad, errores, notificaciones de eventos, etc, aunque pueden contener cualquier información. Junto con cada mensaje, se incluye la fecha y hora del envío, el equipo que lo envía, la prioridad y otros datos adicionales.

La arquitectura del protocolo está basada en un equipo servidor ejecutando el proceso syslogd (demonio de `syslog`) y clientes que envían un pequeño mensaje de texto (habitualmente de menos de 1024 bytes) a este servidor. Los mensajes de `syslog` se suelen enviar como texto plano.

Algunas implementaciones del servidor, como `Syslog-ng`, permiten usar TCP en vez de UDP, y también ofrecen la posibilidad para que los datos viajen cifrados

mediante SSL/TLS [6]. Aunque `syslog` tiene algunos problemas de seguridad, su sencillez ha hecho que muchos dispositivos lo implementen, tanto para enviar como para recibir. Esto hace posible integrar mensajes de varios tipos de sistemas en un solo repositorio central.

Rsyslog

`Rsyslog` es un sistema de procesamiento de registros de sistema que implementa el protocolo `syslog` básico y lo extiende agregando filtros y una configuración flexible. Es un sistema eficiente y rápido que ofrece un diseño modular de alto rendimiento y niveles de seguridad apropiados [7].

Soporta una variedad de fuentes de datos, incluidos archivos, registros de eventos y aplicaciones. Puede almacenar registros en una variedad de destinos, incluidos archivos, bases de datos y sistemas de correo electrónico. Ofrece una variedad de filtros para ayudar a los usuarios a controlar qué registros se almacenan y cómo se presentan. También es altamente configurable, lo que permite a los usuarios adaptarlo a las necesidades específicas de su sistema.

NXLog

A diferencia de otras soluciones de recolección de registros que se centran en sistemas operativos basados en *Linux*, `NXLog` está especialmente diseñado para adaptarse a las necesidades de recolección y envío en entornos *Linux* y *Windows*. Para ello, cuenta con módulos integrados que permiten recopilar registros de diversas fuentes, como el Visor de eventos de *Windows*, servicios y aplicaciones específicas [8].

`NXLog` puede integrarse con Sysmon [9], una herramienta de Microsoft que monitoriza y registra actividades de alto nivel. Estos registros pueden ser enviados directamente a `Elasticsearch`, `Logstash` u otros sistemas de análisis y almacenamiento compatibles. Esto brinda la flexibilidad de integrar `NXLog` con herramientas y plataformas existentes para el análisis y la visualización de datos.

Syslog-ng

`Syslog-ng` consta de la misma base teórica que `Rsyslog` y las diferencias entre estos dos son principalmente técnicas. Estas diferencias se explican más detalladamente en la Sección 3.1.2.

2.2.2. Elasticsearch

`Elasticsearch` es un motor de búsqueda y análisis en tiempo real, diseñado para indexar y buscar grandes volúmenes de datos de forma eficiente. Combina su

funcionalidad con el enriquecimiento de *logs* a través de software como `Logstash` y Beats. Proporciona soporte para todo tipo de datos, ya sean datos numéricos, geo espaciales o texto estructurado y no estructurado [10].

`Elasticsearch` también dispone de módulos que permiten el envío de alertas, la securización de los *logs*, implementación de algoritmos de aprendizaje automático, gestión del ciclo de vida de índices e integraciones con distintos tipos de bases de datos entre otros.

Está basado en Apache Lucene y ofrece una API de tipo RESTful (interfaz de programación de aplicaciones que se basa en el protocolo HTTP) para el manejo y la consulta de datos. También es posible hacer uso del lenguaje de peticiones que proporciona `Elasticsearch` (QueryDSL), en vez de su API.

Actualmente, se usa `Elasticsearch` para referirse a toda la pila en conjunto (Beats, `Logstash`, `Elasticsearch` y `Kibana`; véase la Figura 2.1), aunque en este trabajo cuando se hable de `Elasticsearch` se estará haciendo referencia al motor de indexación.

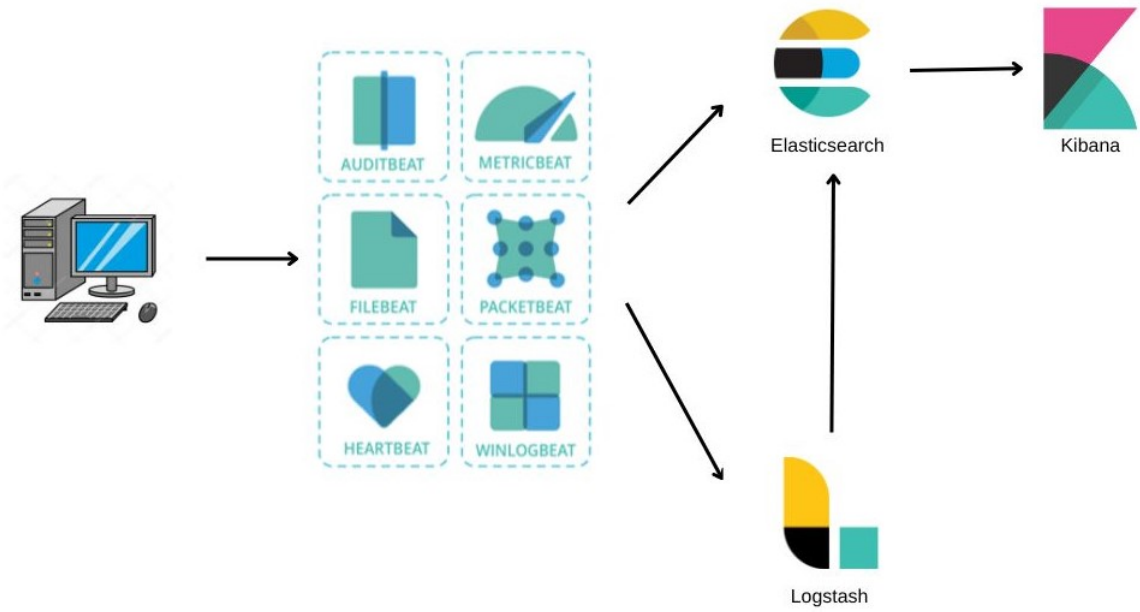

Figura 2.1: Esquema Elasticsearch

2.2.3. Kibana

`Kibana` es una plataforma de visualización y análisis de datos que se alimenta de la información almacenada en índices de `Elasticsearch`. Proporciona una interfaz web que permite ver los datos de manera sencilla mediante *dashboards* (colección de visualizaciones y búsquedas) [11]. Otra funcionalidad importante de `Kibana` es la generación de informes y presentaciones visuales. Los usuarios pueden generarlos

basándose en los datos recogidos en las *dashboards*. Además, `Kibana` es altamente personalizable: dispone de una colección de *plugins* que añaden funcionalidades adicionales como la implementación de inteligencia artificial a los datos, visualización de mapas geoespaciales, etc.

2.2.4. Beats

Beats es una familia de tecnologías y herramientas desarrollada por la empresa Elastic, diseñada para la recolección de datos y su envío, permitiendo recolectar datos de diferentes fuentes como ficheros (`Filebeat`), métricas de sistema (`Metricbeat`), tráfico de red (`Packetbeat`) y rendimiento de los sistemas (`Heartbeat`) [12]. Los Beats tienen dos funciones principales: recolectar información de los sistemas y enviarla al destinatario que se indique cada cierto tiempo. En este proyecto solo se hará uso de `Filebeat` y `Metricbeat`, explicados a continuación.

