Recogida de Información de un IDS basado en IoT

Autor: Jose Manuel Prieto Gordo

Tutor: Joan Caparros Ramírez

Profesor: Víctor García Font

Máster Interuniversitario de Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)

Seguridad Empresarial

Fecha de entrega: 4 de junio 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Recogida de Información de un IDS basado en IoT*
Nombre del autor:	*Jose Manuel Prieto Gordo*
Nombre del colaborador/a docente:	*Joan Caparros Ramírez*
Nombre del PRA:	*Víctor García Font*
Fecha de entrega (mm/aaaa):	*06/2020*
Titulación o programa:	*Máster Interuniversitario de Seguridad de las Tecnologías de la Información y de las Comunicaciones*
Área del Trabajo Final:	*Análisis de las comunicaciones móviles*
Idioma del trabajo:	*Castellano*
Palabras clave	*IOT, SIEM, IDS*

Resumen del Trabajo

El objetivo de este trabajo es hacer un estudio de los un IDS/SIEM dentro de las tarjetas IOT. Para ello se ha implementado en una Raspberry Pi 3 una sonda Wifi, donde pasará todo el tráfico de red, y se ha realizado un script de que hace de sniffer. El proyecto contiene un sistema de detección de intrusos mediante Suricata, unido a los servicios de DHCP y DNS dentro de placa IOT. Además se ha implantado otro sistema para recoger los datos del IDS y poder ofrecerlos forma visual utilizando la herramienta Elastic Stack. Se han realizado una serie de ataques en la red y se han buscado soluciones para los mismos.
Por último, se verán las conclusiones que han sido adecuadas y muy precisas en el desarrollo adecuado de la sonda, de la configuración del IDS y de la posibilidad de mostrar resultados, pudiendo centralizar en un dispositivo de IOT. No ha sido posible que el análisis y visualización se hagan en el mismo dispositivo.

Abstract

The objective of this work is to make a study of the IDS / SIEM within the IOT cards. To do this, a Wifi probe has been implemented in a Raspberry Pi 3, where all network traffic will pass, and a script has been carried out that acts as a sniffer. The project contains an intrusion detection system using Suricata, together with DHCP and DNS services within the IOT board. In addition, another system has been implemented to collect the IDS data and be able to offer them visually using the Elastic Stack tool. A series of attacks have been carried out on the network and solutions have been sought for them.
Finally, we will see the conclusions that have been adequate and very precise in the proper development of the probe, the IDS configuration and the possibility of showing results, being able to centralize in an IOT device. Analysis and visualization could not be done on the same device.

Dedicatoria/Cita

“No hay temor en el verdadero Amor.”

1Juan 18

“No basta saber las cosas, es necesario practicarlas”

Juan Bosco

Agradecimientos:

Quiero dar las gracias a tres personas muy especiales en mi vida:
- A Eva por el cariño, el apoyo, la paciencia, los ánimos en este tiempo de crisis y de sacrificio.

- A mis padres Jose y Angela por darme unos estudios y por enseñarme a esforzarme para conseguir las cosas en la vida.

Índice

Figuras y tablas

Índice de figuras.

Índice de tablas

1. Introducción

1.1. Explicación detallada del problema

La privacidad ha sido un tema muy importante dentro de la sociedad. En la antigüedad, las casas disponían de ventanas desde las que se podía ver el interior de las casas, pero pronto se dispusieron cortinas para velar por la privacidad de los propietarios.

Esto es así, por la poca inmediatez del asunto de la ciber privacidad. Nadie quiere ser espiado en su propia casa, pero, paradójicamente, no se es consciente de los problemas sobre privacidad que existen en los dispositivos que se utilizan diariamente, sin los cuales nadie sale de casa: Los Smartphones. Gracias a las nuevas tecnologías, todo anonimato virtual está siendo constantemente vulnerado.

Un pensamiento recurrente es 'Eso no importa', pero, entrando en redes sociales como Facebook, es posible ver banners de anuncios personalizados con los intereses personales, o búsquedas recientes de los usuarios. Ahí vemos un uso del análisis de las costumbres de las personas que de manera mecánica y pasiva realizan las empresas para obtener beneficio o ventajas económicas.

Mediante la conexión móvil 3G o 4G, estos datos están cifrados, sin embargo, no siempre se conecta a internet a través de los datos móviles. Mucha gente, al ir a centros comerciales, tiendas o restaurantes, rápidamente se conecta a los wifis públicos que encuentra. Es más, si un teléfono reconoce el SSID de la conexión, se conectará automáticamente a la conexión wifi-abierta.

Pero, ¿cuál es el problema de conectarse a un Wifi Público? Mientras que en un wifi con clave nuestra conexión queda cifrada, en un wifi público eso no ocurre, así que cualquiera puede acceder a todo lo que transmite o recibe ese dispositivo. Por suerte, existe el protocolo HTTPS que cifra la comunicación, haciendo posible la privacidad incluso dentro de una red pública.

Todo aquellos que vaya en sin cifrar, como es el protocolo HTTP, es de dominio público, y se puede obtener mucha información acerca de las costumbres de la persona, pero aun si las aplicaciones usan un protocolo seguro, quedan varios puntos de rastreo de la persona, como son las IPs a las que accede mediante protocolos como el HTTPS o DNS. Pese a la insistencia de los expertos en ciberseguridad en que la utilización de estas redes inalámbricas abiertas es potencialmente peligrosa, la concienciación de la gente está lejos de ser la adecuada.

El objetivo de este trabajo es desarrollar un sistema que permita trazar el perfil de una persona a través de los datos obtenidos en un paquete de red, tanto capturas por una red interna, como los obtenidos a través de escuchas en redes públicas. Esta aplicación debe permitir, de lo capturado y analizado, poder sacar datos para hacer el perfil de la persona, y datos estadísticos del uso de aplicaciones y páginas web.

1.2. Objetivos generales

Aquí se describen los objetivos principales del proyecto y los sub-objetivos que podemos probar para conseguirlo.

1.2.1. Objetivos principales

- **Montar un laboratorio que permita falsear un punto de wifi**.
    - Se instalará y configurará el software que cree el punto de acceso Wifi,
    - Se instalará servicio DHCP para la asignación automática de IPs.
    - Se realizará la instalación sobre una Raspberry Pi 3, será necesario la instalación del sistema operativo, y las configuraciones para acceder a ella de forma remota y segura.
    - Se realizará una prueba de conexión desde otro dispositivo para comprobar que funciona correctamente y se tiene salida a internet.
- **Buscar y Desarrollar una herramienta que permita recopilar datos de las conexiones entrantes**.
    - De instalará el IDS y se configurará para que analice el tráfico enrutado por el AP.
    - Se configurará para que analice y detecte conexiones realizadas a IPs en listas negras.
    - Para comprobar que funciona correctamente probaremos con alguna de las IPs contenidas en la lista negra configurada, y se verá si el IDS ha generado el evento correctamente.
- **Desarrollar un sistema que permita recibir, almacenar y enviar los paquetes capturados**.
    - Integrar la IDS con la SIEM(Security Information and Event Management).
    - Se repetirá la prueba de conexión a una IP de la lista negra, y se comprobará que el evento es correctamente recibido y almacenado en el SIEM.
- **Crear una base de datos donde poder almacenar los resultados obtenidos.**
    - Generar una Base de Datos con las pruebas, y los eventos realizados.
    - Poder incluir otras listas negras
- **Mostrar la información de alguna forma.**
    - Estudiar las posibilidades que la herramienta ofrece en cuanto a análisis y visualización de los eventos ocurridos.

1.3. **Metodología y proceso de trabajo**

Para alcanzar estos objetivos, se divide el proyecto en las siguientes fases:

- Fase inicial, explicación del tema, definición de objetivos y planificación del trabajo con el tutor.
- Fase de investigación, primera toma de contacto, comprensión del funcionamiento general de las redes wifi y ·3G, 4G, sus componentes principales y características, profundiza en la seguridad, estudio de los diferentes algoritmos de seguridad y su evolución. Investigación de las distintas vulnerabilidades conocidas, investigación en revistas, en empresas, en libros enfocados a la seguridad y en blogs de ciberseguridad. Fase de investigación de los diferentes ataques que se pueden realizar a las redes wifi, recogida de información de códigos de GitHub, de diferentes plataformas para realizar auditorías/ataques, publicaciones en universidades, etc.
- Montaje del laboratorio de testeo, configuración de los puntos de acceso que serán atacados, instalación de las herramientas necesarias. y realización de diferentes ataques a redes Wifi. Se concluye con la elaboración del documento en el que se recoge el resultado de las pruebas realizadas.
- Realización o uso de alguna herramienta de captación de datos en la wifi y almacenarlos de alguna manera.
- Análisis de los datos almacenados con alguna herramienta o script sencillo donde se pueda programar y probar.
- Estudio legal de la interceptación de las redes wifi, cuales son sus consecuencias.
- Realización del punto final: Conclusiones.
- Elaboración de la presentación del proyecto

1.4. **Planificación**

Dado que no hay mucho tiempo para el desarrollo de todos los objetivos, en la planificación de tareas, se puede observar cómo coexisten parte teórica y parte práctica para ir avanzando en la implementación en la medida de lo posible.

En el hito de la presentación de la PEC2, una vez realizado la el acces point de los diferentes componentes, intentaré hacer la investigación de que IDS instalar para luego usarlo con un SIEM concreto. Quiero tener en cuenta que ahora con tema del conoravirus le puedo dedicar entre 1 y 2 horas al día al desarrollo del proyecto y de algunos trabajos del máster.

En esta tabla mostramos la planificación pensada para este TFM indicando el día de inicio, el final y el número de días total:

Num	Actividad	Fecha Inicio	Fecha Fln	Días
1	Preparación			
1.1	Documentación previa	21/02/20	24/02/20	1
1.2	Orientación TFM	24/02/20	24/02/20	1
2	PEC1- Plan de trabajo			
2.1	Explicación	24/02/20	25/02/20	1
2.2	Objetivos	25/02/20	26/02/20	2
2.3	Metodología	27/02/20	27/02/20	1
2.4	Tareas a realizar	28/02/20	28/02/20	1
2.5	Planificación temporal	02/03/20	02/03/20	1
2.6	Revisión del estado del arte	02/03/20	03/03/20	1
2.7	Riesgos y soluciones	03/03/20	03/03/20	1
3	PEC2 – Fase de Investigación			
3.1.	Configuración DHCP	04/03/20	09/03/20	3,5
3.2	Captura de datos	10/03/20	13/03/20	3,5
3.3	Topología de Red	13/03/20	19/03/20	4
3.4	Implantación de Suricata	19/03/20	27/03/20	4,5
3.5	Como analizar y visualizar la información	27/03/20	31/03/20	4
3.6	Estudio Legal	27/03/20	31/03/20	4
4	PEC3 – Fase de Implementación			
4.1	Instalación de Sistema Operativo	01/04/20	03/04/20	2
4.2	Instalación del SSH	03/04/20	07/04/20	2
4.3	Instalación del Servicio FTP	07/04/20	10/04/20	3
4.4	Configuración de la red	10/04/20	14/04/20	3
4.5	Configuración del punto de acceso	14/04/20	17/04/20	3
4.6	Configuración de Servicios en IOT	20/04/20	24/04/20	4
4.7	Instalación y Configuración de SIEM	24/04/20	28/04/20	5
5	PEC4- Fase de Conclusiones			
5.1	Uso o Creación un analizador de resultados	27/04/20	04/05/20	5,5
5.2	Dashboard que nos permita acceder a estadísticas	05/05/20	13/05/20	7
5.3	Conclusiones	14/05/20	20/05/20	5
5.4	Redacción de la memoria y presentación	21/05/20	01/06/20	8

Tabla 1- Temporalización

1.4.1. **Diagrama de GANT**

Para llevar a cabo el proyecto se realiza la siguiente planificación inicial del mismo, indicando el día de inicio, el final y el número de días total:

Fecha de comienzo	21/02/20
Fecha de hoy	15/04/2020
Fecha prevista de Finalización	27/05/2020

Elemento PSP	Tarea/paquete de trabajo	Fecha de inicio	Duración	Fecha de finalización
1	Preparación			
1.0	Documentación previa	21.02.2020	1	21.02.2020
1.1	Orientación TFM	24.02.2020	1	24.02.2020
2	PEC1- Plan de trabajo			
2.1	Explicación	24.02.2020	1	24.02.2020
2.2	Objetivos	25.02.2020	2	26.02.2020
2.3	Metodología	27.02.2020	1	27.02.2020
2.4	Tareas a realizar	28.02.2020	1	28.02.2020
2.5	Planificación temporal	02.03.2020	2	03.03.2020
2.6	Revisión del estado del arte	02.03.2020	1	02.03.2020
2.7	Riesgos y soluciones	02.03.2020	1	02.03.2020
3	PEC2 – Fase Investigación			
3.1	Configuración DHCP	04.03.2020	4	09.03.2020
3.2	Captura de datos	10.03.2020	4	13.03.2020
3.3	Topología de Red	14.03.2020	4	19.03.2020
3.4	Implantación de Suricata	19.03.2020	5	25.03.2020
3.5	Como analizar y visualizar la información	27.03.2020	4	01.04.2020
3.6	Estudio Legal	27.03.2020	4	01.04.2020
4	PEC3 – Fase Implementación			
4.1	Instalación de Sistema Operativo	01.04.2020	2	02.04.2020
4.2	Instalación del SSH	03.04.2020	2	06.04.2020
4.3	Instalación del Servicio FTP	07.04.2020	3	09.04.2020
4.4	Configuración de la red	09.04.2020	3	13.04.2020
4.5	Configuración del punto de acceso	13.04.2020	4	16.04.2020
4.6	Configuración de Servicios en IOT	16.04.2020	4	21.04.2020
4.7	Instalación y Configuración de SIEM	21.04.2020	5	27.04.2020
5	PEC4- Fase Conclusiones			
5.1	Uso o Creación un analizador de resultado	27.04.2020	5	01.05.2020
5.2	Dashboard que nos permita acceder a esta	02.05.2020	7	12.05.2020
5.3	Conclusiones	12.05.2020	5	18.05.2020
5.4	Redacción de la memoria y presentación	18.05.2020	8	27.05.2020

Ilustración 1- Planificación inicial de trabajo

Elaboración propia

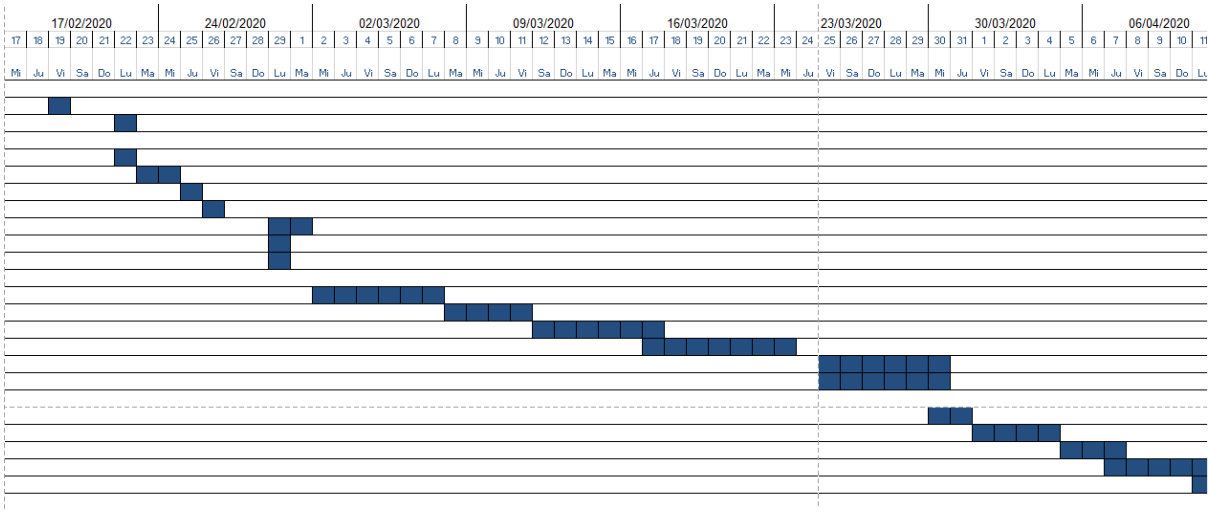

Ilustración 2 Diagrama Gant1
Elaboración propia

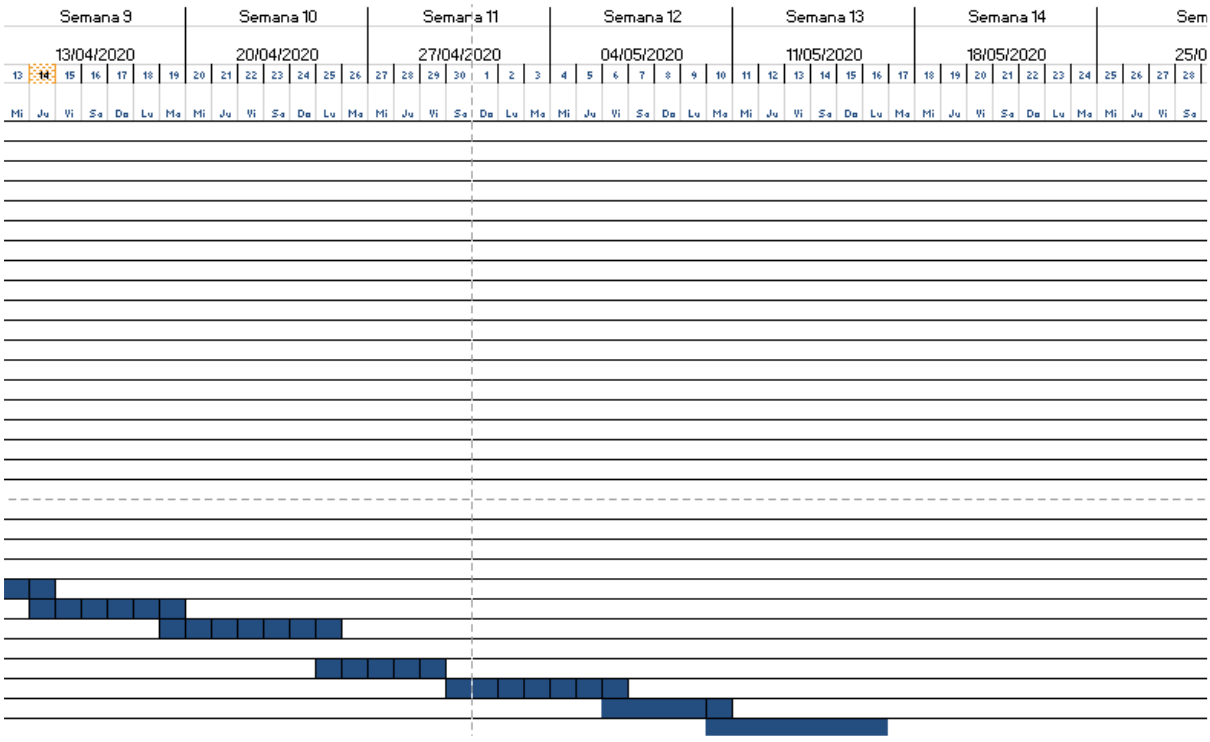

Ilustración 3 Diagrama Gant2
Elaboración propia

1.5. **Presupuesto**

Cabe resaltar la diferencia entre el presupuesto para la investigación y desarrollo de este proyecto, y el presupuesto para implementar la solución final, ya que en el mismo se ha incluido el ordenador de desarrollo del mismo.
Como primera opción se va a intentar desarrollar con una Raspberry Pi 3, pero puede que sea necesario adquirir otros modelos más avanzados (Raspberry Pi 4).

En ese caso, el presupuesto para el desarrollo de la investigación, incluido el hardware sería:

Ordenador donde se desarrolla el proyect: PC i7 16Gb RAM y 500 GB de HD	1200 euros
Raspberry Pi 3	40 euros
Raspberry Pi 4	50 euros
Switch básico TL-SF1005D	10 euros
Conexión a Internet	50 euros/mes
Cableado	10 euros
Edimax EW-7811UN - Adaptador de Red USB	9 euros
Wifi pineapple	99 euros

Tabla 2- Presupuesto

El objetivo del proyecto es obtener un sistema final de bajo coste, el precio total del laboratorio a crear puede estar entre 80-100 euros sin contar la conexión a internet.

1.6. **Estado del arte**

Según el libro de Hacking práctico de (Ramos, 2013) podremos realizar un análisis adecuado de la redes wifi y nos dará la posibilidad de analizar las tramas de estas.
Otra parte importante a tener en cuenta es la suite conocida como Aircrack-ng, son un conjunto de herramientas dedicadas a la auditoría y vulneración de redes inalámbricas. dentro de esto puedo usar el paper (Mathy Vanhoef, 2017) y existe una wiki dentro del proyecto Aircrack-ng con mucha información actualizada.
La Herramienta http://fruitywifi.com/index_esp.html, se podrá usar como una herramienta de codigo abierto para auditoria de redes wireless. Permite desplegar rápida y fácilmente ataques avanzados utilizando la interfaz web o enviando mensajes.
Otra posible herramienta para usar dentro del proyecto es Pandora FMS, es de pago, pero su idea es integrar el análisis de tráfico a las actividades de monitorización, optimización y resolución de problemas.
Desde el punto de vista de hardware podría usar Wi-Fi Pineapple es un dispositivo de hardware creado para buscar todo tipo de vulnerabilidades en las redes inalámbricas de manera que podamos saber si existe alguna debilidad que pueda permitir a un atacante colarse en una red y poner en peligro todos los sistemas y los datos que viajan por esta.
También existen herramientas más generalistas que podemos llegar usar como Wireshark, que es un analizador de protocolos utilizado el análisis buscar problemas en redes de comunicaciones, para desarrollo de software y protocolos, y como una herramienta en clases de FP y universidades. Tcpdump es una herramienta para línea de comandos cuya utilidad principal es analizar el tráfico de la red. Scapy, es una herramienta de manipulación de paquetes para redes, se puede falsificar o decodificar paquetes. Hping, es un generador y analizador de paquetes gratuito para el protocolo TCP / IP. TcpReplay, herramientas para el análisis y manipulación de paquetes de datos, muy utilizada principalmente por administradores de sistemas y redes.
Se pueden usar tener en cuenta herramientas de ataque a redes, como MitMFramework, herramienta que proporciona una gran cantidad de ataques, simplificando, gracias a los plugins, el número de herramientas que se deberían disponer para llevar a cabo los ataques.
Otra opción para contemplar es el uso de IDS. Los IDS de código abierto que destacan en la actualidad son Snort y Suricata. Snort inició su desarrollo a finales de los 90, y desde entonces se ha convertido en un referente. Suricata es más actual y fue diseñado desde un principio para aprovechar las CPUs modernas multinúcleo con procesamiento paralelo. Aunque Snort no se ha quedado atrás y usa también procesamiento en paralelo.

1.7. **Análisis de Riesgos**

Pensando en los posibles riesgos de mi TFM:

Para cada riesgo se ha pensado como puede afectar y posibles soluciones

- Riesgo 1: Proyecto demasiado grande: La definición inicial del proyecto cubre un ámbito demasiado grande para la dedicación teórica y el calendario previstos.
    - Posibles soluciones: Simplificar en caso necesario, descartar temas no incluidos en el ámbito del trabajo, eliminar tareas superfluas, recortar el ámbito en último caso.
- Riesgo 2: Falta de información suficiente respecto a alguna de las áreas: Alguno de los temas de estudio o alguna herramienta no está suficientemente documentado.
    - Posibles soluciones: Buscar documentación en Internet, manuales de uso, vídeos de herramientas, libros, foros de seguridad.
- Riesgo 3: Caso práctico demasiado complejo: El caso práctico elegido es muy complicado y limita el tiempo dedicado para su desarrollo adecuado.
    - Posibles soluciones: Limitar el ámbito del análisis del caso práctico al mínimo necesario. En último caso hacer un caso práctico más sencillo.

2. Fase de Investigación

Durante esta fase se realizará un estudio del estado del arte, buscando las herramientas que serán utilizadas en la implementación final.

2.1. Qué es un IDS y por qué usar IoT.

2.1.1. ¿Qué es un IDS o Sistema de Detección de Intrusos?

Dentro del mundo de la seguridad, un Sistema de Detección de Intrusos (IDS) es un sistema de seguridad que nos permite monitorear un entorno para detectar eventos que pueden ser inapropiados o que pueden perjudicar a un sistema.
Hay diferentes tipos de IDS, tenemos NIDS(Sistema de detección de intrusos en la red).Esta modalidad de IDS usa un hardware de que únicamente funciona para la revisión de forma permanente de la red, en busca de movimientos extraños. También existen los HIDS(Sistema de detección de intrusos en el host), es un software que busca dentro de un equipo capturar paquetes de la red que circulan por el Host, intentando detectar posibles intrusiones dentro del mismo. Por otro lado también tenemos el IPS. Básicamente un IPS son IDS pero proactivos, que se encargan de realizar acciones automáticas ante los posibles eventos que van surgiendo.

2.1.2. ¿Cómo actúa un IDS?

Existen algunas técnicas específicas que usan los IDS en sistemas informáticos, y que pueden ser muy útiles, algunas son, almacenar lo que ocurre, el IDS se encarga de llevar un registro detallado de todos aquellos paquetes de información que ocasionaron una señal de alerta. El almacenamiento se puede realizar en diferentes entornos de Base de Datos. También puede verificar dispositivos de la red, el IDS puede ser capaz de configurar y ver que ocurre ante distintos eventos. Otra de las opciones que tiene un IDS es de mostrar la información y poder enviar alertas al usuario.
Un IDS se encarga de buscar los paquetes y segmentos de la red que monitoriza, buscando coincidencias con reglas que se saben sospechosas. Un IDS tiene un conjunto de sensores, también llamados sondas, que se ubican en diferentes puntos de la red. Las sondas se encuentran configuradas para que actúen sin que modifiquen en nada la comunicación, por lo que no requiere ningún cambio de la infraestructura que se va a monitorizar.
La principal ventaja de los sistemas IDS es, gracias a las sondas, es toda la infraestructura de red y no dispositivos por separado los que son monitorizados. Esto permite conseguir la detección de la mayoría de los ataques y buscar relaciones entre ellos. Esta transparencia, nos ayuda a no ser detectados por parte de los atacantes.

2.1.3. Unimos IDS e IoT

El concepto de IoT es el de conectar cualquier cosa a Internet, desde los objetos más cotidianos que podamos encontrar en nuestro hogar, nuestro coche, elementos de un hospital, un hotel y por supuesto elementos de la industria. Consigue la interconexión de dispositivos y objetos a través de una red (bien sea privada o Internet), dónde todos ellos podrían ser visibles e interactuar. Está tecnología usa dispositivos pequeños, de bajo consumo, tienen sensores y actuadores, deben de poder comunicarse de alguna forma (Ethernet, Wifi, 3G/4G/5G), tienen precios muy bajos. Hay opciones mejores para hacer un IDS que usar un dispositivo de IoT, pero se puede crear una solución funcional y para una PYME o una cosa, cumpliría perfectamente toda la funcionalidad. Gracias a la buena potencia y al bajo coste de este dispositivo, tenemos la posibilidad de hacer proyectos muy interesantes sin necesidad de dejarnos una gran cantidad de dinero. Se usará una Raspberry Pi 3 tipo2, es la que tengo en casa en estos momentos, como un sistema de detección y prevención de intrusiones, es decir, la Raspberry Pi revisará todo el tráfico de nuestra red, y comprobará que nadie nos está intentando atacar.

2.1.4. Modos de Funcionamiento de las Sondas.

Existen dos modos de funcionamiento para estos dispositivos sonda. El primero es instalar todo un equipo, es decir que el IDS se encuentra en un host que actúa como proxy de red. Esta sonda analizará todo el tráfico que atraviese sus interfaces de red. Dependiendo del hardware, en análisis que va a realizar el IDS puede producir retrasos en la red. Se tendrán que realizar pruebas de rendimiento para conocer la velocidad máxima a la que podremos transferir datos. Debemos tener en cuenta el cuello de botella del puerto Fast-Ethernet de la Raspberry Pi. Este tipo de retardos puede ser inadmisible en algunas redes, como puede ser el entorno médico o industrial.

El segundo modo es el denominado “port mirroring”. Para usar esta técnica, se va a necesitar un hardware especifico que permita hacerlo. A esta red se conecta un dispositivo sonda cuya interfaz de adquisición se encuentra en modo “promiscuo”, aceptando todos los paquetes de red sin importar destino, para su posterior análisis. Los switches con la funcionalidad port mirroring, se pueden configurar para que todo el tráfico que pase se reenvíe entre las conexiones que tiene, de esta forma, si todo el tráfico que va y viene de internet lo reenviamos donde tenemos conectada la sonda IDS, todo el tráfico de acceso a internet se puede monitorear y analizar de forma adecuada.

En caso de que la ejecución del IDS junto con el AcessPoint en el mismo dispositivo ralentice demasiado la conexión a internet, se considerará mejor una solución la basada en un switch con port mirroring. Pero debido al “estado de alarma” no se tiene el hardware necesario ni los medios para conseguirlos, por este motivo se centrará en la primera solución, donde una Raspberry Pi hará de Proxy, IDS, y se buscará optimizarla al máximo para evitar cuellos de botella.

2.2. **Captura de datos.**

En primer lugar, en todo sistema de detección de intrusos basado en red tenemos los dispositivos Sonda. Las sondas son los encargados de la captura y procesado del tráfico de red.
La placa que se usará para contener el software de detección de intrusos basado en red es la Raspberry PI 3 Modelo 2. Fue una de las cosas que más motivo para el desarrollo de este TFM.

2.2.1. **Elección del Motor de IDS sobre una Raspberry pi 3.**

Existen varios IDSs de software libre que podrían utilizarse, pero hay que destacar dos productos sobre el resto: Snort y Suricata.

Snort[1]

Snort ha sido el motor de IDS más usado durante años. Este motor tiene una enorme comunidad de usuarios soportándolo y, muchos usuarios que usan sus reglas.
Snort se ha convertido en la tecnología de prevención y detección más ampliamente extendida y confiable del mundo. En año 2005 la revista SC Magazine [2]afirmó que el éxito de Snort se debe al la gran cantidad de usuarios y desarrolladores que hay detrás y los eficiente que es a la hora de detectar incidentes de seguridad que otros motores IDS de la competencia no son capaces de responder.
Snort utiliza un lenguaje basado en reglas que combina beneficios de inspección basados en firmas, protocolos y anomalías. Con su velocidad, potencia y rendimiento, Snort se ha convertido en uno de los IDS más utilizado del mundo.
Snort se dedica a escuchar el tráfico de red y buscar reglas en el flujo de datos que puedan indicar un incidente de seguridad en la red que está monitorizando. Las reglas están configuradas para realizar una determinada acción. Esta acción puede ser pasiva, alertando del incidente, o activa, aplicando acciones en tiempo real. Durante su instalación ya nos provee de cientos de filtros o reglas para backdoor, ddos, finger, ftp, ataques web, CGI, escaneos Nmap…. La comunidad de Snort nos da una base de reglas existentes para casos específicos ya estudiados, y nos permite escribir o modificar sus propias reglas de acuerdo con requisitos particulares de la red. Resulta posible escribir reglas complejas para identificar casi cualquier tipo de tráfico que atraviesa la red. Existen gran cantidad de repositorios de reglas de Snort que se han creado a lo largo de varias décadas. Además, podemos encontrar mucha documentación sobre Snort gracias a su comunidad, hay actualizaciones constantes sobre la aplicación y sobre las reglas.
Snort en la versión estable actual(2.9) no es multihilo, de forma que solo usa un núcleo de los 4 posibles que tiene la Raspberri Pi 3,así que muchos se quedan sin uso. Esto puede ser un problema, hace que la arquitectura de Snort tenga problemas con mucho trafico. La versión 3 de Snort ya está arreglando los problemas de multihilo, pero de momento está en versión Beta

[1] https://www.snort.org/
[2] https://www.businesswire.com/news/home/20050223005435/en/Sourcefire-Recognized-Intrusion-Solution-SC-Magazine

Suricata[3]

Suricata es un sistema de detección de intrusos, al igual que Snort, basado en código abierto con el fin de construir una evolución en los sistemas de IDS. El objetivo de sus creadores, OISF(Open Information Security Fundation), es generar ideas nuevas de seguridad dentro de la industria de detección de intrusiones de IDS de software libre.

