Implementación de un sistema de detección de intrusos IDS mediante la inspección del tráfico a través de la red

Antonio Suárez Bono
Máster en Ciberseguridad y Privacidad

Área de Análisis de Datos

Tutor del Trabajo Final:
Joan Caparrós Ramírez
Responsable de la asignatura:
Andreu Pere Isern Deyà

Convocatoria de Junio, curso 2022/23

Universitat Oberta de Catalunya

Ficha del Trabajo Final

Título del trabajo:	Implementación de un sistema de detección de intrusos IDS mediante la inspección del tráfico a través de la red
Nombre del autor:	Antonio Suárez Bono
Nombre del consultor/a:	Joan Caparrós Ramírez
Nombre del PRA:	Andreu Pere Isern Deyà
Fecha de entrega (mm/aaaa):	06/2023
Titulación o programa:	Máster en Ciberseguridad y Privacidad
Área del Trabajo Final:	Análisis de Datos
Idioma del trabajo:	Castellano
Palabras clave:	IDS, SIEM

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

En el presente Trabajo Final de Máster se aborda el problema de la seguridad en los dispositivos de Internet de las Cosas (IoT) y se propone una solución para detectar comportamientos anómalos o malintencionados en la red. La solución propuesta utiliza un microordenador conectado a la red inalámbrica para analizar el tráfico de la red y producir alertas para el administrador de la misma. Los objetivos del trabajo incluyen la investigación de las herramientas disponibles, profundizar en la instalación y configuración del IDS, y de las herramientas que permita visualizar y analizar los resultados producidos.
Finalmente, el trabajo se ha concluido satisfactoriamente en cuanto que se ha logrado implementar un sistema de detección de intrusos utilizando Snort y a su vez, se ha logrado visualizar un analizar los logs generados desde la pila de aplicaciones ELK.

Abstract (in English, 250 words or less):

The problem of security in Internet of Things (IoT) devices is addressed and then, a solution is proposed to detect anomalous or malicious behavior on the network. The proposed solution uses a microcomputer connected to the wireless network to analyze the network traffic and produce alerts for the network administrator.
The objectives of the work include the investigation of the available tools, delve into the installation and configuration of the IDS, and the tools that allow visualizing and analyzing the results produced.
Finally, the work has been satisfactorily completed in that it has been possible to implement an intrusion detection system using Snort and, in turn, it has been possible to visualize and analyze the logs generated from the ELK application stack.

Índice general

Índice de figuras

Índice de cuadros

Índice de extractos de código

1. Introducción

1.1. Contexto y justificación del trabajo

A lo largo de las últimas décadas, tanto Internet como los sistemas informáticos han generado un gran número de incidentes de seguridad debido al uso de las redes. Las estadísticas del CERT informan que el número de intrusiones aumenta año por año.

Hoy en día, el estilo de vida de las personas de encuentra constantemente cambiando debido al los avances en la tecnología. Cuando miramos detenidamente que dispositivos se encuentran conectados a la red de nuestros hogares podemos encontrar todo tipo de dispositivos .

La adopción masiva de la tecnología IoT (Internet of Things) en nuestros hogares, se ha convertido en un objetivo atractivo para los ciber amenazas. Estas amenazas pueden ir desde desbloquear puertas y espiar a los ocupantes a través de sus propias cámaras hasta apropiarse de asistentes personales que se encuentren controlados por voz [13]. En la primera mitad de año 2022, el número de ataques sobre dispositivos IoT duplica la cifra registrada para el año 2021 [9].

Estos aparatos inteligentes traen los riesgos de los sistemas de control industrial a la vida real, por lo que ha ganado un interés significativo por parte de los investigadores y especialistas en ciberseguridad [20].

La mejor forma de proteger un dispositivo de tecnología IoT ante ciber ataques, es tener en cuenta la seguridad desde el diseño [23]. El problema principal se debe a la heterogeneidad que existe en los protocolos utilizados por los dispositivos de IoT [12].

1.2. Solución que se plantea

En este trabajo, se plantea encontrar e implementar una solución de seguridad que analice el uso de la red por parte de los distintos dispositivos que se encuentran conectados, con la finalidad de detectar comportamientos anómalos o malintencionados.

Para implementar un sistema con tales características, se propone utilizar un microordenador que se conecte a la red inalámbrica de forma que tenga acceso al tráfico que circula por la red. Este tráfico capturado, podrá ser analizado en primera instancia por un IDS (Sistema de Detección de Intrusos). Una vez el IDS haya comprobado en tráfico de red, se generarán eventos que registren posibles accesos no autorizados a la red y comportamientos malintencionados.

En segundo lugar, se propone implementar una herramienta de análisis de eventos que nos permita visualizar la información con mayor facilidad. También se busca en este proyecto, que el sistema de análisis de eventos envíe alertas al administrador de la red con la información más relevante en caso de que se produzca un incidente de seguridad.

1.3. Objetivos del trabajo

Para comprender mejor el alcance del trabajo, se ha realizado un distinción entre el objetivo general y los objetivos específicos del trabajo.

1.3.1. Objetivo general

El objetivo final de este proyecto es la implementación de un IDS (Sistema de Detección de Intrusos) utilizando herramientas hardware de bajo coste que permita analizar el tráfico de desde un

punto conflictivo de la red y alertar ante comportamientos malintencionados.

Para alcanzar este resultado se han definido los siguientes objetivos:

- Investigar las herramientas disponibles para implementar el sistema de detección de intrusos y la herramienta de análisis de logs.
- Profundizar en la instalación y configuración del sistema de detección de intrusos seleccionado.
- Profundizar en la instalación y configuración de la herramienta seleccionada para el análisis y visualización de datos.

1.3.2. Objetivos específicos

Además de la implementación del sistema de detección se intrusos, es posible distinguir una serie de objetivos a nivel educacional o de competencias a adquirir:

- Profundizar en la configuración del hardware donde se va a implementar el sistema.
- Estudiar en profundidad las posibilidades ofrecidas por el sistema de detección de intrusos elegido.
- Explorar las distintas herramientas disponibles para enviar alerte por diferentes vías de comunicación.
- Profundizar en las opciones de visualización y análisis ofrecidas por la herramienta de análisis de logs.

1.4. Revisión del estado del arte

En este apartado, se realiza una investigación simple sobre el estado del arte del producto que se busca implementar. En esta búsqueda, se puede realizar una distinción entre el sistema de detección de intrusos y sistema de análisis de eventos.

1.4.1. Sistema de Detección de Intrusos

La clasificación de los sistemas de detección de intrusos pude variar según su enfoque, su comportamiento ante las intrusiones y los tipos de sistemas que monitorizan.

Una de las clasificaciones más destacadas es la que los distingue según el tipo de sistemas que monitoriza. En ese caso podemos distinguir entre los sistemas que analizan la actividad en red (NIDS) y los sistemas que monitorizan la actividad de terminales determinados (HIDS).

En el cuadro 1.1, se expone una table comparativa de algunos de los productos IDS existentes en el mercado [15]. La tabla incluye algunos de los productos más conocidos en cuanto a sistemas de detección de intrusos (basados en máquina y en red).

IDS	**Licencia**	**Compatibilidad**	**Tipo**
AlienVault	Propietaria	Linux y Windows	HIDS
MacAfee Network Security Plataform	Propietaria	Linux y Windows	NIDS
Palo Alto Networks Next-Generation Firewall	Propietaria	PAN-OS	NIDS
Snort	Libre	Linux, FreeBSD, Windows, MacOS	NIDS
Suricata	Libre	Linux, FreeBSD, Windows, MacOS	NIDS
Zeek	Libre	Linux, FreeBSD, MacOS	NIDS
Ossec	Libre	Linux, FreeBSD, Windows, MacOS	HIDS
Samhain	Libre	Unix, Linux, Cygwin/Windows	HIDS

Cuadro 1.1: Tabla comparativa IDS

Para este proyecto se va a elegir un sistema de detección de intrusos basado en red, que tenga una licencia gratuita (software libre). Las herramientas destacadas en la actualidad en cuanto a la detección de intrusos en red son Snort y Suricata.

La principal desventaja de los sistemas de detección de intrusos en red radica en la necesidad de procesar grandes flujos de datos. Esto quiere decir que, si el tráfico de red excede la capacidad de análisis, esto puede ocasionar el rechazo de paquetes sin que sean previamente analizados. En este caso, el IDS perdería efectividad.

1.4.2. Recopilación y Análisis de Eventos

En este apartado se expondrá una serie de herramienta que podrán ser utilizadas para la implementación del sistema de monitorización de eventos.

A continuación, enumero algunas de las tareas que el sistema de monitorización de eventos tiene que contemplar:

- Recopilación de logs
- Análisis de logs
- Panel de administración para la visualización gráfica.

Para este proyecto se utilizará la pila de aplicaciones ELK. Esta pila de herramientas consiste principalmente de tres proyectos open source que en su conjunto son capaces de recolectar datos de diferentes fuentes, analizar datos y visualizarlos gráficamente en un cuadro de mandos [3]. Las herramientas que componen esta pila son las siguientes:

- **Beats**: Es una plataforma para agentes de datos de un solo propósito, que envía datos a un sistema Logstash o a un Elasticsearch.
- **Logstash**: Es un flujo de datos que permite ingerir datos de múltiples fuentes transformarlos y enviarlos a la base de datos.
- **Elasticsearch**: Motor de búsqueda distribuido, fácil de usar y escalable.
- **Kibana**: Interfaz de usuario gratuita que permite visualizar datos de Elasticsearch.

1.5. Enfoque y seguimiento metodológico

En este apartado se explica como es el flujo de trabajo del proyecto. Para plantear la metodología de trabajo se ha tenido en cuenta los hitos marcados por la asignatura. De esta forma, se pueden distinguir 3 iteraciones a modo de seguimiento. La metodología de trabajo será iterativa y en cascada.

Decimos que la metodología es iterativa debido a que se plantean iteraciones con aproximadamente la duración de un mes y tras cada una de ellas se realiza una entrega informando del estado del proyecto para ser revisado. Además, cada una de las iteraciones busca concluir fases distintas del proyecto. Además, podemos decir que es en cascada debido a que en cada una de las fases del desarrollo se encuentra separada.

El proyecto comienza con una primera fase de planificación, donde se definen los objetivos y se elabora un cronograma que contemple todas las tareas o actividades que serán necesarias en el transcurso del proyecto.

1.5.1. Etapas del proyecto

En este apartado se definen las etapas del proyecto y el propósito de cada una de ellas. El desarrollo del proyecto se encuentra dividido en tres fases según su estado de madurez. Las etapas de desarrollo del proyecto son las siguientes:

- **Investigación**: Esta primera etapa busca recabar la información que sea posible sobre cada una de las herramientas que serán utilizadas en la implementación. Además, se aprenderá a usar cada una de ellas en un entorno virtual.
- **Implementación**: Esta segunda etapa consiste en aplicar los conocimientos aprendidos para implementar el sistema de detección de intrusos en un dispositivo hardware y que sea completamente funcional.
- **Optimización y Pruebas**: Esta tercera etapa del proyecto busca profundizar en posibles herramientas de interés que puedan ser incorporadas al proyecto. Además, en esta etapa se crearán reglas y situaciones que el IDS deberá ser capaz de detectar correctamente.

Además, en cada una de las fases del proyecto se irá actualizando la documentación generada y se revisará la planificación para determinar si el progreso es adecuado y registrar los posibles problemas que hayan podido encontrarse a lo largo del proyecto.

1.5.2. Herramientas utilizadas

Para la elaboración de la planificación temporal y el cronograma se ha utilizado la herramienta **GanttProject**. Este programa de código abierto tiene como objetivo la administración de proyectos utilizando el diagrama de Gantt.

Para realizar el seguimiento de las tareas se ha utilizado **Trello**. Se trata de un software de administración de proyectos con interfaz web.

Para la redacción de la memoria se ha utilizado el editor gratuito **Texmaker**. Este editor permite escribir documentos de texto utilizando el sistema de composición de textos **LaTeX**.

1.6. Planificación del trabajo

En esta sección, se propone un marco general para el enfoque y la creación del cronograma del proyecto. El cronograma de este proyecto ha sido creado utilizando una herramienta de software libre que es Gantt Project.

En primer lugar, se ha dividido la realización del proyecto en paquetes de trabajo específicos para cada uno de los entregables y se han asignado las relaciones entre las actividades del proyecto.

1.6.1. Hitos del cronograma

En este apartado se ha detallado a que fase pertenece cada uno de los hitos del trabajo. En este caso, la realización de este proyecto se encuentra dividida en las siguientes fases:

- Fase 1: Plan de trabajo (*01/03/2023 – 14/03/2023*)
  - Entrega del plan de trabajo – *14/03/2023*
- Fase 2: Investigación (*15/03/2023 – 11/04/2023*)
  - Entrega de seguimiento 1 – *11/04/2023*
- Fase 3: Implementación (*12/04/2023 – 09/05/2023*)
  - Entrega de seguimiento 2 – *09/05/2023*
- Fase 4: Optimización y pruebas (*10/05/2023 – 13/06/2023*)
  - Entrega de la memoria final – *13/06/2023*

Después de la entrega de la memoria final hay periodo para la grabación de la presentación en vídeo (*14/06/2023 – 20/06/2023*), cuya entrega es el día *20/06/2023*.

A continuación, en el cuadro 1.2 se encuentran representados los hitos principales del proyecto. Para cada uno de ellos se indica la fecha límite y si requieren de la realización de una entrega.

Hito	Fecha tope	Entregable
Entrega del plan de trabajo	14/03/2023	Sí
Entrega de seguimiento 1	11/04/2023	Sí
Entrega de seguimiento 2	09/05/2023	Sí
Entrega de la memoria final	13/06/2023	Sí
Presentación en vídeo	20/06/2023	Sí

Cuadro 1.2: Hitos iniciales del cronograma

1.6.2. Definición de las actividades

En primer lugar, se ha comenzado dividiendo el proyecto en paquetes de trabajo para agrupar las actividades. Se ha definido un paquete de trabajo por cada una de las entregas que es necesario realizar en el transcurso del proyecto.

- PT-001: Plan de trabajo
- PT-002: Entrega de seguimiento 1
- PT-003: Entrega de seguimiento 2
- PT-004: Memoria final
- PT-005: Presentación en vídeo

Después de enumerar los distintos paquetes de trabajo que componen el proyecto, se ha procedido a identificar las actividades necesarias a la hora de cumplir con los objetivos definidos.

- PT-001: Plan de trabajo (Fase de Planificación)
  - ACT-001: Establecer los objetivos del proyecto
  - ACT-002: Elaborar la planificación temporal del proyecto
  - ACT-003: Redactar la documentación del plan de trabajo
- PT-002: Entrega de seguimiento 1 (Fase de Investigación)
  - ACT-004: Investigar y seleccionar la tecnología IDS
  - ACT-005: Investigar y seleccionar la tecnología de análisis de logs
  - ACT-006: Investigar y seleccionar la tecnología del sistema de alertas y notificaciones
  - ACT-007: Diseñar la topología de red para la implementación
  - ACT-008: Redactar el primer entregable de seguimiento
- PT-003: Entrega de seguimiento 2 (Fase de Implementación)
  - ACT-009: Configurar el dispositivo donde se instalará el IDS
  - ACT-010: Instalar y configurar el IDS
  - ACT-011: Instalar y configurar el sistema de análisis de logs
  - ACT-012: Instalar y configurar el sistema de alertas y notificaciones
  - ACT-013: Redactar el segundo entregable de seguimiento
- PT-004: Memoria final (Fase de Optimización y Pruebas)
  - ACT-014: Explorar posibles casos de uso para el IDS
  - ACT-015: Probar el funcionamiento completo del sistema

- ACT-016: Detectar y solucionar errores
- ACT-017: Redactar la memoria final

- PT-005: Presentación en vídeo
  - ACT-018: Preparar el caso de pruebas para la grabación
  - ACT-019: Grabar el vídeo de demostración

1.6.3. Estimación de las actividades

Antes de desarrollar el cronograma se ha sido necesario estimar la duración de las actividades, para ajustarse a la realidad en la medida de lo posible. Además, nos permite localizar aquellas actividades más costosas que requerirán una mayor cantidad de tiempo para ser completadas.

En el cuadro 1.3 se calcula una estimación de la duración esperada para cada una de las actividades utilizando la técnica de estimación por tres valores, representada por la formula 1.1.

$$\text{D. esperada} = \frac{\text{D. optimista} + (4 * \text{D. más probable}) + \text{D. pesimista}}{6} \tag{1.1}$$

Código	D. optimista (h)	D. más probable (h)	D. pesimista (h)	D. esperada (h)	Desviación estándar	Varianza
ACT-001	3	4	10	4,83	1,17	1,36
ACT-002	10	12	15	12,17	0,83	0,69
ACT-003	14	16	25	17,17	1,83	3,36
ACT-004	14	16	25	17,17	1,83	3,36
ACT-005	14	16	25	17,17	1,83	3,36
ACT-006	14	16	25	17,17	1,83	3,36
ACT-007	6	8	15	8,83	1,50	2,25
ACT-008	14	16	20	16,33	1,00	1,00
ACT-009	6	8	10	8,00	0,67	0,44
ACT-010	18	20	35	22,17	2,83	8,03
ACT-011	18	20	35	22,17	2,83	8,03
ACT-012	18	20	35	22,17	2,83	8,03
ACT-013	10	12	16	12,33	1,00	1,00
ACT-014	35	40	50	40,83	2,50	6,25
ACT-015	14	16	20	16,33	1,00	1,00
ACT-016	8	12	16	12,00	1,33	1,78
ACT-017	18	20	30	21,33	2,00	4,00
ACT-018	6	8	12	8,33	1,00	1,00
ACT-019	3	4	6	4,17	0,50	0,25
Total:	243	284	425	300,67	30,33	920,11

Cuadro 1.3: Estimación de las actividades

A la hora de construir el cronograma se ha utilizado la duración esperada que hemos calculado previamente con la técnica de estimación por tres valores que podemos encontrar en el cuadro 1.3. Con ello podemos concluir que la duración esperada del proyecto es de 300,67 horas, con una desviación estándar de 30,33 horas.

1.6.4. Cronograma

En el cronograma se ha procurado no solapar tareas en un mismo periodo de tiempo. Además, se ha tenido en cuenta un calendario de trabajo donde únicamente se puede puedan asignar tareas de lunes a viernes de 17:00 a 21:00, excluyendo las festividades y periodos no lectivos contemplados en la web de la UOC.

1.6.5. Diagrama de Gantt

En este apartado se muestran las diferentes actividades que se realizan a lo largo de las etapas del proyecto en forma de diagrama de Gantt. Para comenzar, en la figura 1.1 se muestra un cronograma

global de las fases del proyecto desde principio a fin. Después de mostrar el cronograma a grandes rasgos, se han extraído el cronograma para cada una de las fases.

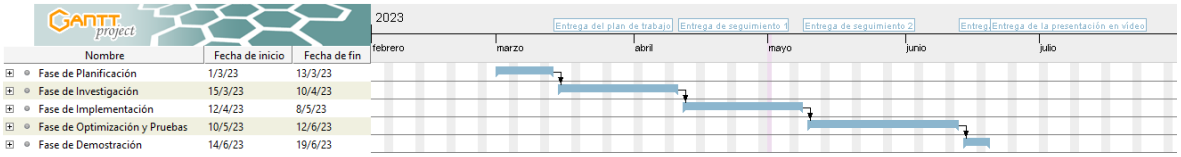

Figura 1.1: Cronograma de las fases del proyecto

En la figura 1.2, se muestra la distribución de las actividades a lo largo de la fase de planificación. Esta fase tiene como propósito la redacción del plan de proyecto.

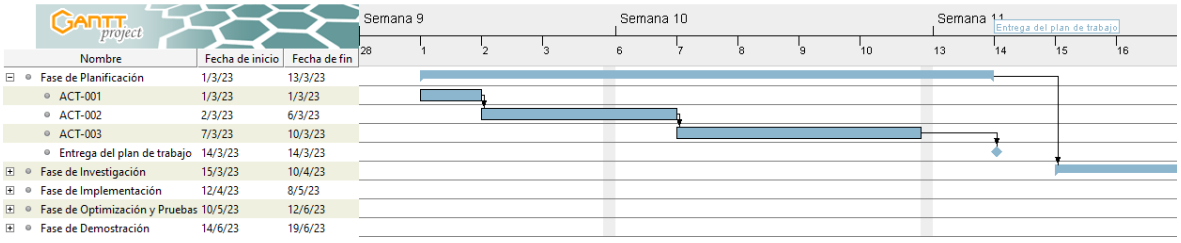

Figura 1.2: Cronograma de la fase de planificación

En la figura 1.3, se ha ampliado la parte del cronograma que correspondes con la primera fase del desarrollo des proyecto, que es la fase de investigación.

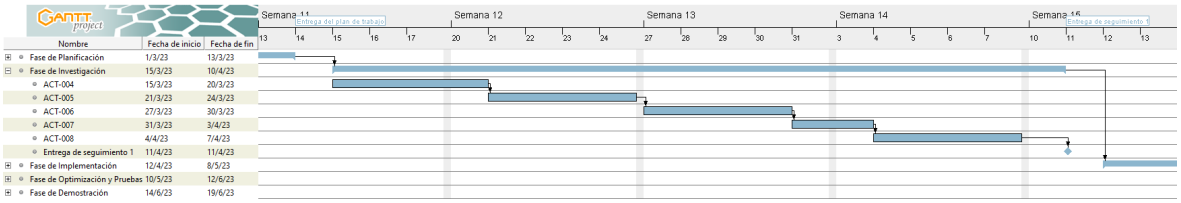

Figura 1.3: Cronograma de las fase de investigación

En la figura 1.4, se ha ampliado la parte del cronograma que corresponde con la segunda fase del desarrollo del proyecto, es decir, la fase de implementación.

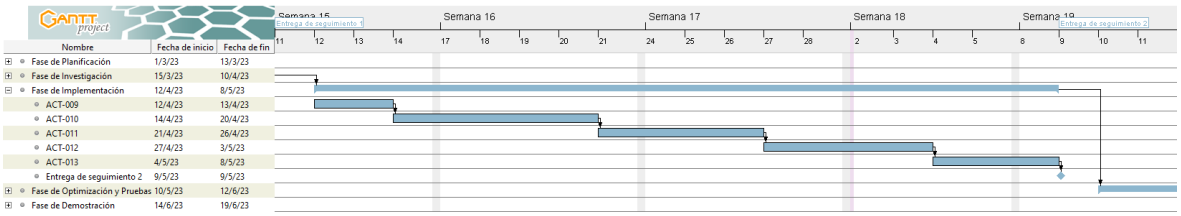

Figura 1.4: Cronograma de las fase de implementación

En la figura 1.5, se muestra el orden y la interdependencia entre las actividades de la última fase del desarrollo del proyecto, que es la fase de optimización y prueba. Esta fase termina con la entrega de versión final de la memoria final.

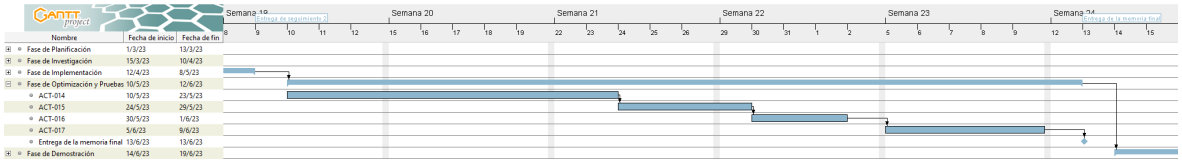

Figura 1.5: Cronograma de la fase de optimización y pruebas

En la figura 1.6, se muestra la etapa posterior a la entrega de la memoria final del TFM. En esta última etapa se encuentran las actividades relacionadas con la preparación y grabación del vídeo de demostración.