Filebeat

Como ya se ha dicho antes, `Filebeat` es un agente que recopila, analiza y envía datos de registro y eventos desde diferentes fuentes a `Elasticsearch` o `Logstash`. Se puede usar para recopilar eventos de una variedad de fuentes, incluyendo archivos de registro, sistemas de registro, aplicaciones, dispositivos y redes. Una vez que `Filebeat` ha recopilado los datos, los analiza y los convierte en un formato que `Elasticsearch` o `Logstash` puede comprender. Luego, envía los datos a `Elasticsearch` donde se pueden almacenar, buscar y analizar o a `Logstash` para su enriquecimiento.

Metricbeat

`Metricbeat` sigue el mismo mecanismo de funcionamiento que `Filebeat`, solo que en este caso en vez de mandarse ficheros, se leen y mandan estadísticas (métricas) de los sistemas y servicios. Las principales estadísticas que se pueden obtener son el uso de la CPU, memoria o entrada/salida de disco a nivel de sistema.

`Metricbeat` cuenta con módulos internos que se encargan de recolectar y procesar los datos automáticamente. Así como la información recopilada por `Filebeat` es enviada a `Logstash`, en el caso de `Metricbeat` no es necesario el enriquecimiento y pueden ser enviados directamente a `Elasticsearch`.

2.2.5. Logstash

`Logstash` es un motor de recolección de datos de código abierto que se encarga de recibir datos, procesarlos y una vez procesados, mandarlos al destinatario que se

indique en su configuración [13]. `Logstash` recibe los datos en crudo (como texto plano) y se encarga de, mediante el parseado, convertirlos en documentos JSON identificando y extrayendo la información relevante para que sea analizada.

`Logstash` también es la herramienta principal para la aplicación de filtros en el intercambio de *logs* ya que se encarga de abrir tuberías de comunicación (llamadas *pipelines*) en diferentes puertos. Cada pipeline consiste en tres etapas: entrada, filtro y salida. Al disponer de varios puertos en la etapa de entrada, es capaz de conectarse a distintas fuentes de información, aplicando a cada una de ellas diferentes filtrados.

La primera etapa de `Logstash` es la fase de entrada, donde los datos se reciben de las fuentes de datos. Una vez que los datos se han recibido, son procesados por los filtros de `Logstash` para convertirlos a un formato legible para `Elasticsearch`. Mediante el uso de una amplia gama de filtros (predefinidos o personalizados) se convierten los *logs* al formato deseado. Estos filtros realizan modificaciones en el contenido del *log* como aislar valores específicos y asignarlos a variables, eliminar información no significativa, realizar conversiones de formato, buscar y reemplazar valores, corregir errores o enriquecer los datos antes de su almacenamiento. Finalmente, los datos procesados se envían a la etapa de salida, donde `Logstash` los dirige hacia su destino final para el almacenamiento y búsqueda de datos. La versatilidad de las salidas permite a los usuarios configurar múltiples destinos según sus necesidades.

2.2.6. ElastAlert2

`ElastAlert2` es una herramienta de alerta de código abierto [14] desarrollada por Jason Ertel que continúa el proyecto original de ElastAlert [15] y se integra con `Elasticsearch`. `ElastAlert2` se encarga de alertar sobre anomalías, picos o patrones de interés de los datos indexados. Se basa en reglas configurables que definen los criterios para generar alertas, escritas en archivos de configuración YAML. Si alguno de estos criterios se cumple, `Elastalert2` manda una alerta según se indique [16]. En la Figura 2.2 se puede ver detalladamente el proceso que se sigue para la generación de estas alertas.

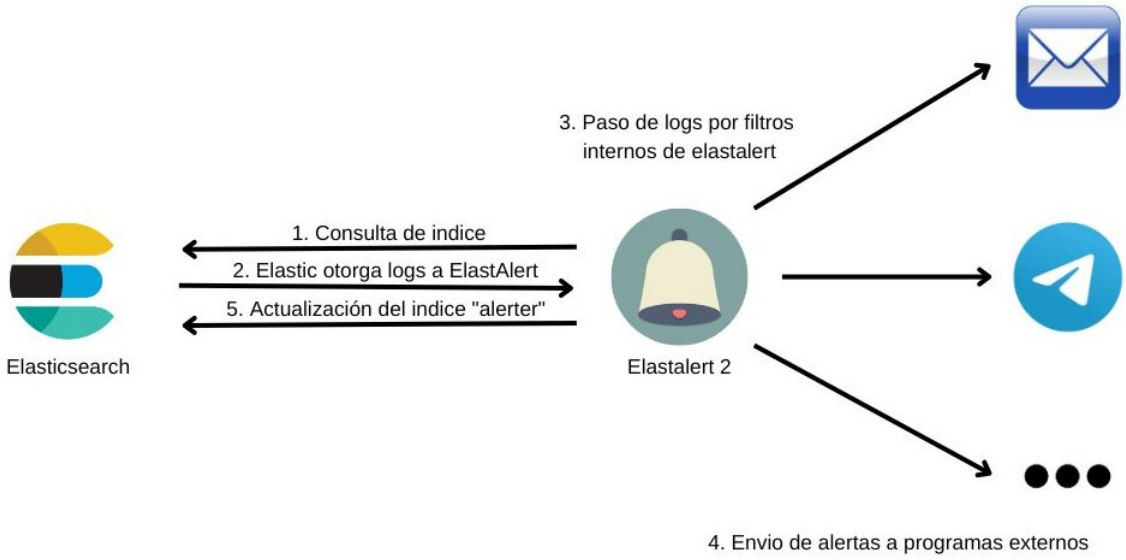

Figura 2.2: Proceso de generación de alertas mediante ElastAlert2

2.2.7. Ansible

`Ansible` es una herramienta de automatización de equipos que permite administrar y configurar sistemas remotos desde un sistema central de manera eficiente y escalable. Se basa en un enfoque declarativo, es decir, se indica el estado en el que ha de acabar la máquina destino al ejecutar el archivo oportuno y `Ansible` se encarga de llegar hasta ese punto [17].

Las máquinas remotas a las que se conecta `Ansible` se definen en un archivo llamado inventario, donde se pueden indicar las máquinas tanto por IP como por nombre de equipo, organizándolas incluso por grupos para ejecutar así las acciones sobre un conjunto de máquinas al mismo tiempo. Las tareas que se van a ejecutar se describen en un fichero específico llamado *playbook*, escrito en formato YAML. Cada tarea representa una acción, así como los permisos específicos para realizarla. Como los ficheros se almacenan en texto plano, `Ansible` dispone de un método de cifrado llamado Ansible Vault, que se encarga de cifrar los datos sensibles haciéndolos solamente accesibles mediante una contraseña maestra.

Capítulo 3

Análisis y diseño del sistema

En este capítulo se presentará la arquitectura del sistema, proceso de análisis y decisiones de diseño e implementación. En la Figura 3.1 se puede ver un esquema de la arquitectura del proyecto, donde se ejemplifica como se producen los flujos de información. Esta arquitectura está compuesta por tres tipos de equipos: *Equipos generadores de eventos* (clientes), *servidor Syslog* y *servidor Elastic*.

