Universidad Internacional de La Rioja
Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Ciberseguridad

Fortificación y Privacidad en Redes Domésticas con Raspberry Pi.

Trabajo fin de estudio presentado por:	Miguel Solana Pérez
Tipo de trabajo:	Desarrollo Software
Director/a:	David Cuesta Gómez
Fecha:	10-09-2024

Resumen

El objetivo de este trabajo ha sido desarrollar una solución integral para mejorar la seguridad y privacidad en redes domésticas utilizando una Raspberry Pi como plataforma central. Se han implementado diversas herramientas de código abierto, incluyendo WireGuard para la creación de una VPN, Pi-hole para el filtrado DNS, Suricata como sistema de detección y prevención de intrusiones, y un Dashboard de monitorización mediante el uso de Elastic Stack. La metodología se centró en la configuración y optimización de estos componentes mediante Docker, para que puedan ser fácilmente replicados y utilizados por usuarios con conocimientos técnicos limitados. Los resultados demostraron que es posible establecer una red doméstica segura y privada, utilizando hardware accesible y herramientas de software libre. Las conclusiones resaltan la viabilidad y efectividad de la solución propuesta, así como su potencial para ser expandida y mejorada en futuros desarrollos.

Palabras clave: Seguridad en redes domésticas, Raspberry Pi, VPN, Detección y prevención de intrusiones, Docker.

Abstract

The objective of this work has been to develop a comprehensive solution to improve security and privacy in home networks using a Raspberry Pi as a central platform. Several open-source tools have been implemented, including WireGuard for the creation of a VPN, Pi-hole for DNS filtering, Suricata as an intrusion detection and prevention system, and a monitoring Dashboard using Elastic Stack. The methodology focused on configuring and optimizing these components so that they can be easily replicated and used by users with limited technical knowledge. The results demonstrated that it is possible to establish a secure and private home network, using affordable hardware and free software tools. The conclusions highlight the feasibility and effectiveness of the proposed solution, as well as its potential to be expanded and improved in future developments.

Keywords: Home network security, Raspberry Pi, VPN, Intrusion detection and prevention, Docker.

Agradecimientos

Este, es un momento muy valioso, ya que con la entrega de este trabajo pongo el punto final y marco como completada una de las metas más importantes y que más clara he tenido desde hace muchos años. Comencé la carrera de Ingeniería Telemática en 2017 sin tener realmente claro a dónde quería ir, pero con el paso de los meses y las asignaturas, descubrí a lo que realmente quería dedicarme. Desde ese momento, mi meta fue terminar la carrera y especializarme en la rama que tanto me llamaba la atención y me gustaba: la ciberseguridad. Con la entrega de este trabajo de fin de máster, puedo decir que lo he conseguido.

A continuación, quiero dedicar unas líneas para agradecer a todas aquellas personas que me han animado y apoyado a lo largo de la realización de este máster.

A mi pareja, Clau, uno de los pilares más importantes en mi vida, y partícipe de todas mis decisiones. Por apoyarme y animarme siempre, en todas las decisiones que tomo, y enseñarme a que siempre hay una solución posible a cada uno de los problemas que llegan. También, hacerle una mención especial por haber sido mi ayudante en las pruebas de este TFM, aportando su granito de arena desde la otra punta del mundo, ayudándome en los test de las aplicaciones desarrolladas.

A mis padres, Asun y Fernando, quienes siempre han estado presentes, animándome y aconsejándome en todas mis decisiones, y empujándome a ir por todas y lograr los objetivos que me proponía.

A mi hermano, Rober, por labrar el camino que me ha orientado en tantas ocasiones, y por haberme animado siempre a continuar con mis objetivos y a no tirar la toalla. Y a mi cuñada Yas, que siempre es un placer hablar con ella, recibir una sonrisa y sentir que todo va a salir bien.

Y, finalmente, a mis suegros, cuñados y mi sobrino Nico, que siempre me han ayudado directa e indirectamente, haciéndome entender que, sin importar lo dura y exigente que pueda ser una semana, al llegar el fin de semana todo se simplifica y se resuelve.

Muchas gracias a todos.

Índice de contenidos

Índice de figuras

1. INTRODUCCIÓN

En el contexto actual de las crecientes amenazas en internet y preocupaciones por la privacidad de datos, la seguridad en redes domésticas se ha convertido en un tema de vital importancia. Este trabajo de fin de máster propone abordar esta problemática mediante la creación de una red doméstica altamente segura utilizando una Raspberry Pi como servidor principal.

1.1. MOTIVACIÓN

La motivación principal de este trabajo ha sido el deseo de desarrollar una solución accesible y práctica para cualquier usuario preocupado por la seguridad en internet, que aspire a fortificar su red doméstica de manera efectiva y sin la necesidad de conocimientos técnicos avanzados. El uso de una Raspberry Pi como hardware principal ofrece una solución económica y de fácil acceso para mejorar la seguridad y la privacidad en el hogar.

Las causas fundamentales de la falta de seguridad en las redes domésticas incluyen la falta de conciencia sobre los riesgos de seguridad en internet, la complejidad de los dispositivos conectados y la falta de estándares de seguridad uniformes en la industria.

Este problema es relevante por diversas razones. En primer lugar, la seguridad y la privacidad en internet son preocupaciones cada vez más importantes en la sociedad moderna, y la protección de las redes domésticas es un aspecto fundamental de esta cuestión. Además, el hogar es el lugar donde se almacena una gran cantidad de datos personales y sensibles, por lo que garantizar la seguridad en este entorno es crucial para proteger la privacidad de los usuarios y sus familias. Por último, la accesibilidad de la solución propuesta significa que puede beneficiar a una amplia gama de usuarios, desde aquellos con conocimientos técnicos limitados hasta aquellos con experiencia avanzada en tecnología.

1.2. PLANTEAMIENTO DEL PROBLEMA

El problema abordado en este trabajo es la falta de seguridad adecuada en las redes domésticas, que se traduce en la exposición a una gran variedad de amenazas cibernéticas, incluyendo intrusiones maliciosas, robo de datos, acceso no autorizado a dispositivos conectados, etc. Este problema aumenta debido a la falta de conciencia y conocimientos técnicos entre los usuarios domésticos, así como por la complejidad creciente de los dispositivos conectados.

La solución propuesta a este problema es la securización de la red doméstica mediante el uso de una Raspberry Pi como servidor principal, que proporcione unas funcionalidades de seguridad básicas como son las siguientes:

- VPN con la que poder conectarnos a esta red de forma externa y mantener en todo momento nuestra seguridad en la web.
- Servidor DNS que filtre las peticiones dentro de las páginas web mediante listas negras de resoluciones de dominio no deseadas.
- Uso de un software que emplee funciones de IDS e IPS para analizar y/o bloquear el tráfico no deseado o malicioso.
- Configuración de un sistema de monitorización de tráfico activo, al que se le enviarán todos los logs para así crear un *dashboard* con el que visualizar los datos de una forma más amigable.

Tal y como se ha descrito en la motivación, el objetivo de este trabajo es que la solución desarrollada pueda ser utilizada por usuarios con pocos o ningún conocimiento técnico, por lo que la mayor parte de la implementación de las herramientas tratará de llevarse a cabo mediante el uso de contenedores Docker para que los usuarios que quieran hacer uso de esta puedan simplemente descargarlo y tenerlo operativo al instante.

1.3. ESTRUCTURA DEL TRABAJO

La estructura de este trabajo está formada por diferentes capítulos divididos por temáticas, dichos capítulos son:

- **Capítulo 1 – Introducción:** Como introducción al trabajo, se define la motivación base de este proyecto, el problema latente que se busca resolver, cómo resolverlo y la estructura que sigue esta memoria.
- **Capítulo 2 – Estado del Arte:** En este capítulo se proporciona un contexto inicial de los conceptos principales que deben entenderse para comprender la motivación de este trabajo. También se explica cada una de las tecnologías que se usarán en esta solución.
- **Capítulo 3 - Objetivos concretos y metodología de trabajo:** Aquí se detallarán de forma concreta los objetivos de este trabajo junto con las tecnologías que se usarán para satisfacer dichos objetivos.
- **Capítulo 4 – Buenas prácticas de configuración de la red doméstica:** Mediante este capítulo se pretende ofrecer al lector una guía de consejos y buenas prácticas para configurar una red doméstica segura.
- **Capítulo 5 - Desarrollo específico de la contribución:** En este capítulo se definirá la implementación técnica de cada una de las funcionalidades elegidas para la fortificación de la red doméstica. Tras ello, se expondrán una serie de pruebas para validar su funcionamiento.
- **Capítulo 6 – Conclusiones y trabajo futuro:** Finalmente se sintetizan las conclusiones obtenidas durante el transcurso de este proyecto, así como opciones de trabajos futuros contempladas.
- **Bibliografía:** En este apartado se incluirán las referencias bibliográficas utilizadas durante la investigación del trabajo.
- **Anexos:** En los anexos se incluirá todo el contenido que pueda ser de interés en relación con el trabajo, pero que no sea esencial para el entendimiento de este.

2. ESTADO DEL ARTE

2.1.INTRODUCCIÓN

En esta sección se busca contextualizar al lector de la situación actual de seguridad en las redes domésticas, así como de la motivación que ha llevado al desarrollo de esta solución para agregar un extra de seguridad y de privacidad en dichas redes. Para ello, se proporciona un contexto inicial sobre cómo está estructurada una red doméstica, y de algunas de las amenazas existentes en ellas. También, se proporciona una explicación del *hardware* elegido para este proyecto, así como de las herramientas *software* que se emplearán para conseguir el objetivo de este trabajo.

2.2.ARQUITECTURA DE UNA RED DOMÉSTICA

La arquitectura típica de una red doméstica incluye un rúter que sirve como el único punto de acceso a Internet. Los dispositivos modernos ofrecen opciones de conectividad avanzadas, como múltiples puertos Ethernet y acceso inalámbrico mejorado, con estándares de seguridad como WPA3 (Delaney, 2024). Sin embargo, las configuraciones predeterminadas pueden dejar la red vulnerable a ataques en aquellos dispositivos con la configuración de seguridad inadecuada.

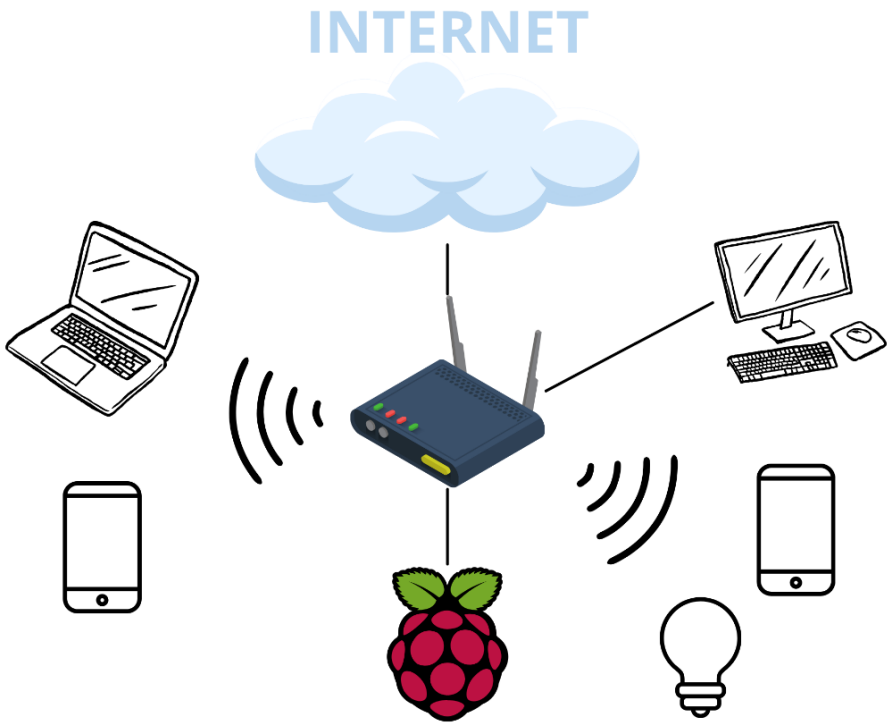
Los dispositivos comunes en las redes domésticas abarcan una amplia gama, desde smartphones y ordenadores tanto de sobremesa como portátiles, hasta cámaras de vigilancia videoconsolas, así como una amplia gama de dispositivos IoT, todos ellos utilizan diversos protocolos para su funcionamiento y conexión. Estos dispositivos pueden acceder a la red local o remota a través de protocolos como HTTP, RTSP, MQTT, DHCP, DNS, etc.

En cuanto a la arquitectura de red, la red doméstica típica separa el interfaz Wi-Fi del interfaz Ethernet a través de un conmutador de red. Sin embargo, algunos enrutadores domésticos presentan configuraciones variadas, como el uso de DMZ, donde los puertos UDP y TCP están abiertos y expuestos. Los dispositivos de red pueden emplear VLANs para separar el tráfico, pero esta configuración es más adecuada para conmutadores que para rúteres (Leos Rivas, 2017).

2.2.1. Esquema de una red doméstica

A continuación, se muestra el esquema de una red doméstica estándar, de la cual se explican sus componentes:

Figura 1: Arquitectura de una red doméstica. Fuente: Elaboración propia.



En esta arquitectura se puede observar cómo hay seis dispositivos conectados al rúter. Por un lado, conectados mediante Wi-Fi se observan dos teléfonos móviles, un ordenador portátil y una bombilla inteligente. Por otro lado, conectados mediante cable, un ordenador de sobremesa y el dispositivo del que se hablará durante este proyecto, una Raspberry Pi. También, mediante la representación común, con una nube, se muestra la salida hacia internet que tiene el rúter doméstico habitualmente.

En la Figura 1 se pueden apreciar los dos tipos de conexiones existentes: la conexión inalámbrica mediante Wi-Fi y la conexión cableada a través de un cable de red. Además, se observa que el único dispositivo que separa la red local de la inmensa red de redes conocida como Internet es el rúter. Por este motivo, a continuación, se proporciona una breve explicación sobre qué es este dispositivo y para qué sirve.

2.2.2. El rúter

Un rúter es un dispositivo esencial en la arquitectura de una red doméstica, encargado de conectar múltiples dispositivos a Internet y gestionar el tráfico de datos entre ellos. Funciona como un sistema de reenvío inteligente que determina las rutas más eficientes para enviar y recibir información, asegurando así una comunicación rápida y segura. Además de facilitar la conexión a Internet, un enrutador permite crear redes locales, permitiendo a los dispositivos comunicarse entre sí para compartir archivos y recursos. Los rúteres modernos vienen equipados con múltiples puertos Ethernet para conexiones cableadas y soporte para conexiones inalámbricas Wi-Fi, utilizando estándares de seguridad avanzados como WPA3. Estas características hacen que los enrutadores no solo sean puntos de acceso a Internet, sino también herramientas fundamentales para proteger la red doméstica contra amenazas externas al ofrecer opciones de seguridad y control sobre el tráfico de la red (Cisco Systems, 2022).

2.3. LA SEGURIDAD EN REDES DOMÉSTICAS

2.3.1. Los pilares de la seguridad en internet

La norma ISO27001 establece los tres pilares fundamentales de la seguridad informática. Dichos pilares son:

- Integridad: Este pilar garantiza que la información sea precisa, completa y confiable, protegiéndola así de ser modificada por error o de forma no autorizada.
- Confidencialidad: La información debe permanecer privada en todo momento. Solamente debe ser accesible para las personas autorizadas, protegiéndose así de su divulgación no autorizada y del uso indebido de esta.
- Disponibilidad: Consiste en la capacidad de acceder a la información en todo momento, independientemente del lugar en el que se encuentre.

Estos, son los pilares más importantes de la seguridad en internet, y se aplican en todos los ámbitos y escenarios. A continuación, se muestra cómo dichos pilares deben integrarse también en la seguridad del hogar.

2.3.2. Panorama actual de la seguridad en el hogar

La seguridad en la red doméstica no se limita a tener un antivirus instalado en un ordenador o a establecer una contraseña para el Wi-Fi. Dentro de la red doméstica, existen multitud de dispositivos conectados, y diversas personas pueden estar navegando a través de ellos. Por ejemplo, se puede tener una *Smart TV* para visualizar contenido en *streaming*; uno o varios ordenadores para consultar el correo electrónico, jugar, realizar compras en línea, entre otras actividades; así como diferentes tipos de dispositivos IoT, como bombillas inteligentes, neveras, interruptores, asistentes virtuales inteligentes como Alexa o Google Home, entre otros, y cualquiera de ellos puede ser la puerta de entrada de un ataque. La seguridad en la red doméstica es el núcleo fundamental para que todo el hogar esté protegido en el día a día, y se minimice el riesgo de que un ataque se haga efectivo (TP-Link, 2020).

Si bien es cierto que un usuario individual no suele ser el objetivo principal de organizaciones de cibercriminales, ya que estas acostumbran a atacar de forma coordinada a grandes empresas de las que pueden conseguir un gran beneficio, sí que lo es de ciberataques dirigidos a víctimas de forma masiva, o de ataques que se dan por navegar y descargar contenido de forma poco prudente, o por no tener una correcta configuración de la red doméstica para prevenir las posibles amenazas.

2.3.3. Posibles amenazas en una red doméstica

Algunas de las posibles amenazas que podemos encontrar en una red doméstica son las siguientes (Hurtado, 2024) (Berenguer Falcó, 2024):

- Ataques de fuerza bruta: Son intentos de adivinar contraseñas como la de la red Wi-Fi o la del rúter (entre otras), mediante repetidas pruebas y con diferentes combinaciones.
- Acceso no autorizado: Cuando una persona no autorizada intenta acceder a la red Wi-Fi. Este caso puede darse debido a una contraseña insegura, o debido a puertos abiertos en dispositivos de la red, como cámaras de seguridad o rúters mal configurados.

- Man in the middle (MitM): Esta amenaza puede ser el resultado del éxito de las 2 anteriores. Una vez se ha accedido a la red de forma no autorizada, el atacante podría espiar todo el tráfico que circula por dicha red con fines maliciosos.
- Phishing: Esta es una amenaza muy común en las redes domésticas. Se da mayoritariamente a través del correo electrónico, mensajes de texto (*smishing*) o incluso llamadas telefónicas (*vishing*). Consiste en utilizar técnicas engañosas haciéndose pasar por entidades legítimas para obtener información confidencial como contraseñas o datos bancarios.
- Vulnerabilidades en dispositivos IoT: Estos dispositivos son cada vez más utilizados en los hogares. Esto significa que la superficie de ataque se está aumentando cada vez más. Estos dispositivos pueden tener una mala configuración que haga que posean vulnerabilidades, y que estas puedan ser aprovechadas por un atacante para acceder a la red.
- *Malware* y Virus: El *malware* y los virus son amenazas comunes que pueden comprometer la seguridad de una red doméstica. Este tipo de *software* malicioso puede propagarse fácilmente a través de diversos medios, como descargas de archivos infectados, correos electrónicos maliciosos, o dispositivos externos como memorias USB y discos duros portátiles. Una vez que el *malware* se introduce en la red, puede infectar no solo los ordenadores y portátiles, sino también otros dispositivos conectados, como termostatos inteligentes, cámaras de seguridad y otros dispositivos IoT.

  El *malware* puede adoptar diferentes formas, cada una con efectos perjudiciales específicos:

  - *Spyware*: Este tipo de *malware* se instala en los dispositivos sin el conocimiento del usuario y monitorea sus actividades, recopilando información sensible como contraseñas, datos bancarios o historiales de navegación.
  - *Ransomware*: Un tipo particularmente dañino de *malware* que cifra los archivos del usuario y exige un rescate para devolver el acceso a ellos. Este tipo de ataque puede ser devastador, especialmente si afecta a dispositivos críticos dentro de la red.
  - Troyanos: Aparentemente inofensivo, este tipo de *malware* se disfraza como software legítimo. Una vez instalado, permite a los atacantes acceso remoto a

la red y los dispositivos, desde donde pueden robar información, instalar otros tipos de *malware*, o incluso controlar los dispositivos de forma remota.

Todos estos ejemplos son amenazas que existen, y que podrían materializarse en nuestra red (Sicajá y otros) (Guanotoa Chuma, 2024).

2.4.SEGURIDAD EN REDES WI-FI

2.4.1. Protocolos de seguridad

Para entender la seguridad en la red doméstica, es necesario conocer los estándares de seguridad del protocolo Wi-Fi. Estos estándares han evolucionado a lo largo del tiempo, mejorando la protección de las redes inalámbricas frente a diversas amenazas. Para ello se proporciona una breve explicación de los principales protocolos de seguridad Wi-Fi: WEP, WPA, WPA2 y WPA3 (Díaz Amorín, 2023):

- WEP (*Wired Equivalent Privacy*): WEP fue el primer estándar de seguridad Wi-Fi, introducido en 1997. WEP utiliza una clave estática hexadecimal de 64 o 128 bits, para cifrar los datos que se transmiten en la red. Sin embargo, debido a vulnerabilidades en su algoritmo de cifrado y a la facilidad con la que se pueden romper las claves, WEP se considera inseguro y obsoleto. Hoy en día, no se recomienda su uso.
- WPA (*Wi-Fi Protected Access*): En respuesta a las vulnerabilidades de WEP, se introdujo WPA en 2003 como una solución temporal mientras se desarrollaba un estándar más seguro. WPA mejora la seguridad al utilizar TKIP (Temporal Key Integrity Protocol), que genera dinámicamente una nueva clave para cada paquete de datos transmitido. Aunque WPA es más seguro que WEP, también tiene vulnerabilidades y ha sido reemplazado por estándares más robustos.
- WPA2 (*Wi-Fi Protected Access II*): WPA2, lanzado en 2004, se convirtió en el estándar de seguridad Wi-Fi más utilizado durante muchos años. Este protocolo reemplaza TKIP con CCMP (*Counter Mode Cipher Block Chaining Message Authentication Code Protocol*), basado en el estándar de cifrado AES (*Advanced Encryption Standard*). WPA2 proporciona un nivel de seguridad significativamente más alto que WPA y se ha mantenido como un estándar confiable durante mucho tiempo. Sin embargo, en los

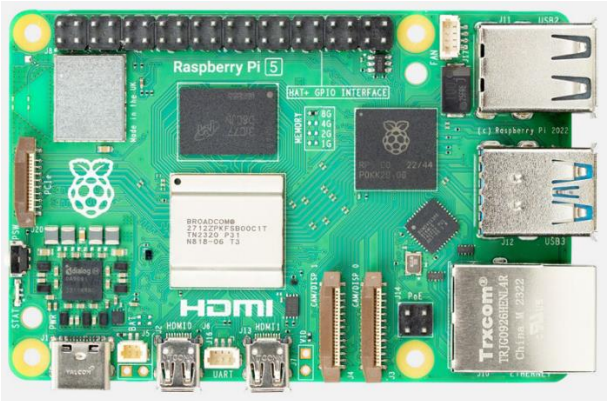
últimos años, se han descubierto algunas vulnerabilidades en WPA2, como el ataque *KRACK*, que han llevado al desarrollo de un nuevo estándar.

- WPA3 (*Wi-Fi Protected Access III*): WPA3, lanzado en 2018, es el estándar más reciente de seguridad Wi-Fi. Ofrece mejoras significativas respecto a WPA2, como un cifrado más robusto y un protocolo de autenticación más seguro (SAE, *Simultaneous Authentication of Equals*), que protege mejor contra ataques de fuerza bruta. Además, WPA3 incluye características específicas para redes públicas y dispositivos IoT, proporcionando una mayor seguridad incluso en entornos menos controlados. WPA3 es actualmente el estándar recomendado para garantizar la máxima seguridad en redes Wi-Fi.

2.5. HARDWARE UTILIZADO

Para llevar a cabo este proyecto, es esencial la elección de un hardware adecuado. Este ha de ser capaz de soportar y manejar todo el software que se le va a cargar, por lo que debe tener unos requisitos mínimos en cuanto a capacidad de cómputo. También debe poder estar encendido constantemente, ya que el objetivo es que los servicios que va a correr estén activos siempre, por lo que tampoco debería tener un consumo excesivo de electricidad. Y finalmente, debe de tratarse de un hardware asequible, fácil de adquirir y fácil de configurar. Uno de los objetivos de este proyecto es diseñar una solución asequible para cualquier tipo de usuario. Teniendo en cuenta todos estos requisitos, la elección de hardware escogida ha sido el microordenador Raspberry Pi 5 en su versión de 8Gb de memoria RAM.

Figura 2: Raspberry Pi 5. Fuente: (Raspberry Pi Ltd, 2024).



2.5.1. Raspberry Pi

Una Raspberry Pi es un microordenador de bajo costo y tamaño reducido, diseñado para ser accesible y versátil. Fue desarrollado por la Fundación Raspberry Pi con el objetivo de promover la enseñanza de la informática en escuelas y países en desarrollo, pero su popularidad ha crecido enormemente debido a su capacidad para ser utilizada en una amplia gama de proyectos (Raspberry Pi Foundation, 2024).

Sus especificaciones técnicas son las siguientes (Raspberry Pi Ltd, 2024):

- Procesador:
  - Modelo: Broadcom BCM2712
  - Arquitectura: Quad-core ARM Cortex-A76
  - Velocidad de reloj: 2.4 GHz
- Memoria RAM:
  - Capacidad: 8 GB LPDDR4X
- GPU (Unidad de Procesamiento Gráfico):
  - Modelo: VideoCore VII
  - Soporte de gráficos: OpenGL ES 3.1, Vulkan 1.2
  - Capacidad: Decodificación de video en 4K H.265 y H.264
- Almacenamiento:
  - MicroSD: Ranura para tarjeta microSD (almacenamiento principal)
  - Soporte para arranque desde USB y red
- Conectividad:
  - Wi-Fi: 802.11ac Wi-Fi (2.4 GHz y 5 GHz)
  - Bluetooth: Bluetooth 5.0, BLE (Bluetooth Low Energy)
  - Ethernet: 1 puerto Gigabit Ethernet
  - USB: 2 puertos USB 3.0, 2 puertos USB 2.0
- Puertos de Video:
  - Micro HDMI: 2 puertos micro HDMI, soporte para dual 4K@60Hz
  - CSI (Camera Serial Interface): 2 puertos MIPI CSI para cámaras y pantallas
- Puertos GPIO:
  - Pinout: 40 pines GPIO, compatible con versiones anteriores

- Audio:
    - Salida de audio estéreo: a través de HDMI y GPIO
    - Salida de audio analógica y video compuesto: a través de pines GPIO dedicados
- Alimentación:
    - Conector: USB-C (con soporte para Power Delivery)
    - Requisito de energía: 5V, 5A (mínimo recomendado)
- Dimensiones:
    - Tamaño: 85.6 mm x 56.5 mm (mismo tamaño que los modelos anteriores)
- Sistema operativo:
    - OS recomendado: Raspberry Pi OS (Linux basado en Debian), pero también soporta otros sistemas operativos.

