ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Integración de una solución para la detección de intrusiones sobre tecnología de contenedores

(Integration of an intrusion detection solution in container technology)

Para acceder al Título de

Graduado en

Ingeniería de Tecnologías de Telecomunicación

Autor: Manuel García Nogales

Septiembre - 2021

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Manuel García Nogales
Director del TFG: Alberto Eloy García Gutiérrez

Título: "Integración de una solución para la detección de intrusiones sobre tecnología de contenedores"
Title: "Integration of an intrusion detection solution in container technology"

Presentado a examen el día: 29/09/2021

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Roberto Sanz Gil
Secretario (Apellidos, Nombre): Alberto Eloy García Gutiérrez
Vocal (Apellidos, Nombre): Mª Angeles Quintela Incera

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente Fdo.: El Secretario

Fdo.: El Vocal Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector Trabajo Fin de Grado Nº
(a asignar por Secretaría)

Resumen

El objetivo principal de este trabajo es implementar una solución mediante herramientas de código abierto en un entorno basado en contenedores que permita monitorizar el tráfico de red no deseado, anomalías de seguridad y detección de intrusiones. Para ello, se procesarán, recopilaran y se presentará la información relevante mediante una interfaz gráfica fácilmente comprensible por cualquier usuario con acceso.

Con este fin, se utilizará Snort como software de detección de intrusiones y Elk stack, que será el encargado de recopilar los datos enviados por Snort, para analizarlos y visualizarlos en tiempo real. Ambas herramientas funcionarán sobre contenedores Docker para así aprovechar las numerosas ventajas que esta tecnología ofrece.

Este proyecto podrá ser implementado en el ámbito educacional y en redes de pequeñas empresas para añadir un nivel más de protección y un control de los eventos que estén ocurriendo en la red tanto en tiempo real como a posteriori.

Palabras clave

Snort, ELK, Docker, Contenedores, vulnerabilidades.

Abstract

The main subjet of this project is to deploy a solution based on open source tools in a Docker container environment to allow monitoring of unwanted network traffic, security anomalies and intrusion detection. To achieve this, the relevant information will be processed, collected and presented through a graphical interface easily understandable by any user authorized to access this data.

To achieve this, Snort will be used as intrusion detection software and Elk stack will be using to collecting the data sended by Snort, and also to analyze and visualize it in real time. Both tools will work on Docker containers to take advantage of the many benefis that this technology offers to the user.

This project could be implemented in the educational field and in small business networks to increse an additional level of protection and control of the events that are occurring on the network both in real time and afterwards.

Key words

Snort, ELK, Docker, containers, vulnerabilities.

Índice de contenidos

Índice de Figuras

Acrónimos

CPU: Central Processing Unit

DDoS: Distributed Denial Of Service

DNS: Domain Name Service

ELK: Elasticsearch Logstash Kibana

GPG: GNU Privacy Guard

HTTP: Hypertext Transfer Protocol

ICMP: Internet Control Message Protocol

IDS: Intrusion Detection System

IP: Internet Protocol

IPS: Intrusion Prevention System

IPv4: Internet Protocol version 4

NIDS: Network Intrusion Detection System

OS: Operating System

RAM: Random Access Memory

SSD: Solid State Drive

TCP: Transmission Control Protocol

TFG: Trabajo de Fin de Grado

UDP: User Datagram Protocol

VM: Virtual Machine

1. Introducción

Como tema general, el trabajo se centrará en implementar un demostrador de un sistema de gestión de logs de seguridad y eventos de red, para un posible futuro montaje en el ámbito educacional y con la idea de ser utilizarlo en prácticas de laboratorio.

Los sistemas de seguridad, como firewalls o IDS, generan mucha información, en forma de logs, que posteriormente debe ser tratada para poder tomar decisiones. En numerosas ocasiones, la gestión de estos datos consiste en sacar ventanas con gráficos o tablas que hagan toda esta información atractiva, comprensible y pueda ser intuitivamente empleada por cualquier usuario sin demasiados conocimientos en la materia.

1.1 Organización del documento

Después del capítulo de introducción, este documento se organiza en otros seis capítulos adicionales que exponen el contenido de la siguiente forma:

En primer lugar, se listarán los aspectos relativos al alcance de este proyecto y los requisitos que deberá cumplir el sistema.

El tercer capítulo explica los aspectos teóricos relacionados con las herramientas y elementos que se han utilizado durante la ejecución del proyecto, clasificándolos y exponiendo el porqué de su elección para formar parte del proyecto.

En el siguiente capítulo, se analizarán todos los aspectos prácticos, desde la instalación de Docker a la instalación y configuraciones de los contenedores que se utilizarán en el funcionamiento del sistema. Además, también se analizarán las herramientas que funcionan dentro de cada contenedor.

En el quinto capítulo se realizará una descripción de la aplicación propuesta mostrando su funcionamiento y las capacidades del sistema en general para que el lector pueda comprender el alcance y las posibilidades que ofrece desde una visión práctica.

El penúltimo apartado se centra en la exposición de las conclusiones obtenidas en la realización del proyecto, además, se proponen otras líneas para intentar mejorar el sistema en el futuro.

Para finalizar se enumeran las fuentes bibliográficas consultadas a lo largo de la realización del TFG.

2. Motivación, objetivos y requisitos iniciales

Internet ha transformado la vida de las personas de manera positiva. Aun así, esta red de redes y sus tecnologías asociadas también han traído un creciente número de amenazas a la seguridad. Por otro lado, la implementación de aplicaciones en contenedores frente a máquinas virtuales está ganando la partida en el mundo de las tecnologías de la información. Esta tecnología está teniendo uno de los crecimientos más grandes en la industria de software de los últimos años y ya son numerosas grandes empresas las que han implantado esta tecnología.

El objetivo de este trabajo es el de emplear la tecnología de contenerización en Docker para hacer frente a estos problemas de seguridad. Para ello, se han empleado varias herramientas corriendo en contenedores que pueden ayudar a solventar los problemas que surgen habitualmente en un entorno educacional o empresarial de una manera más sencilla, escalable, segura y eficiente.

Los requisitos iniciales que debe de cumplir el sistema son los siguientes:

- Decidir el tipo de anomalías a monitorizar en la red.
- Implementar las normas y generar las anomalías.
- Enviar los eventos generados en la herramienta generadora.
- Almacenar la información generada.
- Visualizar y presentar los datos para su análisis.

A continuación, se describen los aspectos que forman parte de los objetivos este proyecto:

- Descripción de la herramienta para la gestionar eventos de una red.
- Análisis de la herramienta de recopilación y visualización de datos.
- Implementar las herramientas en Docker y desplegar los contenedores.
- Explicación de la solución propuesta en base a los requisitos iniciales.
- Resultados obtenidos en el trabajo.

3. Aspectos teóricos

A continuación, se hace una breve introducción de los principales términos, conceptos y principios teóricos relacionados con las herramientas que se implementarán en este trabajo.

3.1 IDS

Un sistema de detección de intrusiones o Intrusion Detection System (IDS), es un sistema usado para detectar accesos sin autorización a un equipo o a una red, es decir, es un software que monitoriza el tráfico de entrada y lo cruza con una base de datos que está actualizada con firmas de ataque que ya son conocidas.

Ante cualquier actividad sospechosa, emiten una alerta o log a los administradores del sistema, que serán los responsables de tomar medidas determinadas. Estos accesos pueden ser ataques esporádicos realizados por usuarios con malas intenciones o repetidos en un periodo de tiempo concreto, lanzados con herramientas automáticas. Estos sistemas sólo detectan los accesos sospechosos emitiendo alertas anticipatorias de posibles intrusiones, pero no tratan de detener la propia intrusión, es decir, se utilizan de una manera reactiva.

Una de las principales ventajas de un IDS es que proporciona la posibilidad ver lo que está pasando en la red en tiempo real en base a unos datos previamente recopilados, puede reconocer modificaciones y automatizar los patrones de búsqueda en los paquetes de datos que han sido enviados a través de una red. Su principal desventaja es, sobre todo, en el caso de las de tipo pasivo, que no es están pensadas para detener o prever los ataques que detecten. Además, son vulnerables a los ataques de tipo DDoS (Distributed Denial of Service) que podrían suponer la inoperatividad de la propia herramienta. [1]

3.1.1 Snort

Para la realización del proyecto se ha decidido utilizar Snort como IDS de referencia. Snort es el sistema de prevención de intrusiones (IPS) de código abierto más utilizada en el mundo. Utiliza una serie de reglas que ayudan a

definir la actividad de red maliciosa y las usa para encontrar paquetes que coincidan con ellas y generar alertas para los usuarios. [2]

Snort, además, es un sistema de detección de intrusiones en red (NIDS) basado en la detección de usos no permitidos en la red, analizando tráfico y capturando paquetes en tiempo real. Además, puede analizar protocolos, buscar contenidos y puede usarse para detectar multitud de ataques, por ejemplo: desbordamientos de búfer, escaneo de puertos e intentos de OS Fingerprinting (recopilación de datos para obtener información del sistema operativo u otro software), entre muchos otros. [3]

Puede ser usado en Linux o en Windows, al ser multiplataforma, además de ser gratuito y versátil, razón por la que es uno de los IDS más utilizados. Todo esto hace que exista bastante información disponible, tanto en sitios oficiales, como en foros o blogs de usuarios.

Snort puede actuar de tres modos diferenciados:

- **Sniffer**: que monitoriza en tiempo real toda la actividad de las redes que se configuran en Snort.
- **Packet logger**: almacena en unos logs o registros, la actividad de la red que se ha configurado para, posteriormente, poder realizar un análisis.
- **NIDS**: Además de monitorizar la red, se configura a snort para generar alarmas a partir del tráfico recibido utilizando firmas. [4]

El objetivo de este trabajo es utilizar este sistema como detector de logs de seguridad y eventos de red, por lo que, de los tres modos posibles, se ha seleccionado el modo NIDS. La elección viene marcada por la existencia de una gran comunidad de usuarios y desarrolladores, que facilitan la configuración y su rápida instalación.

