Safia Guellil

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN EN UNA RED DOMÉSTICA CON DISPOSITIVOS DE BAJO COSTE.

TRABAJO DE FIN DE GRADO

dirigit per Dr. Jordi Castellà Roca

Grado de Ingeniería Informática

UNIVERSITAT ROVIRA I VIRGILI

Tarragona
2022

Resum.

Amb l'arribada del COVID19 a Espanya i la posterior quarantena, Internet ha estat un recurs essencial que ens ha permès comunicar-nos amb la resta del món i mantenir vius els serveis tradicionals. Això ha implicat un gran salt cap a la digitalització, el qual ha convertit la connectivitat en un servei tan bàsic i preuat com ho és la llum o l'aigua. No obstant això, mentre que uns se centraven en buscar una cura, els ciberdelinqüents han aprofitat aquest salt per a arribar a més víctimes, fins a tal punt que qualsevol usuari connectat remotament estaria exposat al perill. Arran d'aquesta problemàtica, s'ha dissenyat i implementat un sistema de monitoratge i detecció d'intrusos a nivell domèstic, enfocat principalment a detectar els atacs més patits a Espanya durant els últims dos anys, com són el phishing i les botnets. A més, permet visualitzar les mètriques analitzades mitjançant taulers i panells que són accessibles des de qualsevol dispositiu connectat a la xarxa. Així mateix, incorpora un sistema d'alertes visuals i auditives que advertiran del succés d'una amenaça en temps real. Finalment, tot el sistema està implementat utilitzant dispositius de baix cost, els quals són fàcilment accessibles per a qualsevol usuari.

Resumen.

Con la llegada del COVID19 en España y la posterior cuarentena, Internet ha sido un recurso esencial que nos ha permitido comunicarnos con el resto del mundo y mantener vivos los servicios tradicionales. Esto ha implicado un gran salto hacia la digitalización, el cual ha convertido la conectividad en un servicio tan básico y preciado como lo es la luz o el agua. Sin embargo, mientras que unos se centraban en buscar una cura, los ciberdelincuentes han aprovechado este salto para llegar a más víctimas, hasta tal punto que cualquier usuario conectado remotamente estaría expuesto al peligro. A raíz de esta problemática, se ha diseñado e implementado un sistema de monitorización y detección de intrusos a nivel doméstico, enfocado principalmente en detectar los ataques más padecidos en España durante los últimos dos años, como son el phishing y las botnets. Además, permite visualizar las métricas analizadas mediante tableros y paneles que son accesibles desde cualquier dispositivo conectado a la red. Asimismo, incorpora un sistema de alertas visuales y auditivas que advertirán del suceso de una amenaza en tiempo real. Finalmente, todo el sistema está implementado utilizando dispositivos de bajo coste, los cuales son fácilmente accesibles para cualquier usuario.

Abstract.

With the arrival of COVID19 in Spain and the subsequent quarantine, the Internet has been an essential resource that has allowed us to communicate with the rest of the world and keep traditional services alive. This has implied a great leap towards digitalization, which has turned connectivity into a service as basic and precious as electricity or water. However, while some focused on finding a cure, cybercriminals have taken advantage of this leap to reach more victims, to such an extent that any remotely connected user would be exposed to danger. As a result of this problem, a monitoring and intrusion detection system has been designed and implemented at a domestic level, focused mainly on detecting the attacks most suffered in Spain during the last two years, such as phishing and botnets. In addition, it allows to visualize the analyzed metrics through dashboards and panels that are accessible from any device connected to the network. It also incorporates a system of visual and audible alerts that will warn of the occurrence of a threat in real time. Finally, the entire system is implemented using low-cost devices, which are easily accessible to any user.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

La declaración del Estado de Alarma en España, el 14 de marzo de 2020, ha sido una situación inesperada que nos ha cogido a todos desprevenidos y desconcertados sobre el futuro. Se trata de un periodo caracterizado por una larga cuarentena que nos hubiera aislado del resto del mundo si no fuera por el gran invento de hace 4 décadas: Internet.

Si bien es cierto que las consecuencias de la pandemia han sido verdaderamente devastadoras, a distintos niveles socioeconómicos, también es evidente que ha impulsado un gran salto hacia la digitalización [24]. El *Estudio de tendencias de recursos humanos*, elaborado por CEOE y Randstad durante octubre del 2021, revela que 2 de cada 3 empresas (64%) consideran que han progresado mucho tecnológicamente y que todavía pueden mejorar [35].

Es más, con la llegada de la cuarentena se normaliza una medida de flexibilidad del trabajo que ha revolucionado a miles de empresas y la vida de cientos de ciudadanos, efectivamente, el teletrabajo. Citando al escritor Stephen Covey, "*El equilibrio entre vida y trabajo es sin duda una de las luchas más significativas encaradas por el hombre moderno."*, se ha convertido en toda una realidad al desplazar el trabajo al hogar.

La pandemia no solamente ha afectado al trabajo. De hecho, no cabe mencionar la gran carencia que ha destapado con la brecha digital en la educación [1], la cual casi paraliza por completo el proceso de aprendizaje de los más jóvenes, si no fuera por las rápidas medidas implementadas para impulsar la alterativa desde los hogares: educación on-line.

En efecto, podríamos hacer un gran listado de todos los campos que la digitalización ha sido un recurso necesario para la supervivencia de los servicios tradicionales durante la pandemia. Hasta tal punto que la conectividad en las casas se ha convertido en un bien tan preciado y necesario para la población, tanto como la electricidad o el agua. Este último hecho se evidencia aún más con los datos de Telefónica, en la cual la demanda de banda ancha creció hasta un 40% [24] en los inicios del confinamiento.

No obstante, no todo es color de rosa. Mientras que el mundo se enfocaba en encontrar una cura para el COVID-19 e invertían sus recursos para poner fin a la pandemia, los ciberdelincuentes oportunistas aprovechaban este impuso digital en beneficio personal. Una situación que ha llegado hasta tal punto que el año 2020 se ha considerado uno de los años más peligrosos en cuanto a ciberdelincuencia [9]. De hecho, según el informe de "*Ciberamenazas y tendencias*" del 2021, durante el 2020 el Centro Criptológico Nacional (CCN-CERT), ha registrado casi el doble de ataques catalogados de peligrosidad muy alta respecto al año anterior. Por otro lado, también tenemos al Instituto Nacional de Ciberseguridad (INCIBE) que ha gestionado 133.155 incidentes, de los cuales el 40% eran de peligrosidad Alta y el 14% Muy Alta y Crítica [9].

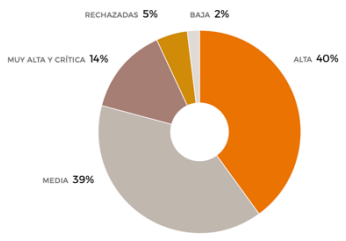


Figura 2. Distribución de incidentes según su criticidad. Informe Ciberamenazas y tendencias 2021CCN_CERT.

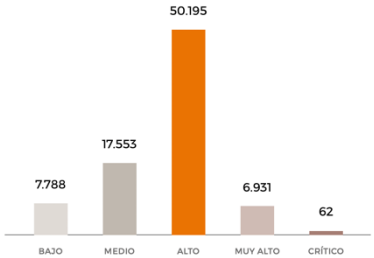


Figura 1. Distribución ciberincidentes de 2020 según la peligrosidad. Informe Ciberamenazas y tendencias 2021CCN_CERT.

Parece que conectarse a Internet desde casa nunca había sido tan peligroso. Muchos usuarios creen que no son objetivo de los atacantes cuando el mero hecho de estar conectados remotamente te expone al riesgo.

Un ejemplo sería Emotet, uno de los principales ataques llevados a cabo durante el 2020 [9], el cual se ha introducido en los hogares mediante herramientas tan habituales como son la ofimática o el correo electrónico. Este enviaba emails Phishing[1] con un Word o enlace incrustado que, al instalarse en el host, iniciaría la descarga de otros malware más peligrosos y se extendería por toda la red local.

Una vez infectado, el host pasaría a formar parte de una red de botnet[2] C2[3] que controlaría su máquina remotamente para distintos fines, como son: la criptominería, sustracción de datos y credenciales, inclusive fines más perversos como es la instauración de material pornográfico infantil que podría imputarnos un delito.

[1] *Phishing*: técnicas para engañar a los usuarios, principalmente mediante la suplantación de identidad.
[2] *Botnet*: red compuesta por una gran cantidad de hosts infectados que controla el atacante.
[3] *C2 o Command and Control*: servidores que se emplean para controlar el malware infestado en los hosts.

1.1. Objetivos

El gran salto hacia la digitalización ha aumentado el uso de Internet para distintos fines. Sin embargo, también ha propiciado el incremento de cibercrímenes, de tal manera que se hace necesario un sistema de monitorización y detección de amenazas en nuestro entorno doméstico.

A continuación, mostramos los objetivos del prototipo de monitorización planteado:

1.1.1. Objetivos Generales

1. Diseñar e implementar un prototipo de sistema de monitorización de la red, en un entorno doméstico, que estaría compuesto de tres subsistemas:
    a. *Sistema de Detección de Intrusos en la Red* (NIDS).
    b. *Sistema de Gestión de Información y Eventos de Seguridad* (SIEM).
    c. Sistema de Alertas.
2. Elaborar una metodología de detección de amenazas en relación con el contexto actual de los ciberataques.
3. Ejecutar el prototipo en dispositivos de bajo poder de cómputo, consumo y coste.
4. Evaluar los subsistemas en una arquitectura adecuada para un entorno local.

1.1.2. Objetivos Específicos

1. Diseñar un sistema modular y escalable, fácilmente adaptable al entorno y con dispositivos reemplazables según las necesidades de la red local.
2. Analizar si el hardware seleccionado es suficiente para la implementación del prototipo.
3. Implementar un sistema de visualización comprensible con dos modos: uno sencillo y otro más extensivo.
4. Incorporar un sistema de alertas visuales y auditivas.

1.2. Justificación

Actualmente, para la prevención de estos ataques disponemos de muchas herramientas, como son los antivirus o los corta fuegos. Aun así, no siempre son suficientes, ya que las amenazas podrían evadir los corta fuegos, los antivirus pueden no estar actualizados, puede aparecer un nuevo virus e incluso, muchas veces somos nosotros mismos quienes facilitamos la instauración de la amenaza sin ser conscientes de ello. De hecho, el informe anual del 2022 *State of the Phish* revela que el 83% alegan haber recibido algún mensaje sospechoso y el 42% de haber cometido alguna acción peligrosa como descargar software o hacer clic en dominios desconocidos [1].

Es más, Emotet se ha convertido en uno de los ataques más propiciados [9], puesto que sus correos electrónicos eran suficientemente creíbles para que el usuario confiara en el remitente y descargara el contenido malicioso. Acto que se acentúa si el remitente era la cuenta de una anterior víctima de robo de credenciales. En este caso el ataque no se detendría en convertir el host en un zombi, sino que se abría paso en los distintos dispositivos conectados a la red local.

Vista esta situación se vuelve imprescindible tener un sistema de monitorización y detección de amenazas (o intrusos) a nivel de red. Este debe estar constantemente evaluando nuestro entorno en busca de amenazas e informarnos y alertarnos al detectar una.

1.3. Planificación

El proyecto inicia en mayo con una primera fase de investigación y definición, y culmina en setiembre con la entrega de la memoria. En general, la planificación ha sido la siguiente:

1. *Mayo*: este primer mes nos hemos centrado en definir el proyecto, los objetivos (generales y específicos) y sus características.
2. *Junio*: a continuación, esta ha sido una fase de investigación y diseño del prototipo.
3. *Julio*: una vez establecido el diseño, hemos iniciado la implementación.
4. *Agosto*: finalmente, en este último mes hemos terminado la implementación e iniciado la evaluación y testeo del prototipo final.

Cabe mencionar que hemos utilizado una metodología de desarrollo secuencial (o waterfall[4]) en la cual no iniciábamos una fase o la implementación de un sistema hasta haber evaluado la anterior. Esta metodología ha sido necesaria debido a la naturaleza del proyecto. Se trata de implementar tres subsistemas, los cuales unos dependen de otros, es decir, si el sistema de detección no funciona, no funcionará tampoco el sistema de alertas o visualización. Por lo tanto, es necesario terminar de implementar el sistema anterior, antes de iniciar con el siguiente. De hecho, la memoria está organizada de esta manera, después de cada implementación existe una evaluación.

[4] https://www2.deloitte.com/es/es/pages/technology/articles/waterfall-vs-agile.html

1.4. Organización de la memoria

La memoria se agrupa en estas 4 secciones:

1.4.1. Background

En este primer apartado, se definen algunos conceptos y herramientas de seguridad introductorias para este proyecto. También se presenta el panorama general de las ciberamenazas en España.

1.4.2. Diseño y Arquitectura

Esta segunda sección comprende principalmente el diseño del arquetipo, los requisitos funcionales y las decisiones de diseño en relación con los tres subsistemas. También, analizamos distintas metodologías de detección de amenazas, especificamos la información necesaria para la visualización y definimos las características del sistema de alertas.

1.4.3. Implementación y Evaluación

En este tercer apartado, detallamos todo el proceso de puesta en producción del prototipo. Por lo tanto, implementamos la arquitectura diseñada en un entorno local y activamos los sistemas de detección, visualización y alertas. Además, debido a la metodología seguida y diseño modular, la evaluación se realiza durante y después de cada implementación, de esta manera al pasar a la siguiente etapa del proyecto garantizamos que la anterior está correcta y funcional.

1.4.4. Conclusiones

Finalmente, comentamos los resultados obtenidos en la sección anterior y planteamos el trabajo futuro.

2. Background

Iniciamos la memoria con unas definiciones sobre algunos conceptos y herramientas de seguridad relacionadas con este proyecto. Además de analizar el panorama general de la ciberseguridad en España. Esta sección está compuesta por los siguientes apartados: i) Intrusión en la red; ii) Clasificación de ataques; iii) Panorama Actual de la Ciberamenazas en España; iv) Ataques botnet; v) Sistema de Detección de intrusos (IDS); vi) Indicadores de Compromiso y Ataques; vii) Sistema de Gestión de la Información y Eventos de Seguridad (SIEM).

2.1. Intrusión en la red

Una intrusión o ataque es un evento que aprovecha las vulnerabilidades de un sistema para provocar un daño o acceder a la información sin el consentimiento del usuario. En general, un sistema se puede considerar seguro cuando puede garantizar: **confidencialidad**, **integridad**, **autenticidad** y **disponibilidad**. Aun así, lamentablemente, los atacantes informáticos siempre encuentran nuevas herramientas para vulnerar los sistemas.

De forma general, y teniendo en cuenta el objetivo del ataque, se podría dividir estas amenazas en 4 grandes grupos [27]:

- **Interrupción**: es un tipo de ataque en el cual un recurso deja de estar disponible para el usuario. Un ejemplo de este ataque sería el DoS o DDoS[5]. Es un ataque en contra de la disponibilidad.
- **Intercepción**: la finalidad de este tipo de ataque es la obtención de información o acceso de recursos del usuario de una forma no autorizada. Es un ataque en contra de la confidencialidad.
- **Modificación**: este ataque va más allá del anterior, no solamente accede a la información, sino que la modifica. Se trata de un ataque en contra de la integridad.
- **Fabricación**: finalmente, en este tipo de amenazas, el atacante inserta nuevos objetos en el sistema, como podrían ser, páginas web, direcciones IP, etc. Se trata de un ataque en contra de la autenticidad.

Figura 3. Tipología de Ataques. https://blogseguridadandrea.wordpress.com/

[5] DoS o DDoS: Ataque de Denegación de Servicio (DoS) o Ataque Distribuido de Denegación de Servicio (DDoS).

2.2. Clasificación de ataques

Según el daño que podemos realizar en el sistema, dividimos en general, los ataques en dos grupos [22]: **ataques pasivos** o **ataques activos**.

2.2.1. Ataques Pasivos

Este tipo de ataques son los más difíciles de detectar, ya que no alteran los paquetes. Se trata de interceptar la información mediante monitorización de la red (*Eavesdroping*) con el fin de capturar y obtener datos valiosos como pueden ser contraseñas o recursos de interés [22].

Los objetivos principales de este ataque son la extracción de información y la monitorización de la red de una forma sutil, mediante el análisis del tránsito de la red, como pueden ser [22]:

- Leyendo las cabeceras de las comunicaciones (origen-destino).
- Control del volumen del tráfico.
- Control de las horas habituales de mayor comunicación.

Este ataque claramente va en contra de la confidencialidad del sistema, pero se puede mitigar con el cifrado de los paquetes mediante TLS.

2.2.2. Ataques Activos

A diferencia de los ataques pasivos, estos ataques alteran de alguna manera el sistema, modificando así el flujo de los datos o creando unos falsos.

Estos ataques se pueden subdividir en 4 tipos [22]:

- **Suplantación de identidad**: se suplanta la identidad por el usuario legítimo con la finalidad de acceder a un recurso gracias a los privilegios de este. Un ejemplo de este ataque podría ser IP Spoofing, en el cual se modifica el paquete con una IP de origen falsa.
- **Reactuación**: se trata de un tipo de ataque en el cual un mensaje legítimo es capturado y repetido varias veces con la intención de conseguir algún efecto no deseado.
- **Modificación de mensajes**: en este caso el ataque consiste en modificar el mensaje legítimo, reordenarlo o retardarlo con el objetivo de producir un efecto no deseado.
- **Degradación Fraudulenta del servicio**: este ataque consiste en no permitir al usuario legítimo el normal uso de un recurso, ya sea mediante la supresión de las peticiones que este usuario realiza a una entidad o directamente el uso de técnicas para la interrupción del servicio. Entre estos ataques encontramos el DoS o DDoS.

2.3. Panorama Actual de la Ciberamenazas en España

En el siguiente apartado mostraremos estadísticas en relación con las ciberamenazas a nivel de España. Estos datos se obtuvieron de los últimos informes (2020 y 2021) del Centro Criptológico Nacional Computer Emergency Response Team[6] (CCN-CERT) y del Sistema Estadístico de Cibercriminalidad[7] (SEC).

2.3.1. Ciberamenazas 2020

Durante el 2020, el INCIBE-CERT ha registrado 133.155 incidencias de seguridad, de las cuales destacan el Malware con un 35,2% y el Fraude (Estafas) con un 32,02% (***Figura 4***) [9]. Si analizamos los incidentes Malware más impactantes que hemos sufrido en España a lo largo de la cuarentena han sido [9]:

1. **Emotet**: un ataque botnet que enviaba campañas de correos electrónicos de phishing con archivos adjuntos maliciosos. Estos al ser descargados por la víctima se ejecutaban códigos (principalmente de monitorización) que conseguían obtener información dentro de los navegadores de la víctima, así como credenciales y datos bancarios, entre otros.
2. **Trickbot**: se trata de otro ataque botnet que empleaba correos electrónicos de phishing en el cual añadía ficheros maliciosos o adjuntaba enlaces para descargarlos. Los trickbots eran conjuntos de herramientas de criptominería, sustracción de datos o enumeración de hosts.
3. **Ryuk**: empleaba ataques como Emotet o de Trickbots para cifrar archivos o sistemas esenciales para el objetivo, de tal manera que luego puedan pedir un rescate.
4. **Metasploit**: ataques que han utilizado el programa open source Metasploit, el cual permite realizar ataques de prueba de penetración y ejecución de código de exploración.
5. **Powershell Empire**: ha permitido evadir soluciones de seguridad y actuar de forma encubierta en los sistemas de las víctimas, de tal manera que puedan manejar los equipos permitiendo el control total al atacante.

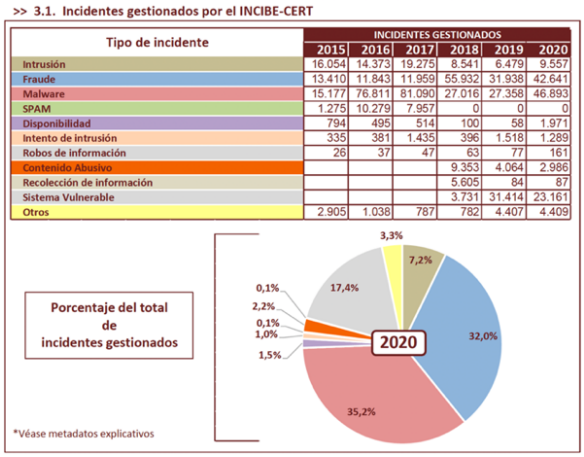


Figura 4. Incidentes gestionados por el INCIBE-CERT 2020.
https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/5377-ccn-cert-ia-13-20-ciberamenazas-y-tendencias-edicion-2020/file.html

[6] https://www.ccn-cert.cni.es/
[7] https://estadisticasdecriminalidad.ses.mir.es/publico/portalestadistico/

2.3.2. Ciberamenazas 2021

Por otro lado, en este último año el INCIBE-CERT ha gestionado aproximadamente un 18% menos de incidencias en comparación con el año anterior, con un total de 109.126 incidencias [35]. Dentro de las cuales vuelven a destacar el Malware y el Fraude con unos valores de 29,9% y 28,6%, respectivamente (**Figura 5**).

En cuanto a los incidentes Malware más destacados este último año [35]:

1. **Emotet**: se trata del mismo ataque botnet mencionado en la sección anterior. Durante el 2021 aún ha seguido vigente.
2. **Mekotio**: (o BestaFera) es un ataque dirigido principalmente a la banca online. Se instaura al igual que Emotet mediante campañas de correos electrónicos Phishing con archivos maliciosos adjuntos que, al infectar el host, provocan ataques dirigidos al sector bancario.
3. **Flubot**: un tipo de ataque destinado principalmente a dispositivos Android. Envía SMS maliciosos suplantando a empresas de logística, como son FedEx, DHL o Correos, invitándoles a descargar software malicioso para el seguimiento del paquete. Este último programario inyecta páginas superpuestas de tal manera que el usuario introduce sus credenciales en sitios maliciosas controlados por los atacantes.
4. **Anatsa**: también conocido como TeaBot o Toddler es un tipo de ataque también destinado a dispositvios Android con una metodología similar a Flubot.
5. **Hive**: se trata de un ataque tipo randsomware que cifra la información del equipo infestado imposibilitando así el acceso a los datos.
6. **Suplantación de identidad (Fraude al CEO)**: principalmente se trata de envíos de correos electrónicos maliciosos personalizados para cada víctima. Estos solicitan información sensible, como son datos personales o bancarios.
7. **Phishing**: por último, se trata de una práctica que engaña a la víctima para que comparta credenciales, datos personales o acceda a sitios peligrosos.

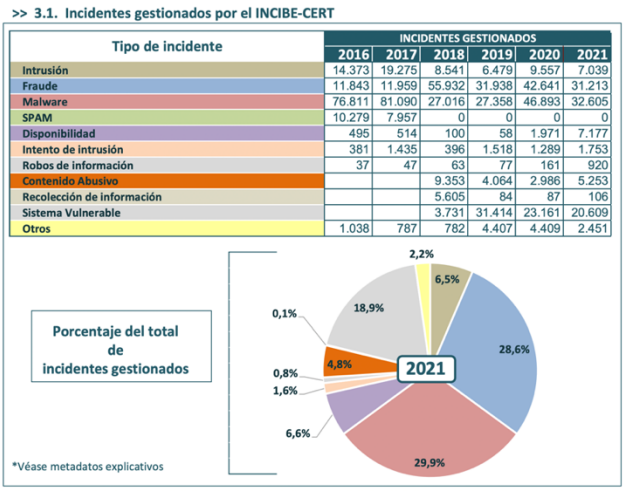


Figura 5. Incidentes gestionados por el INCIBE-CERT 2021. https://www.interior.gob.es/opencms/pdf/prensa/balances-e-informes/2021/Informe-Cibercriminalidad-2021.pdf

2.3.3. Emotet

En la sección anterior, hemos visto algunos de los ataques más frecuentes durante estos dos últimos años, en especial, destacamos los **ataques botnet** como **Emotet** que se ha mantenido en la posición número uno de los ataques con mayor relevancia y efectos.

Emotet es un malware que se propaga principalmente a través de campañas de correos electrónicos maliciosos que ha permitido la expansión de botnets a lo largo de estos últimos años. Este ataque emplea el phishing e ingeniería social para que la víctima confíe y descargue el contenido adjunto o haga clic en un enlace malicioso siendo esta la semilla inicial del ataque botnet.

Se trata de un tipo de ataque que no se dirige a un público en específico, puesto que ha afectado tanto a civiles, empresas y entidades gubernamentales [26].

2.3.4. Phishing

Otro denominador común que hemos visto en los ataques comentados anteriormente ha sido el Phishing. Se trata de un conjunto de técnicas enfocadas en engañar a la víctima, principalmente, ganándose su confianza mediante la suplantación de identidad de terceros, ya sean personas, empresas o entidades. Esta confianza la consigue mediante Ingeniería Social, la cual es una disciplina que emplea la manipulación para conseguir información confidencial de los usuarios [12].