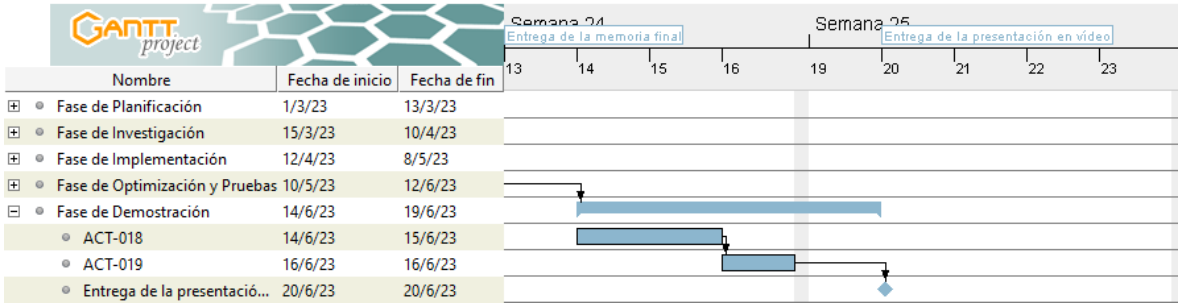

Figura 1.6: Cronograma de de la fase de demostración

1.7. Recursos y presupuesto del proyecto

En esta sección se realizará una estimación de los costes en base a los materiales y al personal (teniendo en cuenta la planificación temporal).

Para este proyecto, se ha consultado documentos oficiales como convenios recogidos en el BOE (Boletín Oficial del Estado). En éstos, estarán incluidos salarios de personal y otros costes necesarios.

En el caso que nos concierne, se ha tomado como referencia los costes de personal recogidos en el Convenio de las TIC (Tecnologías de la Información y la Comunicación) del Boletín Oficial del Estado. Para completar los cálculos, se han tenido en cuenta los gastos de Seguridad Social (el porcentaje se encuentra desglosado en el cuadro 1.4).

Tipo de coste	**Porcentaje**
Contingencias comunes	4,7 %
Desempleo	1,55 %
FOGASA	0,0 %
Formación profesional	0,1 %
Total:	6,35 %

Cuadro 1.4: Desglose de costes de la seguridad social

En cuanto a los recursos software se va a buscar que en la mayoría de las herramientas necesarias para montar el sistema descrito por este proyecto sea de uso libre y que no incurra en costes de licencias.

En cuanto a los recursos hardware, se va a utilizar el ordenar Raspberry Pi. Este dispositivo forma parte de una serie de computadores muy económicos. Este dispositivo es muy conocido por se utilizado para crear prototipos.

Además del ordenador Raspberry se deberá tener en cuenta la necesidad de contar con otros dispositivos periféricos, como puede ser una antena Wifi USB para conectarse a la red que se desea vigilar. Los costes asociados al material son 105,4 €.

Tipo de coste	**Coste por recurso**	**Cantidad**	**Coste**
Programador junior	11,92 €/h	301 h	3587,92 €
Amortización del equipo	7,38 €	-	7,38 €
Kit Raspberry Pi 4 Modelo B - 8GB	95.4 €	x1	95,4 €
Antena Wifi USB	10 €	x1	10 €
Total:			3.700,7 €

Cuadro 1.5: Desglose de costes iniciales

Una vez finalizada la estimación de costes se ha realizado un presupuesto. La suma de los costes será utilizada para establecer una línea base de costes y conformará el presupuesto inicial. Este presupuesto inicial puede variar según el desarrollo del proyecto. El presupuesto inicial se encuentra desglosado en el cuadro 1.5 y es de 3.700,7 €.

1.8. Análisis de riesgos

En esta sección se proporcionará un marco general para la detección de riesgos, así como el plan de prevención y resolución de riesgos. Estos riesgos pueden ser eventos tanto conocidos como desconocidos que pueden suceder en cualquier momento del desarrollo de este proyecto.

Se analizarán estos riesgos teniendo en cuanta tanto factores externos como los factores internos. Para medir el impacto de los riesgos, se hará uso de la matriz de riesgo. Esta herramienta de control nos proporcionará datos sobre los riesgos y un diagnóstico global del proyecto. Por otro lado, la matriz nos permite evaluar si la gestión de los riesgos está siendo efectiva.

1.8.1. Identificación de riesgos

A continuación, en el cuadro 1.6 se propone la matriz de evaluación de riesgo que se plantea para este proyecto, con la intención de medir objetivamente cuales son los riesgos más relevantes.

Probabilidad	**Consecuencias**				
	Insignificante	Menor	Moderado	Mayor	Catastrófico
Certero	Medio	Alto	Muy Alto	Muy Alto	Extremo
Probable	Medio	Medio	Alto	Muy Alto	Muy Alto
Moderado	Bajo	Medio	Medio	Alto	Muy Alto
Poco probable	Muy Bajo	Bajo	Medio	Medio	Alto
Inusual	Muy Bajo	Muy Bajo	Bajo	Medio	Extremo

Cuadro 1.6: Matriz de evaluación de riesgos

Una vez planteada la matriz de evaluación de riesgos, se ha pasado a la identificación de los tipos de riesgos que pueden surgir. En primer lugar, los podemos clasificar según su impacto sobre el proyecto:

- **Riesgos conocidos**: riesgos que han sido identificados y analizados, por lo que se puede plantear una acción preventiva a la situación. Esa lista se presenta tras estas definiciones con los planes de contingencia necesarios.
- **Riesgos desconocidos**: riesgos que han sido identificados y analizados. Por esta razón estos riesgos no se pueden tratar de forma proactiva.

También podemos clasificar los riesgos según sus consecuencias, estas pueden ser clasificadas como negativas o positivas.

- **Riesgos negativos**: estos riesgos son considerados amenazas para el proyecto. En estas situaciones se intentará que las consecuencias que suponen sean mitigadas lo máximo posible.
- **Riesgos positivos**: estos riesgos también pueden ser considerados como oportunidades.

Y por último también están aquellos según su procedencia, que pueden ser interna o externa.

- **Riesgos internos**: estos riesgos son considerados amenazas para el proyecto. En estas situaciones se intentará que las consecuencias que suponen sean mitigadas lo máximo posible.
- **Riesgos externos**: estos riesgos también se consideran oportunidades.

En el cuadro 1.7 se presentan aquellos riesgos que han sido identificados por el ponente, junto a sus consecuencias y su probabilidad.

Riesgo	**Consecuencias**	**Probabilidad**	**Importancia**
Imposibilidad de implementar un requisito	Catastrófico	Poco probable	Muy alta
Implementación inadecuada de un requisito	Moderado	Inusual	Media
Retraso en la finalización de una tarea	Menor	Poco probable	Media
Finalización de una tarea antes de lo planeado	Insignificante	Poco probable	Baja
Desconocimiento de las tecnologías	Menor	Moderado	Media
Avería de un recurso hardware	Moderado	Poco probable	Media
Limitación de la infraestructura objetivo	Mayor	Moderado	Alta

Cuadro 1.7: Riesgos identificados en el proyecto

1.8.2. Planes de contingencia

- Imposibilidad de implementar un requisito

  Si es imposible implementar una funcionalidad por limitaciones técnicos (no lo permite la tecnología elegida) se debe buscar una alternativa lo más parecida posible, o descartar ese requisito. Es posible que la segunda opción cause la cancelación del proyecto si se trata de una funcionalidad crítica.

- Implementación inadecuada de un requisito

  Cuando se ha implementado una funcionalidad y no actúa como debe o como se ha establecido en los objetivos del proyecto, se debe volver a implementar la funcionalidad para ajustarla a lo establecido en los objetivos. Esto supone un coste inevitable en tiempo.

- Retraso en la finalización de una tarea

  Cuando una tarea no ha sido finalizada en el plazo establecido, se debe reorganizar el cronograma dentro de los márgenes establecidos por la fecha límite de los hitos. En caso de que no sea posible realizar dicha reorganización, sería necesario invertir tiempo que no estaba contemplado en la planificación.

- Finalización de una tarea antes de lo planeado

  Este riesgo puede ser considerado como positivo, ya que no causa ningún impacto en el proyecto y puede aportar beneficios. Esto quiere decir que se dispone más tiempo para el resto de

las actividades planteadas si se llegan a terminar las actividades antes de lo esperado por la planificación.

- Desconocimiento de las tecnologías

  Debido a la inexperiencia en las herramientas utilizadas para el proyecto será necesario dedicar el tiempo necesario para aprender su funcionamiento y estudiar la complejidad de las herramientas elegidas antes de comenzar a implementarlas en la solución final.

- Avería de un recurso hardware

  Es posible que algún elemento hardware necesario para la elaboración del trabajo deje de funcionar en un momento dado, lo que impediría avanzar el proyecto. En este caso, se debe sustituir el equipo temporalmente (tomando prestado dicho material) o definitivamente (adquiriendo nuevo hardware). Si la imposibilidad del trabajo se extiende, es posible que se deba reorganizar el cronograma.

  Además, como medida preventiva, será necesario utilizar algún sistema de control de versiones para realizar copias de seguridad del trabajo realizado. De esta forma, si ocurre alguna avería, el retraso del trabajo no será excesivo.

- Limitaciones de la infraestructura objetivo

  Es posible que con el objetivo de elegir herramientas hardware de coste reducido para realizar la implementación, imposibiliten la ejecución del sistema de detección de intrusos. Este riesgo es relevante debido a que si no se detecta a tiempo puede considerar un retraso en la planificación debido a la necesidad de buscar otras alternativas de bajo coste que permitan la ejecución del sistema.

1.9. Implicaciones éticas y legales

De acuerdo con el Reglamento General de Protección de Datos [8] y la Ley Orgánica de Protección de Datos y Garantía de los Derechos Digitales [7], se define como información personal cualquier dato del usuario le pueda identificar como individuo, directa o indirectamente.

Para que un sistema de detección de intrusos detecte atacantes o usuarios malintencionados, es necesario capturar los datos de red y analizarlos para detectar dicho comportamiento sospechoso. Los paquetes de red pueden contener información que identifique a los dispositivos dentro de la red, para determinar el punto de origen de estos. En este aspecto, los datos de esta índole se adhieren a la definición proporcionada por el RGPD y la LOPDGDD para los datos de carácter personal.

Desde un punto de vista ético, el aspecto más relevante a la hora de utilizar un IDS será asegurar que antes de que el sistema empiece a registrar y analizar el tráfico de la red, los usuarios deben ser conscientes de que los datos de sus comunicaciones a través de la red pueden ser almacenadas temporalmente.

Para que el sistema almacene temporalmente información personal de los usuarios de la red. Se debe cumplir con los requisitos exigidos por el RGPD:

- Los usuarios deben poder informarse sobre que información es recogida por el sistema, para que es utilizada, durante que periodo de tiempo y donde será almacenada.
- El sistema debe incluir algún mecanismo que permite realizar el borrado de todos los datos personales de un usuario.
- Debe existir una vía de contacto entre usuario y el administrado de sistema de detección de intrusos de forma que el usuario pueda ejercer sus derechos respecto a sus datos personales.
- Se debe comunicar a los usuarios cualquier brecha de seguridad en un plazo de 72 desde que sucede el incidente de seguridad.

La solución propuesta en este proyecto requeriría incorporar los puntos mencionados anteriormente para poder ser aplicados en un entorno real. Aunque en el caso de este proyecto, no se aplicarán

los puntos indicados anteriormente debido a que se trata de una prueba de concepto que se ejecutará en un entorno controlado.

2. Fase de Investigación

En este capítulo se investigan los componentes que formarán parte de la solución implementada, desde el Sistema de Detección de Intrusos hasta la herramienta utilizada para analizar eventos generados por el IDS. El propósito de este capítulo es profundizar sobre la información recabada en la sección referente al *Estado del arte*.

El capítulo comienza con una comparación entre los diferentes diseños que se pueden realizar sobre la arquitectura de red a la hora de implementar el Sistema de Detección de Intrusos. Posteriormente, se realizar una investigación sobre las herramientas de código abierto más relevantes que podemos encontrar en el mercado. Para terminar, se realizará la elección de las herramientas que se utilizarán en la fase de implementación.

2.1. Sistemas de Detección de Intrusos

En esta sección se busca profundizar en el conocimiento respecto a los sistemas de detección de intrusos. En primer lugar, se ha definido una clasificación en función de diferentes factores. Más adelante, se ha realizado una comparación entre las dos posibles infraestructura de red que podría tener el sistema de detección de intrusos en red en función de su tipo de respuesta. Por último, se han comparado los tres sistemas de detección de intrusos en red que son más populares (Snort, Suricata y Zeek).

2.1.1. Taxonomía de los sistemas de detección de intrusos

Antes de nada, es necesario conocer como se clasifican los sistemas de detección de intrusos (IDS) para entender el tipo de sistema con el que estamos tratando en este proyecto. Esta clasificación puede ser definida en función de distintos criterios [16]:

- Según su fuente de información:
  - Basado en **máquina** (HIDS): Monitoriza y analiza la actividad de un determino terminal.
  - Basado en **red** (NIDS): Detectan ataques dentro de los paquetes capturados en la red.
- Según su análisis:
  - Uso **indebido**: Analiza el tráfico de red comparándolo con una serie de firmas previamente definidas.
  - **Anomalía**: Se encargan de buscar actividad sospechosa en el sistema. Pueden utilizar heurísticas y algoritmos de aprendizaje automático.
- Según su respuesta:
  - **Activa**: Tiene acciones automatizadas que se activan cuando ciertos tipos de intrusos son detectados.
  - **Pasiva**: Se notifica al responsable de seguridad.

La solución desarrollada en este proyecto pertenecería al conjunto de sistemas de detección de intrusos basados en red (NIDS). De esta forma, en los siguientes apartados se explorarán las herramientas disponibles para la implementación de un sistema de detección de intrusos basado en red y que consuma pocos recursos para que la solución sea económica.

2.1.2. Diseño de la arquitectura de red

En este apartado se ha profundizado en como debe estar estructurada la arquitectura de la red en función del tipo de respuesta del sistema de detección de intrusos en red. Así pues, un Sistema de Detección de Intrusos (IDS) se puede clasificar según su rol en la red, ya sea como un IDS en línea (respuesta activa) o un IDS fuera de línea (respuesta pasiva).

Diseño 1: IDS fuera de línea

Un IDS fuera de línea se coloca en una ruta de tráfico de red secundaria y monitorea el tráfico de red sin interactuar directamente con él. En este caso, el IDS no actúa como un gateway, sino como un observador. Aunque no puede bloquear el tráfico malicioso en tiempo real, puede generar alertas al administrador de la red y proporcionar información valiosa sobre la actividad de la red.

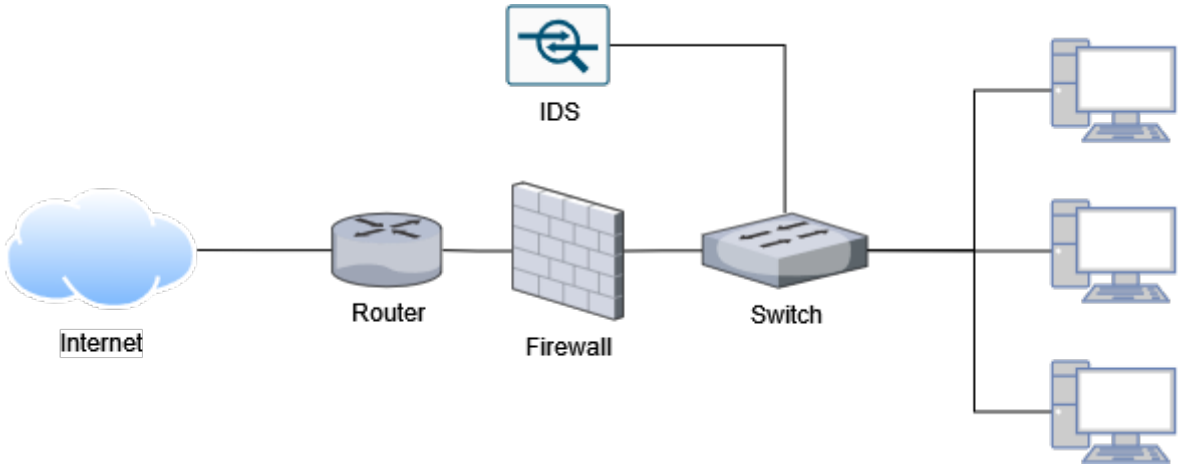

Figura 2.1: Arquitectura de red de un IDS

Diseño 2: IDS en línea

Un IDS en línea se coloca en la ruta del tráfico de red y puede actuar como un gateway. Esto significa que se encuentra en la misma ruta que los paquetes de red que atraviesan la red. Este tipo de IDS tiene la capacidad de bloquear el tráfico malicioso en tiempo real y puede alertar al administrador de la red sobre cualquier intento de intrusión. En este caso, el IDS actúa como una barrera de seguridad entre la red interna y la red externa.

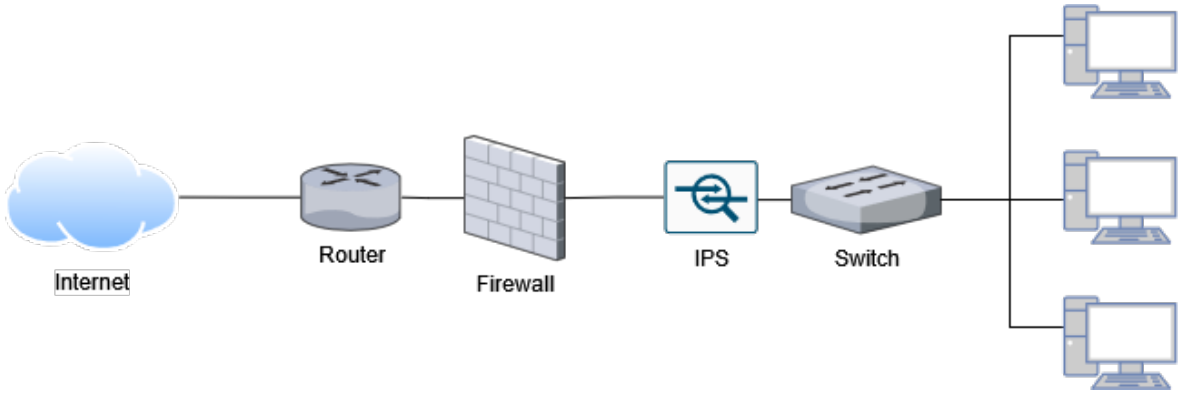

Figura 2.2: Arquitectura de red de un IPS

Diseño a implementar

En resumen, un IDS puede actuar como un gateway o simplemente como un observador de la red, dependiendo de su ubicación y función en la red. Un IDS en línea se coloca en la ruta del tráfico de red y actúa como una barrera de seguridad, mientras que un IDS fuera de línea monitorea el tráfico de red sin interactuar directamente con él.

Teniendo en cuenta esto, la solución a implementar consistirá en un IDS fuera de línea. La razón principal se debe a que uno de los requisitos principales es que la solución sea de bajo coste, mientras que un sistema en línea requeriría de mayor potencia de computo para procesar los paquetes de red en tiempo real y sin afectar al rendimiento de la red.

2.1.3. Evaluación de las tecnologías existentes

En este apartado se realizará una investigación sobre las herramientas de detección de intrusos de código abierto más relevantes que se pueden encontrar en el mercado. Las herramientas de código abierto han sido elegidas para la comparación son: Snort, Suricata y Zeek [22].

Snort

Snort se trata de un Sistema de Prevención de Intrusiones (IPS) de código abierto, siendo uno de los más utilizados en el mundo. El Snort IPS utiliza un conjunto de reglas que permiten definir lo que se considera como actividad de red maliciosa y utiliza esas reglas para encontrar paquetes que coincidan con ellas. De esta forma, en caso de que se encuentren paquetes que coincidan con las reglas se generarán alertes para los usuarios del sistema.

Como se ha comentado, Snort puede ser desplegado en línea, de forma que permita parar aquellos paquetes que puedan ser maliciosos. De la misma forma, esta herramienta puede ser utilizada como un Sistema de Detección de Intrusos.

Figura 2.3: Logotipo de la herramienta Snort

Snort fue desarrollado en 1998 y desde entonces ha tenido muchas actualizaciones y tiene una comunidad muy activa. Por otra parte, Snort carece de interfaz gráfica, por lo que necesitaría de herramientas que cubran esa carencia.

Suricata

Suricata consiste en un software de alto rendimiento utilizado para el análisis de redes y la detección de amenazas. Este IDS es utilizado por un gran número de organizaciones privada y publicas para la protección de sus recursos en red.

Figura 2.4: Logotipo de la herramienta Suricata

Zeek

Zeek es un sistema de detección de intrusos de código abierto que analiza el tráfico de la red de forma pasiva. Mucho operadores utilizan este software como un monitor de seguridad en red, con el propósito de apoyar en la investigación de actividades sospechosas o maliciosas. Zeek también posee otras herramientas más allá del dominio de la seguridad, como herramientas para medir el rendimiento y depurar la red.

Figura 2.5: Logotipo de la herramienta Zeek

El funcionamiento de Zeek es diferente a los sistemas presentados anteriormente. Este IDS en lugar de utilizar reglas para analizar el tráfico de red, utiliza scripts. Estos scrips dan mucha mayor flexibilidad a la hora de implementar filtro que detecten la actividad maliciosa. Por esta misma razón, se ha descartado esta tecnología, debido a su complejidad y su elevada curva de aprendizaje.

2.1.4. Comparación de rendimiento entre Snort y Suricata

Realizando una comparación del rendimiento entre Snort y Suricata se puede llegar a la conclusión de que Suricata es capaz de procesar mayores cantidades de datos de red [11, 17]. Por otra parte, se ha podido observar que Snort consume menos recursos a la hora de analizar la red.

En resumen, se ha decidido utilizar el sistema de detección de intrusos de Snort debido a que nuestro sistema requiere de software que consuma los menores recursos hardware posibles.

2.2. Sistemas de Análisis y Presentación de Datos

Uno de los mayores retos a la hora de implementar un sistema de detección de intrusos consiste en analizar los logs y reportes generados por las herramientas. Estos logs pueden contener un gran volumen de datos, por lo que una herramienta que sea capaz de almacenar y analizar los logs de forma eficaz es esencial. Además, este tipo de herramientas pueden realizar ayudar a realizar consultas rápidas sobre el conjunto de dato y realizar agregaciones de datos para identificar la causa de las alertas en el sistema [10].

2.2.1. ELK Stack

La pila de herramientas ELK [3] consiste en un conjunto de tres herramientas - Elasticsearch, Logstash y Kibana (figura 2.6). Aunque estas herramientas pueden funcionar por independiente, han sido diseñadas para ser utilizadas de forma conjunta. Más adelante, se agregó la herramienta Beats, que permite enviar datos desde los dispositivos finales a Logstash.