Con la ayuda económica del Departamento de Seguridad Interna de los Estados Unidos, se creó una alternativa multihilo a Snort para ayudar a proteger las redes contra intrusiones de seguridad que se pensaba que otros IDS no eran capaz de realizar, ya que puede soportar sistemas multi-core y multiprocesador de alto rendimiento. Las principales mejoras de un diseño de este tipo es que ofrece mayor velocidad y eficiencia en el análisis de tráfico de red y también puede ayudar a dividir la carga de trabajo de IDS / IPS en función de las necesidades de procesamiento.

Suricata sigue la idea Snort, para que la gente pueda hacer la migración a este producto, emplea las mismas reglas y formatos de salida que Snort. pero es más eficiente con el hardware moderno e incluye soporte para lenguaje de scriptting que puede utilizarse tanto en la salida como en la definición de reglas que ayuda a la detección de amenazas más complejas.

Un añadido de Suricata, con respecto a Snort es es la extracción de archivos. Esto permite extraer todos los archivos analizados por una regla para poder estudiarlos posteriormente.

Hay que reconocer que Suricata sigue siendo un producto “nuevo” y menos extendido que Snort, pero nueva la tecnología está ganando en comunidad y usuarios. La mejora en el rendimiento, la compatibilidad con IPv6 , los modelos estadísticos de detección de intrusos y la aceleración GPU son, entre muchas cosas, sus puntos fuertes y a tener en cuenta a la hora de elegir nuestro IDS particular.

Maltrail[4]

Es una herramienta de código abierto hecha en Python, diseñada para analizar el tráfico de red para detectar y registrar posibles amenazas. La diferencia con Snort o Suricata radica en la clasificación de la amenaza, la fuente de la amenaza (si la dirección IP no está en regla en diferentes fuentes abiertas), la inteligencia para identificar actividades sospechosas, la identificación de malware que intenta llamar a casa y el panel de detección que muestra La pantalla es particularmente interesante, o al menos complementa la pila de paneles web de varias herramientas diseñadas para monitorear la seguridad.

El tipo de implementación es el mismo que cualquier IDS, es decir, una sonda de detección que envía eventos a un solo servidor, por lo que una arquitectura distribuida es completamente factible, e incluso las sondas existentes pueden usarse para analizar el tráfico utilizando otras herramientas de código abierto. Para integrar otro proceso. Se ha podido verificar que no aumenta excesivamente los recursos necesarios de la computadora, por lo que en el tráfico "normal", esta es una opción interesante.

[3] https://suricata-ids.org/
[4] https://github.com/stamparm/maltrail

Elección del IDS.

Vamos a hacer una tabla[5] de comparación de los datos de Snort y Suricata:

	Snort	Suricata
Desarrollador	Sourcefire, Inc. (now Cisco)	Open Information Security Foundation (OISF)
Disponibilidad	Desde 1998	Desde 2009
Lenguaje codificado	C	C
Sistema operativo	Multiplataforma	Multiplataforma
Threads	Un hilo	Multi-hilo
Soporte IPv6	Si	Si
Reglas Snort	Si	Si
Reglas de amenazas emergentes	Si	Si
Formato de registro	Unificado	Unificado

Tabla 3- Snort vs Suricata

Tras un análisis de los IDS estudiados , a son tecnologías que actualmente son similares, y los dos usas las reglas básicas de Snort. Pero Suricata está avanzando más rápidamente que Snort. En lo que respecta al TFM, se decide emplear Suricata como motor de IDS dado que su capacidad multi-hilo y escalabilidad creemos que nos dará un mejor rendimiento en el hardware del dispositivo Raspberry Pi 3 Modelo 2 del que disponemos. Esta placa de IoT dispone de un procesador ARM de cuatro núcleos que podrán ser utilizados de manera concurrente por este motor.

Unido a Suricata, se va a probar una herramienta opensource, hecha en Python, destinada a analizar el tráfico de red con el fin de detectar y registrar posibles amenazas. Hasta aquí todo bien, y seguro que diréis: pues otro Snort o Suricata, ¿no?. Pues la respuesta es: parcialmente sí.

La diferencia es la categorización de las amenazas, las diversas fuentes de las que tira (si la dirección IP tiene mala reputación en diferentes fuentes abiertas), la inteligencia para identificar actividad sospechosa, la identificación de malware que intenta llamar a casa, así como la presentación de panel de detección que muestra, me ha resultado especialmente interesante, o cuanto menos, complementario a la pila de paneles web de diversas herramientas destinadas a monitorizar la seguridad.
En este proyecto se va a probar también Maltaril unido a Suricata, ya que podemos encontrar mucho puntos positivos, es muy simple la configuración (archivo único, pocas opciones), la documentación existente es muy completa y las ventajas de la herramienta son más que suficientes para un dispositivo de IOT ya que se puede monitorear en una interfaz o en todas las máquinas de sensores y tiene una impresionante interfaz gráfica.

[5] https://tacticalflex.zendesk.com/hc/en-us/articles/360010678893-Snort-vs-Suricata

2.3. **Topología de Red**

La topología de red a utilizar para implantar un IDS puede ser muy extensa. En lo que se refiere a este TFM y en una red de laboratorio o de trabajo, hemos plantado dos posibles soluciones, que han salido en el apartado anterior.

Cada escenario va en función de decidir si el IDS va a realizar tareas de IPS o no. Y en mi caso de poder disponer del hardware adecuado Esto influye de forma importante en la topología.

Si la sonda va a actuar como IDS/IPS debería estar situada en un punto de red que pueda hacer de proxy y todo el pueda pasar por ella de forma que puede denegar, permitir, borrar paquetes de red, etc dependiendo lo que ocurra, en este caso podemos decir que la sonda hace funciones de Gateway.

En el caso de solo actuar como IDS, la sonda se puede situar sin funciones de Gateway. Este modelo tendría que utilizar técnicas como port mirroring para enviarle el tráfico a analizar.

Se va a analizar la topología de red se va a usar, la primera, viendo ventajas en inconvenientes de usarla.

IDS/IPS con función de Gateway

La idea de este esquema es que la sonda que se crea se coloca en medio entre los dispositivos de la red y el acceso a internet de forma que puede detectar todo tráfico de red y poder descartar el tráfico no adecuado y poder eliminarlo en tiempo real y de forma transparente para toda la infraestructura de red.

Podemos montar esta topología de red únicamente con los siguientes dispositivos de red, usando u Router acceso a internet y una Raspberry pi 3 con sonda IDS configurada a la vez como Access Point:

La topología de red será:

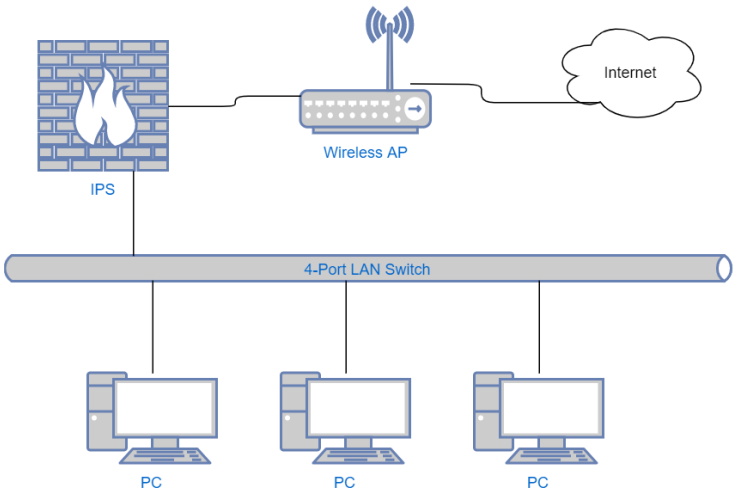

Ilustración 4 - Topología de red
Elaboración propia

La ventajas de usar este modelo de IDS son, poder usar las funciones de IPS, actuando sobre un posible ataque y quitar los paquetes que fueran problemáticos. También permite el bloqueo a páginas con contenido no adecuado, puede actuar como Firewall, no es necesario un switch con port mirroring. Con este modelo de Raspberry, que tiene dos interfaces de red, una ethernet y otra wifi, se podría que configurar como punto de acceso, reduciendo así en número de dispositivos de red.
Los inconvenientes de usar este modelo es que se pueden crear cuellos de botella, sobre todo si el dispositivo no es muy potente como es nuestro caso. El montar la Raspberry pi como punto de acceso, le asigna una funcionalidad más al dispositivo que puede hacer que el rendimiento del sistema no sea el mejor.
Hemos de adaptar a nuestro entorno de trabajo, de una forma muy fina, la base de datos de patrones de posibles ataques. Quizás valga la pena perder tiempo el tiempo que sea necesario en esta parte de análisis, ya que eso nos ahorrará después muchos análisis de falsas alarmas y evitar problemas una vez tengamos todo configurado. Es muy probable que se genere un cuello de botella y disminuya la velocidad de acceso a internet. No obstante, se realizarán diferentes pruebas durante la puesta en marcha del proyecto.

2.4. **Monitoreo con MalTrail**

Maltrail es una herramienta de detección de tráfico malicioso, basado en la arquitectura Tráfico-Sensor-Cliente-Servidor, que utiliza listas negras públicas que contengan direcciones web sospechosas y/o malware, junto con las direcciones estáticas recopiladas por diversos informes antivirus.

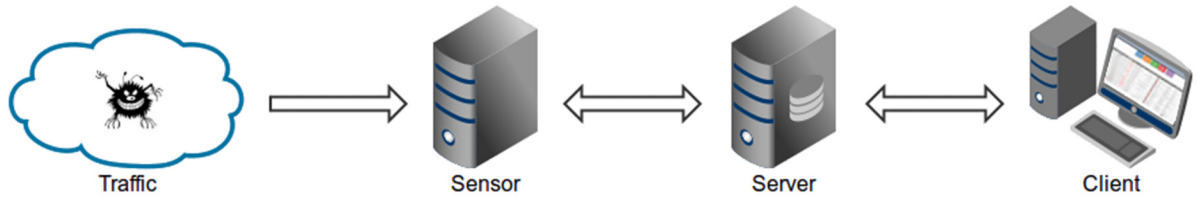

Ilustración 5 - Arquitectura Maltrail
Via Github

Maltrail es un componente independiente que se ejecuta en un nodo o máquina de monitoreo a través del cual pasa el tráfico. Si el resultado es positivo, los detalles del evento se envían al servidor y se almacenan en el directorio de registro apropiado. Si el sensor se ejecuta en la misma computadora que el servidor, los archivos de registro se almacenan en el directorio de registro local; de lo contrario, se envían al servidor remoto a través de mensajes UDP. Para el servidor, es responsable de almacenar los detalles del evento y proporcionar soporte de back-end para la aplicación web donde se mostrarán todos los informes. Por definición, el servidor y el sensor se ejecutan en la misma computadora.
Maltrail es un sistema de detección de tráfico malicioso que utiliza listas públicas (negras) que contienen rutas maliciosas y / o generalmente sospechosas, así como rutas estáticas compiladas de varios informes AV y listas personalizadas definidas por el usuario, donde la ruta puede ser arbitraria El contenido en el nombre de dominio (por ejemplo, zvpprsensinaix.com para el malware Banjori), URL

(por ejemplo, hXXp: //109.162.38.120/harsh02.exe para el archivo ejecutable malicioso conocido), dirección IP (por ejemplo, 185.130.5.231) Para atacantes conocidos) o valores de encabezado HTTP User-Agent (por ejemplo, sqlmap para inyección automática de SQL y herramientas de raspado de bases de datos). Además, utiliza heurística avanzada (opcional) para ayudar a descubrir amenazas desconocidas (por ejemplo, nuevo malware).

2.5. **Implantación de Suricata**

Vamos a poner en marcha un IDS Suricata[6] en modo Gateway en un dispositivo de IoT, uno de los objetivos del proyecto será demostrar que una Raspberry pi es un dispositivo que aun siendo de bajo coste, puede funcionar como sonda IPS.

2.5.1. **Características**

Vamos a ver de manera simple algunas características de Suricata.
El procesamiento multi-hilo es una de las características más importantes de Suricata que permite que múltiples procesos / hilos se ejecuten simultáneamente. Podemos asignar el número de subprocesos por CPU / núcleos y qué subprocesos. De esta manera, es capaz, entre otras cosas, de procesar una gran cantidad de paquetes simultáneamente, aumentando así el rendimiento. Tiene cuatro módulos de hilo: captura de paquetes descifrador, rastreo el flujo de secuencias, inspección de la capa de aplicación y detección / comparación de firmas procesamiento de eventos y salida de alertas.
La Detección de protocolos es otra característica importante. Además de los protocolos IP, TCP, UDP e ICMP, Suricata tiene palabras clave para otros protocolos como FTP, HTTP, TLS, SMB. De esta manera, podemos escribir reglas independientemente del puerto que use un protocolo, ya sea de forma predeterminada o no, ya que se detecta automáticamente.
Las reglas “tipo Snort”, para la detección de amenazas de seguridad se utilizará las reglas que hay desarrolladas por la comunidad y son compatibles con Snort.
También almacena estadísticas y análisis de rendimiento, estas estadísticas se vuelcan el en un archivo.
Suricata incluye una almacenamiento del log HTTP. Suricata, independientemente de las alertas, vuelca todas las peticiones HTTP. El registro de estas peticiones se almacena en formato de Log Apache.
Otra característica es el análisis de ficheros, puede analizar ficheros de imágenes, documentos, etc dentro del trafico de la red, previamente tendremos que descargarlos.
La salida de información puede ser un fichero JSON(basado en XML), que puede unirse con otro tipo de aplicaciones y para poder generar informes.
El control de peticiones DNS también es una característica de Suricata. Se usa para tener un control de las llamadas DNS que se realizan desde nuestra red.

[6] https://suricata.readthedocs.io/en/suricata-5.0.2/

2.5.2. **Reglas Suricata**

Las reglas Suricata especifican qué eventos tiene que detectar sobre el tráfico de red monitorizado. Ya se ha comentado que Suricata y Snort utilizan el mismo formato en la definición de reglas de detección.

Vamos a comenzar viendo un ejemplo de una regla:

```
drop tcp $HOME_NET any -> $EXTERNAL_NET 80 (content:"|00 20 2F a5|"; msg:"example rule";)
```

La estructura de una regla va a ser:

Cabecera de la regla	Opciones de la regla

La forma de definir la cabecera de la regla es:

Rule Action	Protocol	Source IP	Source Port	Direction	Destination IP	Destination Port
drop	tcp	$HOME_NET	any	->	$EXTERNAL_NET	80

Tabla 4- Regla Suricata

Las acciones que se pueden realizar son:

Nombre	Disponible en Suricata	Disponible en Snort
alert	Si	Si
log	-	Si
pass	Si	Si
activate	-	Si
dynamic	-	Si
drop	Si	Si
reject	Si	Si
sdrop	-	Si

Tabla 5- Acciones Suricata

- alert: Genera una alerta usando el método de alerta seleccionado y posteriormente loggea el paquete.
- log: Comprueba el paquete.
- pass: Ignora el paquete.
- activate: Alerta y luego activa otra regla dinámica.
- dynamic: Permanece ocioso hasta que se active una regla, entonces actua como un inspector de reglas.

Los protocolos que se permiten usar:

Nombre	Disponible en Suricata	Disponible en Snort
tcp	Si	Si
udp	Si	Si
icmp	Si	Si
ip	Si	Si
http	Si	-
ftp	Si	-
tls	Si	-
smb	Si	-
dns	Si	-

Tabla 6 - Protocolos suricata

La dirección indica hacia dónde va la comunicación con los siguientes símbolos (<- , <> , ->).

La segunda parte de la regla son las opciones. Se definen entre paréntesis y cada opción está separada por punto y coma (;), luego de cada opción se puede definir la clave y su valor separado por dos puntos (:).
Hay varios tipos de opciones:

- Meta-data. Proporciona la información sobre la regla, pero no tenga alguno afecta durante la detección.
- Payload. Busca patrones (firmas) dentro de la carga útil del paquete.
- Non-Payload. Busca patrones dentro de los demás campos del paquete, que no sean carga útil (por ejemplo, la cabecera).
- Post-detection. Permite activar reglas específicas que ocurren después de que se ejecute una regla.

Seguidamente se describen las principales opciones de las reglas:

- msg. Informa al motor de alerta que mensaje debe de mostrar. Los caracteres especiales de las reglas como : y ; deben de colocarse dentro de la opción msg con el carácter \.
- flow. Se usa junto con los flujos TCP, para indicar qué reglas deberían de aplicarse sólo a ciertos tipos de tráfico.
- content. Permite que Snort realice una búsqueda sensitiva para un contenido específico del payload del paquete.
- referente. Define un enlace a sistemas de identificación de ataques externos.
- classtype. Indica qué tipo de ataques intentó el paquete.

Ejemplo de una regla completa que bloquea la UOC:

```
drop tcp any any -> any any (msg:"UOC es Bloqueada"; content:"uoc.edu";
http_header; nocase; classtype:policy-violation; sid:1;)
```

2.5.3. Otros servicios a utilizar

Como se ha comentado en puntos anteriores, vamos a utilizar una topología de red donde nuestra Raspberry pi hace de punto a acceso y proxy. Esto quiere decir que Suricata puede hacer tareas de IPS, y puede usar la acción “drop” sobre aquellos paquetes de red identificados como peligrosos.
Pero puede que nuestra sonda pueda necesitar algunos servicios adicionales para ayudarse a realizar toda la gestión del tráfico de red.

Servicio DNS

Instalar un servicio de DNS en la Raspberry pi, es una opción a valorar. De esta manera podríamos controlar el acceso a internet de nuestra red y de las peticiones que hacen nuestros equipos locales. Al instalar un servicio más es muy probable que la sonda baje el rendimiento.
Además, es posible que utilizando el servicio hacer restricciones de IP, por horario, tener listas blancas y negras, etc.

Servicio DHCP

Al instalar el servicio DHCP podríamos establecer IP fijas a todos los equipos de nuestra red, siempre y cuando sea una red pequeña. Esto haría que la seguridad de la red fuera mucho mayor y es algo interesante a tener en cuenta. Como el servicio de DNS esto puede afectar al rendimiento de la sonda, pero es algo que tendremos que comprobar en la fase siguiente.
Un problema que tenemos es que el servicio de DHCP tiene estar continuamente funcionando por si un equipo nos pide una IP en cualquier momento, esto puede afectar al calentamiento de la sonda, pero habrá que hacer pruebas.

2.6. Como analizar y visualizar la información

Hasta ahora se ha analizado lo que son los sistemas IDS y como se pueden utilizar para capturar el del tráfico de red. La información capturada se guarda de una forma poco relacional, en ficheros tipo Log difíciles de leer y analizar.
Es necesario realizar una aplicación que se encargue de pasar estos datos a información útil y actualizada donde se pueda analizar por personas que incluso no están familiarizadas con los sistemas de información.
Por esta razón, junto a la implantación de un IDS, suele ser necesaria la instalación de un SIEM.
El significado del término SIEM[7] es una combinación de las siglas SIM y SEM. Las soluciones SIEM son una solución mezclando las categorías de productos como SIM (Security Information Management) y SEM (Security Event Manager). La tecnología SIEM nos va a proporcionar un análisis en tiempo real de las alertas de seguridad generadas por el hardware y software de red, que hemos creado en el apartado del IDS. Las soluciones SIEM son muy variada y hemos analizado las mejores soluciones para nuestro proyecto.

[7] https://www.incibe-cert.es/sites/default/files/contenidos/guias/doc/certsi_diseno_configuracion_ips_ids_siem_en_sci.pdf

El término SIEM (Security Information and Event Management) nos van a dar muchas funcionalidades como poder analizar la información de la red, y de los equipos de seguridad, nos permitirá la gestión de identidades y accesos, la gestión de vulnerabilidades y control de amenazas, gestión de logs y base de datos.

El objetivo es monitorizar y controlar los privilegios de usuario y de servicio, gestionar inventario y otros cambios de configuración del sistema, así como suministrar datos para auditoría de eventos, revisión y respuesta a incidentes. Los SIEM están diseñados para recoger eventos de seguridad buscando de una amplia clase de fuentes de una organización, como paginas y datos de servidores. Una vez que el SIEM tiene los eventos, procesa los datos para estandarizarlos, lleva a cabo el interpretación de los datos "normalizados", genera alertas cuando detecta acto anómalo y produce el resultado que necesitan los administradores. Algunos tipos de SIEM también pueden desempeñarse para aislar las actividades maliciosas. Esta información y las reglas IDS se va guardando en los diferentes Logs que proporciona Suricata.

2.6.1. Análisis de Tecnologías SIEM para dispositivos de IoT

Basándonos en un análisis de los 10 mejores SIEM de 2020[8], vamos a ver cuales son los más usados y por qué son los mejores.

OSSIM, de AlienVault, es una de las herramientas SIEM de código abierto más populares disponibles. Este es un programa altamente rico en funciones con utilidades de recopilación, normalización y correlación de eventos. Cuenta con capacidades de registro y monitoreo a corto plazo, así como evaluación de amenazas a largo plazo y respuestas automáticas integradas, análisis de datos y archivo de datos. La configuración requiere mucha mano de obra, particularmente para Windows, y personalizar el programa según sus necesidades requiere una inversión considerable de tiempo.

OSSEC es una buena elección. Este programa se conoce como una solución de detección de intrusos de código abierto y es popular entre los usuarios de macOS, Linux, BSD y Solaris. Lo mejor de este programa es que cuenta con modos de agente de servidor y sin servidor. Sus utilidades de análisis de registro son competentes y cubren numerosas fuentes, incluidos servidores de correo, FTP y bases de datos. Si desea monitorear múltiples redes desde un solo punto, entonces OSSEC es una opción viable. La comunidad detrás de OSSEC es solidaria y bien estructurada. Puede unirse a la lista de correo o incluso unirse al canal Slack, lo que facilita la colaboración con otros usuarios. El único problema es que las actualizaciones de software pueden ser un poco perjudiciales con esta herramienta.

Sagan es una herramienta SIEM gratuita que ofrece análisis y correlación de registros en tiempo real. También es útil para la normalización de registros , la ejecución de scripts en la detección de eventos, alertas en tiempo real, soporte de registro de varias líneas y monitoreo automático de firewall. Esta es una herramienta ligera con arquitectura de subprocesos múltiples, que le permite utilizar todas las CPU / núcleos para el procesamiento de registros en tiempo real. Es compatible con varias consolas de

[8] https://www.dnsstuff.com/free-siem-tools

seguridad gráfica como BASE, Snorby y EveBox. La principal desventaja de Sagan es que no es especialmente fácil de usar.

Elastic Stack, también conocido como ELK, está compuesto por varias herramientas SIEM gratuitas. Elasticsearch, que ya se ha mencionado en esta guía, es el motor de análisis y búsqueda distribuido basado en JSON. Kibana, otra herramienta incluida en la pila es una ventana en la pila elástica. Beats es la plataforma responsable de los cargadores livianos que envían datos desde máquinas de borde, mientras que Logstash es el canal de recolección de datos. En combinación, estas herramientas ofrecen una solución SIEM más completa que Elasticsearch solo. Aunque este conjunto de herramientas es impresionante, Elasticsearch está en el corazón del conjunto y ofrece la más notable de las utilidades de la pila.

La **ELK Stack** puede ser una solución SIEM de alta calidad. Sus filtros de entrada incluyen capacidades de monitoreo de infraestructura de nube pública que las otras soluciones de código abierto no pueden igualar, por tanto, me he decidió por no ser de las más complejas de configurar y además de tener una gran cantidad de opciones.

Las razones de esta elección son que es un sistema que no solo permite analizar datos del IDS Suricata, sino que puede utilizarse para recoger datos de todo tipo de logs y tratarlos. Permite una monitorización de datos fácil y amigable, adaptable a lo que necesite el usuario, puede instalarse en una Raspberry pi, y tiene una gran comunidad en internet.

2.6.2. Partes de la ELK stack

ELK es la sigla para tres proyectos de software libre: Elasticsearch, Logstash y Kibana. ELK stack[9] es una conjunto de proyectos que, aun siendo desarrollados por Elastic y aunque independientes entre sí, fueron concebidos para proporcionar una relación perfecta entre ellos para solventar problemas derivados de la monitorización masiva de servidores, tratados de logs y presentación de los datos obtenidos. Elasticsearch es un motor de búsqueda y analítica. Logstash es un pipeline de procesamiento de datos del lado del servidor que ingesta datos de una multitud de fuentes simultáneamente, los transforma y luego los maneja y Kibana permite a los usuarios visualizar los datos en cuadros y gráficos con Elasticsearch

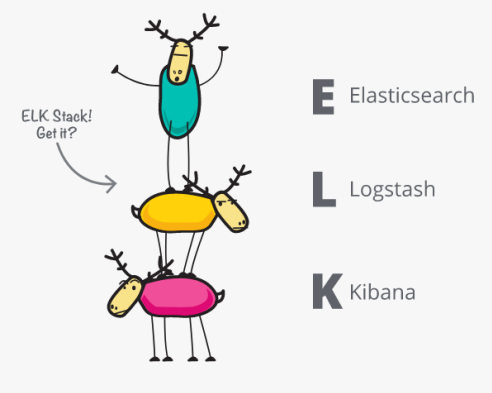

Ilustración 6 - ELK Stack
https://www.elastic.co/

[9] https://www.elastic.co/es/what-is/elk-stack

Qué componentes forman ELK

Elasticsearch

Es una base de datos distribuida. Distribuye toda la información en todos los nodos, tiene una alta disponibilidad y es maneja gran cantidad de datos. Al igual que distribuye la información, distribuye el procesamiento. Cuando se envía una consulta a las base de datos, al estar distribuida, será cada nodo el que procese dicha información y devuelva los resultados. Por tanto, podemos obtener mejores rendimientos.

Logstach

Es la parte de preprocesamiento antes de guardar la información en la base de datos en Elasticsearch que hemos comentado, donde se recogen las entradas, se gestionan los eventos y se envia una salida, antes de almacenarlos en las bases de datos.

Kibana

Es el más visual, dónde vamos a generar las visualizaciones sobre la información y dónde vamos a generar los dashboards

A modo de resumen podemos ver en esta imagen, la arquitectura y la ingesta de datos en ELK:

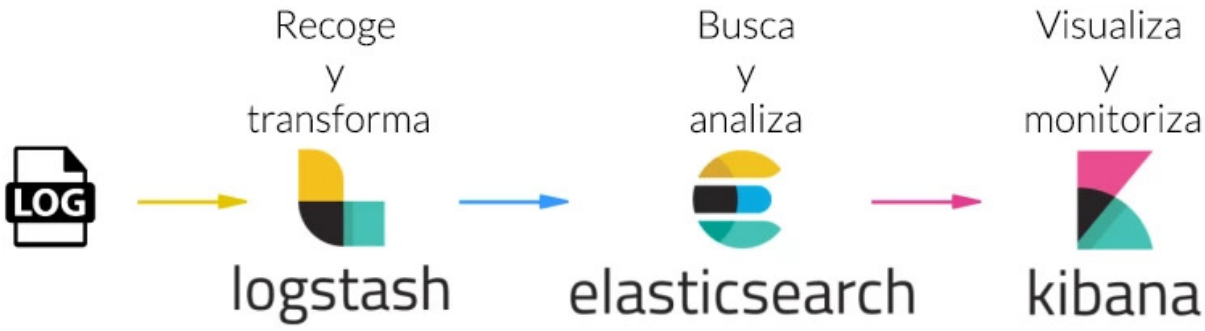

Ilustración 7 - Procesamiento ELK
https://www.elastic.co/

2.7. **Requisitos legales y éticos**

En España con la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (LOPD[10]) que es una actualización de el Reglamento General de Protección de Datos (RGPD) europeo. Para poder cumplir la legalidad de protección de datos hay que desarrollar nuevas técnicas y tener en cuenta todas las circunstancias. Ahora se considera como información personal cualquier dato que le pueda identificar como individuo, directa o indirectamente.

[10] https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16673

De este modo, los nombres de usuarios, las direcciones IP incluso si son dinámicas, son datos personales si pueden llegar a utilizarse para identificar al usuario, y este proyecto puede ocurrir si usamos direcciones fijas.
El sistema almacenará de forma temporal datos personales de los usuarios para comprobar si las comunicación es correcta, para poder cumplir con la legalidad[11], tenemos que hacer los siguiente:

- Derecho de acceso. Informar a los usuarios que sus datos pueden ser almacenados al entrar en nuestra red y se guardará en nuestra Base de datos durante un tiempo.
- Derecho de supresión. Este derecho facilita al interesado la posibilidad de requerir que sus datos personales “desaparezcan” o “no dejen rastro en la red”.
- Derecho a la limitación del tratamiento de tus datos. Solicitar al responsable la conservación tus datos cuando el tratamiento sea ilícito y te has opuesto a la supresión de tus datos y en su lugar solicitas la limitación de su uso
- Derecho a la portabilidad de tus datos. Poder crear formulas por si un usuario nos pide ver la información que tenemos almacenada.
- Derecho a la limitación del tratamiento de tus datos. Los datos que no sean estrictamente necesarios para la prestación del servicio deberán ser eliminados.

Para poder implantar la LOPD en nuestro proyecto habrá que realizar mejoras en el mismo, de modo que podamos asegurar a los usuarios un mínimo de legalidad. El sistema debe informar de que con la sonda almacenamos datos de forma temporal y que nuestra red se va a encargar de interceptar las comunicaciones de forma temporal. No se el modo todavía, pero se creará un formulario o un mensaje para que acepte las condiciones al entrar en nuestra red.
Una mejora al proyecto puede ser buscar una aplicación que almacenan los datos de forma anónima dentro de la base de datos. Datos como nombres de usuario, IP, MAC, … se almacenarían de alguna forma que se supiera nunca de quien son.

[11] https://www.aepd.es/sites/default/files/2019-10/guia-ciudadano.pdf

3. Fase de Implementación

En este primer punto se va a describir la realización desde cero de una punto de acceso con una Raspberry-pi, desde la instalación del sistema operativo a la instalación de la sonda IDS así como el servicio FTP, DHCP y DNS
Se podría haber pensado usar otra opción con un Switch, pero el problema de no poder salir de casa por el coronavirus hace que el material sea menos del que en principio contábamos, así que el proyecto se va a ceñir al uso de una Raspberry pi.

3.1. Instalación del Sistema Operativo.

Se ha decidido usar la versión más usada que dentro de los sistemas operativos usados, Raspbian, en su última versión con todas las actualizaciones.
Actualmente para el proyecto se tiene una Raspberry pi 3 con una sd de 8 gb de tipo 10, lo que hace por lo menos el acceso a los dato sea rápido.

3.1.1. Descargar imagen de Rasbian

Se va a instalar la versión Buster que tiene nuevas novedades pensadas para Raspberry Pi 4, pero vamos usarla en la que nosotros tenemos actualmente. Descargamos el software:

https://www.raspberrypi.org/downloads/raspbian/

Para no sobrecargar la Raspberry pi vamos usar la versión que tiene el entorno de ventanas. Hay que considerar que el sistema va actuar de forma autónoma y funcionará con solo tener conectado el cable de corriente y el cable de red.
Por ello se ha elegido la versión Raspbian Buster Lite que viene con el software mínimo para funcionar y con esta base después instalar todo lo necesario.