2.5.2. Alternativas de hardware

Por otra parte, se han investigado otras alternativas para llevar a cabo este proyecto, algunas de ellas se enumeran a continuación con sus ventajas y desventajas:

- Arduino (Arduino®, 2018):
    - Ventajas: Bajo costo, gran comunidad de soporte, ideal para proyectos de IoT y dispositivos sencillos.
    - Desventajas: No tiene el mismo rendimiento que una Raspberry Pi, ni está diseñado para funcionar como un servidor o para manejar tareas complejas de red y ciberseguridad.
- Intel NUC (Next Unit of Computing) (Penalva, 2023):
    - Ventajas: Potente, versátil, capaz de ejecutar aplicaciones de ciberseguridad más intensivas y manejar mayores cargas de trabajo.
    - Desventajas: Más caro que la Raspberry Pi, consume más energía, y puede ser más complejo de configurar y mantener.
- Orange Pi 5 Pro (Orange Pi, 2024):
    - Ventajas: Rendimiento similar a Raspberry Pi 5, e incluso existe una versión de 16Gb de memoria RAM. Menor precio del dispositivo.

- Desventajas: Es menos conocida y posee una menor comunidad, por lo tanto, pueden existir menos integraciones de software para ella.

2.5.3. Conclusiones

Finalmente, para la elección final los motivos que se tuvieron en cuenta fueron los siguientes:

- Costo: En comparación con otras opciones de igual rendimiento, el coste de la Raspberry Pi es asequible para cualquier usuario que quiera llevar a cabo este proyecto, siendo 131.71€ el kit de iniciación con todo lo necesario para su puesta en funcionamiento (RaspiPC.es, 2024).
- Tamaño y Consumo Energético: Su tamaño total (carcasa incluida) es de 10x7x3 centímetros únicamente, por lo que es un dispositivo pequeño que puede guardarse fácilmente. Por otra parte, tiene un consumo energético de 12W, que, comparado con otros dispositivos como el Intel NUC que tiene 60W (Penalva, 2023), se considera un consumo bajo, y que permite que esté encendido constantemente.
- Comunidad y Soporte: Raspberry Pi posee una gran comunidad de usuarios y desarrolladores que facilita encontrar documentación, soporte y recursos para la configuración de proyectos de seguridad en la Raspberry Pi.
- Facilidad de implementación: Este hardware es fácil de configurar y mantener, además de que existen multitud de guías y videos en internet de cómo hacerlo, lo que es beneficioso tanto para usuarios avanzados como para aquellos con menos experiencia técnica.

Estos motivos han marcado la elección del dispositivo Raspberry Pi 5 como el ganador entre las alternativas que existen actualmente en el mercado.

2.6. DOCKER

2.6.1. ¿Qué es Docker?

Docker es una plataforma de código abierto diseñada para facilitar la creación, implementación y ejecución de aplicaciones mediante el uso de contenedores. Un contenedor es una unidad de software ligera y autónoma que incluye todo lo necesario para que una aplicación se ejecute: el código, las bibliotecas, las dependencias y la configuración del sistema. Docker permite empaquetar una aplicación y sus dependencias en un solo contenedor que puede ejecutarse en cualquier entorno, garantizando que el software funcione de manera uniforme independientemente del entorno donde se despliegue (D., 2024).

Esta capacidad que ofrece Docker de empaquetar y ejecutar aplicaciones de forma aislada y consistente ha hecho que se convierta en una herramienta esencial en aquellos escenarios en los que la portabilidad y eficacia son cruciales. A lo largo de este apartado se expondrá cómo funciona Docker, sus principales ventajas, y por qué ha sido escogido para este proyecto.

2.6.2. Origen y Evolución

Docker fue lanzado en 2013 por la compañía Docker Inc., y desde entonces ha revolucionado la manera en que las aplicaciones son desarrolladas, distribuidas y ejecutadas. Antes de Docker, la virtualización tradicional implicaba el uso de máquinas virtuales (VMs), que, aunque eficaces, eran pesadas y consumían una gran cantidad de recursos. Docker introdujo el concepto de contenedores ligeros que operan sobre el mismo sistema operativo del host, lo que permite ejecutar múltiples contenedores sin la sobrecarga de una VM completa (Docker Inc, Docker, 2024) (Wikipedia, 2024).

2.6.3. Conceptos clave de Docker

2.6.3.1. Contenedores vs Máquinas Virtuales

Los contenedores y las máquinas virtuales son tecnologías que permiten ejecutar múltiples aplicaciones de forma aislada en un solo host. Sin embargo, los contenedores son mucho más ligeros que las máquinas virtuales. Mientras que una VM incluye un sistema operativo

completo junto con la aplicación y sus dependencias, un contenedor comparte el mismo núcleo del sistema operativo del host y solo incluye las bibliotecas y configuraciones necesarias para el correcto funcionamiento de la aplicación. Esto hace que los contenedores sean más eficientes en cuanto a recursos, más livianos y rápidos de iniciar y detener (AWS, ¿Cuál es la diferencia entre Docker y una máquina virtual?, 2023).

2.6.3.2. Imágenes y Contenedores

Una imagen de Docker es una plantilla de solo lectura que incluye todo lo necesario para ejecutar una aplicación, desde el código fuente hasta las dependencias y configuraciones necesarias. A partir de una imagen, Docker puede crear un contenedor, que es una instancia ejecutable de esa imagen. Esto permite que una misma imagen pueda ser utilizada para desplegar múltiples contenedores idénticos en distintos entornos (AWS, ¿Cuál es la diferencia entre imágenes y contenedores de Docker?, 2023).

2.6.3.3. Docker Hub

Docker Hub es un servicio de almacenamiento y distribución de imágenes de Docker. Es un repositorio público donde los desarrolladores pueden subir y compartir sus imágenes, permitiendo a otros usuarios descargarlas y usarlas en sus propios proyectos. Docker Hub facilita la reutilización de configuraciones de contenedores y fomenta la colaboración entre desarrolladores de la comunidad al hacer que las imágenes sean accesibles globalmente. En este proyecto se utilizará Docker Hub para el despliegue de alguno de los servicios como se verá posteriormente.

2.6.4. Ventajas del uso de Docker

2.6.4.1. Portabilidad

Una de las principales ventajas de Docker es la portabilidad. Los contenedores de Docker pueden ejecutarse de manera consistente en cualquier entorno que soporte Docker, ya sea un servidor local, una máquina virtual, un entorno de desarrollo o un proveedor de nube. Esta

portabilidad asegura que las aplicaciones se comporten de la misma manera sin importar el entorno, eliminando así los problemas de “en mi máquina sí funciona”.

2.6.4.2. Aislamiento y Seguridad

Gracias al funcionamiento independiente de los contenedores, Docker proporciona un aislamiento de procesos y recursos entre estos, lo que significa que las aplicaciones que se ejecutan en diferentes contenedores no interfieren entre sí. Este aislamiento no solo mejora la estabilidad de las aplicaciones, sino que también añade una capa adicional de seguridad, ya que en el caso de que un contenedor sea comprometido, este no afecta a otros contenedores ni al sistema host.

2.6.4.3. Eficiencia en Recursos

A diferencia de las máquinas virtuales, los contenedores de Docker son extremadamente ligeros. Pueden arrancar en cuestión de segundos y consumir significativamente menos recursos, lo que los hace ideales para dispositivos con capacidades limitadas, como en el caso de este proyecto, la Raspberry Pi. Esto permite maximizar el uso de recursos en entornos con restricciones de hardware.

2.6.4.4. Escalabilidad y Mantenimiento

Docker facilita la escalabilidad horizontal, ya que permite desplegar múltiples instancias de una aplicación de manera sencilla. Además, la actualización y el mantenimiento de las aplicaciones son más eficientes, ya que los contenedores pueden ser actualizados individualmente sin afectar al resto del sistema. Esto asegura que los usuarios siempre tengan acceso a las versiones más recientes y seguras del software.

2.6.5. ¿Por qué se ha elegido Docker?

Finalmente, la elección de este software para llevar a cabo este proyecto ha sido un factor clave. Los motivos que han marcado esta decisión han sido:

2.6.5.1. Facilidad de Replicación e Implementación

Docker permite empaquetar la configuración completa de una red segura en contenedores que pueden ser fácilmente distribuidos y replicados. Esto ha sido crucial para este trabajo, ya que uno de los objetivos es facilitar que cualquier usuario pueda implementar esta solución en su red doméstica sin tener que lidiar con complejas configuraciones o dependencias. Con Docker, los usuarios pueden simplemente descargar las imágenes necesarias y ejecutar los contenedores para tener la configuración de red segura en funcionamiento en cuestión de minutos.

2.6.5.2. Flexibilidad y Adaptabilidad

Docker permite que diferentes herramientas de seguridad se ejecuten en contenedores separados, lo que facilita la gestión y el mantenimiento de cada componente de la red. Esto también permite que los usuarios personalicen su configuración según sus necesidades, añadiendo o eliminando contenedores según sea necesario. Por ejemplo, un usuario podría optar por no usar un sistema servidor DNS en su red, simplemente eliminando el contenedor correspondiente.

2.6.5.3. Facilidad de Actualización y Mantenimiento

Las actualizaciones de software pueden ser complicadas, especialmente en entornos donde varias aplicaciones interactúan entre sí. Docker simplifica este proceso al permitir que las imágenes de contenedores se actualicen de forma independiente. Los usuarios solo necesitan actualizar la imagen del contenedor y reiniciarlo, asegurando que siempre estén utilizando las últimas versiones de las herramientas de seguridad. Esto es especialmente útil en un entorno de red doméstica donde los usuarios pueden no tener conocimientos técnicos avanzados.

2.7.SOFTWARE UTILIZADO

Tras esta breve introducción a Docker y sus conceptos clave, queda explicar el software utilizado para la realización de este proyecto. A continuación, se proporciona una contextualización de cada una de las herramientas de seguridad utilizadas para esta implementación. Estas herramientas incluyen WireGuard, DuckDNS, Pi-hole, Suricata y Elastic Stack, cada una elegida por su capacidad para cumplir con requisitos específicos de seguridad, eficiencia y análisis de datos. A continuación, se detallan las funciones de cada una y los motivos detrás de su selección.

2.7.1. WireGuard

WireGuard es un protocolo de red privada virtual (VPN) que destaca por su simplicidad y rendimiento superior en comparación con otras soluciones de VPN tradicionales. A diferencia de protocolos más antiguos como OpenVPN o IPSec, WireGuard está diseñado con un enfoque moderno, utilizando cifrados de última generación que garantizan tanto la seguridad como la eficiencia en la transferencia de datos. Además, su implementación en código es relativamente compacta, lo que facilita la auditoría de seguridad y reduce el riesgo de vulnerabilidades (WireGuard®, 2022).

WireGuard ha sido seleccionado por su capacidad para proporcionar conexiones VPN seguras con una configuración mínima y sencilla mediante su interfaz web. Su rendimiento es notablemente superior, con menos sobrecarga de procesamiento, lo que lo hace ideal para dispositivos de bajo consumo como la Raspberry Pi (De Luz, Rendimiento de WireGuard, 2024). En el contexto de este proyecto, WireGuard permite a los usuarios conectarse de manera segura a su red doméstica desde ubicaciones remotas, en especial cuando el usuario se conecta a través de una Wi-Fi pública o insegura, asegurando que los datos transmitidos estén cifrados y protegidos contra interceptaciones no autorizadas.

2.7.2. DuckDNS

DuckDNS es un servicio gratuito de DNS dinámico que permite a los usuarios asociar un nombre de dominio fácil de recordar a su dirección IP pública, incluso cuando esta cambia

regularmente. Esto es particularmente útil para aquellos que desean acceder a su red doméstica desde ubicaciones remotas, ya que mantiene la accesibilidad sin necesidad de pagar por un servicio de DNS estático (De Luz, Duck DNS, el mejor servicio DNS dinámico gratuito que puedes usar, 2024).

DuckDNS ha sido elegido debido a su simplicidad y eficacia en mantener una conexión constante con la red doméstica, independientemente de los cambios en la dirección IP pública. En combinación con WireGuard, DuckDNS facilita el acceso remoto seguro y constante a los recursos de la red, haciendo que la implementación de una VPN en un entorno doméstico sea mucho más manejable para el usuario promedio.

2.7.3. Pi-hole

Pi-hole es un servidor DNS que a su vez funciona como bloqueador de anuncios a nivel de red, filtrando el tráfico no deseado antes de que llegue a los dispositivos conectados en la red. Su función principal es bloquear solicitudes de dominios conocidos por tratarse de anuncios, rastreadores y contenido malicioso, lo que resulta en una navegación más rápida y segura para todos los dispositivos de la red. Esto lo consigue mediante el uso de *White Lists* y *Black Lists* para definir los dominios permitidos, y los dominios excluidos. Además, Pi-hole proporciona un *dashboard* con estadísticas detalladas sobre el tráfico DNS, permitiendo a los usuarios tener una visión clara de las solicitudes de red que se realizan desde sus dispositivos. Además, Pi-hole permite personalizar su configuración para aquellos entusiastas que quieran exprimir al máximo su funcionalidad, desde hacer que funcione como servidor DHCP, agregar más *Black Lists* que se consideren oportunas, hasta editar el nivel de privacidad deseado. (Pi-hole®, 2022).

Pi-hole ha sido uno de los puntos clave de este proyecto, puesto que es una de las partes de la implementación que mayor privacidad y seguridad añade a la red doméstica al bloquear automáticamente sitios potencialmente peligrosos, rastreadores de publicidad, etc. Esto es especialmente útil en entornos donde múltiples dispositivos están conectados, como es el caso de las redes domésticas modernas que incluyen desde smartphones hasta dispositivos IoT. Al utilizar Pi-hole, no solo se reduce la exposición a anuncios invasivos, sino que también

se mitiga el riesgo de acceder a dominios que podrían ser utilizados para ataques de phishing o la propagación de *malware*.

2.7.4. Suricata

Suricata es un motor avanzado de detección y prevención de intrusiones (IDS/IPS), así como de monitoreo de seguridad de red. Desarrollado por la *Open Information Security Foundation* (OISF), Suricata es una herramienta poderosa y versátil que puede analizar en tiempo real el tráfico de red para detectar y mitigar amenazas. Suricata opera a nivel de red, lo que le permite examinar el tráfico en busca de comportamientos sospechosos o patrones que coincidan con firmas de amenazas conocidas. Su capacidad para manejar múltiples protocolos y su alto rendimiento lo hacen ideal para entornos que requieren una seguridad robusta. Además de funcionar como un IDS/IPS, Suricata puede ser utilizado como un Network Security Monitoring (NSM), proporcionando visibilidad detallada sobre lo que ocurre dentro de la red.

Las principales funciones de suricata son:

- Motor de reglas: Suricata permite utilizar un *pull* de reglas flexible para definir patrones y comportamientos específicos los cuales se asocian con amenazas conocidas o comportamientos anómalos. Dichas reglas son clave para la detección de intrusiones.
- Detección de intrusiones: Suricata monitorea el tráfico de red en tiempo real, comparando cada paquete con una base de datos de firmas de amenazas conocidas. Si detecta un patrón que coincide con una firma, genera una alerta.
- Prevención de intrusiones: En su modo de IPS, Suricata no solo detecta, sino que también puede bloquear automáticamente el tráfico malicioso, evitando que las amenazas alcancen los dispositivos de la red.
- Inspección profunda de paquetes (DPI): Suricata puede analizar el contenido de los paquetes más allá de los encabezados, permitiendo la detección de amenazas ocultas dentro del tráfico aparentemente legítimo.
- Soporte para protocolos múltiples: Suricata es capaz de analizar una amplia gama de protocolos, incluyendo HTTP, DNS, TLS, FTP, SMB, y más, lo que le permite detectar amenazas en diversos tipos de tráfico de red.

- Análisis de archivos: Suricata puede extraer y analizar archivos transmitidos a través de la red, buscando *malware* u otras amenazas dentro de los archivos.
- Registro de tráfico y análisis: Suricata ofrece capacidades avanzadas para registrar el tráfico de red, proporcionando un recurso valioso para el análisis forense en caso de incidentes de seguridad.

Suricata es otra de las piezas clave de este proyecto. Ha sido seleccionado debido a su capacidad para proporcionar una defensa integral contra una amplia gama de amenazas en la red doméstica. Su versatilidad, que le permite operar tanto como IDS como IPS, asegura que no solo se detecten las amenazas, sino que también se puedan neutralizar antes de que causen daño. Al ser de código abierto, Suricata es altamente configurable y puede ser adaptado a las necesidades específicas de una red doméstica, haciendo que sea una herramienta fundamental en cualquier estrategia de seguridad de red.

2.7.5. ElasticSearch + Kibana + Filebeat

ElasticSearch, Kibana y Filebeat son tres componentes clave del Elastic Stack (anteriormente conocido como ELK Stack), que se utilizan para la recolección, almacenamiento, análisis y visualización de datos en tiempo real.

2.7.5.1. ElasticSearch: El Motor de Búsqueda y Almacenamiento

ElasticSearch es un motor de búsqueda y análisis de texto distribuido, que permite almacenar, buscar y analizar grandes volúmenes de datos de manera rápida y en tiempo real. Es conocido por su escalabilidad, velocidad y capacidad para manejar consultas complejas sobre grandes conjuntos de datos. En este proyecto ElasticSearch actúa como el núcleo del sistema de almacenamiento y búsqueda, indexando y organizando los datos recibidos de Suricata para su posterior análisis y visualización.

2.7.5.2. Kibana: La Herramienta de Visualización

Kibana es una herramienta de visualización de datos que trabaja en conjunto con ElasticSearch. Proporciona una interfaz gráfica de usuario (GUI) que permite a los usuarios explorar, visualizar y entender los datos almacenados en ElasticSearch mediante gráficos, tablas, mapas y otros tipos de visualizaciones interactivas. Kibana se utiliza para visualizar los

datos recolectados y almacenados en ElasticSearch, permitiendo la creación de *dashboards* que ofrecen una visión clara y comprensible del estado de la red y los eventos de seguridad detectados por Suricata.

2.7.5.3. Filebeat: El Agente de Recolección de Logs

Filebeat es un agente de envío de logs, diseñado para recolectar y enviar archivos de log a ElasticSearch o Logstash. Filebeat lee los archivos de log en tiempo real y los envía a los destinos especificados para su almacenamiento y análisis. Filebeat es responsable de recoger los logs generados por Suricata, y enviarlos a ElasticSearch para su almacenamiento y posterior análisis en Kibana. Esto asegura que los datos de logs estén disponibles en tiempo real para su visualización y análisis.

En conjunto, estos componentes permiten implementar un sistema robusto y escalable para monitorear y proteger la red, proporcionando una visión completa y en tiempo real de lo que sucede en el entorno doméstico.

2.8. CONCLUSIONES DEL CAPÍTULO

A lo largo de este capítulo se ha hablado sobre el panorama actual que existe en las redes domésticas relacionado con la seguridad informática de las mismas. Se ha contextualizado al lector sobre la situación general, y se ha abordado una posible solución a dicho problema mediante la configuración de determinado software del cual se ha hablado también en este capítulo. Las conclusiones que se pueden extraer de toda esta información son:

2.8.1. Resumen del Panorama Actual

En el contexto de la ciberseguridad, las redes domésticas han experimentado una evolución significativa en las últimas décadas. A medida que ha aumentado el número de dispositivos conectados (como dispositivos IoT, smartphones, videoconsolas, etc.), la superficie de ataque para posibles amenazas ha crecido exponencialmente.

A pesar de los avances en tecnología, como el uso de protocolos de seguridad como WPA3 en conexiones Wi-Fi y la incorporación de firewalls en los rúteres, las redes domésticas siguen siendo vulnerables. Esto se debe principalmente a configuraciones predeterminadas

inseguras, la falta de actualizaciones en los dispositivos y en general, una escasa conciencia en seguridad por parte de los usuarios. El panorama actual evidencia la necesidad de soluciones accesibles y efectivas que puedan ser implementadas por usuarios comunes para proteger sus redes domésticas.

2.8.2. Importancia de las Herramientas Seleccionadas

Para abordar este problema en las redes domésticas, se han identificado varias herramientas que son clave en la construcción de un entorno seguro:

- WireGuard: Este protocolo VPN es conocido por su eficiencia y seguridad, permitiendo a los usuarios cifrar su tráfico de red, protegiendo sus datos de posibles interceptaciones y proporcionando un acceso seguro a la red desde ubicaciones remotas.
- Pi-hole: Como un bloqueador de publicidad y filtrado de DNS, Pi-hole ofrece una capa adicional de seguridad al bloquear dominios maliciosos y anuncios, lo que reduce la exposición a *malware* y *trackers*. Además, Pi-hole destaca por su amplia comunidad de soporte y su facilidad de uso lo que lo convierte en una opción popular para el filtrado de DNS
- Duck DNS: Facilita la gestión de DNS dinámicos, permitiendo que los usuarios mantengan acceso remoto a sus redes de manera segura y confiable, sin necesidad de una IP estática.
- Suricata: Este sistema IDS/IPS es crucial para la detección y prevención de intrusiones en tiempo real, permitiendo monitorizar la red e identificar y responder a posibles amenazas dentro de la red. Por otra parte, al ser un software "*open source*", está respaldado por una gran comunidad.
- Elastic Stack (ElasticSearch, Logstash, Kibana): Estas herramientas proporcionan una poderosa solución de monitoreo mediante *dashboards* y análisis, integrándose con suricata y permitiendo a los usuarios visualizar y analizar los datos de la red, identificar patrones sospechosos y mejorar continuamente la seguridad.

- Docker: Docker permite desplegar todas estas herramientas en entornos aislados, haciendo que no interfieran entre ellas, y brindando la posibilidad de una instalación sencilla y rápida para el resto de los usuarios.

2.8.3. Ventajas de Utilizar Raspberry Pi

El uso de Raspberry Pi como plataforma central en este proyecto ofrece varias ventajas significativas:

- Costo Efectivo: Raspberry Pi es una opción económica que permite a los usuarios construir soluciones de seguridad avanzadas sin una gran inversión de capital.
- Versatilidad: Gracias a su capacidad para ejecutar diversos sistemas operativos y su compatibilidad con una amplia gama de software de código abierto, Raspberry Pi se adapta a múltiples necesidades de red y seguridad.
- Facilidad de Implementación: Raspberry Pi es fácil de configurar, y con el uso de Docker, la replicación del entorno configurado es sencilla, lo que facilita a otros usuarios la implementación de la misma solución sin necesidad de profundos conocimientos técnicos.
- Bajo Consumo Energético: A diferencia de otros dispositivos más grandes y costosos, Raspberry Pi consume muy poca energía, lo que la hace ideal para un uso continuo en el hogar.

Estas características convierten a Raspberry Pi en la plataforma perfecta para implementar una solución de seguridad doméstica accesible y escalable.

2.8.4. Necesidad de Conciencia y Educación en Seguridad

Aunque la tecnología es una parte fundamental en la protección de redes domésticas, la conciencia y la educación del usuario final son igual de cruciales. Sin un conocimiento básico de las mejores prácticas de seguridad, incluso las soluciones técnicas más avanzadas pueden quedar comprometidas.

Es vital que los usuarios comprendan conceptos básicos como la importancia de mantener los dispositivos actualizados, usar contraseñas seguras, evitar redes públicas no confiables y

reconocer intentos de phishing. La educación continua en ciberseguridad debe ser promovida para que los usuarios puedan aprovechar al máximo las herramientas de protección que se les proporcionan.

2.8.5. Desafíos Identificados y Futuras Direcciones

A pesar de los avances y las herramientas disponibles, persisten varios desafíos en la seguridad de las redes domésticas:

- Gestión de Dispositivos IoT: Muchos dispositivos IoT aún carecen de medidas de seguridad adecuadas y pueden ser puntos de entrada para atacantes.
- Actualización Continua: Mantener todas las herramientas y dispositivos actualizados para enfrentar nuevas amenazas es un desafío constante. Aunque pueda parecer una tarea sencilla, muchas personas tienden a posponer las actualizaciones debido a la incomodidad o el tiempo que el proceso puede implicar. Sin embargo, retrasar estas actualizaciones puede dejar la red vulnerable a ataques que podrían haberse evitado con versiones más recientes del software.
- Automatización: Hay una creciente necesidad de soluciones que ofrezcan automatización en la detección y respuesta a amenazas, especialmente para usuarios sin experiencia técnica.
- Interfaces de Usuario: Desarrollar interfaces más intuitivas y fáciles de usar es esencial para que más usuarios puedan gestionar efectivamente la seguridad de sus redes.

Futuras investigaciones podrían centrarse en integrar inteligencia artificial para mejorar la detección de amenazas y en desarrollar plataformas más accesibles que permitan a los usuarios gestionar su seguridad con mayor facilidad.

2.8.6. Conclusión General

En conclusión, aunque la seguridad en redes domésticas ha mejorado considerablemente en los últimos años, sigue siendo un área en constante evolución que requiere atención continua. El proyecto propuesto ofrece una solución integral y accesible, basada en tecnologías

probadas y herramientas flexibles que pueden adaptarse a las necesidades cambiantes de los usuarios.

Esta solución no solo protege eficazmente los datos y dispositivos conectados, sino que también proporciona un marco replicable y escalable que puede ser utilizado por cualquier usuario, independientemente de su nivel técnico. Sin embargo, para que estas soluciones sean verdaderamente efectivas, deben ir acompañadas de una mayor conciencia y educación en ciberseguridad por parte de los usuarios.

Este proyecto, por tanto, no solo aborda las necesidades técnicas de seguridad, sino que también promueve un enfoque más amplio de concienciación y responsabilidad en el uso de la tecnología en el hogar.

3. OBJETIVOS CONCRETOS Y METODOLOGÍA DE TRABAJO

3.1. OBJETIVO GENERAL

El objetivo principal de este proyecto es diseñar e implementar una solución de seguridad completa para redes domésticas, usando una Raspberry Pi como la plataforma central. La solución integrará diversas herramientas mediante Docker, lo que permitirá crear un entorno de red seguro que sea fácil de exportar, replicar y escalar según las necesidades.

El proyecto está enfocado en mejorar la seguridad y la privacidad de los usuarios, proporcionando una guía de configuración sencilla que no requiera conocimientos técnicos avanzados. Esto permitirá a los usuarios adaptar la solución a diferentes necesidades y configuraciones de red. Gracias al uso de Docker, la solución podrá actualizarse y ampliarse fácilmente con nuevas herramientas de seguridad, asegurando así una protección continua frente a las últimas ciberamenazas. De esta manera, los usuarios podrán proteger eficazmente sus datos y dispositivos conectados, estableciendo una red doméstica más segura y privada.

3.2. OBJETIVOS ESPECÍFICOS

Los objetivos intermedios para alcanzar el objetivo principal de este trabajo son:

3.2.1. Investigar los riesgos a los que están expuestos los usuarios de internet dentro de su red doméstica.

3.2.2. Investigar la arquitectura de una red doméstica, para comprender su estructura y funcionamiento.

3.2.3. Investigación de posibles alternativas a Raspberry Pi para la realización de este proyecto.

3.2.4. Buscar y crear una guía de buenas prácticas para una configuración segura de la red doméstica.

3.2.5. Desarrollar una arquitectura de seguridad utilizando una Raspberry Pi como plataforma central.

3.2.6. Diseñar una solución completa que integre tecnologías avanzadas de seguridad como las siguientes:

- Filtrado DNS.
- VPN.
- Sistemas de detección de intrusiones (IDS).
- Sistemas de prevención de intrusiones (IPS).
- *Dashboard* de monitorización.