Actualmente, se han detectado y clasificado diferentes tipos de técnicas phishing, dependiendo del objetivo, el fin, el medio o modo de operación [13]. Algunos tipos de ataques phishing que podríamos recibir al conectarnos en nuestra red local podrán ser:

1. **Phishing General o Spray and pray**: consiste en enviar correos spam a los usuarios suplantando la identidad de terceros, e incluyendo información en el mensaje como son URLs maliciosas.
2. **URL Phishing**: se trata de camuflar una URL maliciosa para que parezca de un sitio legítimo. Por ejemplo, *http://www.google.com@members.tripod.com/* este enlace tiene la apariencia de ir a Google, pero en realidad se redirige a la página members.tripod.com.
3. **Business Email Compromise**: son ataques dirigidos a empleados mediante el uso de ingeniería social y correos electrónicos que suplantan la identidad de la empresa o un miembro de rango superior de la misma. Con el teletrabajo estos ataques se reciben en el hogar y son más difíciles de constatar al no estar en el ámbito labora habitual.
4. **Pharming**: es un tipo de ataque que intenta redirigir el tráfico web de la víctima a sitios fraudulentos. Se consigue mediante la explotación de vulnerabilidades del sistema del usuario o los sistemas de nombre de dominios (DNS).
5. **Malware-based phishing**: son ataques phishing que consiguen ejecutar algún software en el dispositivo de la víctima. Por ejemplo, este software puede ir incluido en el adjunto de un correo malicioso.

6. **Content-Injection phishing**: en este tipo de phishing los atacantes consiguen remplazar parte del contenido de un sitio legítimo para obtener información del usuario.
7. **Smishing**: es similar al phishing, pero el engaño se utiliza herramientas de mensajería de texto como son SMS o WhatsApp.

2.4. Ataques Botnet

Muchos ataques como Emotet o Trickbots terminan implicando nuestra máquina en una red botnet, la cual perdemos el control de nuestra máquina.

Una botnet proviene de "*ro**bot net**work*" y significa red de robots. Los equipos que componen una botnet se llaman bots, robots o equipos zombi, los cuales se encuentran unidos a través de Internet al pastor de robots que controla toda la red para llevar a cabo distintos ataques en masa [26].

2.4.1. Formación de una botnet

Como hemos mencionado en la Introducción, cualquier dispositivo conectado a la red está expuesto al riesgo y puede convertirse en víctima, en este caso formando parte de una botnet. Este ataque se compone de 3 fases [26]:

1. ***Infectar a la víctima***: en esta primera fase se introduce el malware de la botnet en el sistema objetivo. Para lograrlo existen diferentes técnicas, entre ellas destaca el uso del phishing, spam e Ingeniería social, los cuales convencen al usuario de cometer una acción peligrosa como es descargar un software o hacer clic en un enlace o dominio.
2. ***Ampliar la botnet***: después de infectar el host, el siguiente paso es ampliar la red. En este caso, ya no solo peligra el dispositivo conectado, sino que toda la red local. Existen diferentes metodologías de ampliación de la red dependiendo del atacante, estas podrían ser desde ataques diccionario y fuerza bruta, hasta enviar a los contactos de la víctima enlaces de distribución de malware con la suplantación de identidad.
3. ***Activar al botnet***: una vez la botnet se ha extendido lo suficiente, inicia el ataque en masa utilizando la potencia de cómputo de las máquinas de las víctimas para distintos fines.

2.4.2. Consecuencias de una botnet

Después de infectar a la víctima, el pastor robot puede realizar distintos tipos ataques, tanto activos como pasivos. A continuación, listamos algunos de los ataques [26]:

1. ***Espiar a la víctima y robar datos personales***: se trata de un ataque pasivo el cual está destinado a recopilar información sensible de la víctima, como podrían ser datos bancarios hasta imágenes comprometedoras.

2. ***Criptominería***: es un ataque activo que aprovecha la capacidad de cómputo de los equipos infectados para la explotación de criptomonedas.

3. ***Infectar con más malware***: descarga más malware al equipo de la víctima, siendo el precursor de otros ataques más específicos.

4. ***Campañas phsihing***: este ataque activo suplanta la identidad de la víctima con la finalidad de utilizar sus recursos, como es el correo electrónico, para esparcir campañas de spam o phishing con el objetivo de infestar a más equipos.

2.5. Sistemas de Detección de Intrusos

Un Sistema de Detección de Intrusos o IDS (*Instrusion Detection System*) es un mecanismo de monitorización del tráfico y eventos de la red, que se emplea con la finalidad de detectar posibles amenazas o intrusiones que puedan comprometer el sistema.

Las funciones principales de un IDS es analizar los paquetes en busca de patrones diseñados que coincidan con ciertas amenazas. Normalmente se instala en dispositivos con duplicación de puerto con el propósito de monitorizar y alertar al administrador sin afectar el flujo de datos [38].

2.5.1. Elementos de un IDS

Los elementos básicos que configuran un IDS son los siguientes [27]:

1. **Fuente de Recolección de datos**: un sistema que se encarga de recolectar los datos para el posterior análisis, como pueden ser, ficheros log o paquetes de la red, entre otras fuentes.

2. **Reglas de contenido de datos**: son los patrones que determinan la estructura para detectar una amenaza.

3. **Filtros**: nos ayudan a delimitar en el análisis de paquetes.

4. **Detectores de eventos anormales en el tráfico de red**: otorga al IDS su funcionalidad de detección de intrusos.

5. **Dispositivo Generador de informes y alarmas**: sensores o alarmas para notificar al administrador en caso de detectar amenazas.

2.5.2. Métodos de Identificación

A continuación, vamos a ver diferentes clases de IDS en función del método que utilizan para detectar amenazas [38]:

- **IDS basado en Firmas**: es un sistema que detecta amenazas en base a un patrón conocido, también llamado firma o signatura. Se trata de un método bastante eficaz en el caso de amenazas conocidas, pero le resulta imposible detectar amenazas no conocidas, ya que necesitaría un patrón previo para identificarlas.

- **IDS heurístico o basado en anomalías**: a diferencia del método anterior, este sistema puede llegar a detectar amenazas desconocidas, gracias al aprendizaje automático. Aun así, no es 100% eficiente, ya que podría clasificar actividades legítimas como amenazas, también llamadas falsos positivos.

- **IDS basado en reputación**: se trata de un sistema que rastrea la reputación de un archivo para lograr identificar ficheros maliciosos.

2.5.3. Clasificación sistemas IDS

Los principales sistemas de detección de intrusos según la arquitectura son [38]: el sistema de detección de intrusos basado en red (NIDS) y el sistema de detección de intrusos basado en host (HIDS).

2.5.3.1. Sistemas de detección de Intrusos basado en Red (NIDS)

Es un sistema que monitoriza el tráfico de la red en busca de anomalías. Por lo tanto, analiza un gran número de paquetes de todos los hosts que están conectados a la red.

Normalmente, suele situarse en un dispositivo por el cual atraviesa todo el tráfico como es un enrutador o HUB. Se trata básicamente de un sistema que implementa un Sniffer que captura y analiza todo el tránsito de la red a diferentes capas del modelo OSI: TCP/IP, transporte e incluso aplicación.

Estos sistemas utilizan port mirroring (o puerto espejo) para obtener una copia de todo el tráfico. Esta funcionalidad se obtiene mediante un TAP (Test Access Point) insertado en un lugar específico en la red [38].

2.5.3.2. Sistemas de detección de Intrusos basado en Host (HIDS)

Este sistema a diferencia del anterior se centra únicamente en detectar las amenazas de un host. Por lo tanto, detecta y analiza las amenazas internas que este podría padecer. Este método lo utilizan, por ejemplo, los antivirus y detecta las intrusiones mirando los datos de los archivos del sistema, tales como los logs y configuraciones.

2.6. Indicadores de Compromiso y Ataque

2.6.1. Indicadores de Compromiso

Los Indicadores de Compromiso (IOC) son evidencias de incidentes relacionados con la ciberseguridad que pueden ayudar a los sistemas de detección a identificar las amenazas. Estas evidencias o patrones se extraen del análisis de los incidentes y pueden tener diferentes formatos como son: ficheros creados, listado de dominios, IPs, payloads, servicos, entre otros [11].

Implementar estos indicadores en un Sistema de Detección de Intrusos permitirá detectar diferentes tipos de amenazas conocidas que han sido registradas y reconocidas por sus signaturas. Sin embargo, estos indicadores no permiten detectar nuevas amenazas.

2.6.2. Indicadores de Ataque

A diferencia de los Indicadores de Compromiso (IOC), los Indicadores de Ataque (IOA) se preocupan de las acciones que se deben llevar a cabo para que un ataque tenga éxito. Por ejemplo, en los ataques que hemos visto anteriormente, como es el caso de Emotet, el procedimiento sería abrir un correo phishing, hacer clic en un enlace de distribución de malware, este se descarga, se ejecuta en segundo plano y se conecta a la red botnet esperando instrucciones [15]. Observamos que en ningún momento se considera el tipo de malware, su signatura, IP, entre otros datos.

Figura 6. Indicadores de Compromiso y Ataque.
https://www.welivesecurity.com/la-es/2022/03/29/que-son-indicadores-ataque/

2.7. Sistema de Gestión de la Información y Eventos de Seguridad

SIEM o Security Information and Event Management es un sistema de administración en tiempo real que permite gestionar una gran cantidad de información proveniente de diferentes fuentes como son sensores, IDS/IPS, cortafuegos, servidores, aplicaciones, ente otros. Además, correlaciona los datos con el fin de encontrar vulnerabilidades e información significativa.

El SIEM es el resultado de combinar las siglas SIM (administración de información de seguridad) y SEM (administración de eventos de seguridad).

El uso del SIEM es bastante común en las organizaciones, ya que les permite monitorizar y analizar eventos en busca de amenazas y vulnerabilidades. Además, les ofrece un seguimiento de los datos que les ayuda en procesos de auditoría o cumplimento de políticas [3].

2.7.1. Funcionamiento de un SIEM

Todas las soluciones SIEM difieren en características, políticas, puntos de control entre otros aspectos, pero en general comparten un conjunto básico de funciones [3]:

1. **Gestión de registros**: normalmente tienen una o varias fuentes de registros que procesar. Estas fuentes provienen de diferentes nodos, ya sea datos de los usuarios, de la red o incluso entornos en la nube. Toda esta información se almacena en tiempo real y de forma automática en una ubicación centralizada.

2. **Correlación de eventos y análisis**: la correlación es una de las funcionalidades más importantes del sistema SIEM, ya que permitirá identificar patrones complejos que ayudarán a mitigar de una forma más rápida las amenazas.

3. **Monitorización de incidentes y alertas de seguridad**: al tratarse de una solución centralizada de todos los registros, permite la monitorización de los sistemas, usuarios y dispositivos. Además de alertar al encontrar una coincidencia con patrones predefinidos.

4. **Gestión de conformidad e informes**: finalmente, sobre todo para el caso de las empresas y organizaciones, puede generar informes de conformidad y cumplimiento de las normas vigentes en tiempo real.

3. Diseño y Arquitectura

En este apartado definimos el diseño y la arquitectura del prototipo: un sistema de monitorización en una red doméstica, con dispositivos de bajo consumo.

Esta sección se distribuye de la siguiente manera: i) Requisitos Funcionales; ii) Diseño del Arquetipo; iii) Decisiones de Diseño;

3.1. Requisitos Funcionales

A continuación, listamos los requisitos necesarios para diseñar e implementar este proyecto. Al estar el compuesto de tres subsistemas diferentes (Detección-Visualización-Alertas), hemos establecido los requisitos funcionales del prototipo general, además de especificar los requisitos de cada uno de ellos.

3.1.1. Requisitos generales

En la **Tabla 1**, podéis observar los requisitos funcionales mínimos del prototipo en general:

Tabla 1. Requisitos Funcionales Generales. Elaboración propia.

#	Nombre del Requisito	Descripción	Observación
1	Sistema	Sistema escalar y modular.	Sistema compuesto por distintas pequeñas unidades que permiten la escalabilidad.
2	Arquitectura	Implementar el prototipo en una red doméstica.	El prototipo debe poder llevarse a cabo en una red local.
3	Recolectar Tráfico	Recolectar todo el tráfico que generan todos los dispositivos conectados a la red.	El tráfico debe proceder tanto de la red Wireless como por Ethernet y en ambas direcciones (peticiones-respuestas).
4	Analizar Tráfico	Analizar en tiempo real.	El análisis debe efectuarse en tiempo real con la mínima pérdida de paquetes.
5	Visualizar el análisis	Visualizar estadísticas del análisis efectuado.	La visualización debe efectuarse a través de paneles y elementos visuales.
6	Alertar	Alertar al encontrar una amenaza.	Las alertas deben efectuarse en tiempo real.
7	Dispositivos Hardware	Emplear dispositivos de bajo coste y consumo.	Los dispositivos seleccionados deben poseer la suficiente capacidad de cómputo para implementar los sistemas. Además, de un consumo bajo tanto económicamente como energéticamente.
8	Disponibilidad	Un sistema siempre disponible.	Se trata de un sistema que se ejecutará 24/7/365 monitorizando la red local.

3.1.2. Requisitos Sistema de Detección de Intrusos en la Red (NIDS)

Tabla 2. Requisitos Funcionales Sistema de Detección de Intrusos en la Red. Elaboración Propia.

#	Nombre del Requisito	Descripción	Observación
1	Análisis	Analizar todos los paquetes de la red.	Analizar el máximo número de paquetes en tiempo real con las menores pérdidas.
2	Falsos positivos	Sistema de detección preciso.	Minimizar la cantidad de falsos positivos.
3	Actualización	Actualizar las reglas de detección.	Añadir nuevas reglas periódicamente y eliminar las obsoletas.
4	Detección	Detección de las principales amenazas actuales en España.	Detección de: Botnets, Phishing, SPAM y enlaces de distribución de Malware.

3.1.2. Requisitos Sistema de Gestión de Información y Eventos de Seguridad (SIEM)

Tabla 3. Requisitos Funcionales Sistema de Gestión de Información y Eventos de Seguridad. Elaboración Propia.

#	Nombre del Requisito	Descripción	Observación
1	Agregación de datos	Agregar datos provenientes del NIDS.	Los datos se recopilarán del NIDS a través de la tarjeta de red (NIC).
2	Visualización	Visualización de las estadísticas.	Visualizar estadísticas en tiempo real a través de tableros o paneles.
3	Modos Visualización	Dos modos de visualización.	Mostrar los datos de forma resumida o en un formato más extenso.
4	Elementos Gráficos	Entorno de visualización agradable y sencillo.	Emplear elementos gráficos sencillos y fácilmente comprensibles para los usuarios.
5	Retención de datos	Retención de los datos durante un largo periodo de tiempo.	Retener los datos permite poder compararlos con eventos pasados.
6	Acceso al SIEM	Acceso al SIEM desde cualquier dispositivo conectado a la red.	Se protegerá el SIEM con unas credenciales de acceso (usuario-contraseña).

3.1.3 Requisitos Sistema de Alertas

Tabla 4. Requisitos Funcionales Sistema de Alertas. Elaboración Propia.

#	Nombre del Requisito	Descripción	Observación
1	Elementos Visuales	Alertar de forma visual cuando se detecte unas amenazas.	Emplear los colores básicos: rojo, amarillo y verde
2	Elementos Auditivos	Alertas mediante una alarma cuando se detecten amenazas importantes.	
3	Categorizar	Categorizar las alertas según el rango de la amenaza.	Utilizar elementos más destacados para más las amenazas más importantes.
4	Desactivar	Activar o Desactivar el sistema de alertas.	El sistema de alertas puede ser desactivado. Esta funcionalidad será necesaria en caso de tests.

3.2. Diseño del Arquetipo

A continuación, definimos el diseño del arquetipo propuesto. En la siguiente sección (Decisiones de diseño) detallamos más esta propuesta según nuestro entorno local y las herramientas seleccionadas.

El sistema que desarrollado es modular, es decir, empleamos diferentes elementos funcionales que unidos forman el conjunto del sistema. Este diseño proporciona una característica fundamental al proyecto: la escalabilidad. De esta manera, podemos reemplazar dichos elementos individuales por otros más o menos potentes, según nuestras necesidades, para adaptarlo a cualquier entorno sin perder la finalidad y el objetivo del conjunto.

En la **Figura 7**, observamos la arquitectura general del arquetipo. El flujo de datos y procedimientos es el siguiente:

1. Se recogen los datos de los distintos dispositivos conectados a la red, ya sea través de Wifi o Ethernet.
2. Se envía una copia del tráfico al Sistema de Monitorización.
3. El sistema de Detección de Intrusos (NIDS) se encargará de analizar los paquetes recibidos y enviará logs al Sistema de Gestión de Información y Eventos de Seguridad (SIEM).
4. El SIEM se encarga de interpretar la información recibida por el NIDS y la visualiza en tableros o paneles.
5. Finalmente, el Sistema de Alertas está pendiente de las amenazas que detecte el NIDS para hacer saltar las alertas.

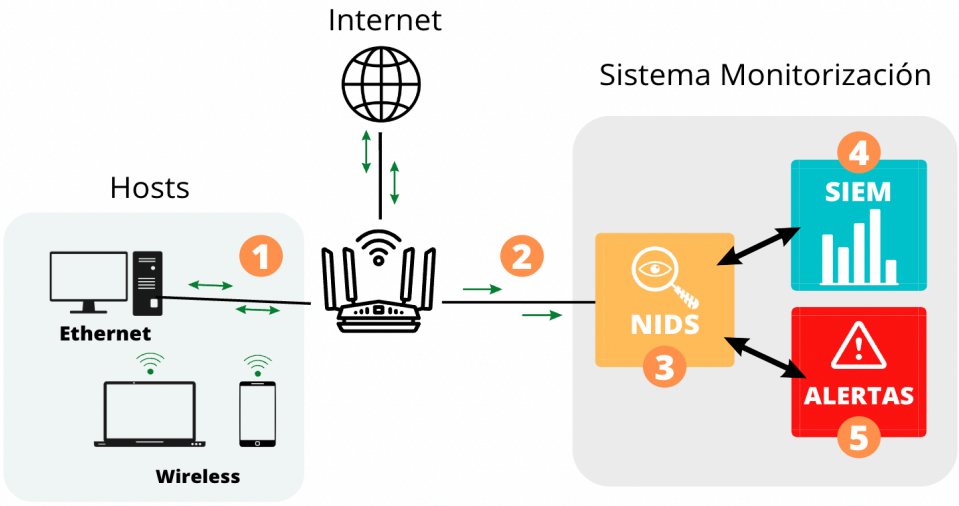


Figura 7. Arquetipo General. Elaboración Propia.

Esta es la idea principal del funcionamiento del sistema, en los próximos aparatos especificaremos las herramientas software y hardware empleadas para la implementación del prototipo.

3.3. Decisiones de diseño

En este apartado mostramos las decisiones de diseño que hemos tomado en relación con la arquitectura, el hardware y el software que emplea dicho prototipo. Por lo tanto, veremos: i) Diseño de la Arquitectura del Sistema; ii) Diseño del Sistema de Detección de Intrusos en la Red (NIDS); iii) Diseño del Sistema de Gestión de Información y Eventos de Seguridad (SIEM); iv) Diseño del Sistema de alertas; v) Diseño Final.

3.3.1. Diseño de la Arquitectura del Sistema

3.3.1.1. Arquitectura de la Red

En esta sección, vemos la adaptación que hemos efectuado en cuanto a la arquitectura para poder conseguir el cumplimiento de los requisitos funcionales definidos.

En nuestro caso, el Router del que disponemos es el modelo lowi-h500sv1 de la compañía Lowi. Este modelo no dispone de la funcionalidad de Port Mirroring[8], por lo tanto, no lo podemos configurar para enviar una copia del tráfico al sistema de monitorización.

De tal manera y con la finalidad de adaptarnos al arquetipo formulado, será necesario incorporar dos nuevos elementos:

- **Switch**: un conmutador que centralizará las comunicaciones, de tal manera que podrá enviar una copia del tráfico a un puerto especifico conectado con la NIDS.
- **Punto de Acceso**: será empleado para generar una WLAN y estará conectado al switch anterior. De esta forma podremos recibir el tráfico de los dispositivos conectados vía WiFi, ya que estos en lugar de conectarse directamente al Router (el cual tendrá desconectado el Wifi) se conectarán al Access Point.

El switch que hemos seleccionado es el modelo **Zyxel gs1200-5**. Este producto en concreto dispone de la funcionalidad port mirroring, además de una interfaz sencilla y cinco puertos (los cuales son más que suficientes para nuestro entorno). También, esta versión en específico se ha empleado en otros proyectos de IDS como es el de Stéphane Potier (3 febrero 2021)[9], por lo tanto, es perfecto para nuestra implementación.

Por otro lado, el dispositivo seleccionado para ejercer de Access Point ha sido el repetidor **ASUS RP-AC51**. Este modelo se puede utilizar como Punto de Acceso, Repetidor o Puente de Medios. Además, es de doble banda (2,4Ghz y 5Ghz), el estándar de seguridad es WPA2 + AES y tiene una velocidad de transmisión de hasta 750 Mbps. En definitiva, es perfecto, ya que sus características son semejantes a las que hasta ahora disfrutábamos con el Wifi del Router y no perjudicará la calidad del servicio de Internet.

[8] *Port Mirroring*: o en español "puerto espejo", se trata de una funcionalidad inteligente de algunos conmutadores o routers que permiten enviar una copia del tráfico desde unos puertos origen hacia unos puertos destino.
[9] https://jufajardini.wordpress.com/2021/02/15/suricata-on-your-raspberry-pi/

Figura 9. Zyxel gs1200-5.
https://www.zyxel.com/

Figura 8. ASUS RP-AC51.
https://www.asus.com/

Por último, con la finalidad de controlar mejor los diferentes dispositivos, les asignamos una IP estática. En la siguiente tabla podéis observar el resumen de IP y el tipo de dispositivo:

Los dispositivos principales del sistema son:

Tabla 5. Listado de IPs dispositivos principales. Elaboración Propia.

Dispositivo	IP
Router	192.168.1.1
Switch	192.168.1.3
Access Point	192.168.1.4
NIDS	192.168.1.16
SIEM	192.168.1.17
Computador Principal	192.168.1.10

Los dispositivos secundarios son:

Tabla 6. Listado IPs dispositivos secundarios. Elaboración Propia.

Dispositivo	IP
Ordenador 1	192.168.1.11
Móvil 1	192.168.1.20
Móvil 2	192.168.1.21
Móvil 3	192.168.1.22
Móvil 4	192.168.1.23
Móvil 5	192.168.1.24
Tablet 1	192.168.1.30
Tablet 2	192.168.1.31
IOT 1	192.168.1.32
Play Station	192.168.1.33

3.3.1.2. Dispositivos Hardware del Sistema de Monitorización

A continuación, decidimos que hardware de bajo consumo utilizamos para la implementación del sistema de monitorización.

Hoy en día, en el mercado podemos encontrar una gran variedad de dispositivos diminutos y asequibles con la capacidad de cómputo suficiente para implementar una diversidad de proyectos. Sin embargo, una de las placas más vendidas[10] en los últimos meses ha sido la Raspberry Pi 4 modelo B.

La Raspberry Pi es un pequeño computador de bajo coste diseñado en el Reino Unido por la Raspberry Pi Foundation[3] y lanzado por primera vez a inicios del 2012[4]. El objetivo de este pequeño dispositivo es poner al alcance de todos la informática o como bien describe la fundación en su página principal: "*Computing for everybody. From industries large and small, to the kitchen table tinkerer, to the classroom coder, we make computing accessible and affordable for everybody*."

Entre las diversas ventajas que podríamos encontrar para este artefacto, destacamos las siguientes:

1. Dispositivo de bajo coste.
2. Tamaño reducido, ideal para la movilidad y la disposición del sistema en cualquier espacio.
3. Bajo consumo de energía eléctrica.
4. Podemos añadir diferentes dispositivos periféricos mediante los pines GPIO.
5. Finalmente, queremos emplear la máxima capacidad de cómputo para la finalidad de monitorización, utilizar otro dispositivo cotidiano como un ordenador, no sería asequible.

Selección Modelo Raspberry Pi

Actualmente la familia de la Raspberry Pi es bastante amplia, de tal forma que entre los modelos domésticos e industriales se puede encontrar más de una docena. Sin embargo, no todos los modelos son adecuados para este proyecto, ya que cada nueva versión viene con mejorías respecto a la anterior, pero a costa de un precio algo más elevado. Por lo tanto, analizamos las prestaciones de cada modelo para elegir la más adecuada en cuanto a calidad-precio.

Para la comparativa, hemos establecido estos criterios de selección:

- **Memoria RAM**: es útil tener una memoria RAM suficientemente grande para poder utilizar programas que consumen mucha memoria en su procesamiento sin ralentizar el sistema.
- **Número de Cores**: determina el grado de paralelismo de la Raspberry Pi.
- **Frecuencia de reloj**: una mayor frecuencia ejecutará más ciclos, por lo tanto, la velocidad de procesamiento de instrucciones es más rápida.
- **Alimentación**: es interesante conocer el consumo, ya que se trata de un sistema que estará siempre disponible, por lo tanto, se necesita una solución económica.
- **Fecha de lanzamiento**.

[10] https://www.amazon.com/-/es/Los-m%C3%A1s-vendidos-Computadoras-de-Placa-Reducida/zgbs/pc/17441247011

- **Precio**: los precios son orientativos, ya que siempre están sujetos a cambios como la escasez[5] de chips. (Obtenido de la página oficial del Kubbi[6] el 7 Jul 2022).

En la **Tabla 7** podemos observar algunas de las características de los principales modelos de Raspberry Pi que ofrecen en la página oficial[11]. Estos modelos son los disponibles en fecha de Julio 2022, por ejemplo, la Raspberry Pi 1 B+ ya no se encuentra disponible a la venta.