3.1.2. Uso de Herramienta para Grabar SD con Sistema Operativo

Se han buscado varias herramientas para instalar el Sistema operative dentro de la SD.
Algunas de ellas son:

- Rufus (https://rufus.ie/)
- Win32 Disk Imager (https://sourceforge.net/projects/win32diskimager/)
- Etcher (https://www.balena.io/etcher/)

Al final se ha optado por Rufus, por ser la que más gente usada y la mejor valorada del mercado.[12]

[12] https://www.slant.co/versus/21904/21912/~rufus_vs_etcher

Con la aplicación Rufus insertamos Raspbian Buster dentro de nuestra memoria SD , hay que elegir el dispositivo, la versión de raspbian que nos hemos descargado en el paso anterior, pulsar “start” y esperar unos minutos a que se instale correctamente.

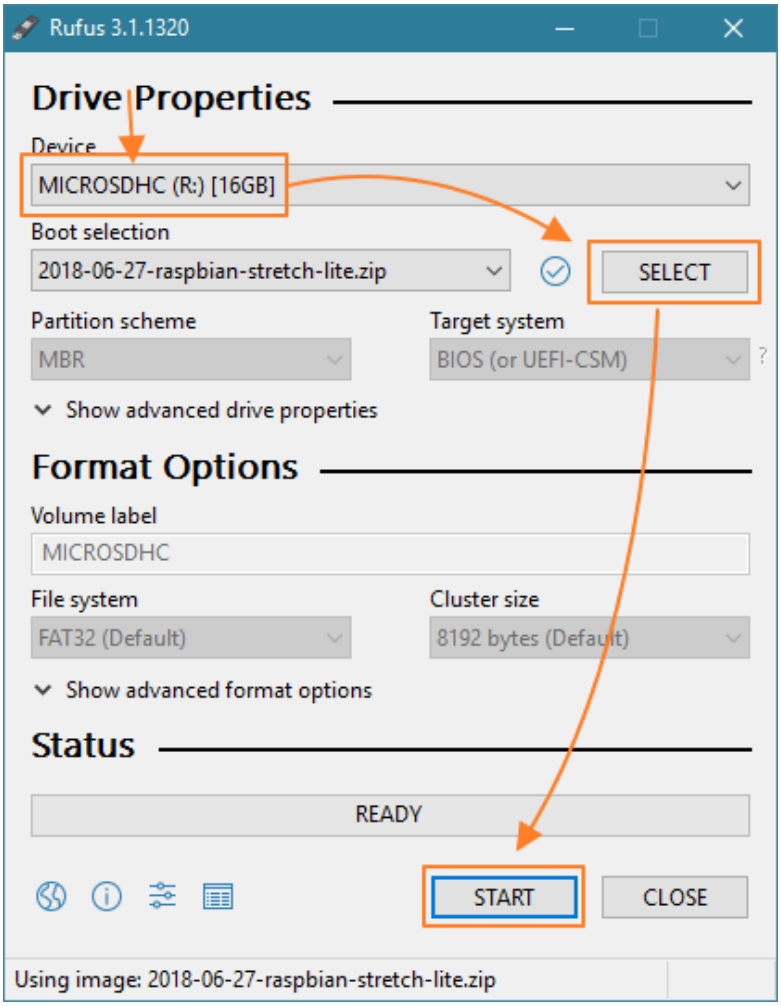

Ilustración 8 - Rufus
https://community.ui.com/

3.1.3. Primer Contacto con Raspberry Pi

Para iniciar la Raspberry-pi, es necesario conectarla a la corriente con un enchufe y no hay que pulsar nada, no tiene ningún botón de arranque.
En el primer arranque de la Raspberry-pi, se tiene que acceder con un teclado conectado al puerto USB y. Conectar por HDMI una pantalla. Para acceder al sistema operativo se utiliza el usuario y clave por defecto:

Usuario:pi

Password:raspberry

La pantalla que va a aparecer es similar a esta:

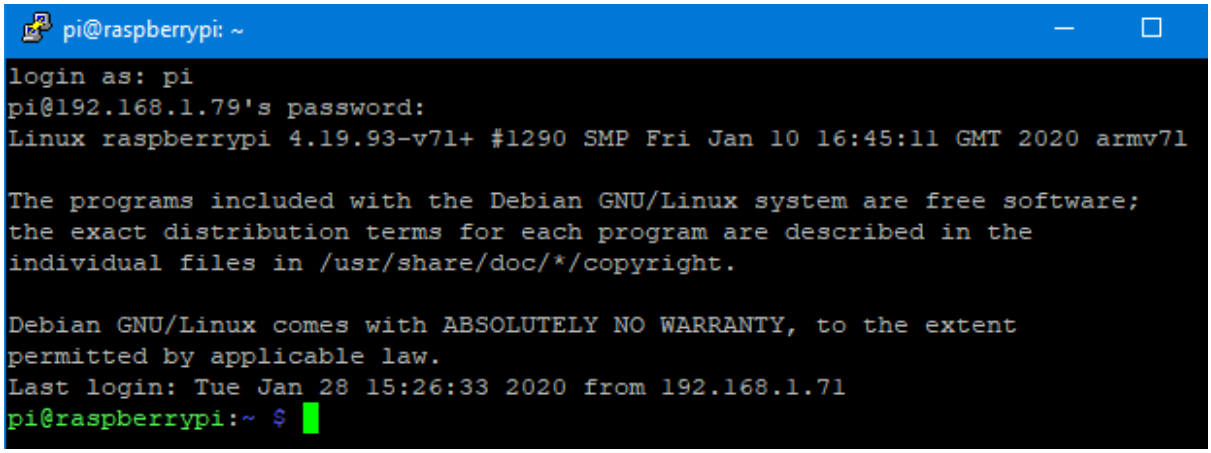

Ilustración 9- Inicio de sesión
Elaboración propia

3.1.4. Actualizar el Sistema a la última versión

Lo siguiente que se va a realizar es actualizar el sistema con las siguientes instrucciones:

```
sudo apt-get update
sudo apt-get upgrade
```

Esencialmente usar ***"apt-get update",*** lo que en realidad estamos haciendo es actualizar los repositorios y "***apt-get upgrade",*** lo que hacemos es una actualización de nuestro sistema con todas las posibles actualizaciones que pudiera haber.

3.2. Instalación del Servicio Secure Shell (SSH)

El protocolo SSH es un método para el inicio de sesión remoto seguro desde un equipo a otro. Proporciona varias opciones para una autenticación sólida, y protege la seguridad e integridad de las comunicaciones con un cifrado seguro. Es una alternativa segura a los protocolos de inicio de sesión no protegidos (como telnet, rlogin) y métodos de transferencia de archivos inseguros (como FTP).
El protocolo se usa en redes corporativas para:

- Brindar acceso seguro a los usuarios y procesos automatizados
- Transferencias de archivos interactivas y automatizadas
- Emisión de comandos remotos
- Administración de la infraestructura de red.

Entre todas las opciones que ofrece la Raspberry Pi, su uso como servidor en una red domestica es muy popular. En caso de que se use para este fin, sería de gran utilidad el poder ejecutar comandos en la Raspberry Pi sin necesidad de conectar una pantalla, un teclado, un ratón y sin tener que moverse hasta la localización física de la Raspberry.

Se busca entonces poder loguearse en la Raspberry vía SSH desde otros ordenadores o incluso desde dispositivos móviles como smartphones o tablets.
SSH se puede activar desde la CLI, introduciendo los siguientes comandos:

```
pi@raspberry:~ $ sudo systemctl enable ssh
Synchronizing state of ssh.service with SysV service script with /lib/systemd/sy
stemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ssh
pi@raspberry:~ $ sudo systemctl start ssh
```

Ilustración 10 - Activar ssh
Elaboración propia

Probamos a conectarnos con un herramienta tupo putty desde un sistema distinto como es Windows:
Añadimos la Raspberry Pi como un host:

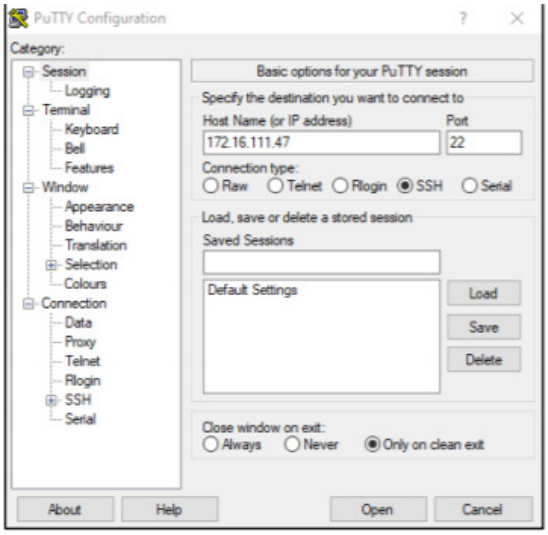

Ilustración 11 - Acceso desde Putty
Elaboración propia

Después debemos meter la contraseña de forma correcta y nos mostrará algo parecido a esto:

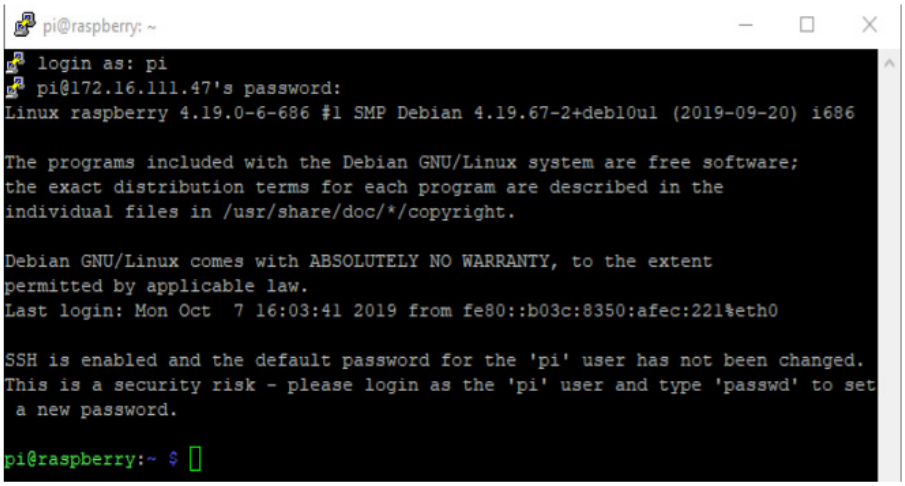

Ilustración 12- Acceso desde putty correcto
Elaboración propia

3.3. **Instalación del Servicio File Transfer Protocol (FTP)**

El protocolo de transferencia de archivos (FTP) es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP, basado en la arquitectura cliente-servidor.
Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde el o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.
El servicio FTP es ofrecido por la capa de aplicación del modelo de capas de red TCP/IP al usuario, utilizando normalmente el puerto de red 20 y 21. Un problema básico de FTP es que está pensado para ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, ya que todo el intercambio de información, desde el login y password del usuario en el servidor hasta la transferencia de cualquier archivo, se realiza en texto plano sin ningún tipo de cifrado, con lo que un posible atacante puede capturar este tráfico, acceder al servidor y/o apropiarse de los archivos transferidos.

Para instalar el servicio, se usan los siguientes comandos para actualizar el repositorio, e instalar "vsftpd", respectivamente:

```
sudo apt-get update
sudo apt-get install vsftpd
```

Se tiene que hacer una configuración básica, para actualizar el archivo de configuración , una vez instalado, editar el archivo de configuración principal.:

```
sudo nano /etc/vsftpd.conf
```

Se modifican las siguientes líneas:

```
  GNU nano 3.2                  /etc/vsftpd.conf                  Modific

# files.
listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
local_umask=022
```

Ilustración 13 - Configuración ftp
Elaboración propia

```
# You may specify an explicit list of local users to chroot() to their home
# directory. If chroot_local_user is YES, then this list becomes a list of
# users to NOT chroot().
# (Warning! chroot'ing can be very dangerous. If using chroot, make sure that
# the user does not have write access to the top level directory within the
# chroot)
chroot_local_user=YES
```

Ilustración 14 - Cont. ftp
Elaboración propia

Y se agregan las siguientes al final:

```
user_sub_token=$USER
local_root=/home/$USER/ftp
```

Ilustración 15 - Agregamos al ftp
Elaboración propia

3.4. **Configuración de la red para crear el punto de acceso en dispositivo IOT**

El primer paso será la instalación del software hostapd. Hostapd es un software de daemon de espacio de usuario que permite que una tarjeta de interfaz de red actúe como un punto de acceso y servidor de autenticación. Se puede realizar desde los mismos repositorios del sistema como administrador. Además, instalaremos el paquete isc-dhcp-server, que nos permitirá levantar un servicio DHCP. De este modo la asignación de direcciones IP a los dispositivos conectados, como hemos comentado, será independiente.

El comando será el siguiente:

sudo apt install hostapd isc-dhcp-server

```
pi@raspberrypi:~ $ sudo apt install hostapd isc-dhcp-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libnl-route-3-200
Suggested packages:
  isc-dhcp-server-ldap
The following NEW packages will be installed:
  hostapd isc-dhcp-server libnl-route-3-200
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/909 kB of archives.
After this operation, 2,267 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
WARNING: The following packages cannot be authenticated!
  libnl-route-3-200 hostapd isc-dhcp-server
Install these packages without verification? [y/N] y
```

Ilustración 16- Instalación hostapd

Elaboración propia

Necesitamos establecer una *IP* fija, que no varíe, para la interfaz inalámbrica del punto de acceso. De esta manera, los clientes podrán saber dónde se encuentra siempre la puerta de enlace predeterminada.
Lo primero es desactivar temporalmente la interfaz inalámbrica, en nuestro caso wlan*0* con el siguiente comando:

```
sudo ifconfig wlan0 down
```

Tras ello, editaremos el fichero */etc/network/interfaces* y agregaremos, o editaremos de existir ya un apartado para *wlna1,* las siguientes líneas:

```
allow-hotplug wlan0
auto wlan0
iface wlan0 inet static
      address 11.0.0.1
      netmask 255.255.255.0
```

Asimismo, configuramos de forma manual aquí que la interfaz de red cableada eth0 obtendrá mediante DHCP sus direcciones. De otra manera, al tener contenido este fichero /etc/network/interfaces, todas las interfaces leerán de aquí sus datos, y eth0 quedará sin configurar., de la siguiente manera:

```
allow-hotplug wlan0
auto eth0
iface eth0 inet dhcp
```

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

allow-hotplug wlan0
auto wlan0
iface wlan0 inet static
        address 11.0.0.1
        netmask 255.255.255.0

allow-hotplug eth0
auto eth0
iface eth0 inet dhcp
```

Ilustración 17 - Configuración wlan
Elaboración propia

Podemos levantar de nuevo la interfaz inalámbrica con:

```
sudo ifconfig wlan0 up
```

Lo siguiente es configurar el servicio *DHCP* en nuestra sonda será editar el fichero */etc/dhcp/dhcpd.conf*. Este fichero contiene la configuración del *daemon* que maneja el servicio *DHCP*.

Estos son los cambios para realizar dentro del fichero:

```
# dhcpd.conf
#
# Sample configuration file for ISC dhcpd
#

# option definitions common to all supported networks...
#option domain-name "example.org";
#option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

# The ddns-updates-style parameter controls whether or not the server will
# attempt to do a DNS update when a lease is confirmed. We default to the
# behavior of the version 2 packages ('none', since DHCP v2 didn't
# have support for DDNS.)
ddns-update-style none;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
authoritative;
```

Ilustración 18 - Configuración DHCP
Elaboración propia

Debemos comentar las líneas correspondientes a *domain-name* y *option domain-name-servers*. Quitamos el comentario de la línea correspondiente a *authoritative*. De esta forma, declaramos que somos el servidor *DHCP* principal de este segmento de red.

Agregamos por último las siguientes líneas, de manera que establecemos la propia configuración del servicio:

```
subnet 11.0.0.0 netmask 255.255.255.0 {
        range 11.0.0.10 11.0.0.20;
        option broadcast-address 11.0.0.255;
        option routers 11.0.0.1;
        default-lease-time 600;
        max-lease-time 7200;
        option domain-name "local";
        option domain-name-servers 8.8.8.8;
}
```

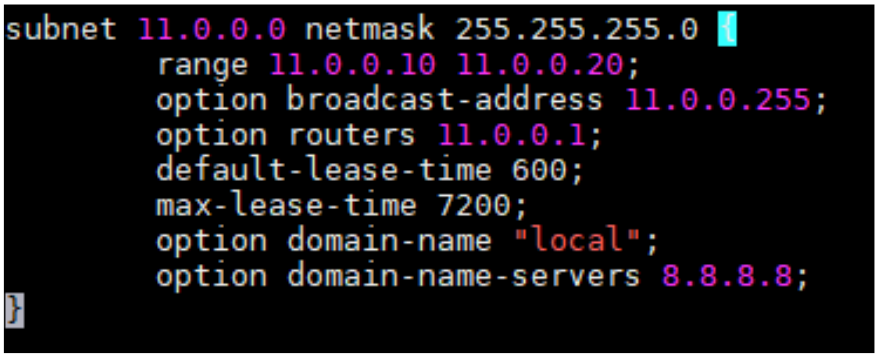

Ilustración 19 - Cont. Configuración DHCP
Elaboración propia

De esta forma, estamos configurando estos elementos:

- Rango *IP* (*range*): El rango de direcciones *IP* en el que realizará el servidor las asignaciones. En nuestro caso repartiremos direcciones *IP* en el rango *0.0.10-11.0.0.20*, un total de 11 direcciones.
- Dirección de *broadcast* (*option broadcast-address*): La dirección de *broadcast* de la red, en nuestro caso 11.0.0.255, siguiendo la nomenclatura usual.
- Puerta de enlace predeterminada *(option routers*): La puerta de enlace de la red, es decir, hacia dónde se dirigirá el tráfico con objetivo el exterior.
- Tiempo estándar de asignación de direcciones (*default-lease-time*): Tiempo que tiene reservada el servidor una dirección *IP* para un cliente. Se usa si no lo especifica el cliente. Lo establecemos a 600 segundos.
- Tiempo máximo de asignación de direcciones (*max-lease-time*): Similar al anterior, pero establece el tiempo máximo de reserva, independientemente de la petición del cliente. Lo establecemos en 7200 segundos.
- Nombre del dominio (*option domain-name*): Nombre de dominio del servidor *DHCP*, simplemente el nombre del *host*. Simplemente introducimos *local*.

Servidores DNS (option domain-name-servers): Servidor DNS al que redirigir las peticiones de resolución de nombres. Queremos no depender de la puerta de enlace que utilicemos como salida a Internet, así que introducimos el servidor DNS público de Google (8.8.8.8).
Por último, incluimos la interfaz wlan0 dentro de la línea INTERFACESv4 del fichero localizado en */etc/default/isc-dhcp-server*.

```
# Defaults for isc-dhcp-server (sourced by /etc/init.d/isc-dhcp-server)

# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
#DHCPDv4_CONF=/etc/dhcp/dhcpd.conf
#DHCPDv6_CONF=/etc/dhcp/dhcpd6.conf

# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPDv4_PID=/var/run/dhcpd.pid
#DHCPDv6_PID=/var/run/dhcpd6.pid

# Additional options to start dhcpd with.
#       Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
#       Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACESv4="wlan0"
INTERFACESv6=""
~
```

Ilustración 20-Uso Interface
Elaboración propia

3.5. **Configuración del punto de acceso**

Procedemos ahora a la creación del punto de acceso en sí. El primer paso es crear un fichero en la dirección */etc/hostapd/hostapd.conf*. Debemos rellenar unas pocas líneas de configuración. Dependiendo de nuestro hardware y la configuración concreta de seguridad que queramos configurar, podría quedar de la siguiente manera:

```
# Basic configuration
interface=wlan0
ssid=probe
channel=1
ctrl_interface=/var/run/hostapd

# WPA2 configuration
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
#wpa_passphrase=probe111
wpa_psk_file=/etc/hostapd/hostapd-psk
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
rsn_pairwise=CCMP

# Hardware configuration
ieee80211n=1
hw_mode=g
driver=nl80211
~
```

Ilustración 21 - Configuración hostapd.conf
Elaboración propia

Vemos que contiene tanto configuración básica de la red inalámbrica como una configuración de seguridad mediante WPA2-CCMP y configuración hardware.

Debemos rellenar ahora el fichero /etc/hostapd/hostapd-psk con la clave de cifrado de la red. La clave de cifrado, que estableceremos como probe111, a modo de ejemplo, deberá introducirse en dicho fichero o bien en formato ASCII o bien en formato de clave hexadecimal de 256 bits.
Esta clave hexadecimal, que utilizaremos por cuestiones de seguridad, se halla combinando la clave en formato ASCII con el SSID de la red y pasando esta combinación por el algoritmo de hash SHA1 un número de 4096 bits y produciendo la salida de 256 bits.
La generación de esta clave la realizaremos mediante la herramienta wpa_passphrase, incluida en Raspbian de forma predeterminada con WPA Supplicant, que contiene diversas utilidades de gestión de redes WPA. La contraseña, por poner un ejemplo, será probe111:

```
sudo wpa_passphrase "probe7" "probe111"
```

```
pi@raspberrypi:~ $ sudo wpa_passphrase "prueba7" "probe111"
network={
        ssid="prueba7"
        #psk="probe111"
        psk=16537326e3933c98bafbdeaeda6993b1bf7de59bacd00e582431aa56b2bd1ed4
}
```

Ilustración 22- creación de clave
Elaboración propia

Dicha clave la copiamos en el fichero */etc/hostapd/hostapd-psk* de la siguiente manera, estableciéndola como clave única de la red:

```
00:00:00:00:00:00 be8cb1bc2d55e1ba204fdc2ebf7015c8ac55490adf34e7d6d689e6f2149c5b2b
~
```

Ilustración 23- Clave creada
Elaboración propia

El primer valor quiere decir que será la clave para cualquier MAC de cliente.
Habiendo ya completado esta configuración, nos dirigiremos al fichero /*etc/default/hostpapd*. Dentro de éste, en la línea correspondiente a DAEMON_CONF, introduciremos nuestro fichero de configuración localizado en /etc/hostapd/hostapd.conf:

```
# Defaults for hostapd initscript
#
# See /usr/share/doc/hostapd/README.Debian for information about alternative
# methods of managing hostapd.
#
# Uncomment and set DAEMON_CONF to the absolute path of a hostapd configuration
# file and hostapd will be started during system boot. An example configuration
# file can be found at /usr/share/doc/hostapd/examples/hostapd.conf.gz
#
DAEMON_CONF="/etc/hostapd/hostapd.conf"

# Additional daemon options to be appended to hostapd command:-
#       -d   show more debug messages (-dd for even more)
#       -K   include key data in debug messages
#       -t   include timestamps in some debug messages
#
# Note that -B (daemon mode) and -P (pidfile) options are automatically
# configured by the init.d script and must not be added to DAEMON_OPTS.
#
#DAEMON_OPTS=""
~
```

Ilustración 24-Deamon_conf

Para realizar la redirección del tráfico entre nuestra red inalámbrica e Internet, debemos configurar un servicio NAT y activar la redirección IPv4 entre las dos interfaces de red.

El primer paso es agregar o editar la siguiente línea en el fichero */etc/sysctl.conf*, de manera que activamos al inicio la redirección en IPv4:

```
net.ipv4.ip_forward=1
```

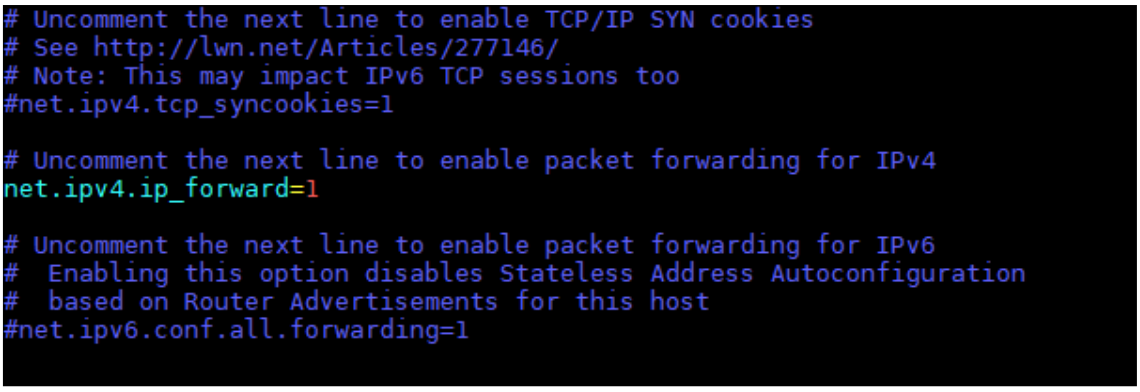

Ilustración 25- /etc/sysctl.conf
Elaboración propia

Para activarla en el mismo momento sin necesitar un reinicio, podemos ejecutar el siguiente comando como root:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Ahora debemos configurar la NAT entre la interfaz que nos ofrece el acceso a Internet (tomaremos eth0 como referencia) y la interfaz del punto de acceso inalámbrico (wlan0 en nuestro caso). Esto podemos hacerlo mediante los siguientes comandos, que configurarán diversas líneas de iptables:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan1 -m state --state
RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan1 -o eth0 -j ACCEPT
```

Estas líneas realizan las siguientes acciones:

1. Activar una NAT tipo MASQUERADE cuyo destino es la interfaz eth0.
2. Realiza un reenvío desde la interfaz eth0 hacia wlan1 si el estado de la conexión en cuestión es RELATED o ESTABLISHED, es decir, si es referente a una conexión ya establecida. Así permitimos solo las conexiones iniciadas desde dentro (red local) hacia fuera (Internet).
3. Realiza un reenvío desde la interfaz wlan1 hacia eth0 sea cual sea, permitiendo cualquier conexión hacia el exterior.

Podemos hacer persistente esta configuración guardando estos comandos iptables en un fichero de script localizado en /etc/firewall-probe.sh con permisos de ejecución, utilizando chmod.

```
#!/bin/bash
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

Ilustración 26 - Fichero firewall-probe.sh
Elaboración propia

Para que las reglas de *iptables* se añadan al inicio, añadiremos la siguiente línea al fichero */etc/rc.local*, que como indica su contenido, es un script que se ejecuta al final de cada inicio del sistema en modo multiusuario (el modo usual).

```
bash /etc/firewall-probe.sh
```

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

# helo i am haker
bash /etc/firewall-probe.sh

exit 0
```

Ilustración 27- Fichero rc.local
Elaboración propia

El último paso de configuración para nuestro equipo de recogida de tráfico es establecer que los servicios asociados a hostapd y isc-dhcp-server arranquen con el inicio del sistema. Podemos realizar esto mediante los siguientes comandos:

```
sudo systemctl unmask hostapd
sudo systemctl enable hostapd
sudo systemctl enable isc-dhcp-server
```

A partir del siguiente reinicio, tendremos nuestro punto de acceso **totalmente configurado para funcionar.**

Ilustración 28-Prueba Sonda
Elaboración propia

3.5.1. **Prueba de concepto**

Además, de cara a futuro, instalaremos también la herramienta de *sniffing* ***tcpdump***. Tcpdump es una herramienta para línea de comandos cuya utilidad principal es analizar el tráfico que circula por la red. Instalamos la herramienta:

```
pi@raspberrypi:~ $ sudo apt install tcpdump
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  tcpdump
0 upgraded, 1 newly installed, 0 to remove and 44 not upgraded.
Need to get 354 kB of archives.
After this operation, 922 kB of additional disk space will be used.
Get:1 http://mirrors.linux-bg.org/raspbian/raspbian stretch/main armhf tcpdump armhf 4.9.2-1~deb9u1 [354 kB]
Fetched 354 kB in 2s (147 kB/s)
Selecting previously unselected package tcpdump.
(Reading database ... 36665 files and directories currently installed.)
Preparing to unpack .../tcpdump_4.9.2-1~deb9u1_armhf.deb ...
Unpacking tcpdump (4.9.2-1~deb9u1) ...
Setting up tcpdump (4.9.2-1~deb9u1) ...
Processing triggers for man-db (2.7.6.1-2) ...
```

Ilustración 29- Instalación tcpdump
Elaboración propia

Podemos desde este punto empezar a escuchar el tráfico en la interfaz de red *wlan0,* en la que están conectados todos los puntos de acceso. Para ello ejecutamos como administrador la herramienta *tcpdump* que hemos instalado .

```
tcpdump -i wlan0
```

Para guardar todo el tráfico que pasa por la interfaz de red de nuestra sonda, necesitaremos realizar un volcado de dicho tráfico de manera constante a un fichero. Para ello vamos a utilizar un script que comience a escuchar el tráfico tras el levantamiento de punto de acceso.

Junto a la herramienta hostapd se encuentra la utilidad hostapd_cli, que permite al invocarse la ejecución de un script cada vez que un cliente se conecta y desconecta. Evidentemente, debe invocase con el daemon de hostapd ya en funcionamiento. Utilizaremos hostapd_cli para establecer que cada vez que un cliente se conecte a nuestro punto de acceso se comience a guardar en un fichero el tráfico correspondiente. Lo primero será asegurarse de que el fichero /etc/hostapd/hostapd.conf contenga la opción de configuración ctfl_interface a la dirección /var/run/hostapd. En ella se encuentra de forma predeterminada la interfaz de control utilizada por hostapd_cli para comunicarse con hostapd.

```
# Basic configuration
interface=wlan0
ssid=probe
channel=1
ctrl_interface=/var/run/hostapd

# WPA2 configuration
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
#wpa_passphrase=probe111
wpa_psk_file=/etc/hostapd/hostapd-psk
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
rsn_pairwise=CCMP

# Hardware configuration
ieee80211n=1
hw_mode=g
driver=nl80211
```

Ilustración 30- /etc/hostapd/hostapd.conf
Elaboración propia

Ahora, crearemos un fichero */etc/hostapd/sniffer.sh*, que contendrá el *script* a ejecutarse cada vez que un cliente se conecte o desconecte del punto de acceso. Dicho *script*, al ser llamado mediante *hostapd_cli*, recibirá como argumentos la interfaz en la que se efectúa la conexión o desconexión, el tipo de evento (conexión o desconexión) y la dirección *MAC* de la estación cliente. Deberá tener permisos de ejecución, haciendo uso de *chmod.*

```
#!/bin/bash

mkdir -p /home/pi/capturas

if [[ $2 == "AP-STA-CONNECTED" ]]
then
	echo "Cliente $3 conectado en $1" >> /tmp/sniffer_log
	tcpdump -U -i $1 ether host $3 -s 0 -w
"/home/pi/capturas/$3__`date +%d-%m-%Y__%T.pcap`" &
	pid=$!
	(flock -e 200
		echo "$pid $3" >> /tmp/sniffer_pids
	) 200>/tmp/lock
```

```
fi

if [[ $2 == "AP-STA-DISCONNECTED" ]]
then
       echo "Cliente $3 desconectado en $1" >> /tmp/sniffer_log
       (flock -e 200
              line=`cat /tmp/sniffer_pids | grep $3`
              pid=`echo $line | cut -f 1 -d ' '`
              kill $pid
       ) 200>/tmp/lock
       (flock -e 200
              cat /tmp/sniffer_pids | egrep -v '^$3 .*$' >
/tmp/sniffer_pids
       ) 200>/tmp/lock
fi
```

Tabla 7 - Script sniffer.sh

```
pi@raspberrypi:/etc/hostapd $ sudo vim sniffer.sh
pi@raspberrypi:/etc/hostapd $ sudo chmod +x sniffer.sh
pi@raspberrypi:/etc/hostapd $ ls
hostapd.conf  hostapd-psk  ifupdown.sh  sniffer.sh
```

Ilustración 31- Permisos a sniffer.sh
Elaboración propia

El *script*, al llamarse, comprobará si el evento es de conexión o desconexión. Si es de conexión, comenzará una captura mediante la herramienta *tcpdump* por la interfaz correspondiente y sobre la dirección *MAC* del cliente, guardando la captura en un fichero cuyo nombre será dicha dirección *MAC* seguida de la fecha y hora del comienzo de la captura. Este detalle de agregar la fecha al nombre previene que *tcpdump* sobrescriba el fichero al realizar un dispositivo varias conexiones.