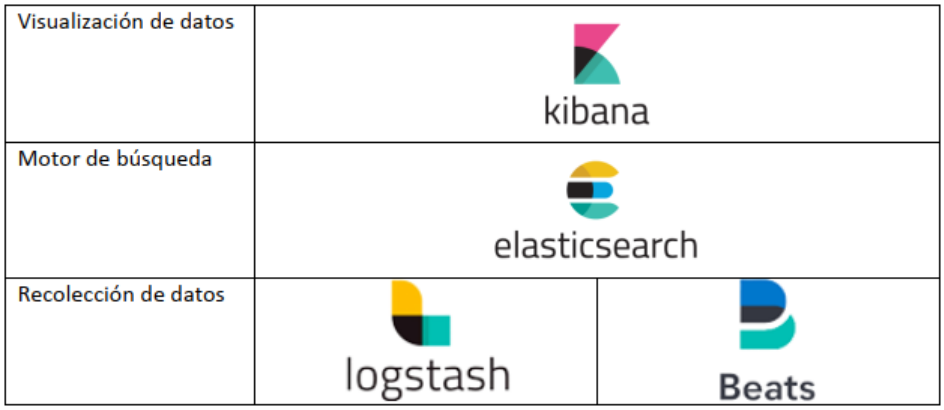

Figura 2.6: Pila de aplicaciones ELK

Elasticsearch

Elasticsearch es un motor de búsqueda de código abierto, distribuido y altamente escalable. Este motor de búsqueda permite indexar y analizar grandes cantidades de datos en tiempo real. Se trata de una base de datos de tipo NoSQL, que permite realizar búsquedas complejas y aplicar operaciones estadísticas sobre el conjunto de datos utilizando el lenguaje Query DSL (Domain Specific Language).

Logstash

Logstash es un herramienta de direccionamiento de eventos basada en plugins. Esta herramienta puede obtener datos desde distintas fuentes, transformarlos y entregarlos en alguna otra parte.

Kibana

Es la herramienta de visualización para Elasticsearch. Esta plataforma ofrece una interfaz web para la búsqueda, visualización y análisis de los datos almacenados en el cluster de Elastersearch.

En la herramienta de Kibana, podemos encontrar las siguientes funcionalidades:

- El Dashboard permite combinar múltiples resultados para visualizarlos en una sola vez.
- El Dashboard permite un gran nivel de personalización.
- Debido a que Kibana tiene una arquitectura basada en plugins, sus funcionalidades pueden ser fácilmente ampliadas para satisfacer necesidades concretas.

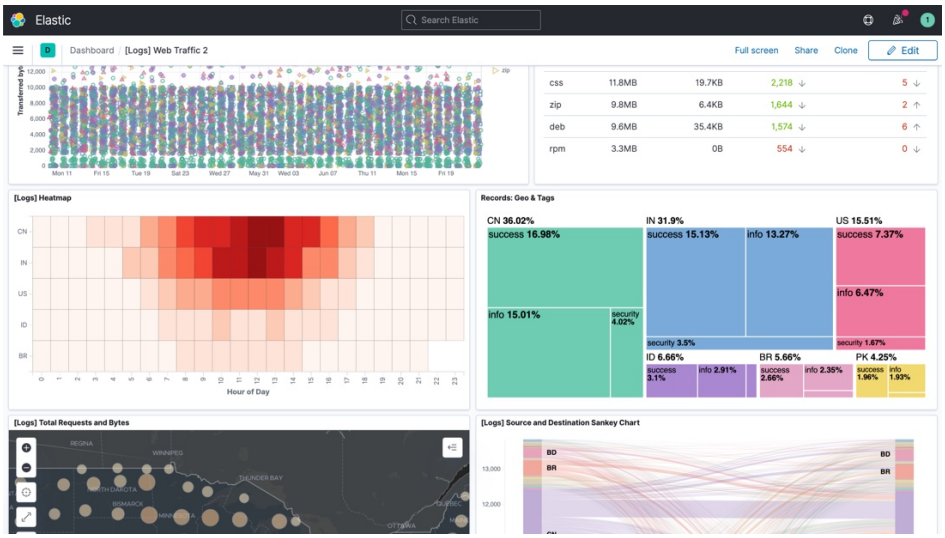

Figura 2.7: Ejemplo de los elementos visuales de Kibana

2.2.2. Alternativas a ELK Stack

Aunque ya se haya decidido el uso de la herramienta de análisis de logs, en este apartado se enumeran algunas de las alternativas que podrían sustituir a la pila ELK [21].

En primer lugar, una de las posibles alternativas a la pila de herrmientas ELK es Splunk. Esta herramienta, también es conocida por ser comprensiva y tener una amplia variedad de características. La desventaja es que la herramienta no es de código abierto.

Otra alternativa de código abierto puede ser Graylog. Esta herramienta utilizada Elasticsearch para el almacenamiento de datos y MongoDB para el almacenamiento de los metadatos.

Algunas de las alternativas basadas en la nube pueden ser Loggly, Sumo Logic y PaperTrails.

2.3. Elección de las tecnologías

En este apartado se ha definido que tecnologías y herramientas se han elegido para los diferentes componentes que será necesario implementar.

En primer lugar, en este proyecto se ha explorado la alternativa de utilizar un sistema de detección de intrusos en red donde el dispositivo sea un participante más en la red. Sin embargo, Debido a las dificultades que se plantean en el tipo de implementación antes mencionada, se realizará una implementación donde el dispositivo tenga la función de punto de acceso inalámbrico. De esta forma las comunicaciones de la red tienen que pasar a través del sistema de detección de intrusos.

En segundo lugar, como sistema de detección de intrusos seleccionado se va a utilizar la herramienta Snort. Aunque Suricata tiene un mayor rendimiento, Snort ha demostrado consumir una menor cantidad de recursos, lo que es relevante para este proyecto.

Por último, en cuanto al sistema de análisis y visualización de eventos, se ha seleccionado la pila original de ELK, es decir, Elasticsearch, Logstash y Kibana. Se ha elegido esta pila de herramientas principalmente por la cantidad de flexibilidad que ofrecen en su configuración. Además, al haber sido desarrolladas como una solución integral, la configuración de los componentes de la pila será más fácil. Por otra parte, Kibana ofrece una gran cantidad de elementos visuales para la representación y agregación de datos.

En la figura 2.8 se muestra la configuración que tendrá los dispositivos de la red. La red estará formada por los siguiente componentes:

- **Red 1**: Bloque de direcciones IP "192.168.20.0/24". Esta red se trata de la red principal del hogar, cuyo router se encuentra conectado con el ISP. Esta red contendrá un equipo personal con IP "192.168.20.50" que se encargará de hospedar la pila de aplicaciones ELK (Elasticsearch, Logstash y Kibana).
- **Red 2**: Bloque de direcciones IP "192.168.25.0/24". Esta red consiste en la red inalámbrica que es hospedada por el dispositivo Raspberry Pi. Cabe destacar que será necesario configurar el dispositivo como un router que permita las conexiones a Internet. Es el dispositivo Raspberry Pi donde se instalará la herramienta Snort y la herramienta de recopilación de datos Filebeat.

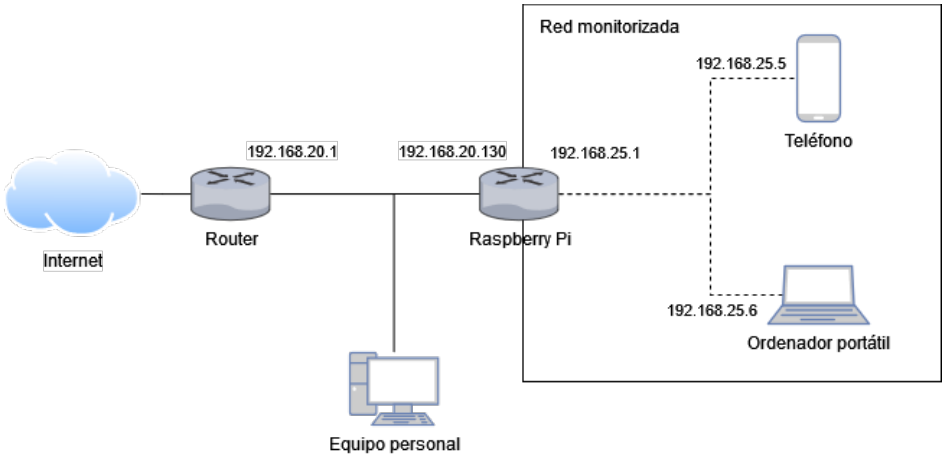

Figura 2.8: Topología de la red implementada

3. Fase de Implementación

En este capítulo se implementa de forma completa la solución propuesta para este trabajo a partir de las herramientas seleccionadas en la *Fase de Investigación*.

El capítulo comienza explicando detalladamente como instalar el Sistema de Detección de Intrusos (IDS) y las herramientas de recopilación de datos. Posteriormente, se instalará la pila de aplicaciones ELK en un equipo de escritorio para el almacenamiento y análisis de los logs generados. Finalmente, se mostrará como implementar los modelos de IDS seleccionados para este proyecto.

3.1. Instalación de Snort en Raspberry Pi con Debian

En este apartado se describe el proceso de instalación de Snort en un dispositivo Raspberry Pi con el sistema operativo Debian. Se utilizará una imagen personalizada creada con *CustomPiOS* para facilitar el proceso de instalación y configuración de las herramientas necesarias. Además, se abordará la instalación de la imagen en la tarjeta microSD, así como la configuración de Snort 3, Filebeat y Metricbeat en el Raspberry Pi.

3.1.1. Creación de la imagen personalizada para Raspberry Pi

Gran parte de los componentes software que van a ser instalados en el dispositivo Raspberry Pi no se encuentran en el repositorio oficial de Debian, por lo que será necesario compilarlos en el dispositivo en el que van a ser utilizados.

En el caso que concierne a este proyecto, los recursos hardware que posee el dispositivo objetivo son limitados. Esto nos lleva a buscar herramientas que permitan precompilar el código fuente de las dependencias y las aplicaciones para que en lugar de utilizar los recursos del Raspberry Pi, utilicemos los recursos de un equipo de mayor potencia para acelerar el proceso de instalación.

La herramienta utilizada para realizar una configuración preliminar del entorno ha sido **CustomPiOS** [18]. Esta herramienta permite abrir una imagen existente, modificarla y reempaquetar la imagen para poder flashearla sobre la tarjeta microSD.

Configuración del proyecto

Este subapartado consiste en clonar el proyecto CustomPiOS y en realizar los pasos necesarios antes de comenzar a preparar el script encargado de instalar las dependencias en el proceso de modificación de la imagen.

En este caso, todo el proceso de personalización de la imagen Debian se va a realizar sobre WSL en un equipo con sistema operativo Windows 11.

1. Instalar requisitos (extracto de código 3.1): En primer paso consiste en instalar las dependencias que permitirán ejecutar el proyecto y así simular el entorno sobre el que se instalarán las aplicaciones.

```
sudo apt install sudo apt install qemu-user-static bash jq git file
   sudo p7zip-full python3
```

Extracto de código 3.1: Instalación de dependencias para CustomPiOS

2. Clonamos el repositorio desde GitHub (extracto de código 3.2). Una vez clonado accedemos al directorio.

```
git clone https://github.com/guysoft/CustomPiOS.git
```

Extracto de código 3.2: Instalación de dependencias para CustomPiOS

3. Creamos el entorno de trabajo sobre el que se modificará la imagen de Debian (extracto de código 3.3). La opción "-v" nos permite elegir la arquitectura de Debian que deseamos utilizar, en este caso se ha elegido la versión de 64 bit. Por otra parte, la opción "g" indica que el proyecto debe descargarse su propia imagen (una alternativa seria utilizar una imagen propia). Por último, "`custom_distro`" es de la distribución que se va a modificar.

```
./src/make_custom_pi_os -g -v raspios_lite_arm64 custom_distro
```

Extracto de código 3.3: Creación de la distribución personalizada

4. Accedemos al directorio "`custom_distro/src`" para continuar con la configuración.
5. Editamos el archivo de configuración. 3.4

```
export DIST_NAME=custom_distro
export DIST_VERSION=0.1.0

# rpi-imager json generator settings
export RPI_IMAGER_NAME="${DIST_NAME}"
export RPI_IMAGER_DESCRIPTION="A raspberrypi distro built with
    CustomPiOS"
export RPI_IMAGER_WEBSITE="https://github.com/guysoft/CustomPiOS"
export RPI_IMAGER_ICON="https://raw.githubusercontent.com/guysoft/
    CustomPiOS/devel/media/rpi-imager-CustomPiOS.png"

export MODULES="base(network,custom_distro)"

export BASE_ARCH=arm64
export BASE_IMAGE_ENLARGEROOT=6000
export BASE_IMAGE_RESIZEROOT=20
```

Extracto de código 3.4: Configuración del proyecto CustomPiOS

Estas variables son parte del archivo de configuración utilizado en el proceso de personalización de la imagen de CustomPiOS. A continuación, se explica las variables que juegan un papel principal en la personalización de la imagen de este proyecto:

- "`BASE_ARCH=arm64`": Esta variable establece la arquitectura base de la imagen. En este caso, se ha configurado como "arm64", lo que indica que la imagen está destinada a la arquitectura ARM de 64 bits.
- "`BASE_IMAGE_ENLARGEROOT=6000`": Esta variable determina el tamaño en megabytes (MB) del sistema de archivos raíz (root) en la imagen base. En este caso, se ha configurado como 6000 MB, lo que significa que el tamaño del sistema de archivos raíz se ampliará a 6000 MB en la imagen personalizada.
- "`BASE_IMAGE_RESIZEROOT=20`": Esta variable establece el porcentaje en el que se redimensionará el sistema de archivos raíz (root) en la imagen personalizada. En este caso, se ha configurado como 20, lo que indica que el sistema de archivos raíz se redimensionará al 20 % del tamaño total de la imagen.

Modificación de la imagen

El archivo "`start_chroot_script`" del proyecto CustomPiOS, que se encuentra en el directorio "`modules/custom_distro`" contendrá una serie de fragmentos de script que se utilizan para instalar y

configurar las diversas herramientas que se utilizarán en el entorno Raspberry Pi y son computacionalmente costosas de compilar.

Los scripts detallados en los siguientes puntos están diseñados para automatizar el proceso de preparación del entorno, la instalación de compiladores y bibliotecas, así como la configuración de las herramientas de recopilación de datos de ELK [5], las dependencias de Snort 3 [24] (Hyperscan, gperftools y LibDAQ). Además, se instala la herramienta zram-swap, que permitirá al sistema utilizar la memoria RAM con mayor eficiencia.

1. **Herramientas de recopilación de datos**: El extracto de código 3.5 muestra un script para instalar las herramientas de ELK encargadas de la recopilación de datos, es decir, Filebeat y Metricbeat.

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo
    gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg
apt-get install -y apt-transport-https
echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg]
    https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo
    tee /etc/apt/sources.list.d/elastic-8.x.list

apt-get update
apt-get install -y filebeat metricbeat

# Enabling Filebeat and Metricbeat services on boot
systemctl enable filebeat
systemctl enable metricbeat
```

Extracto de código 3.5: Instalación de las herramientas de recopilación de datos

2. **Herramientas de compilación y dependencias de Snort 3**: En el extracto de código 3.6 se proporciona un script para instalar varias herramientas de compilación, como Git, Autoconf, CMake y GCC. Estas herramientas son necesarias para compilar software a partir de su código fuente.

```
apt-get install -y git autoconf cmake gcc libtool
apt-get install -y build-essential autotools-dev libdumbnet-dev
    libluajit-5.1-dev libpcap-dev zlib1g-dev pkg-config libhwloc-dev
    cmake liblzma-dev openssl libssl-dev cpputest libsqlite3-dev
    libtool uuid-dev git autoconf bison flex libcmocka-dev
    libnetfilter-queue-dev libunwind-dev libmnl-dev ethtool
    libjemalloc-dev libpcre++-dev
```

Extracto de código 3.6: Instalación de las herramientas de compilación

3. **Dependencia Hyperscan**: en el extracto de código 3.7 se muestra un script con todos los pasos de la instalación de la dependencia Hyperscan, una biblioteca de análisis de patrones de alto rendimiento. El script descarga y compila las dependencias necesarias, como Boost y Colm, antes de compilar e instalar Hyperscan.

```
cd /tmp
apt-get install -y ragel sqlite3 python3 asciidoc
wget http://www.colm.net/files/colm/colm-0.14.7.tar.gz
tar -xzvf colm-0.14.7.tar.gz
cd colm-0.14.7

./configure
make -j$(nproc)
make -j$(nproc) install
ldconfig
```

```
cd /tmp
wget https://boostorg.jfrog.io/artifactory/main/release/1.73.0/source
    /boost_1_73_0.tar.gz
tar -xzvf boost_1_73_0.tar.gz
#git clone https://github.com/intel/hyperscan.git
git clone https://github.com/kunpengcompute/hyperscan.git
mkdir hs-build && cd hs-build

cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/usr/local -
    DBOOST_ROOT=../boost_1_73_0 ../hyperscan

make -j$(nproc)
make -j$(nproc) install
```

Extracto de código 3.7: Instalación de Hyperscan

4. **Dependencia gperftools**: En el extracto de código 3.8 se muestra como instalar gperftools, una colección de herramientas de rendimiento desarrolladas por Google. Estas herramientas incluyen el analizador de perfiles CPU (pprof) y el analizador de memoria (heapprof). El script clona el repositorio de gperftools desde GitHub, lo compila e instala.

```
cd /tmp
git clone https://github.com/gperftools/gperftools.git
cd gperftools

./autogen.sh
./configure
make -j$(nproc)
make -j$(nproc) install
ldconfig
```

Extracto de código 3.8: Instalación de gperftools

5. **Dependencia LibDAQ**: El extracto de código 3.9 muestra como instalar LibDAQ, una biblioteca utilizada por Snort para la captura de paquetes de red. El script clona el repositorio de LibDAQ, lo configura y lo compila antes de instalarlo en el sistema.

```
cd /tmp
git clone https://github.com/snort3/libdaq.git
cd libdaq

./bootstrap
./configure
make -j$(nproc)
make -j$(nproc) install
ldconfig
```

Extracto de código 3.9: Instalación de LibDAQ

6. **Instalación de Snort 3**: En el extracto de código 3.10 se proporciona un script para instalar Snort 3, un sistema de detección y prevención de intrusos en red. El script clona el repositorio de Snort 3 desde GitHubm configura y compila el proyecto utilizando CMake, para finalmente instalarlo en el sistema.

```
cd /tmp
git clone https://github.com/snort3/snort3.git
cd snort3

export CFLAGS="-O3"
```

```
export CXXFLAGS="-O3 -fno-rtti"
./configure_cmake.sh --prefix=/usr/local --enable-tcmalloc
cd build
make -j$(nproc)
sudo make -j$(nproc) install
ldconfig
```

Extracto de código 3.10: Instalación de Snort 3

7. **Configuración Snort 3**: En el extracto de código 3.11 se muestran algunas configuraciones básicas para Snort 3, como la creación de directorios de reglas y logs, la creación de un grupo y un usuario para Snort, y la asignación de permisos adecuados al directorio de logs.

```
mkdir -p /usr/local/etc/snort/{builtin_rules,rules,appid,intel}
groupadd snort
useradd snort -r -M -g snort -s /sbin/nologin -c
    SNORT_SERVICE_ACCOUNT
mkdir /var/log/snort
chmod -R 5700 /var/log/snort
chown -R snort:snort /var/log/snort
```

Extracto de código 3.11: Configuración básica de Snort 3

8. **Optimización utilizando Zram**: En el extracto de código 3.12 se muestra cómo instalar Zram, una tecnología de compresión y descompresión en la memoria RAM utilizada para crear dispositivos de almacenamiento de bloques comprimidos en el kernel Linux. El script clona un repositorio de GitHub que proporciona un script de instalación para Zram y también realiza algunos ajustes en las configuraciones del sistema.

```
cd /tmp
git clone https://github.com/foundObjects/zram-swap.git
cd zram-swap && ./install.sh
echo "vm.vfs_cache_pressure=500" >> /etc/sysctl.conf
echo "vm.swappiness=100" >> /etc/sysctl.conf
echo "vm.dirty_background_ratio=1" >> /etc/sysctl.conf
echo "vm.dirty_ratio=50" >> /etc/sysctl.conf
```

Extracto de código 3.12: Instalación de Zram

Reempaquetado de la imagen

Una vez se ha configurado el archivo de configuración de la imagen personalizada "`custom_distrom`" y se ha modificado el script de inicialización de la imagen modificada, el único paso que quedaría es reempaquetar la imagen para poder ser flasheada en la tarjeta microSD.

Para ello vamos a volver al directorio raíz del proyecto CustomPiOS y vamos a ejecutar la instrucción que se muestra en el extracto de código 3.13.