Por otro lado, hemos omitido el modelo Pico, puesto que sus prestaciones son demasiado bajas como son las 264KB de SRAM o los 133MHz de frecuencia.

También hemos omitido los modelos Zero, ya que son unicore, por lo tanto, a pesar de su alta frecuencia de reloj, no nos permitirá aplicar paralelismo.

Tabla 7. Características modelos Raspberry Pi. Elaboración Propia.

Modelo	RAM	Nº Cores	CPU Clock	Alimentación	Fecha	Precio
Raspberry Pi 1 A+	256 MB	4	700 MHz	120mA	Nov 2014	± 30€
Raspberry Pi 2 B	1 GB	4	900 MHz	5V	Feb 2015	± 40€
Raspberry Pi 3 B	1 GB	4	1.2 GHz	5V / 2.5A	Feb 2016	± 40€
Raspberry Pi 3 B+	1 GB	4	1.4 GHz	5V / 2.5A	Mar 2018	± 40-50€
Raspberry Pi 3 A+	512 MB	4	1.4 GHz	5V / 2.5A	Nov 2018	± 25-40€
Raspberry Pi 4 B	1,2,3,4,8 GB	4	1.5 GHz	5V / 3 A	Ene 2019	± 40-90€

Finalmente, el dispositivo elegido es la Raspberry Pi 4 B de 4 GB, dado que este modelo, además de ser el más vendido[12], consideramos que es adecuado para implementar el prototipo con suficiente capacidad de cómputo, velocidad y precio razonable. Por otro lado, el modelo 8GB de RAM creemos que no se llegará a aprovechar toda la capacidad que este nos brinda, además de no encontrarse disponible en la página web.

[11] https://www.raspberrypi.com/
[12] https://www.amazon.com/-/es/Los-m%C3%A1s-vendidos-Computadoras-de-Placa-Reducida/zgbs/pc/17441247011

En conclusión, el modelo 4B consta de las siguientes características[13]:

- **Procesador**: Broadcom BCM2711
- **Cores**: Quad-Core
- **Arquitectura**: SoC Cortex-A72 (ARM v8) 64-bit
- **Frecuencia de reloj**: 1.5GHz
- **Memoria RAM**: 4GB LPDDR4
- **Rango temperatura**: 0 – 50º
- **GPIO**: 40 pines.
- **Alimentación**: 5V / 3ª
- **Conectividad**: WiFi Dual Band 2.4 GHz y 5.0 GHz IEEE 802.11b/g/n/ac,
- **Bluetooth**: 5.0
- **Ethernet**: Gigabit Ethernet
- **USB**: 2 puertos USB 3.02 y dos puertos USB 2.0.

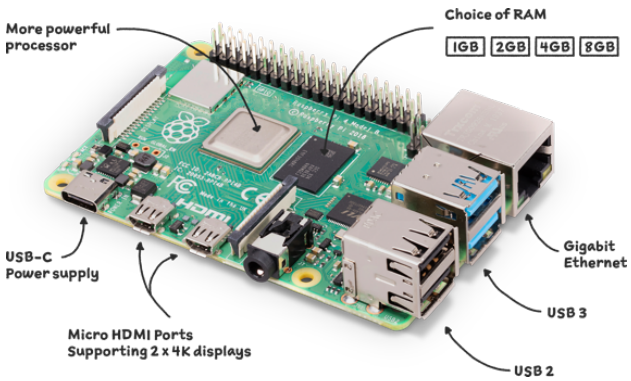


Figura 10. Raspberry Pi 4 B. https://www.raspberrypi.com/

[13] https://www.raspberrypi.com/

3.3.1.3. Arquitectura Final

Finalmente, en la **Figura 11**, mostramos la arquitectura final del proyecto. Esta comprende desde los elementos añadidos a la red para conseguir el port mirroring, hasta los dispositivos de bajo consumo que conforman el sistema de monitorización.

Al tratase de un sistema modular, cada componente dispondrá de su propia Rapsberry Pi, salvo el subsistema de alertas, puesto que este más bien está constituido por dispositivos periféricos que se encuentran pendientes de recibir señales de eléctricas provenientes del NIDS.

En general, el procedimiento es el siguiente:

1. Los dispositivos conectados vía Wifi, en lugar de conectarse directamente al router, lo harán a través del Acces Point conectado al Switch.
2. Por otro lado, los que desean conectarse vía Ethernet deberán hacerlo directamente al conmutador.
3. El switch centraliza las comunicaciones, tanto provenientes de Wifi como Ethernet, y envía una copia del tráfico al NIDS.
4. El NIDS por su parte, analiza los paquetes que le llegan del switch y envía logs al SIEM.
5. Este último, gestiona la información proveniente del NIDS y la interpreta a través de tableros y paneles.
6. Estos paneles son accesibles desde cualquier dispositivo conectado a la red local.
7. Finalmente, el NIDS envía impulsos eléctricos al sistema de Alertas cada vez que se detecte una amenaza.

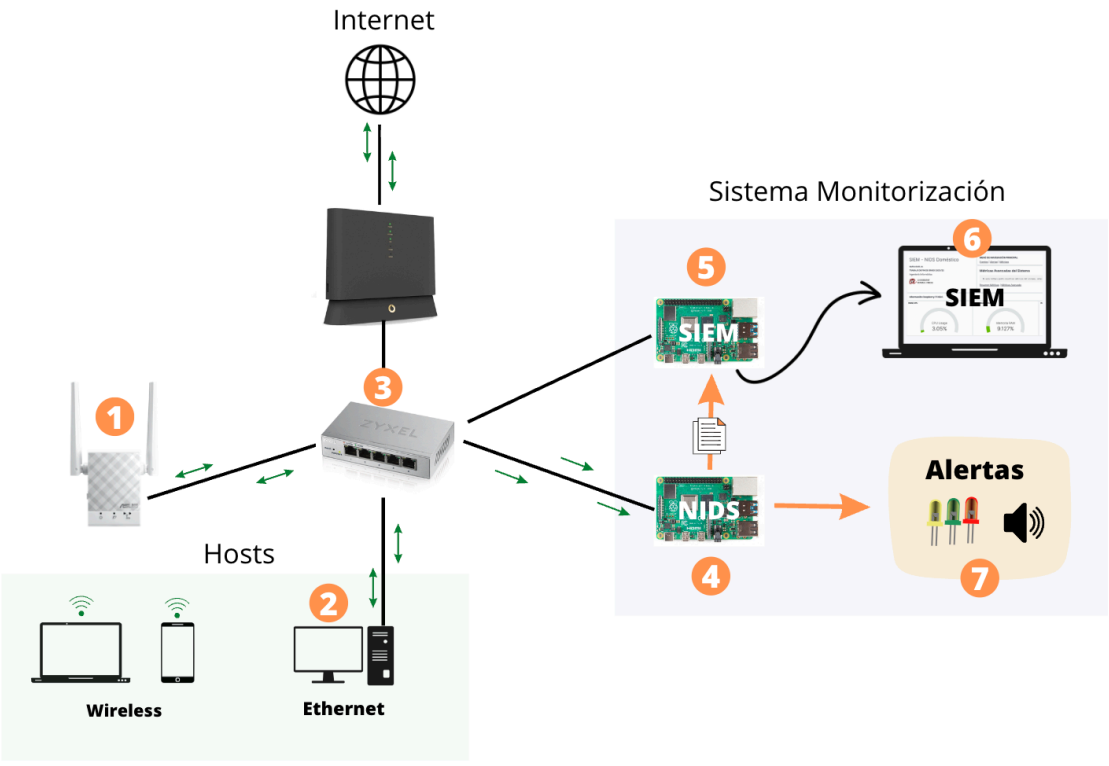


Figura 11. Arquitectura Final. Elaboración Propia.

3.3.2. Diseño del NIDS

En esta sección definimos el proceso de selección del software utilizado como sistema de detección de intrusos en la red o NIDS. Actualmente en el mercado existen diferentes opciones con más o menos características que podrían servir para la finalidad de este proyecto. Por lo tanto, hemos analizado cada una de ellas y elegido las que más se adecuen a nuestros objetivos y herramientas.

Para ello hemos elaborado la siguiente lista de criterios de selección:

1. Software Open Source.
2. Pueda ejecutarse en Linux y compatible con arquitectura ARM.
3. Un software lo más específico para un NIDS.
4. Un proyecto de software que tenga como mínimo 3-4 años de antigüedad.
5. Un proyecto activo y en constante desarrollo.

El primer criterio está seleccionado en base a que el prototipo debe ser medianamente económico. Además, con una rápida búsqueda en Internet, podemos ver diferentes proyectos prometedores desarrollados y mantenidos por y para la comunidad. Por lo tanto, es un criterio que se puede cumplir.

Por otro lado, el segundo criterio está más relacionado con el hardware que utilizaremos. En este caso se trata de la Raspberry Pi 4 B que emplea una arquitectura ARM además el SO es Debian GNU/Linux. De modo que se trata de un requisito fundamental, ya que no todos los proyectos software son compatibles con esta arquitectura.

El tercer criterio es que el software ha de ser lo más específico para un NIDS. Este criterio ha sido elegido, ya que buscamos un programa que esté centrado principalmente en este nicho, lo cual hará que sus prestaciones estén muy centradas hacia lo que buscamos.

Además, este software debe tener alrededor de 3 o 4 años de antigüedad, básicamente por la experiencia que se adquiere durante estos años. Esta les ayuda a mejorar e implementar un proyecto bastante completo en base a las necesidades que han ido surgiendo desde la comunidad.

Finalmente, el último criterio y uno de los más importante es que sea un proyecto bastante activo. Esto evidencia que se trata de un software vivo que se mantiene en contacte mejoría y actualización.

Una vez establecidos los criterios, el siguiente paso ha sido seleccionar un conjunto de herramientas y filtrar en base a estas condiciones para seleccionar el NIDS final.

3.3.2.1. Selección NIDS

La **Tabla 8**, podéis observar un conjunto de herramientas IDS obtenida de un estudio de comparación del 2018 [25]. Al ser un estudio del mercado de 2018, estos proyectos (si siguen activos), de entrada, cumplen con el criterio número 4 (experiencia y antigüedad). Aun así, se ha corroborado con las páginas oficiales de cada software.

Tabla 8. Comparativa IDS. Kumar, A., & Shrivastava, S. (2018, 30 octubre). Comparative Study on Network Vulnerabilities and Intrusion Detection System. ijrar. http://ijrar.com/upload_issue/ijrar_issue_20542796.pdf

IDS	HIDS/NIDS	Unix	Linux	Windows	Mac OS
SolarWinds Log and Event Manager	Both	No	No	Yes	No
Snort	NIDS	Yes	Yes	Yes	No
OSSEC	HIDS	Yes	Yes	Yes	Yes
Suricata	NIDS	Yes	Yes	Yes	Yes
Bro	NIDS	Yes	Yes	No	Yes
Sagan	Both	Yes	Yes	No	Yes
Security Onion	Both	No	Yes	No	No
AIDE	HIDS	Yes	Yes	No	Yes
Open WIPS-NG	NIDS	No	Yes	No	No
Samhain	HIDS	Yes	Yes	No	Yes
Fail2Ban	HIDS	Yes	Yes	No	Yes

Una vez obtenido un listado de candidatos aceptables, iniciamos con los descartes:

1. Primeramente, descartamos todos los HIDS, ya que el criterio de selección número 3 establece que deben ser NIDS, estos son: *OSSEC*, *AIDE*, *Samhain* y *Fail2Ban*.

2. En segundo lugar, descartamos los sistemas no compatibles con Linux incumpliendo así el requisito número 2. En este caso sería únicamente *SolarWinds Log and Event Manager*. Este, además, no cumple con el requisito número 1, ya que actualmente no es de Código Abierto.

3. También descartamos *Open WIPS-NG*, ya que si observamos el repositorio Git[14], vemos que la última actualización ha sido en noviembre del 2018, por lo tanto, incumple el requisito número 5. Además, está más enfocado en actuar como IPS en conexiones WiFi, lo cual es entendible puesto que pertenece al set de Aircrack-ng[15].

4. Por otro lado, también descartamos *Security Onion*, ya que se trata de un sistema no compatible con la arquitectura ARM[16] (criterio número 2). Además, implementarlo iría en contra del requisito funcional genérico "*Sistema escalable y modular*" (ver Requisitos Funcionales), puesto que actualmente Security Onion es todo un SIEM que incorpora Suricata y Zeek para la detección y Elastich search, Kibana, entre otras herramientas para el análisis y visualización de las alertas y eventos.

[14] https://github.com/aircrack-ng/OpenWIPS-ng
[15] https://www.aircrack-ng.org/
[16] https://docs.securityonion.net/en/2.3/hardware.html

5. Otro candidato que no implementaremos es *Bro(Zeek)*[17], básicamente porque está más centrado en la monitorización y no en la detección. Aun así, es un software bastante bueno ya que crea registros que ayudan a detectar anomalías.
6. Finalmente, de las herramientas restantes, compararemos las fechas de los últimos lanzamientos para seleccionar los proyectos más activos:
    a. Snort[18]: Snort v3.1.38.0 del 28 de julio del 2022.
    b. Suricata[19]: Surcata v6.0.6 del 12 de julio del 2022.
    c. Sagan: Sagan v2.0.2 del 29 de diciembre del 2021.

    Como podéis observar todos tienen fechas de lanzamiento en verano del 2022, salvo Sagan, por lo tanto, lo descartamos.

3.3.2.2. Snort vs Suricata

Después del proceso de filtrado anterior, hemos obtenido dos posibles candidatos: Suricata y Snort. Sin embargo, solamente podemos implementar uno. Ambos programas tienen características bastante similares, se trata de proyectos activos, enfocados a proteger la red y con un sistema de Detección Basado en Signaturas. Por lo tanto, los analizamos a nivel de rendimiento y uso de recursos del sistema. Para ello, nos hemos basado en el artículo "*Comparative Study of Snort 3 and Suricata Intrusion Detection Systems*" [20]. Esta investigación compara la última versión de Snort (snort3) con Suricata.

Observamos en los resultados obtenidos como Snort consume menos recursos que Suricata, pero a su vez detecta menos alertas, alrededor de la mitad que Suricata. También observamos que, a pesar de consumir más recursos, Suricata no supera el 10% de uso de memoria o el 25% de CPU.

En conclusión, el software elegido ha sido Suricata gracias a su mayor capacidad de detección.

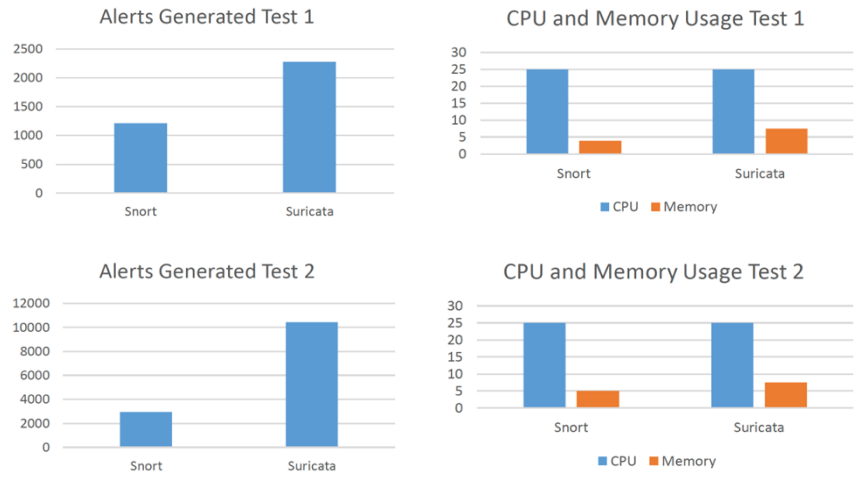


Figura 12. Comparativa entre Suricata y Snort3. Hoover, C. (2022, mayo). Comparative Study of Snort 3 and Suricata Intrusion Detection Systems. ScholarWorks@UARK. https://scholarworks.uark.edu/csceuht/105/

[17] https://zeek.org/
[18] https://github.com/snort3/snort3/releases
[19] https://github.com/OISF/suricata/releases

3.3.2.2. Suricata

Suricata[20] es un software Open Access de detección y prevención de amenazas desarrollado por Open Information Security Foundation[21] (OISF). Este programario fue lanzado por primera vez en 2009 y hasta el día de hoy se mantiene a la vanguardia con las amenazas emergentes [30].

Entre sus funciones principales destacan [R]:

1. Detección intrusos (*IDS - Intrusion Detection*).
2. Prevención de intrusos (*IPS - Intrusion Prevention*).
3. Monitorización de la red (*NSM - Network Security Monitoring*).
4. Procesamiento de paquetes (*PCAP - Packet Capture*).

Figura 13. Logo Suricata. https://suricata.io/

Suricata mantiene una licencia de la GNU (*General Public License*) de **GPLv2**[22] que nos permite ejecutar, copiar, modificar y distribuir una versión mejorada para la comunidad. Por lo tanto, podemos utilizar este software para la finalidad de este trabajo de fin de grado.

Características Suricata

Suricata nos aporta una serie de características[23] muy destacables para este proyecto [30]:

1. **Monitorización completa de la red**: posee un sistema completo en la captura de paquetes que permitirá un fácil análisis de estos.
   Entre ellos destaca:
   a. Registra y analiza los certificados TLS.
   b. Registra las conexiones HTTP.
   c. Registra consultas y respuestas de DNS.
2. **Signaturas**: permite la integración de reglas o signaturas avanzadas y a la vanguardia como son ET Pro Ruleset.
3. **Detección automática del protocolo**: detecta automáticamente los protocolos en cualquier puerto.
4. **Alto rendimiento**: el sistema soporta paralelismo y es muy escalable.
5. **Output JSON**: tanto en el análisis como en la emisión de las alertas, Suricata nos retorna un fichero de salida EVE JSON el cual es muy compatible para diferentes programas de visualización.

[20] https://suricata.io/
[21] https://oisf.net/
[22] https://suricata.readthedocs.io/en/suricata-6.0.5/licenses/gnu-gpl-v2.0.html
[23] https://suricata.io/features/

Reglas

En los siguientes apartados vamos a describir cómo Suricata captura y analiza los paquetes de tal manera que pueda clasificarlos y ejercer las funciones de detección de amenazas.

Signaturas

Las reglas en Suricata o también llamadas signaturas están formadas por tres partes [2]:

Acción Cabecera Opciones

Acción: establece qué hacer cuando la regla coincide.
Cabecera: información sobre el protocolo, la IP, el puerto y dirección de la regla.
Opciones de la regla: son especificaciones adicionales a la regla.

A continuación, mostramos un ejemplo obtenido de la documentación oficial de Suricata, en el cual se delimita en rojo la acción, en verde la Cabecera y finalmente en azul las opciones de la regla.

drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +..)"; flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK .*USA.*[0-9]{3,}/i"; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)

Acciones

Las acciones establecen qué hacer cuando una signatura coincide. En general, en Suricata tenemos 4 tipos de acciones:

- *Pass*: permite pasar al paquete sin inspecciones adicionales.
- *Drop*: suelta el paquete y emite una alerta.
- *Reject*: rechaza el paquete.
- *Alert*: genera una alerta.

Al tratarse de un IDs solamente Podemos utilizer la acción "alert".

Cabecera

Esta sección inicia con el protocolo del paquete, seguidamente de la dirección de este que puede ser, o bien desde la red local network al exterior o la inversa (también puede ser en ambas direcciones especificando *any*). También podemos agregar los puertos origen y destino.

Opciones

Finalmente, esta última parte no tiene una estructura fija, ya que dependerá del protocolo, pero por lo general suele contener estos parámetros:

- *msg*: se trata de la signatura de la regla.
- *sid*: número identificador de la red.
- *classtype*: es la categoría a la cual pertenece la amenaza. Esta debe estar definida en el fichero "classification.config".
- *reference*: este parámetro indica donde encontrar más información sobre la amenaza.
- *priority*: indica la prioridad de la amenaza detectada. Siendo 1 la más alta.

Logs

Finalmente, cabe destacar que Suricata genera diferentes tipos de ficheros logs, pero principalmente destacan los tres siguientes:

- ***eve.json***: es un fichero en JSON que almacena todos los paquetes recibidos y analizado. Además, incorpora también paquetes de alerta, estados y estadísticas. En definitiva, almacena tanto los paquetes recibidos como los análisis y detecciones por parte de Suricata.


```
{
  "timestamp": "2022-09-06T00:08:12.058311+0200",
  "flow_id": 1082853359482817,
  "in_iface": "eth0",
  "event_type": "flow",
  "src_ip": "192.168.1.30",
  "src_port": 53193,
  "dest_ip": "192.168.1.1",
  "dest_port": 53,
  "proto": "UDP",
  "app_proto": "dns",
  "flow": {
    "pkts_toserver": 1,
    "pkts_toclient": 1,
    "bytes_toserver": 83,
    "bytes_toclient": 99,
    "start": "2022-09-06T00:01:57.522177+0200",
    "end": "2022-09-06T00:01:57.526855+0200",
    "age": 0,
    "state": "established",
    "reason": "timeout",
    "alerted": true
  }
}
```


Figura 14. Log de Suricata eve.json. Elaboración Propia.

- ***fast.log***: este fichero almacena las amenazas detectadas, de forma específica, es decir, la signatura, identificador, tipo de amenaza, prioridad, entre otros datos.

```
08/09/2022-23:33:08.080257 [**] [1:2210054:1] SURICATA STREAM excessive
retransmissions [**] [Classification: Generic Protocol Command Decode] [Priority:
3] {TCP} 192.168.1.12:47050 -> 31.13.66.51:5222
```

- ***stats.log***: este último fichero almacena las estadísticas generales de Suricata, como son: la cantidad de paquetes analizados y perdidos, cantidad de alertas detectadas, cantidad de paquetes de cada protocolo, entre otros.

```
-------------------------------------------------------------------------------
Counter                                     | TM Name                   | Value
-------------------------------------------------------------------------------
capture.kernel_packets                      | Total                     | 8319039
decoder.pkts                                | Total                     | 8319045
decoder.bytes                               | Total                     | 8198842017
decoder.invalid                             | Total                     | 18
decoder.ipv4                                | Total                     | 8246817
decoder.ipv6                                | Total                     | 3265
decoder.ethernet                            | Total                     | 8319045
decoder.tcp                                 | Total                     | 5621476
decoder.udp                                 | Total                     | 2619365
decoder.icmpv4                              | Total                     | 8123
```

Figura 15. Log Suricata stats.log. Elaboración Propia.

3.3.3. Diseño de Reglas de Detección

El diseño de las reglas de detección es una de las secciones más importante de este proyecto, ya que se establecen los patrones o signaturas que emplea el sistema NIDS para detectar posibles amenazas.

En particular, vamos a diseñar y analizar diferentes metodologías para poder detectar los ataques más recientes sufridos en España (Panorama Actual de las Ciberamenazas).

En las secciones anteriores, hemos visto que en general, uno de los ataques que más se ha perpetuado en España durante estos últimos años ha sido Emotet. Este ha provocado que muchos dispositivos acaben formando parte de una red botnet que ha utilizado sus recursos para distintos fines.

En la **Figura 16**, podéis observar un resumen de la metodología de este ataque:

1. La victima recibe un correo electrónico Phishing. Este está muy bien enfocado y utiliza métodos de Ingeniería Social para conseguir la confianza del usuario. El correo normalmente viene con unos ficheros adjuntos o enlaces de distribución de malware, dominios peligrosos entre otros.
2. En el caso de que el usuario decida hacer clic en el enlace o descargar el documento, se iniciará el proceso de infección del malware.
3. Una vez infectado el host, este informará al robot pastor y esperará órdenes.
4. En todo caso, este dispositivo ya forma parte de una red botnet, y podrá ser completamente controlado por el atacante y empleado para distintos fines.

Visto el resumen del ataque, analizamos tres puntos de control:

1. **Primer Control**: Detectar el Phishing en general, tanto enlaces como dominios maliciosos. También, incluiremos los correos SPAM. Este control nos advierte del engaño.
2. **Segundo Control**: Detectar los enlaces de descarga de Malware que también pueden tener el formato de la metodología Phishing. Este segundo control, nos alerta en caso de haber cometido una acción peligrosa.
3. **Tercer control**: Detectar las comunicaciones con la botnet. Finalmente, en el caso de haber sido infectados, este último control nos consciencia sobre el estado de la amenaza.

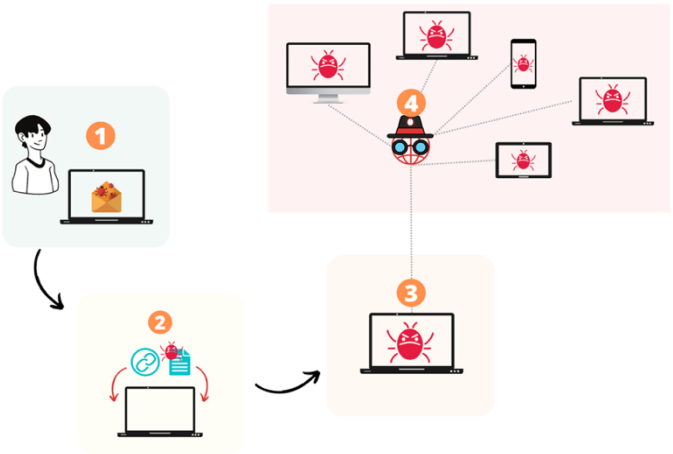


Figura 16. Metodología Ataques Botnet. Elaboración Propia.