Además, guardará el *PID* del proceso de *tcpdump* en un fichero temporal, en una línea que contiene el *PID* y la *MAC* del cliente. En el evento de desconexión, se enviará una señal de finalización al proceso leyendo su *PID* de dicho fichero temporal, y eliminando la línea correspondiente.

Las condiciones de carrera al haber varias instancias de *sniffer.sh* se previenen utilizando la herramienta *flock*. Esta establece una política de bloqueo mediante un cerrojo en */tmp/lock*, de manera que solamente una instancia del *script* podrá acceder de forma simultánea al fichero */tmp/sniffer_pids* que controla los procesos de captura.

Asimismo, el script guardará en un fichero de log */tmp/sniffer_log* las conexiones y desconexiones que se vayan produciendo.

Seguidamente, configuraremos la herramienta *hostapd_cli* como un servicio del sistema que se ejecutará al inicio una vez esté activo el servicio de *hostapd*. Crearemos para esto el fichero */etc/init.d/hostpapd_cli*, con permisos de ejecución y con el siguiente contenido:

```
#!/bin/sh

### BEGIN INIT INFO
# Provides: hostapd_cli
# Required-Start: $remote_fs hostapd
# Required-Stop: $remote_fs
```

```
# Should-Start:
# Should-Stop:
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Enables the sniffer on probe access point
# Description:
### END INIT INFO

PATH=/sbin:/bin:/usr/sbin:/usr/bin
DAEMON_SBIN=/usr/sbin/hostapd_cli
NAME=hostapd_cli
DESC="hostapd cli"
PIDFILE=/var/run/hostapd_cli.pid

[ -x "$DAEMON_SBIN" ] || exit 0
DAEMON_OPTS="-B -a /etc/hostapd/sniffer.sh"
. /lib/lsb/init-functions

case "$1" in
  start)
    log_daemon_msg "Starting $DESC" "$NAME"
    start-stop-daemon --start --oknodo --quiet --exec "$DAEMON_SBIN"
¥
      -- $DAEMON_OPTS >/dev/null
    log_end_msg "$?"
    ;;
  stop)
    log_daemon_msg "Stopping $DESC" "$NAME"
    start-stop-daemon --stop --oknodo --quiet --exec "$DAEMON_SBIN"
    log_end_msg "$?"
    ;;
  restart|force-reload)
    $0 stop
    sleep 4
    $0 start
    ;;
  status)
    status_of_proc "$DAEMON_SBIN" "$NAME"
    exit $?
    ;;

  *)
    N=/etc/init.d/$NAME
    echo "Usage: $N {start|stop|restart|force-reload|status}" >&2
    exit 1
    ;;
esac
exit 0
```

Tabla 8- fichero /etc/init.d/hostpapd_cli

Será necesario dar permisos de ejecución al servicio y habilitarlo al arranque:

sudo chmod +x /etc/init.d/hostapd_cli

sudo systemctl enable hostapd_cli

Con esto tendremos el servicio configurado y listo para ser ejecutado tras un reinicio del Sistema. Ahora, cada vez que un dispositivo se conecte a nuestro punto de acceso, comenzará una escucha en exclusiva para su dirección MAC y se guardará en nuestra sonda.

Damos los permisos al fichero y lo añadimos como servicio para que leyendo los datos de la sonda:

```
pi@raspberrypi:/etc/init.d $ sudo chmod +x hostapd_cli
pi@raspberrypi:/etc/init.d $ sudo systemctl enable hostapd_cli
hostapd_cli.service is not a native service, redirecting to systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable hostapd_cli
```

Ilustración 32- Permisos y añadimos servicio
Elaboración propia

Como podemos ver ya tenemos un historial con la MAC y la hora de conexión del dispositivo, dentro de estos archivos tenemos la conexión establecidas por este:

```
pi@raspberrypi:~ $ ls
capturas  Documents  MagPi  Pictures  Templates
Desktop   Downloads  Music  Public    Videos
pi@raspberrypi:~ $ ls capturas/
48:4b:aa:91:86:bb__04-04-2020__20:20:41.pcap
pi@raspberrypi:~ $
```

```
pi@raspberrypi:~/capturas $ tcpdump -r 48\:4b\:aa\:91\:86\:bb__04-04-2020__20\:20\:41.pcap
reading from file 48:4b:aa:91:86:bb__04-04-2020__20:20:41.pcap, link-type EN10MB (Ethernet)
20:20:41.847310 ARP, Request who-has 11.0.0.1 tell 11.0.0.11, length 28
20:20:41.847372 ARP, Reply 11.0.0.1 is-at b8:27:eb:60:d2:fb (oui Unknown), length 28
20:20:41.849647 ARP, Request who-has 11.0.0.11 tell 11.0.0.11, length 28
```

Ilustración 33 - Conexiones establecidas
Elaboración propia

3.6. **Intento de desarrollo de SIEM en Dispositivo IOT**

Considerando las posibilidades de procesamiento y almacenamiento de Raspberry pi 3, la instalación de SIEM en la Raspberry pi 3 en sí misma debe evaluarse para determinar si es factible o factible. Fácil de instalar en otros dispositivos con más potencia de procesamiento y más posibilidades.
En la fase de implementación, se realiza una prueba de concepto (PoC) en la instalación y el rendimiento del sistema SIEM en la propia Raspberry pi 3, de modo que se pueda determinar si es necesario implementar SIEM en otro dispositivo.

Al intentar instalar los servicios, nos damos cuenta de que la capacidad de procesamiento y de almacenamiento no es la adecuada.
La Pila ELK necesita de bastantes recursos hardware, así que hay que sacarlo del esquema inicial de intentar hacer todo en el mismo dispositivo.

Se tienen problemas con ElasticSearch:

```
pi@raspberrypi:/usr/lib/jvm $ sudo systemctl status elasticsearch
● elasticsearch.service - LSB: Starts elasticsearch
   Loaded: loaded (/etc/init.d/elasticsearch; generated)
   Active: active (exited) since Thu 2020-04-02 15:47:04 CEST; 2s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 11248 ExecStart=/etc/init.d/elasticsearch start (code=exited, status=0/SUCCESS)

Apr 02 15:47:03 raspberrypi systemd[1]: Starting LSB: Starts elasticsearch...
Apr 02 15:47:04 raspberrypi elasticsearch[11248]: [warning] /etc/init.d/elasticsearch: JVM flavor 'sun6' not understood
Apr 02 15:47:04 raspberrypi elasticsearch[11248]: Starting Elasticsearch Server:.
Apr 02 15:47:04 raspberrypi systemd[1]: Started LSB: Starts elasticsearch.
```

Ilustración 34 - Error ElasticSearch
Elaboración propia

También suceden errores con Kibana:

```
pi@raspberrypi:/var/log/elasticsearch $ sudo systemctl status kibana
● kibana.service - Kibana
   Loaded: loaded (/etc/systemd/system/kibana.service; disabled; vendor preset: enabled)
   Active: failed (Result: exit-code) since Thu 2020-04-02 15:57:34 CEST; 8s ago
  Process: 11492 ExecStart=/usr/share/kibana/bin/kibana -c /etc/kibana/kibana.yml (code=exited, status=2)
 Main PID: 11492 (code=exited, status=2)

Apr 02 15:57:34 raspberrypi systemd[1]: kibana.service: Service RestartSec=100ms expired, scheduling restart.
Apr 02 15:57:34 raspberrypi systemd[1]: kibana.service: Scheduled restart job, restart counter is at 5.
Apr 02 15:57:34 raspberrypi systemd[1]: Stopped Kibana.
Apr 02 15:57:34 raspberrypi systemd[1]: kibana.service: Start request repeated too quickly.
Apr 02 15:57:34 raspberrypi systemd[1]: kibana.service: Failed with result 'exit-code'.
Apr 02 15:57:34 raspberrypi systemd[1]: Failed to start Kibana.
Apr 02 15:57:37 raspberrypi systemd[1]: kibana.service: Start request repeated too quickly.
Apr 02 15:57:37 raspberrypi systemd[1]: kibana.service: Failed with result 'exit-code'.
Apr 02 15:57:37 raspberrypi systemd[1]: Failed to start Kibana.
pi@raspberrypi:/var/log/elasticsearch $
```

Ilustración 35 - Error Kibana
Elaboración propia

Y ocurre algo similar con Logstach:

```
pi@raspberrypi:/var/log/elasticsearch $ sudo systemctl restart logstash
pi@raspberrypi:/var/log/elasticsearch $ sudo systemctl status logstash
● logstash.service - logstash
   Loaded: loaded (/etc/systemd/system/logstash.service; disabled; vendor preset: enabled)
   Active: activating (auto-restart) (Result: exit-code) since Thu 2020-04-02 15:58:24 CEST; 28ms ago
  Process: 11604 ExecStart=/usr/share/logstash/bin/logstash --path.settings /etc/logstash (code=exited, status=1/FAILURE)
 Main PID: 11604 (code=exited, status=1/FAILURE)
pi@raspberrypi:/var/log/elasticsearch $
```

Ilustración 36 - Error Logstah
Elaboración propia

Por tanto se decide Instalación de un SIEM en otro equipo que tiene como ventajas que no genera problemas de rendimiento sobre el IDS, pero tenemos ahora otro problema que es necesario un sistema de comunicación de ficheros Log entre sistemas y el nuevo equipo no debería estar siempre en funcionamiento, solo cuando se hagan consultas.

3.7. **Topología de las tecnologías del proyecto**

La arquitectura final que se ha usado en la implementación es la que se puede observar en la siguiente imagen:

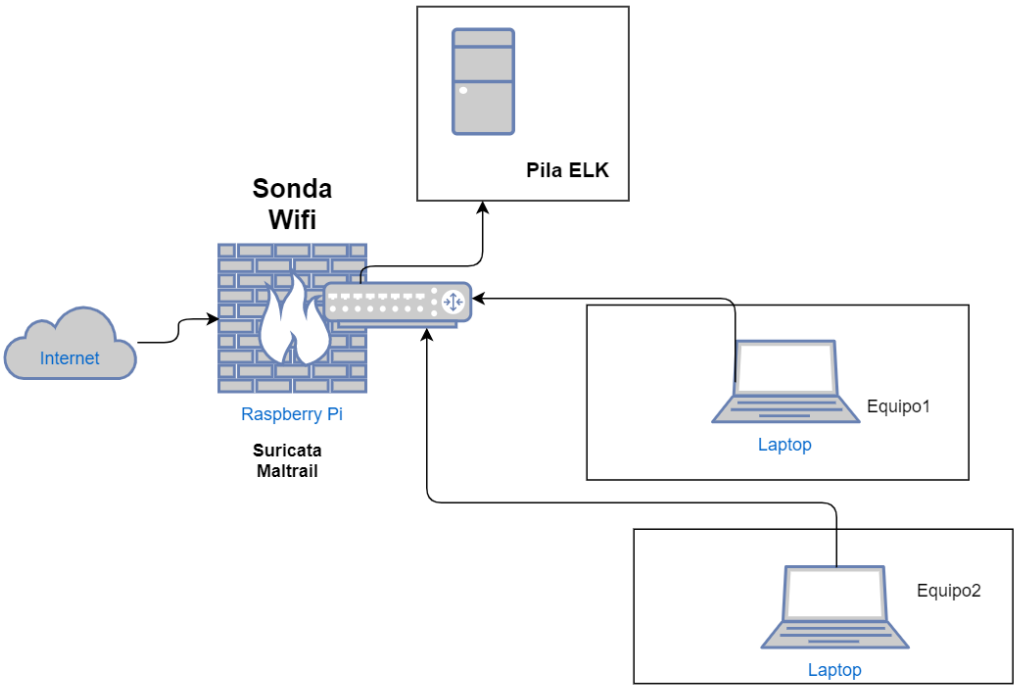

Ilustración 37 - Topología Final
Elaboración propia

Hay que destacar:

- Dentro del Disposito de IOT, la Raspberry Pi 3, se ha incluido los programa de Suricata y MalTrail.
- En una maquina Linux externa, podría haber sido Windows o similar, se ha instalado la Pila ELK, por motivos de eficiencia, y desde un navegador web se podrá obtener la información obtenida.
- Los equipos de mi red wifi se conectan a internet a través del IDS creado para este proyecto.

La arquitectura de la parte de la pila ELK de mi esquema, recoge datos de diferentes fuentes, en mi proyecto solo desde la sonda creada con el dispositivo IOT, pero podría crecer el proyecto en este aspecto, los registros se distribuyen entre todos los servidores que tenemos. Cada servidor puede tener un tipo de registro, y puede haber diferentes rutas para encontrarlos en un servidor.

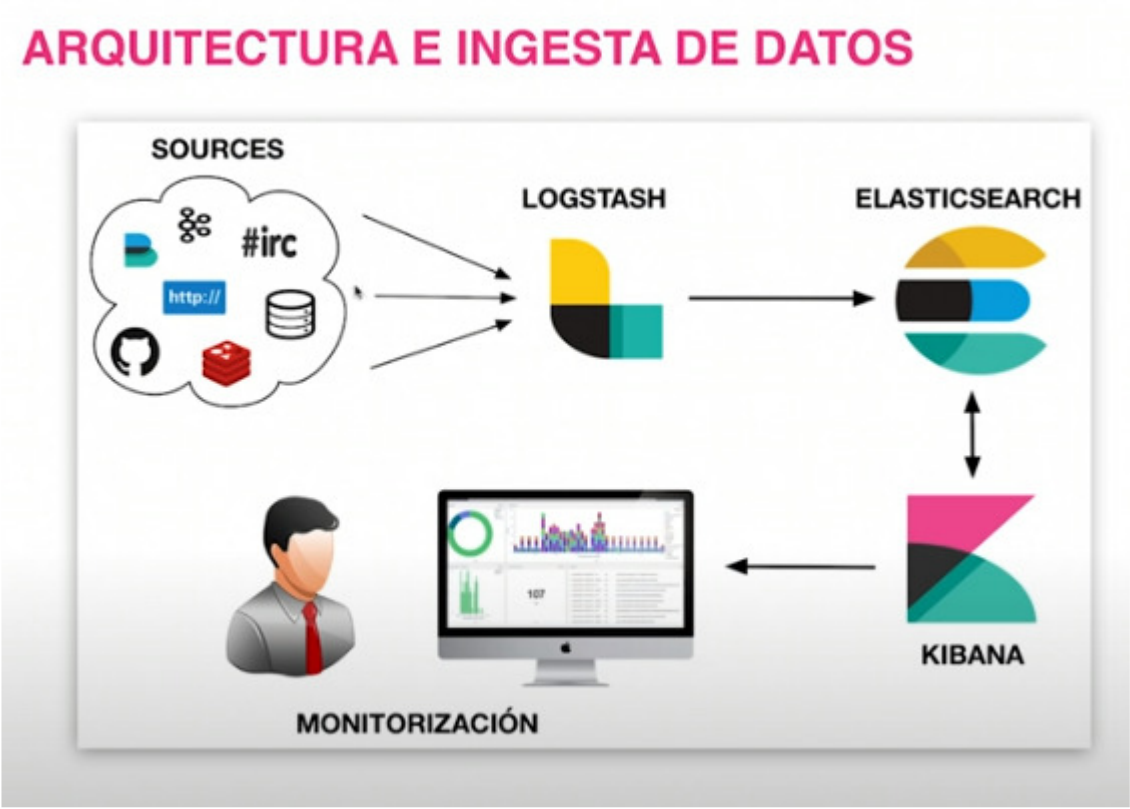

Ilustración 38 - Arquitectura PilaELK
https://www.elastic.co/

Hay personas que no tienen experiencia para entender una línea de log. Algunos logs que tienen mucha información incluida en campos y cuando la información es tanta, a veces es complicado saber a qué representa cada número o cadena de texto.
Tenemos todas las fuentes de datos que envían datos a Logstash, con muchos complementos de entrada y salida, podemos ingresar grandes cantidades de datos. Antes de almacenar estos datos en la base de datos de Elasticsearch, procesará los datos y usará Kibana para instalar un archivo visual que acceda a esta información para monitorearla.

3.8. **Instalación y configuración de MalTrail en Dispositivo IOT**

El siguiente conjunto de comandos debería poner en funcionamiento su Sensor Maltrail (listo para usar con la configuración predeterminada y la interfaz de monitoreo "cualquiera"):

```
sudo apt-get install git python-pcapy
git clone https://github.com/stamparm/maltrail.git
```

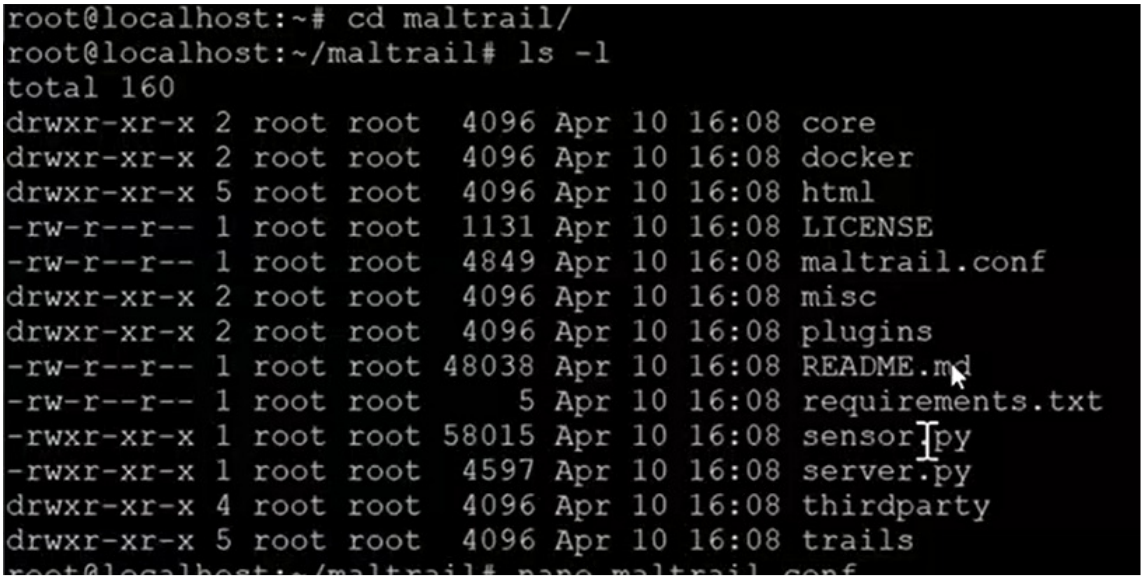

Ilustración 39 - Instalación Mailtrail
Elaboración propia

El siguiente paso ya es ejecutar el sensor. Durante este proceso, actualizará las listas respectivas si fuera necesario.

```
cd maltrail
sudo python sensor.py
```

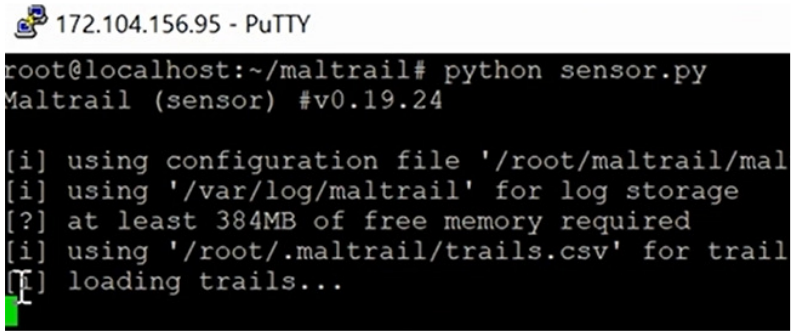

Ilustración 40 - sensor.py
Elaboración propia

Y en nuevo terminal, vamos a ejecutar el servidor:

```
cd maltrail
python server.py
```

Para probar que todo está funcionando, ejecute lo siguiente:

```
ping -c 1 136.161.101.53
cat /var/log/maltrail/$(date +"%Y-%m-%d").log
```

```
stamparm@beast:~$ ping -c 1 136.161.101.53
PING 136.161.101.53 (136.161.101.53) 56(84) bytes of data.
64 bytes from 136.161.101.53: icmp_seq=1 ttl=50 time=115 ms

--- 136.161.101.53 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 115.471/115.471/115.471/0.000 ms
stamparm@beast:~$ cat /var/log/maltrail/$(date +"%Y-%m-%d").log
"2016-01-21 16:01:40.289267" beast 192.168.5.33 - 136.161.101.53 - ICMP IP 136.161.101.53 "conficker (malware)" (static)
stamparm@beast:~$
```

Ilustración 41 - Prueba Maltrail
Origen: Github

Para poder lanzar el cliente, es necesario abril un navegador:

La ip de la maquina: 8338

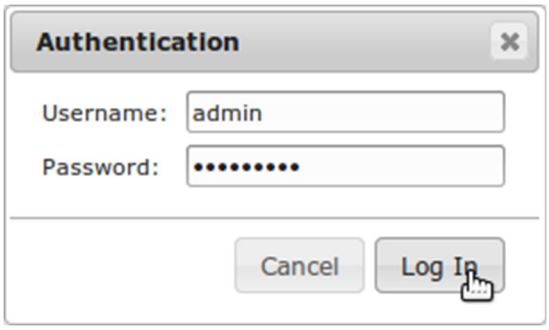

Ilustración 42 - Login Maltrail
Elaboración propia

El usuario por defecto es “admin” con password “changeme!”

Y mostrará lo siguiente en el navegador:

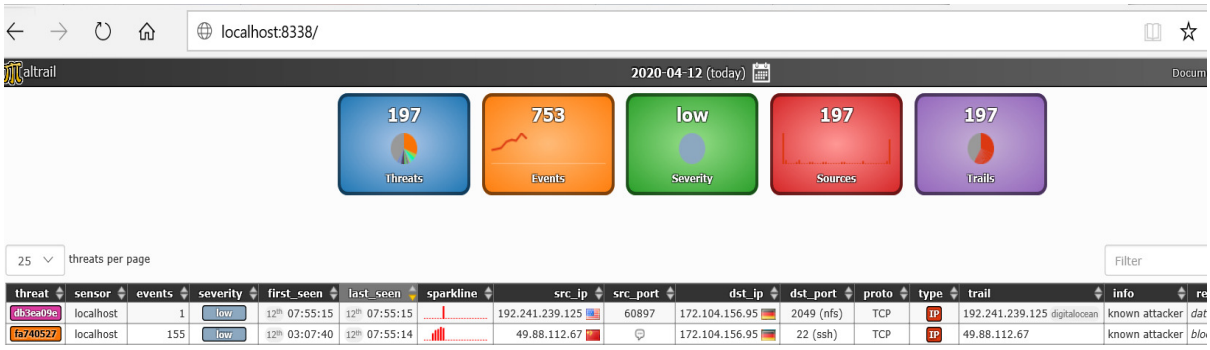

Ilustración 43 - Cliente Maltrail
Elaboración propia

Se restean los logs de Mal Trail y se realiza un ping a una ip maliciosa para comprobar que ocurre y si estamos estableciendo bien la alerta:

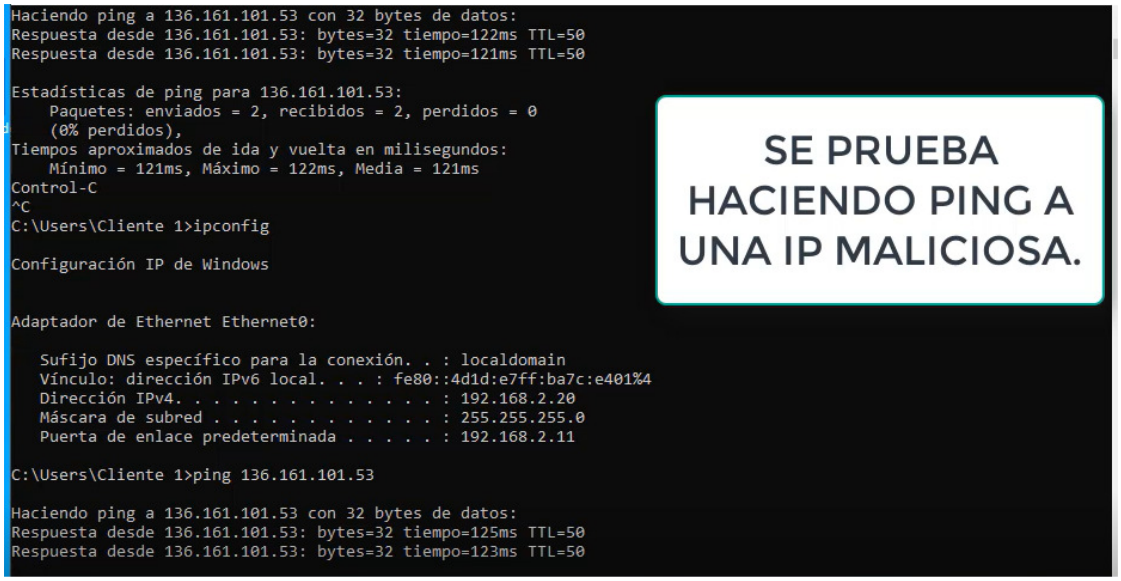

Ilustración 44 - ping maliciosa
Elaboración propia

Y se puedem ver los resultados:

Ilustración 45 – Comporbación ping maliciosa
Elaboración propia

3.9. **Instalación y configuración de Suricata en Dispositivo IOT**

Se ha decidido instalar suricata porque entre otras cosas consume muchos menos recursos, debemos tener la Raspberry pi actualizada con todos los paquetes listos y ejecutar el comando:

```
sudo sudo apt-get install suricata
```

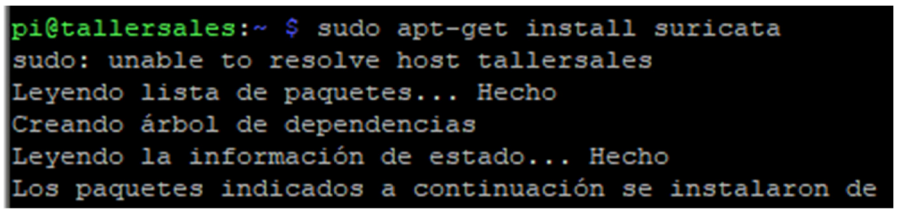

Ilustración 46 - Instalación Suricata
Elaboración propia

Una vez instalado, procedemos a configurarlo, aunque antes de eso, hay que comprobar si el servicio está funcionando correctamente:

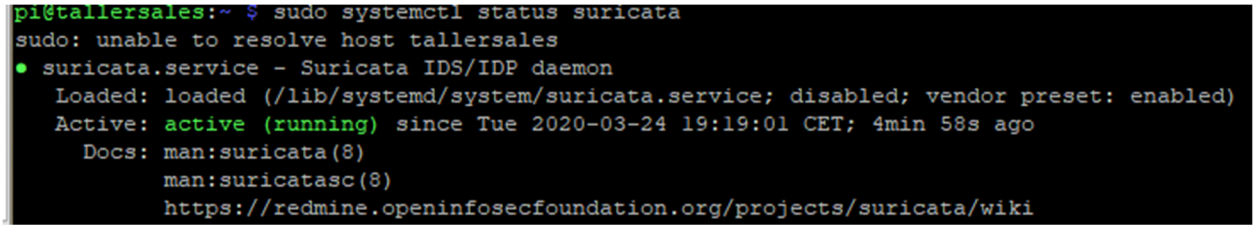

Ilustración 47 - Funcionamiento Suricata
Elaboración propia

Con el comando service podemos comprobar que el servicio está bien instalado. Si da algún error se podrá observar en el fichero /var/log/suricata/suricata.log

3.9.1. **Preparación de reglas**

Cuando hayamos comprobado el estado del servicio, crearemos nuestras primeras reglas. Lo realizaremos creando un archivo dentro de la carpeta “/etc/suricata/rules”. El nombre de la regla que he creado es “ejemplo.rules”, y después de esto, se configura:.

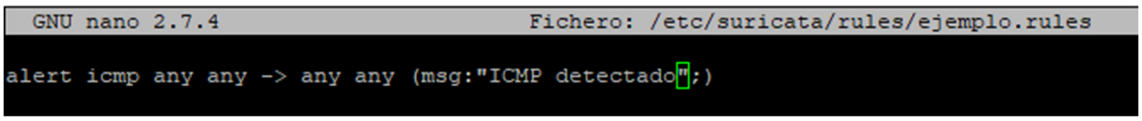

Ilustración 48 - Ejemplo Regla
Elaboración propia

Esta regla lo que hace es, básicamente, sacar una alerta (visible en los logs) cada vez que detecte un paquete ICMP.

Otra forma es usar reglas predefinidas de configuración como el comando suricata-oinkmaster-updater que realiza la actualización de reglas de un repositorio , es otra forma de nuestro sistema pueda estar alerta cuando ocurra alguna de las mismas. Utilizaremos el comando suricata-oinkmaster-Updater para realizar esta actualización de la regla del repositorio https://rules.emergingthreats.net/. Con esto, estamos listos para permitir que el sistema active una alarma cuando se detecte cualquier amenaza

```
# suricata-oinkmaster-updater
```

3.9.2. Configuración del Servicio

Una vez tenemos el IDS Suricata instalado en la Raspberry-pi podemos realizar algunos ajustes en su configuración para adaptarlo a nuestras necesidades.
Los ficheros de configuración de Suricata están en /etc/suricata.
Una vez que hayamos creado esta regla, debemos añadirla al fichero de configuración del programa, **"/etc/suricata/suricata.yaml"**, para que tenga efecto:

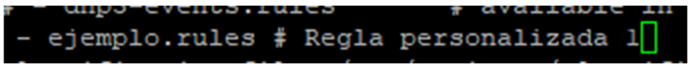

Ilustración 49 - Regla personalizada
Elaboración propia

Para poner en marcha los cambios aplicados (en este caso la regla que hemos añadido), debemos reiniciar el servicio:

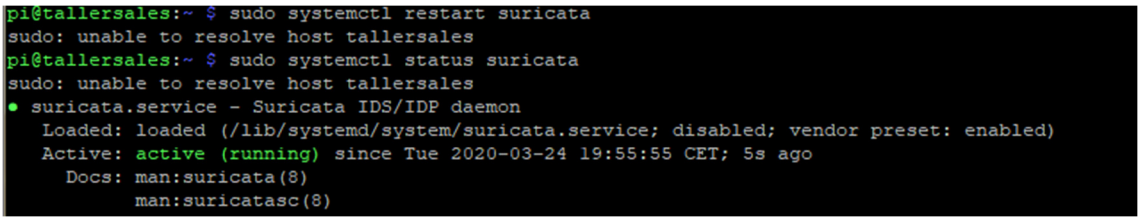

Ilustración 50 - Reinicio de Servicio
Elaboración propia

Ahora para comprobar que el programa y la alerta funcionan correctamente, se realiza un ping al DNS de Google (8.8.8.8).

```
ping 8.8.8.8
```

Una vez realizado, se comprueba el fichero de log "/var/log/suricata/fast.log" y con el comando tail –f, comprobamos si se ha guardado correctamente:

Ilustración 51 - log de suricata
Elaboración propia

Se demuestra que la regla funciona correctamente y recoge bien la regla que se ha establecido, en este caso cualquier tipo de tráfico ICMP.