El funcionamiento de SNORT se basa en comparar un flujo de datos y las reglas que se tienen previamente definidas y almacenadas. Estas reglas permiten identificar elementos maliciosos mediante un lenguaje flexible, aunque no es una de las prioridades de este trabajo, y solo se utilizará un caso típico de detección, como ejemplo de demostración de su funcionamiento y posterior integración del resto de elementos del sistema propuesto.

3.2 Recopilación y visualización de eventos

Algunos de los inconvenientes de utilizar la recopilación de información de eventos a través de logs, como así hace Snort, son la necesidad de almacenar una gran cantidad de datos, el incremento en el tiempo de respuesta o que el sistema sea realmente escalable. No es de extrañar que, en la mayoría de los casos, se requiera de herramientas especializadas que permitan automatizar el análisis del conjunto de logs recopilados. Una de estas herramientas es precisamente ELK, la cual, básicamente, permite filtrar, buscar y analizar datos de una forma fácil e intuitiva y manejar gran cantidad de información de forma totalmente escalable.

3.2.1 ELK Stack

ELK Stack o Elastic Stack es un conjunto de herramientas open source de gran potencial, en tre las que destacan Logstash, ElasticSearch y Kibana, así como una serie de agentes denomindos Beats. Estas herramientas se unifican para crear un potente sistema de gestión de registros permitiendo la monitorización y análisis de los logs enviados desde cualquier sistema o servidor.

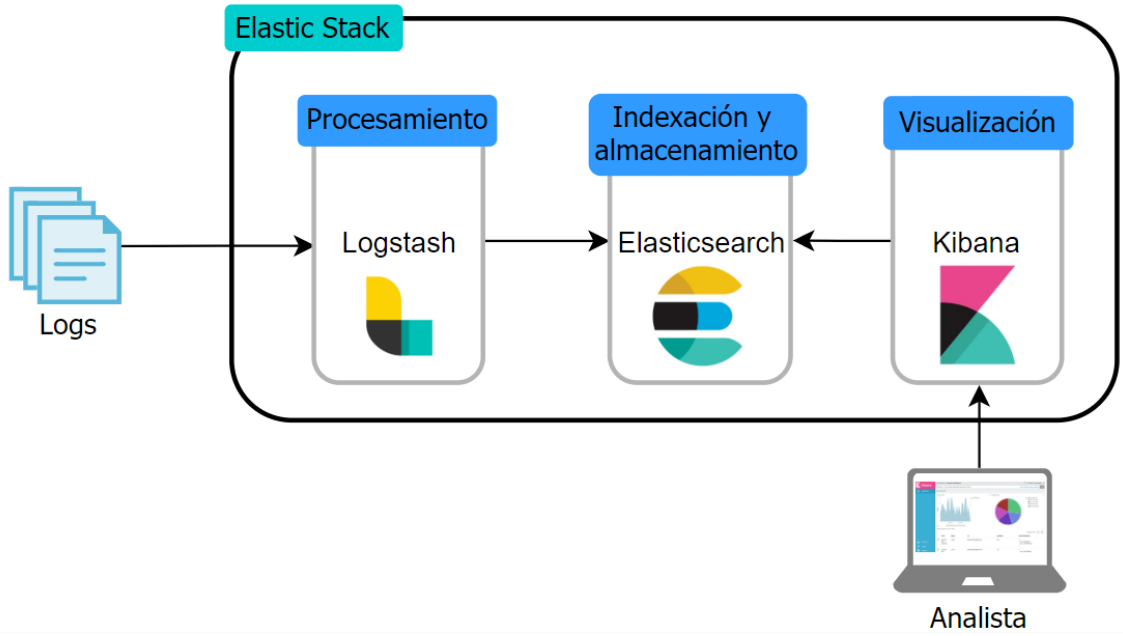

Figura 1. Arquitectura de Elastic Stack

Tal como se muestra en la Fig. 1, la fuente generadora de datos envía toda la información al elemento Logstash, que incluye varios plugins de entrada y salida, y permite inyectar enorme cantidad de información al resto de elementos. Esta información es procesada y a continuación almacenada en la base de datos de Elasticsearch, para finalmente, con Kibana, ejecutar las visualizaciones para acceder a datos concretos y así presentarlos y poder monitorizarlos.

3.2.1.1 Elasticsearch

Fundamentalmente, Elasticsearch es un motor de búsqueda en un entorno NoSQL, pero sin perder su naturaleza de base de datos distribuida. Elasticsearch es compatible con grupos de datos generados desde multitud de tipos de fuentes diferentes, inyectando todos esos datos en los diferentes nodos que componen su cluster de almacenaje, de ahí que permita una alta disponibilidad y una mayor tolerancia a fallos.

De la misma forma que distribuye la información entre los nodos, también distribuye el procesamiento de los datos. Cuando se realiza una búsqueda o una consulta y los datos se encuentran distribuidos, serán los nodos los encargados de procesar la información y devolver los resultados, mejorando así el rendimiento.

Este elemento pone el foco en realizar un análisis previo de los datos, para luego, adecuar su almacenaje para permitir realizar análisis de eventos sobre incidentes de seguridad de una manera fácil y prácticamente en tiempo real. Un analista de seguridad podrá efectuar así, consultas sobre los eventos y logs en base a la información que le interese, por ejemplo, fuente, protocolo, longitud de paquetes, direcciones IP, etc.

3.2.1.2 Logstash

Es la herramienta encargada de realizar la recolección de eventos en tiempo real, análisis de formato y selección del tipo de almacenamiento, justo antes de su procesamiento por parte de Elasticsearch.

Logstash actúa como un agregador: extrae datos de varias fuentes antes de enviarlos, generalmente a Elasticsearch, y aunque no es el único sistema compatible, en las soluciones para IDS suele ser necesario enriquecer/mejorar los registros, por ejemplo, agregando contexto a los mensajes originales, analizando los campos por separado, o filtrando bits de

datos no deseados, y Logstash es el mejor. Sin embargo, esto también añade bastante complejidad al usuario, y es por ello que el propio ELK también permita utilizar otros agregadores más simples, como es el caso de Filebeat. Precisamente, al pretender montar una demostración sencilla de integración de los elementos, más que obtener una solución IDS robusta, en este trabajo se ha decido utilizar Filebeat para enviar los registros desde la fuente hasta Elasticsearch.

3.2.1.3 Filebeat

Filebeat, y los otros miembros de la familia Beats, actúan como un agente ligero implementado en el host que envía los registros a Elasticsearch.

Filebeat es uno de los mejores remitentes de archivos de registro que existen en la actualidad: es liviano, admite el cifrado SSL y TLS, utiliza un buen mecanismo de recuperación incorporado y es altamente confiable. [5]

En nuestro caso es la solución que se ha seleccionado para el envío de logs, al ser liviano y robusto y a que el sistema será implementado en entornos pequeños o medianos.

3.2.1.4 Kibana

Es una interfaz de usuario que permite la visualización de datos para su análisis posterior. En Kibana se puede consultar o leer información desde los índices de Elasticsearch para realizar búsquedas, visualizar y analizar los datos, además de navegar en ELK.

Pone a disposición del usuario diversos tipos de herramientas para visualizar los datos, tales como diagramas, histogramas, tablas, etc. Entre las principales funciones de esta herramienta están:

- Análisis de datos mediante la aplicación de diferentes métricas.
- Exposición de datos a través de diversos tipos de gráficas o diagramas
- Comprobación del estado de ELK
- Generación de reportes
- Exploración y descubrimiento de datos
- Creación de alertas
- Implementación de TLS como cifrado de comunicaciones.

Añadiendo esta capa visual, el conjunto de servicios que conforman ELK hace que sea una opción robusta para desplegar una solución escalable para la detección de intrusiones de seguridad. [6]

3.3 Contenerización con Docker

Docker es una plataforma open source para desarrollar, enviar y ejecutar aplicaciones. Docker permite separar las aplicaciones del resto de la infraestructura para que se pueda ejecutar software más rápidamente. Al aprovechar las metodologías de Docker para enviar, probar e implementar código, se puede reducir significativamente el tiempo entre la escritura del código y su ejecución.

Docker tiene la capacidad de empaquetar y ejecutar una aplicación en un entorno aislado llamado contenedor. La seguridad y sobre todo el aislamiento le permiten ejecutar varios contenedores simultáneamente en un determinado host. Los contenedores contienen todo lo necesario para ejecutar una aplicación y son ligeros, por lo que no dependen de lo que tenga instalado el sistema en el que se están ejecutando. Se pueden compartir contenedores fácilmente mientras trabaja y asegurarse de que los usuarios con las que se comparten obtienen el mismo contenedor que funciona de la misma manera.

Cuando se utiliza Docker, se están creando y usando contenedores, imágenes, redes, complementos y otros elementos. Los siguientes apartados describen de forma general algunos de esos elementos utilizados para la realización de este trabajo. [7]

3.3.1 Imágenes

Una imagen es una plantilla de solo lectura con instrucciones para crear un contenedor Docker. En multitud de ocasiones, una imagen está basada en otra imagen, con alguna personalización adicional. Por ejemplo, puede crear una imagen basada en Snort, pero instalar el servicio de Beats Filebeat y, por ejemplo, Npcap, así como los detalles de configuración necesarios para ejecutar junto con Snort, y que estas herramientas funcionen de manera conjunta, rápida y de manera encapsulada al resto del sistema.