3.2.7. Llevar a cabo toda esta configuración mediante el uso de contenedores Docker que puedan replicarse fácilmente por usuarios sin conocimientos técnicos.

3.2.8. Redactar una memoria de TFM correcta que refleje todo el trabajo de investigación realizado, así como los avances obtenidos.

3.3. METODOLOGÍA DEL TRABAJO

Para este Trabajo Final de Máster, la metodología que se ha llevado a cabo ha seguido los siguientes pasos:

1. En primer lugar, el problema identificado era algo de lo que era consciente desde antes de comenzar el máster en ciberseguridad. Tras iniciar dicho máster y adentrarme poco a poco en el mundo de la ciberseguridad, empecé a ver más claro cómo afrontar este problema de la forma más cercana al usuario final. Modificando la propia red doméstica agregándole un plus de seguridad a ella.
2. Tras esta identificación del problema, se comenzó un proceso de análisis de las tecnologías que mejor podrían complementarse en este desarrollo, y cómo integrarlas entre ellas con el menor impacto posible. En ese momento se decidió utilizar Docker para dicha integración.
3. Por otra parte, se establecieron los requisitos que debía tener el hardware a utilizar. Que fuese económico, y fácil de conseguir. Además, debía poder ser capaz de permanecer constantemente encendido, por lo que, tras analizar diversas opciones, se escogió la Raspberry Pi 5 como núcleo del proyecto.

4. Una vez identificado el software a utilizar, y habiendo adquirido el hardware, se comenzó un proceso de investigación individual de cada una de las tecnologías a utilizar. El objetivo era aprender todo lo posible sobre dichas herramientas de cara al comienzo de la implementación de la solución propuesta, y de la redacción de la memoria del trabajo.
5. Finalmente, tras haber documentado la parte de software y hardware, comenzó el proceso de configuración e implementación del entorno. Una vez configurado, se pasó a realizar una batería de pruebas para testear su correcto funcionamiento.
6. Con todos los resultados obtenidos, se terminó de redactar esta memoria final de este Trabajo de Fin de Máster.

4. BUENAS PRÁCTICAS DE CONFIGURACIÓN DE LA RED DOMÉSTICA

4.1. GUÍA DE BUENAS PRÁCTICAS

Además de llevar a cabo el desarrollo de una solución que aumente la seguridad de la red doméstica, uno de los objetivos de este trabajo es proporcionar una guía de buenas prácticas que pueden aplicarse en cualquier red, mediante unas simples modificaciones para configurarla de manera segura. Esta guía recoge los siguientes puntos (Méndez Colmenares, 2023) (Vallejo, 2022) (Kaspersky Lab, 2024):

- Utilizar WPA2/WPA3: Configura la red inalámbrica con WPA2 o, preferiblemente, WPA3, que es el estándar más reciente y ofrece mayor seguridad frente a ataques.
- Establecer contraseñas fuertes: Utiliza contraseñas complejas y únicas tanto para la red Wi-Fi como para el acceso a la administración del rúter. Evita contraseñas comunes o fáciles de adivinar.
- Utiliza un gestor de contraseñas: Para conseguir contraseñas fuertes y únicas, lo más recomendable es el uso de un gestor de contraseñas. Generalmente además de poder guardar ahí todas tus contraseñas, también podrás generarlas de manera segura, lo que hará que puedas crear claves muy seguras. Los hay tanto gratuitos y como de pago.
- Cambiar el SSID y las contraseñas predeterminadas: Modifica el nombre de la red (SSID) y las contraseñas que vienen por defecto en el rúter para evitar que los atacantes identifiquen fácilmente el modelo de tu rúter y sus posibles vulnerabilidades.
- Usar un firewall eficaz: Asegúrate de que el firewall del rúter esté activado. Esto añade una capa de protección al controlar y filtrar el tráfico entrante y saliente de la red. En algunos casos el rúter permite seleccionar el nivel de restricción del firewall. Por lo general el nivel predeterminado es una buena opción.
- Mantener el firmware y software actualizados: Actualiza regularmente el firmware del rúter y el software de todos los dispositivos conectados. Las actualizaciones suelen incluir parches de seguridad que protegen contra nuevas amenazas.

- Filtrado de direcciones MAC: Cada dispositivo posee una dirección MAC única. Establezca un filtrado de las direcciones MAC de los dispositivos que desee que se conecten al rúter, así solamente podrán conectarse aquellos dispositivos permitidos.
- Crear una red de invitados: Configura una red Wi-Fi separada para los invitados. De esta forma, tus dispositivos principales quedan aislados y protegidos de posibles amenazas que puedan provenir de los dispositivos de los invitados. Generalmente los rúteres actuales permiten habilitar de manera sencilla esta configuración.
- Desactivar WPS y UPnP: Desactiva las funciones de WPS (Wi-Fi Protected Setup) y UPnP (Universal Plug and Play), ya que pueden ser explotadas por atacantes para acceder a tu red.
- Uso de un bloqueador de DNS: Implementa herramientas como Pi-hole (siguiendo la configuración explicada en este trabajo) para bloquear anuncios y sitios web maliciosos a nivel de red, mejorando la privacidad y seguridad de todos los dispositivos conectados.
- Segmentación de la red: Si el rúter utilizado lo permite, crea subredes para aislar dispositivos IoT o aquellos que sean menos seguros del resto de la red. Esto minimiza el riesgo de que un dispositivo comprometido afecte a toda la red.
- Autenticación multifactor (MFA): Activa MFA en los dispositivos y cuentas que lo permitan, añadiendo una capa adicional de seguridad que dificulta el acceso no autorizado.
- Desactivar servicios innecesarios: Desactiva servicios como Telnet, SSH, o cualquier otro que no estés utilizando activamente, para reducir los vectores de ataque.
- Monitorización de la red: Utiliza sistemas de detección y prevención de intrusiones (IDS/IPS) como Suricata para monitorizar el tráfico de la red y detectar posibles amenazas.

Por otra parte, se proporciona también una lista de consejos de seguridad a aplicar cuando se esté fuera de la red doméstica:

- Uso de VPN en redes públicas: Al conectarte a redes Wi-Fi públicas, utiliza una VPN (Red Privada Virtual) para cifrar tus datos y proteger tu privacidad.

- Preferencia por datos móviles: Siempre que sea posible, utiliza los datos móviles en lugar de conectarte a redes públicas, que suelen ser menos seguras.
- Verificación de conexiones: Asegúrate de que las redes públicas a las que te conectes sean legítimas y evita redes con nombres sospechosos o sin contraseña.
- Verificación de HTTPS: Al navegar en redes públicas, verifica que las conexiones a sitios web sean seguras (https), lo que garantiza que la información transmitida está cifrada.
- Configurar dispositivos para olvidar redes públicas automáticamente: Configura tus dispositivos para que no se conecten automáticamente a redes públicas en el futuro, evitando así conexiones no deseadas.
- Activación de Firewalls Personales: Si están disponibles y es posible, activa firewalls en dispositivos móviles y portátiles cuando te conectes a redes externas para añadir una capa adicional de seguridad.

4.2. EDUCACIÓN Y CONCIENCIACIÓN DEL USUARIO

La seguridad de la red también depende del comportamiento de quienes la utilizan. Es crucial que todos los miembros del hogar estén informados sobre buenas prácticas de ciberseguridad, como la identificación de correos electrónicos de phishing, la importancia de no compartir contraseñas, y el uso adecuado de dispositivos conectados. Una red segura empieza con usuarios informados y conscientes de las amenazas a las que pueden enfrentarse.

5. DESARROLLO ESPECÍFICO DE LA CONTRIBUCIÓN

A continuación, en este capítulo se exponen los detalles técnicos de la solución propuesta en este trabajo. En primer lugar, se recordarán los requisitos ya especificados anteriormente, luego se explicará de forma técnica las herramientas utilizadas, así como su configuración e integración de estas, para el correcto funcionamiento en conjunto de todas ellas, y finalmente, se expondrán los resultados de las pruebas sobre dichas herramientas.

5.1. IDENTIFICACIÓN DE REQUISITOS

Este proyecto nace de la necesidad de mejorar la seguridad en las redes domésticas, que cada vez están más expuestas a diversas amenazas digitales. Con la proliferación de dispositivos conectados en nuestros hogares, como ordenadores, smartphones, dispositivos IoT y otros, proteger la red se ha convertido en una prioridad para evitar accesos no autorizados, ataques de *malware* y otras vulnerabilidades.

Entre los principales requisitos, se encuentran:

1. Protección completa de la red doméstica: Crear una solución que cubra tanto la prevención como la detección de intrusiones en la red doméstica
2. Monitorización y detección en tiempo real: Incorporar una herramienta que permita monitorizar el tráfico en tiempo real, para detectar posibles amenazas de forma inmediata.
3. Facilidad de uso e implementación: Se busca una solución fácil y rápida de implementar.
4. Flexibilidad y escalabilidad: Diseñar una arquitectura flexible que pueda adaptarse a diferentes configuraciones de red y que sea escalable para soportar la adición de nuevos dispositivos y servicios en el futuro.
5. Uso de software de código abierto: Con el objetivo de reducir el coste de esta implementación, se utilizará software que no requiera un coste económico.
6. Coste reducido de la implementación: La solución debe estar optimizada para funcionar en hardware económico y con recursos limitados, como la Raspberry Pi.

5.2. DESCRIPCIÓN DE LAS HERRAMIENTA SOFTWARE UTILIZADAS

A continuación, se muestra el software utilizado, así como su instalación y puesta en funcionamiento.

5.2.1. Raspberry Pi OS

El sistema operativo elegido para este proyecto ha sido el sistema operativo nativo de Raspberry, llamado Raspberry Pi OS. Para la instalación de este, se siguen las siguientes instrucciones (detalladas en su página oficial (Raspberry Pi, 2024)):

En primer lugar, se descarga el software de creación de la imagen desde el sitio oficial de Raspberry Pi: https://www.raspberrypi.com/software/

Una vez instalado, se introduce el dispositivo donde se almacenará el SO de la Raspberry Pi (una tarjeta microSD comúnmente), se ejecuta el programa y seleccionamos las características deseadas. Para este proyecto se han escogido las siguientes:

Figura 3: Raspberry Pi Imager. Fuente: Elaboración propia.

Tras este proceso, se introduce la microSD en la Raspberry Pi, y ya estaría listo para iniciarla y hacer la configuración inicial del sistema operativo como cualquier otro. Se recomienda dar a la Raspberry Pi una dirección IP estática, para así conocer y asegurar en todo momento los datos de su conexión.

5.2.2. Docker y Docker Compose

El núcleo de este trabajo funciona mediante contenedores Docker (Docker Inc, Docker, 2024). Y su configuración se realiza mediante la herramienta Docker Compose, la cual permite orquestar y configurar todos los contenedores juntos, de una forma cómoda (Docker Inc, Docker Compose overview, 2024).

Para esta configuración utilizaremos los siguientes comandos (DanieloTech, 2024) (Docker Inc, Install Docker Engine on Debian, 2024):

- Instalar Docker y Docker Compose:

```
$ sudo curl -fsSL https://get.docker.com/ -o get-docker.sh
$ sudo sh get-docker.sh
```

- Añadir nuestro usuario al grupo Docker:

```
$ sudo usermod -aG docker ${USER}
```

- Reiniciar sistema:

```
$ sudo reboot
```

Tras esta configuración, Docker y Docker Compose quedarán instalados y listos para funcionar.

5.2.3. Portainer.io

Algo de lo que todavía no se ha hablado, pero que será de mucha utilidad para la gestión de los contenedores Docker, es el uso del gestor de contenedores “portainer.io” (Portainer, Effortless Container Management for Docker and Kubernetes, 2024). Esta herramienta ofrece una interfaz gráfica de Docker y sus contenedores, permitiendo al usuario ver de una forma más amigable el estado de todos los contenedores, así como sus logs. También permite la gestión de dichos contenedores de manera gráfica, pausarlos, pararlos, reiniciarlos, así como eliminarlos. Para su instalación se ejecutarán los siguientes comandos (DanieloTech, 2024) (Portainer, Install Portainer CE with Docker on Linux, 2024):

- Crear volumen Docker que contendrá los datos gestionados por el servidor Portainer:

```
$ docker volume create portainer_data
```

- Descargar e instalar contenedor Portainer:

```
$ docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data
portainer/portainer-ce:latest
```

- Acceso a Portainer:

Para el acceso a la interfaz gráfica de Portainer, se debe que escribir la siguiente dirección en el navegador: http://0.0.0.0:9443/ en el caso de que se esté accediendo desde la misma Raspberry Pi. En el caso de que se acceda desde otro dispositivo conectado a la red habrá que poner en el navegador la dirección: https://[IP de la Raspberry Pi]:9443. Tras esta configuración, el entorno gráfico para la gestión de los contenedores está listo. A continuación, se muestra una imagen de cómo se ve dicho entorno:

Figura 4: Portainer Home. Fuente: Elaboración propia.

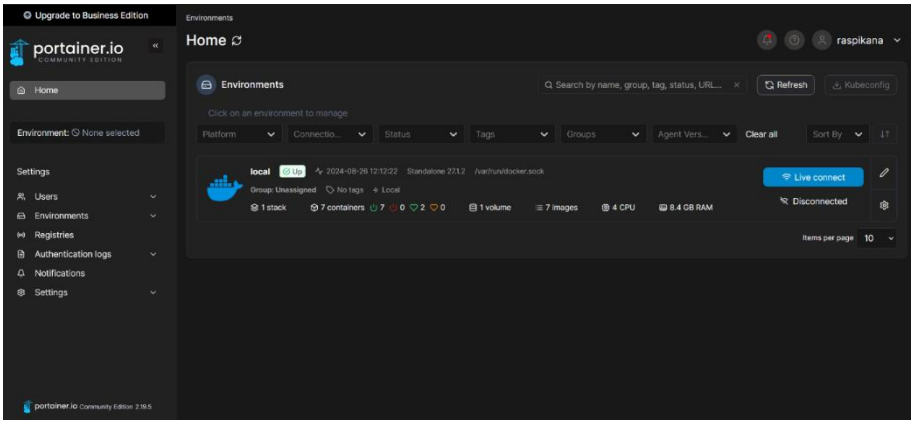


Figura 5: Portainer Dashboard. Fuente: Elaboración propia.

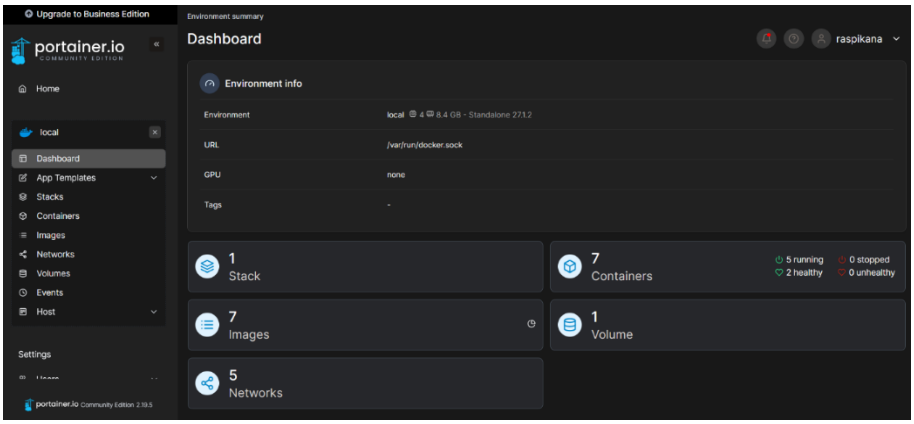
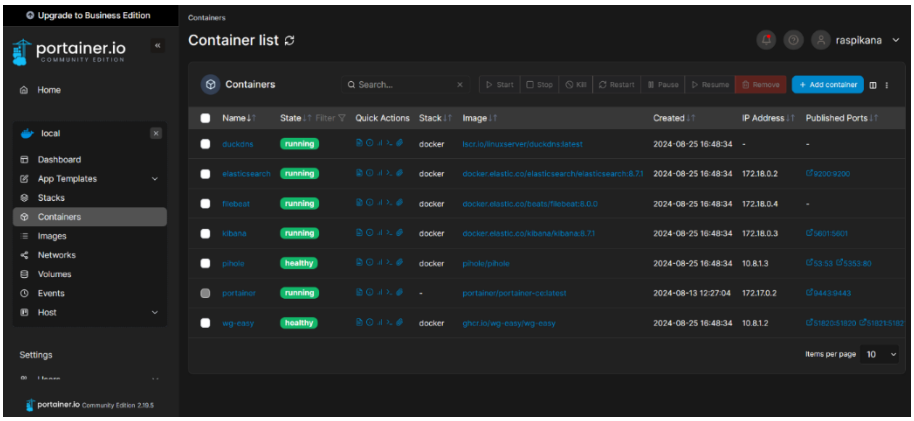


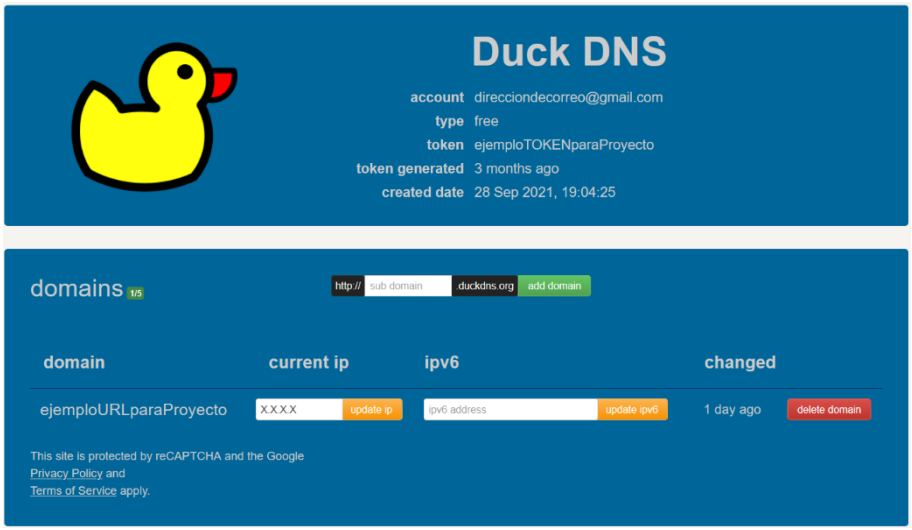
Figura 6: Portainer Containers. Fuente: Elaboración propia.



5.2.4. DuckDNS

DuckDNS es el servicio de DNS dinámico (DDNS) escogido para este proyecto por sus características, ser open source, y su facilidad de configuración. Para llevar a cabo dicha configuración será necesario registrarse en su página oficial https://www.duckdns.org/, una vez creada la cuenta, se debe crear un dominio el apartado *sub domain* y a continuación pulsar en *add domain*. Para este proyecto se ha escogido el nombre "ejemploURLparaProyecto".

Figura 7: DuckDNS. Fuente: Elaboración propia.



Una vez realizada esta configuración, se dispone de un servicio DDNS activo. El siguiente paso, es instalar el cliente de dicho servicio en la Raspberry Pi, para que, en caso de que la IP pública del hogar cambie, la Raspberry Pi actualice automáticamente dicha IP en el servidor DDNS.

Para conseguir esta configuración, se ha utilizado la imagen Docker del autor linuxserver.io (linuxserver.io, 2024). Su código se ha agregado al archivo principal *compose.yaml*, contenido en el Anexo B de este proyecto, el cual se encuentra en el repositorio de GitHub que se cita en el apartado 5.2.8. Por lo tanto, su despliegue será automático en el momento en el que se ejecute el comando para desplegar todos los contenedores.

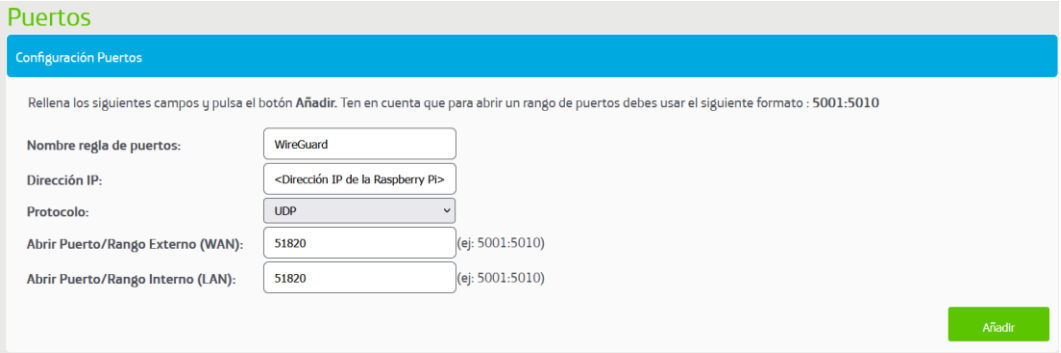
5.2.5. WireGuard

Este servicio, junto con Suricata y Pi-hole son los más importantes de este proyecto. WireGuard es un protocolo de comunicación y una herramienta para crear redes privadas virtuales (VPN). Se destaca por su simplicidad, alta seguridad, rendimiento y eficiencia, lo que lo convierte en una alternativa moderna a protocolos VPN más tradicionales como IPsec y OpenVPN (WireGuard®, 2022). Gracias a la implementación de WireGuard, se consigue poder tener una conexión segura mediante un túnel VPN cifrado cuando el usuario se encuentra fuera de su hogar, y conectado a redes poco seguras, como redes públicas.

Para su implementación, se ha utilizado la imagen Docker del proyecto WireGuard Easy (WireGuard Easy, 2024), en la que se integra WireGuard con Pi-hole, y así se consigue que cualquier dispositivo conectado a la VPN obtenga las funciones de Pi-hole sin necesidad de realizar ninguna configuración más.

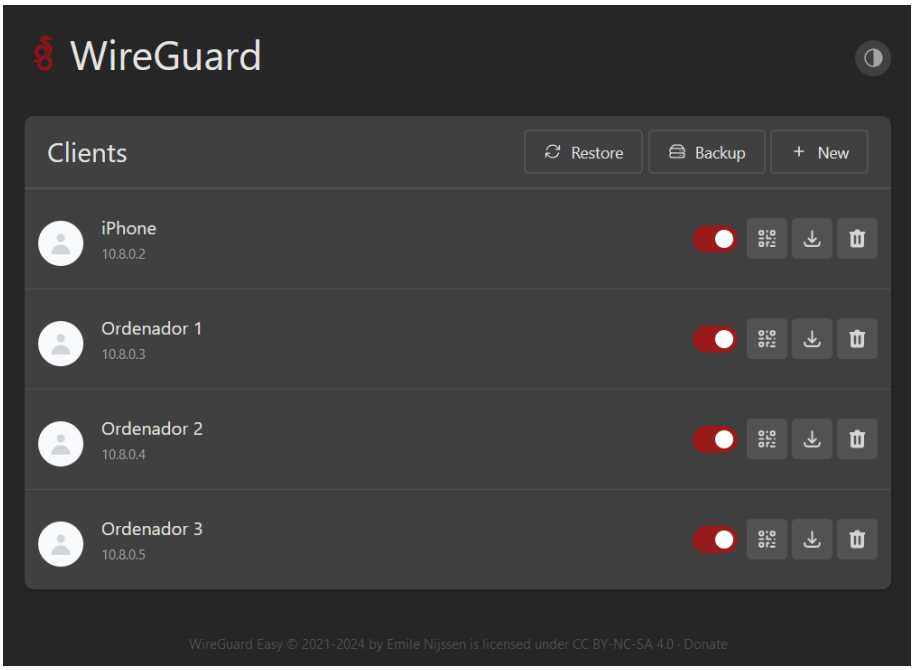
Su código se ha agregado al archivo principal *compose.yaml*, mostrado en el Anexo B de este proyecto, el cual se encuentra en el repositorio de GitHub que se cita en el apartado 5.2.8. Por lo tanto, su despliegue será automático en el momento en el que se ejecute el comando para desplegar todos los contenedores. Solamente quedaría configurar el rúter doméstico para abrir el puerto donde se encontrará escuchando WireGuard. Para ello se ingresa en la configuración del rúter mediante la dirección http://192.168.1.1/, se ingresa en el apartado “puertos” y se crea una configuración como la siguiente:

Figura 8: Configuración de Puerto en Rúter. Fuente: Elaboración propia.



Para el acceso a la interfaz gráfica de WireGuard, se escribe la siguiente dirección en el navegador: http://0.0.0.0:51821/ en el caso de que se esté accediendo desde la misma Raspberry Pi. En el caso de que se acceda desde otro dispositivo conectado a la red se pone en el navegador la dirección: https://[IP de la Raspberry Pi]: 51821. Su interfaz web es:

Figura 9: WireGuard Dashboard I. Fuente: Elaboración propia.



Este proyecto (WireGuard Easy, 2024) tiene la particularidad de que es especialmente sencillo para el usuario dar de alta nuevos dispositivos que quieran conectarse a la VPN. Solamente se necesita instalar en ellos el cliente de WireGuard desde la tienda de aplicaciones deseada, y, en la interfaz del servidor, agregar mediante el botón “+ New” un nuevo dispositivo. Tras pulsar sobre ese botón, se proporciona el nombre deseado, y ya queda el perfil creado correctamente. El siguiente paso es simplemente descargar el archivo de configuración

mediante el botón de descarga que aparece junto al perfil (o si es un teléfono móvil, generando el código QR es más rápido) e importarlo en la aplicación cliente del dispositivo deseado.

Tras esta configuración, el dispositivo está listo para conectarse mediante el túnel VPN y obtener salida a internet a través de una red segura, y utilizando el filtrado DNS que proporciona Pi-hole.

5.2.6. Pi-hole

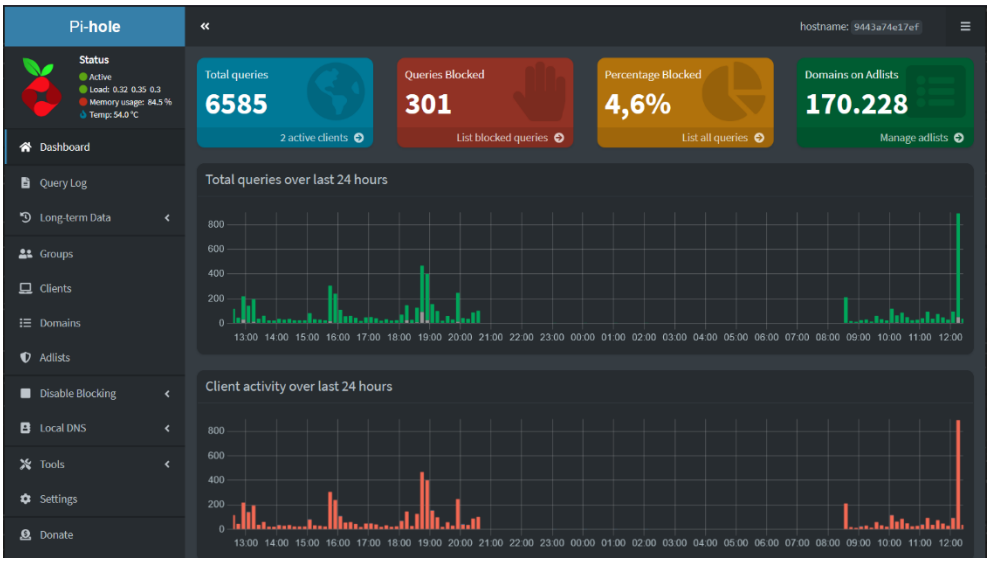
Pi-hole, tal y como se ha descrito anteriormente, también es un elemento clave de la configuración de este proyecto, y es que se trata de un servidor DNS que ofrece un control riguroso sobre las direcciones IP que debe resolver al consultar una página web. Su lógica se basa en el uso de *White Lists* y *Black Lists*. Por defecto trae una *Black List* muy completa y que para el uso común es más que suficiente, aunque proporciona mucha más funcionalidad, desde el poder agregar más listas, hasta que trabaje también como servidor DHCP.