Se puede comprobar que está funcionado viendo los logs propios de suricata via web:

04/14/2020 00:43:04	3	UDP	Generic Protocol Command Decode	192.168.2.11	53	192.168.2.20	63763	1:2200075	SURICATA UDPv4 invalid checksum
04/14/2020 00:43:04	3	UDP	Generic Protocol Command Decode	192.168.2.11	53	192.168.2.20	63763	1:2200075	SURICATA UDPv4 invalid checksum
04/14/2020 00:43:04	3	TCP	Generic Protocol Command Decode	192.168.2.20	53257	192.168.1.1	80	1:2260002	SURICATA Applayer Detect protocol only one direction
04/14/2020 00:43:04	3	UDP	Generic Protocol Command Decode	192.168.2.11	53	192.168.2.20	56363	1:2200075	SURICATA UDPv4 invalid checksum
04/14/2020 00:43:04	3	UDP	Generic Protocol Command Decode	192.168.2.11	53	192.168.2.20	58273	1:2200075	SURICATA UDPv4 invalid checksum
04/14/2020 00:43:01	3	UDP	Generic Protocol Command Decode	192.168.2.11	53	192.168.2.20	64091	1:2200075	SURICATA UDPv4 invalid checksum

Ilustración 52 - Logs de Suricata via web
Elaboración propia

3.10. **Instalación y Configuración de SIEM**

Teniendo en cuenta que la instalación de la pila ELK completa en la Raspberry Pi afectará el rendimiento del sistema, ya que es un dispositivo con insuficientes IDS de Suricata para el manejo de energía. Se intento instalar el componente Log Management solo en Raspberry pi, pero además de los problemas con el paquete de software ARM, siempre debemos tener el servidor ElasticSearch ejecutándose para recopilar datos, y no ha funcionado.

Lo primero es instalar ElasticSearch. Elasticsearch es un motor de análisis de código abierto que puede procesar todo tipo de datos, incluidos texto, números, geoespaciales, estructurados y no estructurados. Elasticsearch fue desarrollado por Apache y fue presentado por primera vez por Elasticsearch N.V en 2010. (Ahora se llama elástico). Elasticsearch es conocido por su API REST simple, características distribuidas, velocidad y escalabilidad. Es el componente central de Elastic Stack. Elastic Stack es un conjunto de herramientas de código abierto para extracción de datos, enriquecimiento, almacenamiento, análisis y visualización. Las pilas elásticas a menudo se denominan pilas ELK (nombradas por Elasticsearch, Logstash y Kibana), y ahora incluyen una gran colección de agentes de transporte llamados Beats para enviar datos a Elasticsearch.

Descargar e instalar la clave de firma pública:

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
```

Guarde la definición del repositorio en /etc/apt/sources.list.d/elastic-7.x.list:

```
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
```

Es posible que deba instalar el apt-transport-https en Debian antes de continuar:

```
sudo apt-get install apt-transport-https
```

3.10.1. Instalación y configuración de Elasticsearch

Pasamos a Instalar el programa:

```
sudo apt-get update && sudo apt-get install elasticsearch
```

Elasticsearch tiene tres archivos de configuración:

- **elasticsearch.yml** para configurar Elasticsearch
- **jvm.options** para configurar los ajustes de Elasticsearch JVM
- **log4j2.properties** para configurar el registro de Elasticsearch

Vamos a modificar la dirección IP del servidor, el puerto y la red de busqueda modificando el fichero **elasticsearch.yml:**

```
network.host: 192.168.0.15
http.port: 9200
discovery.seed_hosts: ["host1", "host2"]
```

Indicando la ip que corresponda al servidor.

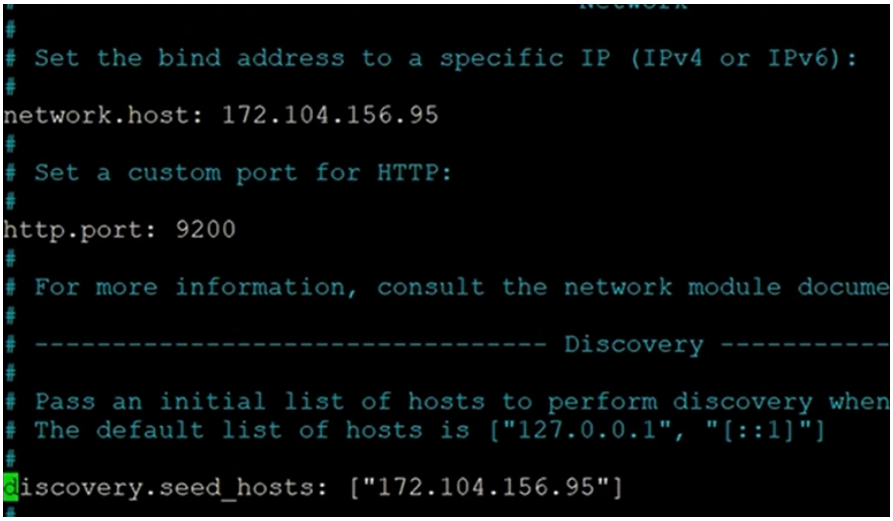

Ilustración 53 - Ejemplo configuración elasticsearch
Elaboración propia

Ahora es necesario reiniciar el servicio:

```
sudo systemctl start elasticsearch.service
sudo systemctl stop elasticsearch.service
```

Vamos a comprobar que está funcionando, poniendo en un navegador la IP y el puerto 9200:

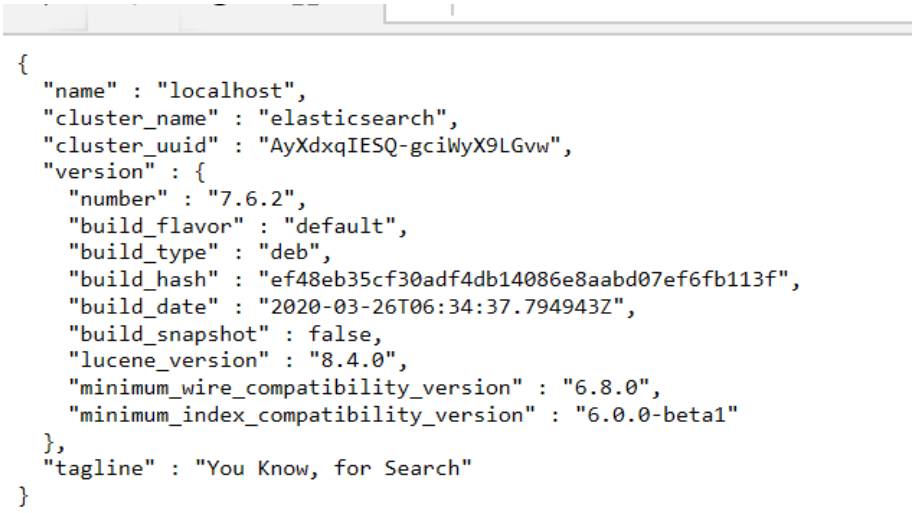

Ilustración 54 - Respuesta ElasticSearch
Elaboración propia

3.10.2.**Instalación y configuración de Logstash**

El siguiente paso es instalar Logtash. Logstash ingerirá, transformará y enviará dinámicamente sus datos, independientemente de su formato o complejidad. Utilizar grok para obtener estructuras a partir de datos no estructurados, descifrar las coordenadas geográficas de las direcciones IP, anonimizar o excluir campos sensibles y facilitar el procesamiento de rutina.
Los datos a menudo se encuentran dispersos o aislados en muchos sistemas en varios formatos. Logstash admite varias entradas que pueden extraer eventos de múltiples fuentes comunes al mismo tiempo. Extraiga fácilmente datos de registros, métricas, aplicaciones web, almacenes de datos y diversos servicios de AWS en un flujo continuo.

Una vez que hemos actualizado los paquetes, con ElasticSeach, instalar logstah es muy sencillo:

```
sudo apt-get update && sudo apt-get install logstash
```

Tenemos que modificar el fichero /etc/logstash/conf.d/logstash.conf para que pueda leer del log que hemos instalado en SURICATA y pueda hacer en análisis de datos de mi sonda.
En cualquier caso, la configuración siempre consta de una sección input y otra output.
También es seguro que se definirá la sección filter, que es uno de los puntos fuertes de Logstash y por lo que destaca sobre otras opciones más ligeras.
Un ejemplo del como deber el formato del archivo:

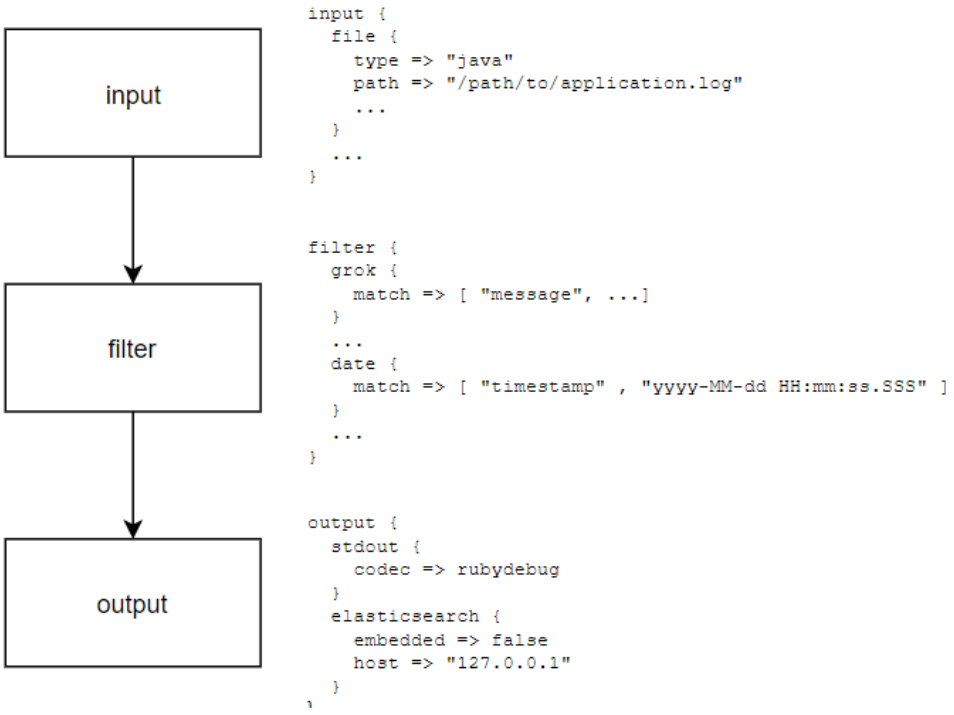

Ilustración 55 - Formato de Fichero Logstash
Origen: Github

El fichero debería quedar así:

```
input {
  file {
    path => ["/var/log/suricata/eve.json"]
    sincedb_path => ["/var/lib/logstash/"]
    codec =>   json
    type => "SuricataIDPS"
  }

}

filter {
  if [type] == "SuricataIDPS" {
    date {
      match => [ "timestamp", "ISO8601" ]
    }
    ruby {
      code => "if event['event_type'] == 'fileinfo';
event['fileinfo']['type']=event['fileinfo']['magic'].to_s.split(',')[0];
end;"
    }
  }

  if [src_ip]  {
    geoip {
      source => "src_ip"
      target => "geoip"
      #database => "/opt/logstash/vendor/geoip/GeoLiteCity.dat"
      add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}" ]
      add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}"  ]
```

```
    }
    mutate {
      convert => [ "[geoip][coordinates]", "float" ]
    }
    if ![geoip.ip] {
      if [dest_ip]  {
        geoip {
          source => "dest_ip"
          target => "geoip"
          #database => "/opt/logstash/vendor/geoip/GeoLiteCity.dat"
          add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}" ]
          add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}"  ]
        }
        mutate {
          convert => [ "[geoip][coordinates]", "float" ]
        }
      }
    }
  }
}

output {
  elasticsearch {
    host => localhost
    #protocol => http
  }
}
```

Es la herramienta grok la que se encarga de parsear de la siguiente manera. El funcionamiento de Logstash se basa en la definición e integración de entradas, códecs, filtros y salidas: Las entradas son diferentes fuentes de datos. El códec se utiliza para convertir diferentes formatos de entrada a otro formato aceptado por Logstash, lo que nos permitirá unificar el formato del registro y posiblemente leer formatos que no sean texto sin formato. Los filtros nos permiten manejar diferentes eventos. Las salidas son objetivos diferentes donde se almacenan los datos procesados.
Esta imagen refleja lo explicado:

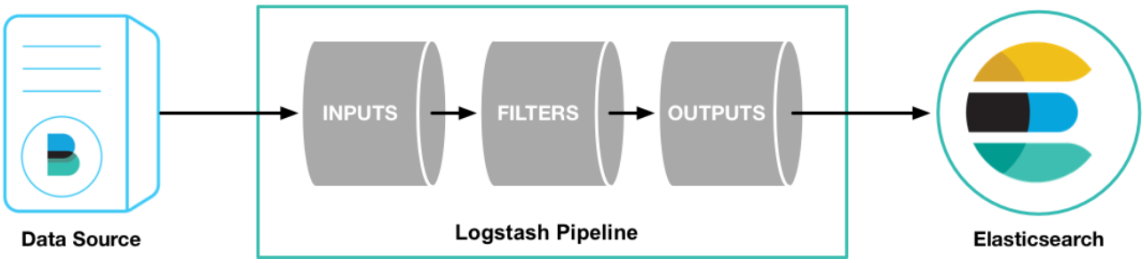

Ilustración 56 - Funcionamiento Logstah
Via: https://enimbos.com/monitor/

3.10.3. **Proceso de transferencia de datos con Filebeat**

Para realizar la carga de datos de la sonda IDS y visualizarla en Kibana, se debe establecer algún proceso de transferencia de archivos. En el proceso de diseño del sistema de carga de datos, aparecieron una serie de problemas, estos problemas afectaron el plan y se explicarán en la parte de mejoras en el proceso de implementación. La solución seleccionada para el diseño del proceso de carga de datos debe tener en cuenta que los datos se cargan desde la Sonda a atraves de Suricata y MalTrail. Los datos deben recogerse por el nodo de ElasticSearch, que es otro equipo.
Se ha decidido usar Filebeat. Se trata de un software escrito en go que va recopilando todos los logs de los ficheros que le indiquemos y los envía a nuestra base de datos elasticsearch.
Para instalar filebeat necesitaremos el repositorio de elastic que añadimos en la entrada de las métricas.

```
apt install filebeat
```

Una vez que lo instalemos tenemos nuestro fichero de configuración en **/etc/filebeat/filebeat.yml**
Pasamos a comentar los aspectos más importantes, de la configuración que debemos de modificar:
La parte de entrada:

```
# List of inputs to fetch data.
filebeat.inputs:
```

```
#------------------------------ Log input -------------------------------
- type: log

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

  # Paths that should be crawled and fetched. Glob based paths.
  # To fetch all ".log" files from a specific level of subdirectories
  # /var/log/*/*.log can be used.
  # For each file found under this path, a harvester is started.
  # Make sure not file is defined twice as this can lead to unexpected behaviour
  paths:
    - /var/log/*.log
    #- c:\programdata\elasticsearch\logs\*
```

Ilustración 57 - Entrada Firebeat
https://www.elastic.co/

La parte de salida del mismo fichero:

```
#================================ Outputs =====================================

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

#-------------------------- Elasticsearch output ------------------------------
output.elasticsearch:
  # Boolean flag to enable or disable the output module.
  #enabled: true

  # Array of hosts to connect to.
  # Scheme and port can be left out and will be set to the default (http and 9200
  # In case you specify and additional path, the scheme is required: http://local
  # IPv6 addresses should always be defined as: https://[2001:db8::1]:9200
  hosts: ["localhost:9200"]
```

```
 # Logging to rotating files. Set logging.to_files to false to disable logging to
 # files.
 logging.to_files: true
 logging.files:
   # Configure the path where the logs are written. The default is the logs direc
   # under the home path (the binary location).
   #path: /var/log/filebeat
```

Ilustración 58 - Filebeat- Salida
https://www.elastic.co

Una vez hemos configurado el sistema, tenemos que ponerlo en marcha:

```
#filebeat modules enable system
```

Instalación y configuración de Kibana

Por último nos queda instalar Kibana. Kibana es una aplicación front-end de código abierto sobre Elastic Stack, que proporciona funciones de visualización y búsqueda de datos indexados en Elasticsearch. A menudo, se hace referencia a Kibana como la herramienta de representación para Elastic Stack (anteriormente conocida como ELK Stack por Elasticsearch, Logstash y Kibana), y también sirve como una interfaz de usuario para monitorear, administrar y proteger los clústeres de Elastic Stack. Y un centro centralizado para soluciones integradas desarrolladas en Elastic Stack. Kibana se desarrolló en la comunidad Elasticsearch en 2013. Al proporcionar portales para usuarios y empresas, se ha convertido en la ventana de Elastic Stack.
Una vez que hemos actualizado los paquetes, con ElasticSeach, instalar Kibana es:

```
sudo apt-get update && sudo apt-get install kibana
```

Para Iniciar el Servicio:

```
sudo systemctl start kibana.service
```

Para la configuración de Kibana, debemos editar el fichero /opt/kibana/config/kibana.yml:

```
server.port: 5601
server.host: "IP_HOST_KIBANA"
elasticsearch.url: "http://IP_HOST_ELASTICSEARCH:9200"
```

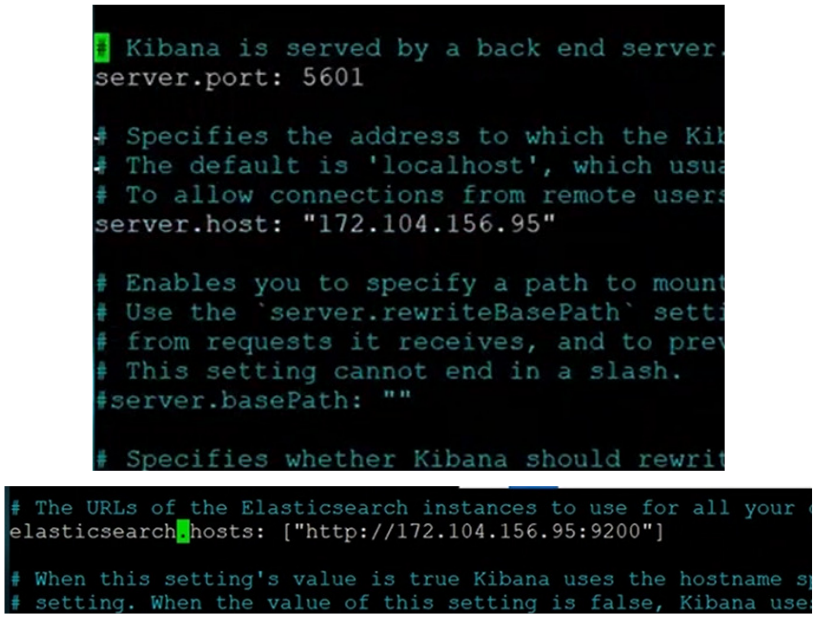

Ilustración 59 - Configuración kibana
Elaboración propia

Para entrar en la aplicación de Kibana, ponemos la IP:5601 en un navegador:

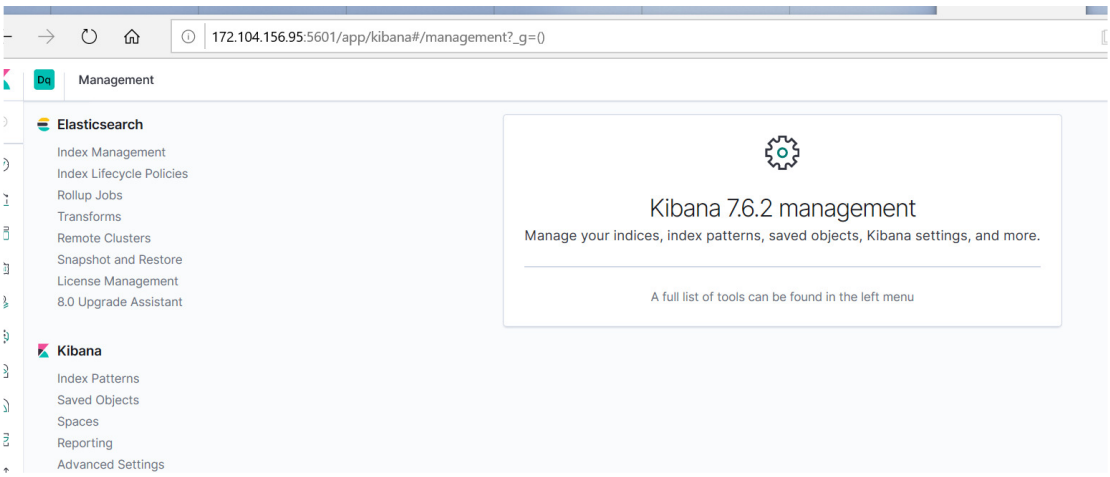

Ilustración 60 - Aplicación de Kibana
Elaboración propia

Lo siguiente es crear los índices, empezamos con los índices de Kibana:

Se han dejado solo activos solamente los de logstash.

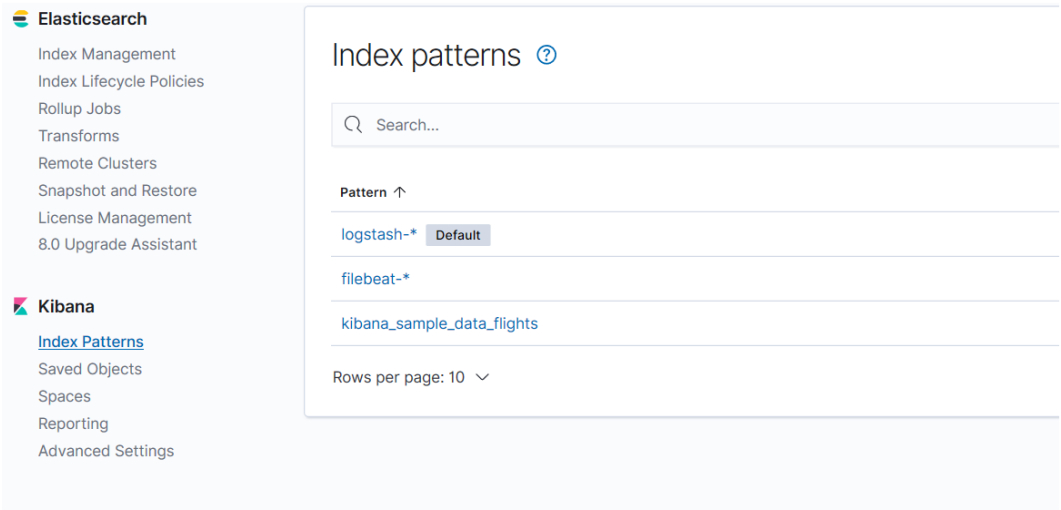

Ilustración 61 Indices1 Logstash
Elaboración propia

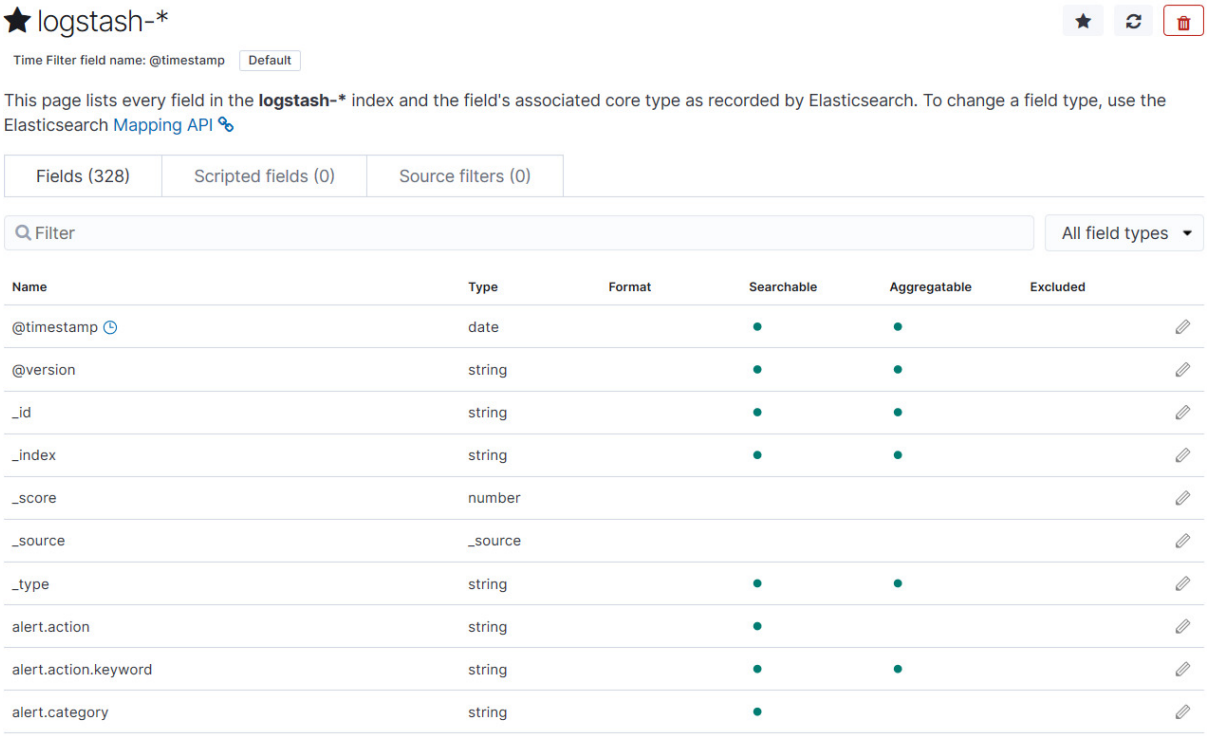

Ilustración 62 – Indices2 Logstash
Elaboración propia

Podemos empezar a ver todo el tráfico que está pasando por la sonda y es capaz de leer Suricata:

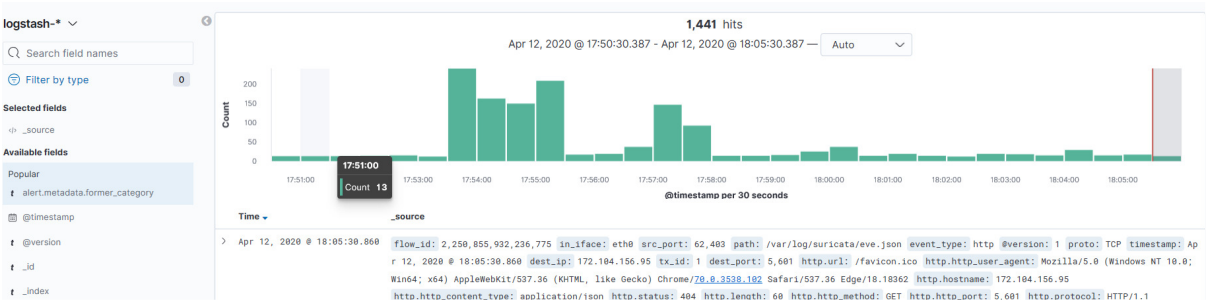

Ilustración 63 - Datos suricata
Elaboración propia

Vamos a crear un gráfico de “pie”, con la información de los logstash que esta leyendo y que se ha configurado en el índice, en los últimos 15 minutos:

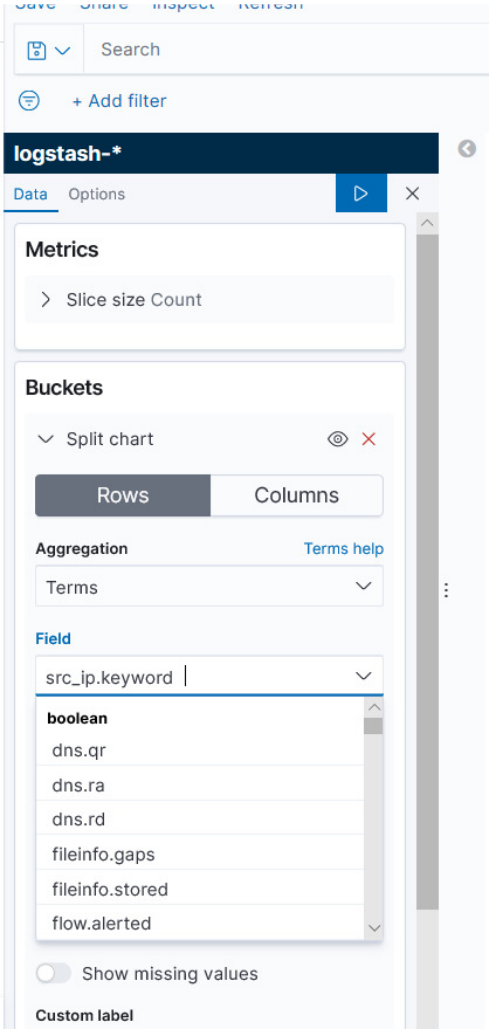

Ilustración 64 - Configuración grafico
Elaboración propia

Se vería el resultado de el gráfico:

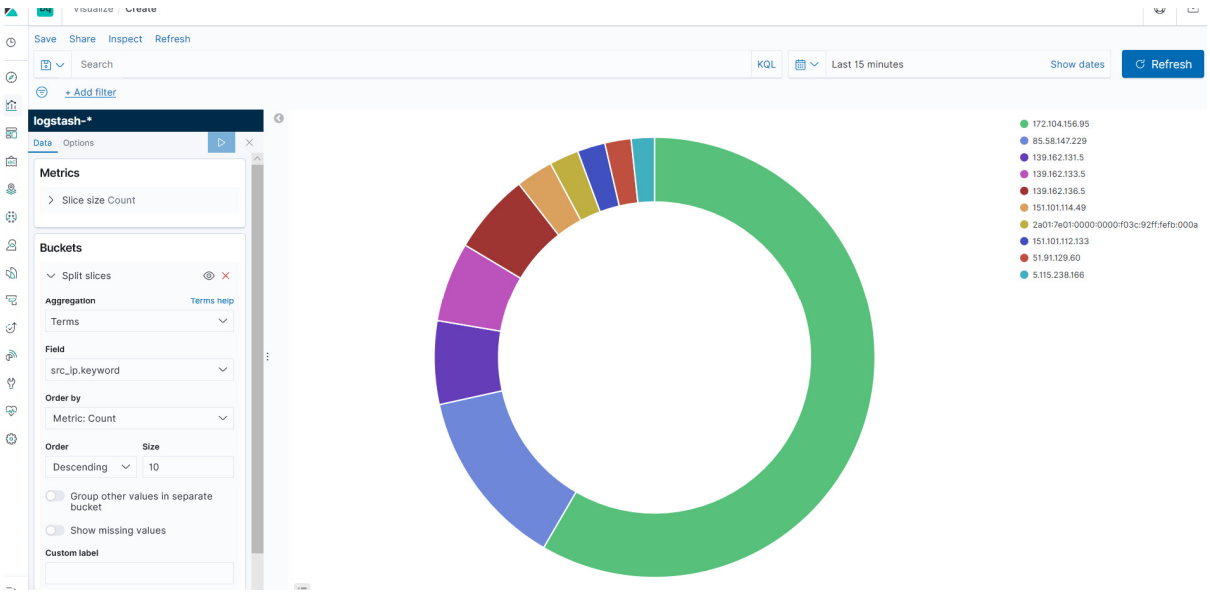

Ilustración 65- Grafico Kibana- tipo Pie
Elaboración propia

En parte de “dashboard”, se puede ir viendo las alertas totales que han ocurrido en los últimos 15 minutos, los tipos de eventos por horas, los “hunting”, que son ataques más específicos sobre algo concreto:

Ilustración 66 - Alertas Suricata
Elaboración propia

Aquí podemos ver un ejemplo de los 2 hunting para poder tomar medidas sobre lo que esta ocurriendo, en este ejemplo se ve que están intentando entrar por el puerto 3306 que corresponde con las base de datos de mysql.

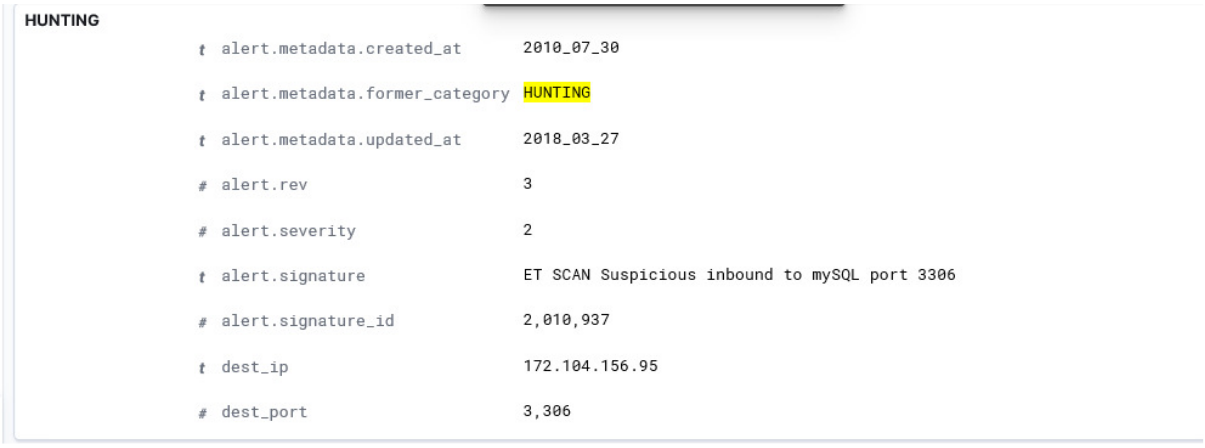

Ilustración 67 - Hunting Suricata
Elaboración propia

Se puede ver la IP que me hizo el ataque:

Ilustración 68 - IP atacante – suricata
Elaboración propia

Vamos a comprobar con Shodan. Shodan es un motor de búsqueda en el que, a diferencia de Google y otros buscadores, no podemos buscar, por ejemplo, una imagen o un texto.

Este motor de búsqueda está enfocado únicamente a buscar sistemas y servicios conectados a internet. Por este motivo, Shodan está clasificado como uno de los motores de búsquedas más peligrosos, por todo el contenido que tiene.

En esta imagen, probando el ataque de mysql se ve que la IP atacante es de San Francisco.

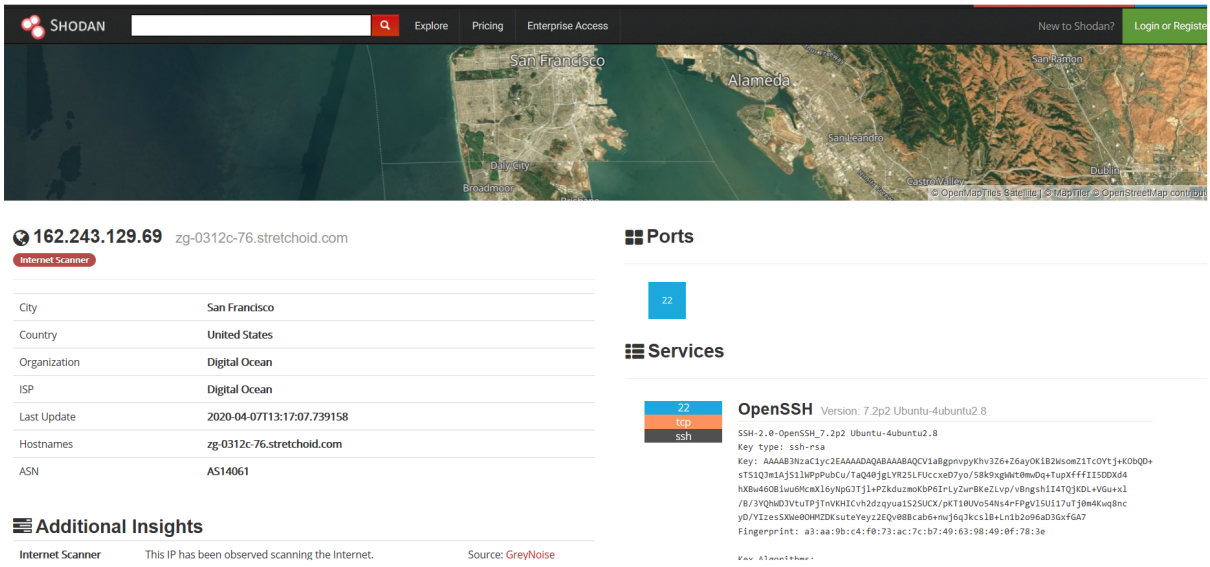

Ilustración 69 - Datos con Shodan
Elaboración propia

Otro ejemplo es la localización o mapeo. Estamos usando los datos que se encuentran almacenados en los logs:

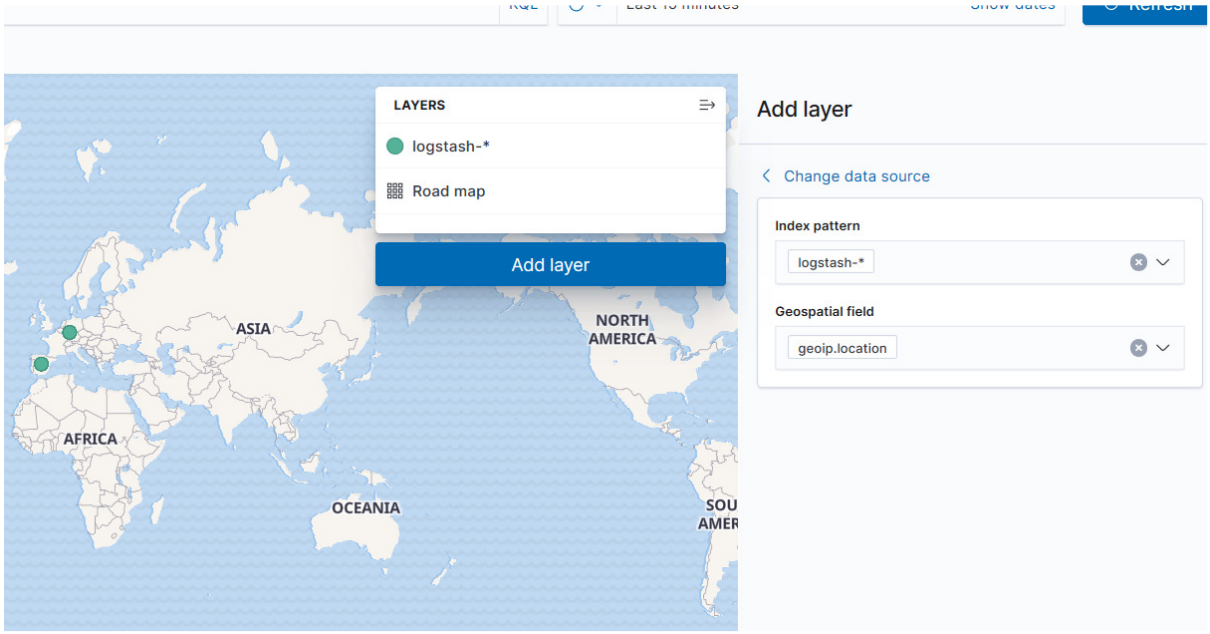

Ilustración 70 - Configuración mapa
Elaboración propia

En esta imagen podemos comprobar que en la parte izquierda, hay un menú, para poder elegir “index patten”, que es el origen de los datos. Hay muchas opciones, pero para que nuestro proyecto se ha elegido “logstash” que recibe los datos de la sonda de Suricata,y debajo se ve el campo “geoespacial” para poder mostrar los datos en el mapa elegido.

Así puedo observar donde están las conexiones que recibo:

Ilustración 71 - Geolocalización de IP
Elaboración propia

Se le pueden añadir muchos filtros a Kibana, para buscar por diferentes formatos, uno típico es saber desde que sistemas operativos se conectan a mi red:

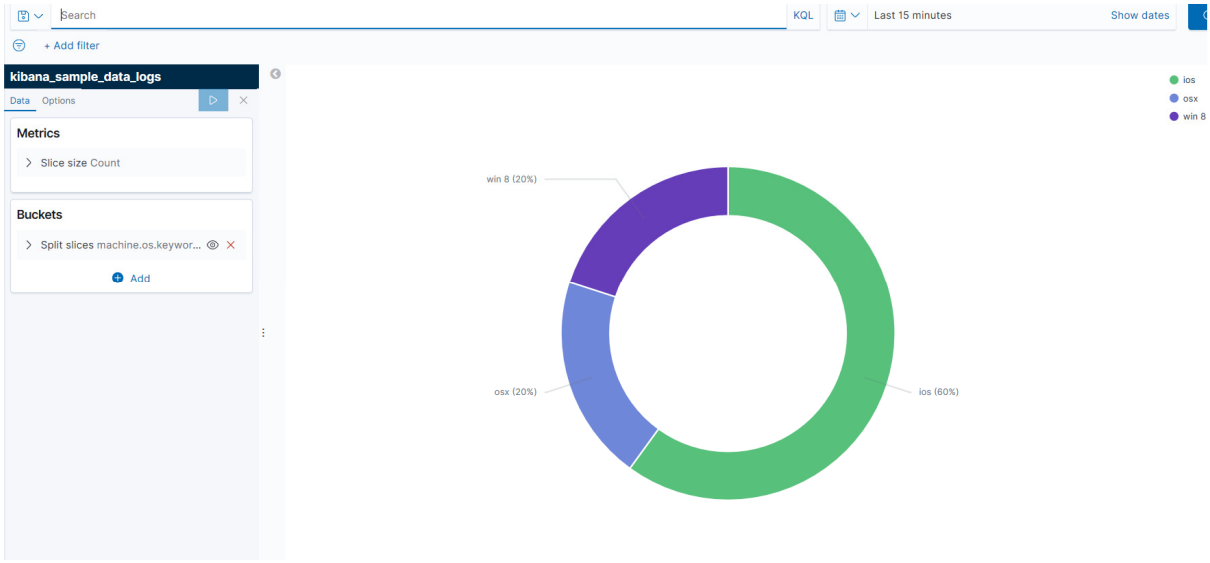

Ilustración 72 - Kibana-ssoo
Elaboración propia

Se puede observar en la parte izquierda la métrica ultizada, en la parte de arriba el tiempo elegido para mostrar el gráfico. En este ejemplo se va que se han concetado un 60% desde un dispositivo con el sistema ios.

Hay que tener en cuenta de que Kibana desde el dashborad, nos avisa con mensajes tipo “popup” de las alertas que van surgiendo en “directo” en el momento de ocurre la situación o anomalia:

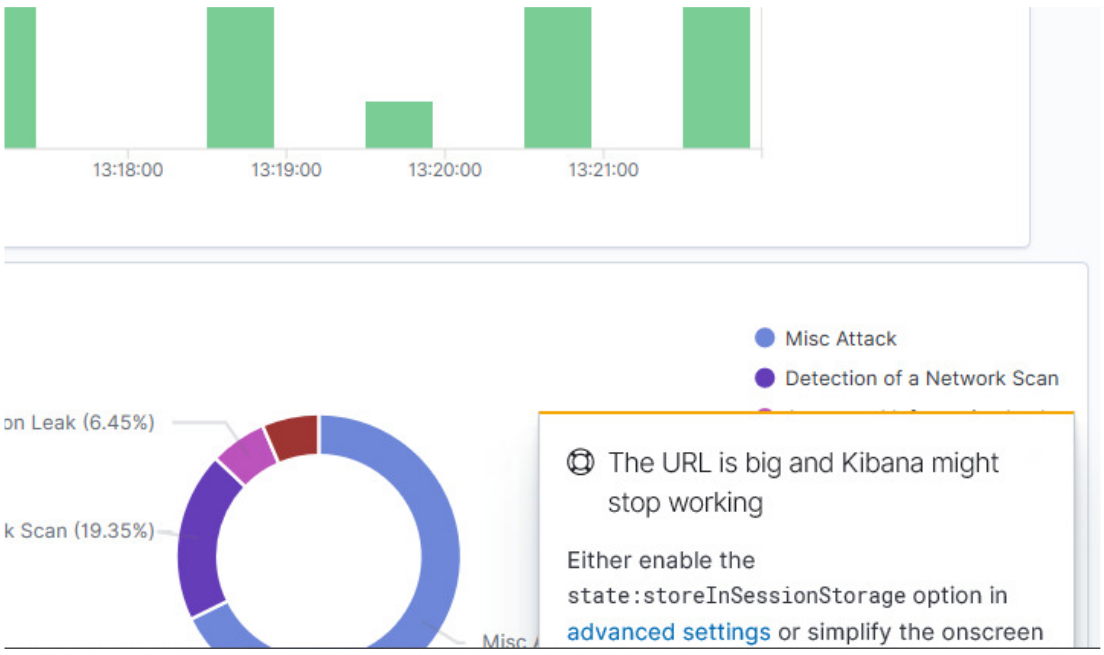

Ilustración 73 - Alertas Kibana
Elaboración propia

En la parte de “Discover” de Kibana muestra las diferentes situaciones que están ocrriendo en tiempo real y podemos ver lo que ocurre en tiempos anteriores:

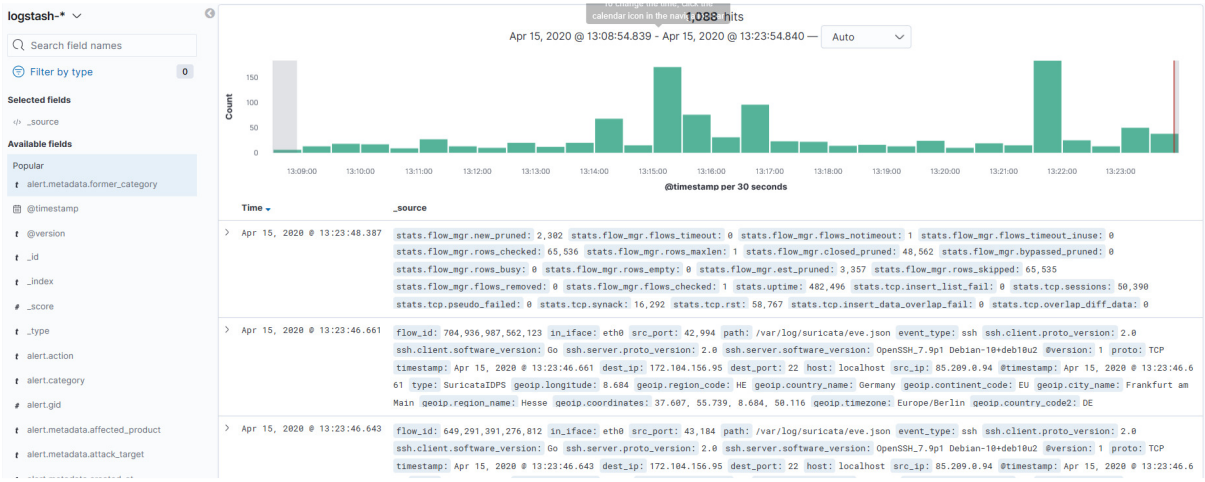

Ilustración 74 - Discover kibana
Elaboración propia

Podemos establecer alertas, crearlas y administrarlas desde una sola UI. Nos podemos mantenter informados con actualizaciones en tiempo real sobre qué alertas se están ejecutando y qué acciones se tomaron.

Se puede crear un “Dashborad” a la carta para ver las alertas que quiero que queden reflejadas dentro de mi estudio o trabajo.
Mostramos una vista general:

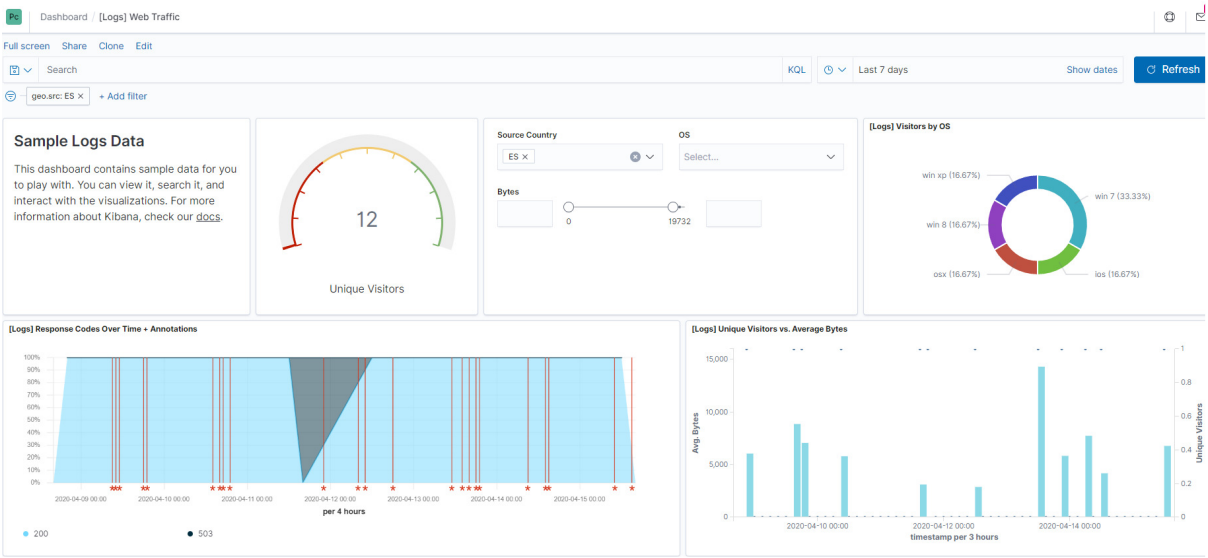

Ilustración 75 - Dashboard general - 1
Elaboración propia

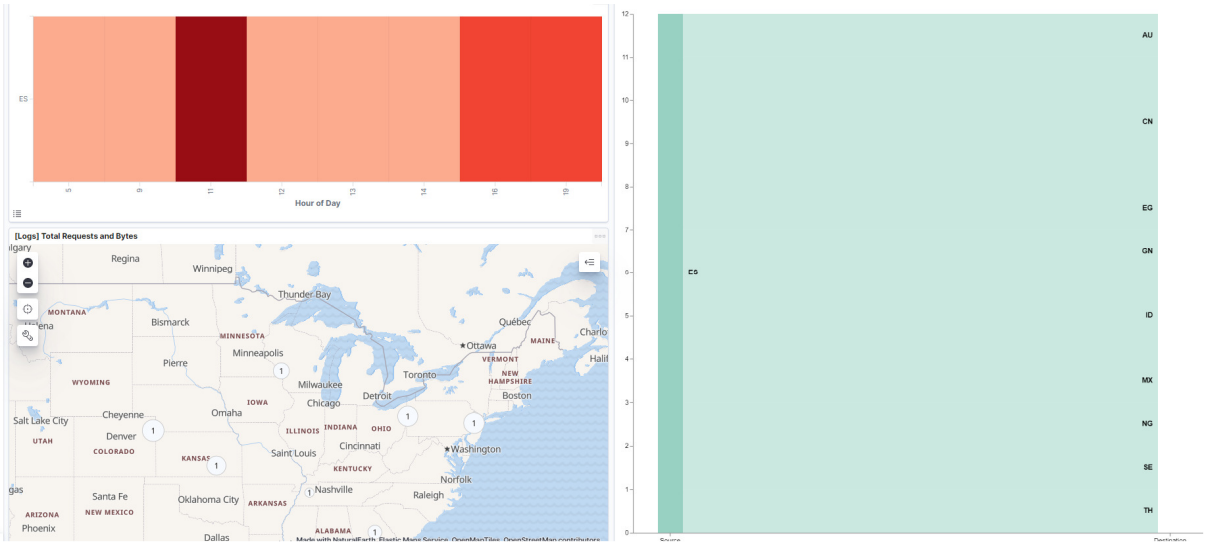

Ilustración 76 Dashboard general - 2
Elaboración propia

Vamos a ver cada parte de forma más especifica:

Podemos ver los visitantes únicos:

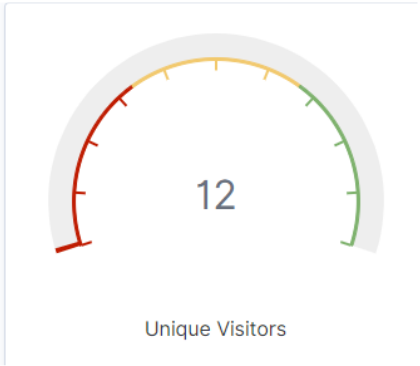

Ilustración 77 - Visitantes unicos
Elaboración propia

Buscar un patrón, para que mismo Dashboard se actualice automáticamente:

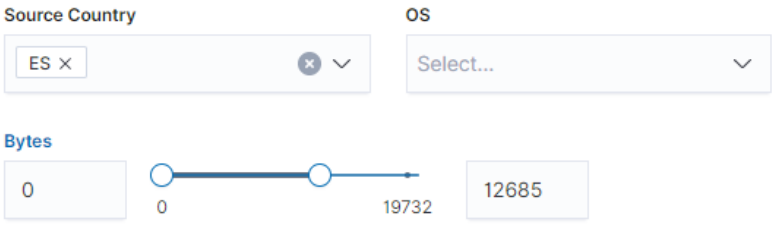

Ilustración 78 - Patrón Dashboard
Elaboración propia

También se puede ver los registros de respuesta a lo largo del tiempo:

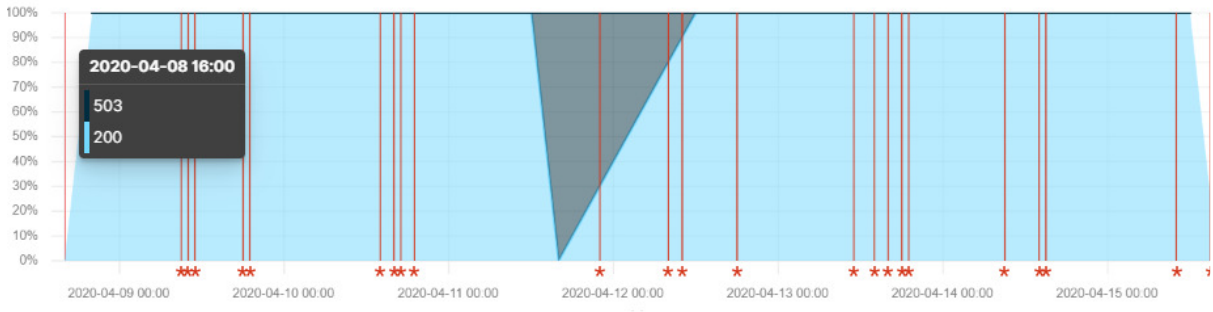

Ilustración 79 - Respuestas
Elaboración propia

Podemos ver los visitantes únicos:

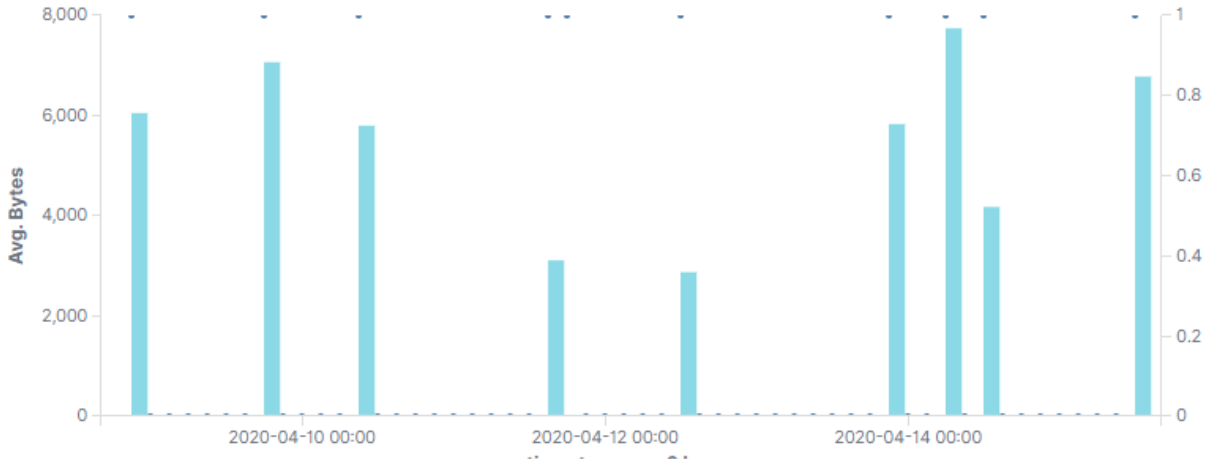

Ilustración 80 - Visitantes únicos
Elaboración propia

Podemos ver los bytes/Host:

[Logs] Host, Visits and Bytes Table

Type	Bytes (Total)	Bytes (Last Hour)	Unique Visits (Total)	Unique Visits (Last Hour)
	16.7KB	0B	3	0
css	15.3KB	0B	3	0
gz	8.7KB	0B	3	0
deb	7.6KB	0B	1	0

Ilustración 81 - Tamaño/host
Elaboración propia

Con la parte de “Canvas” podemos crear informes individualizados y pensados en nuestros posibles clientes.

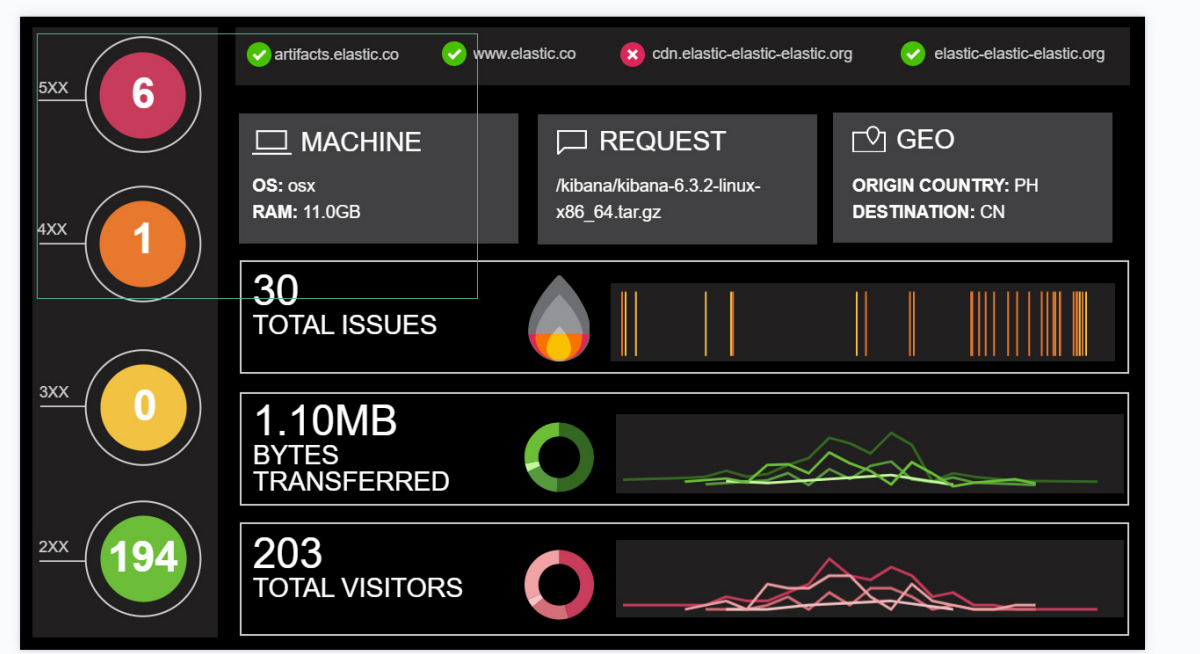

Ilustración 82 - Canvas Kibana
Elaboración propia

3.11. **Otra forma de comunicación de Logs como el SIEM**

Debido a que nuestra máquina podemos hacer que no este siempre encendida para la recogida de los logs, éstos seguirán guardándose en la Raspberry, y serán enviados por SFTP hacia la máquina virtual, siendo este protocolo encriptado y una forma segura de transferencia de archivos.
Se usará logrotate en la Raspberry para que cada día el archivo eve.json sea uno nuevo, y el antiguo se renombrará, para tener los logs organizados por días.
Se editará el fichero:

```
sudo nano /etc/logrotate.d/suricata
```

Por defecto, se nos crea automáticamente el archivo de esta forma :

```
/var/log/suricata/*.log
/var/log/suricata/*.json
{
        rotate 14
        missingok
        compress
        copytruncate
        sharedscripts
        postrotate
                /bin/kill -HUP $(cat /var/run/suricata.pid)
        endscript
}
```

Ilustración 83 - fichero logrotate v1
Elaboración propia

Se establece otra configuración para logrotate. Se dejará de la siguiente forma:

```
/var/log/suricata/*.log
/var/log/suricata/*.json
{
        daily
        rotate 15
        missingok
        nocompress
        dateext
        dateformat -%d-%m-%Y-.log
        create 666 pi pi
        sharedscripts
        postrotate
                /bin/kill -HUP $(cat /var/run/suricata.pid)
        endscript
}
```

Ilustración 84 - fichero logrotate v2
Elaboración propia

Lo que indica que se debe trabajar sobre los archivos .json y .log de la carpeta /var/log/suricata/, de forma diaria y que se deberán conservar los 15 últimos logs, aunque estén vacíos, sin comprimir y con el formato de fecha día-mes-año-hora-minuto, propiedad del usuario y grupo pi, y que después de realizar la rotación se reinciará el proceso de Suricata.

Forzaremos el rotado de los logs con logrotate -f /etc/logrotate.d/suricata:

```
pi@raspberrypi:/var/log/suricata $ ls -l
total 0
-rw-r--r-- 1 pi pi 0 Apr 17 16:27 eve.json
-rw------- 1 pi pi 0 Apr 17 16:27 fast.log
-rw------- 1 pi pi 0 Apr 17 16:27 stats.log
-rw------- 1 pi pi 0 Apr 17 16:27 suricata.log
pi@raspberrypi:/var/log/suricata $ sudo logrotate -f /etc/logrotate.d/suricata
pi@raspberrypi:/var/log/suricata $ ls -l
total 0
-rw-r--r-- 1 pi pi 0 Apr 17 16:38 eve.json
-rw-r--r-- 1 pi pi 0 Apr 17 16:27 eve.json-17042020-16:38
-rw------- 1 pi pi 0 Apr 17 16:38 fast.log
-rw------- 1 pi pi 0 Apr 17 16:27 fast.log-17042020-16:38
-rw------- 1 pi pi 0 Apr 17 16:38 stats.log
-rw------- 1 pi pi 0 Apr 17 16:27 stats.log-17042020-16:38
-rw------- 1 pi pi 0 Apr 17 16:38 suricata.log
-rw------- 1 pi pi 0 Apr 17 16:27 suricata.log-17042020-16:38
pi@raspberrypi:/var/log/suricata $
```

Ilustración 85 - Rotado de logs
Elaboración propia

Y comprobamos el rotado con logrotate -d /etc/logrotate.d/suricata

```
considering log /var/log/suricata/eve.json
  Now: 2020-04-17 16:39
  Last rotated at 2020-04-17 16:38
  log does not need rotating (log has been already rotated)
not running postrotate script, since no logs were rotated
```

Ilustración 86 - Comprobación
Elaboración propia

Teniendo esto funcionando, sólo quedará transferir todos los archivos eve.json hacia la máquina PILA-ELK.

Para recoger los .json, crearemos un archivo con las instrucciones a dar al cliente SFTP:

De tal forma que:

- Cambia el directorio local a /home/pilaelk/transferencias, para dejar ahí temporalmente los archivos .json.
- Cambia el directorio remoto a /var/log/suricata, donde se encuentran los archivos .json a copiar.
- Transfiere los archivos cuyo nombre comience por eve.json-
- Elimina los archivos transferidos.
- Termina la conexión.