Puede crear sus propias imágenes o puede usar solo las creadas por otros, y publicadas en DockerHub. Para construir su propia imagen, se puede crear

un Dockerfile con una sintaxis simple que defina los pasos necesarios para crear la imagen y ejecutarla. Cada instrucción en un Dockerfile crea una capa en la imagen. Cuando cambia el Dockerfile y reconstruye la imagen, solo se reconstruyen las capas que han cambiado. Esto es parte de lo que hace que las imágenes sean tan ligeras, pequeñas y rápidas en comparación con otras tecnologías de virtualización.[7]

3.3.2 Contenedores

Un contenedor es una instancia ejecutable de una imagen. Puede crear, iniciar, detener, mover o eliminar un contenedor mediante la API o la CLI de Docker. Puede conectar un contenedor a una o más redes, adjuntarle almacenamiento o incluso crear una nueva imagen basada en su estado actual. [7]

De forma predeterminada, un contenedor está aislado de otros contenedores y del sistema. Puede controlar cómo de aislados están los contenedores por medio de la red de Docker, el almacenamiento, incluso otros sistemas dependientes de un contenedor, de otros contenedores o de la máquina host. En la *Figura 2* se muestra la arquitectura de un sistema donde se implementan varios contenedores.

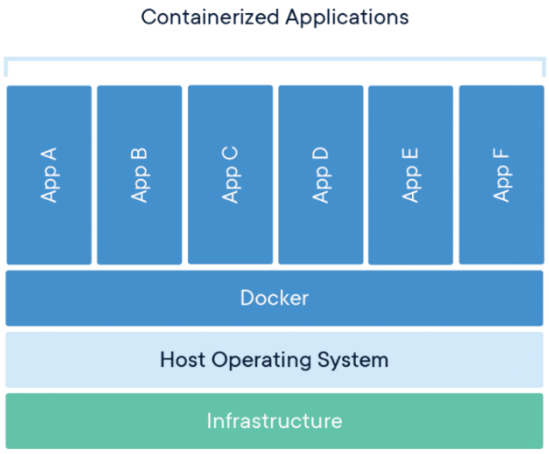

Figura 2. Contenedores Docker

Se pueden ejecutar varios contenedores en el mismo host y que estos compartan recursos y el kernel del sistema operativo, cada uno estará ejecutándose como un proceso aislado. Los contenedores ocupan menos

espacio que las VM (estamos hablando de varios Gigabytes contra unas decenas de Megabytes), se requieren menos sistemas operativos y se pueden manejar más aplicaciones requiriendo menos memoria.

Un Docker está definido por una imagen, así como por las opciones de configuración que se le proporcione al iniciarlo o al crearlo. Una vez se sale de un contenedor, cualquier cambio que no haya sido guardado en la memoria estática se borrará.

Si no se dispone de la imagen, Docker la obtiene de su registro configurado, como si se hubiese ejecutado `docker pull` de manera manual.

A continuación, se crea un nuevo contenedor, como si hubiera ejecutado el comando `docker container create` manualmente.

Docker asigna un sistema de archivos de escritura y lectura al contenedor. Esto permite que un contenedor que se está ejecutando cambie o cree directorios y archivos en su sistema de archivos local, permaneciendo aislado del host.

Además, se crea una interfaz de red para conectar el contenedor a la red, si no se indica ninguna opción de red se conectará a la predeterminada. Asigna al contenedor una dirección IP. Además, los contenedores se pueden conectar a redes externas mediante la propia conexión de red de la máquina host, este tema se tratará más adelante.

3.3.3 Network

Una de las razones por las que Docker es tan potente es que puede conectar entre sí contenedores o conectarlos a otro software que no esté implementado en Docker. Los contenedores y servicios de Docker no necesitan ni siquiera saber que están implementados en Docker. Es más, puede que los contenedores se estén ejecutando en Linux, Windows o una combinación de ambos, se puede usar Docker para administrarlos de una manera independiente de la plataforma donde estén implementados.

El sistema de red de Docker se puede conectar mediante controladores. Existen varios controladores de forma predeterminada y proporcionan las siguientes funcionalidades:

- **Bridge**: Es el controlador de red predeterminado. Si no se especifica un controlador, este será el tipo de red que se estará utilizando. Las redes puente se utilizan generalmente cuando sus aplicaciones se ejecutan en contenedores independientes que necesitan comunicarse. Esta es la mejor opción cuando se necesitan varios contenedores conectados corriendo en el mismo host.

- **Host**: Se utiliza para contenedores independientes, se elimina el aislamiento de red entre el contenedor y la máquina de host, es decir, utiliza la red del host directamente. Es recomendable cuando, por ejemplo, se quiere usar las interfaces del propio host.

- **None**: Utilizada en un contenedor, desactiva todas las redes. Usualmente se usa junto con un controlador de red personalizado. En resumen, se utiliza para indicar que un contenedor no tiene asignada ninguna red. [8]

3.3.4 Modificación de contenedores

Cuando se trabaja con imágenes y contenedores, una de las características básicas es realizar cambios en una imagen de Docker. Cuando se finaliza con los cambios, básicamente se puede crear una nueva imagen con una capa adicional que modifica la capa de la imagen base. [9]

Una vez finalizado el proceso y con la imagen recién creada, debería estar disponible en la lista de imágenes locales. Si se iniciase un contendor de esa imagen ya estarían implementados todos los cambios realizados.

4. Aspectos prácticos

A continuación, se hace referencia a los detalles de instalación y configuración de los sistemas utilizados durante el desarrollo de este trabajo.

4.1 Entorno e instalación

Sobre el host se ha desplegado una red sobre un entorno virtual Oracle VM (VirtualBox) en el que se ha instalado un servidor Ubuntu, en la Figura 3 se muestra un esquema del sistema y sus respectivas IPs.

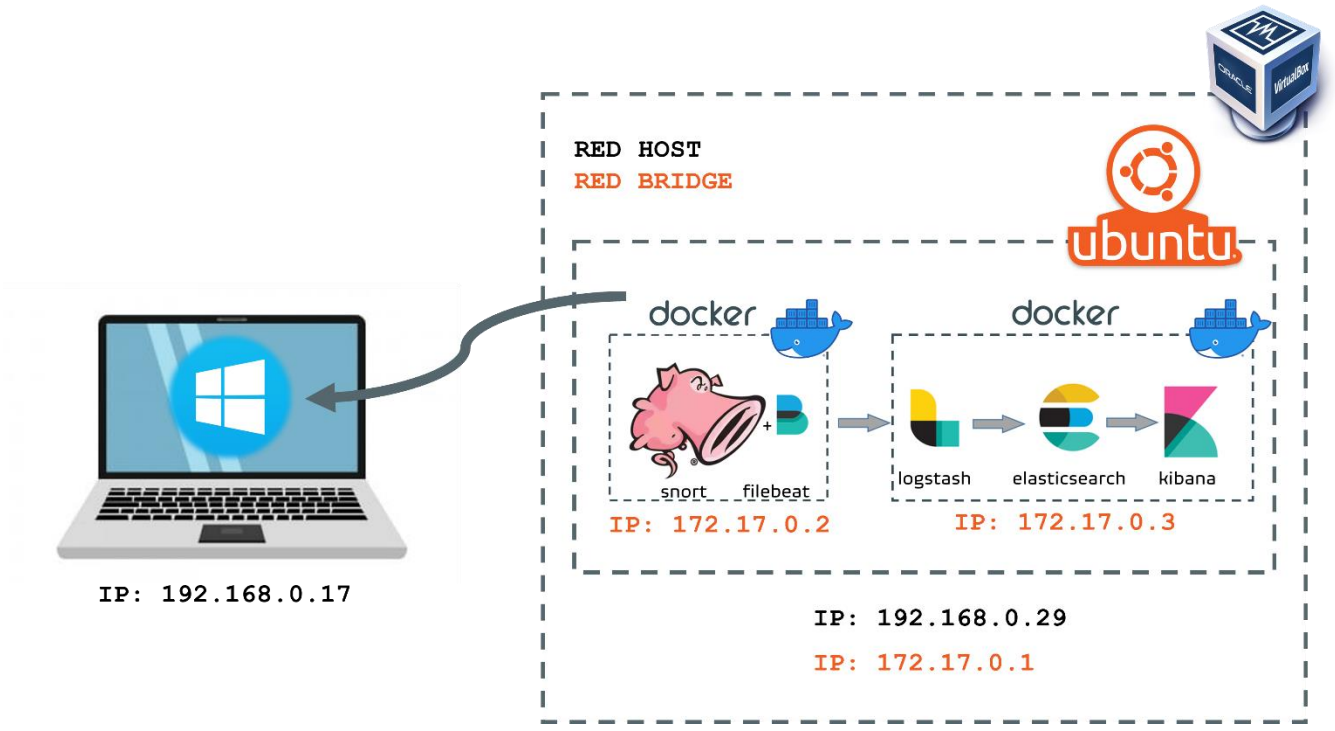

Figura 3. Esquema práctico del sistema.

El equipo host en el que correrá el entorno virtualizado es el siguiente:

- Sistema operativo: Windows 10 Home (64-bit)
- Hardware: Procesador Intel(R) Core i7-7500U CPU @ 2.70GHz 2.90 GHz, 16 GB de RAM y disco SSD 500 GB.

Las características del equipo Ubuntu son las siguientes:

- Sistema operativo: Ubuntu 20.04.02 (64-bit)
- Hardware: 10 GB de RAM, 2x CPU y disco 50 GB.
- Red: Adaptador puente, Intel Dual Band Wireless-AC 8260

4.2 Docker

Una vez dentro de la MV de Ubuntu e identificado como root, se va a proceder a la instalación de Docker. Se recomiendan unos 4GB de RAM asignados a Docker para su correcto funcionamiento.