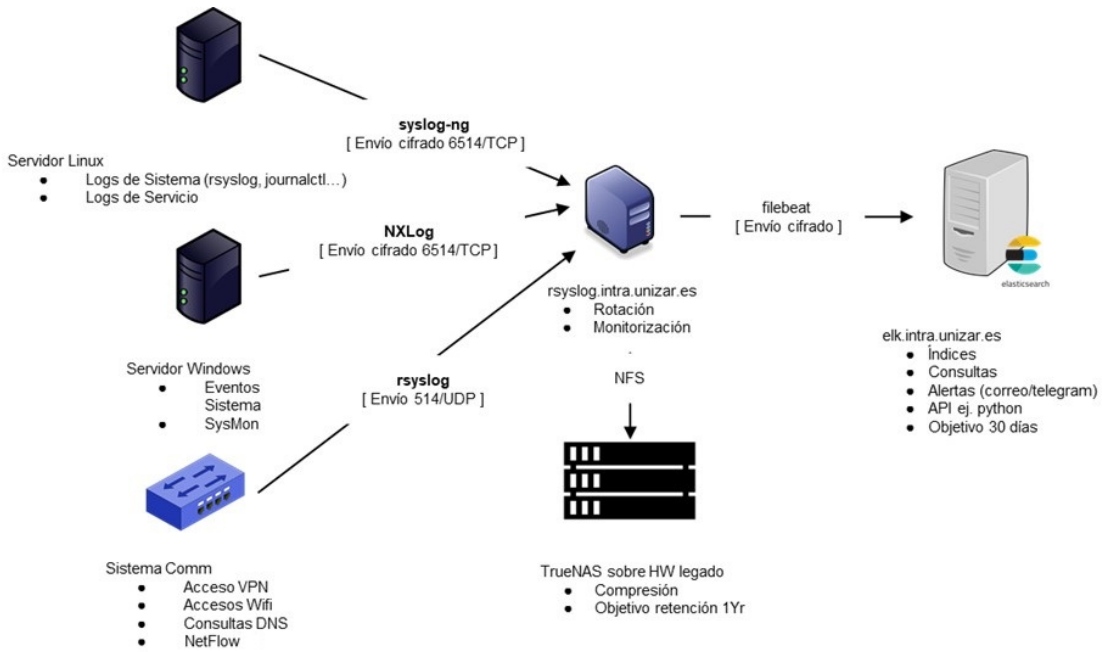

Figura 3.1: Arquitectura del sistema

3.1. Análisis

En esta sección se abordan en primer lugar las distintas alternativas que se valoraron durante el estudio del proyecto, explicando las características propias de cada una y

exponiendo los motivos por los que se eligieron las herramientas finalmente utilizadas.

3.1.1. Elasticsearch vs OpenSearch

Para este proyecto se ha elegido `Elasticsearch` como motor de indexación de documentos por su gran utilización y madurez. `Elasticsearch` está consolidado en el mercado ya que se lanzó en el 2010, mientras que `Opensearch` se lanzó en 2021. Por ello, `Elasticsearch` ya cuenta con una gran comunidad de usuarios que suben información a diversos foros, responden dudas y proponen soluciones a problemas acerca de la pila de `Elasticsearch`. Aún así, una de las mejoras a futuro que se propone es el cambio a `OpenSearch`, ya que ofrece módulos complementarios totalmente gratuitos haciéndolo una gran alternativa. Ambos productos tienen soporte con el resto de herramientas usadas, por lo que la migración de una a otra no supone un gran inconveniente.

3.1.2. Rsyslog vs Syslog-ng

Si bien ambos se presentaron como una actualización sobre el protocolo de registro original conocido como `syslog`, toman rutas bastante diferentes respecto a su implementación. `Rsyslog` fue originalmente solo una mejora sobre syslogd, ya que era una bifurcación del mismo proyecto, usando también la misma sintaxis. `Syslog-ng`, por otro lado, tiene su propia sintaxis y se construyó desde cero, con más funcionalidades y características que el syslogd original.

Una de las principales ventajas de `Syslog-ng` es su flexibilidad. La configuración de `Syslog-ng` es mucho más flexible que la de `Rsyslog`, lo que permite a los administradores de sistemas adaptarlo a sus necesidades específicas. Por ejemplo, `Syslog-ng` permite a los administradores de sistemas definir filtros complejos para seleccionar los registros que desean recopilar. Esto es útil para entornos de producción de gran tamaño, donde se pueden generar millones de registros al día. Otra ventaja de `Syslog-ng` es su escalabilidad, ya que está diseñado para soportar grandes volúmenes de tráfico de registro. `Rsyslog`, por otro lado, puede tener problemas de rendimiento cuando se trata de grandes volúmenes de tráfico de registro. Finalmente, `Syslog-ng` admite una serie de funciones avanzadas que no están disponibles en `Rsyslog`. Estas funciones incluyen compresión de registros, autenticación y cifrado [18].

3.1.3. Kibana Alerting vs ElastAlert2

Para este proyecto se ha elegido `ElastAlert2` debido a su flexibilidad, facilidad de uso y escalabilidad. `ElastAlert2` es más flexible que `Kibana Alerting`, ya que permite crear reglas personalizadas para enviar alertas. El módulo de `Kibana`, por

otro lado, solo tiene reglas predefinidas que pueden no ser adecuadas para todas las necesidades. `ElastAlert2` también permite el envío de alertas a plataformas externas como Slack, Telegram y correo; en cambio, `Kibana Alerting` se limita a enviar alertas exclusivamente a `Kibana`. `ElastAlert2` también es más fácil de usar que `Kibana Alerting`, ya que puede configurarse con solo unos pocos pasos. Además `ElastAlert2` es más escalable, puede manejar más alertas y más datos que el módulo de `Kibana`.

3.2. Diseño

En esta sección se expone en detalle la arquitectura del proyecto, los equipos que la componen y su configuración, tanto de sus propias aplicaciones como la necesaria para permitir comunicaciones con el resto de equipos.

3.2.1. Diseño de la arquitectura del sistema

Como ya se ha comentado antes, la arquitectura del sistema va a constar de diferentes tipos de equipos: clientes, servidor Syslog y la máquina encargada de gestionar `Elasticsearch`.

Entre los clientes se disponen de 3 tipos principales, aquellos cuyo sistema operativo es *Linux*, *Windows* o *Sistema Comm*, en los que se recopilan todos los casos especiales que no entran en los dos anteriores, pero más especialmente los sistemas de comunicaciones como *switches* o *routers*. Estos clientes son los encargados de recolectar los eventos que generan los *logs* que suceden en el entorno de estos clientes, y quedan registrados en ellos en primera instancia, para posteriormente ser enviados al servidor *rsyslog.intra.unizar.es*. Este traspaso de *logs* se realiza mediante `Syslog-ng` y a través del puerto 6514/TCP en entornos *Linux*, con `NXLog` y a través del puerto 6514/TCP en entornos *Windows* y por último con `Rsyslog` y a través del puerto 514/UDP en el resto de equipos (estos últimos equipos también pueden transmitir los *logs* a través de `Syslog-ng`). Una vez los *logs* lleguen al servidor Syslog, se aplican las políticas de rotación, etiquetado y almacenamiento pertinentes y se envían los *logs* a `Elasticsearch` (localizado en *elk.intra.unizar.es*) mediante `Filebeat`.

En `Elasticsearch` se realiza la monitorización y visualización de *logs* gracias a `Kibana`. Concretamente, se organizan los *logs* recibidos de los distintos servicios mediante índices y se añaden funcionalidades adicionales como la explotación de estos índices para el envío de alertas mediante `ElastAlert2` y Python. En el caso de este proyecto las alertas se envían a Telegram y LUCIA. LUCIA es una herramienta desarrollada por el Centro Criptológico Nacional para la gestión de ciberincidentes en las entidades del ámbito de aplicación del Esquema Nacional de Seguridad (ENS)

[19]. Además de los destinos nombrados anteriormente, las alertas también se guardan en `Elasticsearch` para su posterior análisis y visualización.