3.3.3.1. Metodologías detección

A lo largo de estos años, se han desarrollado diferentes técnicas para la detención de amenazas, pero en general podemos clasificarlas en 2 grupos: Blacklists y Algoritmos de Machine Learning [21].

Las blacklists son grandes bases de datos que de forma continuada y periódica reportan Indicadores de Compromiso (IOC) sobre diferentes amenazas como son: dominios y enlaces phishing, certificados SSL maliciosos, huellas digitales JA3/JA3S, entre otros. Muchos de estos repositorios son creados y mantenidos por la propia comunidad y se encuentran disponibles de forma gratuita para todo el mundo.

Aun así, se trata de una forma de detección "after-the-fact"[24], es decir, este sistema solamente logra detectar amenazas conocidas, y en muchos casos ya padecidas [21]. Por lo tanto, no tiene la capacidad de detectar nuevos ataques. Para ello se emplean otras técnicas como el Machine Learning el cual, a través de la extracción de algunas características, el sistema adquiere la capacidad de clasificar y diferenciar entre las amenazas.

Emplear técnicas de Machine Learning es ventajoso, ya que podría ayudarnos a detectar nuevos riesgos e incluso aprender de ellos incorporando sus características en el sistema. Sin embargo, tampoco se trata de un sistema infalible, puesto que su éxito y la tasa de fallos (o falsos positivos) dependerá de la elección de unas características suficientemente delimitables.

En resumen, para la finalidad de este proyecto, hemos puesto a prueba ambas metodologías. Por un lado, comparamos los patrones de las amenazas con las blacklists, el cual es un método de detección bastante robusto. Y, por otro lado, diseñamos un pequeño set de reglas para detectar posibles nuevas amenazas.

Esta segunda metodología la enfocamos únicamente a la detección de enlaces o dominios maliciosos, dado que este suele ser el primer contacto que establecen los atacantes con la víctima intentando engañarles para cometer una acción peligrosa (Punto de Control número 1).

A pesar de ser un sistema, que a priori, parece presentar un bajo nivel de precisión, dado que es inevitable que alguna de sus características también las comparta un enlace benigno (generando falsos positivos). Sin embargo, también puede llegar a detectar posibles amenazas que, de otro modo, empleando la primera metodología, no serían detectados a tiempo.

En conclusión, ambas metodologías son empleadas, pero la segunda está catalogada como "posible ataque", mientras que la primera se trata de una verdadera amenaza.

[24] https://towardsdatascience.com/predicting-the-maliciousness-of-urls-24e12067be5

3.3.3.2. Metodología: BlackLists

Consultar una Blacklist se trata de una metodología que se basa en comprobar que un determinado patrón no esté incluido entre un conjunto de signaturas maliciosas o indicadores de compromiso (IOC). De tal manera, que su presencia indica que se trata una amenaza y su ausencia justamente de lo contrario.

Aun así, este sistema no es 100% preciso dado que por un lado no detecta nuevas amenazas, por lo tanto, serían catalogadas como seguras y, por otro lado, dominios legítimos utilizados anteriormente para fines de Phishing podrían ser catalogados como amenazas si no se han desactivado de la blacklist. Un ejemplo de ello es el repositorio Phishing Database de mitchellkrogza[25] el cual, posee una sección llamada: "*Please Remove my Domain From This List !!"* para alertar sobre falsos positivos originados por ataques u otras razones.

Selección de blacklists

Vista la metodología anterior, hemos establecido un conjunto de condiciones para la selección de las listas:

Tabla 9. Criterios de selección de Blacklists. Elaboración Propia.

#	Condición
1	Open Source.
2	Actualización diaria.
3	Que marque o elimine los patrones que ya no representan una amenaza.
4	En general, mínimos Falsos Positivos.
5	Conjunto considerable de datos.
6	Detección de Phishing, SPAM, enlaces de distribución de Malware y ataques Botnet C2.
7	Incorporar reglas que faciliten la integración de la base de datos con Suricata.
8	Las reglas deben ser del tipo alert.

En una búsqueda rápida por Internet vemos que existen diferentes repositorios gratuitos de IOCs, por lo tanto, y con la finalidad de minimizar los gastos de este proyecto, utilizamos únicamente repositorios Open Source.

Por otro lado, puesto que todos los días podrían aparecer nuevas amenazas es importante que esta base de datos se actualice, como mínimo 1 vez al día. Además, de catalogar los patrones que ya no supongan una amenaza evitando así el máximo número de falsos positivos.

Por último, debe incorporar reglas en formato Suricata. Este último requisito podría no ser del todo obligatorio, ya que nosotros podríamos diseñar las reglas con los datos de la blacklist. Aun así, se trata de una condición bastante favorable. En todo caso, si incorpora reglas en Suricata las cabeceras deben ser del tipo "*alert*", dado que estamos tratando con un IDS y no un IPS (Sistema de Prevención de Intrusos).

[25] https://github.com/mitchellkrogza/Phishing.Database

Blacklists

Mediante una investigación por Internet hemos encontrado diferentes Blacklists que cumplen con los requisitos anteriores. Estas son:

Tabla 10. Blacklists seleccionadas. Elaboración Propia.

#	Condición		Reglas
1	Phishing Filter		Sitios Web Phishing.
2	Abuse.ch	URLhause	Enlaces de Distribución de Malware.
3		SSLBL	Certificados SSL maliciosos (Botnet C2). IPs maliciosas (Botnet C2)
4		Feodo Tracker	IP maliciosas (Botnet C2)
5	Emerging Threats Open Ruleset		Hosts comprometidos DROP (SPAM) Ingeniería Social (Phishing)
6	PUP		Dominios "potentially unwanted programs"

A continuación, detallaremos las características principales de algunos de estos proyectos:

Phishig Filter

Phishing Filter[26] es un repositorio del ingeniero Ming Di Leom[27] que recopila una lista de sitios web phishing proveniente de:

- *PhishTank*[28]: un proyecto open source de Cisco Talos Intelligence Group[29] en el cual cualquiera puede alertar, compartir, rastrear y verificar enlaces phishing. Aun así, se sigue un proceso de votación para incluir el enlace en la blacklist.
- *OpenPhish*[30]: es un proyecto automatizado con inteligencia artificial para identificar sitios web phishing sin intervención humana. Solamente reportan los enlaces de los últimos 14 días y procuran eliminar enlaces phishing caídos.
- *phishunt.io*[31]: este proyecto está lanzado por el ingeniero Daniel López[32]. El proyecto busca posibles enlaces phishing y los verifica con diferentes herramientas como son urlscan.io, Google Safe Browsing, PhishTank or OpenPhish.
- *mitchellkrogza/Phishing.Database*[33]: finalmente Phishin Fatabase es un repositorio del ingeniero Mitchell Krog[34]. Se trata de una gran base de datos de dominios, IP y URL phishing que empela el proyecto PyFunceble para validar el estado de todos los datos phishing categorizados en: ACTIVE, INACTIVE e INVALID.

Esta lista se actualiza 2 veces al día y se encuentra en múltiples formatos para diferentes programas, entre ellos Suricata.

[26] https://gitlab.com/malware-filter/phishing-filter
[27] https://www.linkedin.com/in/mdleom/
[28] https://phishtank.org/
[29] https://www.talosintelligence.com/
[30] https://openphish.com/
[31] https://phishunt.io/
[32] https://www.linkedin.com/in/0xDanielLopez/
[33] https://github.com/mitchellkrogza/Phishing.Database
[34] https://github.com/mitchellkrogza

El formato de las alertas Suricata de esta blacklist sigue esta estructura:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"phishing-filter phishing website detected"; flow:established,from_client; http.method; content:"GET"; http.uri; content:"/zonaseegura-ingresar-viabeta"; endswith; nocase; http.host; content:"zon-vit.com"; classtype:attempted-recon; sid:200066055; rev:1;)
```

La regla inicia con la acción que es un alert (condición 8), seguida de la cabecera que es del tipo HTTP. Vemos también en azul la signatura de la blacklist junto a unos parámetros de evaluación. En general, los parámetros que evalúa esta blacklist son la coincidencia del host y/o la de la sección path de la URL.

Abuse.ch

Abuse.ch[35] es un proyecto Open Source, con un fuerte enfoque en el malware y las redes de bots. Entre sus plataformas encontramos las siguientes blacklists:

- ***URLhause***: es una base de datos centrada en compartir únicamente enlaces de distribución de malware.
- **SSL Blacklist**: este repositorio almacena certificados SSL maliciosos principalmente utilizados por los servidores de C&C de las botnets.
- **Feodo Traker**: esta blacklist comparte los servidores de C&C de los botnets asociados a Dridex, Emotet (aka Heodo), TrickBot, QakBot (aka QuakBot / Qbot) y BazarLoader (aka BazarBackdoor.

A continuación, veremos más en detalle algunas de las características de estas tres blacklists:

A) URLhause

Se trata de una blacklist muy activa que comparte sus datos con Spamhaus DBL, SURBL, Google Safe Browsing (GSB) y diferentes proveedores de software de seguridad que emplean esta base de datos para enriquecer sus productos.

Entre sus características destacamos:

- Etiqueta sus datos con tags que aportan información sobre la amenaza.
- Añade un atributo status que nos indica si el enlace sigue vigente (online/offline).
- Su API comparte los datos en diferentes formatos entre ellos Suricata.
- Actualiza las reglas casa 5 minutos.

A continuación, tenéis un ejemplo de regla:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"URLhaus Known malware download URL detected (42375)"; flow:established,from_client; http.method; content:"GET"; http.uri; content:"/logsite/0lddoc/tka1833163913soxcjh/aug-11-2018-06005952849/nvs-vjxv/"; depth:69; endswith; nocase; http.host; content:"pink99.com"; depth:10; isdataat:!1,relative; metadata:created_at 2018_08_14; reference:url, urlhaus.abuse.ch/url/42375/; classtype:trojan-activity;sid:80905475; rev:1;)
```

La estructura de la regla es muy parecida a la vista anteriormente con Phishing Filter.

[35] https://abuse.ch/

B) SSL Blaklist

Como hemos visto anteriormente esta blacklist ofrece dos sets de reglas dirigidas principalmente a identificar redes botnet C&C. Al igual que URLhause, ofrece reglas en Suricata, las actualiza cada 5 minutos y en general solamente tiene en cuenta datos recientes (máximo de 30 días)[36].

A continuación, podéis ver la estructura de las tres listas:

SSL Certificate: esta primera lista contiene huellas SHA1 de los certificados maliciosos, vemos que en este caso el protocolo es TLS.

```
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"SSLBL: Malicious SSL
certificate detected (Shylock C&C)"; tls_cert_fingerprint;
content:"b0:8a:49:39:fb:88:f3:75:a2:75:7e:ad:dc:47:b1:fb:8b:55:44:39";
reference:url, sslbl.abuse.ch/ssl-
certificates/sha1/b08a4939fb88f375a2757eaddc47b1fb8b554439/; sid:903200000;
rev:1;)
```

Botnet C2 IP: esta lista contiene la combinación IP-Puerto de los servidores botnet C&C. Podemos ver cómo están especificados la IP y puerto destino. Cabe destacar que existen 2 versiones de este set de reglas. La versión extensa o también llamada "agressive" contiene datos antiguos (> 30 días), lo cual podría generar falsos positivos, por lo tanto, seleccionamos la versión reducida y "más fiable".

```
alert tcp $HOME_NET any -> [119.23.227.43] 8848 (msg:"SSLBL: Traffic to malicious
host (likely DCRat C&C traffic)"; flow:established,to_server; threshold: type
limit, track by_src, seconds 60, count 1; classtype:trojan-activity;
sid:904200000; rev:1;)
```

C) Feodo Tracker

Por último, la última blacklist que utilizaremos del proyecto Abuse.ch es Feodo Tracker. Esta base de datos inició en 2010, con el objetivo de rastrear los C&C de la botnet Feodo, pero este último evolucionó y aparecieron más derivados de este malware como son: Emotet, TrickBot, Dridex, QakBot, BazarLoader y BumbleBee [17].

Esta blacklist se actualiza cada 5 minutos, pero nos recomienda actualizarla en nuestro entorno local como mínimo cada 15 min [17]:

*"We recommend you to update the IDS ruleset at **least every 15 minutes** (or even better: every 5 minutes) to receive the best protection against Dridex, Emotet, TrickBot, QakBot and BazarLoader."*

Feodo Track también ofrece un set de reglas más extenso "agressive" el cual no utilizaremos, puesto que aumentaría los falsos positivos. Como podéis ver a continuación, el set de reglas es muy semejante al set Botnet C2 IP indicando la combinación IP:Puerto destino.

```
alert tcp $HOME_NET any -> [37.187.115.122] 6601 (msg:"Feodo Tracker: potential
Dridex CnC Traffic detected"; threshold: type limit, track by_src, seconds 60,
count 1; classtype:trojan-activity; reference:url,
feodotracker.abuse.ch/browse/host/37.187.115.122/; sid:900505528; rev:1;)
```

[36] https://urlhaus.abuse.ch/

Emerging Threats Open Rulset

Emerging Threats es un proyecto de Proofpoint, Inc[37]. Este repositorio está formado por dos sets de reglas Emerging Threats, aportado y mantenido por la comunidad de seguridad y Emerging Threats Pro, mantenido por el equipo de Investigación de Proofpoint. Este proyecto es muy conocido por los usuarios de Suricata, puesto que está incluido en la extensión suricata-update que se encarga de gestionar las reglas.

En general el dataset de ET, no está enfocado en una categoría en específico de reglas, de hecho, podemos encontrar hasta 17 diferentes categorías. En nuestro caso nos centramos en las relacionadas con el propósito de este proyecto.

Por lo tanto, seleccionamos los siguientes sets de reglas:

A) Known CompromisedHost List

Se trata de una lista obtenida de diferentes fuentes que registra un listado de hosts comprometidos por bots, sitios web phishing etc[38].

La estructura de las reglas es la mostrada a continuación. Como podéis observar, a diferencia de las reglas vistas anteriormente, detecta los paquetes con una IP origen con destino a alguna IP de nuestra red local.

```
alert ip
[103.121.196.203,103.129.247.203,103.176.178.151,103.176.178.98,103.177.177.226,1
03.188.176.251,103.188.176.3,103.215.221.178,103.82.20.92,104.154.230.144,104.156
.155.31,104.196.134.157,104.199.121.188,107.172.127.54,107.174.224.121,107.182.12
9.203,107.189.175.180,108.173.208.195,109.205.213.14,109.206.241.13,109.206.241.1
7,112.29.139.34,114.92.195.10,115.242.211.98,116.110.112.158,116.252.197.191,116.
7.245.20,116.7.245.24,116.98.174.206,117.211.202.164] any -> $HOME_NET any
(msg:"ET COMPROMISED Known Compromised or Hostile Host Traffic group 1";
reference:url,doc.emergingthreats.net/bin/view/Main/CompromisedHosts; threshold:
type limit, track by_src, seconds 60, count 1; classtype:misc-attack;
sid:2500000; rev:6268; metadata:affected_product Any, attack_target Any,
deployment Perimeter, tag COMPROMISED, signature_severity Major, created_at
2011_04_28, updated_at 2022_08_26;)
```

[37] https://www.proofpoint.com/es
[38] https://doc.emergingthreats.net/bin/view/Main/CompromisedHost

B) DROP Rules

DROP o "Don't Route or Peer", a pesar de lo sugerentes que puedan ser las iniciales, esta lista se trata de acciones del tipo alert (no drop) provenientes de Spamhaus Block List (SBL)[39]. Esta última blacklist registra un conjunto de IPs de las cuales no se recomienda aceptar correos electrónicos y es mantenida por un conjunto de investigadores y forenses de 10 diferentes países confirmando spam diariamente[40].

La estructura de la alerta es muy parecida a la del dataste anterior:

```
alert ip
[160.180.0.0/16,160.188.0.0/16,160.200.0.0/16,160.235.0.0/16,160.240.0.0/16,161.0
.0.0/19,161.0.68.0/22,161.1.0.0/16,162.208.124.0/22,162.222.128.0/21,162.249.20.0
/22,162.251.92.0/22,163.47.19.0/24,163.50.0.0/16,163.53.247.0/24,163.128.224.0/19
,163.197.0.0/16,163.198.0.0/16,163.216.0.0/19,163.250.0.0/16] any -> $HOME_NET
any (msg:"ET DROP Spamhaus DROP Listed Traffic Inbound group 15";
reference:url,www.spamhaus.org/drop/drop.lasso; threshold: type limit, track
by_src, seconds 3600, count 1; classtype:misc-attack; flowbits:set,ET.Evil;
flowbits:set,ET.DROPIP; sid:2400014; rev:3358; metadata:affected_product Any,
attack_target Any, deployment Perimeter, tag Dshield, signature_severity Minor,
created_at 2010_12_30, updated_at 2022_08_26;)
```

C) Ingeniería Social

Finalmente, incorporamos reglas para detectar Phishing relacionado con la Ingeniería Social[41]. Este tipo de reglas están categorizadas como "social-engineering" y la signatura incluye "Possible Phishing", lo cual, nos puede dar a entender que podrían existir falsos Positivos.

Aun así, este conjunto de reglas serán también una excepción y apartadas del set habitual, puesto que si observamos la estructura del siguiente ejemplo podremos ver que intenta identificar un paquete http providente de Internet con el título "iTunes Connect". Esto evidentemente sería imposible de detectar en un caso benigno y real, puesto que iTunes utiliza TLS, por lo tanto, el contenido del paquete estaría cifrado.

```
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET PHISHING Possible iTunes
Phishing Landing - Title over non SSL"; flow:established,to_client; file_data;
content:"<TITLE>iTunes Connect</TITLE>"; classtype:social-engineering;
sid:2018303; rev:4; metadata:created_at 2014_03_21, former_category
CURRENT_EVENTS, updated_at 2017_06_16;)
```

Emergin Threads, también posee un set relacionado con la detección de Botnets, pero nos hemos percatado que provienen del Feodo Tracker, por lo tanto, ya lo tendríamos incorporado.

[39] https://www.spamhaus.org/
[40] https://doc.emergingthreats.net/bin/view/Main/SpamHausDROPList
[41] **https://rules.emergingthreats.net/open-nogpl/suricata-5.0/rules/emerging-phishing.rules**

PUP Domains Blocklist

Finalmente, el último dataset a incorporar será PUP o "potentially unwanted programs". PUP Filter, al igual que Phishing Filter, se trata de un repositorio desarrollado por el ingeniero Ming Di Leom[42] que utiliza la herramienta Malware Discover[43]

Malware Discover

Malware Discover es un proyecto desarrollado por el Dr. Zhouhan Chen[44] que intenta detectar redirecciones de URL hacia sitio phishing y malware. Básicamente, trata de encontrar los puntos de entrada o dominios que inician el redireccionamiento hacia sitios/enlaces maliciosos. De hecho, en la **Figura 17**, podéis observar una campaña de Malware descubierta por el mismo proyecto.

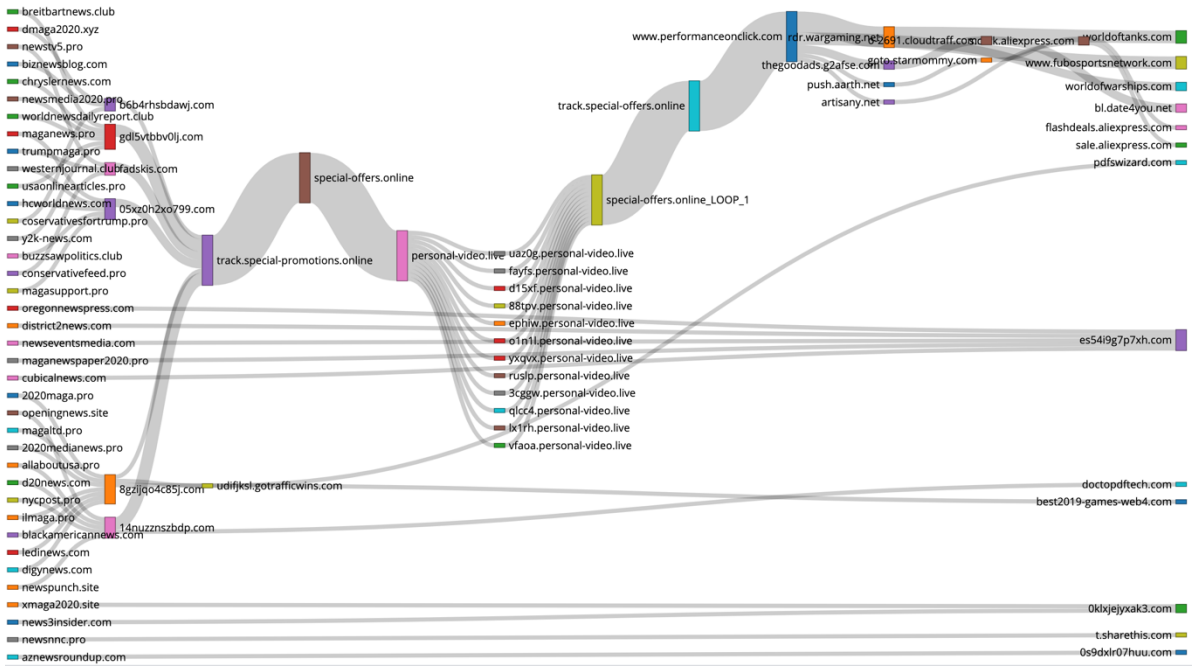


Figura 17. Campaña de Malware descubierta por Malware Discover. https://gitlab.com/malware-filter/pup-filter

En cuanto a la blacklist PUP Filter[45], se actualiza 2 veces al día y tiene diferentes formatos de reglas entre ellos Suricata.