Para comenzar, hay que actualizar la lista de paquetes:

```
$ sudo apt update
```

Ahora, se deberá instalar los paquetes de requisitos previos que permitan a apt usar paquetes a través de https:

```
$ sudo apt install apt-transport-https ca-certificates curl
software-properties-common
```

A continuación, se añade la clave de GPG:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo apt-key add -
```

Luego, se añade el repositorio de Docker a las fuentes de apt:

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"
```

Una vez realizados estos pasos previos se puede instalar Docker:

```
$ sudo apt install docker-ce
```

Con todo esto se podría dar por finalizada la instalación de Docker. Se puede comprobar que todo funciona según lo esperado con el siguiente comando:

```
$ sudo systemctl status docker
```

En la Figura 4, se observa el estado de Docker después del comando anterior:

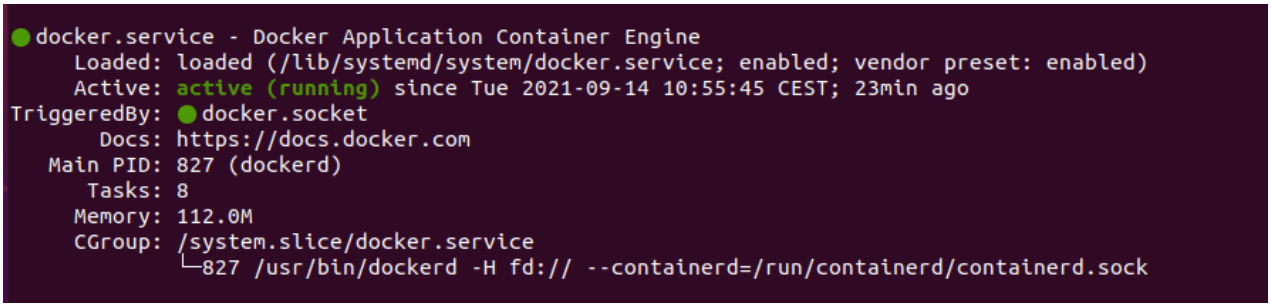

Figura 4. Estado de Docker

A continuación, se mostrará las opciones del comando `docker`, la estructura es la siguiente:

```
$ docker [opciones] [comando] [argumentos]
```

Para ver los comandos disponibles se escribe el comando `docker`.

En la Figura 5 se muestran la lista de comandos disponibles, correspondientes a contenedores e imágenes:

```
Management Commands:
  app*        Docker App (Docker Inc., v0.9.1-beta3)
  builder     Manage builds
  buildx*     Build with BuildKit (Docker Inc., v0.6.1-docker)
  config      Manage Docker configs
  container   Manage containers
  context     Manage contexts
  image       Manage images
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  scan*       Docker Scan (Docker Inc., v0.8.0)
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit codes
```

Figura 5. Comandos en Docker

Los contenedores de Docker se construyen sobre imágenes, que se obtienen de Docker Hub, para descargar cualquier imagen alojada en Docker Hub solo se necesitará su nombre oficial de la imagen utilizando la siguiente secuencia:

```
$ docker pull [nombre_imagen]
```

Una vez descargada la imagen se podría ejecutar utilizando el comando `run`.

Para ver las imágenes que se han descargado:

```
$ docker images
```

El resultado, dependiendo de las imágenes descargadas mostrará algo similar a lo mostrado en la Figura 6.

```
root@ubuntuDocker:/home/lolo# docker images
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
snort-filebeat          latest    f20daea7191b   13 days ago      746MB
lolo/snort-filebeat     latest    080cf7a12268   3 weeks ago      746MB
sebp/elk                latest    d5e175b50ad8   3 months ago     2.6GB
ciscotalos/snort3       latest    2c50bee6393c   19 months ago    774MB
linton/docker-snort     latest    b479f7049241   5 years ago      639MB
```

Figura 6. Listado de imagenes Docker

El siguiente comando ejecuta un contenedor cualquiera, se adjunta de forma interactiva a su sesión de línea de comandos local y se ejecuta /bin/bash.

```
docker run -i -t [contenedor] /bin/bash
```

A continuación, se iniciará el contenedor y se ejecutará en /bin/bash, debido a que el contenedor se ejecuta de manera interactiva y está ejecutándose en la consola (a causa de los parámetros -i y -t), se pueden introducir entradas usando el teclado mientras la salida es registrada en la consola.

Si se requiere salir se debe escribir el comando `exit`, el contenedor se detiene, pero no será eliminado. Se iniciará de nuevo siguiendo los pasos

anteriores o eliminarlo utilizando el comando `rm <id_contenedor>`, este comando borrará el contenedor, pero la imagen seguirá estando disponible.

Para visualizar los contenedores que hay corriendo en la máquina, tanto los activos como los inactivos se utilizara la siguiente secuencia:

```
$ docker ps -a
```

Se visualizará algo similar a la Figura 6:

```
root@ubuntuDocker:/home/lolo# docker ps -a
CONTAINER ID   IMAGE            COMMAND                  CREATED       STATUS
    NAMES
68258c182d02   sebp/elk         "/usr/local/bin/star…"   11 days ago   Exited (137) 11 days ago
    elk
239273f4af50   snort-filebeat   "/bin/bash"              11 days ago   Exited (0) 11 days ago
```

Figura 7. Lista de contenedores activos

Para iniciar un contenedor detenido se puede utilizar el siguiente comando:

```
$ docker start [ID_Contenedor]
```

De la misma forma para detener el contenedor: [10]

```
$ docker stop [ID_Contenedor]
```

Para modificar un contenedor, se comenzará con el Docker que se quiera modificar, una vez dentro del contenedor, instalar las herramientas o cambiar los archivos de configuración que se quiera que queden guardados, una vez terminada la modificación del contenedor se saldrá de el con el comando `exit`.

Para finalizar se creará una nueva imagen confirmando los cambios utilizando la siguiente sintaxis:

```
$ docker commit [CONTENEDOR_ID] [nuevo_nombre_imagen]
```

4.2.1.1 Snort

Como base de Snort se ha seleccionado un Docker de Snort con la versión 2.9.8.0 y con una versión de DAQ 2.0.6. Disponible en Docker Hub. [11]

Para descargar la imagen:

```
$ docker pull linton/docker-snort
```

Para iniciar el contenedor:

```
$ docker run -it --rm linton/docker-snort /bin/bash
```

El parámetro `-it` permite mantener el contenedor en ejecución y a su vez poder interactuar con él lanzando comandos dentro del contenedor. Utilizando el parámetro `--rm` indicando a Docker que, al terminar la ejecución, el contenedor sea eliminado, esto evita tener almacenada la ejecución del contenedor.

Una vez dentro del contenedor de snort, se va a proceder a la instalación de Filebeat: [12]

```
$ wget
https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.6.2-amd64.deb
$ dpkg -i filebeat-7.6.2-amd64.deb
```

A continuación, se habilitará el módulo System de Filebeat:

```
$ filebeat modules enable system
```

Para finalizar, se saldrá del contenedor y se realizará un `commit` para que la próxima vez que se inicie el contenedor ya tenga Filebeat instalado. Los parámetros necesarios son la ID de la imagen del contenedor y el nombre de la nueva imagen, en este caso `snort-filebeat`.

```
$ docker commit b479f7049241 snort-filebeat
```

4.2.1.2 ELK

Para la utilización del ELK se ha seleccionado un docker disponible en Docker Hub, que utiliza la versión 7.13.2 de ELK.[13]

Este contenedor proporciona un servidor y gestión de registros junto con su interfaz gráfica convenientemente centralizado, empaquetando: Elasticsearch, Logstash y Kibana, conocidos como ELK. Necesita al menos 2GB de RAM para ejecutarse.

Desde la versión 5, Elasticsearch no se inicia correctamente debido aun numero bajo de limite en el recuento de nmap cuando es inferior a 262144. En Linux se puede utilizar `sysctl vm.max_map_count` para ver el valor actual, cuya configuración debe realizarse en el host, no dentro de Docker.

Para establecer el valor de forma permanente, se debe actualizar el valor, añadiendo la siguiente línea de código: [14]

```
vm. max_map_count = 262144
```

En el archivo `/etc/sysctl.conf`

Para verificar tras reiniciar se ejecutará:

```
sysctl vm.max_map_count
```

En la Figura 8 se muestra la comprobación:

```
root@ubuntuDocker:/home/lolo# sysctl vm.max_map_count
vm.max_map_count = 262144
root@ubuntuDocker:/home/lolo#
```

Figura 8. Límites del recuento de mmap en el sistema operativo

Para descargar la imagen:

```
$ docker pull sebp/elk
```

Este Docker no necesita ninguna configuración adicional para su ejecución.

4.2.2 Network

En este proyecto se utilizarán los contenedores conectados a dos tipos de redes. En la Figura 9 se muestran las redes disponibles por defecto en Docker.

```
root@ubuntuDocker:/home/lolo# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
840f9d5484bc   bridge    bridge    local
ed45a4fcc4bc   host      host      local
f824ce86912b   none      null      local
```

Figura 9. Redes por defecto en Docker

Para cubrir todas las opciones que se plantean, los contenedores se conectaran a las redes preconfiguradas Host y Bridge.