3.2.2. Máquina cliente

Como ya se ha explicado, la máquina cliente es la encargada de recolectar los *logs* de las distintas aplicaciones y herramientas del sistema para después mandarlos al servidor Syslog y que puedan ser procesados.

Uso de Syslog-ng en equipos Linux

`Syslog-ng` va a ser el encargado de realizar el proceso de recolección, etiquetado y envío en los clientes *Linux*. Para ello, se configura en un fichero las fuentes de las que recoger los *logs*, el etiquetado a aplicar y por último el destino final donde mandarlos.

Se ha diseñado una arquitectura de ficheros que se reciben desde el servidor Syslog mediante `Ansible` cuando se realiza la configuración inicial. Estos ficheros están personalizados para cada aplicación e incluyen el fichero local del que se tienen que recoger los *logs* y el etiquetado a aplicar. Se tendrá otro fichero de configuración por cada servicio que añadirá una etiqueta común para todas los *logs* de esa máquina, indicando el servicio al que pertenece.

Con estos ficheros de configuración se consigue que cuando se manden los *logs* al servidor Syslog la organización y búsqueda de *logs* sea sencilla, ya que gracias a las etiquetas se dispone de una estructura normalizada y coherente.

Uso de NXLog en equipos Windows

En los equipos con sistema operativo *Windows*, `NXLog` va a ser el encargado de realizar el proceso de recolección, etiquetado y envío.

En cuanto a la recopilación y el envío de *logs*, se configura una entrada y una salida de *logs*. La sección de entrada utiliza un módulo llamado *im_msvistalog* que recopila los *logs* de eventos de *Windows*. Además se transforman estos *logs* y se reemplazan caracteres especiales por espacios, para facilitar la legibilidad y el análisis de estos. Por otra parte, la sección de salida emplea un módulo llamado *om_udp*, que habilita el envío de *logs* a través del protocolo UDP a un servidor remoto. Estas dos secciones se conectan entre sí, logrando la recopilación de *logs* desde un archivo, su transformación al formato deseado y su posterior envío al servidor Syslog.

3.2.3. Servidor Syslog

El servidor Syslog es donde se recogen los *logs* enviados por el servidor cliente mediante `Syslog-ng`, que se encarga de organizarlos por carpetas según la configuración indicada. Después `Filebeat` accede a estas carpetas y realiza el envío de *logs* a la máquina Elasticsearch. En el equipo Syslog también se lleva a cabo la configuración de clientes mediante `Ansible`.

Configuración de Syslog-ng

`Syslog-ng` es el encargado de recolectar los *logs* recibidos por los clientes. Como se ha comentado anteriormente, hay distintos tipos de clientes y cada uno de ellos envía los *logs* de una manera diferente. Por ello, hay que configurar distintos tipos de fuentes de datos. De esta manera, es `Syslog-ng` el encargado de abrir los puertos para escuchar a las máquinas clientes y recibir correctamente los *logs*. La conexión de los equipos *Windows* y *Linux* se realiza mediante TLS/SSL, por lo que hay que indicar en el fichero de configuración los certificados necesarios para realizar esta conexión de manera segura. También se ha de indicar el número máximo de conexiones permitidas para evitar la saturación del sistema.

Una vez se han recibido los *logs*, `Syslog-ng` se encarga de enviarlos a la carpeta local que les corresponda, guiándose por las etiquetas añadidos por el cliente. Todos los *logs* se envían en primer lugar a la carpeta `/srv/rsyslog/`, donde se estructuran según el código de servicio, nombre del equipo cliente, proceso, *facility* y fecha en la que se ha generado (en formato *aaaammdd*). Cuando llega el primer log se crean las carpetas y el fichero con el formato necesario para almacenarlo. De esa manera, todos los *logs* posteriores que lleguen con el mismo formato se indexan en el fichero, almacenando así todos en un mismo lugar.

En este caso se ha usado esta configuración de ficheros porque es la más conveniente para este proyecto, pero la configuración es bastante flexible, pudiendo modificarla fácilmente añadiendo nuevas etiquetas en la máquina cliente o utilizando otras ya existentes gracias a `Syslog-ng`.

Base de datos de certificados

Siempre que es posible las comunicaciones entre servidor y clientes se realizan mediante el uso de certificados generados por una Autoridad de Certificación (CA). El servidor Syslog cuenta con una base de datos para gestionar el uso de estos certificados, de manera que desde él se pueden revocar, renovar, etc.

Esta base de datos se almacena en un fichero de texto. Este fichero contiene los

números de serie de estos certificados, si son válidos o ya han caducado, la fecha de validez o la fecha en la que el certificado fue revocado y la información del titular del certificado. De esta manera se puede ver fácilmente los datos de los usuarios que cuentan con un certificado y revocar o renovar el acceso de un cliente con el ID de éste.

También se cuenta con un fichero de configuración de certificados que se aplica a la hora de la creación de estos. Cuando se crea un certificado, se accede a este fichero para conocer los valores de configuración esenciales como la longitud de la clave a usar y el tipo de esta, o algunos campos de interés como el nombre de la organización al que pertenece el cliente, etc.

En el caso de este proyecto y por motivos de seguridad, los certificados se crean con una validez máxima de un año. Una vez transcurrido este período, se ha de llevar a cabo un análisis para determinar si es o no necesario renovar el certificado. Es importante señalar que en cualquier momento existe la posibilidad de revocar un certificado de un cliente en caso de compromiso de seguridad o si ya no resultara necesario recibir *logs* de este cliente por cualquier motivo.

Configuración y uso de Filebeat

`Filebeat` va a ser el encargado de recolectar los distintos *logs* recibidos desde las máquinas cliente y enviarlos a `Logstash`, situado en la máquina Elastic. Para esto es necesario configurar las rutas en los que se encuentran los ficheros de *logs* que se quieren recolectar. Estas rutas son las creadas anteriormente por `Syslog-ng`. También es necesario configurar la salida de los ficheros que encuentre `Filebeat`, que en este caso será `Logstash`. Para ello habrá que indicar la dirección IP que utiliza `Logstash` y el puerto que se utiliza para la conexión.

En el caso de usar módulos para el envío de *logs*, se activan mediante línea de comandos. `Filebeat` lee los módulos activos, donde se indican los ficheros de donde recolectar los diferentes *logs*. Una vez recogidos los *logs* de estos ficheros, `Filebeat` aplica los filtros definidos en la base de datos, parsea y normaliza los *logs*.

Configuración de clientes mediante Ansible

El servidor Syslog es el encargado de realizar la configuración básica para los equipos clientes. De esta manera se acelera el proceso de configuración, pudiendo configurar varios equipos simultáneamente y sin necesidad de ninguna intervención humana.

En este proceso de configuración se realizan copias y modificaciones de ficheros, instalación de software y ejecuciones de comandos en la máquina remota desde la máquina Syslog. Antes de lanzar el *playbook* de `Ansible` es necesario disponer de todos los recursos que se van a usar en el proceso. Por ello, se ha decidido utilizar el servidor

Syslog como una base de datos en la que se almacenan los ficheros de configuración y certificados propios de cada servicio.

Alguna de estas máquinas cliente cuenta con distintos usuarios con diferentes permisos cada uno. En estos casos, se necesitan permisos de superusuario para realizar las modificaciones necesarias y por ello hay que añadir las claves pertinentes en el inventario. Una vez el *playbook* comprueba que las claves son correctas, accede al equipo y realiza todos los pasos indicados.

Una vez está todo configurado se lanza el *playbook*. Durante la ejecución, se visualiza por pantalla el progreso de las diferentes tareas y si se ejecutan correctamente. Al final de la ejecución, se muestra un resumen de todas las tareas y de los posibles fallos encontrados.