Para su configuración se ha utilizado la imagen oficial que proporcionan los creadores del software (Pi-hole®, pihole/pihole, 2024), pero integrada con el proyecto WireGuard Easy (WireGuard Easy, 2024), para que además de funcionar en la red local, también funcione para aquellos dispositivos que se conecten a la VPN. Su código se ha agregado al archivo principal *compose.yaml*, contenido en el Anexo B de este proyecto, el cual se encuentra en el repositorio de GitHub que se cita en el apartado 5.2.8. Por lo tanto, su despliegue será automático en el momento en el que se ejecute el comando para desplegar todos los contenedores.

Para el acceso a la interfaz gráfica de Pi-hole, habrá que escribir la siguiente dirección en el navegador: http://0.0.0.0:5353/admin/login.php en el caso de que se esté accediendo desde la misma Raspberry Pi. En el caso de que se acceda desde otro dispositivo conectado a la red habrá que poner en el navegador la dirección: http://[IP de la Raspberry Pi]:5353/admin/login.php.

Su interfaz web es la siguiente:

Figura 10: Pi-hole Dashboard. Fuente: Elaboración propia.



Una vez activo y funcionando el contenedor Docker que alberga Pi-hole, se puede establecer la Raspberry PI como servidor DNS por defecto de toda la red local, mediante la configuración individual de cada dispositivo, o, para hacerlo más escalable, mediante la configuración del rúter doméstico, cambiando la dirección IP de los servidores DNS. Se recomienda prestar especial atención en este segundo caso puesto que, en caso de error quedaría toda la red doméstica sin conexión a internet.

5.2.7. ElasticSearch + Kibana + Filebeat

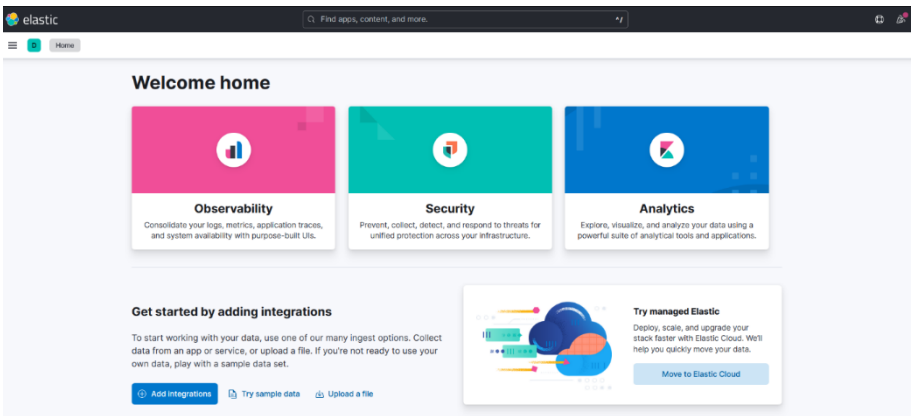
Como ya se ha mencionado en el Estado del arte, en el apartado 2.7.5, el uso de Elastic Stack permite una monitorización continua y dinámica de todo el tráfico que detecta Suricata. ElasticSearch trabaja como motor de búsqueda y de almacenamiento de los datos. Kibana es la interfaz gráfica de esta configuración, permitiendo crear *dashboards* muy completos y personalizables. Y, por último, Filebeat es el agente que se encarga de la recolección de los logs. Esta configuración sigue estando implementada mediante contenedores Docker. Su código se ha agregado al archivo principal *compose.yaml*, contenido en el Anexo B de este proyecto, el cual se encuentra en el repositorio de GitHub que se cita en el apartado 5.2.8. Por lo tanto, su despliegue será automático en el momento en el que se ejecute el comando para desplegar todos los contenedores.

Cabe destacar una pequeña peculiaridad, y es que, como se verá en los apartados 5.2.9 y 5.2.10, donde se aclara y matiza esta implementación, se deben hacer dos configuraciones con Filebeat, una dentro de Docker, y otra fuera de Docker. Esta, es la instancia de Filebeat dentro de Docker, encargada de recoger logs y enviárselos a ElasticSearch.

La segunda instancia de Filebeat, fuera de Docker, explicada en el apartado 5.2.10, es la encargada de recoger los logs de Suricata (el cual está instalado fuera de Docker como se explica en el punto 5.2.9), y crear un *pipeline* con la primera instancia de Filebeat. Con esto se crea la comunicación entre ellas, y permite el envío de datos entre Suricata y ElasticSearch para su procesamiento y visualización en el *dashboard* de Kibana.

Tras hacer el despliegue de los contenedores Docker, ElasticSearch, Kibana y Filebeat tardarán unos minutos en estar 100% operativos, tras ese tiempo, ya se podrá acceder a la interfaz web de Kibana mediante la siguiente dirección en el navegador: http://192.168.1.200:5601/ en el caso de que se esté accediendo desde la misma Raspberry Pi. En el caso de que se acceda desde otro dispositivo conectado a la red habrá que poner en el navegador la dirección: http://[IP de la Raspberry Pi]:5601/. Su interfaz web es la siguiente:

Figura 11: Kibana Dashboard. Fuente: Elaboración propia.



Desde ella se podrán hacer las configuraciones deseadas, así como agregar los *dashboards* que se necesiten. Es muy editable. Cómo crear el *dashboard* para monitorizar Suricata se explica en el apartado 5.2.10.

5.2.8. GitHub y despliegue Docker

Para implementar las herramientas contenidas en Docker explicadas hasta este momento, simplemente habrá que ejecutar los siguientes comandos:

```
$ git clone https://github.com/miguelsp99/Fortificacion-y-Privacidad-en-Redes-Domesticas-con-Raspberry-Pi.git
$ cd Fortificacion-y-Privacidad-en-Redes-Domesticas-con-Raspberry-Pi
$ docker compose up -d
```

Los cuales clonan el repositorio de GitHub, mueven dentro de la carpeta clonada, y levantan los contenedores. Es muy importante que antes del 3º paso, se edite el archivo compose.yaml y se cambie la configuración señalada mediante comentarios, por los valores que le apliquen a cada usuario.

5.2.9. Suricata

Hasta ahora, toda la configuración realizada ha sido sobre Docker con el objetivo de ofrecer una solución sencilla de replicar para el usuario final, pero, durante la implementación de Suricata en Docker, se encontraron problemas en el funcionamiento en modo IPS, y en la conexión con Filebeat para poder enviar los logs a ElasticSearch. Por este motivo, se decidió instalar Suricata en la máquina host y no en un contenedor Docker, y proporcionar al usuario una guía sencilla de instalación para poder lograr el objetivo de ser una implementación simple y fácil. Por lo tanto, a continuación, se detalla dicha guía.

5.2.9.1. Instalación de suricata en modo IDS:

- Actualizar los repositorios:

```
$ sudo apt update
```

- Instalar suricata desde los repositorios oficiales de Debian. No será la versión más nueva, pero sí la más estable:

```
$ sudo apt install suricata
```

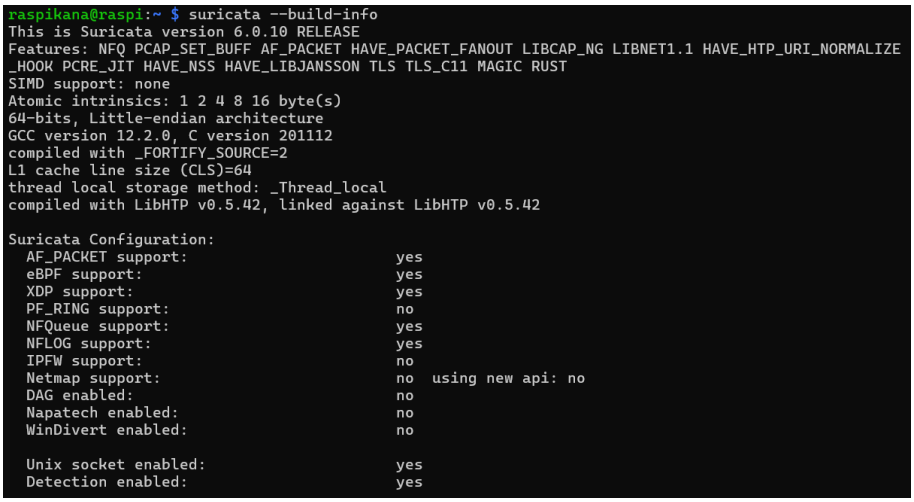
- Comprobar la versión de suricata:

```
$ suricata –build-info
```

El resultado de este comando será lo siguiente:

```
This is Suricata version 6.0.10 RELEASE
Features: NFQ PCAP_SET_BUFF AF_PACKET HAVE_PACKET_FANOUT LIBCAP_NG LIBNET1.1
HAVE_HTP_URI_NORMALIZE_HOOK PCRE_JIT HAVE_NSS HAVE_LIBJANSSON TLS TLS_C11 MAGIC
RUST
SIMD support: none
Atomic intrinsics: 1 2 4 8 16 byte(s)
64-bits, Little-endian architecture
GCC version 12.2.0, C version 201112
compiled with _FORTIFY_SOURCE=2
L1 cache line size (CLS)=64
thread local storage method: _Thread_local
compiled with LibHTP v0.5.42, linked against LibHTP v0.5.42
Suricata Configuration:
  AF_PACKET support:                       yes
  eBPF support:                            yes
  XDP support:                             yes
  PF_RING support:                         no
  NFQueue support:                         yes
```

Figura 12: Versión de Suricata. Fuente: Elaboración propia.

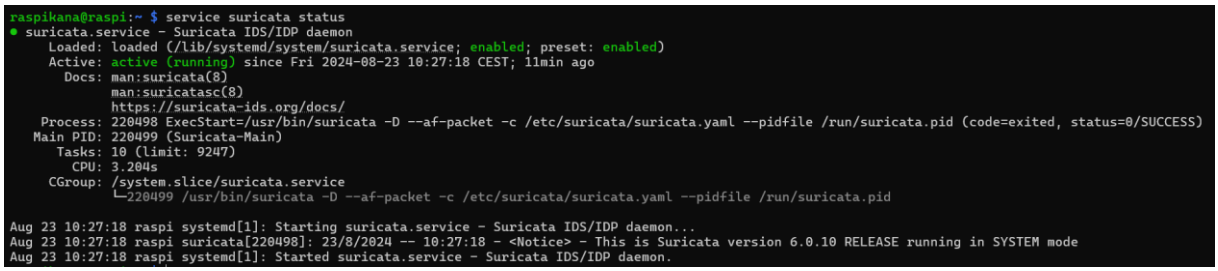


Es especialmente importante que se compruebe que “NFQueue support” tiene como resultado “yes”, ya que esto significa que soporta el modo de configuración IPS.

- Comprobar el estado de suricata:

```
$ service suricata status
```

Figura 13: Estado de Suricata. Fuente: Elaboración propia.



Tras la instalación y activación, se observa cómo el estado queda activo.

- Conocer ruta en la que se encuentra suricata:

```
$ whereis suricata
```

Se obtiene la siguiente salida:

- /usr/bin/suricata
- /usr/lib/suricata
- **/etc/suricata**
- /usr/share/man/man1/suricata.1.gz

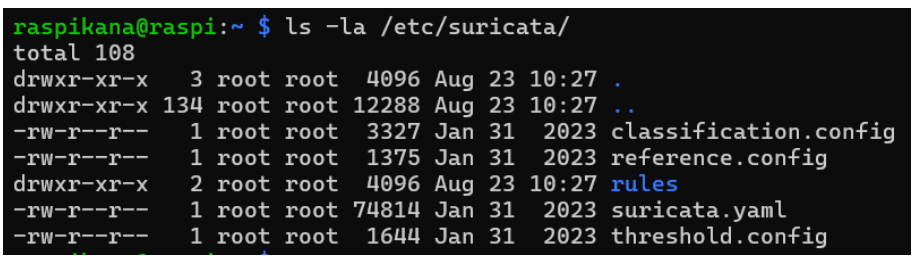
Los ficheros de configuración se encuentran en la ruta **/etc/suricata**.

- Listar ficheros de configuración de Suricata:

```
$ ls -la /etc/suricata/
```

Se observan los siguientes ficheros:

Figura 14: Ficheros de configuración de Suricata. Fuente: Elaboración propia.



El fichero ***suricata.yaml*** es el fichero de configuración principal de suricata, en él se harán las modificaciones oportunas para que funcione correctamente. Para ello, se ejecuta el siguiente comando:

```
$ sudo nano /etc/suricata/suricata.yaml
```

Y las configuraciones a realizar serán las siguientes:

1. En primer lugar, se establece la HOME_NET, en ella se define la IP de la red local, y aquellas subredes que se han creado en Docker, para que Suricata interprete todo como una sola red. La configuración en el caso de este proyecto queda así:

Figura 15: HOME_NET. Fuente: Elaboración propia.

```
##
## Step 1: Inform Suricata about your network
##

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.17.0.0/12,172.18.0.0/12]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    EXTERNAL_NET: "!$HOME_NET"
    #EXTERNAL_NET: "any"
```

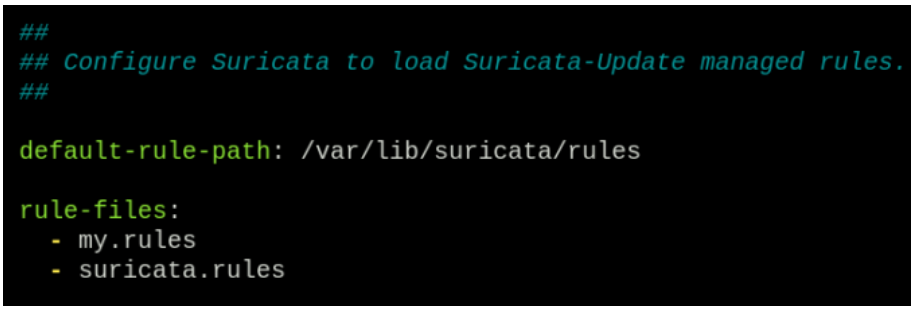
2. El segundo paso es definir la interfaz de red por la que Suricata va a capturar el tráfico. Para este proyecto, la Raspberry Pi se ha conectado mediante cable para asegurar una conexión estable y constante, por lo que se ha elegido la interfaz eth0. (Para saber el nombre de la interfaz se ejecutar el comando `$ ifconfig`).

Figura 16: Interfaz de captura de Suricata. Fuente: Elaboración propia.

```
##
## Step 3: Configure common capture settings
##
## See "Advanced Capture Options" below for more options, including Netmap
## and PF_RING.
##

# Linux high speed capture support
af-packet:
  - interface: eth0
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99
```

3. Finalmente, el último paso será establecer la ruta y los ficheros de reglas que Suricata va a consultar para detectar el tráfico maligno. Esta es la principal funcionalidad que tiene Suricata, todo el tráfico que detecta, lo compara contra una base de datos de firmas conocidas, y si coincide con alguna de ellas, lanza una alerta, o la acción que tenga configurada dicha regla. Esta base de datos se configura en este paso. Se puede agregar tantos ficheros de reglas como se desee, pero hay que prestar especial atención, ya que, en caso de repetirse alguna regla en más de un fichero, aparecerá un error y Suricata dejará de funcionar.

Figura 17: Configuración reglas de Suricata. Fuente: Elaboración propia.



La explicación de las reglas de Suricata se realiza en el apartado 5.2.9.3.

Para este proyecto se utilizarán únicamente dos ficheros de reglas:

- my.rules: Fichero que se ha creado para la realización de pruebas y control del funcionamiento de Suricata.
- suricata.rules: En este fichero se encuentra la base de datos perteneciente al proyecto "Proofpoint Emerging Threats Rules" (Proofpoint Inc, 2024). Este es un proyecto oficial, que actualiza periódicamente el fichero de reglas para mantener una protección adecuada de la red mediante una base de datos de firmas completamente actualizada. Este fichero NO viene por defecto con la instalación de Suricata, para conseguirlo, se deberá ejecutar el siguiente comando:

```
$ sudo suricata-update
```

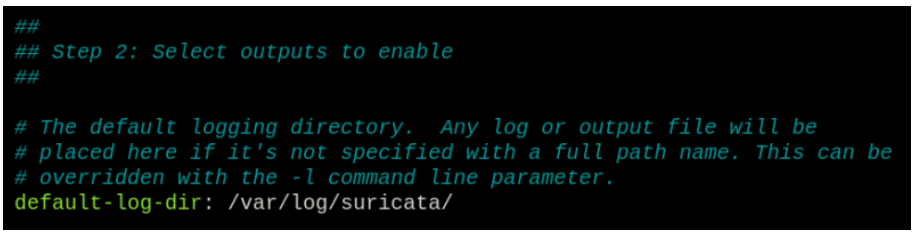
Tras ejecutarlo, se descargará dicho fichero de forma predeterminada en la ruta **/var/lib/suricata/rules**, por este motivo se ha elegido dicha ruta como ubicación por defecto de las regla para el archivo ***suricata.yaml***.

Tras esta configuración, es necesario reiniciar el servicio para que suricata aplique los cambios, para ello se ejecuta el siguiente comando:

```
$ service suricata restart
```

4. El último detalle a tener en cuenta será el consultar cuál es la ruta por defecto del fichero ***suricata.yaml*** que utilizará Suricata para guardar los archivos de logs. Dicha configuración es la siguiente:

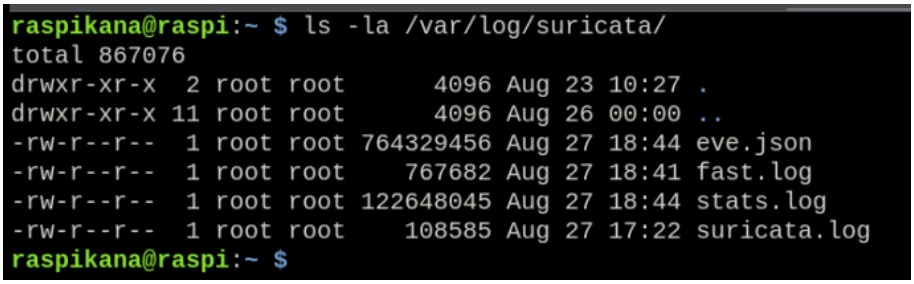
Figura 18: Ruta ficheros logs Suricata. Fuente: Elaboración propia.



Esta ruta es **/var/log/suricata**. Ejecutando el siguiente comando se mostrarán los ficheros de logs existentes:

```
$ ls -la /var/log/suricata
```

Figura 19: Ficheros de logs. Fuente: Elaboración propia.



El fichero ***suricata.log*** contiene los logs del funcionamiento de suricata. Si hay cualquier problema, aparecerá ahí el log correspondiente.

El fichero ***fast.log*** contiene los logs que generan las reglas que se han configurado en los archivos ".rules". Para ver el contenido de dicho fichero en directo, puede ejecutarse el siguiente comando:

```
$ tail -f /var/log/suricata/fast.log
```

Esta es una alternativa para analizar los logs de mayor nivel técnico y que se puede utilizar si el usuario lo desea. Por otra parte, se ha configurado ELK con la integración a Suricata como se ha visto en el apartado 5.2.7 con el objetivo de ofrecer una vista más dinámica y amigable de estos.

En este momento, suricata ya está configurado y funcionando en modo IDS – "Intrusion Detection System". Si se desea una monitorización de únicamente alertas, sin que suricata ejerza acciones de bloqueo sobre los paquetes, podría dejarse así. Si por el contrario se desea que suricata tome acciones sobre determinados paquetes, el siguiente paso es agregarle la funcionalidad de IPS – "Intrusion Prevention System".

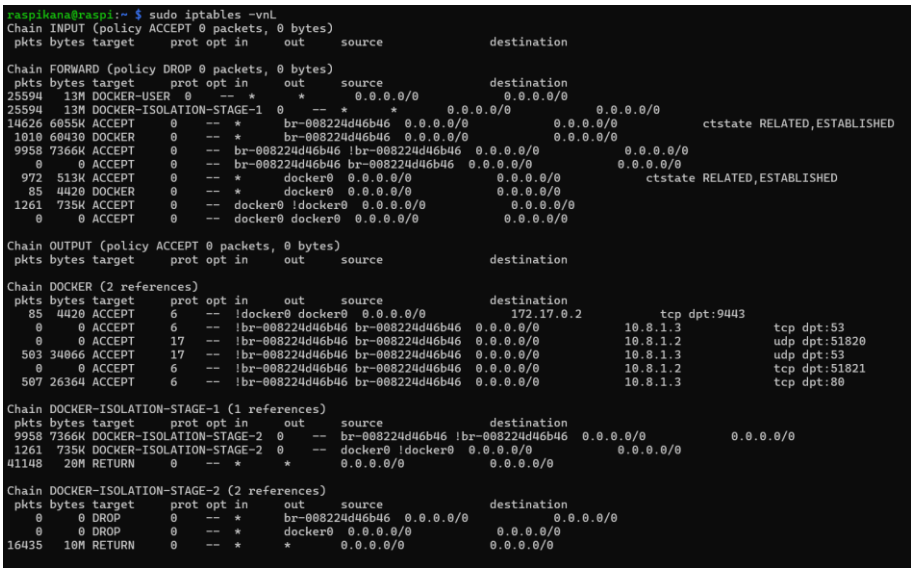
5.2.9.2. Convertir suricata a modo IPS:

Para que Suricata interactúe con el tráfico monitorizado, bloqueando aquel tráfico que sea malicioso, hay que modificara las "iptables" que trae el *kernel* de linux configuradas por defecto.

- Mostrar las iptables existentes:

```
$ sudo iptables -vnL
```

Figura 20: iptables. Fuente: Elaboración propia.



Se observa cómo para las tablas INPUT y OUTPUT no existen reglas. Mientras que para la tabla FORWARD sí que existen una serie de reglas que ha creado Docker por defecto tras su instalación. En caso de no haber instalado Docker, esta debería aparecer vacía también.

El tráfico de interés es aquel que llega y que sale de la red local, por ello, hay que modificar las tablas INPUT y OUTPUT, haciendo que redirijan todo el tráfico a Suricata. Para ello, introduciremos los siguientes comandos:

```
$ sudo iptables -I INPUT -j NFQUEUE
$ sudo iptables -I OUTPUT -j NFQUEUE
```

Tras esta configuración, la Raspberry Pi almacenará todos los mensajes en la cola NFQUEUE, y el servicio de Suricata deberá estar activo y consultando dicha cola para darle salida a todos los mensajes. En caso de que esté desactivado, se paralizarán todas las conexiones entrantes y salientes de la Raspberry Pi hasta que se active este servicio.

Por otro lado, la configuración por defecto de Suricata es en modo IDS, por lo que, para el correcto funcionamiento de suricata como IPS, habrá que ejecutar los siguientes comandos:

```
$ sudo service suricata stop
$ sudo suricata -c /etc/suricata/suricata.yaml -q 0
```

- Con el primer comando se detiene la ejecución por defecto que tiene suricata (IDS), ya que esta no consulta la cola NFQUEUE, y por lo tanto no da salida a todos los paquetes almacenados en dicha cola y como resultado se paralizan las conexiones entrantes y salientes.
- Con el segundo comando, se ejecuta Suricata escuchando en la cola 0 (NFQUEUE), donde se está redirigiendo todo el tráfico INPUT y OUTPUT gracias a la configuración hecha en iptables.

Tras esta configuración, quedaría completa la implementación de suricata como IDS + IPS.

5.2.9.3. Reglas de Suricata

A continuación, se da una breve introducción a la configuración de las reglas de Suricata, puesto que este no es el principal objetivo del proyecto, ya que las únicas reglas personalizadas que se han creado han sido las del fichero my.rules que se explican en este mismo apartado.

Toda la información acerca de las reglas puede consultarse en el repositorio oficial de Suricata: https://docs.suricata.io/en/latest/rules/index.html (OISF, 2024)

A continuación, se muestran las reglas creadas para testear el funcionamiento de Suricata dentro del fichero my.rules:

```
alert icmp any any -> 192.168.0.0/16 any (msg: "ICMP Packet found"; sid:1000001; rev:1;)
alert tcp any any -> any any (msg:"Facebook esta bloqueado"; content:"facebook"; sid:1000002; rev: 1;)
alert http $HOME_NET any -> $EXTERNAL_NET 80 (msg:"peticion GET detectada"; flow:established, to_server; content:"GET"; http_method; sid:1000003; rev:1;)
drop tcp any any -> any 22 (msg:"Conexion SSH detectada"; flow: to_server; app-layer-protocol:ssh; sid:1000004; rev:1;)
```

- En rojo se indica la acción que deberá hacer suricata, las principales son:
  - alert - generar una alerta.
  - pass - detiene la inspección del paquete.
  - drop - descarta el paquete y genera una alerta.
  - reject - envía un error RST/ICMP de alcance al remitente del paquete.
- En morado se indica el protocolo de conexión. Suricata admite multitud de ellos. Para más información consultar la sección de protocolos de su página oficial (OISF, 2024).
- En azul se indica la dirección IP origen y destino respectivamente.
- En amarillo se indica el puerto origen y destino respectivamente.
- En verde se indica la dirección, puede ser de dos formas:
  - Origen -> Destino (unidireccional).
  - Origen <> Destino (bidireccional).

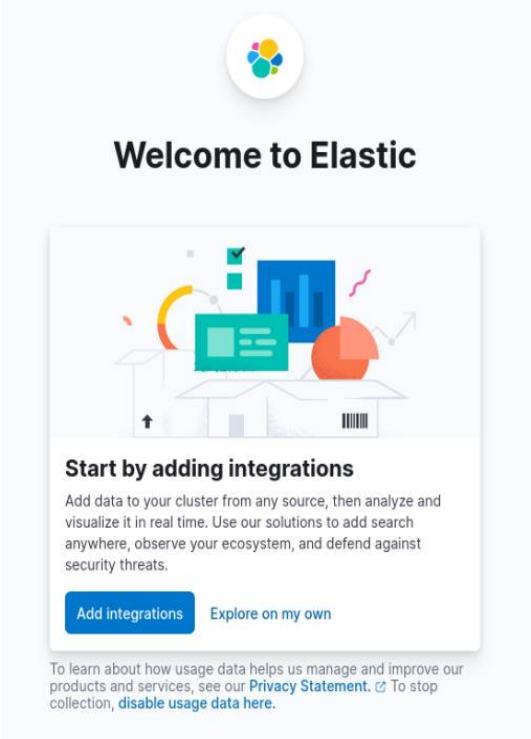
5.2.10. Filebeat + Suricata

Para integrar correctamente Suricata, que se encuentra instalado en la máquina host, con ElasticSearch y Kibana, que están contenidos en Docker, es necesario establecer un *pipeline* entre la máquina host y Docker. Para ello se ha utilizado el agente de recolección de Logs llamado Filebeat. Este se encarga de recoger los logs de Suricata, y enviárselo a la instancia de Filebeat dentro de Docker, explicado en 5.2.7, que a su vez se los enviará a ElasticSearch, para poder visualizarlos en el *dashboard* dentro de Kibana.