4.2.2.1 Host

Como ya se comentó en el Capítulo 3.3.3 se utilizará la red Host cuando se necesite utilizar las interfaces del propio equipo. Esto es necesario si se requiere detectar intrusiones o anomalías en la red donde Docker está instalado. En la Figura 10 se muestran las características de la red Host.

```
root@ubuntuDocker:/home/lolo# docker network inspect host
[
    {
        "Name": "host",
        "Id": "ed45a4fcc4bc24c2081dad78ad7e8daf3cd37a426b93adb679c6dc7f1e2bbbdd",
        "Created": "2021-07-02T10:52:39.755894742+02:00",
        "Scope": "local",
        "Driver": "host",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": []
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

Figura 10. Características de la red Host.

4.2.2.2 Bridge

La red bridge es la red standard que usarán los contenedores por defecto. Son las recomendadas si se necesitan varios contenedores comunicándose en el host de Docker, y a su vez estar aislados de la red en el mismo host. La principal desventaja de este tipo de red es que no permite monitorizar las interfaces del anfitrión.

Este tipo de red será la seleccionada si se quiere monitorizar la red donde estén corriendo otros Docker y ver, por ejemplo, si alguien está intentando acceder a ella, o está intentando atacarla por cualquier motivo.

La Figura 11 muestra las características de la red Bridge.

```
root@ubuntuDocker:/home/lolo# docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "840f9d5484bc2f44312fdccc4df774ee38dddcdbcacccfe4be7b3a9caaed7055",
        "Created": "2021-09-15T19:28:08.972887703+02:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

Figura 11. Características de la red Bridge.

4.3 Preparación de Snort

En este apartado se comenzará con la configuración que se debe llevar a cabo para la configuración de snort, en este caso funcionando desde docker.

Una vez con la consola abierta y logueados como root entraremos al docker de snort con el siguiente comando:

```
$ docker run -it --network=[MODO]snort-filebeat /bin/bash
```

En la variable `--network` se indicará host o bridge dependiendo del modo de red que se quiera implementar y que han sido explicados en el apartado anterior.

Automáticamente cambiará la ruta en el prompt a /opt#, se comprueba como se indica en la Figura 12 la versión de Snort.

```
root@ubuntuDocker:/opt# snort -V

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.8.2 GRE (Build 335)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.5.3
           Using PCRE version: 8.31 2012-07-06
           Using ZLIB version: 1.2.8
```

Figura 12. Versión de Snort instalada.

El archivo de configuración principal en el Docker de Snort se encuentra en la siguiente ruta: `/etc/snort/etc/snort.conf`

Es necesario cambiar las líneas que se muestran en la Figura 13:

```
##################################################
# Step #1: Set the network variables.  For more information, see README.variables
##################################################

# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.0.29/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET !$HOME_NET
```

Figura 13. Configuración del archivo snort.conf.

En la variable `HOME_NET` se especificará la dirección o el rango de direcciones, también se puede dejar por defecto el valor `any` que coincide con cualquier dirección IP.

De la misma forma la variable `EXTERNAL_NET` especifica las redes externas el valor `¡$HOME_NET` significa todas las direcciones que no sean las especificadas en `HOME_NET`.

La Figura 14 muestra el ultimo parámetro a modificar, en la variable `RULE_PATH` se deberá especificar la ruta donde se encuentran los conjuntos de reglas que utilizará Snort.

```
# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules

# If you are using reputation preprocessor set these
var WHITE_LIST_PATH /etc/snort/rules
var BLACK_LIST_PATH /etc/snort/rules
```

Figura 14. Ruta de las reglas de snort.

Para finalizar con la configuración de Snort, se deben de añadir reglas. Los conjuntos de reglas son la arteria principal de la función de detección de intrusos de Snort. Existen varios tipos de reglas: Reglas oficiales, reglas de la comunidad, reglas locales, etc.

Como el propósito de este proyecto no es profundizar en Snort, se va a crear una regla local en el archivo `/etc/snort/rules/local.rules` que nos permita monitorizar cuándo se está haciendo un ping al equipo que se está monitorizando. En la Figura 15 se muestra la definición de la regla.

```
#------------
# LOCAL RULES
#------------
alert icmp any any -> any any (msg:"ALERTA ICMP. Se está recibiendo Ping";sid:1000001;)
# alert tcp any any -> any 6667 (msg:"IRC protocol traffic";sid:1000005;)
#alert tcp any 6667 -> any any (msg:"C&C Server sent netinfo command";content:"!netinfo";
sid:1000006;)
```

Figura 15. Regla para detectar tráfico ICMP.

Esta regla genera una alerta al detectar la presencia de paquetes ICMP en el tráfico capturado, sin tener en cuenta la dirección IP ni los puertos origen y destino. Por último, se añade un identificador especifico de la regla, que deberá ser siempre mayor que 1000000.

4.3.1 Filebeat

Para configurar el módulo Filebeat se deberá editar el archivo filebeat.yml situado en `/etc/filebeat/filebeat.yml`

En primer lugar, se habilitan las entradas donde obtiene Filebeat los datos. Como se muestra en la Figura 16, se deberá habilitar poniendo a true el módulo log y cambiar la ruta de donde se obtienen los mismos.

```
#=========================== Filebeat inputs =============================

filebeat.inputs:

# Each - is an input. Most options can be set at the input level, so
# you can use different inputs for various configurations.
# Below are the input specific configurations.

- type: log

  # Change to true to enable this input configuration.
  enabled: true

  # Paths that should be crawled and fetched. Glob based paths.
  paths:
    #- /var/log/*.log
    #- /var/log/snort/*log*
    - /var/log/snort/alert
    #- c:\programdata\elasticsearch\logs\*
```

Figura 16. Configuración de entradas de Filebeat.

En la Figura 16, se configura Filebeat para conectarse al servidor Kibana, en este ejemplo en la dirección 192.168.0.29, dirección del host.

Cabe destacar que el servidor puede estar alojado en otro equipo de la red o en un Docker aislado conectado a la red bridge donde tendría una dirección de la subred 172.17.0.0/16.

De igual forma como también se observa en la Figura 17, se configurará el servidor Filebeat para conectarse al servidor ElasticSearch en la dirección 192.168.0.29 en el puerto 9200.

Con esto se daría dar por Finalizada la configuración de Filebeat de una manera bastante sencilla e intuitiva modificando simplemente el archivo Filebeat.yml

```
#============================== Kibana =====================================

# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.
# This requires a Kibana endpoint configuration.
setup.kibana:

  # Kibana Host
  # Scheme and port can be left out and will be set to the default (http and 5601)
  # In case you specify and additional path, the scheme is required: http://localhost:5601/path
  # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
  host: "192.168.0.29:5601"

  # Kibana Space ID
  # ID of the Kibana Space into which the dashboards should be loaded. By default,
  # the Default Space will be used.
  #space.id:


#============================== Outputs =====================================

# Configure what output to use when sending the data collected by the beat.

#-------------------------- Elasticsearch output ------------------------------
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["192.168.0.29:9200"]

  # Protocol - either `http` (default) or `https`.
  #protocol: "https"
```

Figura 17. Configuración de servidores Kibana y ElasticSearch.

4.4 Preparación de ELK Stack

Para ejecutar el contenedor de ELK desde la imagen descargada se deberá utilizar el siguiente comando:

```
$ docker run -p 5601:5601 -p 9200:9200 -p 5044:5044 -it --
name elk sebp/elk
```

El parámetro `-p` publica los puertos que son necesarios para el correcto funcionamiento de la pila de ELK, el parámetro `-it` permite mantener el contenedor en ejecución y poder interactuar con él y el parámetro `--name` le asigna el nombre elk al contenedor.

Como se ha comentado, el comando anterior publica los siguientes puertos, que son necesarios para el correcto funcionamiento de ELK stack:

- 5601 (interfaz web de Kibana).
- 9200 (interfaz JSON de Elasticsearch).
- 5044 (interfaz de Logstash Beats, recibe registros de Beats como Filebeat).

La imagen expone (pero no publica):

- Interfaz de transporte de Elasticsearch en el puerto 9300. Esta interfaz de transporte es utilizada principalmente por la API de cliente Java de Elasticsearch y para ejecutar Elasticsearch en un clúster.
- API de monitoreo de Logstash en el puerto 9600. [15]

En la Figura 18 se muestra como encajan todas las piezas juntas:

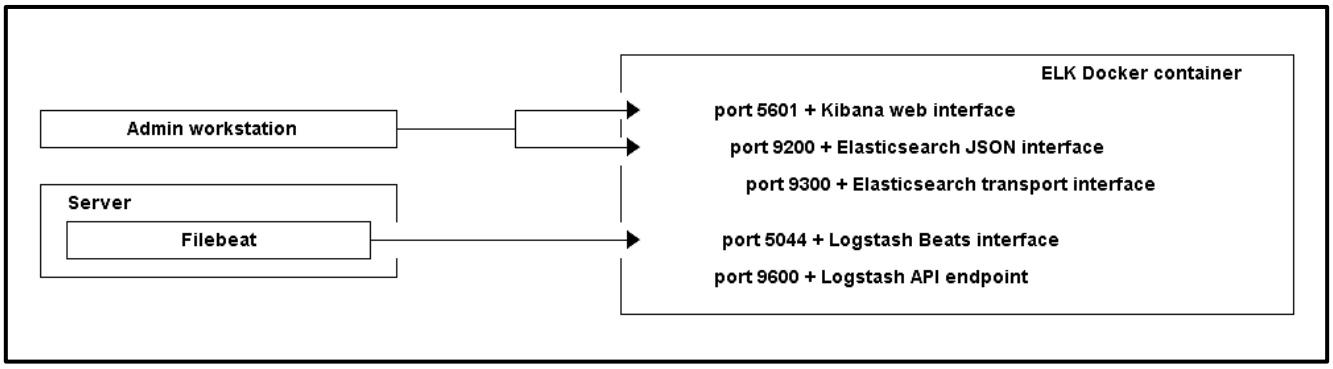

Figura 18. Elementos que intervienen e interactúan con el docker ELK

5. Descripción de la aplicación propuesta

Una vez configurados los contenedores de Docker y los elementos instalados dentro de ellos, se procederá a hacer una demostración de la solución completa. Como se ha comentado, el sistema consta de dos contenedores Docker independientes, en la Figura 19 se muestra un esquema del sistema completo.