Un ejemplo de regla podría ser el siguiente:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"pup-filter PUP website
detected"; flow:established,from_client; http.method; content:"GET"; http.host;
content:"1800petmeds.icu"; classtype:web-application-activity; sid:300000001;
rev:1;)
```

Observamos que en este caso solamente trata de identificar el host, no se centra el resto de la URL.

[42] https://www.linkedin.com/in/mdleom/
[43] https://github.com/zhouhanc/malware-discoverer
[44] https://www.linkedin.com/in/zhouhan-chen-658001b4/
[45] https://gitlab.com/malware-filter/pup-filter

3.3.3.3. Metodología: Machine Learning

La segunda metodología para poder detectar posibles amenazas es a través del Machine Learning o aprendizaje automático. Se trata de una disciplina de la Inteligencia Artificial que permite detectar patrones o realizar predicciones a través de un conjunto de características suficientemente delimitables.

Para la finalidad de este proyecto, diseñamos un set de reglas que permite a nuestro NIDS tener la capacidad de predecir posibles enlaces o dominios maliciosos que aún no hayan sido incorporados en ninguna blacklist. En este caso, nos basamos en la investigación de **Ruth Eneyi Ikwu**[46] "*Extracting Feature Vectors From URL Strings For Malicious URL Detection*" [21] (doctora en ciberseguridad y ciencia de datos) que nos ha facilitado las características para diferenciar entre enlaces benignos de los malignos.

Además, debido a que el dataset empleado por la Dra. Eneyi data del 2016, **URL dataset (ISCX-URL2016)**[47] **[21],** los resultados podrían no estar del todo acorde con la situación actual postpandemia. Por lo tanto, hemos decidido emplear la librería que la doctora ha utilizado y compartido por su GitHub[48] para evaluar un nuevo dataset de agosto del 2022.

Una vez generadas las estadísticas, comparamos los resultados obtenidos con los de la investigación y decidimos cuales son los parámetros que delimitan mejor entre los dos conjuntos para finalmente crear las alertas en Suricata. Cabe mencionar que estas nuevas alertas, no son tratadas como amenazas reales, ya que muchas veces podrían coincidir con casos benignos. Por lo tanto, son separadas de las alertas generadas por las blacklists, además de ser categorizadas como posibles amenazas pendientes de consulta.

Estructura de un enlace

Antes de iniciar con la extracción de características, vamos a repasar la estructura básica de un enlace. En la podéis observar los diferentes componentes de la URL [21]:

1. *Protocolo*: inicia con un protocolo que puede ser http o https dependiendo del nivel de seguridad.
2. *Host*: es la sección que identifica el sitio web de la entidad. En general, los sitios web benignos suelen utilizar palabras memorables, ya que se trata del punto de acceso a su sitio en Internet.
3. *Path*: especifica la localización de ficheros, nombres de directorios, y en general recursos del servidor. Normalmente, los atacantes suelen manipular esta sección para incorporar nuevos elementos[21].

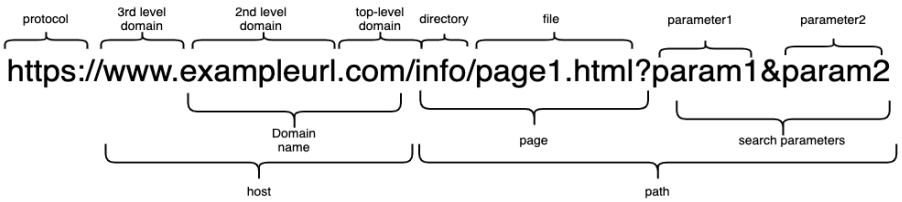


Figura 18. Formato de un enlace. https://frontend.turing.edu/lessons/module-3/rest-architecture-and-urls.html

[46] https://www.linkedin.com/in/ruth-eneyi-i-83a699118/
[47] https://www.unb.ca/cic/datasets/url-2016.html
[48] https://gist.github.com/eneyi/5c0b33129bcbfa366eb9fe79e96c1996#file-get_lexical_features-py

Extracción de Características

Como hemos comentado anteriormente, la selección de las características es un punto clave para el éxito del sistema. Según la investigación de la dra. Eneyi, se puede extraer información de 3 fuentes [21]:

1. **Características Léxicas**: se trata de estadísticas obtenidas de la propia cadena URL, como son la longitud, cantidad de puntos, números, etc.
2. **Características basadas en Host**: son características extraídas de la sección host del enlace. Podemos extraer información como es el país de registro, propiedades del dominio, etc.
3. **Características del Contenido**: por último, estas características están relacionadas con el contenido que se obtiene del propio código HTML descargado. Aquí podremos observar objetos incrustados, etiquetas, elementos ocultos, etc.

En nuestro análisis, solamente nos centramos en las características léxicas, ya que estás son las más seguras de evaluar puesto que no se requiere descargar ningún tipo de contenido.

Selección de características

A continuación, seleccionamos las características léxicas que empleamos en el análisis. Este tipo de características se basan en analizar diferentes parámetros de forma estadística con el fin de encontrar valores que delimiten los dos conjuntos [21]. En la **Figura 19** podemos observar 18 característica léxicas que ha reunido la dra. Eneyi.

Como el sistema final es un NIDS que utiliza Suricata como software de detección, hemos considerado qué tipo de características podemos o no detectar con esta herramienta. Por ejemplo, la característica número 10 de la **Figura 19** que evalúa la entropía (aleatoriedad del enlace) es muy difícil de implementar en una regla.

Además de las características de la **Figura 19**, hemos añadido estas 6 características propias con relación a los parámetros especiales y keywords:

Tabla 11. Características Léxicas. Elaboración Propia.

#	Tipo	Elemento	Características
1	Símbolo	“?”	Se emplea el interrogante para indicar el inicio de la Query string o cadena de consulta[49]
2		“=”	se utiliza normalmente para añadir los valores de los parámetros en los enlaces.
3		“~”	Representa el inicio de un directorio.
4		“@”	en el caso de incrustar un correo electrónico en el enlace o redirección de url.
5	Keyword	password	básicamente podrían incrustar el parámetro password dentro de una página de login
6		redirect	para la redirección.

[49] https://es.wikipedia.org/wiki/Query_string

SN	FEATURE NAME	DESCRIPTION
1	URL Length	Total number of characters in URL
2	URL Scheme	The URL's scheme or ***protocol*** represents the set of rules that decide how files would be displayed and formatted and how data would be transported across the web.
3	URL Path Length	The length of the URL path which would also include subdirectories on the web server
4	URL Host length	The total number of characters in the domain or host name. For example 'google.com' will have a character length of 10.
5	URL Host is ip	If the host or domain part of the URL string is an ip address rather than a domain name. For example 'google.com/help' is represented as '8.8.8.8/help'
6	Number of Digits	Number of digits in URL strings [0-9]. For example 'the boot567.au' has a num-digit-count of 3.
7	Number of parameters	The number of values being queried on the site if a search is being performed. For example, 'amazon.com?color=pink&size=20'
8	Number of fragments	Fragments are optional components of a url string that leads to another resource that is a subordinate of a promary resource. This is usually precedded by a '#' sign. for example www.examplewebsite.com/aboutus#whatwedo
9	is Encoded	URL is URL-encoded using a '%' character and a two-character hex value corresponding to their **UTF-8** character.
10	URL Entropy	Shannon entropy of URL string
11	Similar To Alexa	Average similarity of URLs host name to any Alexa site
12	Number of Subdirectories	Number of subdirectories in URL path
13	Number of periods	Number of '.' in URL string
14	Has Keyword 'client'	URL String Contains the keyword 'client'
15	Has Keyword 'admin'	URL String contains the keyword 'admin'
16	Has keyword 'server'	URL String contains the keyword 'server'
17	Has keyword 'login'	URL String contains the keyword 'login'
18	has port	URL string has a port number in it for example examplewebsite.com:8080

Figura 19. Características Léxicas. https://towardsdatascience.com/extracting-feature-vectors-from-url-strings-for-malicious-url-detection-cbafc24737a

Conjunto de Características Final

Finalmente, el conjunto de características seleccionadas para la evaluación estará compuesto de los siguientes 21 elementos:

1. HTTP o HTTPS.
2. El Host es una IP.
3. Contiene el Puerto.
4. Longitud de la URL.
5. Longitud del HOST.
6. Longitud del PATH.
7. Cantidad de Puntos.
8. Cantidad de Subdirectorios.
9. Cantidad de Dígitos.
10. Cantidad de Parámetros “&”.
11. Cantidad de Interrogantes “?”.
12. Cantidad de Fragmentos “#”.
13. Cantidad de Tilde “~”.
14. Cantidad de Arroba “@”.
15. Cantidad de Igual “=”.
16. Palabra clave admin.
17. Palabra clave server.
18. Palabra clave client.
19. Palabra clave login.
20. Palabra clave password.
21. Palabra clave redirect.

En el caso del set de dominios, al no disponer de la sección path o del protocolo, solamente evaluamos algunas de las características anteriores.

Dataset agosto 2022

Después de seleccionar las características, hemos elegido el dataset para el análisis. En este caso, se trata de un dataset conformado tanto por enlaces benignos como maliciosos, de esta manera podemos comparar los resultados. Además, han sido recolectados en agosto del 2022 para realizar un análisis de la situación actual.

Enlaces Malicosos

El set de enlaces maliciosos está compuesto de 3 conjuntos de datos bien diferenciados:

- *Enlaces Malware:* este conjunto se ha obtenido de URLHause[50], una blacklist que comparte enlaces maliciosos de distribución de malware. El set consta de 74,526 enlaces reales.
- *Enlaces Phishing General*: este segundo conjunto de datos se ha obtenido del repositorio de GitHub Phishing Database[51]. Se trata de un proyecto muy activo que registra dominios, sitios web y amenazas en relación con el phishing. El set consta de 722,342 enlaces. Se trata de un gran conjunto de enlaces phishing con diferentes finalidades, el cual nos da una perspectiva general de las características de estos ataques.
- *Dominios Phishing*: este último dataset se ha obtenido del mismo repositorio Git anterior, Phishing Database con un total de 396,433 datos. Ha sido seleccionado, ya que nos resulta interesante conocer estadísticas únicamente de la sección del dominio.

Enlaces Benignos

A diferencia de los enlaces maliciosos, para los enlaces benignos no es tan sencillo encontrar un dataset actual (agosto 2022) para el estudio. Por lo tanto, los hemos generado. Para ello hemos seguido la metodología que ha empleado el *Canadian Institute for Cybersecurity* para generar el dataset 2016 Benign URLs [40]. Esta metodología es la siguiente:

- Recolectar el top de páginas web de Alexa.
- Pasar el dominio por el web crawler Heritrix para extraer URLs.
- Eliminar dominios y duplicados.
- Verificar con Virustotal que no se trata de enlaces maliciosos.

En nuestro caso, hemos realizado unas pequeñas variaciones. En lugar de Heritrix, hemos empleado la librería Python SEO Crawler / Spider[52], puesto que es bastante sencillo manejarla. Además, para la verificación de enlaces maliciosos, la API de Virustotal (la versión gratuita) ofrece muy pocas evaluaciones al día. Por lo tanto, debido a que necesitamos evaluar una considerable cantidad de datos la hemos descartado. En su lugar, hemos comprobado que los enlaces no se encuentren en las siguientes blacklists: URLhause, Phishing Database, Phishunt.io[53] y PhishTank[54].

[50] https://urlhaus.abuse.ch/
[51] https://github.com/mitchellkrogza/Phishing.Database
[52] https://advertools.readthedocs.io/en/master/advertools.spider.html
[53] https://phishunt.io/
[54] https://phishtank.org/

Implementación Dataset Enlaces Benignos

1. Primero hemos descargado el Top 1 Million de Alexa[55], en el cual vienen listados los dominios más visitados. De este fichero hemos seleccionado los 10,000 primeros.
2. Después hemos creado un sencillo programa en Python el cual:
    a. Añade la cabecera https a los dominios anteriores. Si estas páginas no tuvieran el certificado SSL, el crawl nos devolvería una url con http o notfound.
    b. Realiza un crawl con dos parámetros:
        i. *follow_links = True*: este parámetro indica que nos retornará los enlaces que vaya encontrando.
        ii. *CLOSESPIDER_TIMEOUT = 5*: este segundo parámetro, básicamente limita el tiempo de inspección a 5 segundos. Esto ha sido añadido, ya que los principales dominios son grandes páginas web con miles de enlaces como son Google o Youtube, los cuales mantiene el programa ocupado durante un largo periodo de tiempo. Con esta limitación, tenemos un pequeño conjunto de cada sitio web habiendo así más variedad.
    c. Finalmente almacena los enlaces en Results.txt.
3. Debido a que el tiempo de la inspección es bastante largo hemos detenido el programa alrededor del dominio número 3791. Obteniendo así un resultado de hasta 174,126 enlaces el cual es suficiente.

A continuación, podéis ver el código ejecutado en Python:

```
import advertools as adv
import pandas as pd
import os

list = []

# Lectura de los enlaces:
def prepare():
    with open('TOP.csv') as file:
        for line in file:
            content = line.split(',')
            list.append("https://"+content[1].strip())

# Web Crawl:
def crawl(url):
    adv.crawl(url, 'Output.jl',
              follow_links=True, custom_settings={'CLOSESPIDER_TIMEOUT': 5})
    crawl_df = pd.read_json('Output.jl', lines=True)
    if os.stat('Output.jl').st_size != 0:
        data = crawl_df["url"].to_list()
        os.remove("./Output.jl")
        write(data)
    else:
        with open('NotFound.txt', 'a') as file:
            file.write('%s\n' % url)

# Resultados:
def write(list):
    with open('Results.txt', 'a') as file:
        for url in list:
            file.write('%s\n' % url)



# Progrma Principal
if __name__ == '__main__':

    prepare()
    # Web Crawl:
    for url in list:
        crawl(url)
```

[55] https://gist.github.com/chilts/7229605

Dataset Final

Finalmente, hemos realizado una selección aleatoria (sin duplicado) de 100.000 elementos en ambos datasets benignos y maliciosos.

Antes de proceder a la selección hemos tenido que reducir los candidatos eliminando:

- Los enlaces maliciosos del set de benignos: comparamos las urls con las blacklists anteriores. No hemos obtenido ninguna conciencia.
- Eliminar dominios en los sets de enlaces (URL), incluso los que incluyan el símbolo "/" al final.
    - https://google.com
    - https://google.com/
- Comprobar que no existan duplicados: no hemos encontrado ningún caso.

Por lo tanto, el resultado final ha sido:

Benignos:
- Enlaces Benignos: 100,000 enlaces.
- Dominios Benignos: 100,000 enlaces.

Maliciosos:
- Enlaces Phishing: 100,000 enlaces.
- Enlaces malware: 74,268 enlaces.
- Dominios maliciosos: 100,000 dominios.

En el caso de los enlaces malware, el data set inicial contenía menos de 100,000 enlaces.

Ejecución del Análisis

Después de la selección de las características y datasets vamos a ejecutar el análisis mediante la librería *LexicalURLFeature* que nos ofrece la dra. Eneyi en su investigación [21]. Cabe mencionar que para algunos parámetros hemos realizado algunas pequeñas modificaciones que no alteran el funcionamiento general de la herramienta.

A continuación, podéis ver el programa principal implementado que llama a las funciones de la librería LexicalURLFeature:

```
from clases import LexicalURLFeature
import csv

def main():

    # Output:
    file = open('Phishing_Links.csv', 'w')
    writer = csv.writer(file, delimiter=';')
    header = [
        'HTTP/HTTPS',
        'Longitud URL', 'Longitud HOST', 'Longitud PATH',
        'Puntos', 'Subdirectorios', 'Parametros', 'Digitos', 'Fragment', 'Interrogante', 'Special', 'Arroba',
        'Host_IP', 'Puerto',
        'Key=admin', 'Key=server', 'Key=client', 'Key=login', 'Key=password', 'Key=redirect',
        'TLD']
    writer.writerow(header)

    # Input:
    f = open("ALL-phishing-links.txt", "r")

    # Execute:
    for x in f:
        url = LexicalURLFeature(x)

        writer.writerow([
            int(url.url_scheme() == 'https'),   # HTTP/HTTPS.
            url.url_length(),                   # Longitud URL.
            url.url_host_length(),              # Longitud HOST.
            url.url_path_length(),              # Longitud PATH.
            url.number_of_periods(),            # Cantidad de puntos.
            url.number_of_subdirectories(),     # Cantidad de Subdirectorios.
            url.number_of_parameters(),         # Cantidad de Parametros.
            url.number_of_digits(),             # Cantidad de Digitos.
            url.number_of_fragments(),          # Cantidad de Hastag.
            url.number_of_interrogante(),       # Cantidad de ?
            url.number_of_special(),            # Cantidad de ~
            url.number_of_arroba(),             # Cantidad de @
            url.number_of_igual(),              # Cantidad de =
            int(url.url_host_is_ip()),          # Verificar  Host es IP.
            int(url.url_has_port_in_string()),  # Verificar si contiene Puerto.
            int(url.has_admin_in_string()),     # Contiene Keyword=admin.
            int(url.has_server_in_string()),    # Contiene Keyword=server.
            int(url.has_client_in_string()),    # Contiene Keyword=client.
            int(url.has_login_in_string()),     # Contiene Keyword=login.
            int(url.has_password_in_string()),  # Contiene Keyword=password.
            int(url.has_redirect_in_string()),  # Contiene Keyword=redirect.
            url.get_tld()                       # TLD.
        ])