Para la configuración de Filebeat hay que seguir unos sencillos pasos que se detallan a continuación.

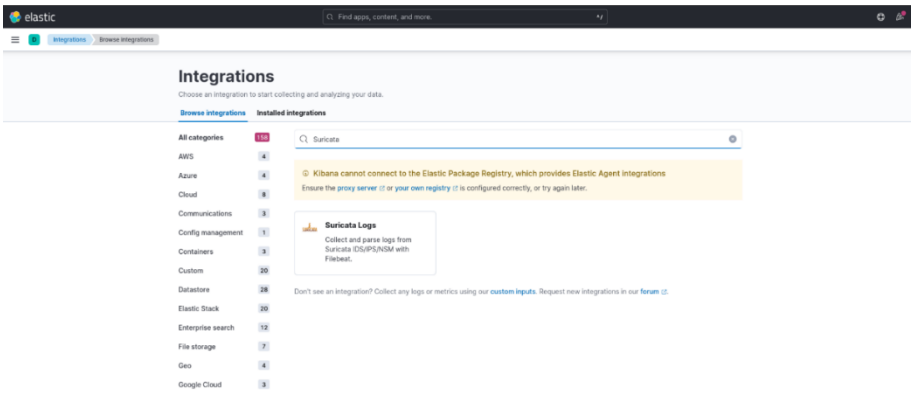
- En primer lugar, se accede a la interfaz gráfica de Kibana, la cual se puede ver en la Figura 11: Kibana Dashboard. Si no aparece esa vista, debería aparecer algo como lo siguiente:

Figura 21: Kibana Dashboard II. Fuente: Elaboración propia.



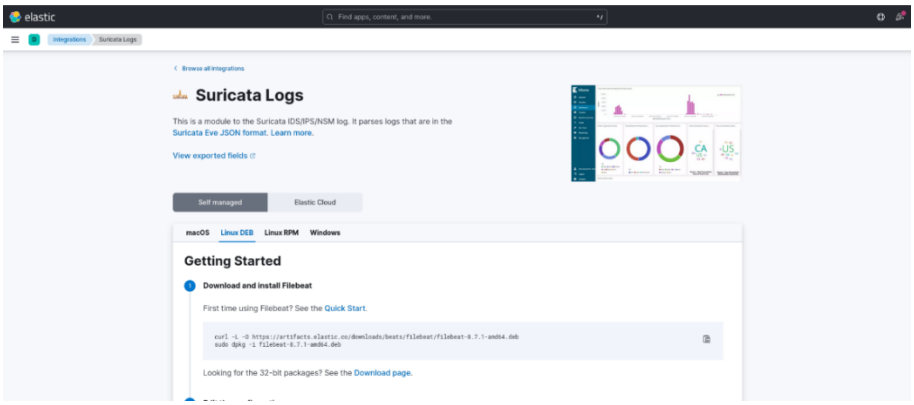
- Ya en esa vista, se pulsa en el botón "*Add Integrations*" y se busca la integración con Suricata.

Figura 22: Integración Suricata – Filebeat I. Fuente: Elaboración propia.



- El siguiente paso es seguir las instrucciones que aparecen en la interfaz web, ejecutando los comandos que aparecen dentro del apartado “Linux DEB”, puesto que Raspberry Pi OS tiene como base Debian.

Figura 23: Integración Suricata – Filebeat II. Fuente: Elaboración propia.



Algunos detalles importantes a tener en cuenta en la configuración son:

- En el apartado 1, al ser una Raspberry Pi, no tiene arquitectura “amd”, si no “arm”, por lo que habrá que sustituir dichos comandos, por los siguientes:

```
$ curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-
8.7.1-arm64.deb

$ sudo dpkg -i filebeat-8.7.1-arm64.deb
```

- En el apartado 2, la configuración a aplicar es la siguiente:

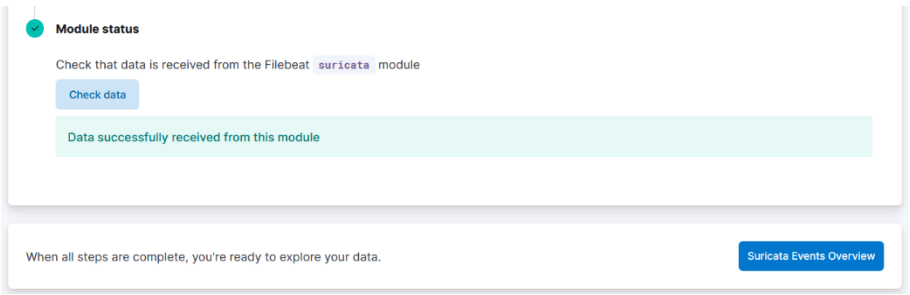
```
output.elasticsearch:
  hosts: ["0.0.0.0:9200"]
  username: "elastic"
  # password: "<password>"
  # If using Elasticsearch's default certificate
  # ssl.ca_trusted_fingerprint: "<es cert fingerprint>"

setup.kibana:
  host: " 0.0.0.0:5601"
```

- En el apartado 3 hay que asegurarse de que en la ruta /etc/filebeat/modules.d/suricata.yml, está activa la configuración de suricata de la siguiente forma:

```
- module: suricata
  # All logs
  eve:
    enabled: true
```

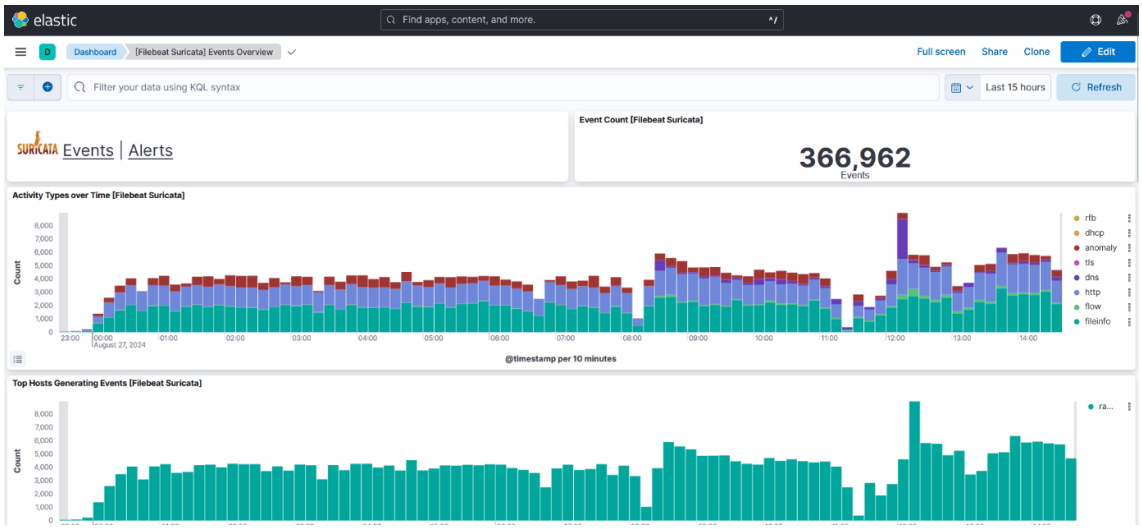
- Finalmente, con los comandos del apartado 4 finalizamos la integración con Filebeat. Se debe esperar unos minutos, recargar esa misma página, y pulsar en el botón “check data”. Si todo ha salido correcto, se debería ver así:

Figura 24: Check Data Integración. Fuente: Elaboración propia.



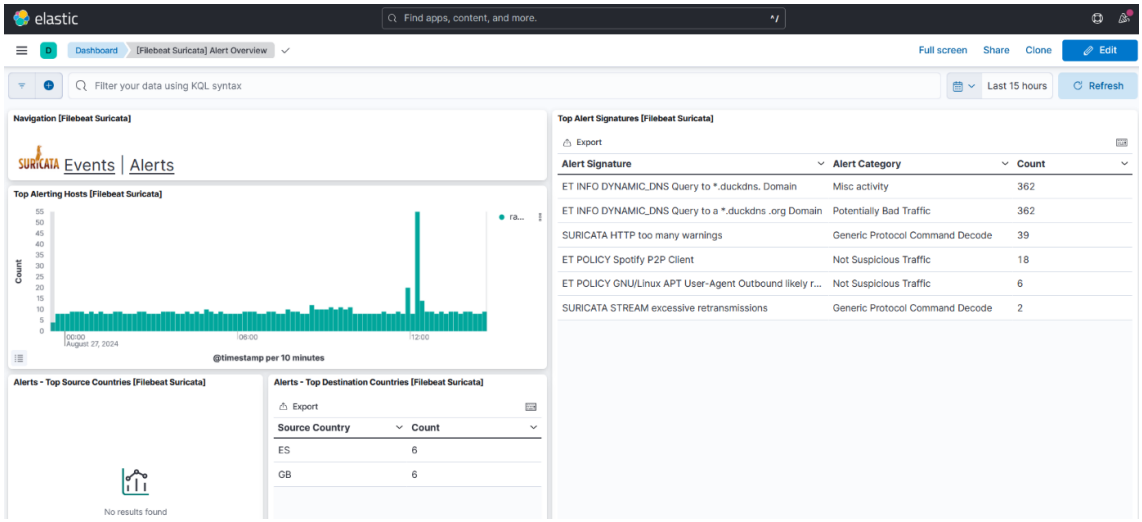
Tras esta configuración, también se ha creado el *dashboard* para visualizar correctamente todos los logs generados por Suricata. Este *dashboard* es plenamente configurable a gusto del usuario. Para este proyecto, se utilizará el que se crea por defecto. Para verlo, se debe pulsar en el botón “Suricata Events Overview”. Su vista debe verse de la siguiente manera:

Figura 25: Eventos Suricata. Fuente: Elaboración propia.



Esta es la vista de los eventos que monitoriza Suricata. Al pulsar en el botón “Alerts” se puede observar las alertas que han saltado en el rango de tiempo monitorizado. Estas alertas son las que se han configurado en los ficheros “.rules”, concretamente en el fichero “suricata.rules” en la ruta /var/lib/suricata/rules. En la vista web debe aparecer algo similar a lo siguiente:

Figura 26: Alertas Suricata. Fuente: Elaboración propia.



Tras los pasos explicados en este punto, el entorno ha quedado configurado en su totalidad y es completamente funcional. En el siguiente punto se van a exponer las pruebas de funcionalidad realizadas para validar el correcto funcionamiento de cada una de las herramientas que se han implementado.

5.3. PRUEBAS DE FUNCIONALIDAD

Tras la identificación de requisitos, y la descripción de las herramientas software utilizadas, queda el testeo y la exposición de los resultados obtenidos tras el uso de estas. A continuación, se muestra el correcto funcionamiento de dichas herramientas, satisfaciendo los objetivos propuestos inicialmente.

5.3.1. WireGuard

Se han realizado dos tipos de pruebas para evaluar WireGuard. Comprobar la funcionalidad, y comprobar el rendimiento.

5.3.1.1. Pruebas de funcionalidad.

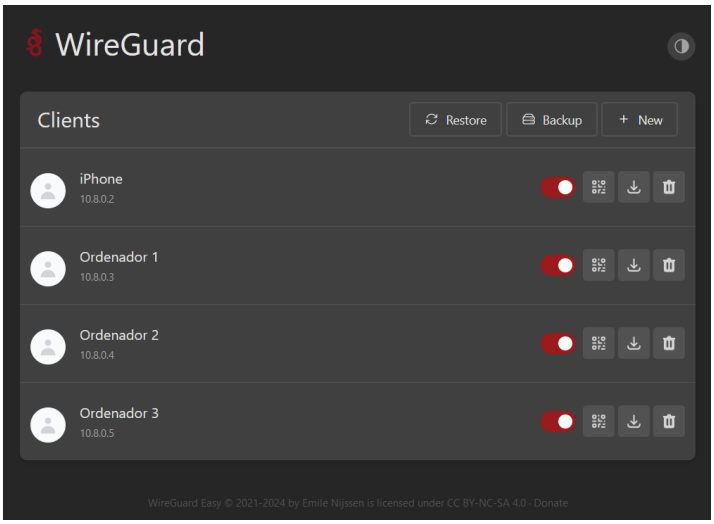
En estas pruebas, se ha estudiado la conectividad con la Raspberry Pi, y la IP pública asociada en tres tipos de escenarios diferentes.

- Escenario 1: Máquina host conectada a la red local mediante cable.
- Escenario 2: Máquina host, conectada a una red Wi-Fi independiente de la red local.
- Escenario 3: Escenario 2 con la VPN activada.

1. Escenario 1 – Máquina host conectada a la red local mediante cable.

Sin activar la VPN, se accede al dashboard de WireGuard:

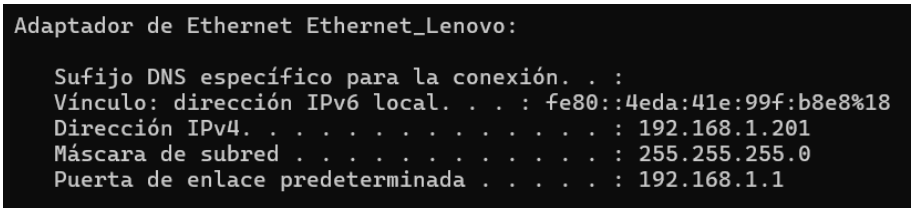
Figura 27: WireGuard Dashboard II. Fuente: Elaboración propia



En este momento, no hay ningún dispositivo conectado. Se utiliza como máquina host el Ordenador 1. Para el siguiente paso, desde la máquina host, se ejecuta el comando:

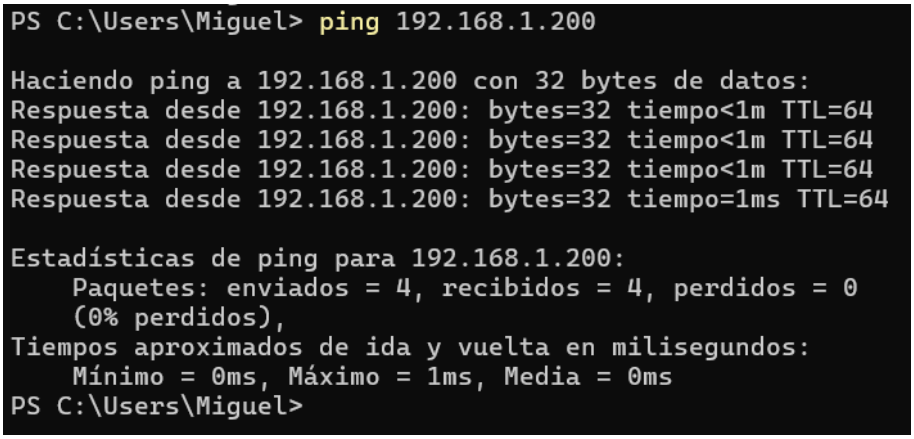
```
$ ifconfig
```

Figura 28: ifconfig escenario 1. Fuente: Elaboración propia.



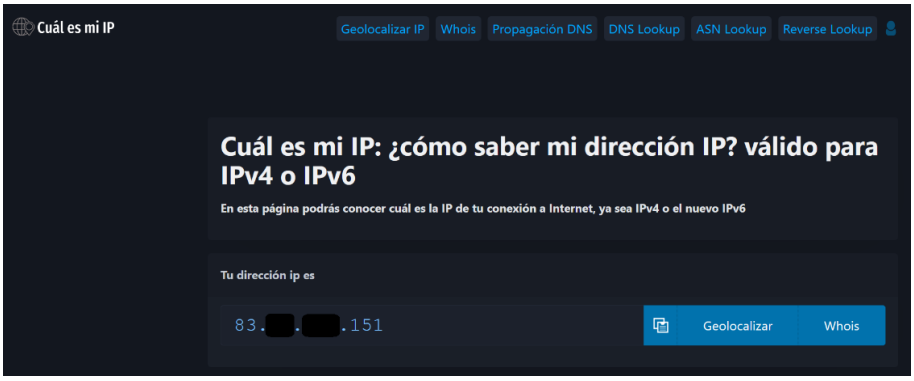
La máquina host tiene la dirección IP 192.168.1.201. Está conectada por cable a la red local. Se hace un ping a la dirección IP de la Raspberry Pi y se obtiene un resultado exitoso, puesto que están en la misma red:

Figura 29: ping escenario 1. Fuente: Elaboración propia.



El resultado al consultar la IP pública es (se limita la visualización por seguridad):

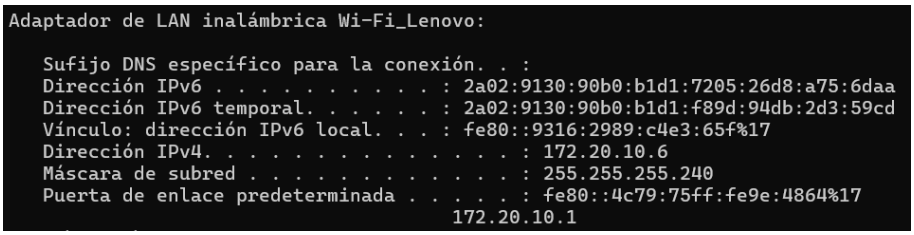
Figura 30: IP pública escenario 1. Fuente: (Grupo ADSLZone, 2024).



2. Escenario 2 – Máquina host, conectada a una red Wi-Fi independiente de la red local.

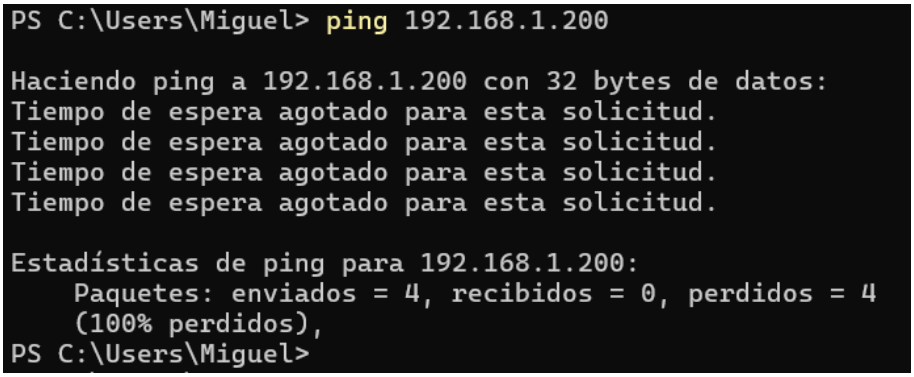
Se analizan los mismos pasos que para el caso 1.

Figura 31: ifconfig escenario 2. Fuente: Elaboración propia.



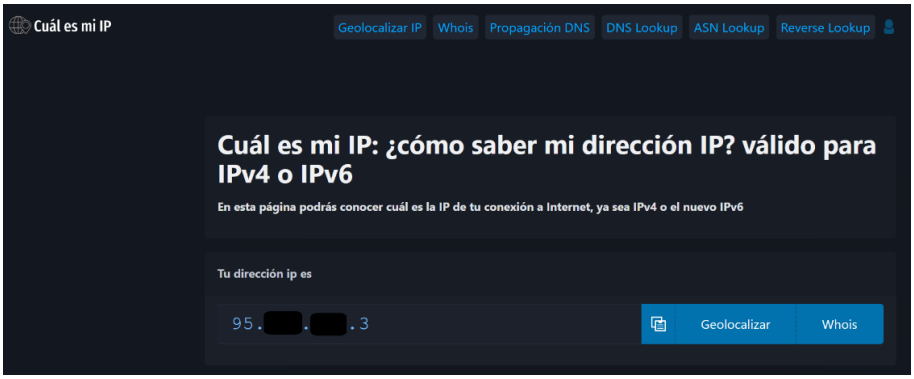
Se observa cómo la IP obtenida no es una IP del rango 192.168.1.0/24, como en la red local, ni una IP del rango 10.8.1.0/24, que son las que da WireGuard al conectarse a la VPN. Se envía un ping a la dirección de la Raspberry Pi, obteniéndose un error debido a que nunca llega a alcanzar dicha conexión. Se evidencia que no están en la misma red local.

Figura 32: ping escenario 2. Fuente: Elaboración propia.



El resultado al consultar la IP pública es (se limita la visualización por seguridad):

Figura 33: IP pública escenario 2. Fuente: (Grupo ADSLZone, 2024).

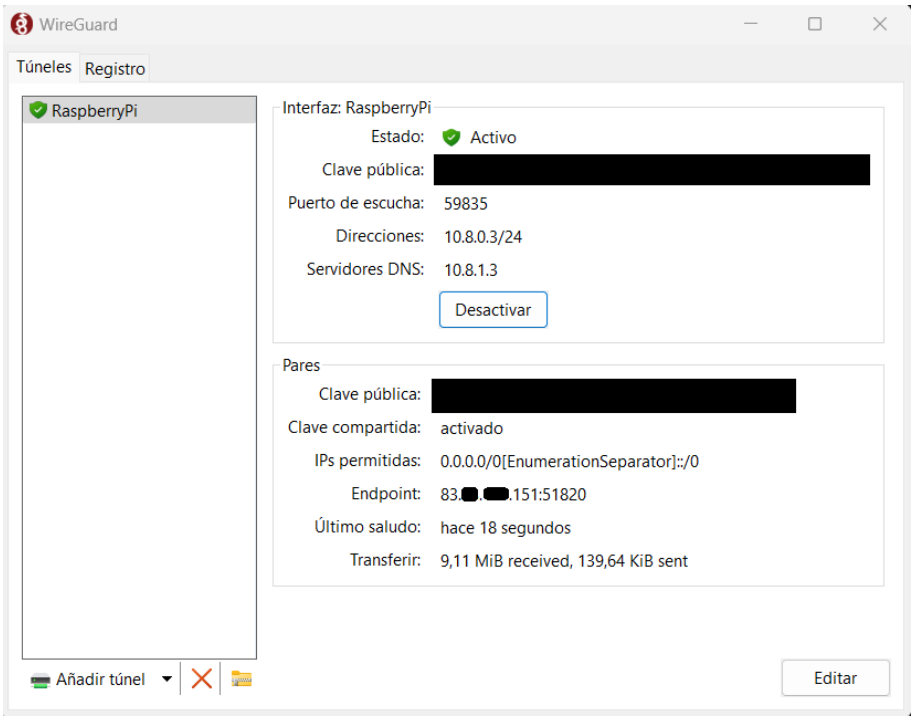


Pese a estar censurada dicha información, se observa claramente que no son similares tan siquiera, por lo que son dos redes completamente independientes.

3. Escenario 3 – Escenario 2 con la VPN activada.

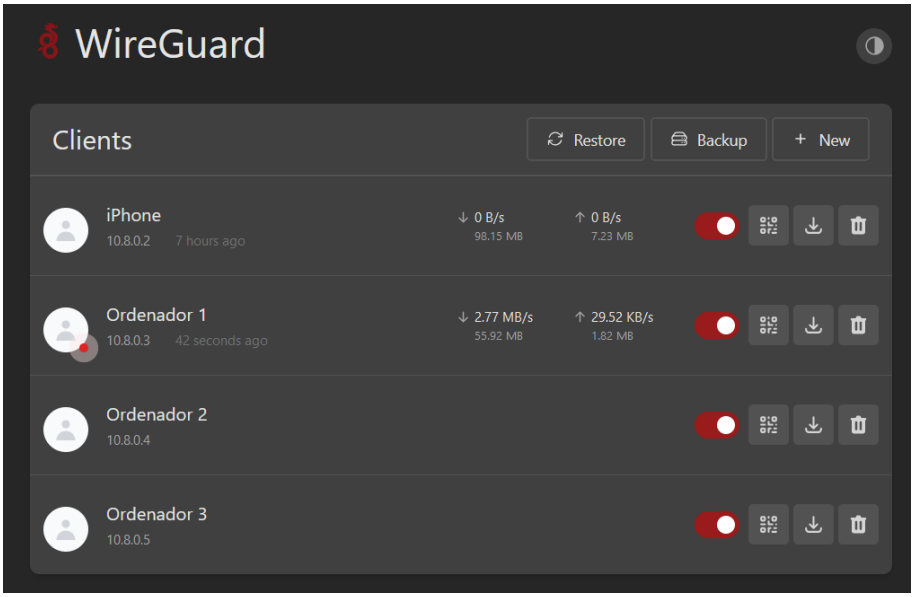
Para este caso, se ha mantenido la misma conexión que en el escenario 2. Se ha activado la conexión mediante VPN en la máquina host, y se han realizado las mismas comprobaciones. A continuación, se muestra la imagen del cliente VPN en dicha máquina:

Figura 34: Cliente WireGuard. Fuente: Elaboración propia.



Al consultar el dashboard de WireGuard, se observa la conexión activa de dicha máquina.

Figura 35: WireGuard Dashboard III. Fuente: Elaboración propia.