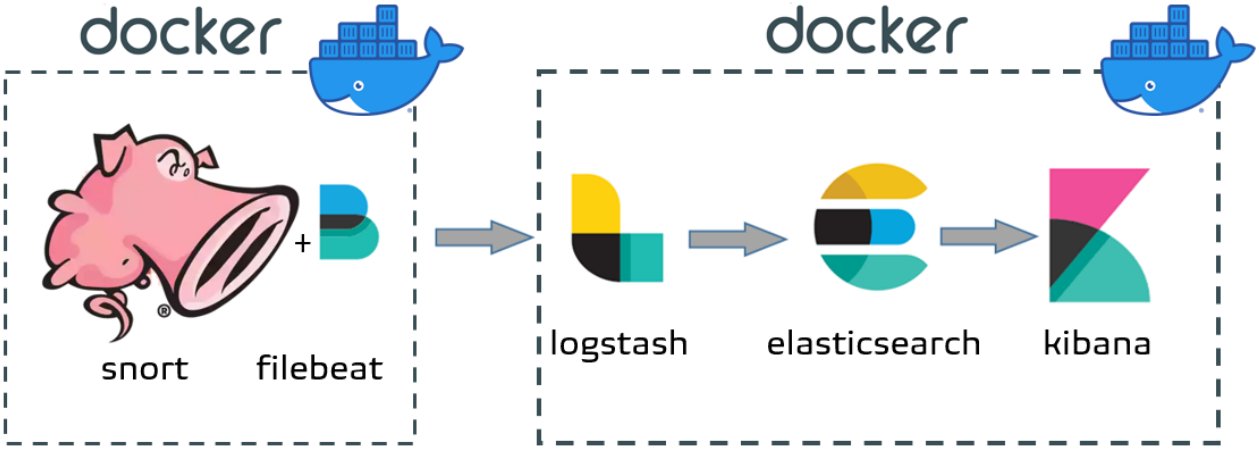

Figura 19. Esquema de la solución propuesta

5.1 Escenario

Se ha configurado La máquina virtual de Ubuntu con una IP estática de la forma que se muestra en la Figura 20.

Figura 20. Configuración de IP estática.

En la Figura 21 se muestran los parámetros y características de las interfaces de red en el host.

```
root@ubuntuDocker:/home/lolo# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:92:f3:5b:2d  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4419<UP,BROADCAST,RUNNING,PROMISC,MULTICAST>  mtu 1500
        inet 192.168.0.29  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::c039:c0ab:cf96:f01f  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:bd:1b:fb  txqueuelen 1000  (Ethernet)
        RX packets 1794713  bytes 1898819692 (1.8 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 21707  bytes 1670871 (1.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Bucle local)
        RX packets 672  bytes 58214 (58.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 672  bytes 58214 (58.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 21. Interfaces de red en el host

5.2 Análisis del tráfico en tiempo real con Snort

Como se ha comentado anteriormente, según la red en la que se quiera monitorizar existen 2 opciones con snort modificando el parámetro `--network`:

```
$ docker run -p 80:80 -it --network=host snort-filebeat
/bin/bash
$ docker run -p 80:80 -it --network=bridge snort-filebeat
/bin/bash
```

5.2.1 Host

Al lanzarlo en la red host, el contenedor usará la misma red y las interfaces del host. Para validar la configuración se utilizará el siguiente comando:

```
$ snort -T -c /etc/snort/etc/snort.conf -i enp0s3
```

El parámetro `-T` indica que teste y reporte el resultado de la configuración, el parámetro `-c` indica que se use la configuración del archivo especificado y el parámetro `-i` que escuche en la interfaz especificada.

En la Figura 22 se muestra que la configuración ha sido validada correctamente.

```
      --== Initialization Complete ==--

  ,,_     -*> Snort! <*-
 o"  )~   Version 2.9.8.2 GRE (Build 335)
  ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
          Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
          Copyright (C) 1998-2013 Sourcefire, Inc., et al.
          Using libpcap version 1.5.3
          Using PCRE version: 8.31 2012-07-06
          Using ZLIB version: 1.2.8

          Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.6  <Build 1>
          Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
          Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
          Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
          Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
          Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
          Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
          Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
          Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
          Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
          Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
          Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
          Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
          Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
          Preprocessor Object: SF_POP  Version 1.0  <Build 1>

Snort successfully validated the configuration!
Snort exiting
```

Figura 22. Validación de la configuración de Snort

Con el siguiente comando ejecutaremos snort para que empiece a monitorizar:

```
$ snort -A fast -c /etc/snort/etc/snort.conf -q -i enp0s3
```

El parámetro `-A fast` indica que se lanzará Snort en modo rápido, por otro lado, el prametro `-q` indica que trabajará en modo silencioso.

5.2.2 Bridge

Al lanzarlo en la red Bridge los contenedores se conectarán a la red 172.17.0.0 como se puede ver en la Figura 23 el mismo servidor Docker utiliza la IP 172.17.0.1, cada contenedor que se inicie adquiere un IP posterior al último conectado.

```
lolo@ubuntuDocker:~$ ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:92ff:fef3:5b2d  prefixlen 64  scopeid 0x20<link>
        ether 02:42:92:f3:5b:2d  txqueuelen 0  (Ethernet)
        RX packets 17  bytes 956 (956.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 47  bytes 6767 (6.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 23. Interfaz que se crea al instalar Docker

En la Figura 24 se muestran las IPs asignadas los contenedores de Snort (172.17.0.2) y ELK (172.17.0.3) después de inspeccionar la red bridge con el comando:

```
$ docker network inspect bridge
```

```
        "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
        "a41ed81328acbb5e394a7d18ef9462e2cfd3b62d79c4c63d5c7916fb6bd68f36": {
            "Name": "festive_kalam",
            "EndpointID": "6e31c4a3229d8eaaf84de5153c0f78606efec1d06529e08e458349ea64ad59ef",
            "MacAddress": "02:42:ac:11:00:02",
            "IPv4Address": "172.17.0.2/16",
            "IPv6Address": ""
        },
        "c850b003b951d802e8d116e7cf8378fd843647a6994539bbc2d0a1f859355e7c": {
            "Name": "elk",
            "EndpointID": "f8938187f3f55825ef417f0426373d19efe15edb04c30f1398be0e2bfbd490f5",
            "MacAddress": "02:42:ac:11:00:03",
            "IPv4Address": "172.17.0.3/16",
            "IPv6Address": ""
        }
```

Figura 24. Caracteristicas de la red bridge

Para lanzar Snort se deberá comprobar el nombre de la interfaz dentro del contenedor de Snort.

En la Figura 25 se muestra la configuración de red dentro del contenedor de snort.

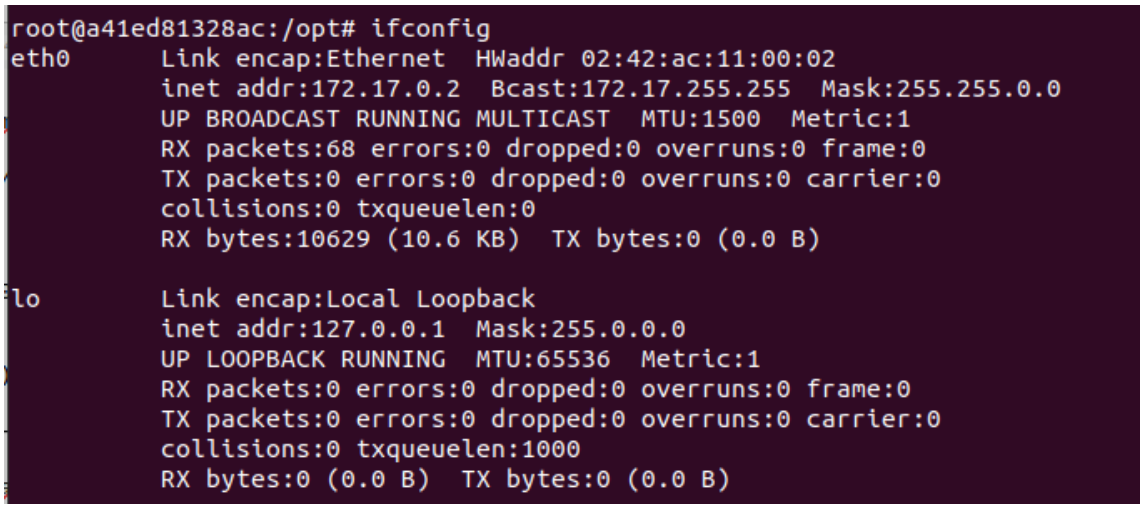

Figura 25. Interfaces de red en el contenedor de snort.

Por lo que los comandos para validar y lanzar snort en este tipo de red serían los siguientes:

```
$ snort -T -c /etc/snort/etc/snort.conf -i eth0
$ snort -A fast -c /etc/snort/etc/snort.conf -q -i eth0
```

5.3 Agente Filebeat

Se utilizará el siguiente comando desde el docker de snort para crear los paneles de Filebeat en el servidor Kibana, en la Figura 26 se muestra el resultado.

```
$ filebeat setup
```

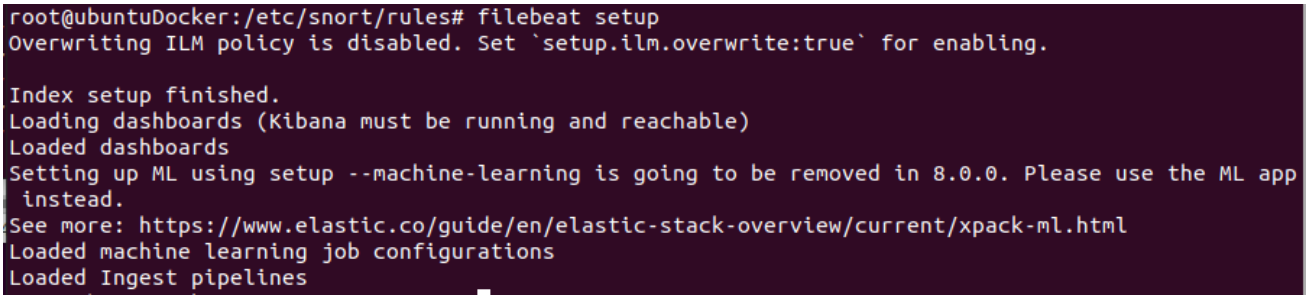

Figura 26. Creación de los paneles de Filebeat en Kibana.