```
sudo ./custom_distro/src/build_dist
```

Extracto de código 3.13: Reempaquetado de la imagen modificada

Una vez generada la imagen, la podremos encontrar en el directorio "`custom_distro/src/workspace`".

3.1.2. Instalación de la imagen en la tarjeta microSD

El primer paso es instalar el sistema operativo que se va a utilizar en la Raspberry Pi en una tarjeta microSD. Es este caso se ha seleccionado el sistema operativo Respberry OS Lite, que consiste

en una imagen mínima basada en la última versión de Debian. Tal y como se ha podido observar el el apartado anterior, se ha generado una imagen modificada que deriva de la imagen Debian previamente mencionada.

Para realizar dicha instalación se ha utilizado el software Raspberry Pi Imager para flashear la tarjeta microSD. Esta elección se ha realizado principalmente por las facilidades que ofrece a la hora de realizar configuraciones iniciales. Algunas de las configuraciones destacadas son: la conexión a la red inalámbrica, el servicio SSH, etc.

El proceso de instalación consta de los siguientes pasos [6]:

1. Descargar el software Raspberry Pi Imager:

   Lo primero que debemos hacer es descargar Raspberry Pi Imager desde el sitio web oficial [14]. Una vez completada la instalación y abierta la aplicación, encontraremos una ventana similar a la que se muestra en la figura 3.1.

Figura 3.1: Ventana principal de Raspberry Pi Imager

2. Conectar la tarjeta microSD al ordenador.
3. Abrir Raspberry Pi Imager:

   Si aun no lo hemos hecho, abrimos Raspberry Pi Imager en el ordenador, teniendo en cuenta que si estamos utilizando Windows, es posible que debamos ejecutar el programa como administrador.

4. Seleccionar la imagen de Debian:

   En Raspberry Pi Imager, pulsamos en "Choose OS" (Elegir sistema operativo) y seleccionamos "Use Custom". Luego, navegamos hasta el directorio donde se encuentra la imagen modificada y la seleccionamos.

5. Seleccionar la tarjeta microSD:

   En Raspberry Pi Imager, pulsamos en "Choose SD Card" (Elegir tarjeta SD) y seleccionamos la tarjeta microSD que acabas de insertar en el lector de tarjetas de tu ordenador.

6. Configuración inicial de la imagen Debian:

   En Rasberry Pi Imager, antes de comenzar a escribir la imagen sobre la tarjeta microSD, accedemos al menú de opciones avanzadas, que se encuentra en la esquina inferior derecha.

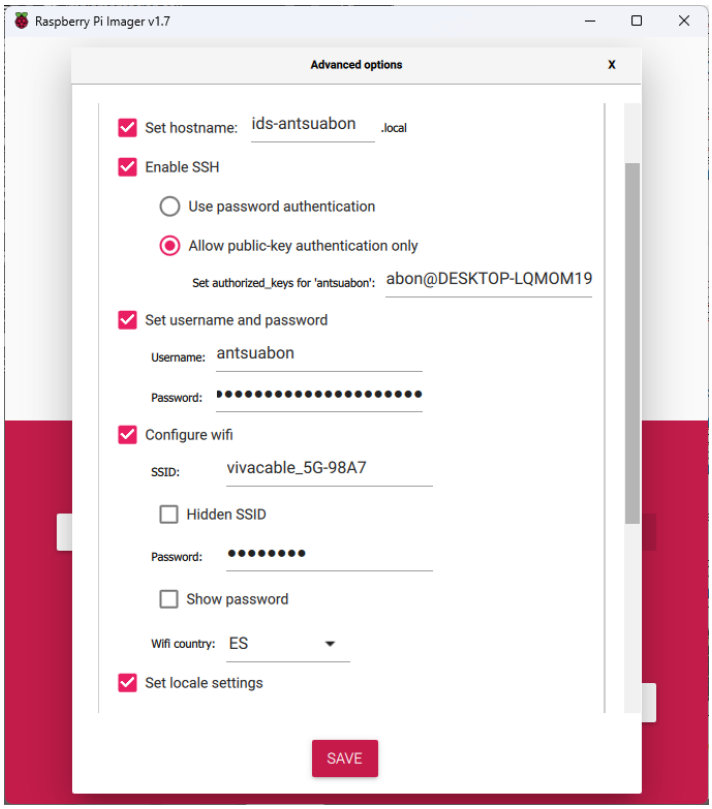

Figura 3.2: Configuración de Raspberry Pi Imager

Configuración más relevante:

- **Hostname**: ids-antsuabon.local

  Esta configuración, nos permite localizar el dispositivo en la red sin necesidad de conocer su dirección IP.

- **SSH**: Se ha configurado el usuario "antsuabon", agregando la clave publica del usuario ssh local para poder acceder a la Raspberry Pi sin necesidad de aportar la contraseña.

  Esta configuración nos permitirá conectarnos a la Raspberry Pi sin necesidad de hacer uso de ninguna pantalla. Esto nos permitirá continuar con las configuraciones de los siguientes apartados.

  Para la generación de la clave pública se ha utilizado el comando "ssh-keygen". Una vez realizado este paso, es posible acceder a la clave generada en el directorio "∼/.ssh".

- **Red inalámbrica**: Se han configurado las credenciales de la red inalámbrica.

- **Zona horaria**: Se ha configurado la región, el teclado y el idioma por si fuera necesario en el futuro.

- Se ha habilitado la opción que permite saltarse la configuración inicial en el primer inicio de sesión.

7. Flashear la imagen de Debian:

   Una realizada la configuración, pulsamos en "Write" (Escribir) para flashear la imagen de Debian en la tarjeta microSD. Este proceso puede tardar varios minutos, dependiendo de la velocidad de tu ordenador y de la tarjeta microSD.

8. Instalar la tarjeta microSD en la Raspberry Pi:

Una vez que Raspberry Pi Imager haya terminado de flashear la imagen de Debian en la tarjeta microSD, retiramos la tarjeta microSD del lector de tarjetas del ordenador y la colocamos en la ranura para tarjetas microSD de la Raspberry Pi.

9. Encender la Raspberry Pi:

   Conectamos la Raspberry Pi a una fuente de alimentación y la encendemos. Pasados unos minutos, podremos acceder a la Raspberry Pi por el protocolo SSH. De esta forma, podremos continuar con las configuraciones necesarias para hacer funcionar el IDS y que los eventos sean enviados correctamente a la herramienta de análisis de logs.

Al final de esta sección, ya disponemos de una Raspberry Pi conectada a una red inalámbrica local, a la que podemos acceder de forma remota para hacer todas las configuraciones oportunas.

3.1.3. Configuración de las herramientas en el Raspberry Pi

Para evitar problemas a la hora de comunicar el IDS con la pila de aplicaciones ELK. Se ha configurado nuestro equipo para que tenga la IP estática "192.168.29.50".

Por otra parte, se ha configurado un alias para el equipo donde serán procesados los logs. En el extracto de código 3.14, se muentra el contenido del fichero "etc/hosts" de la Raspberry Pi, incluyendo el alias de nuestro equipo.

```
127.0.0.1       localhost
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

127.0.1.1       ids-antsuabon
192.168.20.50   workstation-antsuabon
```

Extracto de código 3.14: Contenido del fichero /etc/hosts

Configuración de Snort 3

En este apartado se muestra como se ha configurado Snort en el dispositivo Raspberry Pi [24]. Una vez configurado el IDS el único paso que faltaría es crear las reglas, cuya tarea se realizará en la fase de optimización y pruebas.

Para realizar la configuración, se han seguido los siguientes pasos:

1. Se ha creado un fichero de reglas en la ruta "/usr/local/etc/snort/rules/local.rules". Este fichero será utilizado más adelante para indicar a Snort que reglas queremos aplicar.
2. Cambiar en el fichero de configuración "/usr/local/etc/snort/snort.lua" la configuración de la red local, tal y como se observa en el extracto de código 3.15.

```
-- HOME_NET and EXTERNAL_NET must be set now
-- setup the network addresses you are protecting
HOME_NET = '192.168.25.0/24'

-- set up the external network addresses.
-- (leave as "any" in most situations)
EXTERNAL_NET = 'any'
```

Extracto de código 3.15: Configuración de la variable de red en Snort

3. Configurar la ruta donde de donde se van a importar las reglas (extracto de código 3.16). Esta configuración también es realizada en el fichero "snort.lua".

```
ips =
{
    include = 'rules/local.rules',
    variables = default_variables
}
```

Extracto de código 3.16: Configuración del fichero de reglas

4. Configuración del formato de salida para las alertas: En el archivo "snort.lua", se definieron dos formatos de alertas: "`alert_fast`" y "`alert_json`". El formato "`alert_fast`" guarda las alertas en un archivo, mientras que el formato "`alert_json`" guarda las alertas en un archivo con formato JSON (figura 3.17).

```
-- event logging
alert_fast = { file = true }
alert_json =
{
    file = true,
    limit = 100,
    fields = 'timestamp iface src_addr src_port dst_addr dst_port
       proto action msg priority class sid'
}
```

Extracto de código 3.17: Configuración del tipo de alertas

5. Creación de un fichero "`custom_tweaks.lua`": En este archivo se habilitó el uso de Hyperscan para la inspección de IPS, AppID y HTTP, así como para las coincidencias de expresiones regulares (PCRE) (extracto de código 3.18).

```
-- Enable hyperscan for IPS, AppID and HTTP inspection
-- Enable hyperscan for pcre/regex matches
search_engine = { search_method = "hyperscan" }
detection = { hyperscan_literals = true, pcre_to_regex = true }
```

Extracto de código 3.18: Configuración de las optimizaciones de Snort

6. Creación del servicio que ejecute Snort con su configuración automáticamente en el arranque (extracto de código 3.19). Este archivo de servicio define cómo se ejecuta Snort, qué archivo de configuración utiliza y qué archivo de ajustes personalizados utiliza. También especifica el usuario y grupo bajo los cuales se ejecutará Snort, así como las opciones de reinicio en caso de fallo.

```
[Unit]
Description=Snort 3 Intrusion Detection and Prevention service
After=syslog.target network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/snort -c /usr/local/etc/snort/snort.lua --
   plugin-path /usr/local/etc/snort/extra --tweaks /usr>ExecReload=/
   bin/kill -9 $MAINPID
User=snort
Group=snort
Restart=on-failure
RestartSec=5s
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_RAW CAP_IPC_LOCK
AmbientCapabilities=CAP_NET_ADMIN CAP_NET_RAW CAP_IPC_LOCK

[Install]
```

```
WantedBy=multi-user.target
```

Extracto de código 3.19: Configuración del servicio de Snort

Configuración de Filebeat

El archivo que contiene la configuración de Filebeat es "/etc/filebeat/filebeat.yml". Para que la configuración funcione correctamente es necesario comprobar que las propiedades descritas en el extracto de código 3.20 se encuentren presentes.

Este archivo de configuración de Filebeat indica que hay un solo input o entrada, que es un tipo de registro o log (type: log) y se encuentra habilitado (enabled: true). El archivo de log que se va a recopilar se encuentra en la ruta /var/log/snort (paths: - /var/log/snort), que es el archivo de registro generado por Snort, el sistema de detección de intrusos que ha sido seleccionado para la elaboración del proyecto.

El output o salida se envía a Elasticsearch (output.elasticsearch:), que es un motor de búsqueda y análisis distribuido, utilizado para la búsqueda y análisis de grandes cantidades de datos. El host de Elasticsearch es workstation-antsuabon (equipo donde se encuentra la pila ELK instalada) y el puerto utilizado es el predeterminado de Elasticsearch, 9200 (hosts: ["workstation-antsuabon:9200"]).

```
filebeat.inputs:
- type: filestream
  enabled: true
  paths:
    - /var/log/snort/alert_json.txt
  parsers:
    - ndjson:
      overwrite_keys: true
      add_error_key: true
      expand_keys: true

output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["workstation-antsuabon:9200"]
  protocol: "https"

  # Public Key Infrastructure (PKI) certificates
  ssl.certificate_authorities:
    - /etc/pki/ca.crt
  ssl.certificate: "/etc/pki/ids-antsuabon.crt"
  ssl.key: "/etc/pki/ids-antsuabon.key"

  # Authentication credentials
  username: "elastic"
  password: "elastic-password"
```

Extracto de código 3.20: Configuración de Filebeat

Además, se han realizado las configuraciones necesarias para utilizar el protocolo HTTPS y una autenticación basada en certificados (estos certificados serán generados posteriormente cuando se configuren los elementos principales de la pila ELK).

En resumen, este archivo de configuración de Filebeat es utilizado para recopilar el archivo de registro generado por Snort y enviarlo a Elasticsearch para su posterior análisis y búsqueda.

Configuración de Metricbeat

El archivo que contiene la configuración de Metricbeat es "/etc/metricbeat/metricbeat.yml". Para que la configuración funcione correctamente es necesario comprobar que las propiedades descritas en el extracto de código 3.21 se encuentren presentes.

Este archivo se utiliza para indicar a Metricbeat cómo debe recopilar, procesar y enviar datos de monitorización de diferentes servicios y sistemas a Elasticsearch.

La propiedad "setup.kibana" es utilizada para configurar la conexión entre Metricbeat y Kibana, que es la interfaz de usuario utilizada para visualizar los datos recopilados. Por otra parte, la propiedad "output.elasticsearch" es utilizada para configurar la conexión entre Metricbeat y Elasticsearch, que es la base de datos utilizada para almacenar los datos recopilados.

```
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["workstation-antsuabon:9200"]
  protocol: "https"

  # Public Key Infrastructure (PKI) certificates
  ssl.certificate_authorities:
    - /etc/pki/ca.crt
  ssl.certificate: "/etc/pki/ids-antsuabon.crt"
  ssl.key: "/etc/pki/ids-antsuabon.key"

  # Authentication credentials
  username: "elastic"
  password: "elastic-password"
```

Extracto de código 3.21: Configuración de Metricbeat

Además, se han realizado las configuraciones necesarias para utilizar el protocolo HTTPS y una autenticación basada en certificados (estos certificados serán generados posteriormente cuando se configuren los elementos principales de la pila ELK).

En resumen, este archivo de configuración indica que Metricbeat debe enviar los datos de monitorización recopilados a Elasticsearch y Kibana alojados en la máquina con el nombre de host "workstation-antsuabon".

3.2. Ejecución de la pila ELK desde contenedores Docker

En esta sección se explica el proceso de instalación de la pila de aplicaciones ELK (Elasticsearch, Logstash y Kibana) sobre un entorno Docker y por separado del dispositivo Raspberry Pi.

En primer lugar, se ha separado el Sistema de Detección de Intrusos de las herramientas de análisis de logs para evitar sobresaturar el dispositivo. Debido a los bajos recursos del dispositivo, es altamente probable que no pueda llevar la carga de una base de datos (Elasticsearch) y el sistema de detección de intrusos (Snort).

En segundo lugar, se ha decidido utilizar la tecnología de contenedores Docker, puesto que facilita crear, desplegar y ejecutar aplicqciones en diferentes entornos de manera fácil y eficiente. Esto significa que Docker permite a los desarrolladores crear aplicaciones que funcionan de manera confiable en cualquier entorno, independientemente de la configuración del sistema operativo o las bibliotecas disponibles.

En los siguientes apartados, se explica como he configurado las distintas aplicaciones de la pila ELK dentro de un entorno de Docker Compose, que me permite orquestar diferentes contenedores para que se comuniquen entre si y así crear un entorno más complejo [2, 1]. Se comienza dando una

explicación general de la estructura del fichero, para en los siguientes apartados profundizar en los diferentes servicios.

3.2.1. Instalación de Docker

Antes de comenzar, es necesario tener instalado Docker el sistema sistema. En mi caso he seguido los pasos necesarios para instalar Docker Desktop en Windows. Se puede descargar Docker desde la página oficial [4].

Una vez completada la instalación y ejecutado el Docker Compose cuya construcción será detallada en los siguientes apartados, obtenemos el mismo resultado que se puede observar en la figura 3.3.

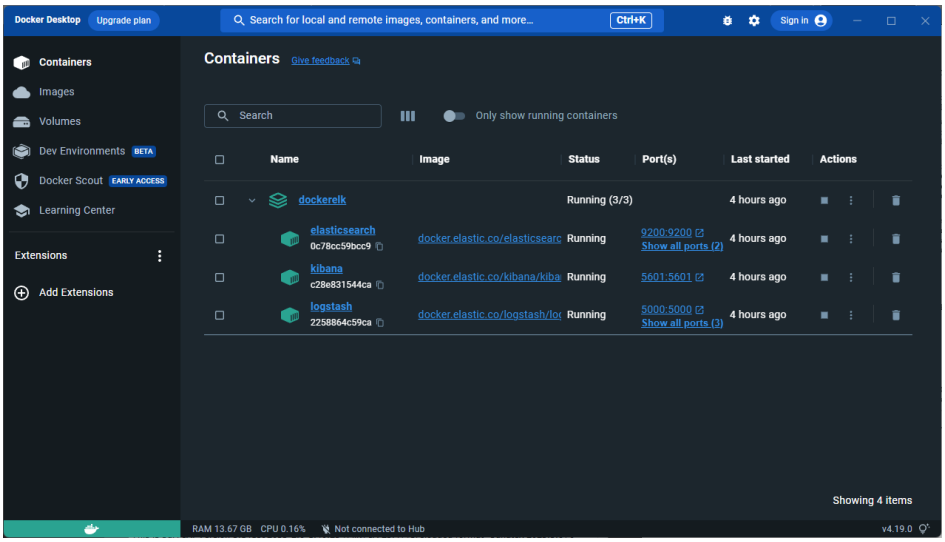

Figura 3.3: Contenedores de la pila ELK en ejecución

3.2.2. Creación del archivo Docker Compose

Para la construcción del entorno donde se va a desplegar la pila de aplicaciones ELK se ha utilizado la herramienta Docker Compose.

El archivo "docker-compose.yml" contiene la configuración de Docker Compose. En el extracto de código 3.22 se muestra la estructura general del fichero de configuración. Este fichero describe cómo se deben ejecutar los contenedores de Elasticsearch, Logstash y Kibana, y cómo deben interactuar entre sí.

```
version: '3.8'
services:
  create-certs:
    ...
  elasticsearch:
    ...
  kibana:
    ...

volumes:
  es_data:
    driver: local
  kibana_data:
    driver: local
```

```
networks:
  elk:
    driver: bridge
```

Extracto de código 3.22: Estructura principal del fichero docker-compose.yml

Este fichero se encuentra dividido en 3 partes: services, volumes y networks.

La sección "services" describe los servicios que se ejecutaran. En este caso, los servicios son Elasticsearch y Kibana. Cada uno de estos servicios será definido en las siguientes secciones. Para este proyecto no ha sido necesario configurar Logstash.

La sección "volumes" es utilizada para crear volúmenes Docker que permitan persistir datos independientemente de los contenedores que se puedan instanciar en un momento determinado. En este caso, se crea el volumen "es_data", que es utilizado para almacenar los datos de Eslasticsearch y el volumen "kibana_data', que es utilizado para almacenar los datos de Kibana'.

La sección "networks" es utilizada para crear una red Docker que comunique los diferentes contenedores.

3.2.3. Servicio Docker para crear los certificados

Este servicio utiliza la imagen de Elasticsearch para crear certificados y configuraciones relacionadas en un entorno de Elasticsearch y Kibana (extracto de código 3.23).

Una vez creadom se realiza una verificación de salud para asegurarse de que los archivos de certificado existan y luego ejecuta un script de bash para realizar tareas adicionales, como la creación de una Autoridad de Certificación (CA) y generación de certificados para varias instancias.

También se establecen permisos de archivos, espera la disponibilidad de Elasticsearch, establece una contraseña para el usuario "kibana_system" y realiza otras acciones relacionadas con la seguridad y configuración.

```
create_certs:
  container_name: create_certs
  image: docker.elastic.co/elasticsearch/elasticsearch:${
    STACK_VERSION}
  healthcheck:
    test: ["CMD-SHELL", "[ -f config/certs/elasticsearch/
      elasticsearch.crt ]"]
    interval: 1s
    timeout: 5s
    retries: 120
  user: "0"
  command: >
    bash -c '
      if [ x${ELASTIC_PASSWORD} == x ]; then
        echo "Set the ELASTIC_PASSWORD environment variable in
    the .env file";
        exit 1;
      elif [ x${KIBANA_PASSWORD} == x ]; then
        echo "Set the KIBANA_PASSWORD environment variable in
    the .env file";
        exit 1;
      fi;
      if [ ! -f config/certs/ca.zip ]; then
        echo "Creating CA";
        bin/elasticsearch-certutil ca --silent --pem -out
    config/certs/ca.zip;
```

```
␣␣␣␣␣␣␣␣unzip␣config/certs/ca.zip␣-d␣config/certs;
␣␣␣␣␣␣fi;
␣␣␣␣␣␣if␣[␣!␣-f␣config/certs/certs.zip␣];␣then
␣␣␣␣␣␣␣␣echo␣"Creating␣certs";
␣␣␣␣␣␣␣␣echo␣-ne␣\
␣␣␣␣␣␣␣␣"instances:\n"\
␣␣␣␣␣␣␣␣"␣␣-␣name:␣elasticsearch\n"\
␣␣␣␣␣␣␣␣"␣␣␣␣dns:\n"\
␣␣␣␣␣␣␣␣"␣␣␣␣␣␣-␣workstation-antsuabon\n"\
␣␣␣␣␣␣␣␣"␣␣␣␣␣␣-␣elasticsearch\n"\
␣␣␣␣␣␣␣␣"␣␣␣␣␣␣-␣localhost\n"\
␣␣␣␣␣␣␣␣"␣␣␣␣ip:\n"\
␣␣␣␣␣␣␣␣"␣␣␣␣␣␣-␣127.0.0.1\n"\
␣␣␣␣␣␣␣␣"␣␣-␣name:␣ids-antsuabon\n"\
␣␣␣␣␣␣␣␣"␣␣␣␣dns:\n"\
␣␣␣␣␣␣␣␣"␣␣␣␣␣␣-␣ids-antsuabon\n"\
␣␣␣␣␣␣␣␣"␣␣␣␣␣␣-␣localhost\n"\
␣␣␣␣␣␣␣␣"␣␣␣␣ip:\n"\
␣␣␣␣␣␣␣␣"␣␣␣␣␣␣-␣127.0.0.1\n"\
␣␣␣␣␣␣␣␣>␣config/certs/instances.yml
␣␣␣␣␣␣␣␣bin/elasticsearch-certutil␣cert␣--silent␣--pem␣-out␣
    config/certs/certs.zip␣--in␣config/certs/instances.yml␣--ca-
    cert␣config/certs/ca/ca.crt␣--ca-key␣config/certs/ca/ca.key;
␣␣␣␣␣␣␣␣unzip␣config/certs/certs.zip␣-d␣config/certs;
␣␣␣␣␣␣fi;
␣␣␣␣␣␣echo␣"Setting␣file␣permissions";
␣␣␣␣␣␣chown␣-R␣root:root␣config/certs;
␣␣␣␣␣␣echo␣"Waiting␣for␣Elasticsearch␣availability";
␣␣␣␣␣␣until␣curl␣-s␣--cacert␣config/certs/ca/ca.crt␣https://
    elasticsearch:9200␣|␣grep␣-q␣"missing␣authentication␣
    credentials";␣do␣sleep␣30;␣done;
␣␣␣␣␣␣echo␣"Setting␣kibana_system␣password";
␣␣␣␣␣␣until␣curl␣-s␣-X␣POST␣--cacert␣config/certs/ca/ca.crt␣-u␣
    "elastic:${ELASTIC_PASSWORD}"␣-H␣"Content-Type:␣application/
    json"␣https://elasticsearch:9200/_security/user/kibana_system
    /_password␣-d␣"{\"password\":\"${KIBANA_PASSWORD}\"}"␣|␣grep␣
    -q␣"^{}";␣do␣sleep␣10;␣done;
␣␣␣␣␣␣echo␣"All␣done!";
␣␣␣␣'
  working_dir: /usr/share/elasticsearch
  volumes:
    - ./certs:/usr/share/elasticsearch/config/certs
  networks:
    - elk
```

Extracto de código 3.23: Servicio de Elasticsearch del fichero docker-compose.yml para la generación de los certificados

3.2.4. Servicio Docker para Elasticsearch

Este servicio utiliza la imagen oficial de Elasticsearch de Docker y expone el puerto 9200 para acceder a la API de Elasticsearch. También configura el tamaño de la memoria y monta un volumen para almacenar los datos de Elasticsearch.

El extracto de código 3.24 constituye el servicio "elasticsearch" que forma parte del fichero "docker-compose.yml".