Al ejecutar el comando `$ ifconfig` se obtiene la misma información para la interfaz Wi-Fi del ordenador, pero, además, se obtiene una nueva interfaz más, la que da WireGuard al conectarse:

Figura 36: ifconfig escenario 3. Fuente: Elaboración propia.

```
Adaptador desconocido RaspberryPi:

   Sufijo DNS específico para la conexión. . :
   Dirección IPv4. . . . . . . . . . . . . . : 10.8.0.3
   Máscara de subred . . . . . . . . . . . . : 255.255.255.0
   Puerta de enlace predeterminada . . . . . : ::
                                               0.0.0.0
```

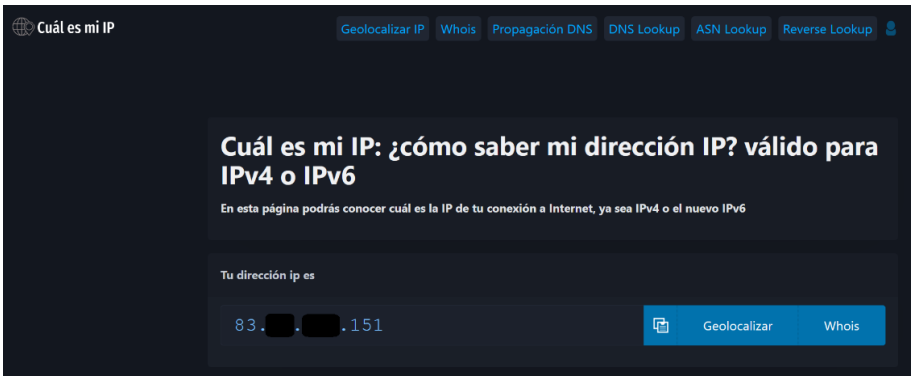
Lo que ha sucedido es que la Raspberry Pi, crea una subred dentro de la red doméstica (10.8.1.0/24 configurada en el fichero *compose.yaml*), y ofrece ese rango de IPs a las máquinas que se conectan. Al hacer ping a la dirección IP de la Raspberry Pi, se logra la conexión:

Figura 37: ping escenario 3. Fuente: Elaboración propia.

```
PS C:\Users\Miguel> ping 192.168.1.200

Haciendo ping a 192.168.1.200 con 32 bytes de datos:
Respuesta desde 192.168.1.200: bytes=32 tiempo=64ms TTL=63
Respuesta desde 192.168.1.200: bytes=32 tiempo=45ms TTL=63
Respuesta desde 192.168.1.200: bytes=32 tiempo=115ms TTL=63
Respuesta desde 192.168.1.200: bytes=32 tiempo=24ms TTL=63

Estadísticas de ping para 192.168.1.200:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 24ms, Máximo = 115ms, Media = 62ms
PS C:\Users\Miguel>
```

Además, como se puede observar, el TTL es una unidad menor, lo que indica que los paquetes han dado un salto más que en el escenario 1. Esto es a causa de la conexión VPN. Finalmente, el resultado al consultar la IP pública es el mismo que en el escenario 1:

Figura 38: IP pública escenario 3. Fuente: (Grupo ADSLZone, 2024).



Por lo tanto, se puede concluir este apartado de forma satisfactoria tras comprobar que conectado a la VPN, se obtiene salida a internet como si se estuviese en la red local, y esto se realiza mediante una conexión cifrada desde la máquina host a la Raspberry Pi, lo que lo hace especialmente seguro para aquellas situaciones en las que el usuario se encuentre conectado a una red no segura.

5.3.1.2. Pruebas de rendimiento.

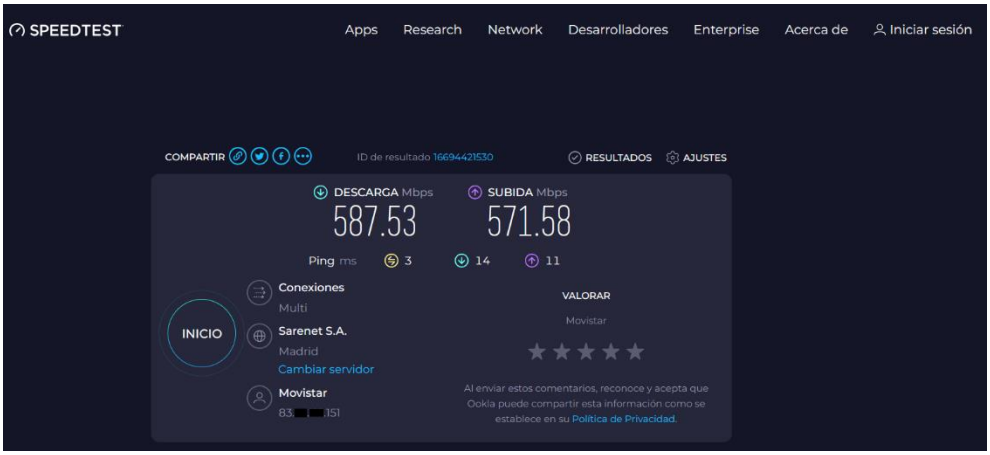
Para las pruebas de rendimiento, se han evaluado dos escenarios:

- Escenario 1: Rendimiento de conexión 5G con VPN desactivada, frente a VPN activada.
- Escenario 2: Rendimiento de conexión Wi-Fi pública país extranjero, con VPN desactivada, frente a VPN activada.

1. Escenario 1: Rendimiento de conexión 5G con VPN desactivada, frente a VPN activada.

Antes de exponer los resultados de esta evaluación, cabe destacar que la red a la que está conectada la Raspberry Pi mediante cable tiene una tarifa de 600MB simétricos de velocidad. La evidencia de estos resultados se expone en la siguiente imagen. Es una prueba de velocidad realizado desde la página web speedtest.net:

Figura 39: SpeedTest red Raspberry Pi. Fuente: (Ookla, LLC., 2024).

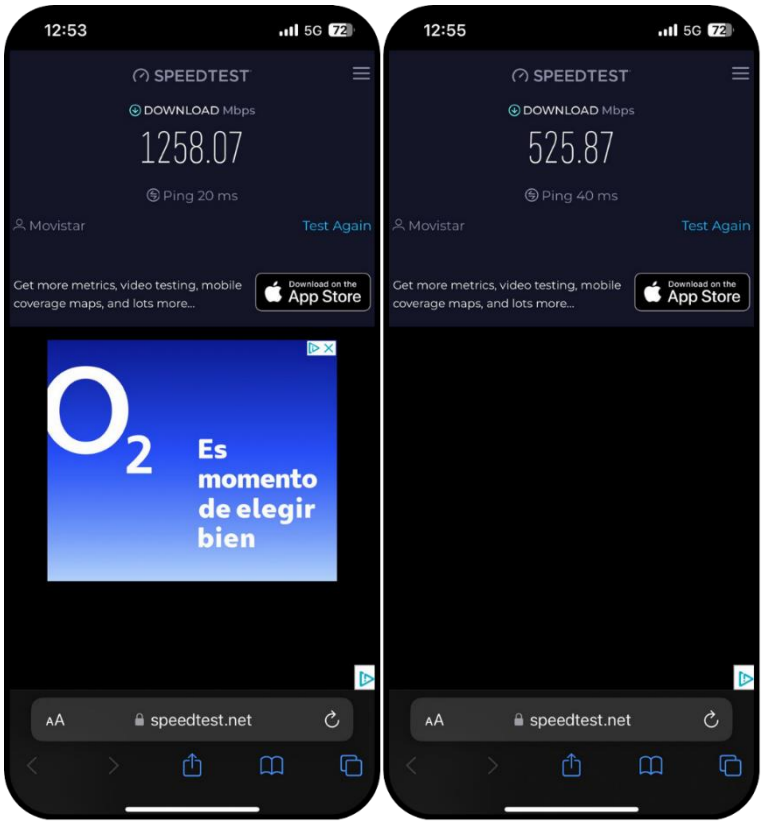


Tras esta aclaración, se exponen los resultados de realizar una prueba de velocidad con un dispositivo móvil, concretamente un iPhone 13. A la izquierda se observa la prueba de

velocidad sin conectarse a la VPN, obteniéndose un resultado de 1.258,07 Mbps, frente a la prueba de la derecha con la VPN activada y un resultado de 525,87Mbps.

Se observa que aparece un cuello de botella en la conexión. Existe una diferencia de 732,2Mbps. Pero dicho cuello de botella está proporcionado en su mayor parte por la tarifa de internet contratada, siendo únicamente 61,66Mps la diferencia entre la conexión cableada de la Raspberry Pi y la conexión VPN del dispositivo móvil.

Figura 40: SpeedTest escenario I. Fuente: Elaboración propia.

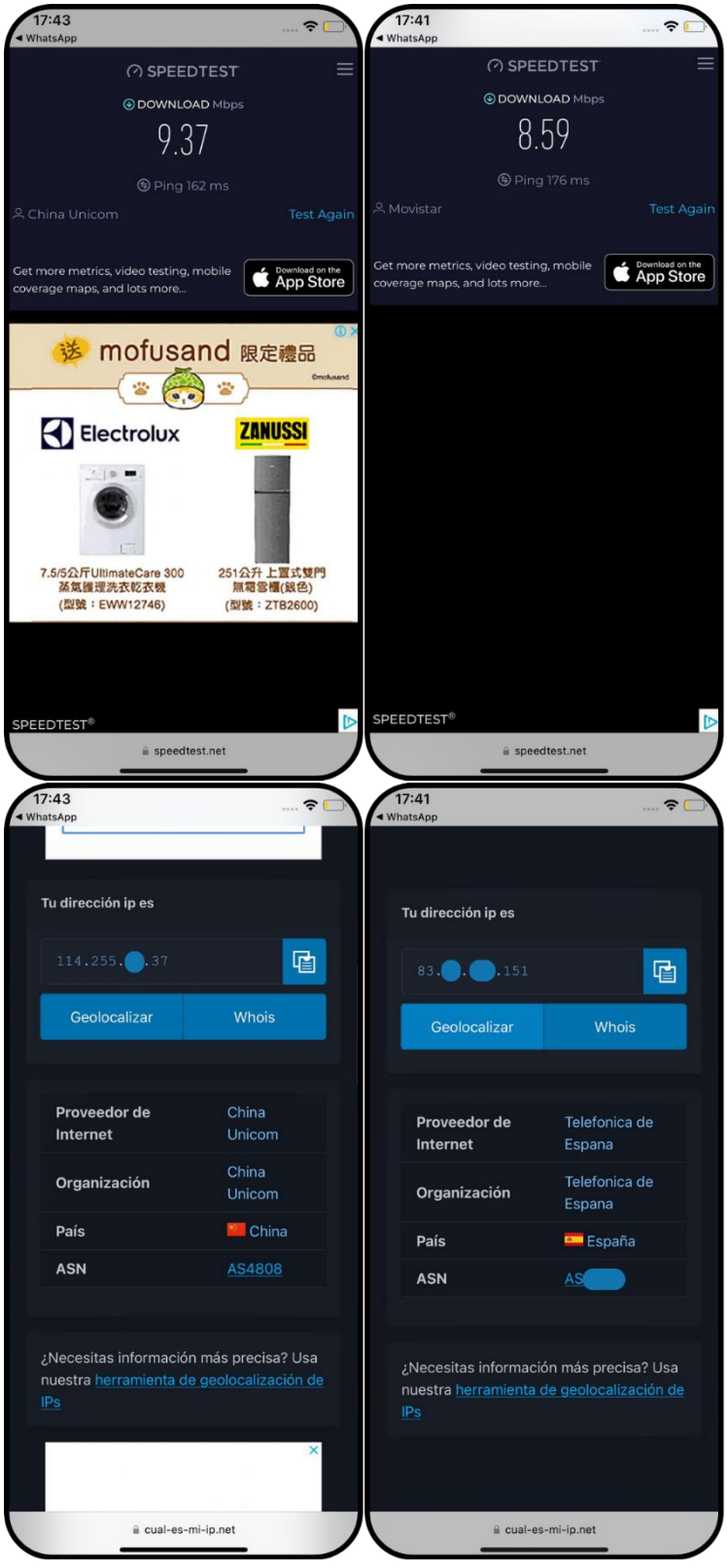


2. Escenario 2: Rendimiento de conexión Wi-Fi pública país extranjero, con VPN desactivada, frente a VPN activada.

En este escenario se ha simulado una prueba real del funcionamiento de la VPN, siendo este el objetivo para el cual está pensada esta herramienta. Para ello, se ha contado con la ayuda de una tercera persona, ubicada en ese momento en el Aeropuerto Internacional de Pekín-Capital, en China. En este país, aplicaciones como WhatsApp o TikTok están bloqueadas y no funcionan cuando el usuario está conectado a una red pública o privada dentro del país, debido a la censura existente en el mismo. Para esta prueba, en primer lugar, se ha realizado

un test de velocidad de internet y se ha consultado la dirección IP pública sin estar conectado a la VPN (fotos de la izquierda), y posteriormente, se ha activado la VPN y se han repetido las mismas pruebas (fotos de la derecha), obteniéndose los siguientes resultados.

Figura 41: Pruebas VPN desde China. Fuente: Elaboración propia.



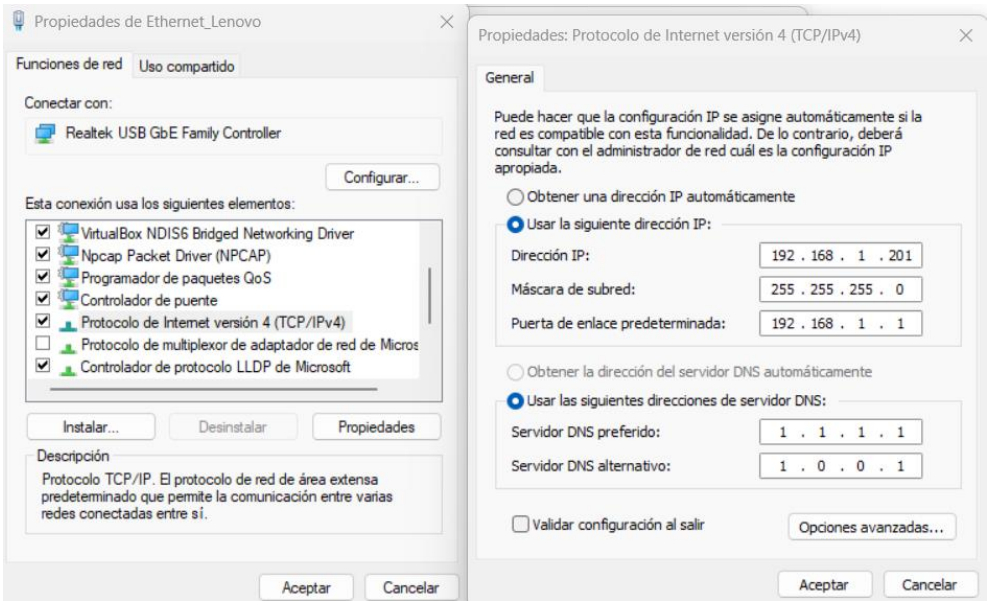
Estos resultados muestran un comportamiento muy productivo de la VPN. En primer lugar, se observa cómo, pese a estar a miles de kilómetros de distancia, la diferencia de velocidad es insignificante, de solamente 0,78Mbps. Esto es debido a que una de las cosas de las que presume WireGuard, es de ser una tecnología novedosa e innovadora, que ofrece un cifrado seguro junto a un muy buen rendimiento (WireGuard®, 2022), y eso es lo que se ha comprobado gracias a estas pruebas. En segundo lugar, aunque esto se explica en el próximo apartado, en las capturas de pantalla se evidencia que gracias a la conexión VPN y el funcionamiento de Pi-hole, los anuncios y los rastreadores, quedan totalmente eliminados y bloqueados. Por otra parte, también se muestra que, tras conectarse a la VPN, tanto la dirección IP pública, como la ubicación, cambian de China, a las que aplican a la red local donde se ubica la Raspberry Pi (tal y como se observa en las pruebas de funcionalidad de este mismo apartado). Y, finalmente, lo más interesante de esta conexión es, que el usuario final, gracias a la VPN, consigue tanto una conexión segura dentro de la red pública puesto que WireGuard ofrece una tunelización cifrada hasta el servidor VPN (WireGuard®, 2022), como acceder satisfactoriamente a las aplicaciones que inicialmente estaban restringidas en China, como es el uso de WhatsApp. Evidencia de esto es la obtención de dichas pruebas mientras el usuario continuaba en el país destino.

5.3.2. Pi-hole

En segundo lugar, se muestra el funcionamiento de Pi-hole como bloqueador de rastreadores y de anuncios, para asegurar una mayor privacidad en la navegación web. Estos resultados se han obtenido tanto para los dispositivos de la red local en los que se ha configurado como servidor DNS la Raspberry Pi, como para aquellos dispositivos que están conectados mediante VPN a la red local, ya que estos, utilizan por defecto Pi-hole como servidor DNS, gracias a la configuración realizada en el fichero *compose.yaml*.

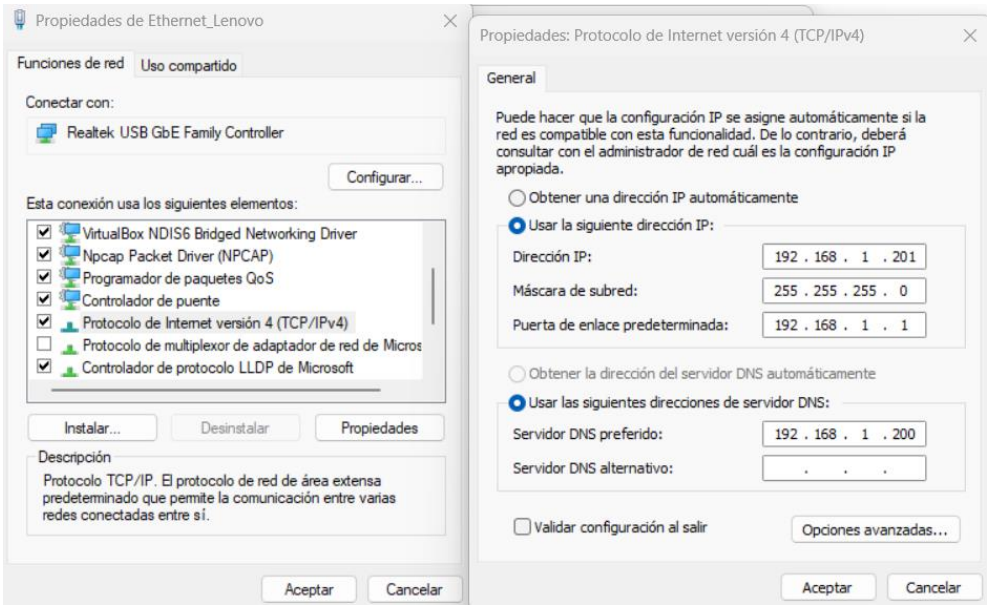
En esta prueba se utilizarán dos escenarios. En el primer escenario se configura la máquina host de forma que no utilice Pi-hole como servidor DNS. En el segundo escenario, se configura la máquina host de manera que sí que utilice Pi-hole como servidor DNS.

Figura 42: Configuración servidor DNS máquina host escenario I. Fuente: Elaboración propia.



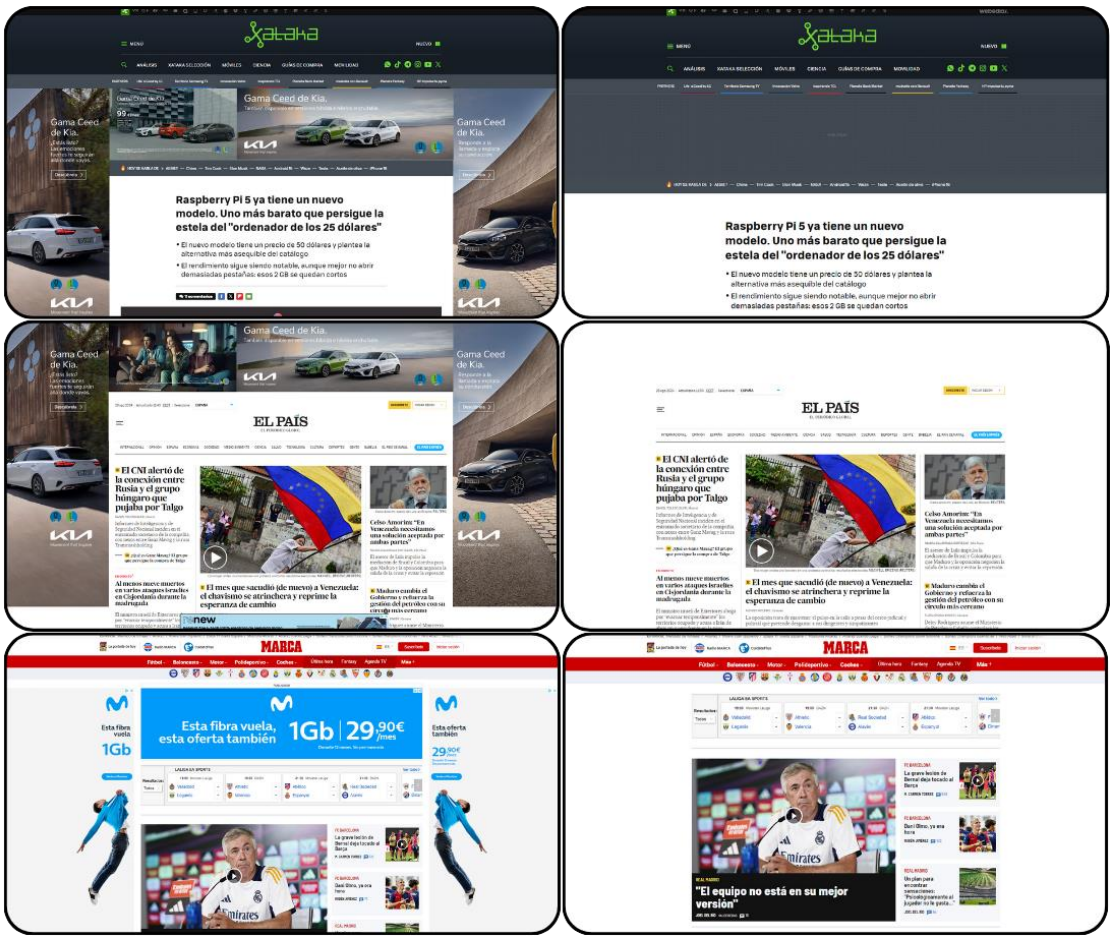
Para esta prueba se ha elegido a CloudFare como servidor DNS con la dirección 1.1.1.1 (Cloudflare, Inc., 2024).

Figura 43: Configuración servidor DNS máquina host escenario II. Fuente: Elaboración propia.



Una vez realizada esta configuración, se han elegido una serie de páginas web con algunos anuncios, los cuales hacen que la navegación sea más lenta, y la lectura de la información, más difícil. Las imágenes del acceso a dichas páginas web se muestran a continuación. En la izquierda las imágenes correspondientes al escenario I, y en la derecha las imágenes correspondientes al escenario II:

Figura 44: Pruebas Pi-hole. Fuentes: (Pastor, 2024); (ElPais, 2024); (MARCA, 2024).



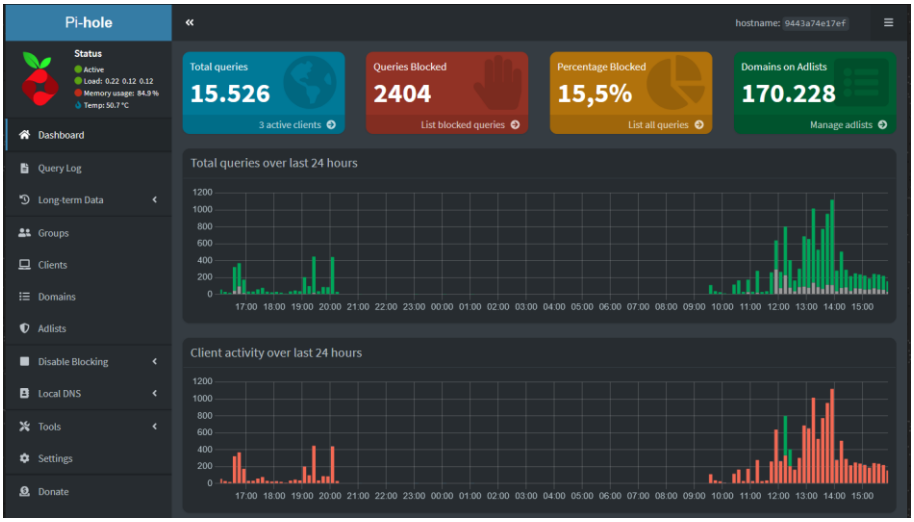
Como puede apreciarse en las imágenes del escenario I, en todas ellas aparece publicidad de diferentes compañías, tanto de automóviles como de operadoras telefónicas, que nada tienen que ver con la información que se busca leer en un periódico. Esto hace que la página cargue más lenta, el movimiento por ella sea más tedioso, se tengan más distracciones, y se carguen una serie de rastreadores que monitorizan al usuario si se pulsa en alguno de los anuncios publicitarios. Esto, puede llegar a ser un problema de seguridad, teniendo en cuenta la privacidad de los datos que se comparten, y la seguridad del sitio web al que redirigen.

Tal y como se puede observar en las imágenes de la derecha, gracias al uso de Pi-hole se consigue un filtrado de anuncios, publicidad y rastreadores, que proporcionan al usuario una presentación más segura, limpia y rápida.

Tras el análisis de estos resultados, se puede afirmar que la implementación de Pi-hole satisface a la perfección este problema y el objetivo del trabajo.

Al acceder a la interfaz de Pi-hole, se observa cómo ha gestionado y resuelto todo este tráfico. A continuación, se muestran unas imágenes de su dashboard con esta información:

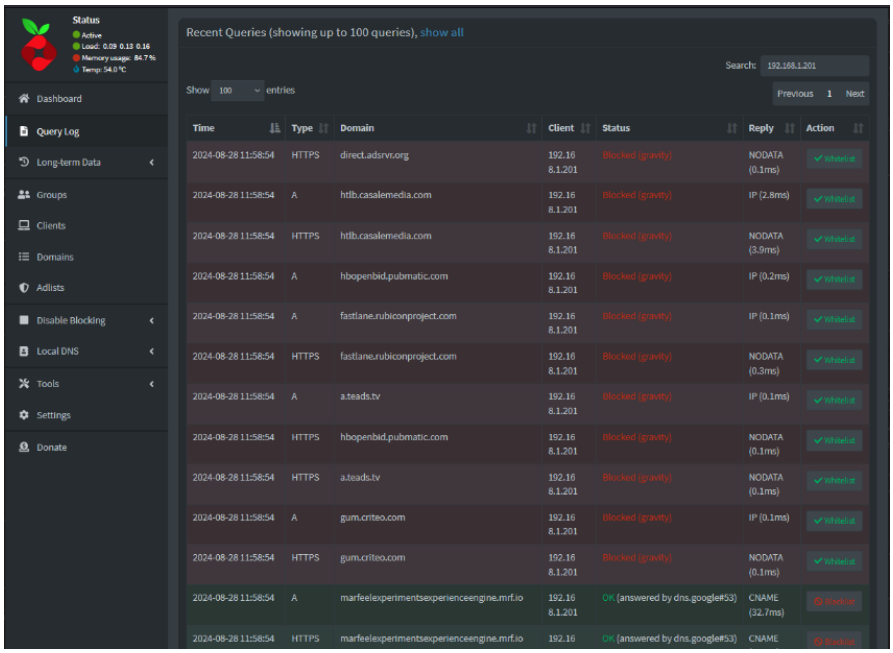
Figura 45: Pi-hole Dashboard II. Fuente: Elaboración propia.



En esta vista se observa la ventana principal de Pi-hole. Sus datos tienen sentido puesto que el rango de tiempo sin actividad corresponde al rango de tiempo en el que los dispositivos que utilizaban como servidor DNS a Pi-hole, estaban apagados.