En el caso de este proyecto, una vez se ha accedido al equipo cliente se procede a instalar el software necesario (`Syslog-ng` para el etiquetado y envío de *logs* y *cryptography* para la generación de certificados) y al intercambio de los ficheros de configuración antes indicados. Para finalizar, se realiza un reinicio de los servicios en los que se haya aplicado alguna modificación en su configuración, asegurando así su correcta aplicación.

Dado que estos documentos incluyen información confidencial como direcciones IP de máquinas, certificados para la comunicación o incluso usuarios y contraseñas, todos ellos están cifrados mediante `Ansible-Vault`.

3.2.4. Servidor Elastic

El servidor Elastic dispone de `Elasticsearch` para la búsqueda y análisis de *logs*, `Logstash` para la aplicación de filtros a aplicar a los *logs* recibidos y `Kibana` para la visualización de estos *logs* mediante una interfaz gráfica. Adicionalmente, se implementa `ElastAlert2` para el envío de alertas a aplicaciones externas como Telegram o LUCIA.

Parseado y enriquecimiento de *logs* mediante Logstash

`Logstash` es el encargado de recolectar y enriquecer los *logs* enviados por el servidor Syslog mediante `Filebeat`. Para ello, hay que configurar las fuentes de origen y los filtros encargados de estas funciones. Dado que `Logstash` permite tener más de una fuente de datos (cada una de ellas abre un puerto distinto para realizar la conexión), hay que asegurarse de que todas ellas estén autenticadas.

Una vez configuradas las fuentes de *logs*, se pasa a la parte de filtros para el enriquecimiento de estos. Cada fuente de *logs* utiliza una pipeline diferente, por lo que se pueden aplicar diferentes filtros a cada una. En el caso de este proyecto se va a

hacer uso de dos filtros principales (grok y geoIP) en cada fuente. GeoIP proporciona información acerca de las coordenadas GPS (latitud, longitud), ciudad, país y número de sistema autónomo de la dirección IP que ha generado el log recibido. *Grok* identifica y extrae de la línea de log el valor de cada campo, proporciona información sobre el contenido de los *logs* como el nombre de la maquina, la fecha y hora en la que se género el log, el nivel de severidad, información de la dirección IP, el mensaje del log, etc.

Al sistema Syslog de este proyecto llegan distintos tipos de *logs*, por lo que hay que realizar un filtro grok para cada una de estas variantes. Estos filtros se añaden en la sección de filtros de `Logstash`. Cuando llega un log, `Logstash` comprueba si cualquiera de los filtros coincide con su estructura. En caso positivo, se pasa por el filtro. Si ninguno de los filtros coincide, `Logstash` envía directamente a `Elasticsearch` el log en su formato original.

Una vez filtrados y enriquecidos los *logs*, estos se envían a un índice de `Elasticsearch`. Se pueden añadir diferentes salidas según los campos del log y mediante condicionales elegir una u otra, lo que permite ordenar los *logs* de una manera muy eficiente.

Integración con Elasticsearch y Kibana

`Elasticsearch` es la herramienta usada para realizar la búsqueda y monitorización de *logs*. `Elasticsearch` se basa en el uso de índices para almacenar documentos y se utilizan para organizar los datos de manera que se puedan buscar y analizar de manera eficiente. La configuración de `Elasticsearch` es la más compleja del proyecto y aunque se realiza en su gran mayoría mediante la interfaz gráfica de `Kibana` y su API REST, sigue siendo necesario realizar algunos cambios básicos en el fichero de configuración de `Elasticsearch`. Concretamente, es necesario modificar la parte de certificados que permite crear una conexión segura entre `Elasticsearch`, `Kibana` y `Logstash` mediante SSL/TLS.

Aparte de esta configuración básica, es necesario realizar varias configuraciones adicionales que permitan el funcionamiento óptimo de `Elasticsearch` para así sacarle el máximo provecho. Una de estas configuraciones es el ciclo de vida de los índices. Normalmente, el ciclo de vida consta de tres fases bien diferenciadas: la fase *hot*, *warm* y *cold* (véase la Figura 3.2).

La configuración del ciclo de vida se realiza mediante el API REST que proporciona `Elasticsearch`. Primero, se crea un fichero de texto con las características que necesita el entorno, donde se indica el comportamiento que debe tener `Elasticsearch` con los índices. En el caso de este proyecto, la política consta de dos fases. En la fase *hot*, en la que el índice entra después de su creación, la rotación del índice se realiza cuando

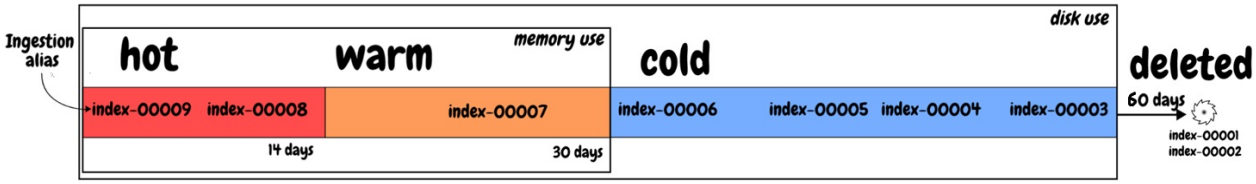

Figura 3.2: Ciclo de vida de los índices en Elasticsearch (fuente: [1])

hayan pasado 30 días desde su creación o cuando este llegue a un tamaño de 25Gb. Cuando se cumpla una de estas dos condiciones, el índice pasara a la fase *cold*, donde `Elasticsearch` le quita permisos de escritura y lo mantiene como un índice de solo lectura. Una vez pasados 30 días desde este cambio, el índice pasa a la última fase, donde se borra por completo evitando así acumular demasiados *logs* en `Elasticsearch` (fase *delete*).

Caben destacar varias cosas relacionadas con el ciclo de vida de los índices. La primera es que para que se roten, el nombre del índice debe finalizar en -00001. Así, en cada rotación se creará un nuevo índice con el número incrementado. La segunda es que `Elasticsearch` no aplica estas políticas a un índice, si no que las aplica a un alias. Esto quiere decir que cuando se genere un índice *vpn-enlace-traffic*, en realidad se estará creando el índice *vpn-enlace-traffic-00001*, asociándole el alias *vpn-enlace-traffic*. Al cambiar de fase, se crea un nuevo índice de escritura *vpn-enlace-traffic-00002* que tiene el mismo alias que el `-00001`, por lo que los nuevos *logs* llegan al mismo destino.

Tras la configuración de los índices, hay que tratar los datos que se mandarán a estos. Explicado de manera simplificada, una vez procesados y enriquecidos los logs se almacenan en `Elasticsearch` asociados a un índice. Cuando se envían los eventos a `Elasticsearch`, se indica el índice al que se deben asociar.

Una vez configurado y puesto en marcha `Elasticsearch` y `Logstash`, el siguiente paso es poner en marcha y configurar `Kibana` para acceder y visualizar los *logs* recibidos. La configuración de `Kibana` es similar a la de `Elasticsearch`. Durante la primera ejecución se crean nuevos parámetros de configuración que permiten establecer la conexión con `Elasticsearch`. Concretamente, estos parámetros son la dirección IP o nombre de dominio al que `Kibana` debe enlazarse, la IP pública a través de la cual `Kibana` estará disponible o una lista de rutas a los archivos de certificados de CA que `Kibana` utilizará para verificar la identidad de `Elasticsearch` y establecer una conexión segura a través de HTTPS.