```
elasticsearch:
  image: docker.elastic.co/elasticsearch/elasticsearch:${
    STACK_VERSION}
  container_name: elasticsearch
  ulimits:
    memlock:
      soft: -1
      hard: -1
  environment:
    ES_JAVA_OPTS: "-Xmx256m -Xms256m"
    ELASTIC_PASSWORD: ${ELASTIC_PASSWORD}
    discovery.type: single-node
    xpack.security.enabled: true
    xpack.security.http.ssl.enabled: true
    xpack.security.http.ssl.key: certs/elasticsearch/
      elasticsearch.key
    xpack.security.http.ssl.certificate: certs/elasticsearch/
      elasticsearch.crt
    xpack.security.http.ssl.certificate_authorities: certs/ca/ca
      .crt
    xpack.security.transport.ssl.enabled: true
    xpack.security.transport.ssl.key: certs/elasticsearch/
      elasticsearch.key
    xpack.security.transport.ssl.certificate: certs/
      elasticsearch/elasticsearch.crt
    xpack.security.transport.ssl.certificate_authorities: certs/
      ca/ca.crt
    xpack.security.transport.ssl.verification_mode: certificate
  healthcheck:
    test:
      [
        "CMD-SHELL",
        "curl -s --cacert config/certs/ca/ca.crt https://
          localhost:9200 | grep -q 'missing authentication
          credentials'",
      ]
    interval: 10s
    timeout: 10s
  ports:
    - 9200:9200
  volumes:
    - es_data:/usr/share/elasticsearch/data
    - ./certs:/usr/share/elasticsearch/config/certs
  networks:
    - elk
  depends_on:
    create_certs:
      condition: service_healthy
```

Extracto de código 3.24: Servicio de Elasticsearch del fichero docker-compose.yml

A continuación, se explican uno por uno los elementos que forman parte del servicio previamente definido.

- "elasticsearch": Es el nombre del servicio de Elasticsearch que se va a crear.
- "image": Se utiliza la imagen docker.elastic.co/elasticsearch/elasticsearch con una versión específica $STACK_VERSION.

- "ulimits": Establece los límites de recursos para el contenedor. En este caso, se configura un límite de memoria (memlock) en -1, lo que indica que no hay límite.
- "environment": Aquí se definen las variables de entorno utilizadas por Elasticsearch:
  - "ES_JAVA_OPTS": Configura las opciones de Java para Elasticsearch, estableciendo un límite máximo (-Xmx256m) y mínimo (-Xms256m) de memoria asignada.
  - "ELASTIC_PASSWORD": Especifica la contraseña utilizada para acceder a Elasticsearch.
  - "discovery.type": Establece el tipo de descubrimiento de nodos de Elasticsearch. En este caso, se utiliza single-node, lo que indica que se trata de un único nodo en el clúster.
  - "xpack.security.enabled": Habilita la seguridad de X-Pack en Elasticsearch.
  - "xpack.security.http.ssl.enabled": Habilita el soporte SSL/TLS para las comunicaciones HTTP en Elasticsearch.
  - "xpack.security.http.ssl.key": Ruta al archivo de clave privada utilizada para las comunicaciones SSL/TLS en Elasticsearch.
  - "xpack.security.http.ssl.certificate": Ruta al archivo de certificado utilizado para las comunicaciones SSL/TLS en Elasticsearch.
  - "xpack.security.http.ssl.certificate_authorities": Ruta al archivo de autoridad de certificación utilizado para verificar los certificados de clientes en las comunicaciones SSL/TLS en Elasticsearch.
  - "xpack.security.transport.ssl.enabled": Habilita el soporte SSL/TLS para las comunicaciones de transporte en Elasticsearch.
  - "xpack.security.transport.ssl.key": Ruta al archivo de clave privada utilizada para las comunicaciones de transporte SSL/TLS en Elasticsearch.
  - "xpack.security.transport.ssl.certificate": Ruta al archivo de certificado utilizado para las comunicaciones de transporte SSL/TLS en Elasticsearch.
  - "xpack.security.transport.ssl.certificate_authorities": Ruta al archivo de autoridad de certificación utilizado para verificar los certificados de nodos en las comunicaciones de transporte SSL/TLS en Elasticsearch.
  - "xpack.security.transport.ssl.verification_mode": Establece el modo de verificación de certificados de transporte en Elasticsearch.
- "healthcheck": Define la comprobación de estado de salud para el contenedor de Elasticsearch. En este caso, se utiliza el comando curl para realizar una solicitud a https://localhost:9200 y se verifica si se obtienen las credenciales de autenticación requeridas.
- "ports": Expone el puerto 9200 del contenedor al mismo puerto del host.
- "volumes": Asocia volúmenes al contenedor para persistir los datos de Elasticsearch y los certificados en rutas específicas.
- "depends_on": Esto indica que el servicio elasticsearch solo se iniciará si el servicio create_certs se inicia y se encuentra en un estado saludable.

3.2.5. Servicio Docker para Logstash

Este servicio utiliza la imagen oficial de Logstash de Docker y expone el puerto 9600 para acceder a la API de Logstash. También configura el tamaño de la memoria.

El extracto de código 3.25 constituye el servicio "logstash" que forma parte del fichero "docker-compose.yml".

```
logstash:
  image: docker.elastic.co/logstash/logstash:8.7.1
```

```
  container_name: logstash
  environment:
    LS_JAVA_OPTS: "-Xmx256m -Xms256m"
  ports:
    - "5000:5000/tcp"
    - "5000:5000/udp"
    - "9600:9600"
  depends_on:
    - elasticsearch
  networks:
    - elk
```

Extracto de código 3.25: Servicio de Logstash del fichero docker-compose.yml

A continuación, se explican uno por uno los elementos que forman parte del servicio previamente definido.

- La imagen de Docker utilizada es "docker.elastic.co/logstash/logstash:8.7.1", que es la versión 8.7.1 de Logstash proporcionada por Elastic.
- El nombre del contenedor se define como "logstash".
- Se definen variables de entorno con la configuración para Logstash, incluyendo el uso de memoria JVM ("-Xmx256m -Xms256m").
- Se exponen los puertos 5000 (TCP y UDP) y el puerto 9600 para que se pueda acceder a Logstash desde el exterior.
- Se especifica que este contenedor depende del contenedor de Elasticsearch, lo que significa que Elasticsearch se iniciará antes de Logstash.
- Se conecta el contenedor a la red de Docker "elk" para que pueda comunicarse con los otros contenedores de Elasticsearch y Kibana.

3.2.6. Servicio Docker para Kibana

Este servicio utiliza la imagen oficial de Kibana y la configura para ser accedida desde el exterior en el puerto 5601 y que se comunique con los servicios previamente definidos.

El extracto de código 3.26 constituye el servicio "kibana" que forma parte del fichero "docker-compose.yml".

```
kibana:
  image: docker.elastic.co/kibana/kibana:8.8.1
  container_name: kibana
  environment:
    ELASTICSEARCH_HOSTS: https://elasticsearch:9200
    ELASTICSEARCH_USERNAME: kibana_system
    ELASTICSEARCH_PASSWORD: ${KIBANA_PASSWORD}
    ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES: config/certs/ca/ca
      .crt
    XPACK_ENCRYPTEDSAVEDOBJECTS_ENCRYPTIONKEY: "..."
  healthcheck:
    test:
      [
        "CMD-SHELL",
        "curl -s -I http://localhost:5601 | grep -q 'HTTP/1.1
          302 Found'",
      ]
    interval: 10s
```

```
    timeout: 10s
    retries: 120
  ports:
    - 5601:5601
  networks:
    - elk
  volumes:
    - kibana_data:/usr/share/kibana/data
    - ./certs:/usr/share/kibana/config/certs
  depends_on:
    elasticsearch:
      condition: service_healthy
```

Extracto de código 3.26: Servicio de Kibana del fichero docker-compose.yml

A continuación, se explican uno por uno los elementos que forman parte del servicio previamente definido.

- se utiliza la imagen "docker.elastic.co/kibana/kibana:8.8.1", que es la versión 8.8.1 de Kibana.
- El nombre del contenedor se define como "kibana".
- se definen varias variables como "ELASTICSEARCH_HOSTS", "ELASTICSEARCH_USERNAME", "ELASTICSEARCH_PASSWORD", "ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES" y "XPACK_ENCRYPTEDSAVEDOBJECTS_ENCRYPTIONKEY". Estas variables se utilizan para configurar la conexión de Kibana con Elasticsearch.
- Se define un chequeo de salud para el contenedor. Este chequeo consiste en ejecutar un comando de prueba para verificar si el servicio Kibana está funcionando correctamente. En este caso, se utiliza el comando curl para hacer una solicitud HTTP a http://localhost:5601 y se comprueba si la respuesta es un código de estado "302 Found".
- Se expone el puerto 5601 para que se pueda acceder a Kibana desde el exterior.
- Se monta el volumen "kibana_data" en /usr/share/kibana/data, que se utiliza para persistir los datos de Kibana. También se monta el directorio local ./certs en /usr/share/kibana/config/certs, que contiene certificados SSL necesarios para la comunicación segura con Elasticsearch.
- Se especifica que este contenedor depende del contenedor de Elasticsearch, lo que significa que Elasticsearch se iniciará antes de Kibana.

3.2.7. Ejecución del archivo Docker Compose

Una vez explicados los diferentes servicios que componen el fichero de configuración de Docker Compose. Tan solo, queda guarda el archivo "docker-compose.yml" en un directorio y ejecuta el comando del extracto de código 3.27 en la terminal para iniciar la pila ELK.

```
docker-compose up
```

Extracto de código 3.27: Ejecución de Docker Compose

Esto creará los tres servicios de Elasticsearch, Logstash y Kibana utilizando las imágenes de Docker que has sido especificados en el archivo "docker-compose.yml". Si todo funciona correctamente, podremos acceder a la interfaz web de Kibana en http://localhost:5601 y ver los logs de Snort procesados por Logstash, junto a las métricas generadas por Metricbeat.

Una vez los certificados han sido generados, es posible copiar estos ficheros al dispositivo Raspberry Pi ejecutando el script presente en el extracto de código 3.28.

```
sudo cp certs/ca/ca.crt certs/ids-antsuabon/ids-antsuabon.crt certs/ids-
   antsuabon/ids-antsuabon.key /tmp
```

```
scp /tmp/ca.crt antsuabon@ids-antsuabon.local:/tmp
scp /tmp/ids-antsuabon.crt antsuabon@ids-antsuabon.local:/tmp
scp /tmp/ids-antsuabon.key antsuabon@ids-antsuabon.local:/tmp

sudo mkdir /etc/pki
sudo mv /tmp/ca.crt /tmp/ids-antsuabon.crt /tmp/ids-antsuabon.key /etc/
   pki
sudo chown -R root:root /etc/pki
sudo chmod 600 /etc/pki/{ca.crt,ids-antsuabon.crt,ids-antsuabon.key}
```

Extracto de código 3.28: Script para copiar certificados al Raspberry Pi

3.3. Implementación de los modelos de IDS

En esta sección se presentan las implementaciones de los modelos de Sistema de Detección de Intrusos (IDS) tanto fuera de línea como en línea. El IDS fuera de línea se encarga de monitorizar la red y detectar posibles intrusiones analizando capturas de paquetes previamente almacenadas. Por otro lado, el IDS en línea se configura como un punto de acceso inalámbrico utilizando un dispositivo Raspberry Pi, permitiendo el análisis en tiempo real del tráfico de red y la detección de intrusiones.

3.3.1. IDS fuera de línea

Es importante destacar que el IDS en línea requiere una configuración adecuada y la selección de las herramientas adecuadas para capturar y analizar el tráfico de red en tiempo real. Esto permite detectar y responder rápidamente a posibles intrusiones o actividades maliciosas, fortaleciendo la seguridad de la red.

En este caso, se utiliza el programa "airtun-ng" para crear un adaptador virtual que permite capturar y analizar el tráfico de red en tiempo real. Esto proporciona la capacidad de detectar intrusiones y actividades maliciosas a medida que ocurren en la red.

En este proyecto se ha desarrollado un script en Python diseñado para monitorear y analizar la actividad de una red inalámbrica [19]. A continuación se presentan los fragmentos de código necesarios para la configuración del adaptador virtual necesario para ejecutar el IDS en modo fuera de línea.

1. **Configuración inicial**: En esta sección se definen las variables necesarias para la configuración del IDS fuera de línea. El script hace uso de las siguientes variables:
   - `INTERFACE`: especifica el nombre de la interfaz de red inalámbrica que se utilizará.
   - `TARGET_NETWORK`: especifica el nombre del punto de acceso al que se dirigirá el ataque.
   - `PASSWORD`: especifica la contraseña para el punto de acceso.
   - `BSSID`: especifica la dirección MAC del punto de acceso.
   - `CHANNEL`: especifica el número de canal en el que se encuentra el punto de acceso.

```
INTERFACE = "wlan0"
TARGET_NETWORK = "<<ACCESS_POINT_NAME>>"
PASSWORD = "<<WIRELESS_PASSWORD>>"
BSSID = "<<BSSID>>"
CHANNEL = "<<CHANNEL>>"
```

Extracto de código 3.29: Configuración inicial de las variables

2. **Recopilación de procesos, función de terminación y configuración de señales**: En este fragmento de código se definen las funciones necesarias para recopilar los procesos en ejecución, manejar la terminación del script de forma adecuada y configurar las señales para el manejo de eventos.

```
# Process collection
PROCESS_LIST = []

# Termination function
def exit_gracefully(*args):
    print("[-] The virtual interface has been stopped...")
    for process in PROCESS_LIST:
        process.send_signal(signal.SIGKILL)
    signal.signal(signal.SIGINT, signal.SIG_DFL)
    os.kill(os.getpid(), signal.SIGINT)

signal.signal(signal.SIGINT, exit_gracefully)
signal.signal(signal.SIGTERM, exit_gracefully)
```

Extracto de código 3.30: Lógica para la terminación de procesos

3. **Proceso de captura de paquetes**: En este fragmento de código se inicia el proceso de captura de paquetes utilizando la herramienta .ªirodump-ng". Se especifica la interfaz, el canal, la dirección MAC del punto de acceso y se guarda la salida en un archivo CSV llamado .ºutput".

```
airodump_process = subprocess.Popen([f"airodump-ng {INTERFACE} --
    channel {CHANNEL} --bssid {BSSID} --write output --output-format
    csv"], shell=True, stdout=subprocess.DEVNULL)
PROCESS_LIST.append(airodump_process)
time.sleep(15)
```

Extracto de código 3.31: Inicialización del proceso Airodump-ng

4. **Proceso de interfaz virtual**: En esta sección se inicia el proceso de creación de la interfaz virtual utilizando la herramienta .ªirtun-ng". Se especifica la dirección MAC del punto de acceso, la contraseña y el nombre del punto de acceso.

```
airtun_process = subprocess.Popen([f"airtun-ng -a {BSSID} -p {
    PASSWORD} -e {TARGET_NETWORK} -t 0 {INTERFACE}"], shell=True,
    stdout=subprocess.DEVNULL)
PROCESS_LIST.append(airtun_process)
time.sleep(5)
```

Extracto de código 3.32: Inicialización del proceso Airtun-ng

5. **Habilitación de la interfaz virtual**: Este fragmento de código se encarga de habilitar la interfaz virtual creada anteriormente.

```
os.system("ifconfig at0 up")
```

Extracto de código 3.33: Habilitación de la interfaz virtual

6. **Bucle principa**l: En esta sección se inicia el bucle principal del script, que permite que el IDS fuera de línea siga funcionando y capturando paquetes de forma continua.

```
print("[+] The virtual interface has been started...")
while True:
    # The service is running
    pass
```

Extracto de código 3.34: Bucle principal del script

En este proyecto, aunque se logró hacer funcionar el sistema utilizando .ªirtun-ng"para crear el adaptador virtual y capturar el tráfico de red en tiempo real, se encontraron problemas de estabilidad. Estos problemas pueden manifestarse en desconexiones frecuentes, pérdida de paquetes o una conexión

poco confiable en general. Es por ello que se terminó adoptando la opción de configurar el dispositivo como un IDS en línea.

3.3.2. IDS en línea

Para crear un sistema de detección de intrusos en línea utilizando el Raspberry Pi, será necesario hacer circular el tráfico a través del dispositivo. En ese caso la forma más sencilla de hacer que el tráfico pase a través del Raspberry Pi es configurarlo como un punto de acceso inalámbrico.

Para esta arquitectura de red se hará uso del los siguientes dos adaptadores de red:

- Un adaptador de red inalámbrico para crear el punto de acceso.
- Un adaptador de red cableado para dar acceso a Internet a todos aquellos dispositivos que se conecten al punto de acceso.

A continuación, se enumeran los pasos a seguir para configurar desde cero un punto de acceso inalámbrico que permita a los usuarios conectarse a la red y obtener direcciones IP a través del servidor DHCP [6]. Además, se configurará el dispositivo para permitir el acceso a la red externa.

1. **Instalación de DNSMasq y HostAPD**: En el extracto de código 3.35, se instalan los paquetes necesarios para configurar el punto de acceso y se detienen los servicios en caso de que se encuentren en ejecución.

```
sudo apt install dnsmasq hostapd
sudo systemctl stop dnsmasq
sudo systemctl stop hostapd
```

Extracto de código 3.35: Deshabilitar punto de acceso inalámbrico

2. **Configuración de una dirección IP estática**: Se configurará una dirección IP estática para la interfaz de red encargada de hospedar el punto de acceso inalámbrico, de esta forma quedaría configurado la puerta de enlace de la nueva red de área local y se evitaría que el servicio "`wpa_supplicant`" intente gestionar la interfaz. Para ello, se configurará el fichero "/etc/dhcpcd.conf" con el contenido del extracto de código 3.36. Una vez completada la configuración, será necesario reiniciar el servicio "dhcpcd".

```
interface wlan0
    static ip_address=192.168.25.1/24
    nohook wpa_supplicant
```

Extracto de código 3.36: Configuración de una dirección IP estática

3. **Configuración del servidor DHCP**: Escribimos en el fichero "/etc/dnsmasq.conf" el contenido mostrado en el extracto de código 3.37. Esto hará que el servidor DHCP provea direcciones IP entre "192.168.25.2" y "192.168.25.20" durante periodos de 24 horas. Además, se utilizará el servidor "8.8.8.8" para la resolución de nombres DNS.

```
interface=wlan0
server=8.8.8.8
dhcp-range=192.168.25.2,192.168.25.20,255.255.255.0,24h
```

Extracto de código 3.37: Configuración del servidor DHCP

4. **Configuración del punto de acceso inalámbrico**: El extracto de código 3.38 muestra el contenido del archivo de configuración "/etc/hostapd/hostapd.conf", que se encarga de configurar los parámetros del punto de acceso inalámbrico. En este caso se ha configurado una red inalámbrica en la interfaz "wlan0" con el nombre "Test-IDS" y la contraseña "YOURPWD".

```
country_code=ES
interface=wlan0
```

```
ssid=Test-IDS
channel=9
auth_algs=1
wpa=2
wpa_passphrase=YOURPWD
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP CCMP
rsn_pairwise=CCMP
```

Extracto de código 3.38: Configuración del punto de acceso inalámbrico

Ahora lo único que nos queda es indicarle al sistema donde se encuentra el fichero de configuración. Para ello agregaremos la línea del extracto de código 3.39 al archivo "/etc/default/hostapd".

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Extracto de código 3.39: Referencia a la configuración del punto de acceso inalámbrico

5. **Puesta en marcha del punto de acceso inalámbrico**: Reiniciamos y habilitamos el servicio para aplicar los cambios (extracto de código 3.40).

```
sudo systemctl unmask hostapd
sudo systemctl enable hostapd
sudo systemctl start hostapd
```

Extracto de código 3.40: Habilitando el punto de acceso inalámbrico

6. **Configuración de enrutamiento**: Para que los dispositivos que se conecten al punto de acceso puedan acceder a la red principal y por consiguiente a Internet, aplicamos la configuración del extracto de código 3.41 para configurar contafuegos y persistir los cambios realizados en iptables.

```
sudo sh -c "echo 'net.ipv4.ip_forward=1' >> /etc/sysctl.conf"

sudo iptables -t nat -A  POSTROUTING -o eth0 -j MASQUERADE
sudo mkdir -p /etc/iptables
sudo sh -c "iptables-save > /etc/iptables/rules.v4"
```

Extracto de código 3.41: Configuración para habilitar el enrutamiento

Después de configurar el cortafuegos, agregamos la instrucción del extracto de código 3.42 al fichero "/etc/rc.local" para que cada vez que se inicie el sistema se restauren las reglas predefinidas del cortafuegos.

```
iptables-restore < /etc/iptables/rules.v4
```

Extracto de código 3.42: Línea para restaurar la configuración de enrutamiento

Una vez seguidos todos estos pasos dispondríamos de un dispositivo Rasberry Pi con unas interfaces configuras como en las presentes en la figura 3.4. Es decir, con una interfaz "wlan0" que hará de pasarela para que los dispositivos del nuevo punto de acceso puedan acceder a la red accedida por la interfaz "eth0".

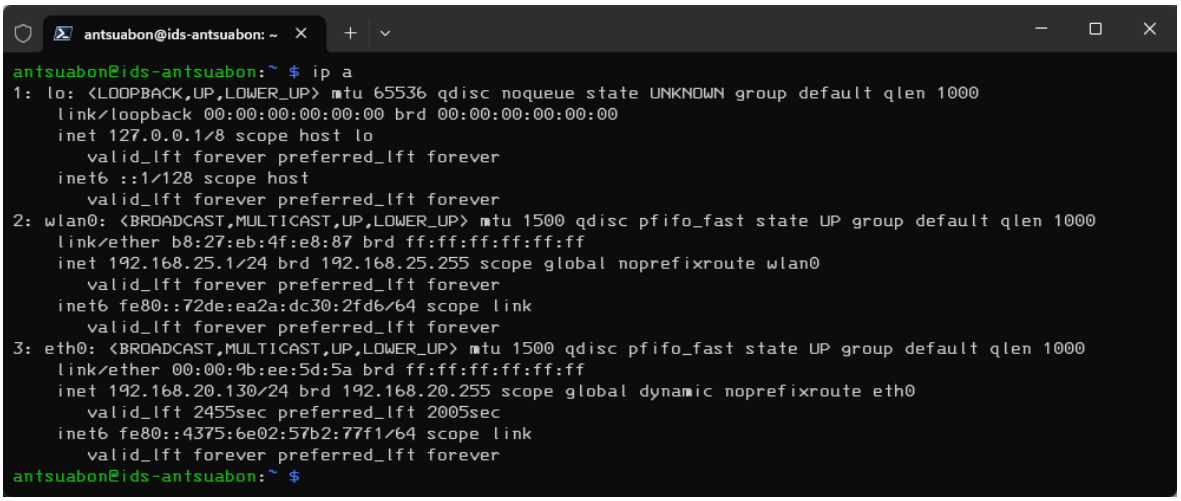

Figura 3.4: Interfaces de red configuradas para el IDS en línea

4. Fase de Optimización y Pruebas

En este capítulo se pretende abarcar todas las pruebas que nos ayuden a comprobar y verificar el correcto funcionamiento del sistema y cada una de sus partes.

El plan de pruebas pude considerarse que se encuentra dividido en tres partes. Una primera parte donde se comprueba el funcionamiento de Snort utilizando un conjunto reducido de reglas. En la segunda parte, se pretende configurar Kibana y explorar algunas de sus funcionalidades. Finalmente, se reflexiona sobre los resultados obtenidos y algunos de los problemas encontrados a lo largo del desarrollo.

4.1. Redacción de reglas para Snort

En esta sección se va a proceder a redactar una serie de reglas de Snort que posteriormente, nos permitirá la realización de pruebas sobre el sistema. En este caso, las reglas planteadas pueden dividirse en dos conjuntos según su enfoque. En cuanto a la reacción esperada, todas las reglas diseñadas tiene como único objetivo notificar al usuario, sin bloquear el tráfico de la red.

4.1.1. RuleSet 1: Herramientas de enumeración (NS)

En primer conjunto abarca reglas diseñadas para identificar y alertar sobre posibles amenazas de seguridad en una red. Estas reglas se basan en patrones de tráfico sospechoso o malicioso, como intentos de intrusión, ataques de denegación de servicios o tráfico asociado a malware conocido. El objetivo principal de estas reglas es detectar actividades anómalas en tiempo real.

Para este proyecto se han elegido un conjunto de reglas pequeño con el fin de crear una prueba de concepto. Las reglas que describo a continuación buscan detectar cuando un atacante esta intentando escanear la disponibilidad de un determinado equipo o servicio.

1. La regla presente en el extracto de código 4.1 busca detectar escaneos ICMP (Protocolo de Mensajes de Control de Internet) enviados desde cualquier dirección externa hacia la red interna. La regla se activará cuando se detecte un paquete ICMP de tipo 8 (solicitud de eco) con código 0. La detección de este tipo de paquetes puede indicar un intento de escanear la disponibilidad de un equipo o servicio.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (
    msg: "NS - ICMP Scan";
    icode:0; itype:8;
    classtype:icmp-event;
    sid:100005;
    rev: 1;
)
```

Extracto de código 4.1: Regla Snort (NS): ICMP Scan

2. La regla presente en el extracto de código 4.2 busca detectar escaneos TCP enviados desde cualquier dirección externa hacia el puerto 80 en la red interna. El puerto 80 es el puerto estándar utilizado por el protocolo HTTP. La regla se activará cuando se detecte un paquete TCP sin estado (stateless) en el flujo de tráfico, lo que indica un intento de escaneo de disponibilidad.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (
    msg:"NS - TCP Scan";
    flow:stateless;
    classtype:attempted-recon;
    sid:100006;
```

```
    rev:1;
)
```

Extracto de código 4.2: Regla Snort (NS): TCP Scan

3. La regla presente en el extracto de código 4.3 busca detectar escaneos TCP utilizando el flag FIN (Fin de la conexión) en cualquier dirección externa hacia cualquier puerto en la red interna. El escaneo TCP FIN es una técnica utilizada para determinar si un puerto está abierto o cerrado.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (
    msg:"NS - TCP FIN Scan";
    flow:stateless;
    flags:F;
    classtype:attempted-recon;
    sid:100007;
    rev:1;
)
```

Extracto de código 4.3: Regla Snort (NS): TCP FIN Scan

4.1.2. RuleSet 2: Información de identificación personal (PII)

El segundo conjunto de reglas se encuentra enfocado en salvaguardar la información personal y la privacidad de los usuarios. Estas reglas se encuentran dirigidas a detectar intentos de acceso no autorizado o exfiltración de datos confidenciales. Con estas reglas, se busca proteger la información sensible y evitar posibles violaciones de la privacidad.