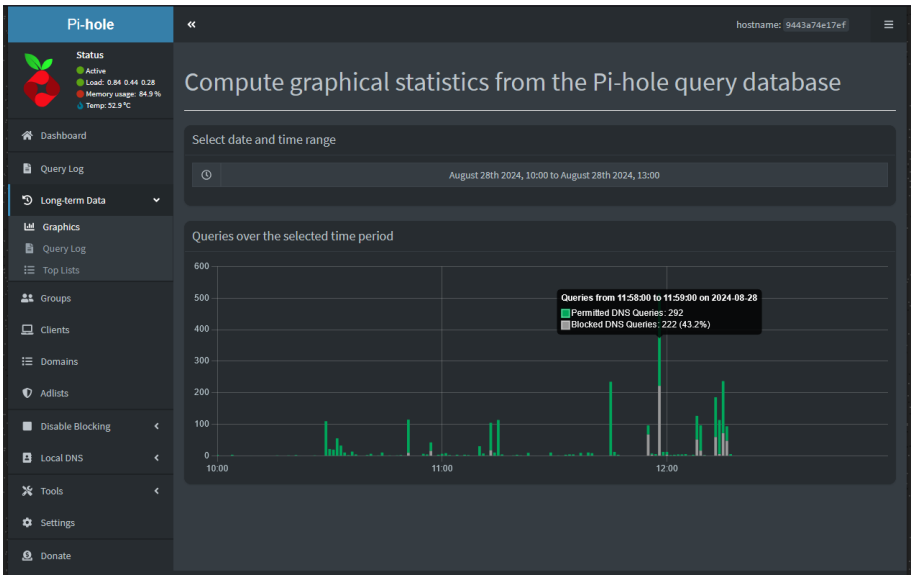
Arriba en el cuadro de “Total queries”, se observa que hay 3 clientes activos. Dichos clientes corresponden a la máquina host de la red local donde se estaban realizando las pruebas anteriores, a un teléfono móvil conectado por VPN, y a la propia Raspberry Pi.

Figura 46: Pi-hole Query Log I. Fuente: Elaboración propia.



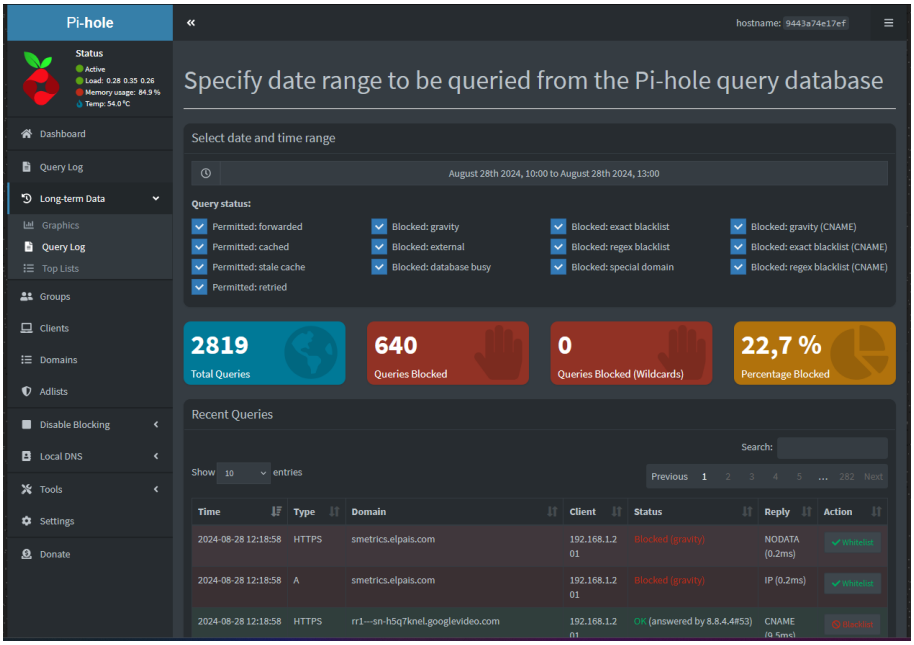
En esta imagen, se muestran todos los logs que crea Pi-hole. Se ha filtrado por la dirección IP de la máquina host, obteniéndose así todas las peticiones que se han hecho. En la foto se observa cómo algunas de ellas (en rojo) se han bloqueado por estar presente en la *black list* configurada, y otras (en verde) han sido aceptadas.

Figura 47: Pi-hole Graphics. Fuente: Elaboración propia.



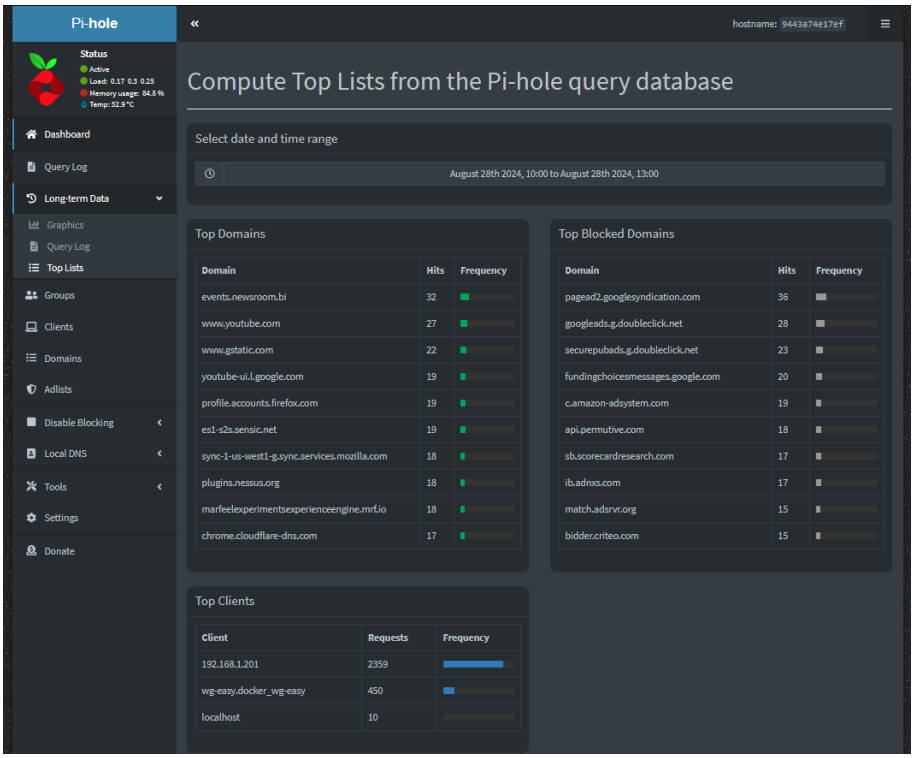
Esta es una vista en modo gráfico, que ofrece el filtrado por rango de tiempo.

Figura 48: Pi-hole Query Log II. Fuente: Elaboración propia.



Esta es una vista en modo logs nuevamente, que ofrece el filtrado por rango de tiempo.

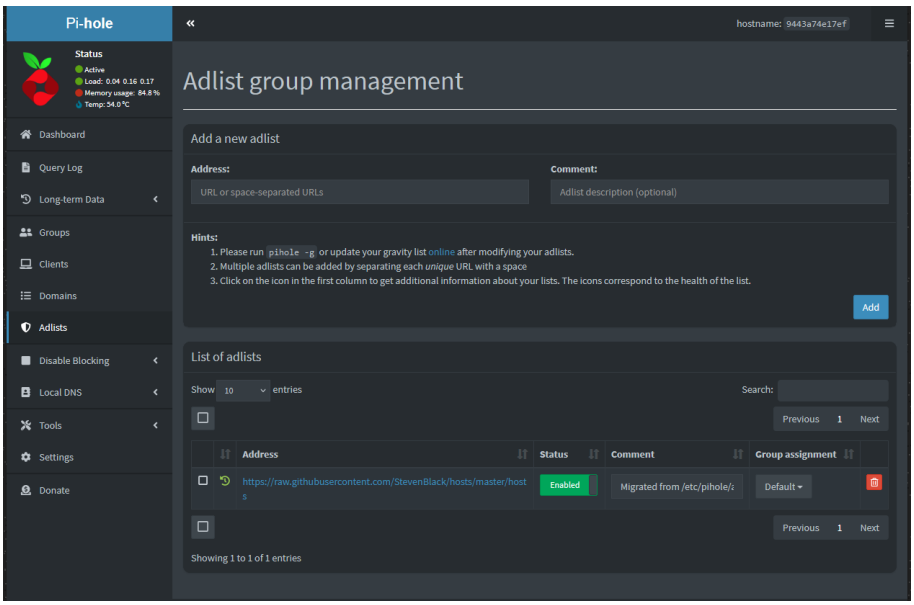
Figura 49: Pi-hole Top Lists. Fuente: Elaboración propia.



En esta vista, se observa el top de dominios admitidos, el top de dominios bloqueados, y los clientes que más interactúan con Pi-hole. En este caso, solamente existen 3 clientes.

Finalmente, la siguiente imagen muestra dónde se encuentra configurada la black list que utiliza Pi-hole para dirigir el tráfico. En caso de querer agregar más listas, se debería hacer en esta parte:

Figura 50: Pi-hole Adlists. Fuente: Elaboración propia.



Tras este análisis, se puede concluir dando por cumplido el objetivo de Pi-hole en la red doméstica: proteger a los usuarios de rastreadores, publicidad no deseada, y sitios con contenido malicioso, aumentando de esta manera la seguridad en la navegación web.

5.3.3. Suricata junto a Elastic Stack

A continuación, el tercer y último servicio implementado para este proyecto, es la herramienta Suricata actuando como detección y prevención de intrusiones en el sistema. Tras la configuración explicada en el apartado 5.2.9, Suricata está activa y detectando el tráfico que tiene como origen, o como destino, la dirección IP de la Raspberry Pi.

Este detalle es debido a que para que Suricata detecte todo el tráfico de la red local, este tiene que pasar a través de la Raspberry Pi. Esto puede conseguirse de varias maneras, ya sea mediante el uso de un swich o un rúter que permita una configuración de *port mirroring,* o el uso de un *network TAP (Test Acces Point)*, dispositivos de los que no se disponen para este proyecto. Otra opción es configurando la interfaz en modo promiscuo, teóricamente se debería ver todo el tráfico de la red local a través de ella. Esta y otras posibles configuraciones se han probado para este proyecto, pero no se ha logrado dicha monitorización con éxito. Por este motivo, ese será un posible punto de partida para posibles trabajos futuros, los cuales se expondrán en el apartado 6.CONCLUSIONES Y TRABAJO FUTURO.

A pesar de este inconveniente, se ha proseguido con la configuración completa de Suricata para que actúe como IDS + IPS y la configuración de ElasticSearch, Kibana y Filebeat, para una correcta monitorización de todo el tráfico detectado por suricata. Para estas pruebas, se ha eliminado el fichero de *suricata.rules* del archivo ***suricata.yaml***, dejando únicamente el fichero *my.rules*. De esta manera se puede probar el comportamiento de cada una de las funcionalidades de Suricata, trabajando con un archivo más cómodo y pequeño. La imagen del dashboard de cómo aparecen las alertas del fichero suricata.rules en Kibana, puede visualizarse en la Figura 26: Alertas Suricata. Fuente: Elaboración propia.

A continuación, se muestra la configuración realizada al apartado de las reglas dentro del archivo ***suricata.yaml***:

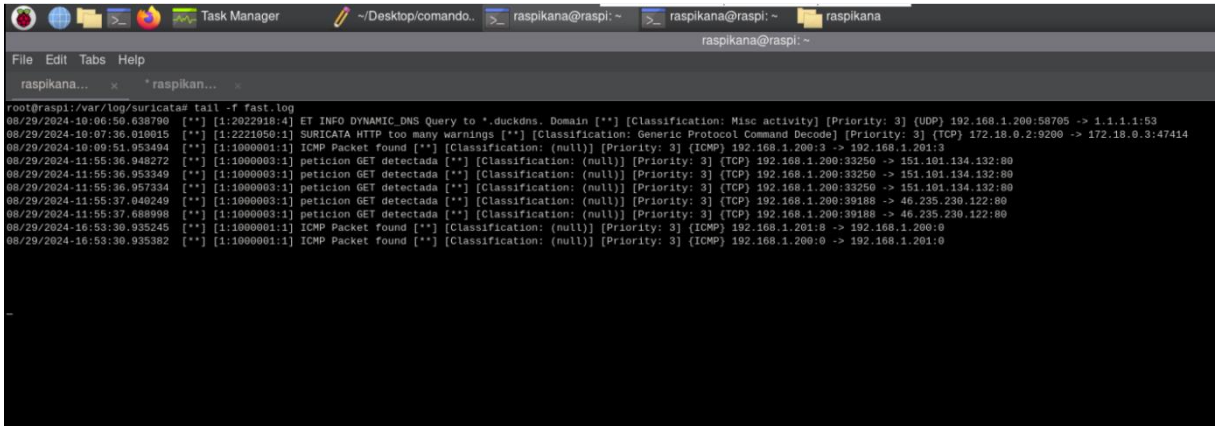
```
## Configure Suricata to load Suricata-Update managed rules.
default-rule-path: /var/lib/suricata/rules
rule-files:
  - my.rules
  #- suricata.rules
```

En el apartado 5.2.9.3, se explican las reglas que contiene el fichero my.rules.

```
alert icmp any any -> 192.168.0.0/16 any (msg: "ICMP Packet found"; sid:1000001; rev:1;)
alert tcp any any -> any any (msg:"Facebook esta bloqueado"; content:"facebook"; sid:1000002; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET 80 (msg:"peticion GET detectada"; flow:established, to_server; content:"GET"; http_method; sid:1000003; rev:1;)
drop tcp any any -> any 22 (msg:"Conexion SSH detectada"; flow: to_server; app-layer-protocol:ssh; sid:1000004; rev:1;)
```

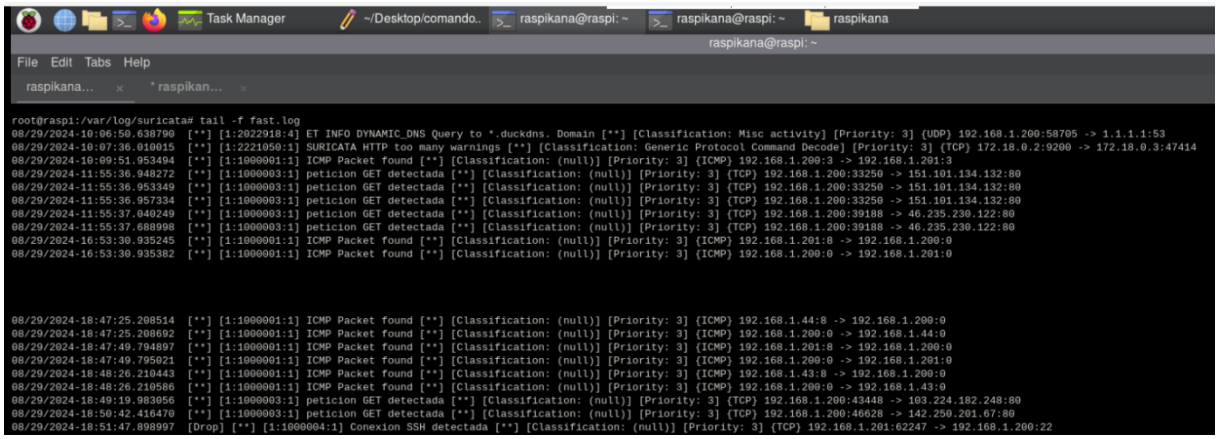
A continuación, se expone su funcionamiento mediante una serie de imágenes. En primer lugar, aparece el fichero de logs sin haber realizado ninguna prueba. Las alertas que aparecen son los restos de pruebas anteriores. Las nuevas que salgan a continuación serán las que se tengan en cuenta para las pruebas.

Figura 51: Suricata Logs I. Fuente: Elaboración propia.



A continuación, en primer lugar, se prueban 3 de las reglas descritas.

Figura 52: Suricata Logs II. Fuente: Elaboración propia.



Para la primera, “ICMP Packet found”, se ha hecho una conexión mediante el comando ping desde 3 dispositivos diferentes, obteniéndose el registro de las 3 conexiones correctamente.

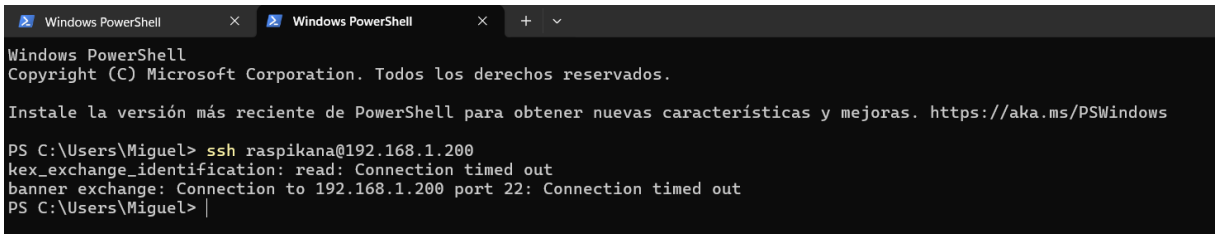
Para la tercera regla, “petición GET detectada”, se ha ejecutado el comando curl para dos URLs diferentes. Uno para la dirección xataka.es y otro para la dirección google.es tal y como se muestra a continuación:

Figura 53: Suricata test I. Fuente: Elaboración propia.

Para ambos comandos, se ha detectado el paquete enviado, y se ha alertado con éxito.

Finalmente, para probar la cuarta regla, se ha solicitado una conexión ssh desde otro dispositivo. En este caso, como la regla establece que todas las conexiones ssh se descarten (drop), esta conexión no ha sido establecida, si no que se ha desechado correctamente:

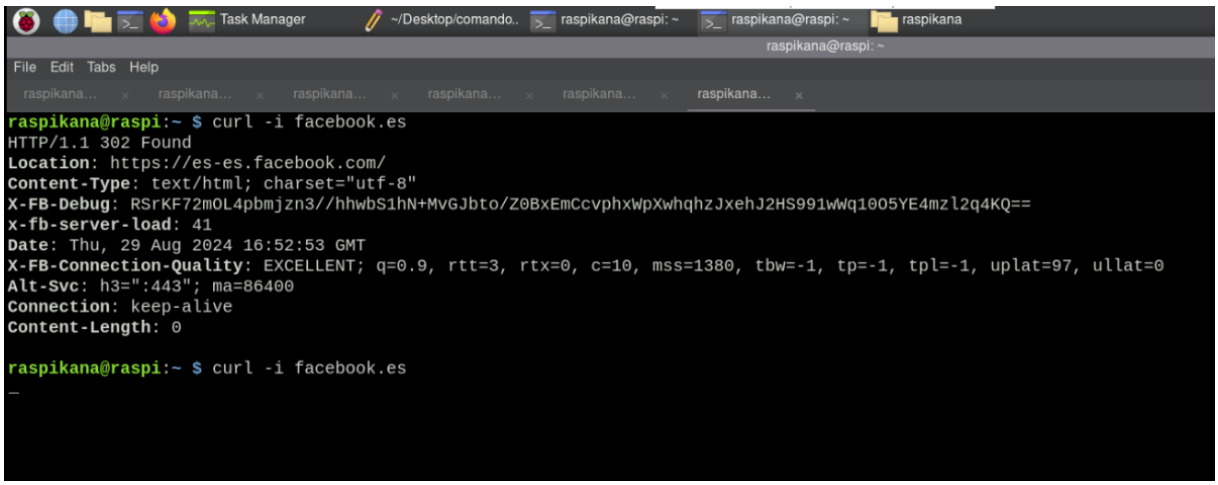
Figura 54: Suricata test II. Fuente: Elaboración propia.



Para la segunda regla, “Facebook está bloqueado” se ha probado de dos formas distintas.

- En primer lugar, se ha hecho un “curl” a la dirección facebook.es mientras en la regla estaba configurado como “alert”.
- Tras ello, se ha cambiado la regla a “drop” se ha reiniciado suricata, y se ha vuelto a probar el comando “curl” para ver cómo se comportaba. Como era de esperar, su funcionamiento ha sido el correcto. En primer lugar, se ha obtenido la respuesta al curl de forma satisfactoria, y se ha obtenido la alerta configurada, y en segundo lugar no se ha obtenido respuesta, y se ha obtenido el mensaje “Drop” configurado. A continuación, se muestra las evidencias gráficas:

Figura 55: Suricata test III. Fuente: Elaboración propia.

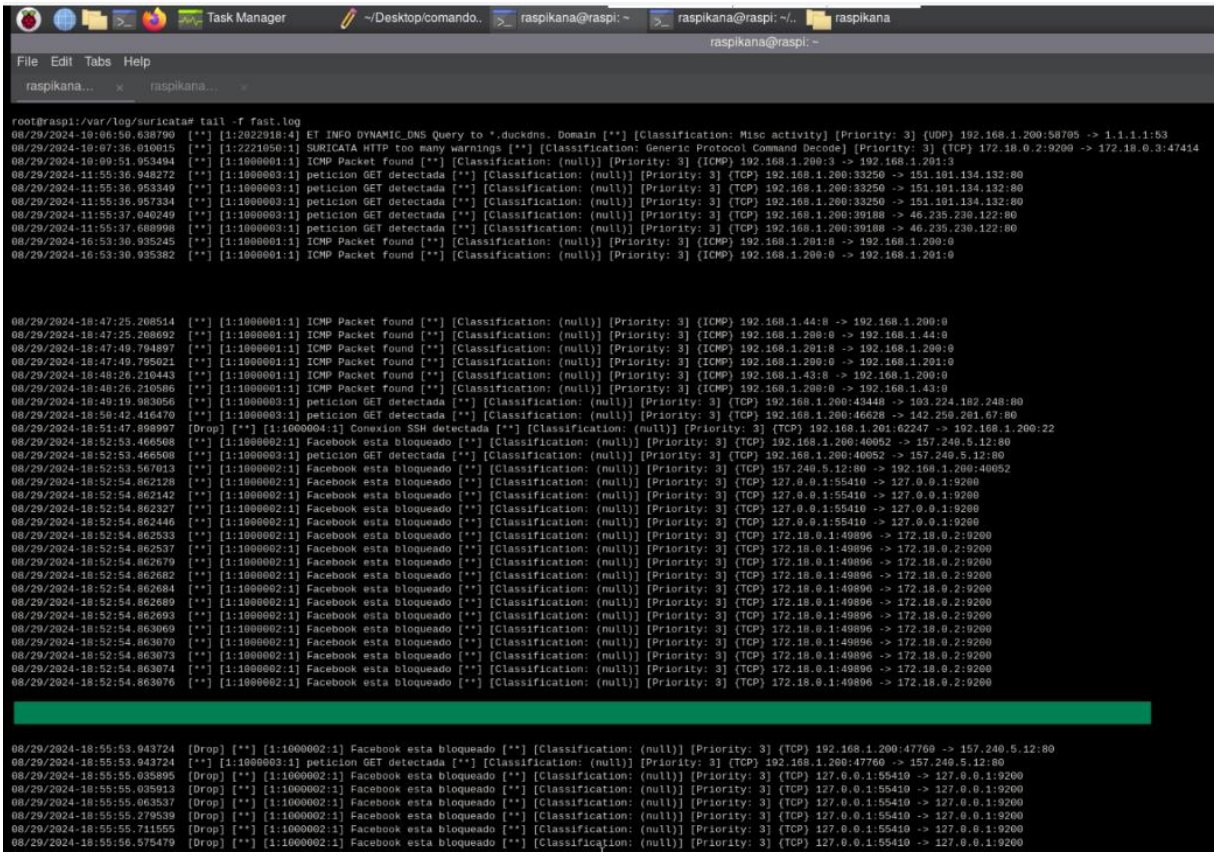


En esta imagen aparecen los dos comandos curl, el primero habiendo obtenido una respuesta exitosa, mientras que el segundo se ha quedado así, sin obtener ninguna respuesta, ya que suricata estaba filtrando ese tráfico.

A continuación, en la imagen de los logs se muestra con la separación de una línea verde ambos momentos de la prueba. Por otra parte, debe tenerse en cuenta que el tráfico con dirección IP distinta a la subred 192.168.1.0/24 es tráfico interno de la Raspberry Pi,

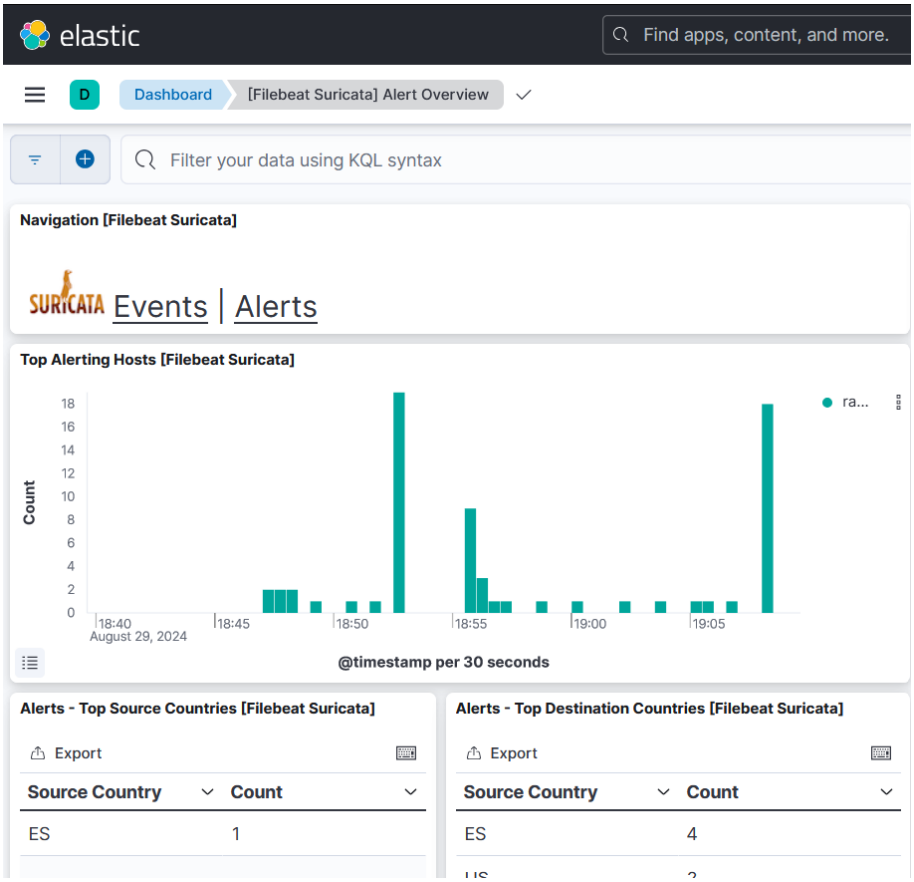
concretamente de Kibana, que también cumple la condición que se ha configurado en la regla, es decir, serían falsos positivos debido a que es una regla muy básica que se ha configurado para estas pruebas.

Figura 56: Suricata Logs III. Fuente: Elaboración propia.



Una vez comprobado el correcto funcionamiento de Suricata como IDS y como IPS, ahora queda ver cómo se visualiza esto en el dashboard creado mediante Kibana. Para ello, se accede a la herramienta y la vista es la siguiente:

Figura 57: Dashboard Kibana I. Fuente: Elaboración propia.



En la tabla de logs se observa cómo aparecen todas las alarmas que Suricata ha detectado, incluyendo si están permitidas (alert) o si han sido denegadas (drop).

Figura 58: Dashboard Kibana II. Fuente: Elaboración propia.