Creación de reglas y generación de Alertas

El sistema también incluye la creación de reglas que genera alertas según el contenido de los *logs* recibidos por los agentes. Estas alertas se pueden generar a través

de `Kibana`, aunque en el caso de este proyecto se ha elegido la implementación mediante `ElastAlert2` (véase la Sección 3.1.3). Para crear una nueva regla es necesario indicar el tipo de regla que se quiere implementar, el índice o índices de `Elasticsearch` de los que se obtendrán los *logs* a analizar, la condición que se debe cumplir para ejecutar la alerta, a qué aplicaciones externas se envía la alerta generada y cada cuánto tiempo se realiza la consulta en el índice implicado. Una vez generada la alerta, esta se guarda en un índice nuevo de `Elasticsearch`, donde se añade la información indicada en la regla referente al log que ha causado la creación de esta alerta.

Una de las opciones con las que cuenta `ElastAlert2` es el uso de Python3 para la modificación de las alertas generadas. En este proyecto se va a hacer uso de esta herramienta para registrar información de usuarios (ubicación y datos personales) que accedan a la VPN de UNIZAR desde fuera de España.

Para la ejecución automatizada de las reglas se utiliza `CRON`, un administrador regular de procesos en segundo plano que ejecuta procesos en intervalos regulares.

3.2.5. Securización de las conexiones

Cabe destacar que todas estas comunicaciones están securizadas mediante el uso de certificados de cliente y servidor, de cuya creación se encarga una CA y cuyo manejo y almacenamiento se realiza en el servidor Syslog (explicado en la Sección 3.2.3).

Capítulo 4

Casos de uso

En este capítulo se describen una serie de casos de uso para ilustrar el funcionamiento del sistema desarrollado. Concretamente, se presentan tres casos de uso: la identificación y seguimiento de usuarios, la alerta de accesos sospechosos al servicio de VPN y la alerta de acceso de cuentas con privilegios.

4.1. Identificación y seguimiento de usuarios

Gracias a la flexibilidad de `Kibana`, se puede realizar un seguimiento exhaustivo de las acciones realizadas por un mismo usuario. Para estas pruebas se va a usar de ejemplo la IP 193.X.X.83 (anonimizada por cuestiones de seguridad), que ha sido vulnerada y por lo tanto se considera peligrosa. Filtrando en `Kibana` por esta IP, se consiguen los diferentes inicios de sesión que ha realizado y el resto del tráfico que ha generado con la red.

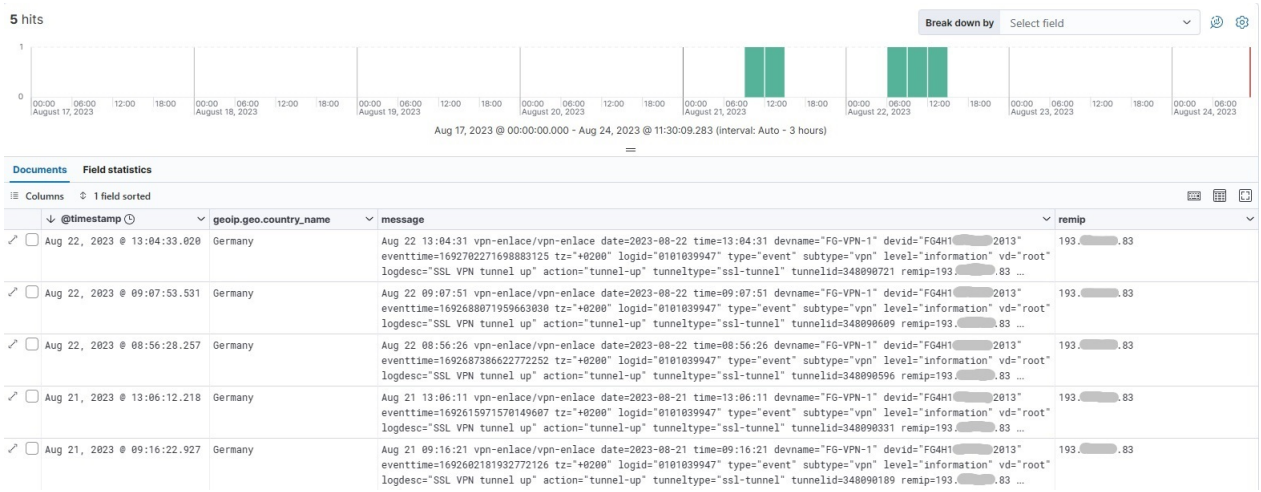

Figura 4.1: Tráfico generado por la IP 193.X.X.83

En la Figura 4.1 se muestran los logs generados por esta IP. Observando los logs se ven datos como una breve descripción del log (en este caso *SSL VPN tunnel up*, que

indica la apertura de un túnel para acceder al servicio de VPN), el lugar desde el que se ha realizado la conexión (Alemania) y el tiempo de esta (los días 21 y 22 de agosto de 2023, a diferentes horas).

4.2. Alerta de accesos sospechosos a la VPN

Otro caso de uso interesante es la detección de accesos sospechosos al servicio de VPN de UNIZAR. Todos los accesos VPN quedan registrados en el índice correspondiente de `Elasticsearch`, y son evaluados por `ElastAlert2` para detectar situaciones que puedan indicar un potencial compromiso.

Se considera que un acceso es potencialmente peligroso cuando la hora y lugar de la conexión de un usuario no se corresponde con el comportamiento habitual (se tienen en cuenta las conexiones realizadas en el último mes desde la fecha en la que se genera la conexión). Estas conexiones tienen un campo que indica su nivel de peligrosidad.

Para calcular este nivel de peligrosidad se tienen en cuenta tres campos: el lugar y la hora de conexión (la hora se mide en franjas horarias de 6 horas) y el número de incidencias. Cuando el nivel de peligrosidad es mayor que 0, `ElastAlert2` envía un mensaje a Telegram y LUCIA [19] en el que indica el ID del *log*, el lugar y la hora en la que se ha realizado la conexión y la IP del equipo final. Desde `Kibana` y filtrando con los datos obtenidos en la notificación de Telegram, se consigue información más detallada del *log*. En la Figura 4.2 se muestra un ejemplo de una alerta (con sus datos anonimizados) que ha llegado a Telegram tras activarse su desencadenante en `ElastAlert2`.

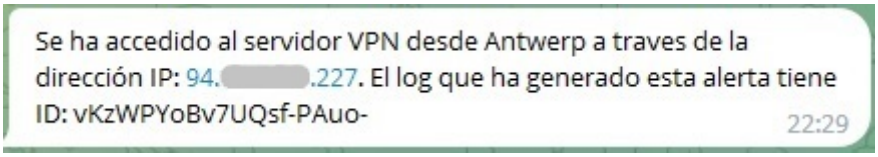

Figura 4.2: Ejemplo de alerta enviada a Telegram

Estos accesos también se ven en `Kibana`. Se ha configurado un *dashboard* en el que mediante puntos se ven las ubicaciones aproximadas de las IP que han generado las alertas.