La implementación de reglas de PII ayuda a garantizar el cumplimiento de estas normativas al detectar y prevenir la exposición no autorizada de datos personales.

Las reglas de PII actúan como una capa adicional de detección de posibles brechas de seguridad. Si un atacante intenta transferir datos de identificación personal fuera de la red o realizar acciones sospechosas relacionadas con la PII, estas reglas pueden generar alertas en tiempo real.

A continuación, se enumerará una serie de reglas que tienen como objetivo detectar si por la red se encuentra circulando información de identificación personal. El conjunto elegido se a elegido a modo de prueba para verificar el funcionamiento de las expresiones regulares.

1. **Documento Nacional de Identidad (DNI)**:

```
alert tcp any any -> any any (
    msg:"PII - Spanish National Identification Number (DNI)";
    classtype: sdf;
    pcre:"/\b\d{8}[A-HJ-NP-TV-Z]\b/i";
    sid:100001;
    rev:1;
)
```

Extracto de código 4.4: Regla Snort (PII): Documento Nacional de Identidad (DNI)

Explicación detallada de la expresión regular utilizada:

- `\b` indica un límite de palabra para asegurar que coincida con el número completo.
- `\d{8}` coincide con 8 dígitos seguidos.
- `[A-HJ-NP-TV-Z]` coincide con una letra mayúscula que representa la letra de control del DNI español, excluyendo las vocales y la letra O.
- `/i` al final de la expresión regular indica que la coincidencia debe ser insensible a mayúsculas y minúsculas.

2. **Fecha en formato DD/MM/YYYY**:

```
alert tcp any any -> any any (
    msg:"PII - Date in DD/MM/YYYY format";
    classtype: sdf;
    pcre:"/\b([3][01]|[12]\d|[0]?[1-9])[\/-]([1][12]|[0]?[1-9])
       [\/-](\d{4}|\d{2})\b/";
    sid:100002;
    rev:1;
)
```

Extracto de código 4.5: Regla Snort (PII): Fecha en formato DD/MM/YYYY

Explicación detallada de la expresión regular utilizada:

- `\b` indica un límite de palabra para asegurar que coincida con el texto de fecha completo.
- `([3][01]|[12]\d|[0]?[1-9])` coincide con los días en formato DD. Puede ser 31, 01-29 o 1-9.
- `[\/-]` Coincide con una barra diagonal o un guión, que es utilizado de separador.
- `([1][12]|[0]?[1-9])` coincide con los meses en formato MM. Puede ser 12, 01-09 o 1-9.
- `[\/-]` coincide con una barra diagonal o un guión, que es utilizado de separador.
- `(\d{4}|\d{2})`: Coincide con el año en formato YYYY o YY. Puede ser un año de cuatro dígitos o un año de dos dígitos.
- `\b` indica otro límite de palabra para asegurar que el texto de fecha esté completo.

3. **Código Internacional de Cuenta Bancaria (IBAN)**:

```
alert tcp any any -> any any (
    msg:"PII - International Bank Account Number (IBAN)";
    classtype: sdf;
    pcre:"/\b[a-zA-Z]{2}\d{2}(?:\s?\d{4}){5}\b/";
    sid:100003;
)
```

Extracto de código 4.6: Regla Snort (PII): Código Internacional de Cuenta Bancaria (IBAN)

Explicación detallada de la expresión regular utilizada:

- `\b` indica un límite de palabra para asegurar que coincida con el número de cuenta IBAN completo.
- `[a-zA-Z]{2}` coincide con dos letras que representan el código de país del IBAN.
- `\d{2}` coincide con dos dígitos que representan los dígitos de control del IBAN.
- `(\s?\d{4}){5}` coincide con cinco grupos de cuatro dígitos precedidos de un espacio opcional. Estos grupos representan el número de cuenta y otros detalles.
- `\b` indica otro límite de palabra para asegurar que el número de cuenta IBAN esté completo.

4. **Dirección de correo electrónico**:

```
alert tcp any any -> any any (
    msg:"PII - Email Address";
    classtype: sdf;
    pcre:"/\b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}\b/";
    sid:100004;
    rev:1;
)
```

Extracto de código 4.7: Regla Snort (PII): Dirección de correo electrónico

Explicación detallada de la expresión regular utilizada:

- `\b` indica un límite de palabra para asegurar que coincida con la dirección de correo electrónico completa.
- `[A-Za-z0-9._%+-]+` coincide con uno o más caracteres alfanuméricos, punto, guión bajo, porcentaje o signo más, que forman la parte local de la dirección de correo electrónico.
- `@[A-Za-z0-9.-]+` coincide con el símbolo de arroba seguido de uno o más caracteres alfanuméricos, punto o guion medio, que forman la parte del dominio de la dirección de correo electrónico.
- `\.[A-Za-z]{2,}` coincide con un punto seguido de dos o más letras que representan la extensión del dominio de nivel superior (TLD) de la dirección de correo electrónico.
- `\b` indica otro límite de palabra para asegurar que la dirección de correo electrónico esté completa.

4.2. Prueba de las reglas de Snort

En esta sección se busca probar el correcto funcionamiento de las reglas diseñadas en la sección anterior. Estas pruebas consisten en un proceso iterativo en el que se han ido modificando las reglas de Snort hasta adaptarse al objetivo de las pruebas realizada.

Al igual que en la sección anterior, esta sección se encuentra divida en dos apartados donde se probará cada una de las reglas aplicadas de la forma más adecuada.

4.2.1. RuleSet 1: Herramientas de enumeración (NS)

Para comprobar el correcto funcionamiento de las reglas de Snort especializadas en detectar escaneos de red, se han realizado los escaneos para los que fueros diseñadas las reglas. A continuación, se listan los pruebas realizadas.

1. En la primera prueba se ha utilizado el comando "ping" para enviar cuatro paquetes ICMP de solicitud eco (figura 4.1) a la dirección IP "192.168.25.5" (un dispositivo móvil conectado a la red inalámbrica).

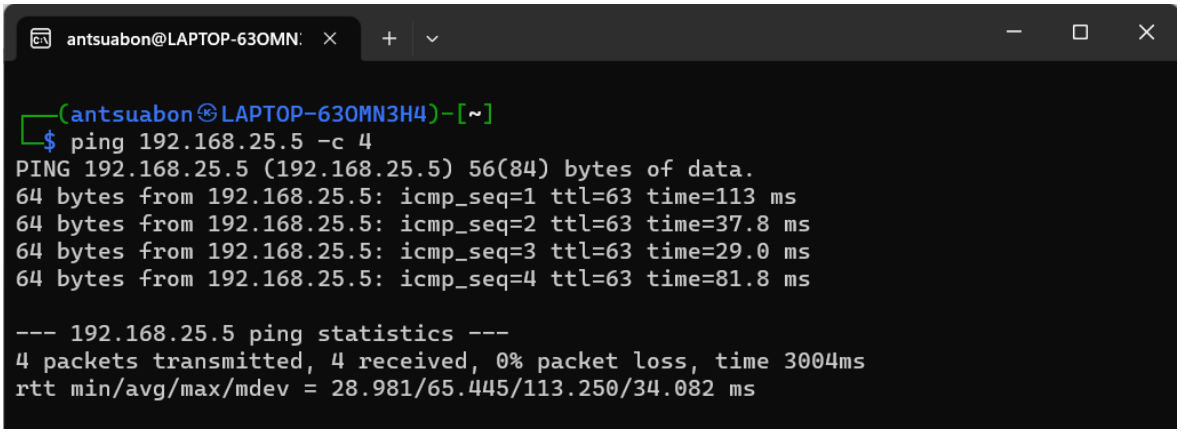

Figura 4.1: Escaneo de disponibilidad utilizando la herramienta "ping"

Realizando la prueba ilustrada en la figura anterior, se ha producido las 4 alertas esperadas para detectar la petición ICMP de tipo eco (figura 4.2).

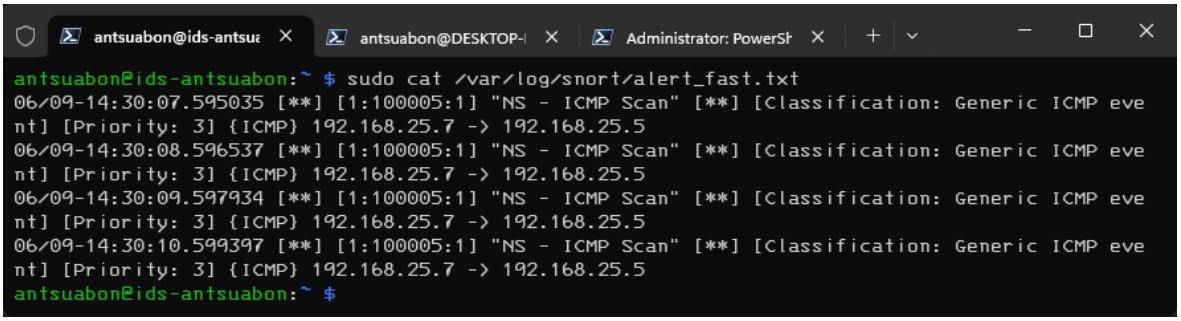

Figura 4.2: Alertas de Snort provocadas por la herramienta "ping"

2. Para la segunda prueba se ha utilizado la herramienta "nmap", con la que se ha realizado un escaneo TCP (figura 4.3) en la dirección "192.168.25.1" (dirección de la puerta de enlace).

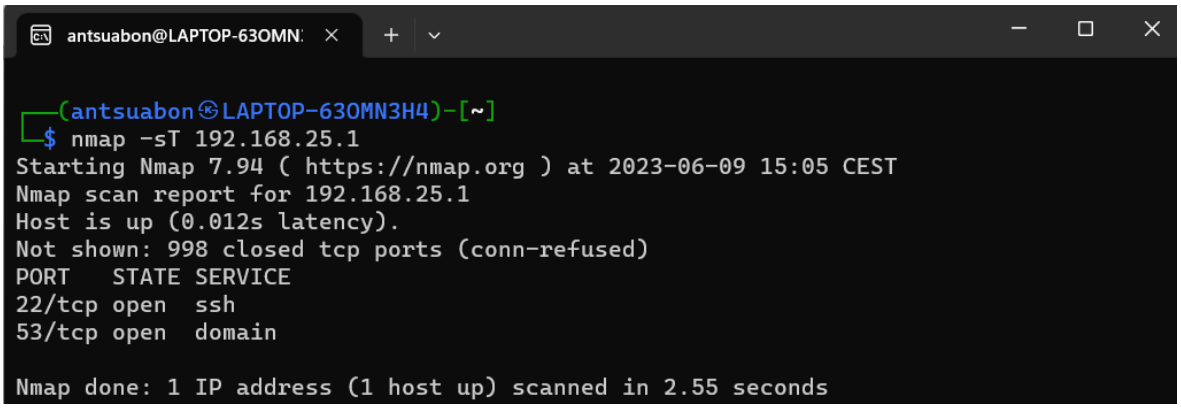

Figura 4.3: Escaneo de puertos TCP con "nmap"

Realizando la prueba ilustrada en la figura anterior, se han producido dos alertas de escaneo de red TCP para el puerto 80 (figura 4.4).

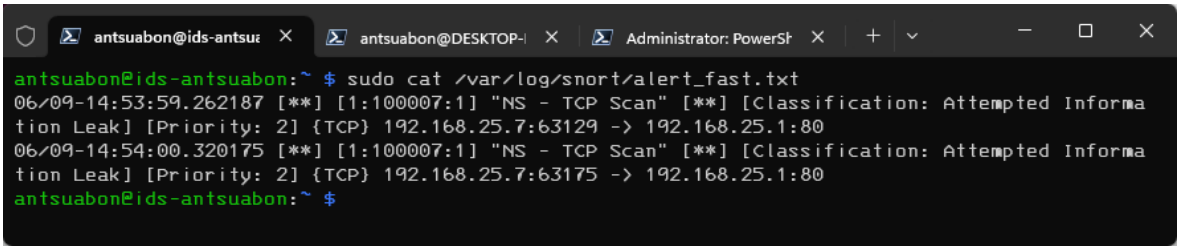

Figura 4.4: Alertas de snort provocadas por el escaneo de puertos TCP

3. En la tercera prueba se ha utilizado la herramienta "nmap" para realizar un escaneo TCP FIN en los puertos 22 y 53 (figura 4.5) de la dirección IP "192.168.25.5" para determinar si estos servicios se encuentran operativos.

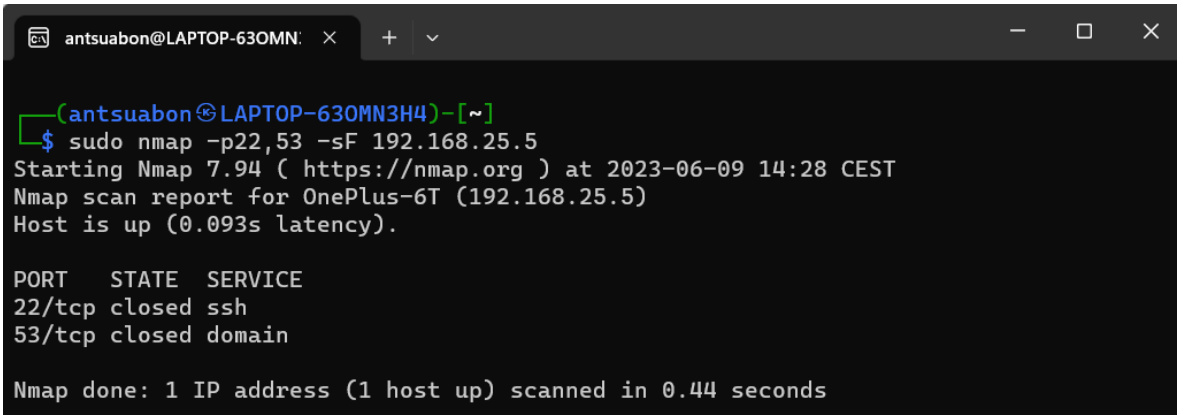

Figura 4.5: Escaneo de puertos TCP FIN con "nmap"

Realizando la prueba ilustrada en la figura anterior, se han generado las 2 alertas esperadas (figura 4.6), es decir, una alerta para el escaneo del puerto 22 y otra alerta para el escaneo del puerto 53.

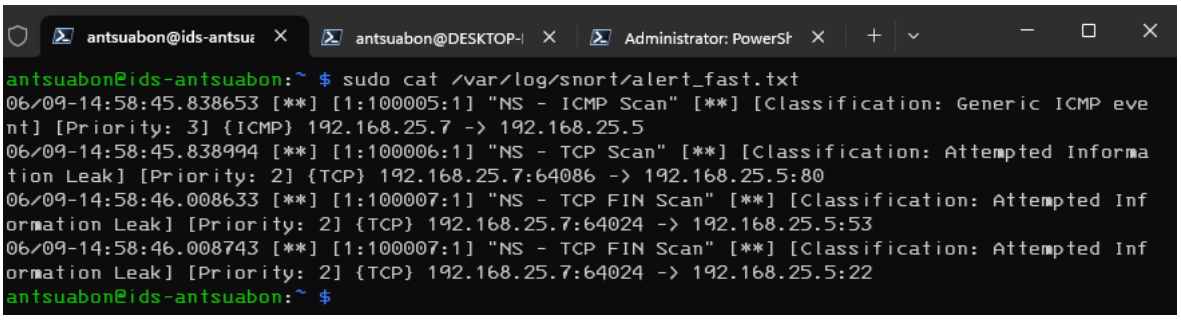

Figura 4.6: Alertas de Snort provocadas por el escaneo de puertos TCP FIN

4.2.2. RuleSet 2: Información de identificación personal (PII)

Para probar las reglas de Snort de forma aislada al resto del sistema se ha diseñado un script en Python que utilizando la herramienta "netcat" establece una conexión TCP entre dos equipos de la red y envía paquetes de datos específicos para provocar que se disparen las alertas de Snort.

Este script incluye varias listas de payloads para diferentes tipos de información personal identificable (PII).

Para cada una de las reglas se han generado 3 casos de prueba:

- **Caso positivo**: En el mensaje aparece un ejemplo correcto de la información que busca detectar la regla.
- **Caso negativos**:
  - El mensaje no contiene información relacionada con la regla.
  - En el mensaje aparece un ejemplo incorrecto de la información que busca detectar la regla.

A continuación, se enumeran los payloads de cada una de las pruebas diseñadas en la sección anterior.

1. Documento Nacional de Identidad:

```
# Payloads for Spanish National Identification Number (DNI)
payloads_dni = [
    "This␣is␣a␣test␣packet␣without␣a␣DNI.",
    "This␣is␣a␣packet␣with␣a␣valid␣DNI:␣12345678Z.",
    "This␣is␣a␣packet␣with␣an␣invalid␣DNI:␣987654321."
]
```

Extracto de código 4.8: Prueba de la regla Snort: Documento Nacional de Identidad

2. Número de cuenta bancaria internacional:

```
# Payloads for International Bank Account Number (IBAN)
payloads_iban = [
    "This␣is␣a␣test␣packet␣without␣an␣IBAN.",
    "This␣is␣a␣packet␣with␣a␣valid␣IBAN:␣ES12␣3456␣7890␣1234␣5678␣
       9012.",
    "This␣is␣a␣packet␣with␣an␣invalid␣IBAN:␣DE1234567890."
]
```

Extracto de código 4.9: Prueba de la regla Snort: Número de cuenta bancaria internacional

3. Fechas en formato DD/MM/YYYY:

```
# Payloads for Date in DD/MM/YYYY format
payloads_date = [
    "This␣is␣a␣test␣packet␣without␣a␣date.",
    "This␣is␣a␣packet␣with␣a␣valid␣date:␣31/12/2023.",
    "This␣is␣a␣packet␣with␣an␣invalid␣date:␣12-31-2023."
]
```

Extracto de código 4.10: Prueba de la regla Snort: Fecha en formato DD/MM/YYYY

4. Direcciones de correo electrónico:

```
# Payloads for Email Address
payloads_email = [
    "This␣is␣a␣test␣packet␣without␣an␣email␣address.",
    "This␣is␣a␣packet␣with␣a␣valid␣email␣address:␣test@example.com.",
    "This␣is␣a␣packet␣with␣an␣invalid␣email␣address:␣invalid.email."
]
```

Extracto de código 4.11: Prueba de la regla Snort: Dirección de correo electrónico

La función "send_packets" es utilizada para enviar los paquetes a un servidor utilizando la herramienta netcat. Este script itera sobre la lista de payloads que se ha descrito anteriormente y utilizar "subprocess.run" para ejecutar el comando "nc". Una vez enviado el paquete se realiza una pausa para distribuir el envío de todos los paquetes en el tiempo.

```
import subprocess
import time

# Define the function to send packets to the server
def send_packets(server_addess, server_port, payloads):
    for payload in payloads:
        subprocess.run(['nc', server_addess, server_port, '-w', '1'],
           input=payload.encode(), stdout=subprocess.DEVNULL, stderr=
           subprocess.DEVNULL)
        time.sleep(1)
```

```
# Call the function to send the payloads to the server

payloads = payloads_dni + payloads_email + payloads_iban + payloads_date
send_packets("192.168.25.5", "2525", payloads)
```

Extracto de código 4.12: Regla Snort (PII): Email Address

En la última parte del script, se llama a la función "`send_packets`" con la dirección IP del servidor (192.168.25.5) y el puerto (2525), junto con la lista de payloads que se construyó combinando todas las listas de payloads anteriores. Esta dirección IP pertenece a un dispositivo móvil que se encuentra en la red desplegada por la Raspberry Pi.

Finalmente, en la figura 4.7 se muestra el registro de Snort donde se puede apreciar que tal y como se había previsto, solo se han lanzado cuatro alertas, es decir, una por cada subgrupo de payloads.

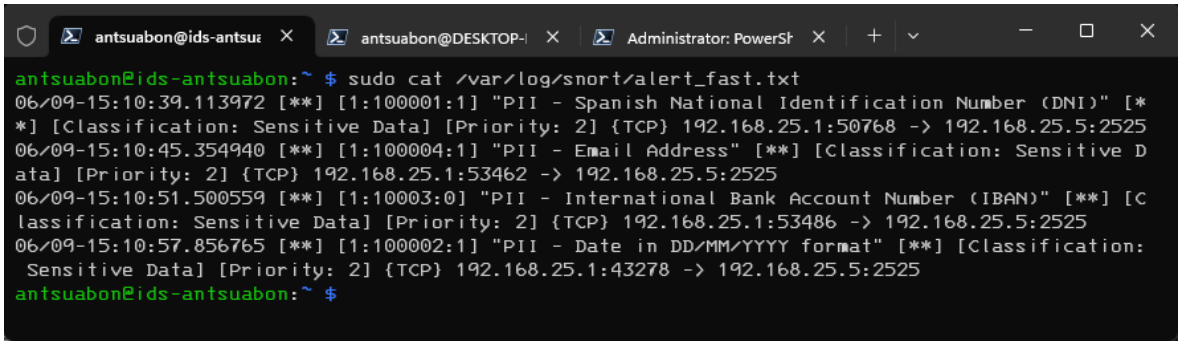

Figura 4.7: Resultado de las pruebas de las reglas de tipo PII

4.3. Visualización de logs

En esta sección se pretende explorar como configurar y utilizar los elementos de la pila de aplicaciones ELK para la visualización y el análisis de logs. Los logs o registros son eventos o mensajes que se generan durante la ejecución de una aplicación o servicio.

En este proyecto, se han utilizado los registros generados por el sistema de detección de intrusos Snort. Para la generación de datos se prueba se han utilizado los casos de prueba ya diseñados en los apartados anteriores.