if __name__ == "__main__":
    main()
```

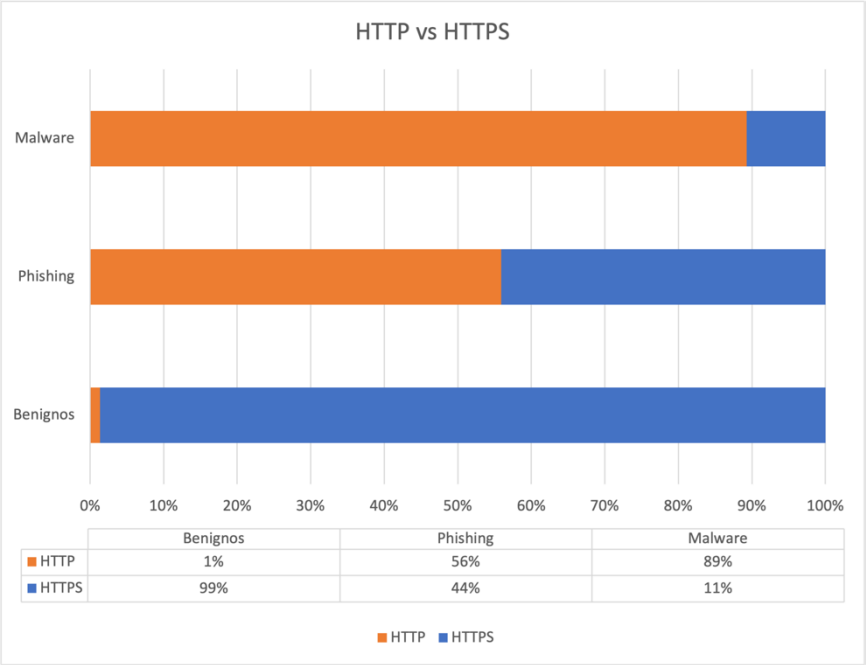
Análisis de resultados

Finalmente, después de la ejecución del código, analizamos los resultados para poder diseñar las reglas con las conclusiones obtenidas.

1. HTTP vs HTTPS

Como era de esperar, en el set de enlaces benignos alrededor del 99% contenían el certificado SSL. Por otro lado, en cuando a los enlaces Phishing, vemos que alrededor del 44% contienen este certificado, de tal manera que 1 de cada 3 enlaces phishing es HTTPS. Finalmente, en cuanto a los enlaces de distribución de Malware, vemos que la mayoría de los enlaces, alrededor del 89% no contiene este certificado.

En general, en cuanto a enlaces benignos y malware era un resultado bastante esperado, puesto que los primeros se preocupan de la confidencialidad y los segundos todo lo contrario. En cuanto a los enlaces Phishing es bastante preocupante este 43%, ya que engañan al usuario con el pretexto de estar en una página web segura.



	Benignos	Phishing	Malware
HTTP	1%	56%	89%
HTTPS	99%	44%	11%

Figura 20. HTTP vs HTTPS. Elaboración Propia.

2. Sección host es una IP

En general, hemos encontrado muy pocos casos en los cuales la sección host de la url sea una IP. De hecho, en el set de dominios, tanto en los maliciosos como benignos, el resultado ha sido prácticamente nulo.

Por otro lado, en cuanto a los enlaces, dentro del conjunto de enlaces Phishing, hemos encontrado también muy pocos casos siendo estos un total de 4%. Este resultado es el esperado, ya que va en contra de la propia filosofía phishing, puesto que, al observar un dominio numérico, el usuario podría no sentir esta confianza de acceder al sitio web y en consecuente ser engañado.

Finalmente, en los enlaces de distribución de Malware vemos que alrededor del 11% de las urls el host es una IP. Lo que representaría aproximadamente 1 de cada 10 enlaces.

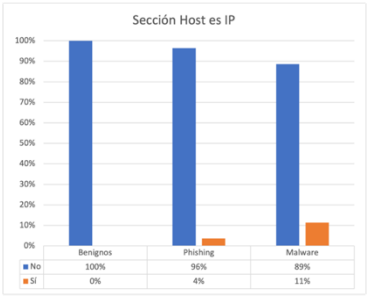


Figura 22. Sección Host es IP. Elaboración Propia.

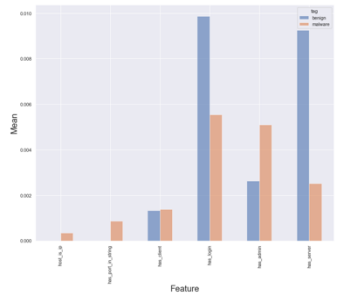


Figura 21. Extracción de Características. https://towardsdatascience.com/extracting-feature-vectors-from-url-strings-for-malicious-url-detection-cbafc24737a

3. Contiene Puerto

Otro dato interesante es conocer si el enlace o dominio contiene especificado un puerto. En este caso obtuvimos unos resultados bastante dispares entre algunas bases de datos.

En cuanto al set de los dominios prácticamente el 100% no incluían el puerto, tanto en los benignos como maliciosos, resultado semejante al de los enlaces phishing. Este último caso, es superado incluso por el set de benignos el cual contiene un 1% de enlaces con puerto, aun así, no es una cantidad muy significativa. Por el contrario, en el set de enlaces de malware hemos obtenido un resultado impresionante de hasta 73% de enlaces con el puerto especificado. Esto representaría casi 3 de cada 4 enlaces malware. De hecho, podemos observar, como en los resultados de la dra. Eneyi (**Figura 21**) solamente los enlaces maliciosos incluyen el puerto. Aun así, en la tabla X observamos los 5 puertos más repetidos, vemos que el primero apenas aparece 33 veces dentro del dataset, lo cual indica que los valores de los puertos son bastante diversos.

Puerto	Cantidad
55717	33
63424	29
8080	24
46805	23
38484	23

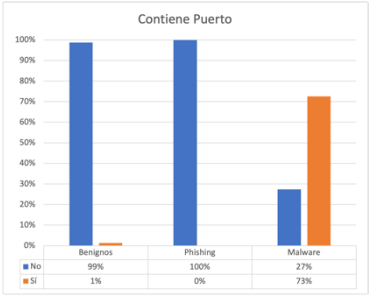


Figura 23. Contiene Puerto. Elaboración Propia.

4. Longitud

La longitud podría ser otro parámetro importante en el momento de distinguir enlaces maliciosos. En este caso vamos a diferenciar entre 3 longitudes: Longitud del HOST, Longitud del PATH y Longitud de toda la URL.

4.1.Longitud de toda la URL

Al analizar las longitudes completas de los enlaces, observamos que en general, los enlaces Phishing son los más largos, seguidos por los enlaces benignos y, por último, los de malware.

Una de las conclusiones que ha obtenido la dra. Eneyi en su investigación es [21]:
"*Malicious URLs are generally shorter in length than benign URLs.*"

Si consideramos que el dataset que ha utilizado es de Malware, entonces coincidimos con los resultados. Esta conclusión, es de hecho impactante, ya que tendemos a pensar que los enlaces maliciosos son más largos al contener más parámetros, símbolos especiales, dominios largos inentendibles, etc. Sin embargo, vemos que esta tendencia si se cumple con los enlaces Phishing.

Enlaces Malware:

Si nos centramos únicamente en los enlaces Malware, vemos que la gráfica es más o menos simétrica, más compacta que la de los enlaces benignos y phishing, y con un promedio de 38 caracteres y una mediana de 34. Observamos que el máximo está situado alrededor de los 43 caracteres, además el rango intercuartílico comprende los valores de 32 y 36, lo que indica que los enlaces de distribución de malware tienen una longitud bastante pequeña con pocas variaciones.

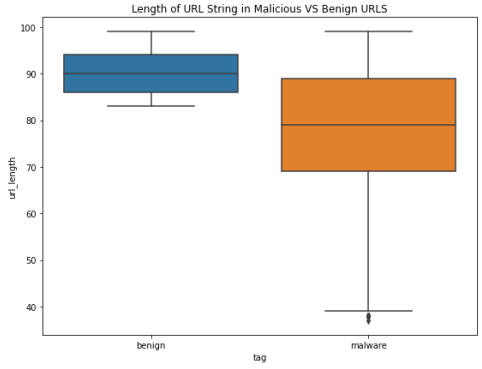


Figura 25. Longitud URL. https://towardsdatascience.com/extracting-feature-vectors-from-url-strings-for-malicious-url-detection-cbafc24737a

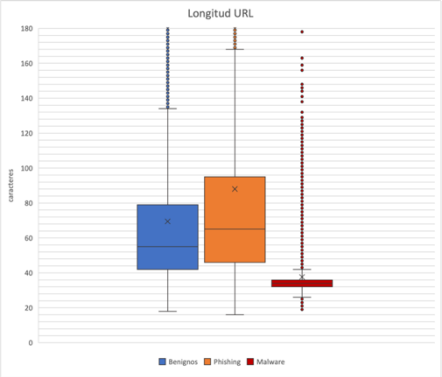


Figura 24. Longitud URL. Elaboración Propia.

4.2.Longitud del Host

En este caso, solamente contabilizamos la longitud de la sección host de la url. Además, también tenemos en cuenta las longitudes del dataset de dominios. En general, observamos que los enlaces y dominios maliciosos son los que tienen la longitud de host más larga.

Otra característica destacable es que los enlaces Phishing son los que tienen una mayor variabilidad, vemos como los valores comprenden desde 0 hasta un máximo de 47 caracteres (**Figura 26**). Caso contrario ocurre con los enlaces Malware, los cuales la gráfica es mucho más compacta con un rango intercuartílico de 3 unidades.

En cuanto a los dominios, observamos también como los maliciosos son más largos, de hecho, el promedio gira entorno a 30 caracteres hasta un máximo de 55 (**Figura 27**).

En conclusión, vemos en este caso, como los enlaces y dominios benignos tienen una longitud de host más pequeña. Esto podría estar relacionado con que en general las entidades y organizaciones no-maliciosas tienden a tener un nombre de dominio más memorables, puesto que es el punto de acceso a sus sitios web.

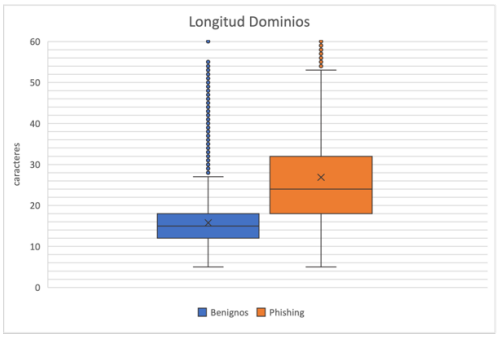


Figura 27. Longitud Dominios. Elaboración Propia.

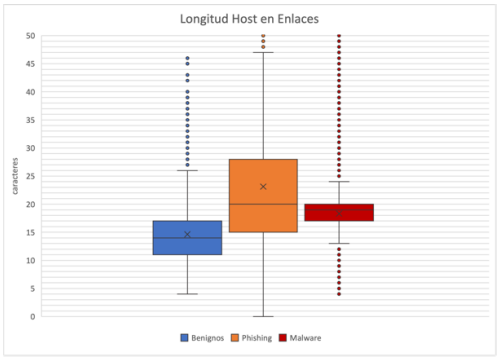


Figura 26. Longitud Host Enlaces. Elaboración Propia.

4.3.Longitud del PATH

Por último, analizaremos la longitud de la sección path. Vemos que en general hay pequeñas variaciones entre el set de enlaces Benignos y Phishing, de hecho, más o menos coinciden en el máximo y mínimo. Por otro lado, en cuanto a los enlaces malware, observamos que alrededor del 52% del dataset contiene 8 caracteres, este dato es bastante interesante, ya que se trata de un gran porcentaje de los datos.

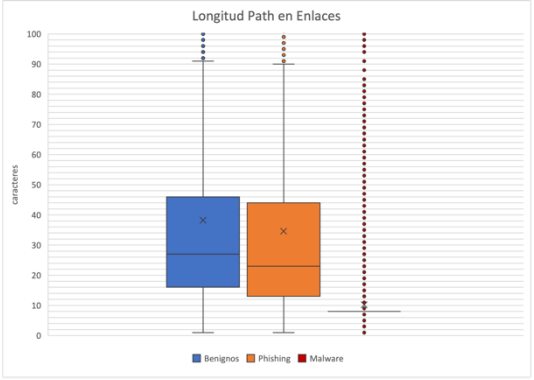


Figura 28. Longitud Path. Elaboración Propia.

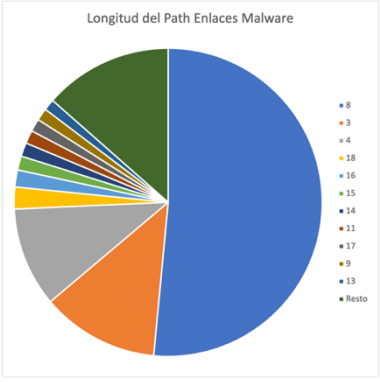


Figura 29. Longitud Path enlaces Malware. Elaboración Propia.

5. Cantidad de Dígitos

Otra característica que vamos a medir es la cantidad de dígitos en el enlace. En general, observamos que los enlaces y dominios maliciosos son lo que más contienen dígitos sobre todo los enlaces Phishing.

Dominios:

En cuanto a los dominios, vemos que los enlaces benignos en su mayoritaria no contienen dígitos. Por el contrario, los dominios maliciosos tienen una gráfica asimétrica y positiva, con un promedio de 2 a 3 dígitos hasta un máximo de 7. En todo caso, cabe destacar que el 50% de los datos (primer y segundo cuartil) se sitúan en el valor 0, lo cual indica que la mitad de los datos no tienen dígitos.

Enlaces:

Por otro lado, en cuanto a los enlaces, en general, los de phishing son los que mayoritariamente tienen más dígitos, la gráfica es asimétrica y positiva, con un gran rango (de 0 a 50) lo cual nos indica que tiene una gran variabilidad de valores.

En cuanto a los enlaces de distribución de Malware, nuevamente vemos que la gráfica es bastante compacta, con un rango pequeño. Observamos que la distribución del diagrama es un poco asimétrica y negativa con una mediana de 15 dígitos. Es importante destacar que el mínimo se encuentra en 9, lo cual significa que un enlace malware sin dígitos es un dato atípico dentro del dataset. Este último valor es el esperado puesto que como hemos visto anteriormente, el 73% de enlaces malware contienen el puerto (**Figura 23**) y un 11% además el host es una IP (**Figura 22**).

En conclusión, la cantidad de dígitos, sobre todo en enlaces malware se trata de una característica significativa aún más cuando se tiene en consideración la estructura IP:Puerto.
En este caso, la IP puede alcanzar las 12 cifras: 255.255.255.255
Y el puerto hasta las 5 cifras [10]:

- Puertos bien conocidos (well-known port): 0 al 1023
- Puertos registrados: 1024 al 49151
- Puertos Dinámicos: 49152 al 65535

Por lo tanto, obtenemos un total de 17 dígitos sin contar además los dígitos adicionales en la sección del path.

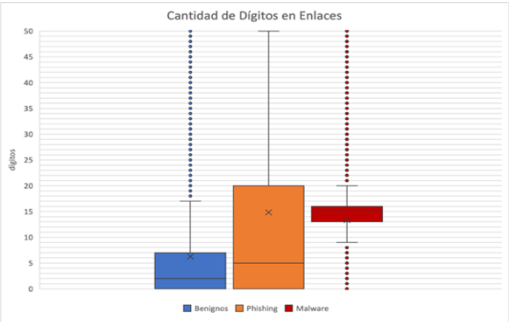


Figura 31. Cantidad de Dígitos en Enlaces. Elaboración Propia.

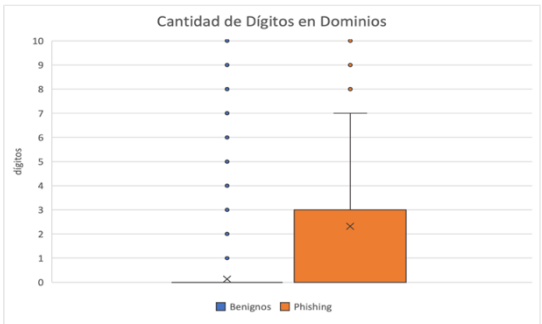


Figura 30. Cantidad de Dígitos en Dominios. Elaboración Propia.

6. Cantidad de Puntos

Otro factor importante es la cantidad de puntos que nos podría indicar la existencia de subdominios. En este caso es relevante diferenciar entre el set de dominios y el set de enlaces, puesto que el segundo en la sección path podría contener aún más puntos.

En general, observamos que en ambos datasets las cantidades no son muy dispares, comprenden entre un rango de 1 a 5 puntos.

Dominios:

En cuanto al set de dominios, vemos que los benignos en su mayoría contienen un único punto. Todo lo contrario, ocurre con el set de dominios maliciosos el cual puede llegar a contener hasta un máximo de 3 puntos. Aun así, al igual que el caso de los dígitos se trata del 50% de los datos, el resto contienen un único punto.

Enlaces:

Por otro lado, vemos en la **Figura 32** como la cantidad de puntos aumenta si se trata de un enlace benigno, phishing o malicioso.

Detectamos que la mayoría de los enlaces benignos contienen entre 1 a 3 puntos. Si comparamos este resultado con el de los dominios benignos, podemos concluir que estos 2 puntos adicionales podrían tratase de subdominios, de la extensión al final del enlace o ambos simultáneamente.

Por el contrario, los dominios phishing el promedio se sitúa en los 3 puntos y puede llegar a tener un máximo de 4 puntos. Lo cual es un ligero aumento de puntos en comparación con los benignos.

Finalmente, en la gráfica de enlaces de distribución de malware, observamos que el mínimo comprende los 2 puntos y el máximo hasta 5 puntos. Este resultado es bastante interesante puesto que un enlace malware con un único punto se considera un "outlier". Además, si tenemos en cuenta que el 11% de los enlaces el host es una IP (**Figura 22**) podemos llegar a relacionar este resultado.

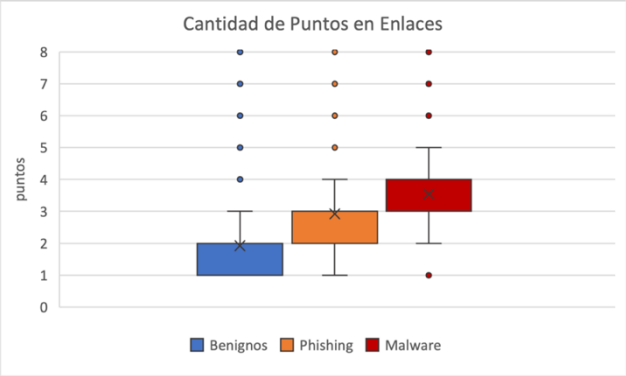


Figura 32. Cantidad de Puntos en Enlaces.

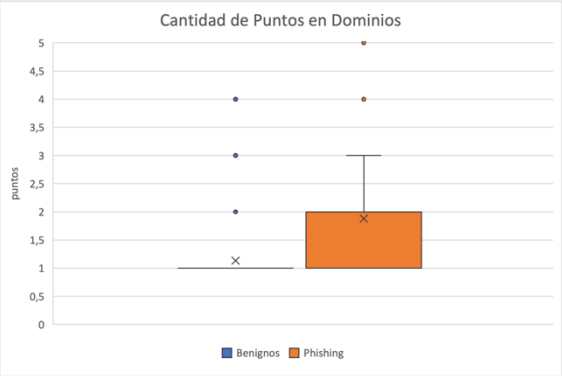


Figura 33. Cantidad de Puntos en Dominios.

7. Cantidad de Subdirectorios

Además de los puntos, otra medida interesante es contabilizar la cantidad de subdirectorios. Estos nos podrían indicar los directorios hasta localizar un recurso. En general, se espera que los enlaces benignos contengan más subdirectorios, ya que en muchos casos siguen el formato del blog, especificando en el enlace el año, categoría, tag entre otros datos hasta llegar al recurso.

En la **Figura 34** observamos que el mínimo de subdirectorios en los 3 diagramas es 1, lo cual es un dato esperado puesto que se trata de la primera barra inclinada que da inicio a la sección path de la url. Cabe destacar que las dos barras inclinadas iniciales que siguen al protocolo no se han tenido en cuenta https://.

En general, vemos como los enlaces phishing son los que contienen un mayor número de subdirectorios (aún más que los benignos) con un máximo de 7 subdirectorios. Este dato es algo sorprendente a lo normalmente esperado. Sin embargo, el 75% de los datos (tercer cuartil) giran en torno a los 3 subdirectorios, lo cual es un valor bastante parecido al del set de benignos.

Finalmente, vemos como los enlaces de distribución de malware en su mayoritaria únicamente contienen un subdirectorio, este se trata de la primera barra inclinada que inicia la sección de path.

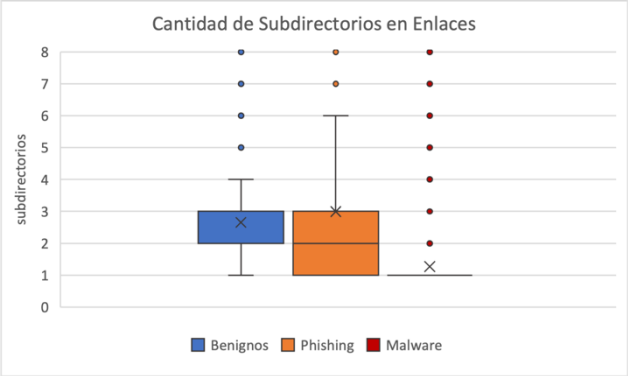


Figura 34. Cantidad de Subdirectorios. Elaboración Propia.

8. Símbolos Especiales

Los símbolos especiales que vamos a analizan son: &, ?, #, ~, @ y =.

En general, observamos que el carácter tilde "~", no ha tenido ninguna aparición dentro de los 3 datasets. Seguido a este está el carácter fragmentos "#" que ha aparecido únicamente un 2% en los enlaces Benignos y algunas pocas apariciones en los enlaces Phishing.

Estos últimos dos datasets han dado altos resultados con los símbolos ampersand "&", interrogante "?" e igual "=".

La presencia del símbolo interrogante "?", en general, indica que se está realizando una consulta o query (Query String Begin). Esta consulta suele ir acompañada por el ampersand "&" o Query String Separator, que se encarga de separar los diferentes parámetros. Por otro lado, el símbolo igual "=" suele indicar el valor de dichos atributos. Normalmente, estos 3 símbolos suelen ir en conjunto como podéis observar en la **Figura 36**.

A pesar de que existan altos valores para estos últimos 3 símbolos dentro del set de phishing, también lo son para el set de benignos (**Figura 35**). Por lo tanto, incorporaros en las alertas podrían causar muchos falsos positivos.

Por el contrario, el símbolo arroba "@" aparece en un 3% en los enlaces phishing, este símbolo no es muy habitual encontrarlo en los enlaces benignos puesto que especifica un nombre de usuario o dirección de correo electrónico.

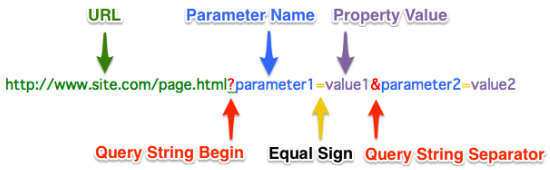


Figura 36. Caracteres Especiales en un enlace. https://community.mixpanel.com/data-management-10/use-url-query-strings-to-add-properties-articles-from-the-archive-2675

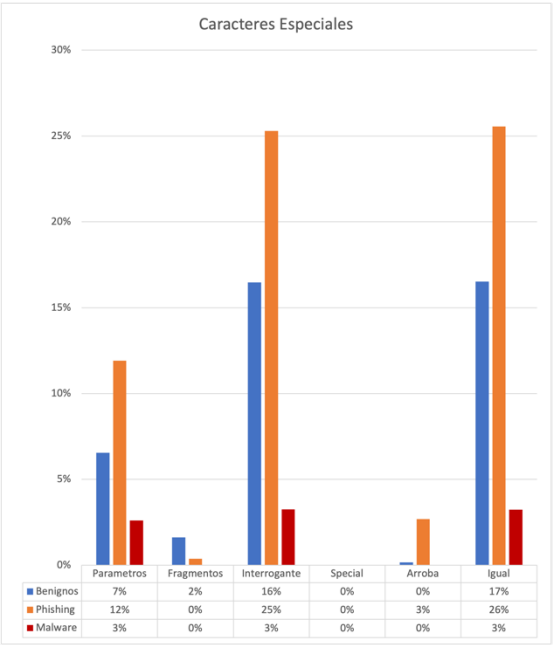


Figura 35. Cantidad de Caracteres Especiales. Elaboración Propia.

9. TDL

Otra característica interesante es analizar el Top Level Domain, el cual representa el último segmento del dominio. Vemos en los resultados de Eneyi (**Figura 38**) como los TLD *.info*, *.cn*, *.ru* y *.ca* son los 4 principales utilizados por enlaces maliciosos. Por otro lado, los TLD *.net*, *.com*, *.ua* y *.cc* son de los más empleados por enlaces benignos.

En nuestro caso, evaluaremos los top 20 TDL de cada conjunto y eliminamos los que coincidan con los enlaces benignos. El resultado es el siguiente:

Tabla 12. TOP TDL en enlaces y dominios maliciosos. Elaboración Propia.

Dominios Phishing	xyz	club	tk	app	me	ml	top	br	ga	shop
Enlaces Phishing	xyz	br	tk	in	app	club	info	ga	id	page
Enlaces Malware	81	234	194	119	131	125	102	15	148	47

En general, podemos observar en la **Tabla 12** que los TLD de los enlaces malware son numéricos. Este dato es bastante esperado puesto que como hemos visto anteriormente un 11% el host es una IP (**Figura 22**).

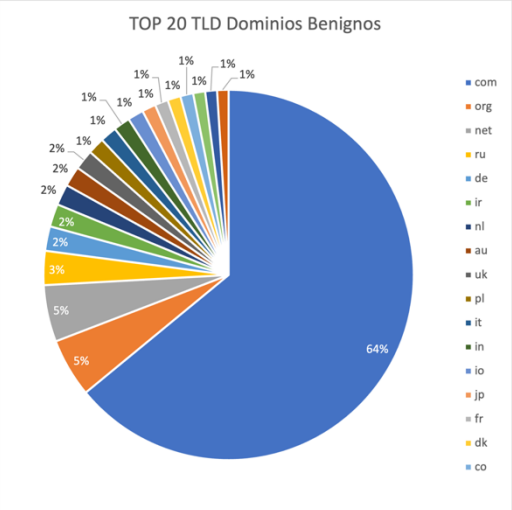


Figura 37. TOP 20 TLD Dominios Benignos. Elaboración Propia.

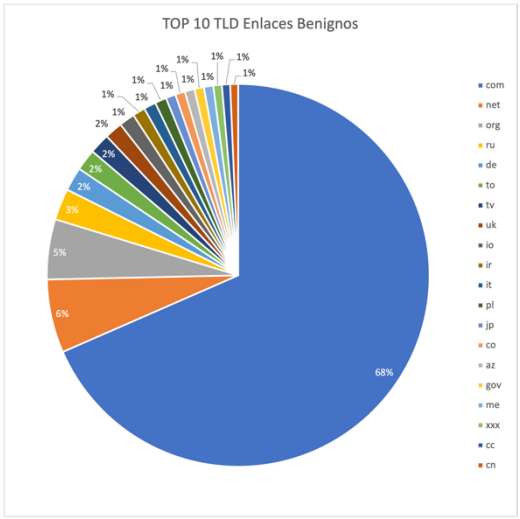


Figura 38. TOP 20 TLD Enlaces Benignos. Elaboración Propia.

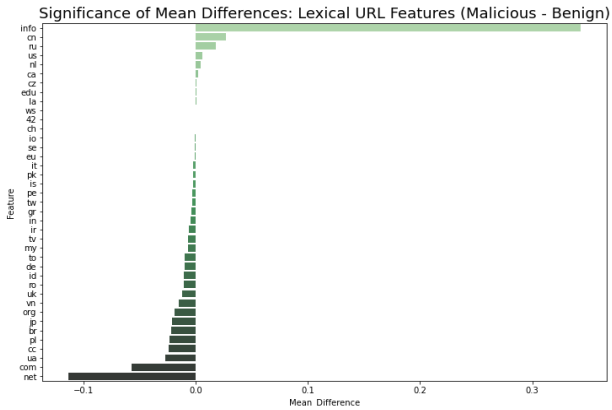


Figura 39. TOP TLD enlaces benignos y maliciosos. https://towardsdatascience.com/extracting-feature-vectors-from-url-strings-for-malicious-url-detection-cbafc24737a

10. Keywords

Por último, evaluaremos la aparición de algunas palabras clave o keywords que suelen estar incorporadas en los enlaces maliciosos. Estas son: *login*, *server*, *admin*, *client*, *password* y *redirect*.

Observamos que en general, en el set de enlaces Malware y Benignos se han obtenido resultados muy cercanos al 0% o 1% del total del dataset.

En cuanto a los enlaces phishing, este resultado ha sido algo mayor al resto de datasets. Podemos ver en la que alrededor del 17% de los datos contienen la keyword "*login*", seguida de un 3% con la keyword "*admin*" y en un tercer lugar la keyword "*client*" con un 2%.

Estos últimos datos, si los comparamos con los resultados de la dra. Eneyi (**Figura 21**), vemos que la presencia de *login* es muy empleada en enlaces benignos incluso mayor que la de maliciosos. Por lo tanto, no podemos emplear *login* como característica delimitable. Además, no es recomendable implementarla como característica delimitante puesto que cada vez que intentáramos iniciar sesión nos saltaría la alerta del *login*.

En general, las keyword *admin* o *client* son más empleadas por los enlaces maliciosos que por lo benignos, lo cual significa que podrían ser características interesantes de valorar.

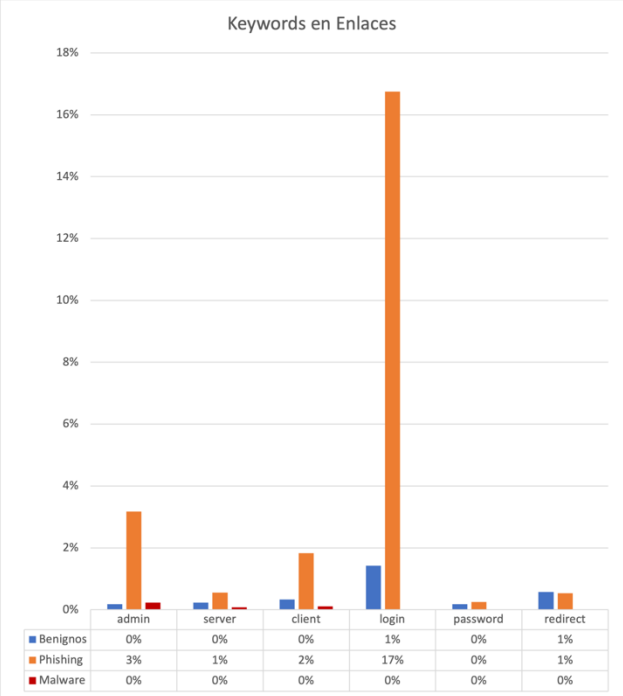


Figura 40. Cantidad de Keywords en Enlaces maliciosos.

3.3.3.4. Conclusiones Generales

Después del análisis anterior, seleccionamos la información relevante que nos permitirá delimitar entre enlaces maliciosos y benignos. En todo caso, cabe recordar que se tratan de posibles amenazas, ya que, al no existir diferencias extremadamente significativas, podrían causar muchos falsos positivos.

Conclusión 1: HTTP vs HTTPS

En general, observamos que la mayoría de los enlaces tanto phishing como malware no utilizan el protocolo de seguridad, predominando entonces el HTTP. Aun así, es preocupante el pequeño porcentaje que sí lo emplea, alrededor del 44% y 11% (**Figura 20**).

Conclusión 2: Host es IP

En cuanto al host ha resultado que el 11% de los enlaces Malware son una IP (**Figura 22**), un dato significativo, ya que en general los enlaces benignos utilizan un nombre de dominio.

Conclusión 3: Puerto

Por otro lado, la presencia del puerto en los enlaces Malware es alrededor de 73% (**Figura 23**). Este dato es bastante interesante, dado que en general los enlaces benignos no los incluyen. Aun así, solo comprobaremos que la sección host contenga el puerto, ya que no hemos encontrado unos valores de puerto predominantes en el conjunto.

Conclusión 4: Longitud

La longitud es una característica que solamente tenemos en cuenta la del host y path, ya que la longitud completa es un atributo que podría causar muchísimos falsos positivos debido a que las gráficas son bastan dispersa (**Figura 24**). En el caso de la longitud del path, hemos observado que el 52% de los enlaces malware contienen exactamente 8 caracteres, además el 12% y 10% contienen 3 o 4 caracteres respectivamente (**Figura 28**). Por otro lado, en cuanto a los dominios, estos tienen un promedio de alrededor de 30 caracteres que coincide con el máximo de los dominios benignos (**Figura 27**)por lo tanto, también lo tendremos en consideración.

Conclusión 5: Dígitos

Los dígitos son otro parámetro característico de los enlaces Malware, con un mínimo de 9 dígitos hasta un máximo de 20 dígitos por enlace (**Figura 31**). Esta gran cantidad podría deberse a que muchos enlaces incluyen el puerto o la sección del host se trata de una IP. También vemos este aumento de dígitos en los dominios Phishing con una media de 2 hasta un máximo de 7 dígitos (**Figura 30**). Lo cual es bastante significativo, dado que los dominios benignos la gran mayoría no contienen ninguno. Finalmente, en cuanto a los enlaces Phishing, existe mucha variabilidad en el set de los datos, pero en general la cantidad de dígitos es bastante alta llegando al máximo de 50 dígitos por enlace. Por lo tanto, tendremos en cuenta a partir a del promedio 15 dígitos, el cual coincide aproximadamente con el máximo de los enlaces benignos.

Conclusión 6: Puntos

En cuanto a los puntos hemos observado que los enlaces y dominios maliciosos en general contienen más puntos que los benignos. Vemos que los enlaces malware tienen un mínimo de 2 puntos hasta un máximo de 5 puntos, mientras que los benignos el máximo se establece en los 3 puntos (**Figura 32**). Por otro lado, en cuanto a los dominios, los maliciosos alcanzan un máximo de 3 puntos mientas que los benignos la gran mayoría solamente tienen 1 punto (**Figura 33**).

Conclusión 7: Subdirectorios

En general, los resultados no han sido suficientemente notorios y delimitantes como para incorpóralos en el set de reglas. Observamos como el promedio de subdirectorios en enlaces phishing y benignos son bastante parecidos, y en cuento a los enlaces malware apenas contienen 1 directorio (la barra final indicando el inicio de la sección path) (**Figura 34**).

Conclusión 8: Símbolos Especiales

En relación con los símbolos especiales, en general los resultados entre benignos y maliciosos son bastante semejantes, por lo tanto, no podemos utilizarnos como características delimitantes. Sin embargo, en relación con el carácter arroba "@" ha aparecido alrededor de un 3% (**Figura 35**) en los enlaces phishing, lo incorporaremos, ya que a diferencia de los anteriores se trata de un símbolo atípico en los enlaces.

Conclusión 7: TLD

En cuanto a los TLD, valoramos los más frecuentes (TOP 20) que no sean comunes entre los enlaces y dominios benignos. Estos son: *.xyz*, *.tk*, *.info* y *.club*. Por otro lado, en cuanto a los enlaces malware, seleccionamos los primeros 4 TLD números: *.81*, *.234*, *.194* y *.119*.

Conclusión 8: Keywords

Finalmente, hemos detectado que la keyword más empleada solamente por enlaces maliciosos es *admin* seguida de *client*.

Conclusión Final

Observamos que los enlaces Malware han sido, en general, más fáciles de delimitar que los enlaces o dominios Phishing. Esto podría estar relacionado con que el Phishing está más orientado a engañar al usuario, por lo tanto, debe ser lo más parecido a un enlace benigno. Por otro lado, una vez ganada la confianza del usuario, mediante Phishing e Ingeniería Social, se incorporan estos enlaces de distribución de Malware.

3.3.3.5. Diseño de Reglas

Finalmente, y gracias a las conclusiones anteriores, hemos formulado un conjunto de reglas de detección para enlaces y dominios phishing y posibles enlaces malware, en general, enlaces maliciosos.

En la **Tabla 13**, vemos que hemos podido formular hasta 22 reglas, de las cuales 11 están relacionadas con enlaces de distribución de malware, 10 con el Phishing y 1 genérica.