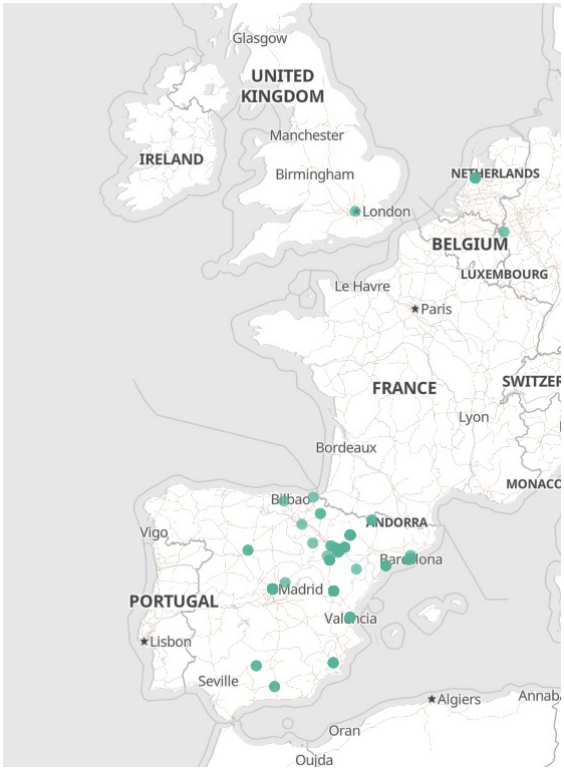

Figura 4.3: Visualización geoespacial de accesos a la VPN

Una vez recibida la notificación de Telegram, se puede filtrar el ID del *log* en `Kibana`. De esta manera se obtiene información más detallada acerca del evento que ha generado el *log*. En la Figura 4.3 se muestra una representación de las conexiones realizadas a la VPN en las últimas 24 horas. Se puede apreciar que todos los puntos son verdes, lo que indica que estas conexiones tienen su comportamiento habitual o es de las primeras veces que se conectan desde fuera de España, por lo que no se consideran un peligro.

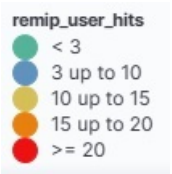

Figura 4.4: Leyenda del número de conexiones en el mapa VPN

Según el valor del nivel de peligrosidad el color del punto cambia, indicando así de una manera visual el valor de este campo. Pulsando en el punto, se muestra más detalle los datos de la alerta, como su índice de peligrosidad, la hora en la que se ha generado y los datos personales de la cuenta que ha accedido a la VPN. En la Figura 4.4 se muestra el mapa de color utilizado para los puntos.

4.3. Alerta de acceso de cuentas privilegiadas

Otro caso de uso que se ha desarrollado es la notificación a través de Telegram de la conexión a sistemas críticos (en este caso la conexión a la VPN de Administradores de Sistemas) a través de cuentas con permisos privilegiados, considerando que este tipo de accesos son especialmente sensibles.

Las cuentas con permisos privilegiados, como las de administrador, tienen acceso a toda la información y recursos del sistema. Esto las convierte en un objetivo atractivo para los atacantes, que pueden utilizarlas para robar datos, instalar código dañino o causar otros daños. Al alertar mediante `ElastAlert2` cuando una cuenta con permisos de administrador acceda a un servicio, se puede detectar rápidamente cualquier actividad sospechosa. Esto permite a los administradores del sistema responder en un tiempo razonable y tomar medidas para mitigar el riesgo.

Para determinar si un acceso se ha realizado desde una cuenta de administrador, se accede a una variable de campo obtenida a través de los *logs* recibidos en *Elasticsearch*.

Figura 4.5: Logs generados por accesos de cuentas que pertenecen al grupo UZ-ADMIN

En la Figura 4.5 se muestra un ejemplo de logs generados por accesos de cuentas con permisos privilegiados al servicio VPN de Administradores de Sistemas (con datos anonimizados). Estos logs se guardan en el índice *vpn-enlace-traffic* (donde también se guardan los accesos de cuentas no privilegiadas), por lo que se aplica un filtro de manera que únicamente se muestren los logs de accesos que pertenezcan al grupo *UZ-ADMIN* y que hayan ejecutado la acción *tunnel-up*. Así, solo se muestren los logs que se generan al realizar una conexión a la VPN de Administradores de Sistemas mediante una cuenta con permisos privilegiados.

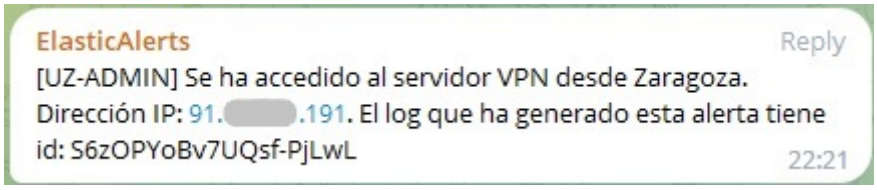

Figura 4.6: Alerta generada por un acceso de una cuenta con permisos privilegiados

En la Figura 4.6 se muestra un ejemplo de una alerta generada por un acceso a la VPN de una cuenta con permisos privilegiados y enviada a Telegram (con datos anonimizados). Filtrando en `Kibana` con el ID obtenido en Telegram, se obtiene más información acerca del evento que ha generado el *log*.

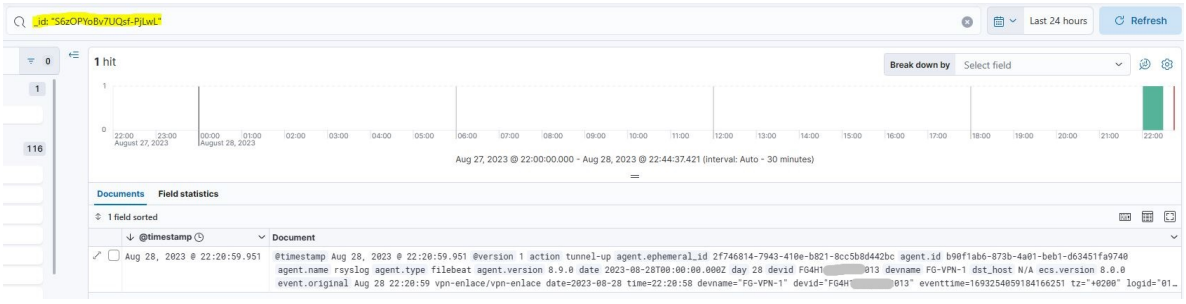

Figura 4.7: Log generado por un acceso de una cuenta con permisos privilegiados

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Un sistema centralizador de *logs* es una solución de seguridad basada en software con capacidad de monitorizar la red y los dispositivos conectados a ella. De esta manera, permiten detectar, responder y neutralizar ciberamenazas de prácticamente cualquier tipo.

En este proyecto se ha diseñado e implementado un sistema centralizador de *logs*. El sistema utiliza `Elasticsearch` como herramienta principal, así como múltiples agentes y herramientas externas para proporcionar las funcionalidades necesarias. El parseo y filtrado de *logs* se realiza mediante `Logstash`. Sin embargo, este sistema tiene limitaciones, y para arquitecturas en las que llegan múltiples tipos de *logs* se hace imposible el parseo de todos ellos. Para evitar esto se propone el uso de módulos de `Filebeat` o la implementación de inteligencia en el sistema, de manera que esta tarea se realice de forma automática. En las pruebas realizadas se han encontrado algunas limitaciones de `Logstash`. La herramienta `ElastAlert2` se usa para generar alertas cuando se cumplen ciertos criterios. El uso es sencillo, pero la programación de todas las reglas para los diferentes casos de uso lleva tiempo. Con la implementación de inteligencia, se ahorra tiempo en esta tarea, dejando en manos de `ElastAlert2` la generación de alertas según parámetros indicados y casos pasados. En este proyecto, además, se ha hecho uso de una CA interna, encargada de la generación de los certificados para la comunicación de los equipos de manera segura.

El sistema desarrollado permite detectar a través de `Kibana` diferentes accesos sospechosos y rastrear de manera más precisa el origen de estas conexiones con los datos obtenidos gracias a los filtros de Logstash.