```
lcd /home/pilaelk/elk/transferencias
cd /var/log/suricata
mget eve.json-*
rm eve.json-*
quit
```

Ilustración 87 - Script sftp

Elaboración propia

Para mayor seguridad, implementaremos certificados RSA para las conexiones SSH y SFTP, además de que así no tendremos que introducir la contraseña cada vez.

Para ello, deberemos generar certificados en la Raspberry:

```
pi@raspberrypi:/var/log $ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pi/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pi/.ssh/id_rsa.
Your public key has been saved in /home/pi/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:z3mUly/og1lr4cAAhg2lvlM+2G5byJ768F0qki4lLr0 pi@raspberrypi
The key's randomart image is:
+---[RSA 2048]----+
|     .=.         |
|     ..+         |
|     .. .        |
|   .    .    . . |
|    . . So  o o  |
|  . .B . ooooo . |
| o o=.* . =*oo. .|
|. + o*o= ooo=  . |
| .E+o=Boo  ...   |
+-----[SHA256]-----+
pi@raspberrypi:/var/log $
```

Ilustración 88 - Generar rsa

Elaboración propia

En nuestra máquina local, crearemos el archivo authorized_keys y le daremos permisos 700.

```
pilaelk@elk-mint:~$ cd .ssh
pilaelk@elk-mint:~/.ssh$ touch authorized_keys
pilaelk@elk-mint:~/.ssh$ chmod 700 authorized_keys
pilaelk@elk-mint:~/.ssh$
```

Ilustración 89 - authorized keys

Elaboración propia

Ahora, con la utilidad ssh-copy-id, ejecutaremos:

```
pilaelk@elk-mint:~/.ssh$ ssh-copy-id -i id_rsa.pub pi@192.168.1.10
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
pi@192.168.1.10's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'pi@192.168.1.10'"
and check to make sure that only the key(s) you wanted were added.

pilaelk@elk-mint:~/.ssh$ ssh pi@192.168.1.10
Linux raspberrypi 4.19.97-v7+ #1294 SMP Thu Jan 30 13:15:58 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Apr 17 16:01:38 2020 from 192.168.1.69
pi@raspberrypi:~ $
```

Ilustración 90 - conexión ssh
Elaboración propia

Nos pedirá una vez la contraseña del usuario de la Raspberry y ya podremos iniciar sesión por SSH o SFTP hacia la Raspberry con nuestro certificado.

Se crea el script de copia:

```
pilaelk@elk-mint:~/elk$ nano sftp.sh
#!/bin/bash

echo "Empezando proceso de transferencia SFTP..."
sftp -b /home/pilaelk/elk/sftp.txt pi@192.168.1.10
cp /home/pilaelk/elk/transferencias/* /home/pilaelk/elk/json/
echo "Archivos copiados hacia json."
cp /home/pilaelk/elk/transferencias/* /home/pilaelk/elk/historico/
echo "Archivos copiados hacia historico."
rm /home/pilaelk/elk/transferencias/*
echo "Archivos eliminados de transferencias."
```

Ilustración 91 - Script de copia
Elaboración propia

Comprobaremos su funcionamiento, no sin antes dar permisos de ejecución al script:

```
pilaelk@elk-mint:~/elk$ chmod u+x sftp.sh
pilaelk@elk-mint:~/elk$ ls -l sftp.*
-rwxrw-r-x 1 pilaelk pilaelk 386 abr 17 17:31 sftp.sh
-rw-rw-r-- 1 pilaelk pilaelk  90 abr 17 17:28 sftp.txt
pilaelk@elk-mint:~/elk$
```

```
pilaelk@elk-mint:~/elk$ ls historico/
pilaelk@elk-mint:~/elk$ ./sftp.sh
Empezando proceso de transferencia SFTP...
sftp> lcd /home/pilaelk/elk/transferencias
sftp> cd /var/log/suricata
sftp> mget eve.json-*
sftp> rm eve.json-*
sftp> quit
Archivos copiados hacia json.
Archivos copiados hacia historico.
Archivos eliminados de transferencias.
pilaelk@elk-mint:~/elk$ ls historico/
eve.json-17042020-16:38
pilaelk@elk-mint:~/elk$
```

Ilustración 92- Funcionamiento script

Elaboración propia

Efectivamente,se puede comprobar que los archivos son copiados hacia las carpetas historico y json, para guardar todos los archivos .json y para importarlos a su vez para su lectura en Logstash.

Crearemos dos entradas de crontab para que se ejecute este script en cada reinicio y una vez al día. Se decide que se ejecute a la 1 de la mañana para darle un margen al servicio Logrotate de la Raspberry. Pondremos un sleep 120 para que pueda iniciar la red correctamente.

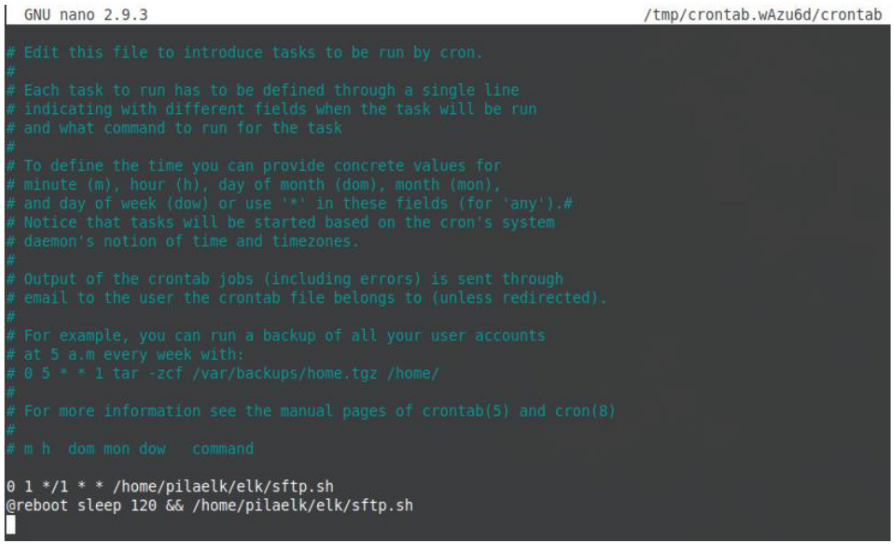

Ilustración 93 - Entradas en el Cron
Elaboración propia

Cambiaremos finalmente la configuración de Logstash para que recoja todos los logs que empiecen por eve.json-:

*eve.conf ×

```
input {
    file {
        path => ["/home/pilaelk/elk/json/eve.json-*"]
        codec =>    json
        mode => ["read"]
        type => "SuricataIDPS"
        start_position =>   "beginning"
    }
}

output {
    stdout  {   codec =>  rubydebug   }
    elasticsearch {
        hosts =>    ["localhost:9200"]
        index =>    "suricata-%{+YYYY.MM.dd}"
        document_type =>    "suricata"
    }
}
```

Ilustración 94 - Nuevo Logstash
Elaboración propia

3.12. **Pila ELK como servicios.**

Para automatizar el inicio de la pila ELK, usaremos systemctl.
Se crearan tres archivos el ordenador de la pila ELK:

/lib/systemd/system/elasticsearch.service

```
[Unit]
Description=ElasticSearch
After=Network.target

[Service]
User=pilaelk
ExecStart=/home/pilaelk/elk/elasticsearch-7.6.2/bin/elasticsearch

[Install]
WantedBy=multi-user.target
```

Ilustración 95 - Service Elasticsearch
Elaboración propia

Se indica la descripción del servicio, que se inicie después del servicio de red, se ejecute con el usuario pilaelk y la ruta del programa.

/lib/systemd/system/kibana.service

```
[Unit]
Description=Kibana
After=elasticsearch.service

[Service]
User=pilaelk
ExecStart=/home/pilaelk/elk/kibana-7.6.2-linux-x86_64/bin/kibana

[Install]
WantedBy=multi-user.target
```

Ilustración 96 - Servicio Kibana
Elaboración propia

Igual que el anterior, pero se ejecuta después del inicio del servicio Elasticsearch.

/lib/systemd/system/logstash.service

```
[Unit]
Description=Logstash
After=kibana.service

[Service]
User=pilaelk
ExecStart=/home/pilaelk/elk/logstash-7.6.2/bin/logstash -f /home/pilaelk/elk/logstash-7.6.2/config/eve.conf

[Install]
WantedBy=multi-user.target
```

Ilustración 97 - Servicio de logstash
Elaboración propia

Igual que el anterior, pero su inicio será después del de Kibana, además de indicarle el archivo eve.conf.
Ejecutaremos una recarga de los scripts de systemctl:

```
sudo  systemctl daemon-reload
```

Y se habilitaran los servicios para que se ejecuten al inicio:

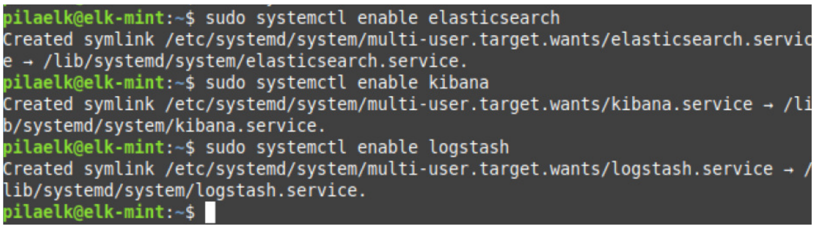

Ilustración 98 - Inicio Pila servicios
Elaboración propia

Reiniciaremos el sistema para comprobar si los servicios se inician correctamente.

Ilustración 99 - Cerrar sesión
Elaboración propia

Y se puede comprobar que están todos iniciados:

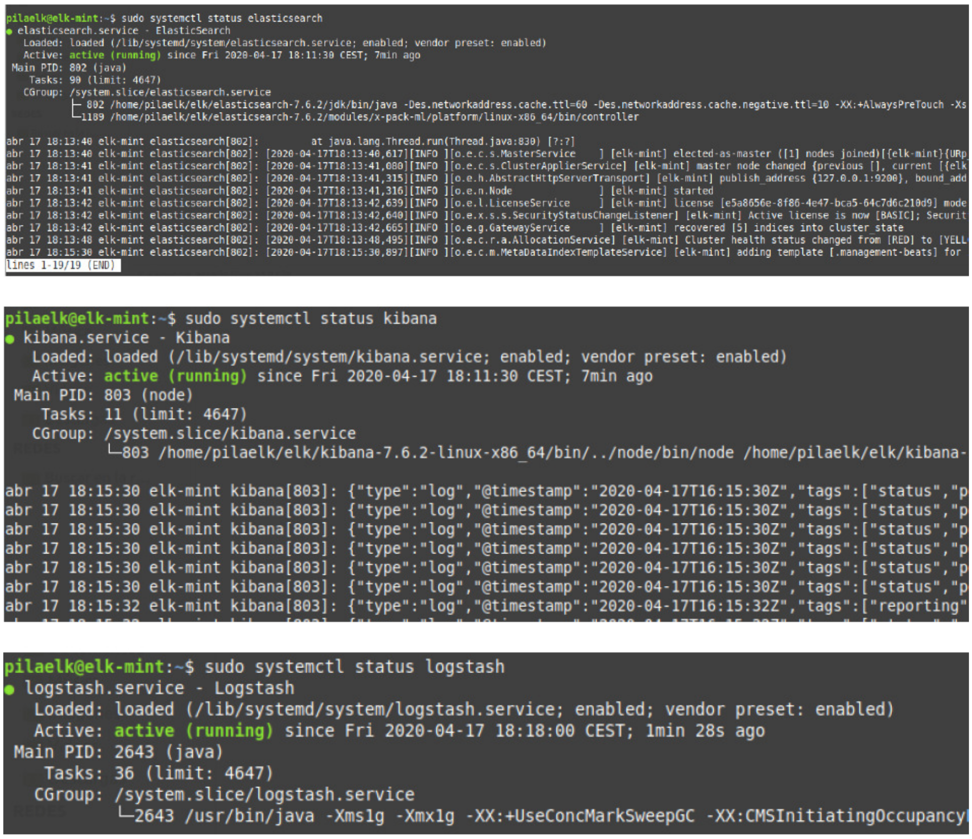

Ilustración 100 - Inicio servicios correcto
Elaboración propia

Como se puede observar, los servicios funcionan correctamente al iniciar la máquina con la Pila.ELK.

Además, automáticamente se están importando los datos rotados de la Raspberry, que ha copiado nuestro script a la carpeta json:

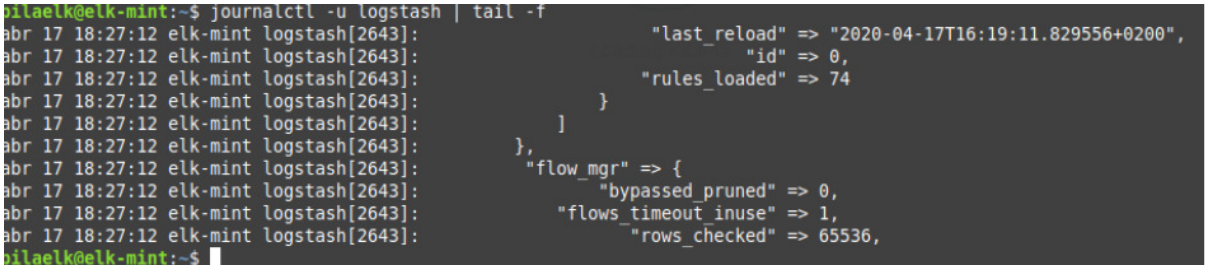

Ilustración 101 - rotación de logs
Elaboración propia

Aquí pueden comprobarse en Kibana:

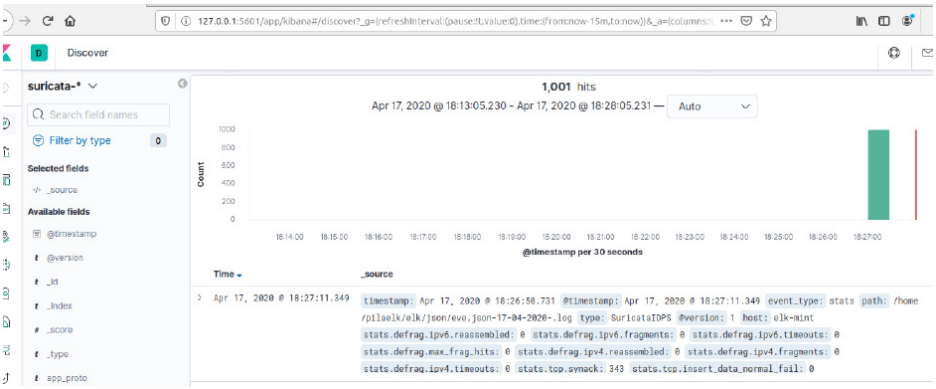

Ilustración 102 - Comprobación Kibana
Elaboración propia

Queda de una forma adecuada la automatización y el proceso de copia de datos e inicio de los servicios.

3.13. **Estudios diferencial y curiosidades del proyecto**

En esta fase se va desarrollar una explicación de las cosas más curiosas del proyecto y que le pueden un valor diferencial, se mostraran ataques reales sobre el punto wifi y que seremos capaces de analizar con nuestro proyecto. Al final del esta fase veremos unas conclusiones y unas líneas de futuro y ampliación de este trabajo.

3.13.1. **Uso de Logrotate**

Logrotate es una utilidad de Linux que se ejecuta automáticamente como servicio, y que nos permite rotar logs, es decir, mover en este caso el contenido de eve.json hacia otro archivo eve.json-fecha, vaciando el archivo original. De esta forma, cada día o cada vez que forcemos el rotado del log, se obtendrá un archivo nuevo que contendrá el texto de log que moveremos a nuestra máquina .externa mediante SFTP.
Por defecto, se crea automáticamente el archivo de esta forma:

```
 GNU nano 3.2                  /etc/logrotate.d/suricata

/var/log/suricata/*.log
/var/log/suricata/*.json
{
        rotate 14
        missingok
        compress
        copytruncate
        sharedscripts
        postrotate
                /bin/kill -HUP $(cat /var/run/suricata.pid)
        endscript
}
```

Ilustración 103 - conf.Basica logrotate
Elaboración propia

Vamos realizar otra configuración para logrotate. Se dejará de la siguiente forma:

```
 GNU nano 3.2                                        /etc/logrotate.d/suricata

/var/log/suricata/*.log
/var/log/suricata/*.json
{
        daily
        rotate 15
        missingok
        nocompress
        dateext
        dateformat -%d-%m-%Y-.log
        create 666 pi pi
        sharedscripts
        postrotate
                /bin/kill -HUP $(cat /var/run/suricata.pid)
        endscript
}
```

Ilustración 104 - conf.Basica Propia
Elaboración propia

Lo que se realiza es lo siguiente:

- Las dos rutas de las dos primeras líneas indican que se debe trabajar sobre los archivos .json y .log de la carpeta /var/log/suricata/.
- Después, entre llaves, indicaremos lo siguiente:
  - El rotado de los logs se hará automáticamente de forma diaria
  - Se deberán conservar los 15 últimos logs, aunque estén vacíos.
  - No se realizará ningún tipo de compresión del archivo.
  - Se indica el formato de fecha día-mes-año-hora-minuto, (%d es el día, %m mes, %Y el año).
- Se indica al programa que los logs rotados sean propiedad del usuario y grupo pi
- Y, finalmente, que después de realizar la rotación se reinicie el proceso de Suricata.

3.13.2. Implementación de un ataque básico

Se ha creado un archivo (/etc/suricata/rules/test.rules) donde se pueden establecer una regla que indica que al realizar una conexión ICMP (ping) desde dentro de la red ($HOME_NET) hacia fuera ($EXTERNAL_NET).

Lo podemos ver aquí:

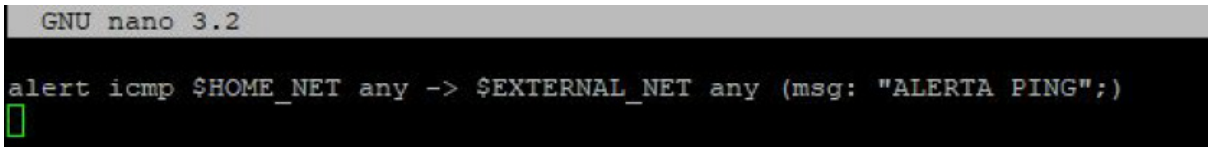

Ilustración 105 - Regla icmp
Elaboración propia

Para que el archivo de reglas que se acaba de crear sea utilizado, se debe editar el archivo de configuración de Suricata (suricata.yaml):

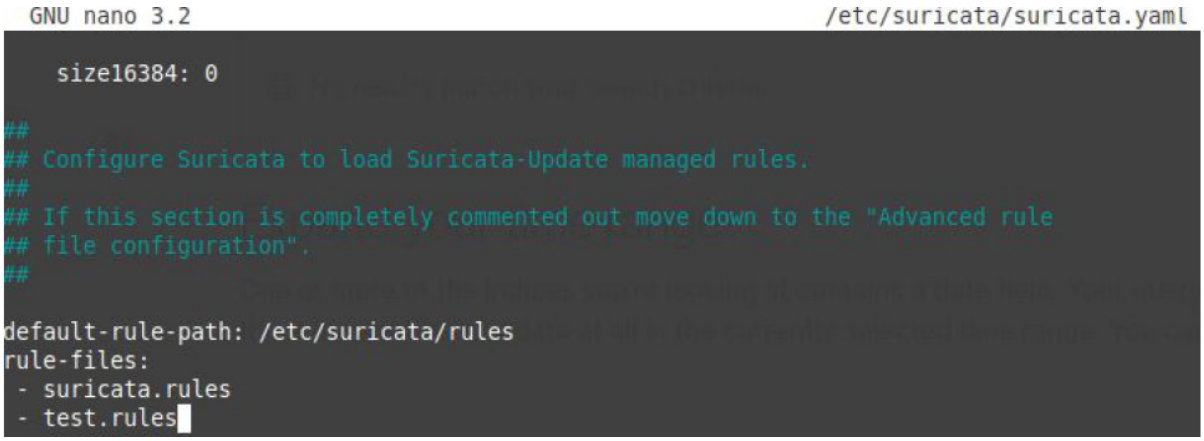

Ilustración 106 - Regla en suricata.yaml
Elaboración propia

Es necesario reiniciar el servicio para que pueda volver a cagar las nuevas reglas:

```
sudo systemctl restart suricata
```

Iniciaremos un ping hacia 1.0.0.1 (IP de CloudFlare, fuera de la red), conectados a nuestra red WiFi:

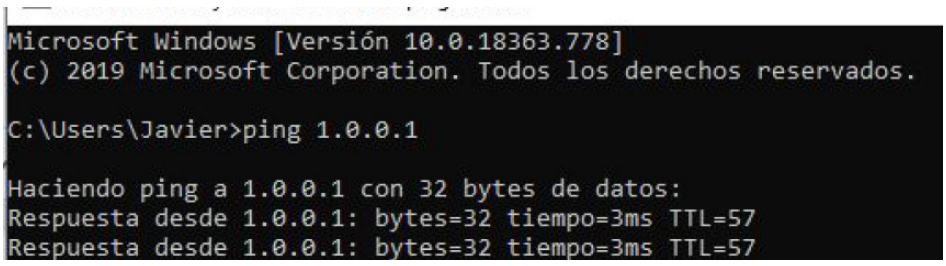

Ilustración 107-ping a cloudflare
Elaboración propia

Es necesario volver a rotar los logs:

```
pi@raspberrypi:/var/log/suricata $ sudo logrotate -f /etc/logrotate.d/suricata
pi@raspberrypi:/var/log/suricata $ ls
eve.json                fast.log                stats.log                suricata.log
eve.json-17-04-2020-.log  fast.log-17-04-2020-.log  stats.log-17-04-2020-.log  suricata.log-17-04-2020-.log
pi@raspberrypi:/var/log/suricata $
```

Ilustración 108 - Rotar suricata
Elaboración propia

Y se puede ejecutar el script manualmente para hacer la prueba de concepto:

```
pilaelk@elk-mint:~$ elk/sftp.sh
Empezando proceso de transferencia SFTP...
sftp> lcd /home/pilaelk/elk/transferencias
sftp> cd /var/log/suricata
sftp> mget eve.json-*
sftp> rm eve.json-*
sftp> quit
Archivos copiados hacia json.
Archivos copiados hacia historico.
Archivos eliminados de transferencias.
pilaelk@elk-mint:~$
```

Ilustración 109 - copia de logs
Elaboración propia

Podemos comprobar el fichero copiado a json que vamos a poder importar:

Ilustración 110 - eve.json
Elaboración propia

Y aquí podemos comprobar que se está importando el archivo en el log de Logstash

```
pilaelk@elk-mint:~$ journalctl -u logstash | tail -20
abr 17 18:53:35 elk-mint logstash[2643]:         "bytes_toclient" => 0,
abr 17 18:53:35 elk-mint logstash[2643]:         "bytes_toserver" => 54,
abr 17 18:53:35 elk-mint logstash[2643]:                    "end" => "2020-04-17
T18:34:00.234883+0200",
abr 17 18:53:35 elk-mint logstash[2643]:                "alerted" => false,
abr 17 18:53:35 elk-mint logstash[2643]:                    "age" => 0,
abr 17 18:53:35 elk-mint logstash[2643]:          "pkts_toserver" => 1,
abr 17 18:53:35 elk-mint logstash[2643]:                  "start" => "2020-04-17
T18:34:00.234883+0200"
abr 17 18:53:35 elk-mint logstash[2643]:     },
abr 17 18:53:35 elk-mint logstash[2643]:       "src_port" => 57941,
abr 17 18:53:35 elk-mint logstash[2643]:          "proto" => "TCP",
abr 17 18:53:35 elk-mint logstash[2643]:     "@timestamp" => 2020-04-17T16:53:34
.266Z,
abr 17 18:53:35 elk-mint logstash[2643]:       "@version" => "1",
```

Ilustración 111 - log de Logstash
Elaboración propia

Después de que se importen los logs, se filtrará en Kibana por dos etiquetas, como se explicó anteriormente:

- event_type: alert -> Se indica que sólo se muestren las alertas.
- src_ip: 10.1.1.15 -> IP de nuestro ordenador conectado a la red WIFI

Se puede observar que aparecen estas alertas:

Ilustración 112 - POC en Kibana
Elaboración propia

Se podría filtrar también por el mensaje que hemos puesto: “ALERTA PING”, y aparecerán todas las alertas que se realicen de esa regla de test.rules en nuestra red local.
Esto sería una simulación de ataque. Podríamos implementar varias reglas en un mismo archivo, como por ejemplo que fuese al revés, desde la red externa hacia la interna, y en los diferentes protocolos (tcp, udp, telnet...).
Por ejemplo, si disponemos de un servidor en nuestra red local que no debería ser accedido desde la red externa, configuraríamos alertas TCP, UDP, DNS, ICMP, etcétera, para que nos saltara una alerta si se realizara alguna conexión desde fuera de la red local.

3.13.3. **Simulación real de un ataque**

Se va realizar una simulación más real de un ataque, realizando una conexión SSH hacia un servidor que solo debería ser accedido dentro de nuestra red local. Se configurará una alerta para que nos avise si eso ocurre, ya que el servidor no deberá ser accedido mediante la red externa, con lo cual, existirá un agujero de seguridad en nuestra red.
Lo primero será configurar una regla en nuestro IDS Suricata, para que avise de cualquier conexión SSH desde cualquier host de la red externa hacia el servidor 10.1.1.50 (PC con Ubuntu que simulará ser un servidor, conectado por Wi-Fi a nuestra red), que nunca debería ser accedido mediante Internet.
Se usará el mismo archivo test.rules donde agregaremos la nueva regla:

```
 GNU nano 3.2                                  /etc/suricata/rules/test.rules

alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg: "ALERTA PING";)

alert tcp $EXTERNAL_NET any -> 10.1.1.50 22 (msg: "ALERTA SSH HACIA UBUNTU";)
```

Ilustración 113 - test.rules
Elaboración propia

Quedaría así, lo que indica:

- Marcar como alerta.
- El protocolo será TCP.
- Proviene desde la red externa.
- Proviene desde cualquier puerto.
- El destino de la conexión será el host 10.1.1.50.
- Hacia el puerto 22 (SSH).
- El mensaje (alert.signature) será ALERTA SSH HACIA UBUNTU.

Se debe reiniciar Suricata para que aplique las nuevas reglas.

Procederemos ahora a realizar el ataque. Simularemos que la red externa es la red LAN 192.168.1.0/24

Se debe añadir una ruta estática en nuestro Equipo, ya que sólo tiene acceso a la red 192.168.1.0/24. Y se indica el siguiente salto para acceder a la red 10.1.1.0/24 es la Raspberry Pi como sonda wifi(192.168.1.10).

```
pilaelk@elk-mint:~$ sudo ip route add 10.1.1.0/24 via 192.168.1.10
pilaelk@elk-mint:~$ ping 10.1.1.1
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=0.806 ms
64 bytes from 10.1.1.1: icmp_seq=2 ttl=64 time=0.712 ms
64 bytes from 10.1.1.1: icmp_seq=3 ttl=64 time=0.813 ms
^C
--- 10.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.712/0.777/0.813/0.046 ms
```

Ilustración 114 - Cambio de ruta
Elaboración propia

También se comprueba el acceso a 10.1.1.50, nuestro equipo Ubuntu que hace de Servidor de pruebas.