A continuación, se inicia el servicio de Filebeat

```
$ service filebeat start
```

Si todo ha sido correcto y tiene comunicación con kibana y filebeat saldrá una confirmación de configuración correcta como se muestra en la Figura 27.

```
2021-09-17T02:44:10.372Z        INFO    instance/beat.go:298    Setup Beat: filebeat; Version: 7.6.2
2021-09-17T02:44:10.372Z        INFO    [index-management]      idxmgmt/std.go:182      Set output.elasticsearch.inde
x to 'filebeat-7.6.2' as ILM is enabled.
2021-09-17T02:44:10.372Z        INFO    elasticsearch/client.go:174     Elasticsearch url: http://192.168.0.29:9200
2021-09-17T02:44:10.373Z        INFO    [publisher]     pipeline/module.go:110  Beat name: ubuntuDocker
Config OK
```

Figura 27. Iniciación del servicio de Filebeat

5.4 ELK

Desde otra ventana de consola logueado como root, se lanzará el docker de ELK con el siguiente comando:

```
$ docker run -p 5601:5601 -p 9200:9200 -p 5044:5044 -it \--
name elk --network=host sebp/elk
```

En la figura 28 se muestra el proceso de inicialización del docker del servidor de ElasticStack.

```
root@ubuntuDocker:/home/lolo# sudo docker run -p 5601:5601 -p 9200:9200 -p 5044:
5044 -it \--name elk --network=host sebp/elk
WARNING: Published ports are discarded when using host network mode
 * Starting periodic command scheduler cron                              [ OK ]
 * Starting Elasticsearch Server                                         [ OK ]
waiting for Elasticsearch to be up (1/30)
waiting for Elasticsearch to be up (2/30)
waiting for Elasticsearch to be up (3/30)
waiting for Elasticsearch to be up (4/30)
waiting for Elasticsearch to be up (5/30)
waiting for Elasticsearch to be up (6/30)
waiting for Elasticsearch to be up (7/30)
waiting for Elasticsearch to be up (8/30)
waiting for Elasticsearch to be up (9/30)
waiting for Elasticsearch to be up (10/30)
waiting for Elasticsearch to be up (11/30)
waiting for Elasticsearch to be up (12/30)
waiting for Elasticsearch to be up (13/30)
waiting for Elasticsearch to be up (14/30)
waiting for Elasticsearch to be up (15/30)
waiting for Elasticsearch to be up (16/30)
Waiting for Elasticsearch cluster to respond (1/30)
logstash started.
 * Starting Kibana5                                                      [ OK ]
```

Figura 28. Inicialización de ELK

De manera análoga al docker de Snort, para lanzar el docker de ELK en la red Bridge se deberá sustituit el parámetro `-network=host` por el parámetro: `--network=bridge` de este modo para acceder al servidor de kibana se debe apuntar a la dirección 172.17.0.3.

Como se observa en la Figura 29, se ha generado tráfico ICMP desde el host principal de Windows que está conectado a la VM por red puenteada, este equipo tiene la IP 192.168.0.17

```
Microsoft Windows [Versión 10.0.19042.1165]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\UX490>ping 192.168.0.29

Haciendo ping a 192.168.0.29 con 32 bytes de datos:
Respuesta desde 192.168.0.29: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.0.29: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.0.29: bytes=32 tiempo=1ms TTL=64
Respuesta desde 192.168.0.29: bytes=32 tiempo=1ms TTL=64

Estadísticas de ping para 192.168.0.29:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 0ms, Máximo = 1ms, Media = 0ms
```

Figura 29. Generación de tráfico ICMP

A continuación para acceder a la consola de kibana se deberá abrir el explorador y acceder a la dirección del servidor de la forma que se muestra en la Figura 30.

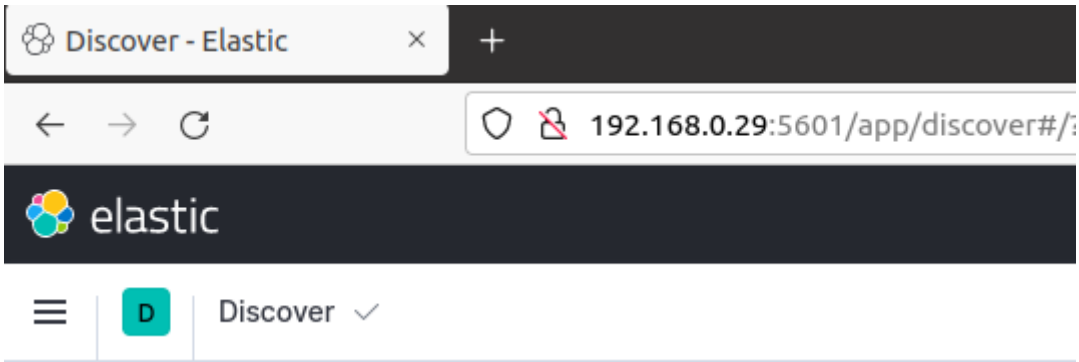

Figura 30. URL de acceso al dashboard de kibana.

Una vez dentro, ha detectado automáticamente el módulo de Filebeat, y en la Figura 31 se observa el aspecto general de la consola de kibana.

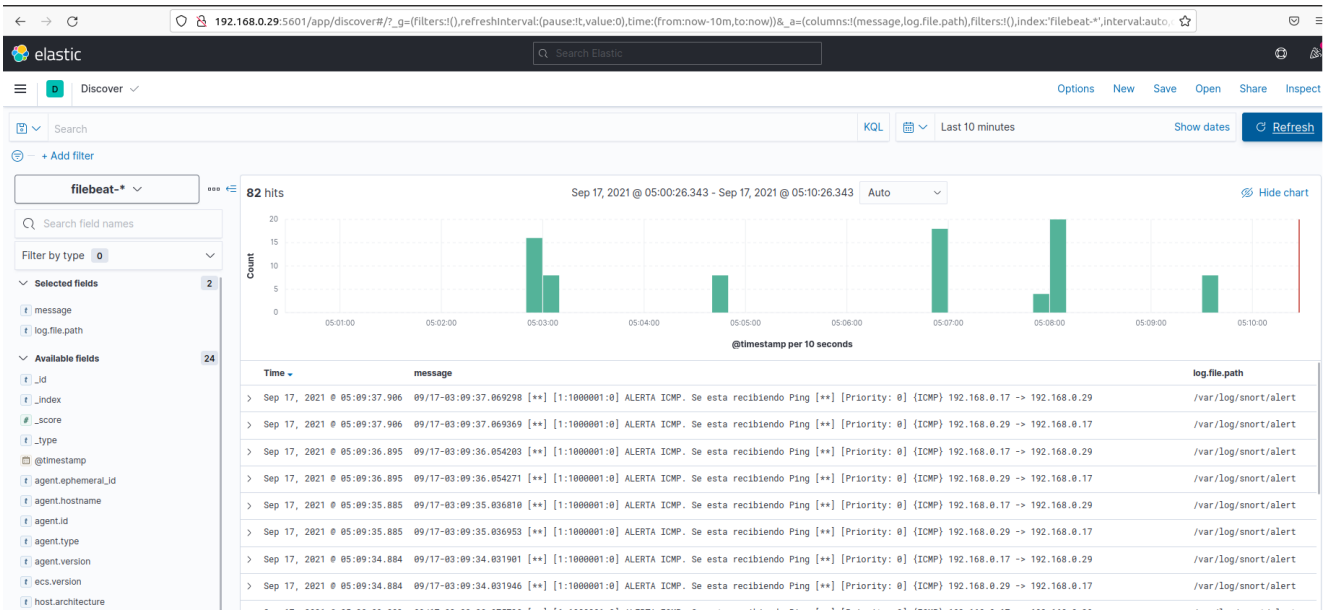

Figura 31. Consola principal kibana

En la esquina superior derecha, al lado del botón de refresco, se puede modificar para que se muestre el rango de tiempo que interese observar, en la Figura 32 se puede apreciar cómo se puede configurar fecha y tiempos e incluso acotar el rango minuciosamente para ver datos a tiempo real.