Por último, se ha conseguido ordenar los *logs* de manera eficiente. Al momento de escritura de esta memoria, el personal de la universidad es capaz de realizar distintos filtros de manera simple y rápida para así ver en detalle conexiones de un equipo en

concreto durante un periodo de tiempo indicado. También se han conseguido almacenar los *logs* durante el tiempo estipulado por el el Real Decreto 311/2022, de 3 de mayo, cumpliendo así con los requisitos de seguridad legalmente establecidos.

5.2. Trabajo futuro

A lo largo del desarrollo del proyecto se han mencionado los posibles trabajos futuros relacionados con el tema de la gestión centralizada de *logs* concretamente:

- *Uso de OpenSearch como motor de indexación:* Es interesante tener en cuenta otros entornos y profundizar más en ellos. Por ejemplo, `Opensearch` dispone de funcionalidades interesantes, como el cifrado integrado para los datos en reposo y en tránsito, la autenticación y autorización avanzada, el control de acceso de alta precisión y la auditoría.

- *Procesado de nuevas fuentes de logs y desarrollo de nuevos casos de uso*, como podrían ser diferentes aplicaciones de los equipos actualmente configurados o equipos con diferentes sistemas operativos como macOS y servidores de bases de dato como MySQL o MongoDB.

- *La implementación de módulos adicionales* (como módulos de informes de rendimiento y análisis de causa-raíz), que permitan indagar más en los incidentes para así realizar acciones más directas y precisas.

- *Dotar de alta disponibilidad al clúster* con las herramientas de observabilidad, replicando el servidor que ofrezca dicho servicio, para permitir un funcionamiento constante en caso de fallos.

Capítulo 6

Bibliografía

[1] Optimizing elasticsearch – part 2: Index lifecycle management, 2019. https://blog.nviso.eu/author/Kris%20Boulez/.

[2] Víctor Calvo. Monitorización de servidores con tecnologías ilo (hp), idrac (dell) e ipmi con zabbix, 2020. Monitorización de servidores con tecnologías iLO, iDRAC e IPMI con Zabbix.

[3] Christopher Phillips Anton A. Chuvakin, Kevin J. Schmidt. *Logging and Log management. The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management.* Syngress, 2013.

[4] Alexander La rosa. Monitorización de logs: nada de la hermanastra fea, 2020. https://pandorafms.com/blog/es/monitorizacion-de-logs/.

[5] Syslog. https://www.paessler.com/es/it-explained/syslog.

[6] Syslog. https://es.wikipedia.org/wiki/Syslog.

[7] Rsyslog, the rocket-fast system for log processing, 2021. https://www.rsyslog.com/.

[8] Nxlog docs, introduction. https://docs.nxlog.co/userguide/intro/index.html.

[9] Sysmon v15.0, 2023. https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon.

[10] What is elasticsearch? https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html.

[11] ¿cómo funciona kibana en elasticsearch?, 2022. https://keepcoding.io/blog/funciona-kibana-elasticsearch/.

[12] Sergio García. ¿qué son los beats de elastic? + ejemplo de uso de filebeat, 2020. https://davinciti.com/beats-elastic-ejemplo-filebeat/.

[13] ¿qué son los beats de elastic? + ejemplo de uso de filebeat, 2020. https://www.elastic.co/es/logstash/.

[14] Jertel. Elastalert2, 2021. https://github.com/jertel/elastalert2.

[15] Elastalert - easy and flexible alerting with elasticsearch, 2014. https://elastalert.readthedocs.io/en/latest/.

[16] Elastalert 2 - automated rule-based alerting for elasticsearch, 2021. https://elastalert2.readthedocs.io/en/latest/elastalert.html.

[17] Winton Huang. Ansible 101 getting started. *Medium*, 2020.

[18] Why chose syslog-ng over rsyslog. https://www.syslog-ng.com/community/b/blog/posts/why-chose-syslog-ng-over-rsyslog-1421994805.

[19] Lucia, 2023. https://www.ccn-cert.cni.es/gestion-de-incidentes/lucia.html.

[20] Sancho Lerena. Logs: qué son y por qué monitorizarlos, 2023. https://pandorafms.com/blog/es/logs/#:~:text=Los%20logs%20se%20pueden%20monitorizar%20y%20establecer,reglas%20que%20permitan%20avisar%20cuando%20algo%20sucede.

Anexos A

Horas de trabajo

En la Figura A.1 se muestra el tiempo invertido en cada una de las tareas de este trabajo, explicadas a continuación. En la Figura A.2 se muestra un diagrama de Gantt ilustrando la evolución temporal.

Se comienza con la definición de la arquitectura del sistema. Durante esta etapa se definen los equipos que se van a utilizar y las herramientas a utilizar en cada uno de ellos. Posteriormente, se estudia `Syslog-ng` y se realiza la conexión de los equipos generadores de logs con el servidor Syslog. Durante esta etapa se empieza a automatizar la implementación de la configuración de los equipos recolectores de logs mediante `Ansible` (esto se realiza a lo largo de todo el proyecto). Posteriormente, se estudian los diferentes software de la pila `Elasticsearch`. Tras conocer el funcionamiento de la pila `Elasticsearch`, se procede a su implementación en el servidor Syslog y el servidor Elastic. Posteriormente se procede al desarrollo de nuevos filtros en `Logstash`. Conforme se van desarrollando estos filtros surgen nuevos casos de uso, por lo que se empieza con la implementación de alertas mediante `ElastAlert2`. Una vez está todo aplicado se empieza a visualizar y discutir los resultados. Finalmente, se procede con la elaboración de la memoria.

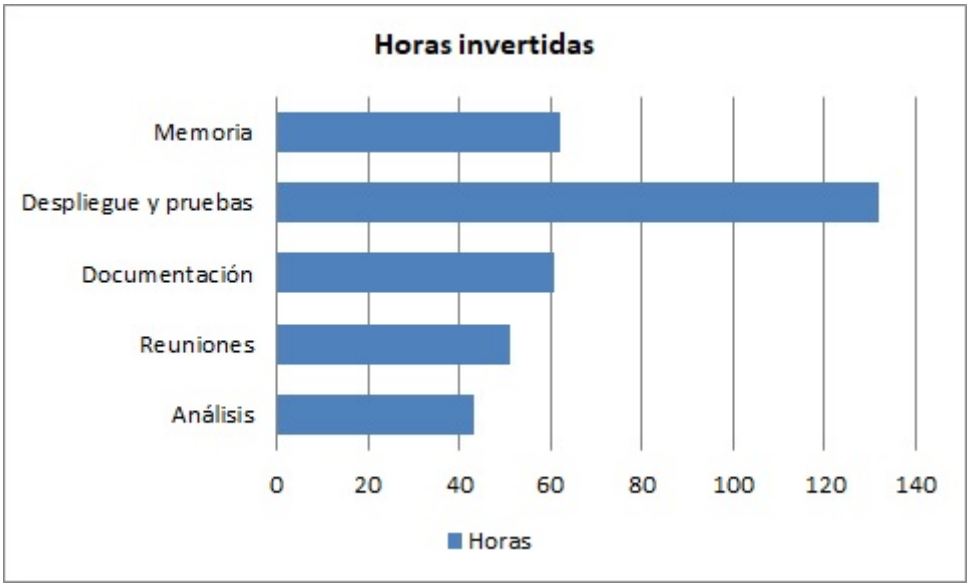

Figura A.1: Horas invertidas (total: 349 horas)

Figura A.2: Cronograma