```
pilaelk@elk-mint:~$ ping 10.1.1.50
PING 10.1.1.50 (10.1.1.50) 56(84) bytes of data.
64 bytes from 10.1.1.50: icmp_seq=1 ttl=63 time=120 ms
64 bytes from 10.1.1.50: icmp_seq=2 ttl=63 time=147 ms
```

Ilustración 115 ping de prueba
Elaboración propia

Se realiza ahora la conexión SSH:

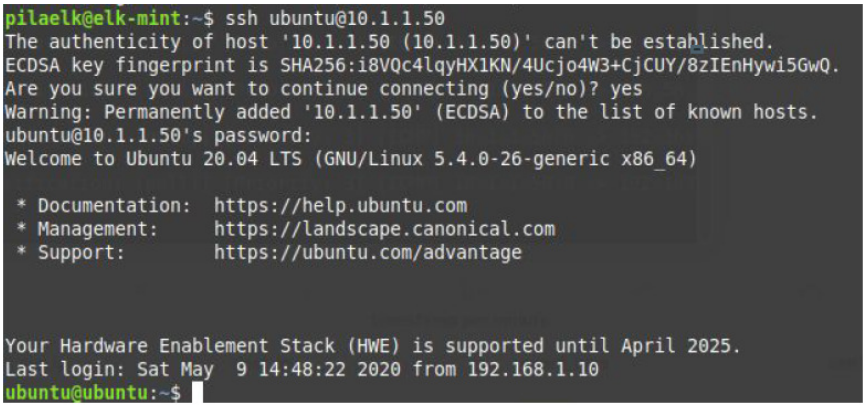

Ilustración 116 – ssh
Elaboración propia

Cerraremos ahora la conexión y procederemos a la importación forzada de los datos hacia ELK.

Forzamos el rotado de los logs en la Raspberry:

```
sudo lograte – f /etc/logratote.d/suricata
```

A continuación de manera manual se ejucatará el script sftp.sh de ELK-Mint:

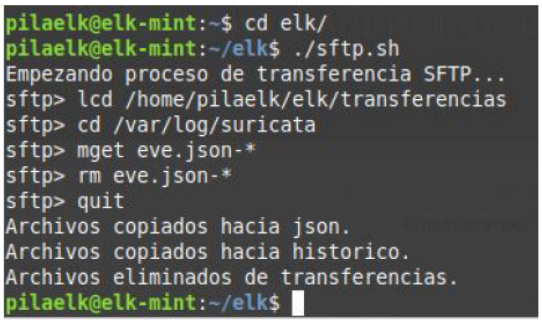

Ilustración 117 - Transferencia sftp
Elaboración propia

Después se ejecuta Logstash importando el archivo:

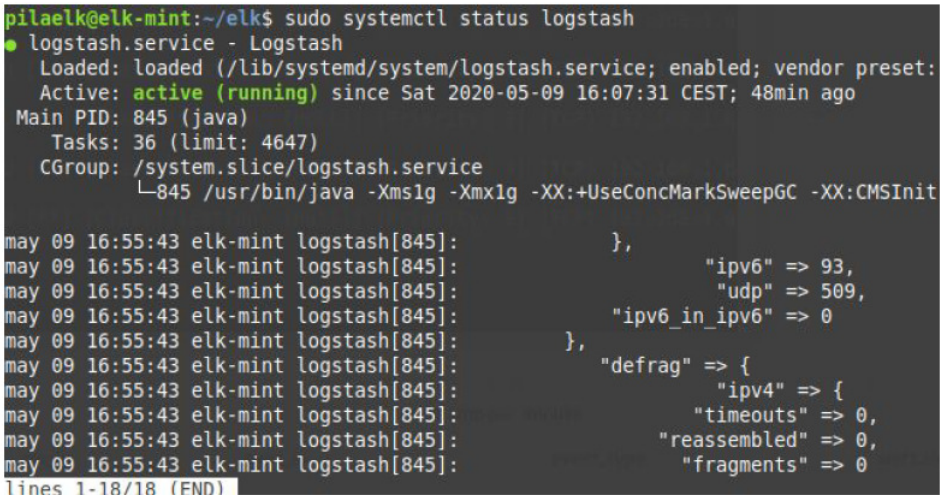

Ilustración 118 Prueba logstash
Elaboración propia

Y en Kibana se creará un filtro por el campo “alert.signature” de esta manera:

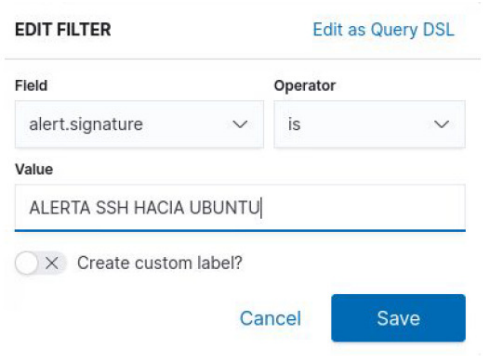

Ilustración 119 - Filtro Kibana
Elaboración propia

Aparecerán las conexiones SSH realizadas desde la red externa hacia el Servidor Ubuntu de pruebas, con su fecha, hora, IP de origen e IP de destino:

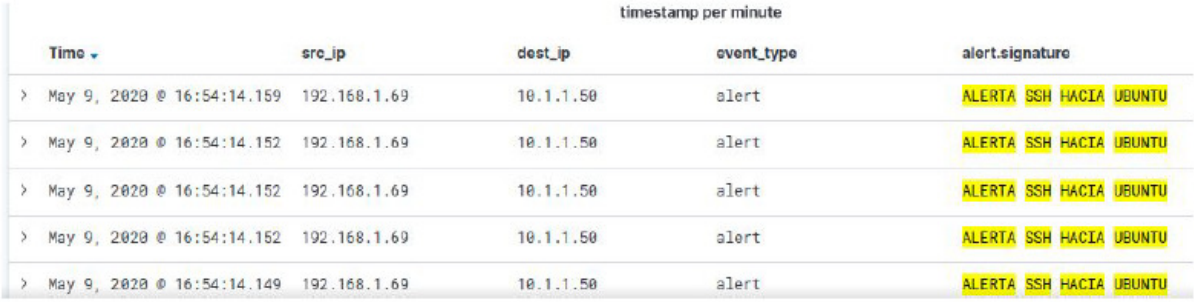

Ilustración 120 - Comprobación del ataque
Elaboración propia

3.13.4. Alternativa de IDS con Snort vs Suricata

Por realizar en el proyecto algo donde podamos ver diferencias, se ha realizado el mismo análisis usando la herramienta Snort en lugar de Suricata. Se han realizado las pruebas en una maquina virtual independiente, para no interferir en el proyecto.

1. ¿Qué es Snort?

Snort es un sistema de Detección de intrusos (IDS) basado en red (IDSN) open source. Nos permite mediante un lenguaje de creación de reglas establecer los patrones que se emplearán a la hora de monitorizar nuestro sistema. Ofrece listas con reglas ya predefinidas que podemos seleccionar y adaptar a nuestras necesidades.

Snort puede funcionar como sniffer, pudiendo ver a tiempo real el tráfico que pasa por nuestra red. Además, nos permite guardar las alertas en un archivo los logs para su posterior análisis (todo lo dicho anteriormente los veremos más detenidamente a continuación).

2. Instalación de Snort

Para instalar Snort lo más aconsejable es irse a la página oficial www.snort.org donde tenemos a nuestra disposición los archivos de instalación para cualquier sistema operativo que tengamos. En mi caso particular he utilizado Ubuntu.

Para un sistema tipo debian podemos introducir el comando "*sudo apt-get install snort*" y automáticamente se nos abrirá un menú de configuración:

-Introduciremos nuestra ip

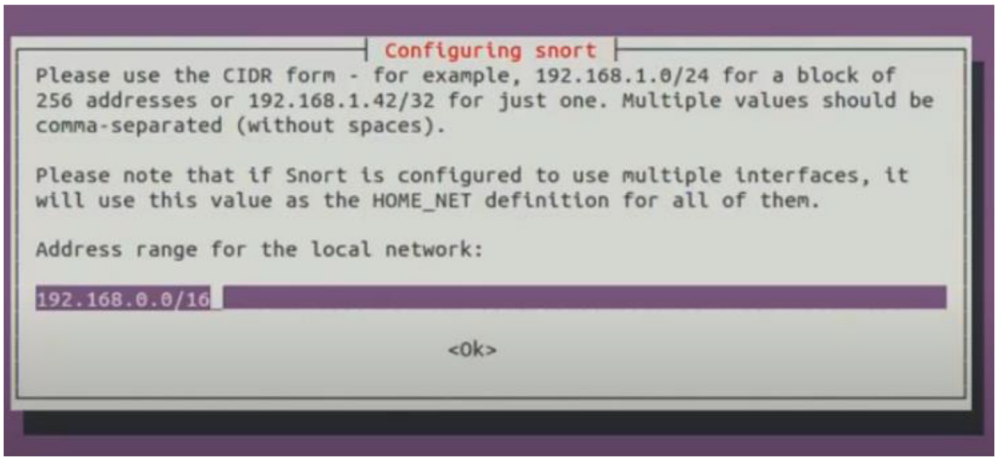

Ilustración 121 - Instalación Snort
Elaboración propia

Y ya se ha instalado snort, es un proceso rápido y sencillo.

Una vez instalado podemos comprobar si el servicio se está ejecutando de manera correcta introduciendo la orden "*sudo service snort status*":

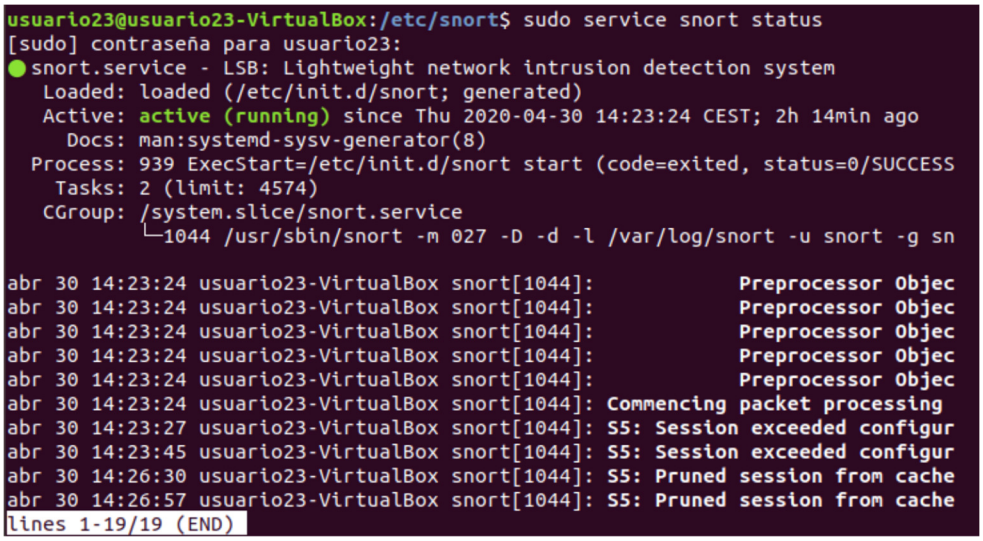

Ilustración 122 - Servicio Snort
Elaboración propia

Se puede ver la versión que se nos ha instalado en nuestro sistema con el comando "*snort -V*":

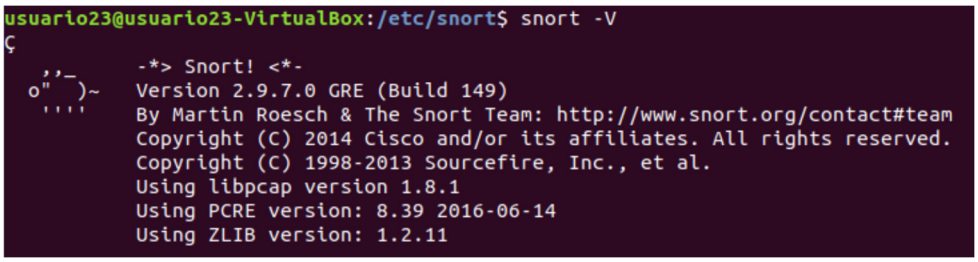

Ilustración 123 - Versión snort
Elaboración propia

3. Creación de nuestra primera regla

Como ya se ha explicado snort emplea reglas para alertar de aquel tráfico que cumpla con el patrón de dichas reglas. A continuación, se verá cómo crear nuestras primeras reglas básicas en nuestro IDS snort.

El primer paso es crear un archivo donde definiremos nuestras reglas, en este ejemplo será el fichero llamado proyecto.rules, este estará alojado en la carpeta /etc/snort/rules/:

```
usuario23@usuario23-VirtualBox:~$ sudo touch /etc/snort/rules/proyecto.rules
```

Una vez creado el fichero debemos conocer la sintaxis de las reglas, las reglas de Snort se dividen en dos partes: cabecera y opciones.

En la cabecera se indica la acción asociada a una regla, el tipo de paquete, las direcciones origen y destino, así como los puertos utilizados.

La estructura de la cabecera es la siguiente:

<acción> <protocolo> <red origen> <puerto origen> <dirección> <red destino><puerto destino>

Se pueden utilizar cinco tipos de acciones:

- Log: se genera solo la información de registro asociada al contenido del paquete.
- Alert: se genera una alerta y un log del paquete que activa la regla.
- Pass: deja pasar el paquete asociado a la regla si registrar un log.
- Activate: genera una alerta y la activación de una regla dinámica (dynamic)
- Dynamic: es una regla que está inactiva hasta que se activa con un activate. Tiene como objetivo ofrecer información adicional a la que se obtendría con la regla inicial a la que está asociada.

En las opciones, se especifica qué información se debe encontrar en un paquete para que se active la acción asociada a la regla. Se escriben todas dentro de un paréntesis y puede haber una o más opciones separadas por un';'

Hay cuatro categorías para estas opciones:

- Generales: msg, reference, gid, sid, classtype, rev, etc... Dan información sobre la regla, pero no afectan a la detección.
- Detección de contenido: content, nocase, rawbytes, depth, offset, distance, uircontent, etc... Buscan patrones dentro de la carga útil del paquete.
- Detección non-payload: fragoffset, ttl, tos, id, ipopts, fragbits, etc... Busca patrones en otros campos del paquete que no sean la carga útil.

- Postdetección: logto, session, resp.

Una vez que sabemos sintácticamente como trabajan las reglas vamos a crear una sencilla, por ejemplo, que nos avise si se ha realizado un ping desde cualquier ip origen y puerto origen a cualquier ip destino y puerto de destino, que nos envíe un mensaje que diga “se ha realizado un ping” y que el número de sid sea el1000001:

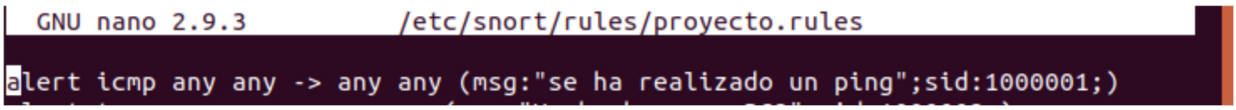

Ilustración 124 - Regla Snort
Elaboración propia

El último paso es incluir nuestro fichero en /etc/snort/snort.conf, ya que si no lo hacemos snort no tendrá nuestras reglas en cuenta a la hora de monitorizar:

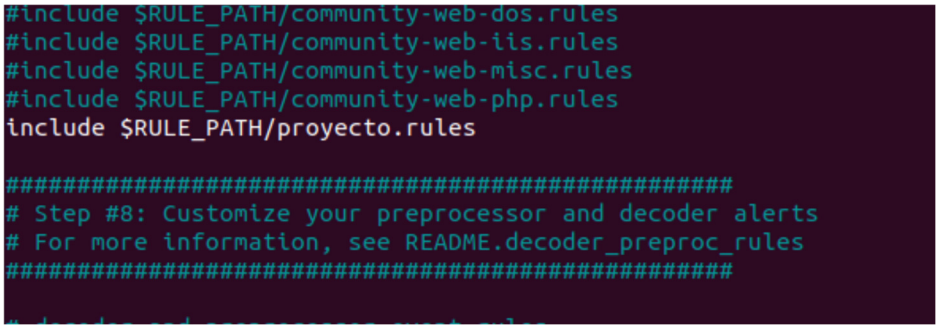

Ilustración 125 - snort.conf
Elaboración propia

Para que nos salga por consola introduciremos el comando :

sudo snort -q -v -A console --daq-dir /usr/local/lib/daq -c /etc/snort/snort.conf -i enp0s3.

La opción -i hace referencia a nuestra interfaz física, en mi caso es enp0s3.

Vamos a forzar a que nos salga el aviso haciendo ping a la dirección 8.8.8.8 por ejemplo:

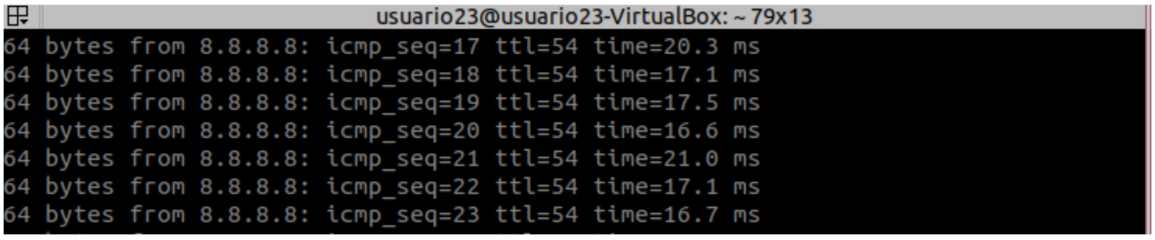

Ilustración 126 - Prueba1 Snort
Elaboración propia

Y como podemos observar nuestro snort nos ha avisado de ello:

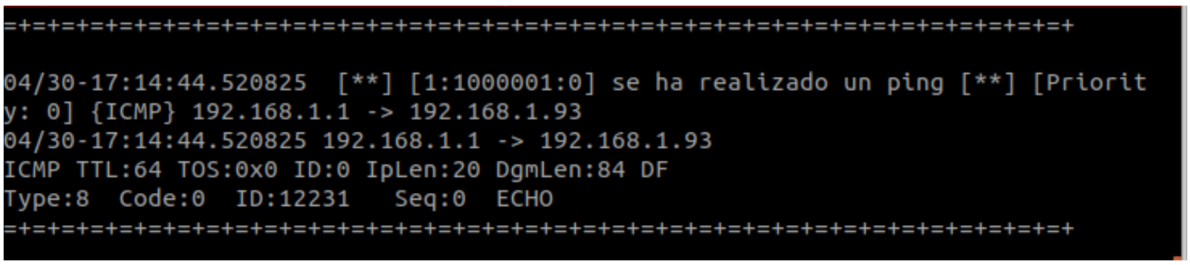

Ilustración 127 - Prueba2 Snort
Elaboración propia

4. Más ejemplos prácticos

A continuación, veremos algunas normas que nos pueden ser útiles.
Si queremos controlar los paquetes udp podemos establecer esta norma:

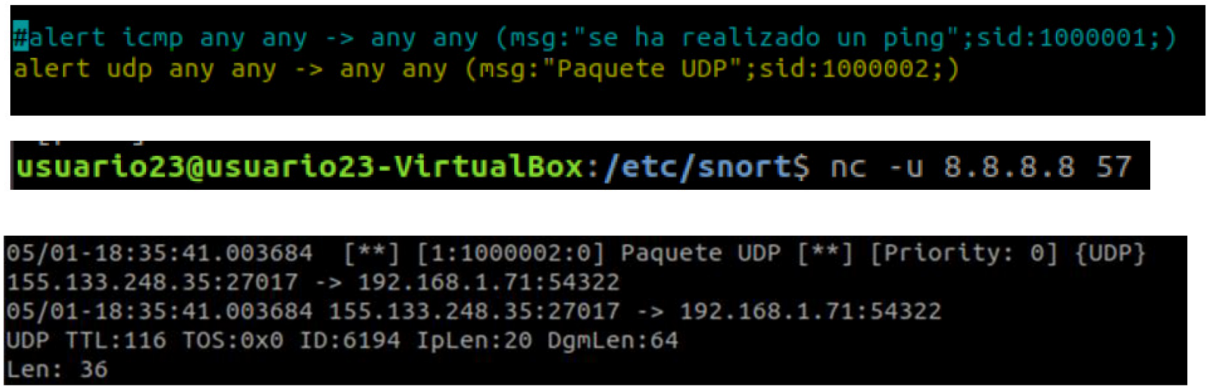

Ilustración 128 - Regla UDP
Elaboración propia

Si queremos detectar cuándo se hace peticiones con la palabra “youtube” por ejemplo, podemos emplear esta regla:
Si nos interesan sólo las peticiones GET, podemos poner varios content:

alert tcp any any -> any any (msg:” YouTube"; content: "GET "; content: "youtube"; sid:1000003;)

Estas dos últimas reglas son muy fáciles de burlar ya que podemos expresar la misma dirección en notación porcentaje como por ejemplo “y%utube”, para que esto no ocurra es mucho más eficaz que busque en la URL del paquete HTTP con el modificador “http_uri”:

alert tcp any any -> any any (msg:"Youtube"; content: "GET “; content: "youtube"; http_uri; sid:1000012;)

Podemos añadir que el evento asociado a una regla sólo se produzca si esta se produce un número de veces:

alert ... (..., detection_filter: track by_dst, count 5, seconds 60;)

Aunque hacer tus propias reglas es bastante enriquecedor a la hora de entender su sintaxis, la realidad es que ya existen archivos muy completos con reglas que vienen por defecto cuando instalamos snort, se encuentran en la carpeta /etc/snort/rules:

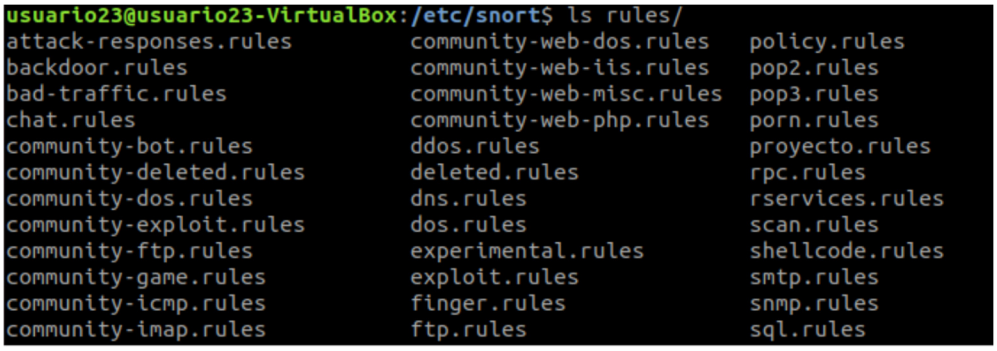

Ilustración 129 /etc/snort/rules
Elaboración propia

Si se quiere incluir algún archivo de reglas en tu ids Snort solo tienes que descomentarlo en el fichero snort.conf:

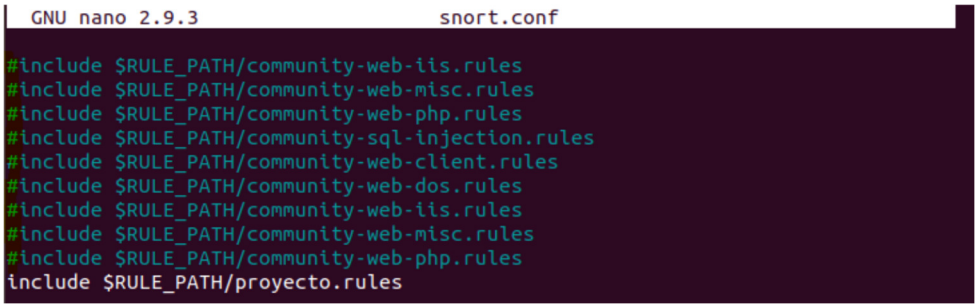

Ilustración 130 - Incluir reglas
Elaboración propia

Podemos descargar más reglas en esta dirección:

http://rules.emergingthreats.net/open/snort-2.9.0/emerging.rules.tar.gz

5. Registro de eventos textual

Si queremos que las alertas generadas por snort queden registradas en un fichero de texto para su posterior análisis debemos añadir la línea “output alert_fast: <nombre del fichero>” en snort.conf:

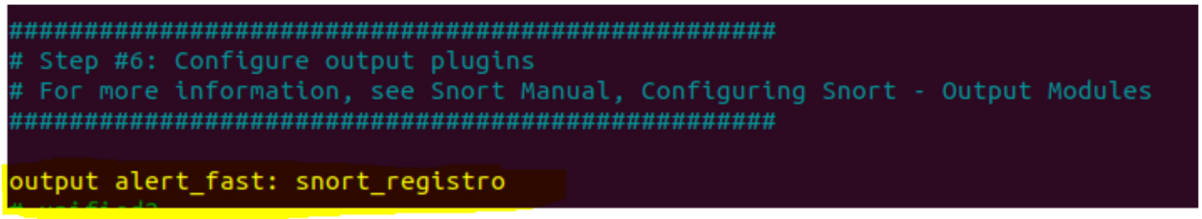

Ilustración 131 - output alert
Elaboración propia

Como se puede comprobar se ha creado nuestro fichero en la carpeta /var/log/snort:

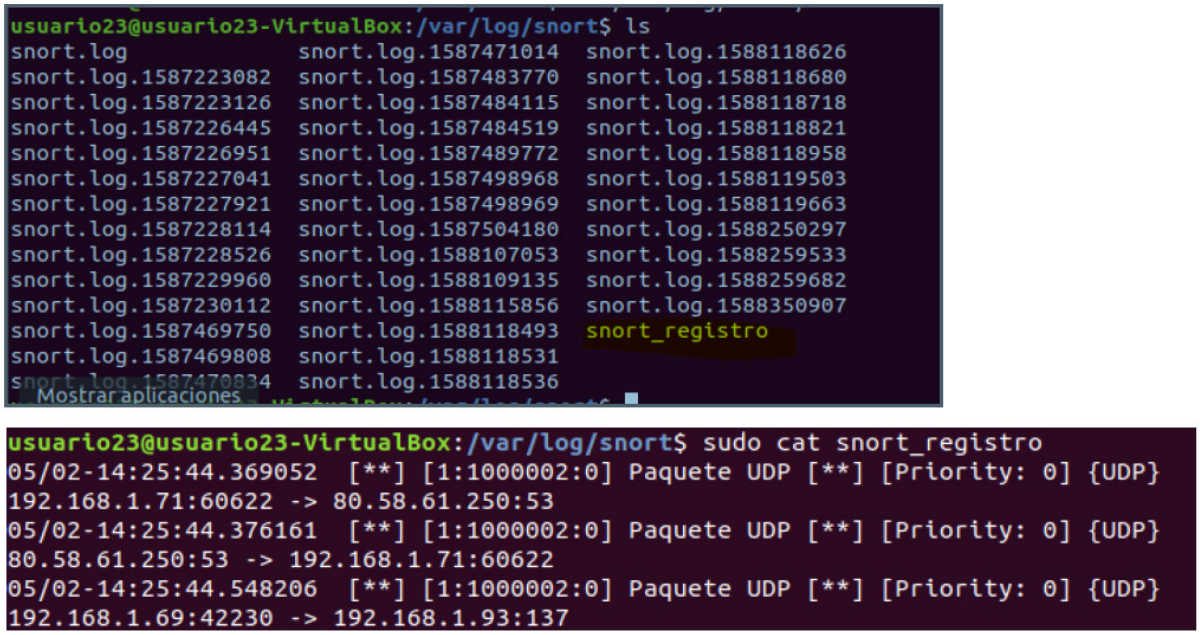

Ilustración 132 - Comprobación
Elaboración propia

Podemos cambiar el nombre del fichero a snort_fast, indicarle que guarde información del paquete, y limitar esa salida a 10k con:

```
output alert_fast: snort_registro packet 10k
```

En otro formato (Son dos plugin de salida bastante parecidos):

```
output alert_full: snort_registro 10k
```

6. Motivación. Unified2

Snort tiene un diseño antiguo (que no obsoleto), y su funcionamiento es fundamentalmente monohilo. El envío de alertas se realiza en el mismo hilo que el procesado de paquetes, por lo que todo el tiempo que pasemos enviando una alerta (y esperando la respuesta del receptor) es tiempo en el que estamos encolando paquetes y no estamos procesando, lo que puede provocar que tengamos que dejar de procesar algunos paquetes.

La situación se agrava si el receptor es otro programa que debe esperar que la CPU lo ejecute, es decir, comunicación entre procesos, como ocurre con la pantalla, syslog, una base de datos…

Para solucionarlo, snort encolará las alertas y los paquetes que la han provocado en un fichero binario, con un formato liviano, y otro programa (quizás, incluso, con menos prioridad de CPU) se encargará de transformar esas alertas en registros en una base de datos.

Este formato es unified2. Para configurarlo, añadimos en la configuración snort.conf, en la sección correspondiente (output plugins):

output unified2: filename merged.log, limit 128, mpls_event_types, vlan_event_types

4. Fase de Conclusiones

En esta fase veremos unas conclusiones y unas líneas de futuro y ampliación de este trabajo. Se buscaba obtener un estudio sobre los puntos wifi abiertos e intentar hacer ver que esta en peligro nuestra privacidad y datos personales.

4.1. **Conclusiones**

De este proyecto se pueden establecer una serie de conclusiones:

- Bajo mi opinión se llega a una conclusión clara, la pérdida de datos y la cantidad de información que podemos llegar a enviar a través de la red es casi infinita. Propagamos toda nuestra vida, intereses, datos personales e incluso bancarios a través de una red que realmente no sabemos si es segura o quien puede estar analizando dichos datos. Muchas veces la encriptación del tráfico no es una solución efectiva.
- Se necesita una mayor conciencia sobre los riesgos de la red. A nivel de usuario básico hay mucho por aprender y enseñar. Son un gran foco para el robo de información privada además de la total desinformación sobre este tema.
- Se han probado diferentes soluciones de sistemas de detección de intrusos IDS para un dispositivo IOT(Raspberry pi3). En este proyecto, se han estudiado algunas topologías y herramientas que pueden cumplir perfectamente con este propósito, pero existen alternativas que se pueden tener en cuenta dependiendo del caso particular.
- Se ha verificado que Suricata IDS y la PILA ELK son aplicaciones muy potentes y tienen la capacidad de configurarse de acuerdo con los objetivos prácticamente de cualquier proyecto de tipo IDS.
- Se ha conseguido el objetivo que era el análisis y obtención de datos, aunque no se ha conseguido la implementación de los programas iniciales como queríamos y se ha tenido que utilizar el material que se tenía en ese momento. El principal problema que se hatenido para no poder implementar dichos programas ha sido el sistema usado, en este caso Debian en la Raspberry pi 3 y la poca capacidad de no poder instalar la Pila ELK en el dispositivo IOT.
- Por otro lado, la Pila ELK pide muchos requisito para funcionar en la propia Raspberry pi3. Se decidió durante el proceso del proyecto por instalarla en otra maquina externa y usar un sistema de envío y recepción de ficheros de logs y datos.
- Hubiera sido una conclusión que todo el producto estuviera en el mismo dispositivo de IOT, pero con las herramientas elegidas no ha sido posible.
- Al final, todos los objetivos principales establecidos al comienzo del proyecto se han cumplido satisfactoriamente: investigar la última tecnología, elegir una combinación de herramientas y obtener un sistema completamente funcional usando un dispositivo IOT. También se logró un objetivo específico: estudiar topologías alternativas y explorar opciones para detectar, mostrar y notificar eventos de seguridad.

4.2. **Líneas de futuro**

Ayuda en la privacidad

Se podría hacer un estudio de forma pública revelando información que se ha obtenido sobre ciertos dispositivos y formas de proteger nuestro sistema. Sobre todo, intentar acercarnos hacia una concienciación más real sobre el tema:

- Claves públicas.
- Autenticación mutua fuerte.
- Claves secretas (secretos con alta entropía).
- Contraseñas (secretos con baja entropía).
- Otros criterios, como el reconocimiento de voz u otras características biométricas

Detectar intrusiones

Será interesante verificar la función de detección de intrusos de Suricata, se recomienda instalar el servicio Vulnerable al ataque y usar herramientas tipo metasploit para confirmar el posible ataque y se buscaría que el sistema informará de tales alertas.

Notificar alertas

La configuración básica de la PILA ELK ignora muchas notificaciones, aunque Individualmente no importa, por lo que sigue siendo divertido recibir notificaciones cuando sea posible dentro de un cierto rango, el número de este tipo de notificaciones ha aumentado en el tiempo, esto puede ser sospechoso y vale la pena investigarlo. Suricata y otras aplicaciones incluyen el tipo de reglas de adecuado que detectan tales anomalías.

Tiempo de Conexión

El sistema puede evolucionar para saber el tiempo de conexión de cada dispositivo conectado a mi sonda wifi. Se ha probado con ping, pero hubiera estado mejor probar otras opciones.

Mejorar la Presentación

Otra cosa que me veo como línea de futuro es la mejora de la presentación en Kibana, es un aplicación muy amplia y compleja y sería necesario más tiempo para crear un verdadero Dashboard y poder analizar datos de una forma adecuada.

5. Bibliografía

5.1. Libros Consultados

- Anderson, C. (24 de 4 de 2012). *The Man Who Makes the Future: Wired Icon Marc Andreessen.* Obtenido de Wired: http://www.wired.com/epicenter/2012/04/ff_andreessen/
- Manovich, L. (2011). *The Language of New Media.* Cambridge: MIT Press.
- Mathy Vanhoef, F. P. (2017). Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. 10.
- Ramos, A. A. (2013). *Hacking práctico de Redes Wifi y radiofrecuencia.* RA_MA.

5.2. Páginas web consultadas

Información sobre IDS

- Diseño y Configuración de IPS, IDS y SIEM en Sistemas de Control Industrial. INCIBE. URL:https://www.incibe-cert.es/sites/default/files/contenidos/guias/doc/certsi_diseno_configuracion_ips_ids_siem_en_sci.pdf
- Reglas Suricata. URL:https://fwhibbit.es/suricata-ids-jugando-con-las-reglas
- Información sobre Suricata: https://blog.elhacker.net/2017/04/ids-ips-suricata-reglas-rules.html

Información de IOT

- ¿Qué es IOT? : URL: https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html
- Uso de IOT. URL: https://www.sothis.tech/internet-of-things-iot-arte-conectar-cualquier-cosa-internet/
- Macchina.io, “IoT Edge Device Software Development and Secure Remote Access Solutions”, 2019. [Online]. URL: https://macchina.io/.

Configurar una Raspberry pi como punto de acceso wifi:

- Recogida de datos en un punto wifi: URL: https://fwhibbit.es/aire-i-introduccion-y-recogida-de-datos-mediante-sonda
- Sistema de Analsis de un punto wifi: URL: https://fwhibbit.es/aire-ii-instalacion-y-uso-del-sistema-de-analisis
- Raspberry-pi org. "Setting up a Raspberry Pi as an access point in a standalone network (NAT)". URL: https://www.raspberrypi.org/documentation/configuration/wireless/accesspoint.md
- Configura Raspberry Pi como punto de acceso WiFi: URL: http://msrobotics.net/index.php/laboratorio-pi/227-configura-Raspberry-pi-como-punto-de-acceso-wifi

- Raspberri Pi. Hotspot. URL: https://github.com/jayluxferro/RaspberryPi-Hotspot

Configuración de Maltrail:

- Fuente del programa: https://github.com/stamparm/maltrai
- Configuración Maltrail: Monitorizar Tráfico Malicioso en tu Red: https://esgeeks.com/maltrail-monitorizar-trafico-malicioso/

Configuración de ElasticSearch:

- Guia de referencia: https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html

Configuración de LogStash:

- Guia de referencia : https://www.elastic.co/guide/en/logstash/current/installing-logstash.html
- How to: https://enimbos.com/monitorizacion-de-aplicaciones-usando-elk-stack/

Configuración de Kibana:

- Guia de referencia: https://www.elastic.co/guide/en/kibana/current/windows.html

Configuración de la pila

- https://vcatalan.com/2017/12/pila-elk.html
- https://openwebinars.net/blog/que-es-elk-elasticsearch-logstash-y-kibana/
- https://www.avantica.net/es/blog/elk-stack-implementacion-facil-con-docker