Si diferenciamos entre las peticiones DNS y HTTP, el conjunto de reglas aumenta a 31.

Tabla 13. Diseño de Reglas. Elaboración Propia.

#	REGLA	HTTP	DNS	CATEGORÍA
0	Si el puerto desino es **80** (HTTP)	X		Posible Enlace Malicioso
1	Si el host es una IP **{1-3}.{1-3}.{1-3}.{1-3}**	X		Posible Enlace Malware
2	Si contiene el Puerto. **":"{1,5}**	X		Posible Enlace Malware
3	Si la longitud del dominio es + 30 **caracteres**.	X		Posible Dominio Phishing
4	Si la longitud del path es **8 caracteres**.	X		Posible Enlace Malware
5	Si la longitud del path es **3 caracteres**.	X		Posible Enlace Malware
6	Si la longitud del path es **4 caracteres**.	X		Posible Enlace Malware
7	Si el enlace contiene + **9 dígitos**.	X		Posible Enlace Malware
8	Si el enlace contiene + **15 dígitos**.	X		Posible Enlace Phishing
9	Si el enlace contiene entre **4 y 6 puntos**.	X		Posible Enlace Malware
10	Si el dominio/host contiene más de **3 puntos**.	X	X	Posible Enlace Phishing
11	Si el enlace contiene el símbolo **arroba** "**@**".	X		Posible Enlace Phishing
12	Si la IP termina en **.81**	X	X	Posible Enlace Malware
13	Si la IP termina en: **.234**	X	X	Posible Enlace Malware
14	Si la IP termina en: **.194**	X	X	Posible Enlace Malware
15	Si la IP termina en: **.119**	X	X	Posible Enlace Malware
16	Si el dominio termina en **.xyz**	X	X	Posible Enlace Phishing
17	Si el dominio termina en **.tk**	X	X	Posible Enlace Phishing
18	Si el dominio termina en **.info**	X	X	Posible Enlace Phishing
19	Si el dominio termina en **.club**	X	X	Posible Enlace Phishing
20	Si el path contiene keyword **admin**.	X		Posible Enlace Phishing
21	Si el path contiene keyword **client**.	X		Posible Enlace Phishing

En la sección de la implementación y evaluación, podrán ver estas reglas en un formato compatible con Suricata.

3.3.4. Diseño del SIEM

Anteriormente, en la sección background, hemos visto que es un SIEM y sus principales características, pero esta no es la única arquitectura de gestión de seguridad que existe hoy en día.

Actualmente, existen otras estructuras para gestionar los sistemas como son el SOAR o CSMA.

- **SOAR** (Security Orchestration Automation and Response) se trata de una arquitectura que mejora la detección y las respuestas a las amenazas mediante la correlación de grandes volúmenes de datos y la automatización de tareas [32].
- **CSMA** (CyberSecurity Mesh Architecture): es una de las últimas estrategias en seguridad con una arquitectura de malla y en forma de capas. Entre sus características destaca la inteligencia de ciberseguridad la cual, ente otras funciones, permite proteger cada dispositivo de forma independiente teniendo en cuenta sus características, firewall entre otros aspectos [36].

Como podemos observar, la última arquitectura es bastante interesante, puesto que se considera el dispositivo final al que desea proteger. Por otro lado, la arquitectura SOAR agrupa y analiza un gran cúmulo de datos provenientes de distintas fuentes, lo cual hace más rigurosos sus análisis, además de la automatización paródica de tareas.

En nuestro caso, al tratarse de un sistema destinado a proteger un entorno local, no serán necesarias complejas arquitecturas como son estas dos últimas. Un sistema SIEM es suficiente.

3.3.4.1. Estrategia SIEM

El primer paso para iniciar con este diseño es establecer un listado de metas u objetivos, también llamados casos de negocio, que nos ayudarán a particionar el sistema en pequeñas tareas [31].

A lo largo de internet, podemos encontrar diferentes guías que nos pueden ayudar a establecer puntos de control para tener en cuenta en el momento de diseñar un SIEM. Pero por lo general suelen estar dirigidas para entornos Empresariales, por lo tanto, la mayoría no serían servibles para nuestro proyecto. En el diseño de este SIEM local, tenemos que enfocarnos en qué información sería útil gestionar en un entorno doméstico.

Principalmente, hemos identificado tres tipos de información:

- **MÉTRICAS**: Información relacionada con el estado de las máquinas que ejecutan los sistemas SIEM e NIDS.
- **EVENTOS**: Información relacionada con el estado de la red.
- **ALERTAS**: Información sobre las amenazas que detecta el NIDS.

Esta información la mostramos en paneles diferentes y cada una de ellas en dos modalidades:

- *Resumen*: muestra información general y básica de cada sección.
- *Avanzado*: muestra información más detallas y desglosada que la sección Resumen.

Son importantes ambos modos, ya que el primero nos permite ver rápidamente una periférica del sistema y el segundo se accede cuando vemos algo anormal en la vista “Resumen”, lo cual nos permitirá identificar el origen de la anomalía.

Figura 41. Módulos SIEM. Elaboración propia.

3.3.4.2. Métricas de las máquinas

Estas métricas muestran el estado de los dispositivos que ejecutan los sistemas SIEM e NIDS. Se trata de una información importante, ya que podría ocurrir que no soporten la carga de trabajo o el disco de almacenamiento esté lleno, entre otras incidencias. En la **Tabla 14** podéis ver los principales elementos que visualizaremos.

Tabla 14. Información Panel de Métricas. Elaboración Propia.

#	Elemento	Características
1	CPU	Uso de la CPU. Cantidad de procesos. Top procesos que consumen más CPU. Consumo de la CPU a lo largo del día.
2	RAM	Uso de la memoria RAM. Top procesos que consumen más RAM. Consumo de la memoria RAM a lo largo del día. Memoria SWAP.
3	DISCO	Espacio Disponible en el disco. Lecturas y escrituras del disco a lo largo del día.

3.3.4.3. Información del estado de la Red

Se trata de información básica sobre diferentes aspectos de la red. Esta información se recibirá de los paquetes tipo "stat" que envía Suricata. En la **Tabla 15** podemos ver las características específicas. En esta sección, también hemos incluido las páginas sin certificado SSL más visitadas, esta información no se ha considerado una alerta, ya que si se tratara de una amenaza aparecería en el panel de Alertas, más bien se ha incluido para conocer el comportamiento de los usuarios de la red en relación con el acceso a páginas no seguras.

Tabla 15. Información Panel de Eventos. Elaboración Propia.

	Elemento	Características
1	PAQUETES	Paquetes capturados por el NIDS. Paquetes perdidos por el NIDS. Paquetes capturados según el momento del día (momento del día con mayor actividad). Top protocolos red más utilizados. Top protocolos de transporte más utilizados (TCP/UDP). Porcentaje de HTTP vs HTTPS.
2	HOSTS	Hosts conectados Ahora (últimos 10 min). Actividad de los hosts a lo largo del día.
3	SSL	TOP páginas visitadas sin el certificado. TOP usuarios visitados sin el certificado.

3.3.4.4. Información sobre las Amenazas

Por último, en el panel de amenazas vemos información relacionada con las detecciones del NIDS. En esta sección separamos las amenazas provenientes de las blacklists de las "posibles amenazas" obtenidas del estudio de características.

Tabla 16. Información Panel de Amenazas. Elaboración Propia.

#	Elemento	Características
1	GENERAL	Cantidad de alertas detectadas. Alertas detectadas a lo largo del día. Top hosts que han generado más alertas.
2	POSIBLES AMENAZAS	Top Reglas que han generado más alertas. Listado Reglas.
3	BLACKLISTS	TOP blacklists que han generado más alertas. Listado de Alertas de cada blacklist. TOP alertas de cada blacklist.

3.3.4.5. Selección de herramientas

La herramienta seleccionada para la implementación del SIEM, han sido el set ELK Elasticsearch-Kibana-Logstash, aunque este último lo hemos sustituido por el set Beats (Filebeat y Metricbeat) debido a que ofrecen un mejor rendimiento [29].

Esta decisión se ha tomado en base a tres criterios.
Primeramente, Elastic aparece en el artículo "*Soluciones open source para la gestión de logs en ciberseguridad*" [33] escrito por Rubén Gª Ramiro (Security Manager de Telefónica). El cual propone y muestra las ventajas del conjunto de herramientas Elastic como posible SIEM.

En segundo lugar, vemos en la **Figura 42** como Elastic está posicionado en el *Gartner Magic Quadrant for SIEM*[56] los cuales desde hace años se encargan de analizar y evaluar las diferentes soluciones SIEM que existen en el mercado. De entre todas las herramientas que aparecen en el cuadrante, Elastic es la única Open Source.

Por último, esta herramienta cumple con los siguientes criterios que nos permitirá implementarla:

1. Herramientas Open Source: buscamos una solución de bajo coste.
2. Agregación de datos a través de la Interfaz de red: prominentes del NIDS.
3. Permitir el acceso al SIEM desde cualquier dispositivo de la red.
4. Permitir la correlación de la información.
5. Retención de los datos durante un largo período.
6. Almacenamiento de Registros en bases de datos orientadas a documentos: al ser logs de distinta índole (distintos parámetros para cada paquete de red), facilita la integración.
7. Incorpore una interfaz que permita diseñar paneles personalizados.
8. Que exista algún proyecto similar en el cual se haya utilizado la herramienta o conjunto de herramientas con un NIDS como es Suricata: encontramos el proyecto del ingeniero de sistemas Héctor Herrero Hermida[57] [19].

Figura 42. Gartnier Magic Cuadrant.
https://www.elastic.co/es/campaigns/2021-gartner-magic-quadrant-siem

[56] https://www.elastic.co/es/campaigns/2021-gartner-magic-quadrant-siem
[57] https://www.linkedin.com/in/hectorherrero/

3.3.4.6. Agregación de datos

A continuación, vamos a ver como es el flujo de envío de datos desde el NIDS hacia el SIEM y como se relacionan las herramientas seleccionadas en el apartado anterior. En la **Figura 2** podéis ver el esquema básico de la agregación de datos al SIEM. Este funciona de la siguiente manera:

1. [NIDS] Suricata analiza el tráfico de la red local y genera logs.
2. [NIDS] Filebeat está atento a estos logs y los envía de forma continua a Elasticsearch (SIEM).
3. [SIEM] Elasticsearch analiza los datos y los visualiza con Kibana.
4. [SIEM] Kibana visualiza los datos analizados mediante paneles y gráficas.

Figura 43. Agreación de datos al SIEM. Elaboración Propia.

3.3.4.7. Elasticsearch

Elasticsearch es un motor de búsqueda y análisis Open Source desarrollado por Shay Banon en 2004 y lanzado en febrero de 2010. Se trata de una herramienta que permite realizar búsquedas sobre grandes volúmenes de datos, con una gran velocidad y alta escalabilidad [4].

Entre sus características y funcionalidades destacamos:

1. **Recogida de datos:** recaudar datos de varias fuentes como pueden ser los logs, métricas de sistema, aplicaciones web, entre otros.
2. **Procesamiento**: procesar y normalizar estos datos, que pueden provenir con diferentes formatos: formatos: numéricos, espaciales, textuales, estructurales y no estructurales
3. **Documentos JSON:** los datos se guardan en una colección de documentos JSON.
4. **Búsqueda rápida:** utiliza una estructura llamada índice invertido para realizar búsquedas de texto completo de manera rápida.

Elasticsearch ha sido desarrollado a base de Apache Lucene, una API para recuperación de la información implementada en Java (aunque ahora se encuentra disponible para más lenguajes), y se basa sobre una licencia Apache.

3.3.4.8. Kibana

Kibana es una herramienta Open Source que permite la búsqueda de datos y su visualización de forma personalizada. Esta herramienta pertenece al conjunto de Elastic Stack y ha sido desarrollada en 2013 por la comunidad de Elasticsearch [41].

Entre sus principales características destacamos:

1. **Interfaz gráfica:** dispone de una interfaz gráfica que permite crear elementos de visualización personalizados.
2. **Búsqueda de información:** permite realizar búsquedas sobre diferentes fuentes de datos.
3. **Visualización:** permite la visualización de datos de forma personalizada y a través de diferentes elementos gráficos.
4. **Paneles:** permite agrupar diferentes gráficas bajo un mismo panel o tablero, facilitando de esta manera su compartición y visualización.
5. **Funcionalidades avanzadas**: dispone de funcionalidades avanzadas como son el Machine Learning, monitorización, diseño de alertas o controles, gestión de grandes volúmenes de datos, entre otros.

Kibana es una herramienta gratuita con la licencia Elastic o SSPL que nos permite crear nuestras propias visualizaciones independientemente de la fuente de datos.

3.3.4.9. Filebeat

Filebeat es una herramienta que pertenece al set Beats[58] de Elasticsearch que también pertenece al conjunto de Elastic Stack. La funcionalidad principal de Filebeat es la lectura de ficheros, sobre todo logs, aunque pueden ser de diferentes tipos. El uso más común que se le puede dar a Filebeat es la recogida de datos en diferentes equipos remotos para su futura centralización en Elasticsearch [18].

Entre sus características destacan:

1. Gestión de logs: gestiona la cantidad de logs, la estructura, el tiempo, entre otras características, antes de enviar al equipo destino.
2. Dispone de módulos con una escultura básica para los logs de diferentes servicios. Por ejemplo, Filebeat tiene módulos para Suricata, Snort y Zeek (Bro), herramientas IDS que hemos visto anteriormente.
3. Permite enviar datos a equipos remotos.

3.3.4.10. Metricbeat

Finalmente, la última herramienta que implementaremos es Metricbeat. Esta, al igual que Filebeat, también pertenece al set de Beats de Elasticsearch y nos permite la recolección de métricas del sistema como son la cpu, cores, memoria RAM, disco, etc. Además, se trata de una herramienta ligera que no nos afectará negativamente en el rendimiento [28].

[58] Beats: son agentes de datos ligeros, se encargan de extraer y recolectar los datos de los hosts o servidores [18].

3.3.5. Diseño Sistema de Alertas

Por último, mostramos el diseño del sistema de Alertas. Este está compuesto de dos elementos: un sistema visual y un sistema auditivo. Debido a que estamos empleando la versión gratuita del set de Elastic, no podemos utilizar el sistema de alertas que incorpora Kibana, por lo tanto, tendremos que desarrollar uno manualmente

Este sistema se trata de un servicio que estará atento a los logs que genera el NIDS, ya que al detectar una amenaza activará una alerta. El hardware de estas alertas son leds (rojo, amarillo y verde) y un buzzer activo (alarma) que están conectados a una protoboard y a los pines GPIO de las Raspberry Pi.

3.3.5.1. Categorización de las Alertas

Un aspecto importante es la diferenciación entre los tipos de alertas, ya que no podemos activar la alarma con una "posible" amenaza.

Dentro de los ficheros log que genera Suricata al detectar una amenaza incluye un parámetro de prioridad, este puede llegar a comprender desde 1 a 255, (aunque normalmente se emplean las 4 primeras[59]) siendo el número 1 la prioridad más alta (Reglas Suricata).

En nuestro caso, empleamos las prioridades 1, 2 y 3.

Tabla 17. Características Sistema de Alertas. Elaboración Propia.

Prioridad	Definición	Elementos
1	Prioridad Máxima.	Led Rojo. Activar Alarma.
2	Prioridad Intermedia.	Led Amarillo
3	Prioridad Mínima. "Posible Amenaza"	Led Verde.

En la tabla anterior, vemos que el led verde se enciende con las posibles amenazas generadas por el set de reglas creadas por nosotros. Normalmente, el color verde se encuentra asociado a la correctitud del sistema, pero en ese caso, al tratarse de "posibles amenazas", no queremos alertar con colores alarmantes como son el rojo o el amarillo, puesto que podría confundir a los usuarios.

[59] https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Meta-settings#:~:text=The%20priority%20keyword%20comes%20with,The%20highest%20priority%20is%201.

3.3.6. Diseño Final

El diseño final es el mostrado en la **Figura 44**. El procedimiento general es el siguiente:

1. Un usuario se conecta a la red, por ejemplo, vía Acces Point, e inicia la comunicación.
2. El Acces Point envía estos datos al switch el cual redirige las peticiones al Router, pero a su vez copia el tráfico y lo retorna por el puerto 5, donde está el NIDS.
3. Una vez el NIDS recibe los datos, Suricata los analiza en busca de amenazas y genera ficheros log, tanto de estado, eventos como alertas.
4. Estos logs son interceptados por Filebeat el cual envía los datos a Elasticsearch (SIEM).
5. Una vez Elasticsearch recibe los datos, los analiza y transforma, de tal manera que Kibana pueda visualizarlos por una pantalla. En este caso, la Raspberry Pi SIEM dispone de su propia pantalla.
6. De hecho, cualquier usuario de la red que conoce las credenciales de acceso, podrá visualizar los paneles de Kibana.
7. Finalmente, el sistema de Alertas está pendiente de escuchar los logs que genera Suricata en relación con las amenazas. En el caso de detectar alguna, alerta de forma visual o auditiva según la peligrosidad detectada.

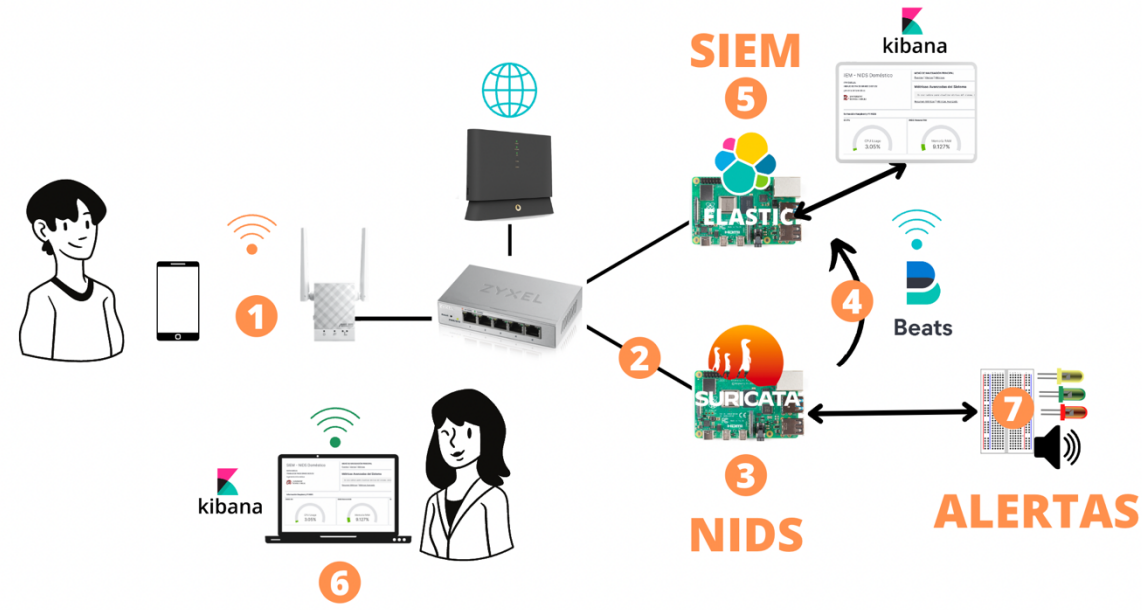


Figura 44. Diseño Final del Prototipo. Elaboración Propia.

4. Implementación y Evaluación

En esta fase detallamos los procedimientos seguidos para crear el arquetipo diseñado anteriormente, además de la evaluación de cada uno de los subsistemas. Esta sección comprenderá la implementación y evaluación de: i) Arquitectura; ii) NIDS; iii) SIEM; iv) Sistema de Alertas; v) Evaluación Final.

4.1. Arquitectura

La implementación de la arquitectura consiste en configurar los siguientes dispositivos:

1. Configurar Switch para hacer port mirroring.
2. Configurar Repetidor como Acces Point.
3. Añadir IPs estáticas y Filtro MAC
4. Instalar SO en las Rapsberry Pi.
5. Evaluar Arquitectura.

Con esto tenemos la base lista para iniciar la implementación del NIDS y del SIEM.

4.1.1. Configuración Port Mirroring

La configuración del port mirroring es bastante sencilla gracias a la interfaz intuitiva que ofrece el switch seleccionado.

Para configurar el port mirroring, primero accedemos desde el navegador al switch, mediante la ip 192.168.1.3. Una vez dentro, localizamos la pestaña "*Mirroring*" y editamos los siguientes parámetros:

- *Port Mirroring*: Enable
- *Mirror Diretion*: Both
- Todos los puertos serán "Mirrored Port" salvo el puerto número 5.

En este caso, la distribución de puertos del switch es la siguiente:

- **Puerto 1**: Router.
- **Puerto 2**: AP.
- **Puerto 3**: (vacío) disponible para cualquier dispositivo que quiera conectarse vía Ethernet.
- **Puerto 4**: Raspberry Pi con el SIEM.
- **Puerto 5**: Raspberry Pi con el NIDS.

Como el NIDS está conectado en el puerto 5, este recibirá una copia de todo el tráfico que circula por la red.

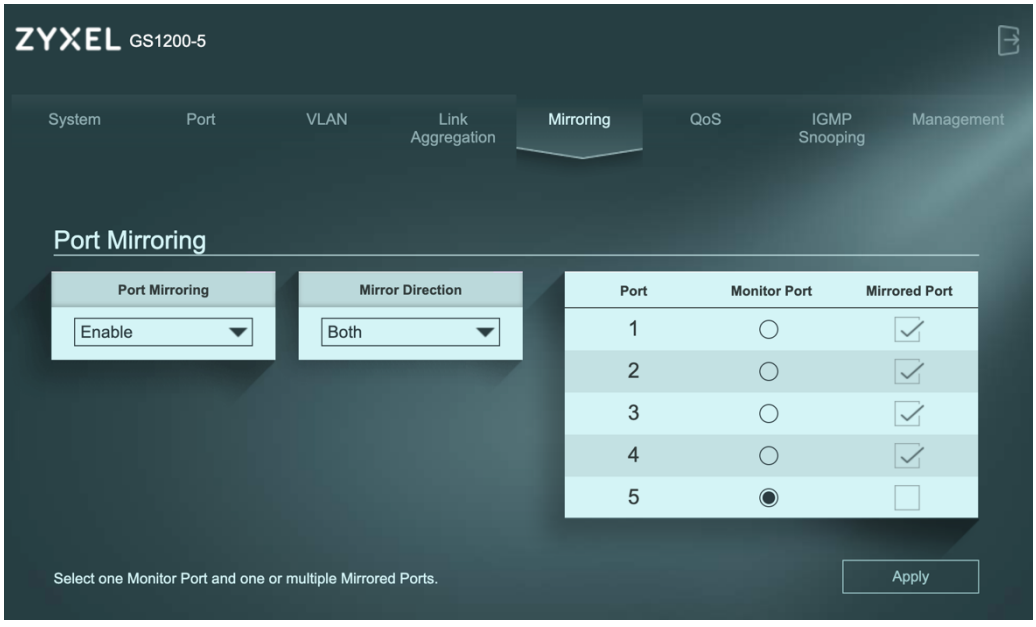


Figura 45. Configuración Port Mirroring. Elaboración Propia.

4.1.2. Configuración Access Point

Después de configurar el switch, tenemos que garantizar que el tráfico que circula vía WiFi pueda ser capturado y enviado al NIDS para su análisis. Para ello, configuramos el repetidor ASUS RP-AC51 como un Access Point y lo conectamos a un "Mirrored Port" del Switch anterior. Comprobamos la conectividad y deshabilitaremos la funcionalidad del WiFi del Router, de esta manera los dispositivos que quieran conectarse de forma Wireless lo harán únicamente desde el AP.

La instalación del switch se ha llevado a cabo mediante la aplicación móvil *ASUS Extender*[60]. Esta nos ha permitido configurar el SSID, credenciales de acceso al dashboard, la IP del AP y comprobar la conectividad. Una vez establecida la conexión entre el Router y el repetidor, procederíamos entonces a acceder al panel de control del dispositivo mediante las credenciales anteriormente creadas y la IP 192.168.1.4.

Dentro del panel, vamos a localizar en el menú lateral la sección "Administration" y dentro de este último veremos una ventana sobre Modos de Operación. Seleccionamos del listado Access Point y con esto ya tendríamos el dispositivo configurado correctamente.

Finalmente, accedemos al panel de control del Router a través de la IP 192.168.1.1 y localizamos la sección WiFi. Dentro de esta, en la pestaña "General", deshabilitadnos los dos SSID (2,4GHz) y (5GHz).

Con esta última configuración, ya tenemos la red local configurada y funcional.

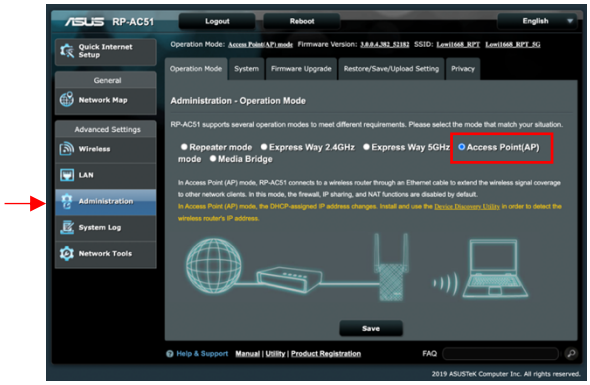


Figura 48. Configuración AP. Elaboración Propia.

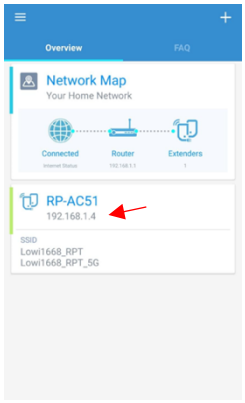


Figura 47. Conexión AP. Elaboración Propia.

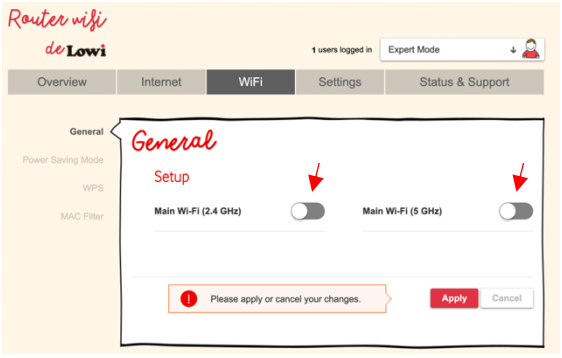


Figura 46. Configuración Router. Elaboración Propia.

[60] https://play.google.com/store/apps/details?id=com.asus.aiextender&hl=es&gl=US

4.1.3. Configuración Filtro Mac e IP estáticas

Aplicaremos un filtro MAC para que únicamente los disipativos registrados en la lista tengan acceso a la Red. Como hemos desactivado el Wifi en el router, esta acción la realizaremos desde el Acces Point ya configurado.

En la **Figura 50** podéis observar algunas IPs estáticas establecidas desde el Router.

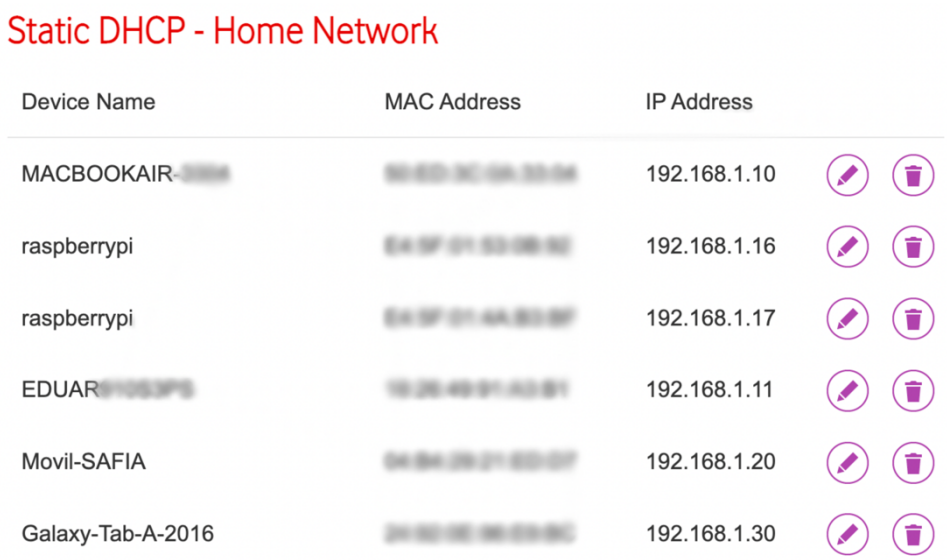


Figura 50. IPs estáticas. Elaboración Propia.

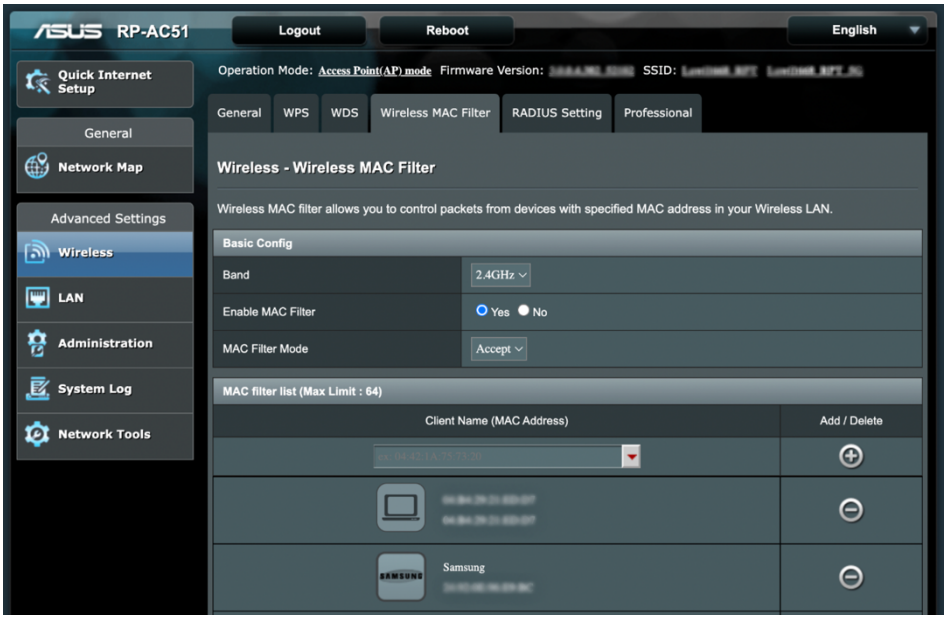


Figura 49. Filtro MAC. Elaboración Propia.

4.1.4. Configuración Raspberry Pi

Finalmente, terminamos la implementación de la arquitectura con la instalación de los Sistemas Operativos en las Raspberry Pi. Además, una vez instalados y funcionales, verificamos el funcionamiento del Port mirroring y configuramos el acceso vías ssh para poder gestionarlas fácilmente desde nuestro dispositivo habitual.

Para la instalación de los Sistemas Operativos en las tarjetas SD, emplearemos el software *Raspberry pi Imager*[61]. Este dispone de una interfaz muy sencilla que nos permitirá seleccionar que sistema operativo deseamos instalar y en pocos minutos tenerlo instalados en la SD.

En este caso, para la Raspberry Pi que actúa de NIDS instalamos *Raspberry Pi Lite*, una versión sin escritorio, dado que no es necesario. Por otro lado, la Raspberry que actúa de SIEM está al tener incorporada una pantalla, por lo tanto, instalamos el mismo SO pero con el desktop incorporado.

Una vez terminada la instalación, incorporamos las tarjetas SD a los dispositivos y los conectamos al Switch. En este caso, la Raspberry que actúa de NIDS es conectada específicamente al puerto número 5 que es el que recibe la copia del tráfico.

Después de conectar los dispositivos al switch y comprobar su conectividad, iniciamos la configuración mediante la comanda *raspi-config*, esta la empleamos para: configurar el timezone, el teclado, el idioma, cambiar el hostname y habilitar el ssh. Además de actualizar el sistema.