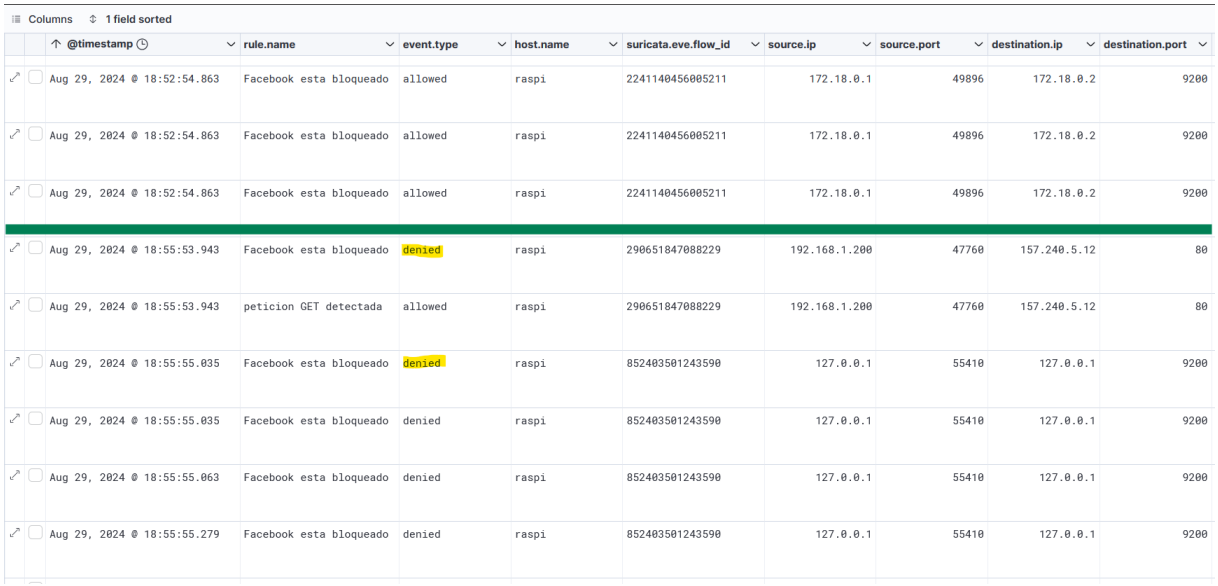
Columns 1 field sorted

@timestamp	rule.name	event.type	host.name	suricata.eve.flow_id	source.ip	source.port	destination.ip	destination.port
Aug 29, 2024 @ 18:47:25.208	ICMP Packet found	allowed	raspi	255682190061186	192.168.1.44	0	192.168.1.200	0
Aug 29, 2024 @ 18:47:25.208	ICMP Packet found	allowed	raspi	255682190061186	192.168.1.200	0	192.168.1.44	0
Aug 29, 2024 @ 18:47:49.794	ICMP Packet found	allowed	raspi	787570941567249	192.168.1.201	0	192.168.1.200	0
Aug 29, 2024 @ 18:47:49.795	ICMP Packet found	allowed	raspi	787570941567249	192.168.1.200	0	192.168.1.201	0
Aug 29, 2024 @ 18:48:26.210	ICMP Packet found	allowed	raspi	524761895155211	192.168.1.43	0	192.168.1.200	0
Aug 29, 2024 @ 18:48:26.210	ICMP Packet found	allowed	raspi	524761895155211	192.168.1.200	0	192.168.1.43	0
Aug 29, 2024 @ 18:49:19.983	peticion GET detectada	allowed	raspi	2164795160613973	192.168.1.200	43448	103.224.182.248	80
Aug 29, 2024 @ 18:50:42.416	peticion GET detectada	allowed	raspi	1882216382744949	192.168.1.200	46628	142.250.201.67	80
Aug 29, 2024 @ 18:51:47.898	Conexion SSH detectada	denied	raspi	586802710947907	192.168.1.201	62247	192.168.1.200	22
Aug 29, 2024 @ 18:52:53.466	Facebook esta bloqueado	allowed	raspi	1138491264405678	192.168.1.200	40052	157.240.5.12	80

Rows per page: 500

También se observa el cambio realizado en la regla de facebook como en la Figura 56, separado con la línea verde el momento en el que se ha cambiado la regla de alert a drop.

Figura 59: Dashboard Kibana III. Fuente: Elaboración propia.



Columns 1 field sorted

@timestamp	rule.name	event.type	host.name	suricata.eve.flow_id	source.ip	source.port	destination.ip	destination.port
Aug 29, 2024 @ 18:52:54.863	Facebook esta bloqueado	allowed	raspi	2241140456005211	172.18.0.1	49896	172.18.0.2	9200
Aug 29, 2024 @ 18:52:54.863	Facebook esta bloqueado	allowed	raspi	2241140456005211	172.18.0.1	49896	172.18.0.2	9200
Aug 29, 2024 @ 18:52:54.863	Facebook esta bloqueado	allowed	raspi	2241140456005211	172.18.0.1	49896	172.18.0.2	9200
Aug 29, 2024 @ 18:55:53.943	Facebook esta bloqueado	denied	raspi	290651847088229	192.168.1.200	47760	157.240.5.12	80
Aug 29, 2024 @ 18:55:53.943	peticion GET detectada	allowed	raspi	290651847088229	192.168.1.200	47760	157.240.5.12	80
Aug 29, 2024 @ 18:55:55.035	Facebook esta bloqueado	denied	raspi	852403501243590	127.0.0.1	55410	127.0.0.1	9200
Aug 29, 2024 @ 18:55:55.035	Facebook esta bloqueado	denied	raspi	852403501243590	127.0.0.1	55410	127.0.0.1	9200
Aug 29, 2024 @ 18:55:55.063	Facebook esta bloqueado	denied	raspi	852403501243590	127.0.0.1	55410	127.0.0.1	9200
Aug 29, 2024 @ 18:55:55.279	Facebook esta bloqueado	denied	raspi	852403501243590	127.0.0.1	55410	127.0.0.1	9200

Tras estas pruebas, se puede concluir que tanto Suricata, como la comunicación de este con Elastic Stack funcionan correctamente. Cabe destacar que Kibana es muy configurable, y que el usuario puede editar el dashboard a su gusto para conseguir una vista más personalizada del contenido mostrado.

6. CONCLUSIONES Y TRABAJO FUTURO

6.1. CONCLUSIONES

En un contexto donde la seguridad en las redes domésticas se vuelve cada vez más crítica debido al aumento de dispositivos conectados y las crecientes amenazas de seguridad, este trabajo ha abordado la necesidad de desarrollar una solución de seguridad accesible y efectiva para los usuarios comunes. La solución propuesta se ha basado en la utilización de una Raspberry Pi como plataforma central para integrar diversas herramientas de código abierto, configuradas la mayor parte mediante contenedores Docker, con el fin de proteger la red doméstica de posibles vulnerabilidades.

El problema principal identificado fue la falta de medidas de seguridad robustas en redes domésticas, que a menudo dependen de configuraciones predeterminadas y carecen de la protección necesaria contra accesos no autorizados, *malware* y otras amenazas. Este trabajo ha abordado este problema mediante la implementación de un sistema que combina filtrado DNS (Pi-hole), una VPN (WireGuard), y un sistema de detección y prevención de intrusiones (Suricata), todo ello gestionado a través de una interfaz visual utilizando Elastic Stack.

La solución desarrollada no solo ofrece una mejora de la seguridad y privacidad de la red, sino que también asegura que los usuarios puedan implementarla sin necesidad de conocimientos técnicos avanzados, gracias a la utilización de Docker, y a las guías simplificadas proporcionadas, que facilita la replicación y actualización del entorno de seguridad. Los resultados obtenidos han demostrado que, mediante el uso de herramientas de código abierto y hardware de bajo costo, es posible ofrecer una protección completa que cumple con los objetivos planteados al inicio del proyecto.

6.2. RELACIÓN ENTRE OBJETIVOS INICIALES Y RESULTADOS OBTENIDOS

A lo largo de este proyecto, se han cumplido satisfactoriamente todos los objetivos planteados al inicio salvo uno de ellos, que se ha cumplido parcialmente. Cada uno de estos objetivos intermedios fue diseñado para contribuir al objetivo principal de desarrollar una solución de seguridad integral para redes domésticas. A continuación, se presenta la relación entre los objetivos y los resultados:

3.2.1 Investigar los riesgos a los que están expuestos los usuarios de internet dentro de su red doméstica. Para este objetivo, se realizó una revisión exhaustiva de la literatura y el análisis de fuentes actualizadas sobre ciberseguridad, y se identificaron las principales amenazas que afectan a una red doméstica, las cuales se definen en el apartado 2.3.3 Posibles amenazas en una red doméstica.

3.2.2 Investigar la arquitectura de una red doméstica, para comprender su estructura y funcionamiento. En este caso, se realizó un análisis detallado de la arquitectura típica de una red doméstica, lo que incluyó la identificación de puntos críticos de seguridad y la forma en que los diferentes dispositivos se interconectan. Esta comprensión fue fundamental para diseñar una solución que abordara los puntos débiles de la red. En los apartados 2.2 ARQUITECTURA DE UNA RED DOMÉSTICA y 2.3.2 Panorama actual de la seguridad en el hogar, se refleja dicho contenido.

3.2.3 Investigación de posibles alternativas a Raspberry Pi para la realización de este proyecto. En este caso, el apartado 2.5.2 Alternativas de hardware, refleja el hardware que se valoró, sus ventajas y desventajas, y los motivos por los que finalmente se eligió la Raspberry Pi.

3.2.4 Buscar y crear una guía de buenas prácticas para una configuración segura de la red doméstica. Este punto es un extra de seguridad para ofrecer a los usuarios, se hizo una investigación sobre las mejores configuraciones y las más recomendadas, y se expuso en el apartado 4 BUENAS PRÁCTICAS DE CONFIGURACIÓN DE LA RED DOMÉSTICA.

3.2.5 Desarrollar una arquitectura de seguridad utilizando una Raspberry Pi como plataforma central. 3.2.6 Diseñar una solución completa que integre tecnologías avanzadas de seguridad como las siguientes: Estos dos objetivos se han cumplido puesto que se ha conseguido de forma exitosa implementar toda la estructura propuesta inicialmente dentro de la Raspberry Pi. Todo el detalle de su implementación puede verse en los apartados 5.2 DESCRIPCIÓN DE LAS HERRAMIENTA SOFTWARE UTILIZADAS y 5.3 PRUEBAS DE FUNCIONALIDAD.

3.2.7 Llevar a cabo toda esta configuración mediante el uso de contenedores Docker que puedan replicarse fácilmente por usuarios sin conocimientos técnicos. Este objetivo se ha cumplido parcialmente puesto que se ha implementado la configuración de todas las herramientas, salvo Suricata mediante contenedores Docker, lo que permite que la solución sea fácilmente replicable por cualquier usuario, independientemente de su nivel de conocimientos técnicos. Para el caso de Suricata + Filebeat, se ha proporcionado una guía paso a paso para facilitar su configuración, ya que no se ha conseguido implementar dentro de un contenedor Docker de manera que funcione correctamente en ambos modos IDS + IPS, solamente funcionó como IDS. Este contenido se encuentra en el apartado 5.2 DESCRIPCIÓN DE LAS HERRAMIENTA SOFTWARE UTILIZADAS.

3.2.8 Redactar una memoria de TFM correcta que refleje todo el trabajo de investigación realizado, así como los avances obtenidos. La memoria del TFM ha sido redactada de manera exhaustiva, detallando cada fase del proyecto, desde la investigación inicial hasta la implementación y validación de la solución. Este documento refleja fielmente el alcance del trabajo realizado y los logros obtenidos.

6.3.CONTRIBUCIONES DEL TRABAJO

Las principales contribuciones de este trabajo se resumen en los siguientes puntos:

- Implementación de una solución completa de seguridad para redes domésticas: Se ha logrado desarrollar una solución que integra múltiples capas de seguridad, abordando diversas amenazas informáticas a las que están expuestas las redes domésticas modernas.
- Accesibilidad y usabilidad: La solución ha sido diseñada para ser fácilmente replicable y utilizable por usuarios con conocimientos limitados en ciberseguridad, lo que amplía su aplicabilidad en entornos domésticos donde la seguridad es crítica pero los recursos y conocimientos son limitados.
- Optimización para hardware de bajo costo: El uso de una Raspberry Pi como plataforma central demuestra que es posible implementar medidas de seguridad

avanzadas sin la necesidad de invertir en hardware costoso, lo que hace que la solución sea accesible a un mayor número de usuarios.

- Flexibilidad y escalabilidad mediante Docker: La utilización de contenedores Docker ha permitido crear un entorno de seguridad modular, fácilmente actualizable y adaptable a diferentes configuraciones de red, asegurando así la longevidad y efectividad de la solución en el tiempo.

6.4. TRABAJO FUTURO

Aunque este proyecto ha alcanzado los objetivos planteados, existen varias líneas de trabajo futuro que podrían aportar valor añadido y mejorar aún más la solución propuesta:

- Monitorizar todo el tráfico de la red local: Como se comentaba en el apartado 5.3.3, actualmente Suricata solo monitoriza el tráfico con origen/destino su dirección IP. Una posible mejora sería conseguir que monitorizase todo el tráfico de la red local.
- Integración con SIEM (*Security Information and Event Management*): Para mejorar la monitorización y análisis de eventos de seguridad, sería valioso integrar la solución con un sistema SIEM, lo que permitiría una gestión centralizada de incidentes de seguridad y un análisis más profundo de las amenazas.
- Automatización de actualizaciones y mantenimiento: Asegurar que las herramientas de seguridad estén siempre actualizadas es crucial. Un trabajo futuro podría centrarse en automatizar el proceso de actualización y mantenimiento del sistema, minimizando la intervención del usuario y reduciendo la posibilidad de vulnerabilidades debido a software desactualizado.
- Desarrollo de una interfaz gráfica simplificada: Aunque la solución actual es funcional, desarrollar una interfaz gráfica más intuitiva podría mejorar la experiencia del usuario, haciendo que la configuración y el monitoreo de la red sean aún más accesibles para personas sin conocimientos técnicos.
- Expansión de la Arquitectura a Redes Empresariales Pequeñas: El modelo de seguridad implementado podría adaptarse para su uso en pequeñas y medianas empresas (PYMEs), donde la necesidad de seguridad es igualmente crítica, pero donde los recursos para implementar soluciones comerciales pueden ser limitados.

- Análisis de Rendimiento y Optimización: Una línea futura podría centrarse en analizar en profundidad el rendimiento de la solución bajo diferentes cargas de trabajo y optimizar los componentes más exigentes en términos de recursos, para asegurar que el sistema siga siendo eficiente incluso en entornos con un gran número de dispositivos conectados.

Por último, este proyecto ha demostrado la viabilidad de implementar una solución de seguridad integral en redes domésticas utilizando herramientas de código abierto y hardware accesible. A medida que la tecnología avanza y las amenazas digitales se vuelven más sofisticadas, el trabajo futuro podrá expandir y mejorar esta base, proporcionando a los usuarios mayores niveles de protección y usabilidad.

REFERENCIAS BIBLIOGRÁFICAS

Kaspersky Lab. (2024). *Cómo configurar una red doméstica segura*. https://latam.kaspersky.com/resource-center/preemptive-safety/how-to-set-up-a-secure-home-network

Arduino®. (05 de Febrero de 2018). *What is Arduino?* https://www.arduino.cc/en/Guide/Introduction

AWS. (Diciembre de 2023). *¿Cuál es la diferencia entre Docker y una máquina virtual?* https://aws.amazon.com/es/compare/the-difference-between-docker-vm/

AWS. (Diciembre de 2023). *¿Cuál es la diferencia entre imágenes y contenedores de Docker?* https://aws.amazon.com/es/compare/the-difference-between-docker-images-and-containers/

Berenguer Falcó, J. (13 de Julio de 2024). *Amenazas más comunes en redes domésticas.* https://www.redeszone.net/noticias/seguridad/amenazas-seguridad-redes-domesticas/

Cisco Systems, I. (20 de Abril de 2022). *¿Qué hace un router?* https://www.cisco.com/c/es_mx/solutions/small-business/resource-center/networking/how-does-a-router-work.html

Cloudflare, Inc. (2024). *The free app that makes your Internet safer.* https://one.one.one.one/

D., C. (30 de Mayo de 2024). *¿Qué es Docker y cómo funciona?* https://www.hostinger.es/tutoriales/que-es-docker

DanieloTech. (18 de Febrero de 2024). *Si tienes una Raspberry Pi NECESITAS INSTALAR ESTO! | Docker y Portainer*. https://youtu.be/-7vvELophxU?si=LAcQF0_O3UrtwhkX&t=236

De Luz, S. (19 de Mayo de 2024). *Duck DNS, el mejor servicio DNS dinámico gratuito que puedes usar*. https://www.redeszone.net/tutoriales/redes-cable/duck-dns-servicio-dns-dinamico-gratuito/

De Luz, S. (24 de Mayo de 2024). *Rendimiento de WireGuard*. https://www.redeszone.net/tutoriales/vpn/wireguard-vpn-configuracion/#277350-rendimiento-de-wireguard

Delaney, J. (3 de Agosto de 2024). *The Best Wi-Fi Routers for 2024.* https://www.pcmag.com/picks/the-best-wireless-routers

Díaz Amorín, J. (Junio de 2023). *Estudio de la adopción de los estándares de seguridad en redes WiFi domésticas.* E.T.S.I. de Sistemas Informáticos (UPM): https://oa.upm.es/74773/

Docker Inc. (2024). *Docker*. https://www.docker.com/

Docker Inc. (2024). *Docker Compose overview*. https://docs.docker.com/compose/

Docker Inc. (2024). *Install Docker Engine on Debian*. https://docs.docker.com/engine/install/debian/

ElPais. (28 de Agosto de 2024). *El Pais El Periódico Global*. https://elpais.com/#

Grupo ADSLZone. (2024). *Cuál es mi IP: ¿cómo saber mi dirección IP? válido para IPv4 o IPv6*. https://www.cual-es-mi-ip.net/

Guanotoa Chuma, A. M. (22 de Febrero de 2024). *Implementación de un firewall de siguiente generación que minimice el riesgo que corren los niños al navegar por internet en una red doméstica [Tesis de pregrado, Universidad Técnica del Norte].* https://repositorio.utn.edu.ec/handle/123456789/15568

Hurtado, D. F. (15 de Marzo de 2024). *Manual de buenas prácticas de seguridad informática en redes domésticas. [Monografia].* Repositorio Institucional UNAD.: https://repository.unad.edu.co/handle/10596/39430

Leos Rivas, M. (5 de Abril de 2017). *Securing the Home IoT Network.* SANS: https://www.sans.org/white-papers/37717/

linuxserver.io. (2024). *linuxserver/duckdns*. https://hub.docker.com/r/linuxserver/duckdns

MARCA. (28 de Agosto de 2024). *Diario online líder en información deportiva*. https://www.marca.com/

Méndez Colmenares, D. (2023). *Cómo mejorar la seguridad de tu red doméstica*. https://www.ceupe.com/blog/seguridad-de-tu-red-domestica.html

OISF. (2024). *Suricata Rules*. https://docs.suricata.io/en/latest/rules/index.html

Ookla, LLC. (2024). *SPEEDTEST*. https://www.speedtest.net/es

Orange Pi. (2024). *Orange Pi 5 Pro*. http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-5-Pro.html

Pastor, J. (20 de Agosto de 2024). *Raspberry Pi 5 ya tiene un nuevo modelo. Uno más barato que persigue la estela del "ordenador de los 25 dólares"*. https://www.xataka.com/ordenadores/raspberry-pi-5-tiene-nuevo-modelo-uno-barato-que-persigue-estela-ordenador-25-dolares

Penalva, J. (13 de Abril de 2023). *Intel NUC 13 Pro Arena Canyon, análisis*. xataka: https://www.xataka.com/analisis/intel-nuc-13-pro-arena-canyon-analisis-caracteristicas-precio-especificaciones

Pi-hole®. (3 de Diciembre de 2022). *Pi-hole documentation*. https://docs.pi-hole.net/

Pi-hole®. (Agosto de 2024). *pihole/pihole*. https://hub.docker.com/r/pihole/pihole

Portainer. (2024). *Effortless Container Management for Docker and Kubernetes*. https://www.portainer.io/

Portainer. (2024). *Install Portainer CE with Docker on Linux*. https://docs.portainer.io/getting-started/install-ce/server/docker/linux

Proofpoint Inc. (2024). *Proofpoint Emerging Threats Rules*. https://rules.emergingthreats.net/

Raspberry Pi. (2024). *Raspberry Pi OS*. https://www.raspberrypi.com/software/

Raspberry Pi Foundation. (2024). *Computing for everybody*. Raspberry Pi: https://www.raspberrypi.org/about/

Raspberry Pi Ltd. (Abril de 2024). *Raspberry Pi 5 - Product brief*. https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf

RaspiPC.es. (2024). *Kit de Inicio Raspberry Pi 5 8GB*. https://www.raspipc.es/2036

Sicajá, L. F., Morales, V. A., Velásquez, J. G., León, M., & Velásquez, N. O. (s.f.). *EL IMPACTO DE LAS REDES DOMÉSTICAS EN LA SEGURIDAD IOT THE ROLE OF HOME NETWORKS IN IOT SECURITY.* https://www.researchgate.net/profile/Marvin-Leon-3/publication/376033661_Impacto_redes_domesticas_en_seguridad_IoT/links/6567d5d2b1398a779dc70b17/Impacto-redes-domesticas-en-seguridad-IoT.pdf

TP-Link, E. (04 de Agosto de 2020). *¿Qué es la seguridad de la red doméstica?* https://www.tp-link.com/es/blog/585/-que-es-la-seguridad-de-la-red-domestica-/

Vallejo, A. (28 de Abril de 2022). *Ocho consejos para que nuestra WiFi sea más segura en casa*. https://www.xatakahome.com/la-red-local/ocho-consejos-nuestra-wifi-sea-segura-casa

Wikipedia. (27 de Julio de 2024). *Docker*. https://es.wikipedia.org/wiki/Docker_(software)#cite_note-infoq-201303-8

WireGuard Easy. (Agosto de 2024). *Using WireGuard Easy with Pi Hole*. https://github.com/wg-easy/wg-easy/wiki/Using-WireGuard-Easy-with-Pi-Hole

WireGuard®. (2022). *WireGuard - Fast, modern, secure VPN tunnel*. https://www.wireguard.com/

Anexo A. LISTA DE ABREVIATURAS

- CCMP - Counter Mode Cipher Block Chaining Message Authentication Code Protocol
- DHCP - Dynamic Host Configuration Protocol
- DMZ - Demilitarized Zone
- DNS - Domain Name System
- GPIO – General-purpose input/output
- GPU - Graphics Processing Unit
- HDMI – High-Definition Multimedia Interface
- HTTP - Hypertext Transfer Protocol
- IDS – Intrusion Detection System
- IoT – Internet of Things
- IoT – Internet of things
- IPS – Intrusion Prevention System
- ISO - International Organization for Standardization
- MitM – Man in the middle
- MQTT - MQ Telemetry Transport
- NUC - Next Unit of Computing
- RAM - Random Access Memory
- RTSP - Real Time Streaming Protocol
- SAE - Simultaneous Authentication of Equals
- SIEM - Security Information and Event Management
- TKIP - Temporal Key Integrity Protocol
- USB - Universal Serial Bus
- VLANs - Virtual Local Area network
- VM – Virtual Machine
- VPN – Virtual Private Network
- WEP - Wired Equivalent Privacy
- WPA - Wi-Fi Protected Access
- WPA2 - Wi-Fi Protected Access II
- WPA3 - Wi-Fi Protected Access III

Anexo B. compose.yaml

```
services:
  wg-easy:
    environment:
      # Change the server's hostname (clients will connect to):
      - WG_HOST=ejemploURLparaProyecto.duckdns.org

      # Change the Web UI Password:
      - PASSWORD=changeme

      # This is the Pi-hole Container's IP Address
      - WG_DEFAULT_DNS=10.8.1.3
      - WG_DEFAULT_ADDRESS=10.8.0.x
    image: ghcr.io/wg-easy/wg-easy
    container_name: wg-easy
    volumes:
      - '~/.wg-easy:/etc/wireguard'
    ports:
      - "51820:51820/udp"
      - "51821:51821/tcp"
    restart: unless-stopped
    cap_add:
      - NET_ADMIN
      - SYS_MODULE
    sysctls:
      - net.ipv4.ip_forward=1
      - net.ipv4.conf.all.src_valid_mark=1
    networks:
      wg-easy:
        ipv4_address: 10.8.1.2

  pihole:
    image: pihole/pihole
    container_name: pihole
    environment:
      # Change the Web UI Password:
      - WEBPASSWORD=changeme
    volumes:
```

```
      - '~/.pihole/etc-pihole:/etc/pihole'
      - './pihole/.pihole/etc-dnsmasq.d:/etc/dnsmasq.d'
    ports:
      - "53:53/tcp"
      - "53:53/udp"
      - "5353:80/tcp"
    restart: unless-stopped
    networks:
      wg-easy:
        ipv4_address: 10.8.1.3

  duckdns:
    image: lscr.io/linuxserver/duckdns:latest
    container_name: duckdns
    network_mode: host #optional
    environment:
      - PUID=1000 #optional
      - PGID=1000 #optional
      - TZ=Europe/Madrid
      - SUBDOMAINS=ejemploURLparaProyecto #Cambiar por el Subdominio deseado
      - TOKEN=ejemploTOKENparaProyecto #Cambiar por el Token deseado
      - UPDATE_IP=ipv4 #optional
      - LOG_FILE=false #optional
    volumes:
      - './duckdns/config:/config' #optional
    restart: unless-stopped

  ElasticSearch:
    container_name: ElasticSearch
    image: docker.elastic.co/ElasticSearch/ElasticSearch:8.7.1
    environment:                    ['CLI_JAVA_OPTS=-Xms1g                    -
Xmx1g','bootstrap.memory_lock=true','discovery.type=single-
node','xpack.security.enabled=false', 'xpack.security.enrollment.enabled=false']
    ports:
      - 9200:9200
    networks:
      - elastic
    volumes:
      - ./elkStack/ElasticSearch/database:/usr/share/ElasticSearch/database
```

```
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536
        hard: 65536
    deploy:
      resources:
        limits:
          cpus: '1.0'
        reservations:
          cpus: '0.5'

  kibana:
    image: docker.elastic.co/kibana/kibana:8.7.1
    container_name: kibana
    ports:
      - 5601:5601
    networks:
      - elastic
    deploy:
      resources:
        limits:
          cpus: '1.0'
        reservations:
          cpus: '0.5'
    depends_on:
      - ElasticSearch

  filebeat:
    image: docker.elastic.co/beats/filebeat:8.0.0
    container_name: filebeat
    user: root
#    volumes:
#      - './filebeat/config/filebeat.yml:/usr/share/filebeat/filebeat.yml'
    networks:
      - elastic
    command: filebeat -e -strict.perms=false
```

```
    depends_on:
      - ElasticSearch
      - kibana

networks:
  wg-easy:
    ipam:
      config:
        - subnet: 10.8.1.0/24
  elastic:
```