4.3.1. Configuración de la visualización de logs

En primer lugar, se explorará la estructura de los datos analizados, además de realizar las configuraciones oportunas para que éstos puedan ser visualizados correctamente. En el extracto de código 4.13 se muestra un ejemplo de como son los registros generados por la aplicación Snort y cuales son las claves relevantes a la hora de interpretar el resultado.

```
{
    "timestamp" : "06/09-15:10:39.113972",
    "iface" : "wlan0",
    "src_addr" : "192.168.25.1", "src_port" : 50768,
    "dst_addr" : "192.168.25.5", "dst_port" : 2525,
    "proto" : "TCP", "action" : "allow",
    "msg" : "PII - Spanish National Identification Number (DNI)",
    "priority" : 2, "class" : "Sensitive Data",
    "sid" : 100001
```

```
}
```

Extracto de código 4.13: Ejemplo de salida de datos en formato JSON para Snort 3

Las claves que aparecen en las entradas del fichero JSON:

- `timestamp`: Indica la marca de tiempo en la que se generó el evento.
- `iface`: Representa la interfaz de red a través de la cual se capturó el tráfico.
- `src_addr`: Indica la dirección IP de origen del paquete capturado.
- `src_port`: Indica el puerto de origen del paquete capturado.
- `dst_addr`: Indica la dirección IP de destino del paquete capturado.
- `dst_port`: Indica el puerto de origen del paquete capturado.
- `proto`: Indica el protocolo utilizado por el paquete capturado.
- `action`: Representa la acción tomada por Snort con respecto al paquete capturado.
- `msg`: Proporciona una descripción o mensaje asociado al evento capturado.
- `priority`: Indica la prioridad asignada al evento capturado.
- `class`: Representa la clase o categoría a la que pertenece el evento capturado.
- `sid`: Identificador único de la regla de Snort que se activó con este evento.

A continuación, se detallan los pasos necesarios para configurar correctamente la visualización de los logs generados por Snort en Kibana:

1. Verificar que los registros generados por Snort estén siendo enviados a Elasticsearch a través de Filebeat.
2. En Kibana, acceder a la sección "Observability" y seleccionar "Logs" en el menú lateral.
3. En la página de configuración de logs, pulsar sobre "Settings" para acceder a la configuración de fuentes de logs.
4. En la sección "Logs sources", cambiar la fuente de los logs de "Indices" a "Data view" para que puedas visualizar los datos correctamente.
5. A continuación, seleccionar la vista de datos que ha sido creada previamente para los registros de Snort. Si aún no ha sido creada una vista de datos, deberá hacerse siguiendo los pasos necesarios para definir la estructura de los datos y los filtros relevantes (figura 4.8).
6. El siguiente paso consiste en configurar las columnas que deseamos que sean visibles desde la vista de "Logs" (figura 4.9).
7. Una vez seleccionada la vista de datos, podrán verse los registros de Snort en la página principal de Kibana, en la pestaña "Logs". Aquí se puede explorar y analizar los registros utilizando las funcionalidades proporcionadas por Kibana.

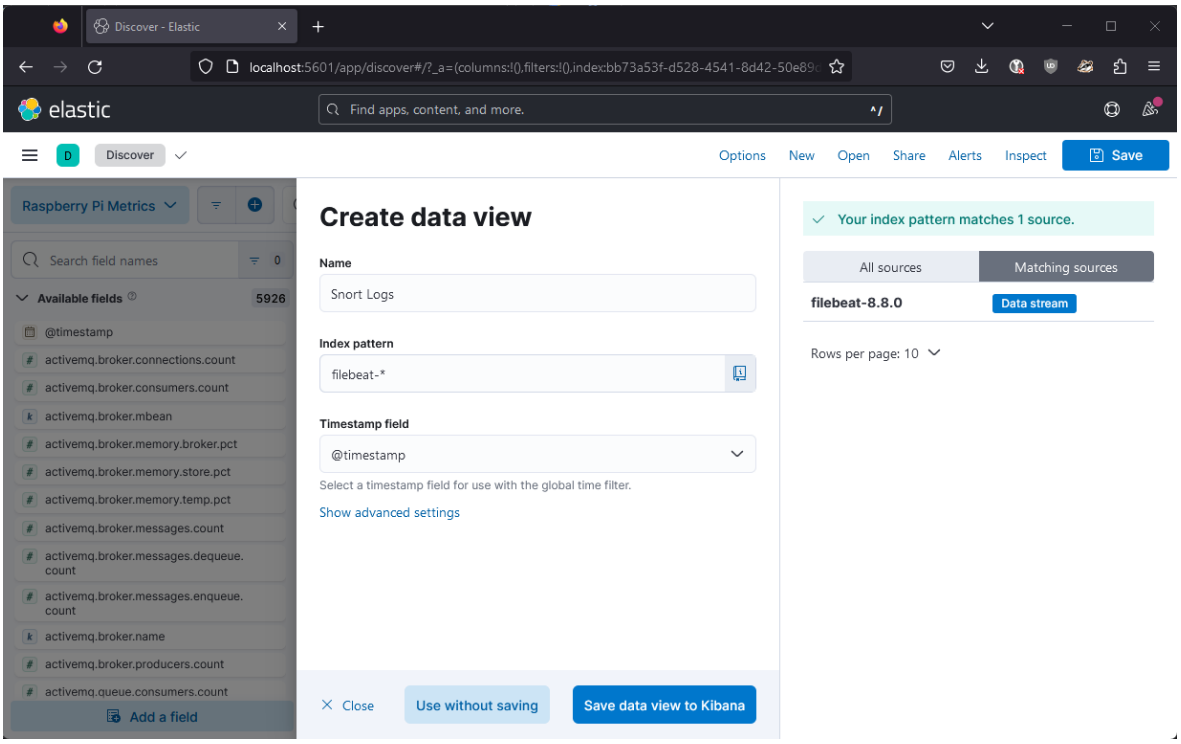

Figura 4.8: Creación de una vista de datos en Kibana

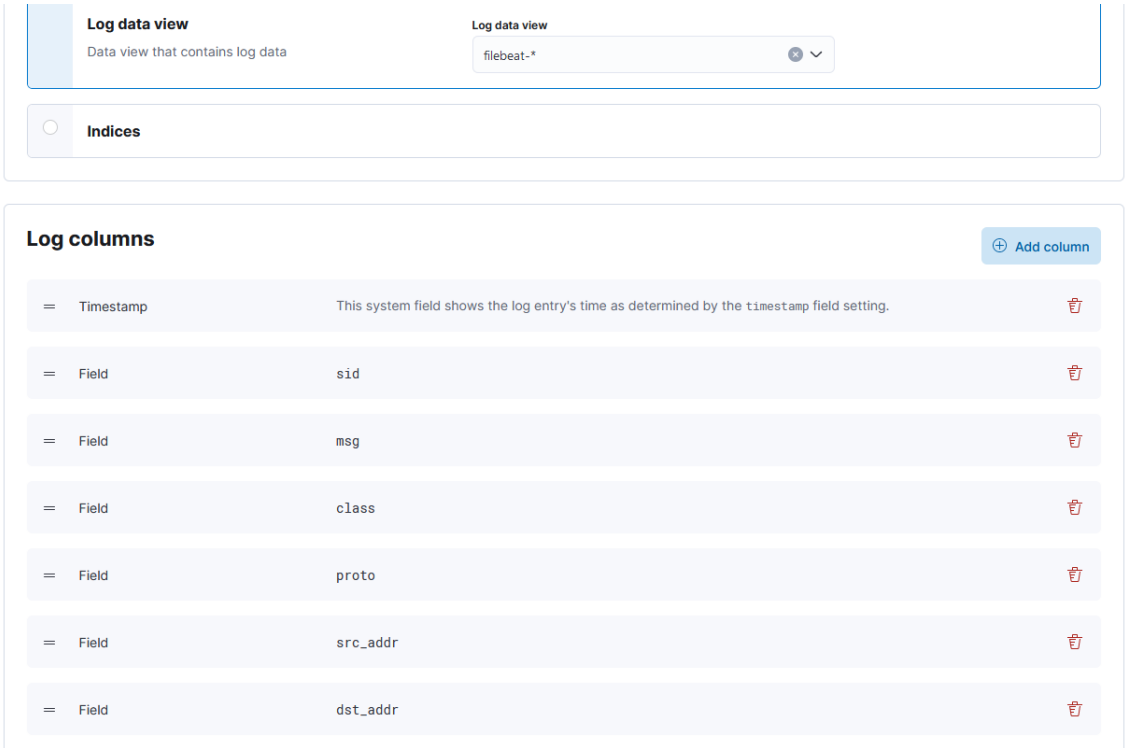

Figura 4.9: Personalización de la vista de logs en Kibana

4.3.2. Prueba de la visualización de logs

En esta sección se realizará una prueba de la visualización de logs utilizando la pila ELK (Elasticsearch, Logstash, Kibana) para analizar los registros generados por Snort, el sistema de detección de intrusos utilizado en el proyecto. Mediante la configuración previa realizada en la sección anterior, se mostrarán los registros de Snort en Kibana (figura 4.10), lo que permitirá explorar, filtrar y analizar los datos de los eventos capturados. A través de esta prueba, se podrá verificar la correcta configuración y funcionamiento de la visualización de logs en la pila ELK, aportando una mayor comprensión de los eventos de seguridad detectados por Snort.

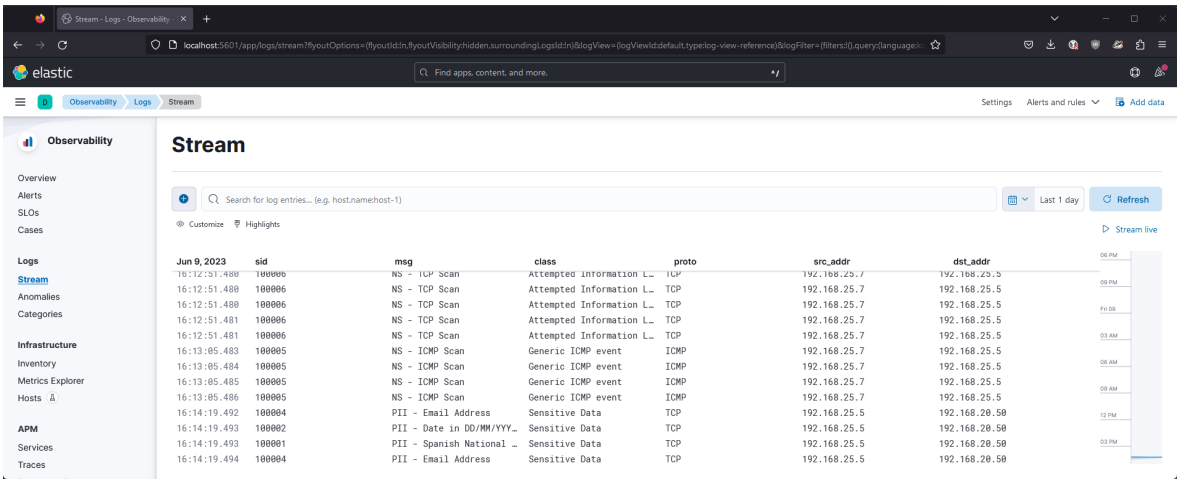

Figura 4.10: Registros de Snort vistos desde Kibana

4.4. Monitorización del Raspberry Pi

Las métricas permiten evaluar el estado actual y el rendimiento del dispositivo. asociados a las métricas del dispositivo.

En este aparatado se investigan los paneles de Kibana que se encuentran en el menú "Infrastructure", que se encuentra contenido dentro del menú de "Observability". Este apartado se encuentra diseñado para monitorizar y analizar la infraestructura en tiempo real.

Esta vista proporciona una visión general de los componentes y recursos de infraestructura, como servidores, contenedores, máquinas virtuales, redes y servicios, y permite realizar un seguimiento de su rendimiento y estado. Una vez se pulsa sobre alguno de los dispositivos mostrados, es posible ver un resumen con los datos más relevantes del dispositivo. Algunas de estos datos son:

- **Métricas**: Se mostrarán diversas métricas que proporcionan información sobre el rendimiento y la salud del dispositivo. Estas métricas pueden incluir el uso de CPU, la carga promedio, el consumo de memoria, el uso de almacenamiento, el tráfico de red, el rendimiento de disco, entre otros.
- **Registros**: Muestra el listado de logs que han sido generados con el dispositivo en cuestión como origen. En este caso los registros contienen las alertas generadas por el sistema de detección de intrusos Snort.
- **Procesos**: Lista sobre el estado de los procesos y la cantidad de recursos que consumen. En la figura 4.11, se muestra el listado de procesos en ejecución el dispositivo Raspberry Pi, entre los que se puede encontrar "snort", "metricbeat" y "filebeat".
- **Metadatos**: Resumen general del dispositivo, que puede incluir su nombre, dirección IP, sistema operativo, versión, ubicación física, entre otros detalles descriptivos.

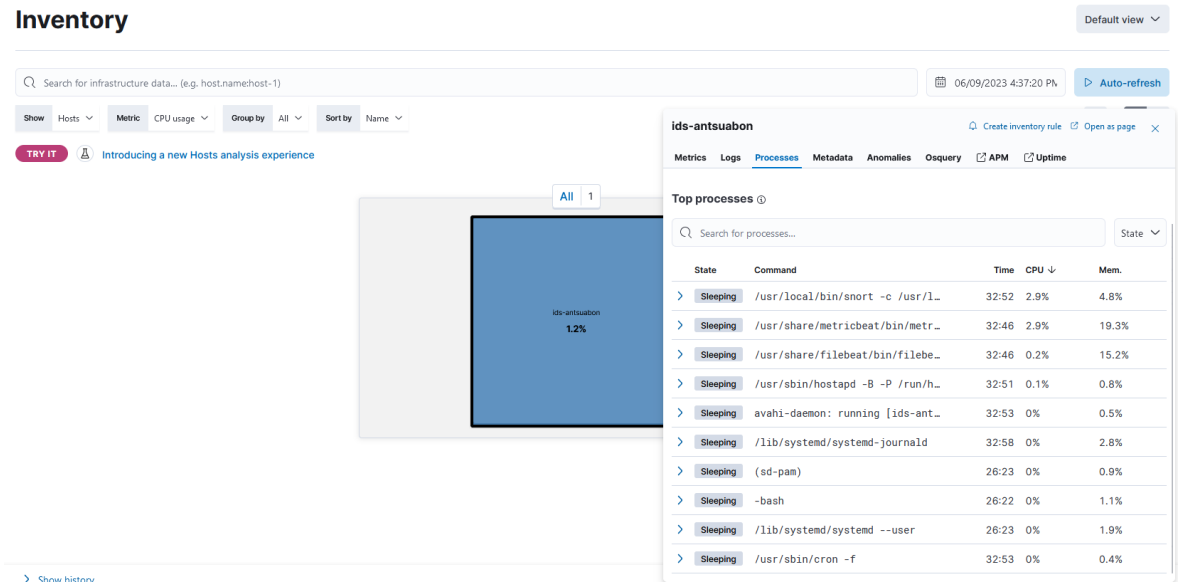

Figura 4.11: Inventario de dispositivos con métricas disponibles

Desde la figura anterior, es posible acceder a la vista de detalles donde se puede apreciar las métricas del dispositivo. En la figura 4.12, se muestra un resumen sobre el dispositivo (identificador, sistema operativo y valor actual de las métricas) y un gráfico sobre el uso de CPU. Por otra parte, en la figura 4.13 contiene información sobre el nivel de carga del dispositivo, el uso de la memoria RAM y el tráfico de red.

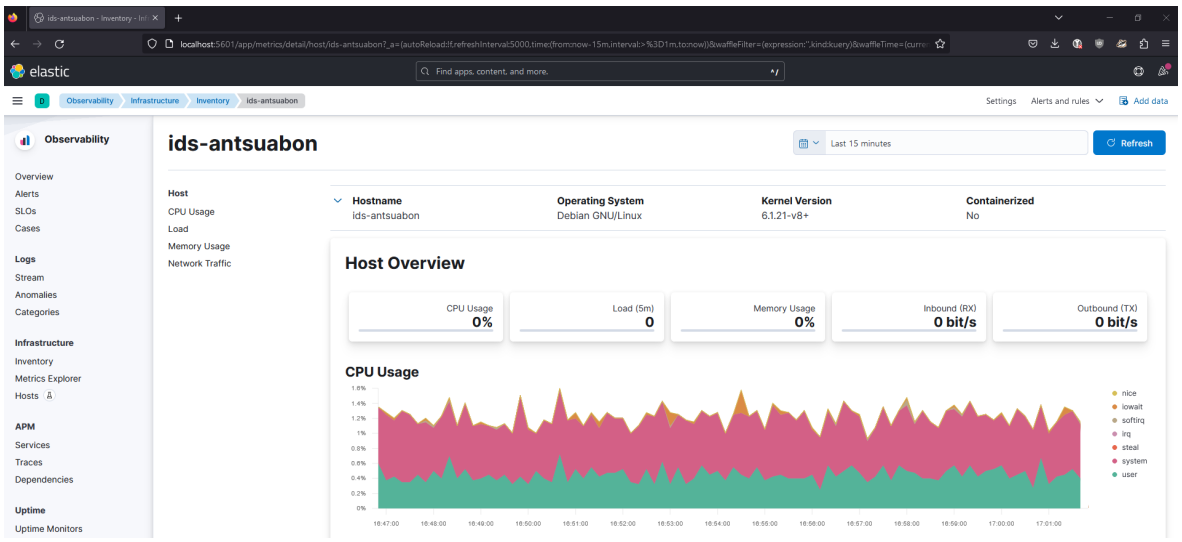

Figura 4.12: Vista de detalles del dispositivo Raspberry Pi (1)

Figura 4.13: Vista de detalles del dispositivo Raspberry Pi (2)

Al recopilar métricas detalladas sobre los procesos en ejecución en la Raspberry Pi, Metricbeat puede ayudar a identificar cuellos de botella o áreas de mejora en el rendimiento del sistema. Por ejemplo, si se detecta un alto uso de la CPU en un proceso específico, se puede investigar y optimizar ese proceso para mejorar el rendimiento general.

4.5. Dashboard de Kibana

En esta sección se muestra el Dashboard de Kibana que se ha desarrollado para este proyecto. El Dashboard de Kibana permite crear paneles personalizados que muestran visualizaciones de datos en tiempo real. Además, éste permite agregar diferentes tipos de visualizaciones, como gráficos de barras, gráficos circulares, líneas de tiempo, mapas y mucho más.

Una funcionalidad interesante de este panel es que permite aplicar filtros y consultas para restringir los datos que se muestran en tiempo real. Esto permite obtener una vista general de los datos y detectar patrones, tendencias o anomalías.

El propósito del Dashboard desarrollado es permitir al usuario visualizar de manera intuitiva y en tiempo real los eventos y alertas generados por Snort. En la figura 4.14, se muestra el resultado de este desarrollo.

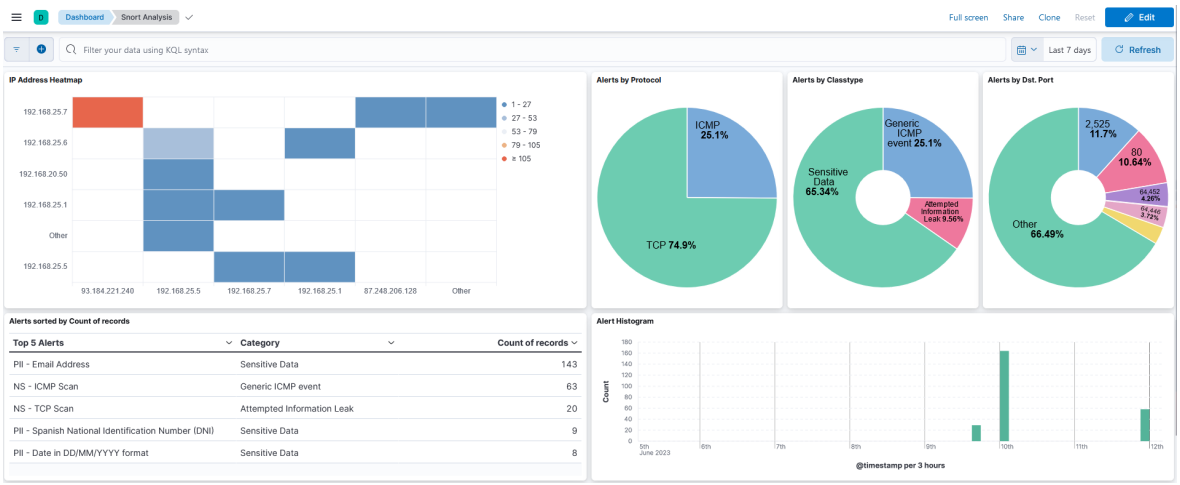

Figura 4.14: Vista completa del Dashboard de Kibana

A continuación, se explica de forma individual cada uno de los indicadores mostrados en el Dashboard.

1. **Alerts by Protocol**: Consiste en un diagrama de sectores, donde cada sector representa el número de alertas que han aparecido con un determinado protocolo en una determinada franja de tiempo.
2. **Alerts by Classtype**: Consiste en un diagrama de sectores, donde cada sector representa el número de alertas que han aparecido con un determinado tipo de alerta en una determinada franja de tiempo.
3. **Alerts by Dst. Port**: En este diagrama de sectores cada sector representa el número de alertas que han aparecido con un determinado puerto de destino en una determinada franja de tiempo. Este indicador puede ayudar a determinar que servicio se encuentra en ejecución o que servicio puede estar siendo explotado por un atacante.

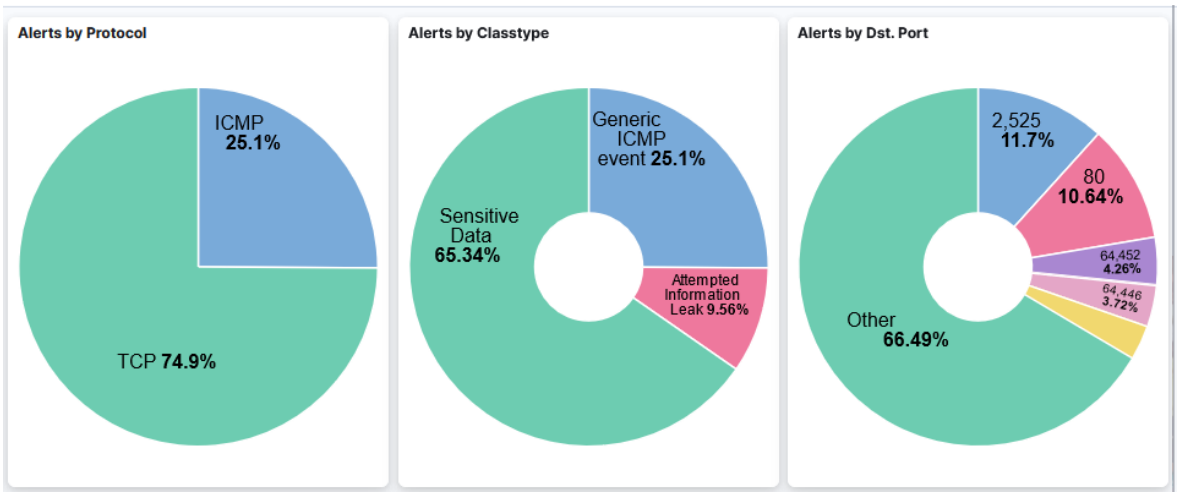

Figura 4.15: Diagramas de sectores del Dashboard de Kibana

4. **Alerts Histogram**: Consiste en un histograma que permite visualizar en que franja de tiempo se han producido las alertas y en que cantidad. Este indicador nos ayudará a detectar concentraciones anormales de alertar en determinadas franjas de tiempo.