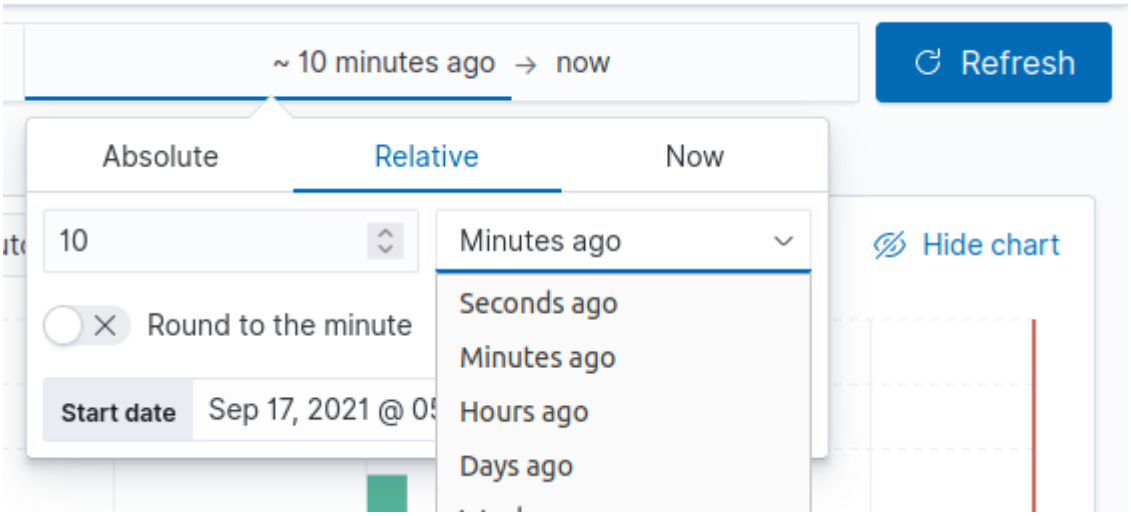

Figura 32. Configuración del rango de tiempo.

En la columna de la derecha se pueden filtrar los datos que sean interesantes para mostrar al usuario, en la Figura 33 se han filtrado los campos `message` y `log.file.path` para mostrar el mensaje de la alerta y a su vez observar de que fuente está recogiendo los logs de alerta.

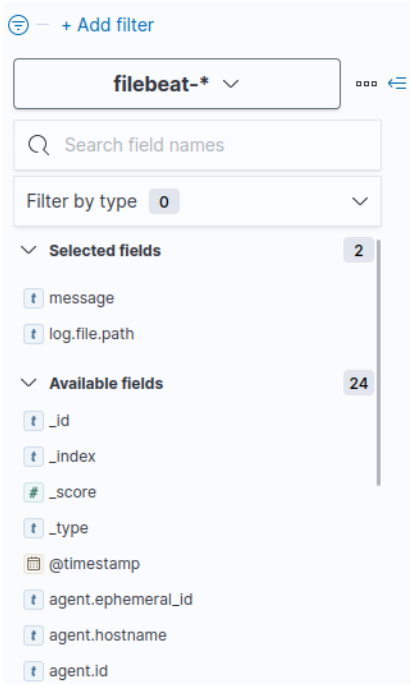

Figura 33. Filtrado de datos a mostrar.

Para finalizar, la Figura 34 muestra el panel principal con los gráficos de barras y los datos del tiempo a la que se recibió la alerta, mensaje de alerta y de donde se están recopilando esos datos.

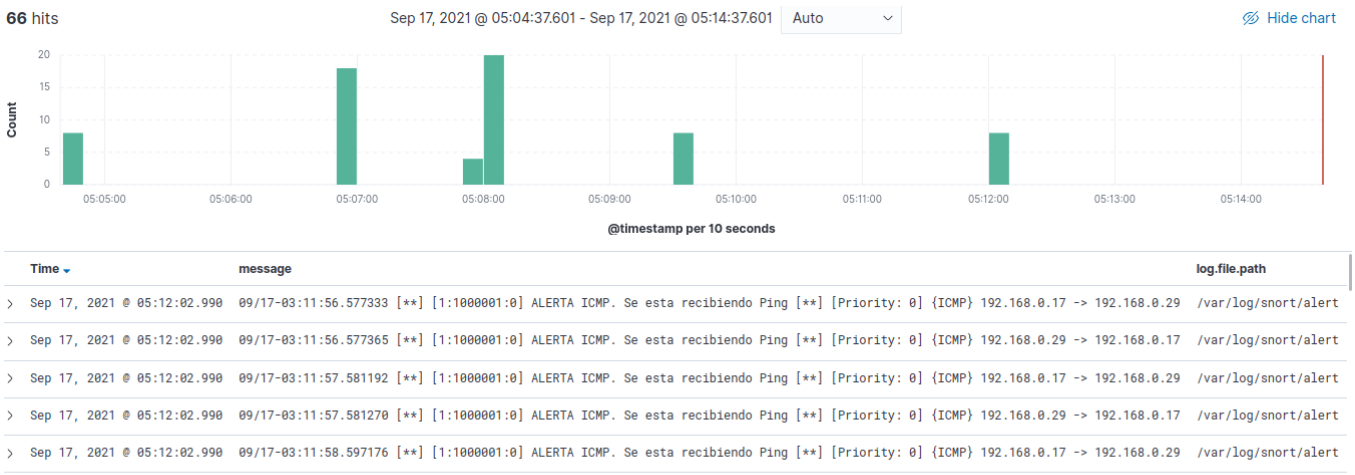

Figura 34. Panel principal de kibana.

6. Conclusiones y líneas futuras

Hoy en día y cada vez más, se están viviendo numerosos episodios de ataques informáticos a instituciones y empresas tanto públicas como privadas. Estos ataques afectan tanto a clientes de dichas compañías como a la seguridad de instituciones a nivel nacional o regional.

Una empresa para ser competitiva o atractiva debe contar con recursos y sistemas con un alto nivel de disponibilidad, lo que exige una gestión efectiva y un complejo proceso en la gestión y la protección de los datos. Es por esto por lo que las instituciones u organizaciones tienen que prestar especial atención a protegerse con herramientas de detección de intrusiones debido a que un ataque podría suponer la pérdida o filtración de multitud de datos ocasionando daños económicos y morales de niveles incalculables.

Al implementar un sistema como el del presente trabajo se ha podido comprobar como la tecnología de contenedores, en concreto con Docker, puede ser una buena opción para la implementación de un sistema de detección de intrusiones.

Las ventajas que pueden ofrecer sistemas de este tipo son numerosas, desde la encapsulación y aislamiento de todo el trabajo y los datos, pasando por una fácil instalación y configuración en cualquier entorno y sobre todo por poder optimizar los recursos gracias a su ligereza y al consumo mínimo de los mismos. Además, es una tecnología que es portable y autosuficiente ya que no necesita de otro software para funcionar.

Para finalizar, durante la elaboración del proyecto realizado se han considerado una serie de líneas de futura implantación interesantes, como para tener en consideración para futuros proyectos.

Por ejemplo, la implantación de este sistema junto a un conjunto de reglas personalizadas para utilizar en laboratorios de prácticas, monitorizar el tráfico de una pequeña empresa para detectar usos indebidos de la red o incluso para implantar en aulas y monitorizar si en exámenes con dispositivos electrónicos como podrían ser los exámenes de prácticas, los alumnos acceden o descargan material no autorizado durante la realización de los mismos.

7. Bibliografía

[1] Incibe. ¿Qué son y para qué sirven los SIEM, IDS e IPS? 2020. Disponible en: https://www.incibe.es/protege-tu-empresa/blog/son-y-sirven-los-siem-ids-e-ips [Última consulta: 25-08-2021].

[2] Snort. What is Snort? 2021. Disponible en: https://www.snort.org/ [Última consulta: 25-08-2021]

[3] Rasilla Villegas, Naiara. 2020. Desarrollo de una prueba de concepto de un detector de intrusiones para su aplicación en prácticas de laboratorio. Trabajo de fin de grado. Grado en Ingeniería de Tecnologías de la Telecomunicación. Universidad de Cantabria. Santander. Citado en: 26-08-2021. Recuperado a partir de: https://repositorio.unican.es/xmlui/bitstream/handle/10902/19231/427284.pdf

[4] M. Roesch, Snort: Lightweight intrusion detection for networks, Lisa, volumen. 99, 2000, págs. 231-238.

[5] Logz. Filebeat vs. Logstash: The Evolution of a Log Shipper. 2020. Disponible en: https://logz.io/blog/filebeat-vs-logstash/ [Última consulta: 26-08-2021]

[6] Mejía Viteri, Alberto. 2019. Sistema de gestión de eventos relacionados con anomalías en la seguridad en un entorno Open Source. Trabajo final de máster. Máster Universitario en Seguridad Informática (Ciberseguridad) Universidad de Cádiz. Puerto Real. Citado en: 28-08-2021. Recuperado a partir de: https://rodin.uca.es/xmlui/handle/10498/22569

[7] Docker docs. Docker overview. 2021. Disponible en: https://docs.docker.com/get-started/overview/ [Última consulta: 21-08-2021].

[8] Docker docs. Docker. Networking overview. 2021. Disponible en: https://docs.docker.com/network/ [Última consulta: 21-08-2021].

[9] PhoenixNAP. How to Commit Changes to a Docker Image with Examples. 2019. Disponible en: https://phoenixnap.com/kb/how-to-commit-changes-to-docker-image [Última consulta: 23-08-2021].

[10] Digitalocean. How To Install and Use Docker on Ubuntu 20.04. 2020. Disponible en: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04 [Última consulta: 30-08-2021].

[11] Docker hub. linton/docker-snort. 2016. Disponible en: https://hub.docker.com/r/linton/docker-snort [Última consulta: 01-09-2021].

[12] Tech Expert Tips. Filebeat - Sending the Syslog Messages to ELK. 2018. Disponible en: https://techexpert.tips/elasticsearch/filebeat-sending-syslog-messages-elasticsearch/ [Última consulta: 02-09-2021]

[13] Docker hub. Elasticsearch, Logstash, Kibana (ELK) Docker image. 2021. Disponible en: https://hub.docker.com/r/sebp/elk/ [Última consulta: 01-09-2021].

[14] Elastic. Virtual memory. 2021. Disponible en: https://www.elastic.co/guide/en/elasticsearch/reference/current/vm-max-map-count.html [Última consulta: 01-09-2021].

[14] Elk-docker. Elasticsearch, Logstash, Kibana (ELK) Docker image documentation. 2021. Disponible en: https://elk-docker.readthedocs.io/ [Última consulta: 02-09-2021]