Figura 4.16: Histograma de alertas del Dashboard de Kibana

5. **Alerts sorted by Count of records**: Este indicador consiste únicamente de una table agregada por el nombre de la alerta, de forma que se muestran de forma ordenada las alertas junto a su categoría en función del número de veces que aparece.

Alerts sorted by Count of records

Top 5 Alerts	Category	Count of records
PII - Email Address	Sensitive Data	143
NS - ICMP Scan	Generic ICMP event	63
NS - TCP Scan	Attempted Information Leak	20
PII - Spanish National Identification Number (DNI)	Sensitive Data	9
PII - Date in DD/MM/YYYY format	Sensitive Data	8

Figura 4.17: Tabla de alertas según su frecuencia del Dashboard de Kibana

6. **IP Address Heatmap**: Este indicador muestra la relación entre las distintas direcciones IP de origen y destino. Utilizando este indicador podemos observar que combinación de dirección IP de origen y destino ha hecho saltar más alertas.

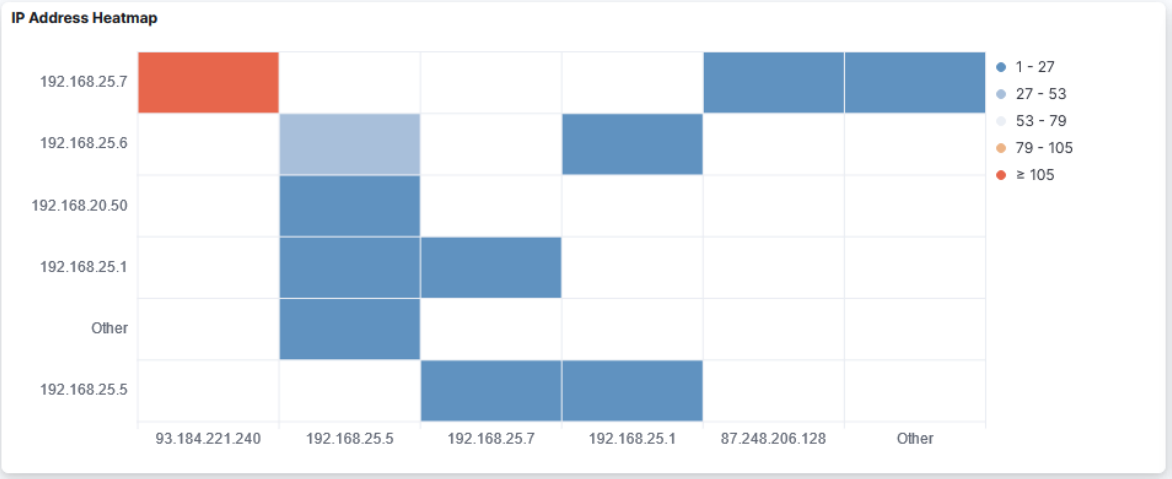

Figura 4.18: Diagrama de mapa de calor del Dashboard de Kibana

7. **Src. IP Address by Alert Name**: Consiste en un gráfico que muestra la proporción de alertas que han saltado en función de la dirección IP de origen para cada regla de Snort que se ha definido. Esto nos permitirá determinar cuales son los equipos más problemáticos de la red, es decir, son el origen de determinados tipos de alerta.

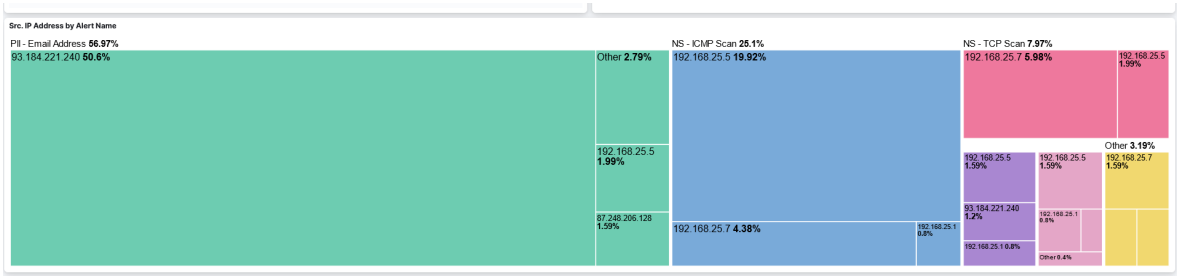

Figura 4.19: Diagrama de mapeo de árboles del Dashboard de Kibana

4.6. Alertas de Kibana

El apartado de Alertas en Observabilidad de Kibana es una funcionalidad que te permite configurar y recibir notificaciones automáticas sobre eventos o condiciones específicas en los datos y métricas. El uso de esta funcionalidad nos permite detectar amenazas sin necesidad de estar constantemente monitorizando el sistema.

A continuación, se explican las alertas creadas para probar el comportamiento del sistema.

- **Regla para Snort**: En la figura 4.20 se muestra la configuración de una regla de Kibana que hace saltar una alerta cuando el número de alertas perteneciente a una única clase excede el valor de 10 en los 5 últimos minutos.

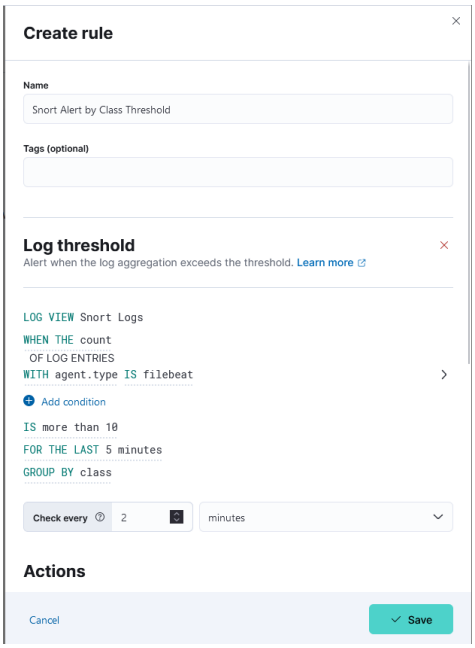

Figura 4.20: Alerta de Kibana sobre el número de registros por categoría

- **Regla para Metricbeat**: En la figura 4.21 se muestra la configuración de una regla de Kibana que hace saltar una alerta cuando el uso de la CPU alcanza un determinado umbral en el

dispositivo durante los últimos 5 minutos. En este caso, se lanzará una advertencia al superar el 30 % y se lanzara una alerta al superar el 50 %.

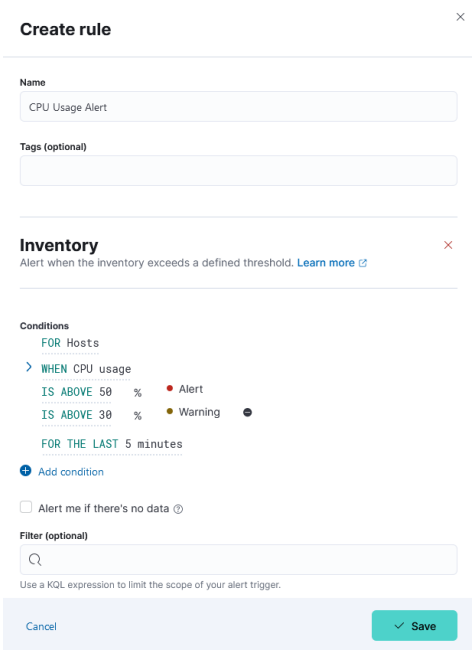

Figura 4.21: Alerta de Kibana sobre el uso de CPU

Una vez se han creado las reglas, se han generado las siguientes dos situaciones para que salten las alertas de la figura 4.22. Este panel una vista consolidada de todas las alertas activas, su estado, detalles y métricas asociadas, lo que facilita la supervisión y respuesta a eventos importantes en tiempo real.

- Para lanzar la regla relacionada con Snort se ha ejecutado la herramienta "ping" entre dos equipos de la red, enviando 15 peticiones de tipo eco.
- Para lenzar la regla relacionada con el uso de la CPU se ha utilizado la herramienta "stress" para someter al dispositivo hasta los niveles de estrés deseados.

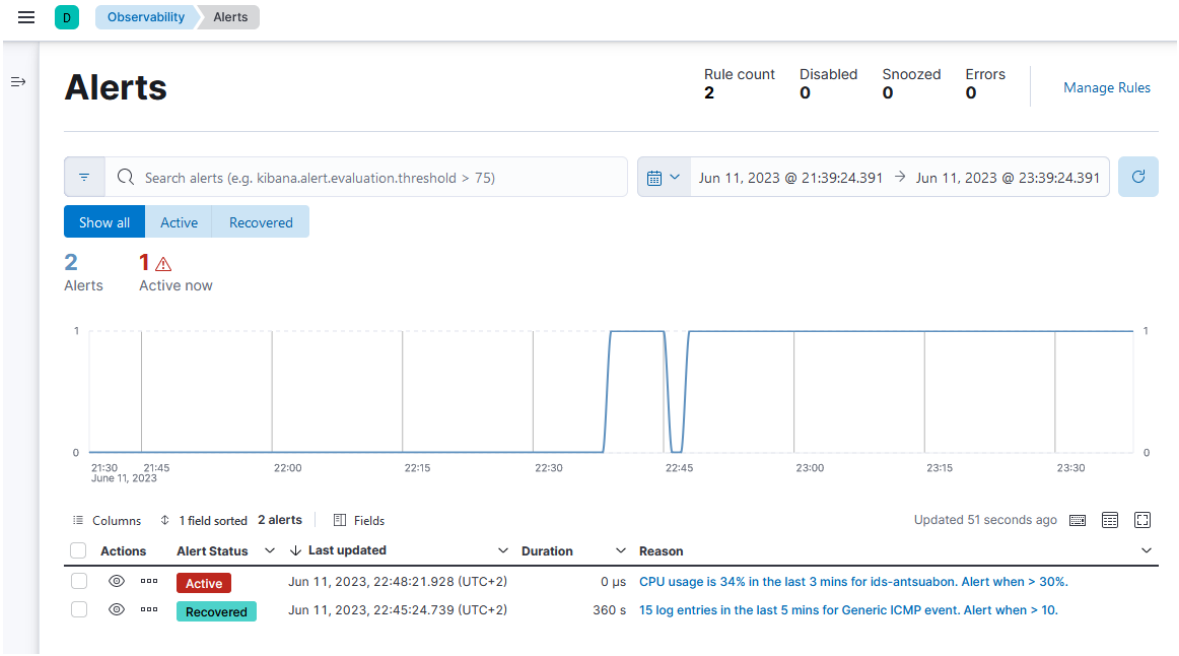

Figura 4.22: Panel de alertas de Kibana

Enlazando con la prueba anterior, se ha configurado la aplicación de Kibana para utilizar una cuenta de Gmail como servidor de correo electrónico. El conector de correo electrónico es una función de integración dentro de la plataforma de Observabilidad de Kibana que permite enviar notificaciones de alertas y eventos importantes a través de correo electrónico.

En la figura 4.23 se muestra un ejemplo donde se ha alertado que han aparecido 15 entradas de log pertenecientes a la categoría "Generic ICMP event".

Figura 4.23: Notificación de la alerta vía email

El conector de correo electrónico permite enviar notificaciones por correo electrónico de forma inmediata cuando se detecte un evento de seguridad relevante. Esto permite a los administradores y equipos de seguridad recibir una alerta temprana y responder rápidamente ante posibles ataques o violaciones de seguridad.

4.7. Seguimiento del proyecto

Durante el seguimiento del proyecto, se han identificado varios problemas y se han encontrado soluciones efectivas para cada uno de ellos. Estos desafíos incluyeron la escasez de semiconductores, la instalación costosa a nivel de hardware, la limitación de recursos para ejecutar la pila ELK y la configuración de certificados para Elasticsearch. Además, se consideraron alternativas que, aunque finalmente descartadas, demostraron la exploración de diferentes enfoques a la hora de implementar los componentes del sistema.

4.7.1. Problemas encontrados

En el transcurso del desarrollo del proyecto se han identificado varios problemas y se han encontrado soluciones para cada uno. A continuación se presentan los problemas encontrados, su descripción y el estado de cada uno de ellos.

Identificador	P-001
Título	Crisis de los semiconductores
Estado	Solucionado
Descripción	Este problema consiste en la imposibilidad de adquirir un dispositivo Raspberry Pi 4 debido a la escasez de chips y semiconductores.
Solución	Se ha implementado el sistema de detección de intrusos sobre un Raspberry Pi 3 A+.

Cuadro 4.1: Problema encontrado: Crisis de los semiconductores

Identificador	P-002
Título	Instalación costosa a nivel de hardware
Estado	Solucionado
Descripción	Este problema consiste en que la instalación de Snort 3 y de otros componentes necesarios resulta muy costosa debido a la necesidad de compilar binarios a partir del código fuente.
Solución	Se ha utilizado una herramienta que permite customizar distribuciones de Debian para Raspberry Pi (CustomPiOS). Este cambio permite preinstalar todos los requisitos software en una imagen personalizada, dejado únicamente las configuraciones para ser realizadas sobre el dispositivo.

Cuadro 4.2: Problema encontrado: Instalación costosa a nivel de hardware

Identificador	P-003
Título	Recursos necesarios para ejecutar la pila de aplicaciones ELK
Estado	Solucionado
Descripción	Este problema consiste en la imposibilidad de instalar todos los componentes de la pila ELK debido a que este dispositivo posee menos capacidades hardware.
Solución	Se ha realizado la implementación de los componentes de la pila ELK en un equipo aislado, siendo las herramientas de recopilación de logs y méticas las únicas herramientas de la pila ELK que serán instaladas en el dispositivo Raspberry Pi.

Cuadro 4.3: Problema encontrado: Recursos necesarios para ejecutar la pila de aplicaciones ELK

Identificador	P-004
Título	Configuración de certificados para Elasticsearch
Estado	Solucionado
Descripción	Este problema consiste en que si se utilizan las configuraciones básicas de la pila ELK de forma insegura, es decir, sin autenticarse, no es posible acceder a muchas de las funcionalidades, entre las que se encuentra el sistema de alertas de Kibana.
Solución	Se ha configurado el entorno para que permita la autenticación utilizando certificados.

Cuadro 4.4: Problema encontrado: Configuración de certificados para Elasticsearch

4.7.2. Alternativas desechadas

Además de los problemas encontrados, también se consideraron alternativas que fueron descartadas. En este apartado se explican que líneas de investigación han sido descartadas debido a imposibilidades tecnológicas o problemas encontrados.

Identificador	AD-001
Título	Implemenatación de un IDS fuera de línea
Descripción	En un principio se planteó la implementación de un sistema de detección de intrusos fuera de línea. Pero fue descartado por dos motivos: ▪ La dificultad de implementar un IDS inalámbrico recae principalmente en que es necesario descifrar los paquetes que circulan en la red. ▪ La interfaz virtual necesaria para poder descifrar los datos de la red era inestable.

Cuadro 4.5: Alternativa desechada: Implementación de un IDS fuera de línea

5. Conclusiones

En general, las conclusiones de este proyecto han sido positivas, puesto que, he logrado profundizar en la configuración de un Sistema de Detección de Intrusos utilizando la herramienta *Snort*. Por otra parte, este proyecto me ha permitido conocer herramientas que desconocía, como la herramienta de *Kibana*, para la visualización y análisis de datos, o bien, la herramienta de *CustomPiOS* para la creación de una imagen Debian con todas las herramientas previamente instaladas.

Enumero las conclusiones o reflexiones que he podido obtener de este proyecto:

- Se ha logrado profundizar en el uso de herramientas que faciliten la instalación de los elementos necesarios para hacer funcionar la implementación en Raspberry Pi.
- Se ha desarrollado con éxito un conjunto de reglas de Snort orientadas a detectar ciertos tipos de escaneo de red y algunos patrones presentes en información de identificación personal.
- Se ha logrado desarrollar un conjunto de alertas para que Kibana notifique a los usuario cuando encuentra una determinada condición en el conjunto de logs. Además, esta herramienta se encargará de enviar una notificación al correo electrónico para garantizar una rápida actuación.
- No ha sido necesario el uso de Logstash debido a que Snort 3 es capaz de generar salidas en formato JSON.
- La pila de herramienta *ELK* consume demasiados recursos como para ser instalada en el dispositivo Raspberry Pi junto al Sistema de Detección de Intrusos.
- En un principio se consiguió implementar un modelo del sistema de detección de intrusos para redes inalámbricas, pero debido a la inestabilidad de las herramientas utilizadas para conseguirlo, se concluyó que la solución más estable consistía en configurar el dispositivo en un punto de acceso que inspeccione el tráfico que circula a través de él.
- La herramienta de Kibana ofrece un gran nivel de flexibilidad respecto a la visualización de datos.

5.1. Evaluación de los objetivos alcanzados

En este trabajo se ha logrado implementar un sistema de detección de intrusos utilizando un dispositivo de bajos coste (Raspberry Pi) para analizar el tráfico de red y alertar ante comportamientos malintencionados.

- Investigar las herramientas disponibles para implementar el sistema de detección de intrusos y la herramienta de análisis de logs:

  En este proyecto se han investigado algunas de las herramientas más populares a la hora de implementar un sistema de detección de intrusos, eligiendo finalmente la herramienta Snort debido a su rendimiento y a su bajo consumo de recursos.

- Profundizar en la instalación y configuración del sistema de detección de intrusos seleccionado:

  Se ha investigado el proceso de instalación de Snort 3 y de las configuraciones necesarias para optimizar el software y hacer que funcione correctamente el el dispositivo Raspberry Pi. Además, se ha investigado la personalización de una imagen Debian utilizando CustomPiOS y así reducir el número de componentes que deben ser instalados directamente sobre el dispositivo.

- Profundizar en la instalación y configuración de la herramienta seleccionada para el análisis y visualización de datos:

  En este proyecto se ha investigado el proceso de instalación y configuración de la pila de herramientas ELK. Esta instalación se ha realizado utilizando contenedores Docker. Además, se han

realizado las configuraciones oportunas para que la comunicación sea cifrada, es decir, utilizando el protocolo SSL.

5.2. Trabajos futuros

A continuación, enumero las ideas que pueden llevar a la realización de trabajos futuros que profundicen los temas desarrollados:

- Profundizar en la creación de reglas para Snort. Se puede seguir estudiando el uso de expresiones regulares para la detección de exfiltraciones de información personal, detectar malware u otros usos inadecuados de la red.
- Explorar alternativas hardware con mayores recursos que permitan hospedar una pila ELK completa junto al sistema de detección de intrusos, de forma que se pueda lograr un sistema que funcione con mayor autonomía.
- Buscar sistemas de detección de intrusos compatibles con la captura de datos en redes inalámbricas o encontrar una forma estable de descifrar los datos de la red para que puedan ser utilizados por Snort.
- Vías alternativas de alerta: explorar como configurar Kibana para que permita alertar utilizando Microsoft Teams, Slack, y otras herramientas relacionadas con el entorno profesional.

6. Bibliografía

[1] Docker - ELK 7.6 : Elastic stack with docker compose - 2020, . URL https://www.bogotobogo.com/DevOps/Docker/Docker_ELK_7_6_Elastic_Stack_Docker_Compose.php.

[2] Docker compose overview, . URL https://docs.docker.com/compose/.

[3] El elk stack: De los creadores de elasticsearch. URL https://www.elastic.co/es/what-is/elk-stack.

[4] Get docker. URL https://docs.docker.com/get-docker/.

[5] Instalación de metricbeat – sospedia. URL https://sospedia.net/instalacion-de-metricbeat/.

[6] The raspberry pi guide. URL http://raspberrypi-guide.github.io/.

[7] Ley orgánica 3/2018, de 5 de diciembre, de protección de datos personales y garantía de los derechos digitales. *Protección de datos personales*, page 145–252, 2020. doi: 10.2307/j.ctv17hm980.5.

[8] Reglamento (ue) 2016/679 del parlamento europeo y del consejo de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la directiva 95/46/ce (reglamento general de protección de datos) (rgpd). *Protección de datos personales*, page 17–144, 2020. doi: 10.2307/j.ctv17hm980.4.

[9] Enisa threat landscape 2022, Nov 2022. URL https://www.enisa.europa.eu/publications/enisa-threat-landscape-2022.

[10] Marcin Bajer. Building an iot data hub with elasticsearch, logstash and kibana. In *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 63–68, 2017. doi: 10.1109/FiCloudW.2017.101.

[11] Boštjan Brumen and Jernej Legvart. Performance analysis of two open source intrusion detection systems. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1387–1392, 2016. doi: 10.1109/MIPRO.2016.7522356.

[12] Sedat Görmüş, Hakan Aydın, and Güzin Ulutaş. Security for the internet of things: a survey of existing mechanisms, protocols and open research issues. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 33(4):1247–1272, 2018.

[13] Ryan Heartfield, George Loukas, Anatolij Bezemskij, and Emmanouil Panaousis. Self-configurable cyber-physical intrusion detection for smart homes using reinforcement learning. *IEEE Transactions on Information Forensics and Security*, 16:1720–1735, 2020.

[14] Raspberry Pi Ltd. Raspberry pi OS. URL https://www.raspberrypi.com/software/.

[15] Rudibel Perdigón Llanes and Arturo Orellana García. Sistemas para la detección de intrusiones en redes de datos de instituciones de salud. *Revista Cubana de Informática Médica*, 13(2), 2021.

[16] Carlos G Romero, Luis A Balseca, Fabián Sáenz, and Javier Díaz. Estado del arte en la detección de intrusiones en las redes 802.11 i. *Maskay*, 6(1):35–39, 2016.

[17] Syed Ali Raza Shah and Biju Issac. Performance comparison of intrusion detection systems and application of machine learning to snort system. *Future Generation Computer Systems*, 80: 157–170, 2018.

[18] Guy Sheffer. CustomPiOS. URL https://github.com/guysoft/CustomPiOS. original-date: 2017-06-19T07:35:35Z.

[19] Wireless Sniffer: Snort, Snorby, Barnyard2-Abraza la Web · martes, 13 Diciembre, and 2016 Reply. Wireless sniffer: airtun-ng vs dot11decrypt (wireless IDS) – abraza la web. URL https://abrazalaweb.net/2016/12/wireless-sniffer-airtun-ng-vs-dot11decrypt-wireless-ids/.

[20] Aliya Tabassum, Aiman Erbad, and Mohsen Guizani. A survey on recent approaches in intrusion detection system in iots. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 1190–1197. IEEE, 2019.

[21] Ankit AnandSigNoz Team. Top 14 elk alternatives [open source included] in 2022: Signoz, Nov 2022. URL https://signoz.io/blog/elk-alternatives/.

[22] Abdul Waleed, Abdul Fareed Jamali, and Ammar Masood. Which open-source ids? snort, suricata or zeek. *Computer Networks*, 213:109116, 2022. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2022.109116. URL https://www.sciencedirect.com/science/article/pii/S1389128622002420.

[23] Mark R Warner, Cory Gardner, Ron Wyden, and Steve Daines. Internet of things cybersecurity improvement act of 2017. In *Proceedings 115th United States Congress*, page 1691, 2017.

[24] Yaser Mansour. Snort 3 on oracle linux 8. URL https://snort-org-site.s3.amazonaws.com/production/document_files/files/000/004/026/original/Snort_3_GA_on_OracleLinux_8.pdf?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAU7AK5ITMJQBJPARJ%2F20230607%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20230607T185041Z&X-Amz-Expires=172800&X-Amz-SignedHeaders=host&X-Amz-Signature=55fe2a45a993a4bccbd95cda997db110ebca497c9d1a38b73949a77e2a80a982.