```
sudo raspi-config
```

Los hostname serán efectivamente NIDS y SIEM, además configuraremos el usuario safia para cada una de las máquinas.

Finalmente, probamos el acceso vía túnel a los dispositivos mediante la comanda:

```
ssh safia@192.168.1.16
```

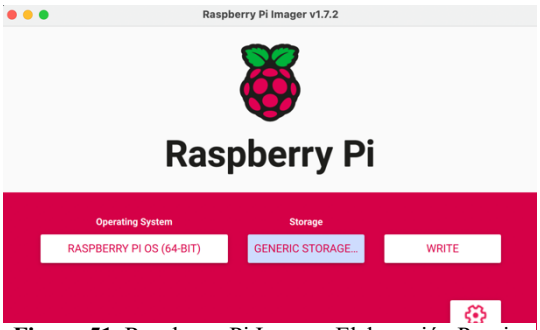


Figura 51. Raspberry Pi Imager. Elaboración Propia.

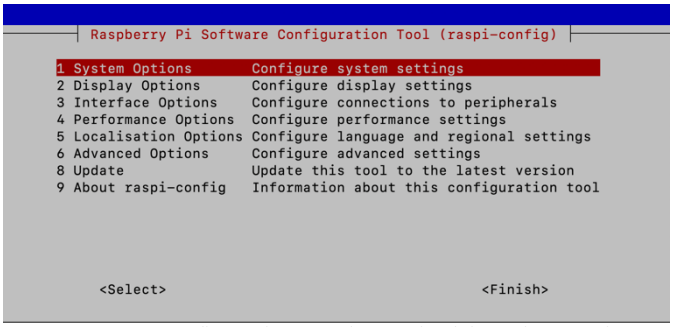


Figura 52. Configuración Raspberry Pi. Elaboración Propia.

[61] https://www.raspberrypi.com/software/

4.1.5. Verificación Port Mirroring

Finalmente, y antes de iniciar con la implementación del NIDS, tenemos que verificar que estamos recibiendo correctamente los paquetes, sobre todo, los enviados vía Wireless, que en general suelen ser la mayoría.

Para la prueba accederemos desde el dispositivo con IP 192.168.1.10, el cual está conectado vía WiFi al Acess Point, a una página sin certificado SSL (HTTP), esto nos permite ver de forma desencriptada el contenido del paquete.

Accederemos a la página: http://redesdecomputadores.umh.es/aplicacion/HTTP.htm

Desde la máquina NIDS utilizaremos la herramienta tcpdump para la captura del paquete. Además de este ejemplo se han realizado pruebas más sencillas, por ejemplo, con el protocolo ICMP y pings, pero, en definitiva, se puede comprobar que la recolección de paquetes por parte del dispositivo NIDS y port mirroring es correcta.

```
sudo tcpdump -nnSX port 80 and host 192.168.1.10
```

OUTPUT

```
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
15:10:34.813290 IP 192.168.1.10.63506 > 193.147.147.81.80: Flags [P.], seq
1164831686:1164832287, ack 2693924623, win 65535, length 601: HTTP: GET /aplicacion/HTTP.htm
HTTP/1.1
        0x0000:  4500 0281 0000 4000 4006 21e0 c0a8 010a  E.....@.@.!.....
        0x0010:  c193 9351 f812 0050 456d ebc6 a092 070f  ...Q...PEm......
        0x0020:  5018 ffff 0122 0000 4745 5420 2f61 706c  P....".."GET./apl
        0x0030:  6963 6163 696f 6e2f 4854 5450 2e68 746d  icacion/HTTP.htm
        0x0040:  2048 5454 502f 312e 310d 0a48 6f73 743a  .HTTP/1.1..Host:
        0x0050:  2072 6564 6573 6465 636f 6d70 7574 6164  .redesdecomputad
        0x0060:  6f72 6573 2e75 6d68 2e65 730d 0a43 6f6e  ores.umh.es..Con
        0x0070:  6e65 6374 696f 6e3a 206b 6565 702d 616c  nection:.keep-al
        0x0080:  6976 650d 0a43 6163 6865 2d43 6f6e 7472  ive..Cache-Contr
        0x0090:  6f6c 3a20 6d61 782d 6167 653d 300d 0a55  ol:.max-age=0..U
        0x00a0:  7067 7261 6465 2d49 6e73 6563 7572 652d  pgrade-Insecure-
        0x00b0:  5265 7175 6573 7473 3a20 310d 0a55 7365  Requests:.1..Use
        0x00c0:  722d 4167 656e 743a 204d 6f7a 696c 6c61  r-Agent:.Mozilla
        0x00d0:  2f35 2e30 2028 4d61 6369 6e74 6f73 683b  /5.0.(Macintosh;
        0x00e0:  2049 6e74 656c 204d 6163 204f 5320 5820  .Intel.Mac.OS.X.
        0x00f0:  3130 5f31 355f 3729 2041 7070 6c65 5765  10_15_7).AppleWe
        0x0100:  624b 6974 2f35 3337 2e33 3620 284b 4854  bKit/537.36.(KHT
        0x0110:  4d4c 2c20 6c69 6b65 2047 6563 6b6f 2920  ML,.like.Gecko).
        0x0120:  4368 726f 6d65 2f31 3033 2e30 2e30 2e30  Chrome/103.0.0.0
        0x0130:  2053 6166 6172 692f 3533 372e 3336 0d0a  .Safari/537.36..
        0x0140:  4163 6365 7074 3a20 7465 7874 2f68 746d  Accept:.text/htm
        0x0150:  6c2c 6170 706c 6963 6174 696f 6e2f 7868  l,application/xh
        0x0160:  746d 6c2b 786d 6c2c 6170 706c 6963 6174  tml+xml,applicat
        0x0170:  696f 6e2f 786d 6c3b 713d 302e 392c 696d  ion/xml;q=0.9,im
        0x0180:  6167 652f 6176 6966 2c69 6d61 6765 2f77  age/avif,image/w
        0x0190:  6562 702c 696d 6167 652f 6170 6e67 2c2a  ebp,image/apng,*
        0x01a0:  2f2a 3b71 3d30 2e38 2c61 7070 6c69 6361  /*;q=0.8,applica
        0x01b0:  7469 6f6e 2f73 6967 6e65 642d 6578 6368  tion/signed-exch
        0x01c0:  616e 6765 3b76 3d62 333b 713d 302e 390d  ange;v=b3;q=0.9.
        0x01d0:  0a41 6363 6570 742d 456e 636f 6469 6e67  .Accept-Encoding
        0x01e0:  3a20 677a 6970 2c20 6465 666c 6174 650d  :.gzip,.deflate.
        0x01f0:  0a41 6363 6570 742d 4c61 6e67 7561 6765  .Accept-Language
        0x0200:  3a20 656e 2d47 422c 656e 2d55 533b 713d  :.en-GB,en-US;q=
        0x0210:  302e 392c 656e 3b71 3d30 2e38 2c65 733b  0.9,en;q=0.8,es;
        0x0220:  713d 302e 370d 0a49 662d 4e6f 6e65 2d4d  q=0.7..If-None-M
        0x0230:  6174 6368 3a20 2237 6237 3736 3538 3662  atch:."7b776586b
        0x0240:  6639 3063 3731 3a34 3636 220d 0a49 662d  f90c71:466"..If-
        0x0250:  4d6f 6469 6669 6564 2d53 696e 6365 3a20  Modified-Since:.
        0x0260:  4d6f 6e2c 2030 3720 4d61 7920 3230 3037  Mon,.07.May.2007
        0x0270:  2031 353a 3531 3a30 3720 474d 540d 0a0d  .15:51:07.GMT...
        0x0280:  0a
```

4.2. NIDS

Después de tener la arquitectura implementada, testeada y funcional, iniciaremos con la propia implementación del Sistema de Detección de Intrusos en la Red (NIDS).

Esta sección comprende: i) Implementación de Suricata; ii) Actualización de las Blacklists; iii) Implementación de Reglas;

Pero antes, implementamos unas configuraciones generales en la interfaz eth0:

Activar la interfaz en modo promiscuo:

```
sudo ip link set dev eth0 promisc on
sudo ip add sh eth0
```

OUTPUT

```
2: eth0: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default
qlen 1000
    link/ether e4:5f:01:53:0b:92 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.16/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
       valid_lft 63792sec preferred_lft 52992sec
    inet6 fe80::2460:7874:bb:e5f5/64 scope link
       valid_lft forever preferred_lft forever
```

Desactivar las funciones LRO/GRO de la interfaz, para que no interfiera en la captura de paquetes[62].

```
sudo ethtool -K eth0 gro off lro off
sudo ethtool -k eth0 | grep receive-offload
```

OUTPUT

```
generic-receive-offload: off
large-receive-offload: off [fixed]
```

[62] https://www.xmodulo.com/install-suricata-intrusion-detection-system-linux.html

4.2.1. Implementación Suricata

En esta sección detallaremos el proceso de instalación, configuración y verificación de Suricata. La versión seleccionada será la última disponible durante la implementación de este proyecto.

Esta es la versión del 12 de julio del 2022: **Suricata 6.0.6**.

NOTA: para la implementación se ha consultado la documentación oficial de Suricata [39] y el artículo de *DigitalOcean* [8].

4.2.1.1. Instalación Suricata

Antes de proceder a la instalación, miraremos qué versión ofrece el repositorio de Debian. Como podemos observar en el output, la versión del repositorio es anterior a la actual (6.0.1-3), por lo tanto, tendremos que compilar Suricata manualmente desde el código fuente.

```
sudo add-apt-repository ppa:oisf/suricata-stable
sudo apt-cache policy suricata
```

OUTPUT

```
suricata:
  Installed: (none)
  Candidate: 1:6.0.1-3
  Version table:
     1:6.0.1-3 500
        500 http://deb.debian.org/debian bullseye/main arm64 Packages
     1:6.0.1-2~bpo10+1 100
        100 http://http.debian.net/debian buster-backports/main arm64 Packages
```

A continuación, listamos las comandas ejecutadas para descargar y compilar la última versión de Suricata. Este proceso es un poco largo, sobre todo en el momento de compilación.

```
sudo apt-get update
wget https://www.openinfosecfoundation.org/download/suricata-6.0.6.tar.gz
tar -xvf suricata-6.0.6.tar.gz
cd suricata-6.0.6
sudo ./configure --prefix=/usr/ --sysconfdir=/usr/local/etc/ --localstatedir=/var/
sudo make install
sudo ldconfig
```

Con el siguiente make crearemos los directorios específicos para el funcionamiento del software. Como podéis observar, los ficheros de configuración se localizarán en el directorio */usr/local/etc/suricata* y los outputs (como son las alertas o eventos) se almacenarán en */var/log/suricata*.

```
sudo make install-conf
```

OUTPUT

```
install -d "/usr/local/etc/suricata/"
install -d "/var/log/suricata/files"
install -d "/var/log/suricata/certs"
install -d "/var/run/"
install -m 770 -d "/var/run/suricata"
```

A continuación, en el directorio */etc/systemd/system* crearemos un servicio para Suricata[63].

/etc/systemd/system/suricata.service

```
[Unit]
Description=Suricata IDS/IPS
After=network.target
Requires=network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/suricata -c /usr/local/etc/suricata/suricata.yaml -i eth0
ExecReload=/bin/kill -HUP $MAINPID


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

Añadimos el directorio local en la variable $PATH.

```
sudo export PATH=$HOME/.local/bin:$PATH
```

Finalmente, comprobamos que el servicio funciona y lo habilitamos.

```
sudo systemctl status suricata
sudo systemctl enable suricata
```

Para concluir con la instalación, desde el repositorio de Debian instalaremos **jq**, una línea de comandos que nos ayudará a manipular ficheros JSON. Esta herramienta es bastante útil en el momento de visualizar los paquetes que analiza Suricata.

```
sudo apt-get install jq
```

[63] https://forum.suricata.io/t/failed-suricata-service-suricata-ids-idp-daemon/2349/7

4.2.1.2. Configuración Suricata

Después de la instalación, iniciaremos con la configuración de Suricata, esta consiste en dos pasos.
El fichero de configuración es: */usr/local/etc/suricata/suricata.yml*.

Primero añadimos en la variable **HOME_NET** el rango de IPs de nuestra red local. De esta manera, Suricata podrá diferenciar entre los paquetes que generan los hosts conectados a la red con los paquetes externos provenientes de Internet.

/usr/local/etc/suricata/suricata.yml

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

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    #HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
    HOME_NET: "[192.168.1.0/24]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

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

Segundo, en la sección **af-packet** añadimos la interfaz de red, en este caso será *eth0*.

/usr/local/etc/suricata/suricata.yml

```
# Linux high speed capture support
af-packet:
  - interface: eth0
```

Con estos dos sencillos pasos tenemos la configuración básica establecida.

4.2.1.3. Validación Suricata

A continuación, verificamos que la configuración de Suricata es correcta y que esta pueda recolectar y analizar la copia de paquetes que recibe de la red local. Además, haremos pruebas con alguna regla para analizar la capacidad de Detección.

A continuación, ejecutamos la siguiente comanda para validar la configuración anterior:

```
sudo suricata -T -c /usr/local/etc/suricata/suricata.yaml -v
```

OUTPUT

```
<Info> - Running suricata under test mode
<Notice> - This is Suricata version 6.0.1 RELEASE running in SYSTEM mode
<Info> - CPUs/cores online: 4
<Info> - fast output device (regular) initialized: fast.log
<Info> - eve-log output device (regular) initialized: eve.json
<Info> - stats output device (regular) initialized: stats.log
<Info> - 1 rule files processed. 1 rules successfully loaded, 0 rules failed
<Info> - Threshold config parsed: 0 rule(s) found
<Info> - 1 signatures processed. 1 are IP-only rules, 0 are inspecting packet payload, 0
inspect application layer, 0 are decoder event only
<Notice> - Configuration provided was successfully loaded. Exiting.
<Info> - cleaning up signature grouping structure... complete
```

Como podemos observar, el output de la comanda es correcto. Por lo tanto, podemos inicializar suricata y dejarla procesar unos minutos mientras recoge y analiza paquetes.

```
sudo systemctl start suricata
```

A continuación, y pasados unos minutos, comprobaremos si Suricata está recolectando y analizando los paquetes de la red local. Para ello localizamos el fichero */var/log/suricata/eve.json* y visualizamos su contenido con la siguiente comanda:

```
sudo tail -f /var/log/suricata/eve.json
```

También podemos analizar un evento en concreto, por ejemplo, desde la máquina 192.168.1.10 hacemos un ping al router (192.168.1.1).

Para identificar el paquete dentro del extenso fichero json que genera suricata, utilizaremos la línea de comandos jq para filtrar los paquetes con el protocolo ICMP.

```
sudo tail -f /var/log/suricata/eve.json | jq 'select(.proto=="ICMP")'
```

El resultado es el siguiente:

OUTPUT


```
{
  "timestamp": "2022-08-06T01:44:46.550139+0200",
  "flow_id": 602329053930358,
  "in_iface": "eth0",
  "event_type": "flow",
  "src_ip": "192.168.1.10",
  "dest_ip": "192.168.1.1",
  "proto": "ICMP",
  "icmp_type": 8,
  "icmp_code": 0,
  "response_icmp_type": 0,
  "response_icmp_code": 0,
  "flow": {
    "pkts_toserver": 8,
    "pkts_toclient": 8,
    "bytes_toserver": 11056,
    "bytes_toclient": 11056,
    "start": "2022-08-06T01:35:40.009078+0200",
    "end": "2022-08-06T01:36:00.038449+0200",
    "age": 20,
    "state": "established",
    "reason": "unknown",
    "alerted": false
  },
  "community_id": "1:LZUz1BTZ4wyOP5bzCbbkszU9OaM="
}  …
```


Como podéis observar en el Output anterior, el paquete ha sido recolectado correctamente. Por lo tanto, podemos afirmar que Suricata está recolectando y analizando los paquetes.

Por otro lado, vemos que este fichero rápidamente aumenta de tamaño, a pesar de poseer 64Gb en esta tarjeta SD. Para solucionarlo, utilizamos la herramienta Logrotate para eliminar logs antiguos.

Para configurarlo, creamos un fichero en */etc/logrotate.d/suricata*[64] con la siguiente configuración.

/etc/logrotate.d/suricata

```
/var/log/suricata/*.log /var/log/suricata/*.json
{
    daily
    maxsize 1G
    rotate 30
    missingok
    notifempty
    nocompress
    create
    sharedscripts
    postrotate
    systemctl restart suricata.service
    endscript
}
```

Finalmente, ejecutamos la siguiente comanda para comprobar que todo está configurado correctamente:

```
sudo logrotate -f /etc/logrotate.conf
```

[64] https://jufajardini.wordpress.com/2021/02/15/suricata-on-your-raspberry-pi/#log-files

Dentro del directorio */var/log/suricata/* también encontramos el archivo **fast.log**, el cual almacena las amenazas encontradas. Este actualmente se encuentra vacío, ya que no hemos especificado ninguna regla aún. Para comprobar su funcionamiento (y en general la detección de alertas por parte de Suricata) vamos a crear un fichero personalizado con alguna regla sencilla. Este fichero se llamará **misreglas.rules** y lo ubicaremos en */usr/local/etc/suricata/rules*.

Dentro de este archivo, vamos a añadir una regla sencilla para detectar PING:

/usr/local/etc/suricata/rules/misreglas.rules

```
alert icmp any any -> any any (msg: "ICMP detectado!";)
```

A continuación, añadiremos el fichero con la regla en la configuración de Suricata */usr/local/etc/suricata/suricata.yml*.

/usr/local/etc/suricata/suricata.yml

```
## Configure Suricata to load Suricata-Update managed rules.
##

default-rule-path: /usr/local/etc/suricata/rules

rule-files:
  - misreglas.rules
```

Reiniciamos el servicio y hacemos ping desde la máquina 192.168.1.10 a la máquina 192.168.1.16.

```
sudo systemctl restart suricata
```

Podemos observar entonces las alertas tanto en el fichero **fast.log** como en el fichero **eve.log**.

/var/log/suricata/fast.log

```
08/06/2022-02:06:18.152441  [**] [1:0:0] ICMP detectado! [**] [Classification: (null)]
[Priority: 3] {ICMP} 192.168.1.10:8 -> 192.168.1.16:0
08/06/2022-02:06:18.152513  [**] [1:0:0] ICMP detectado! [**] [Classification: (null)]
[Priority: 3] {ICMP} 192.168.1.16:0 -> 192.168.1.10:0
```

```
sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'
```

/var/log/suricata/eve.log


```
{
  "timestamp": "2022-08-06T02:06:18.152441+0200",
  "flow_id": 552047992263545,
  "in_iface": "eth0",
  "event_type": "alert",
  "src_ip": "192.168.1.10",
  "src_port": 0,
  "dest_ip": "192.168.1.16",
  "dest_port": 0,
  "proto": "ICMP",
  "icmp_type": 8,
  "icmp_code": 0,
  "community_id": "1:JCTZnvojrpId+9u2ClpoNmz8tl4=",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 0,
    "rev": 0,
    "signature": "ICMP detectado!",
    "category": "",
    "severity": 3
```


Con el último test, hemos comprobado que Suricata está funcional tanto en la recolección y análisis de paquetes como en la detección de alertas siguiendo un patrón definido por una regla o signatura.

Por último, estudiamos el fichero **stats.log** para comprobar cuantos paquetes está capturando y si existe perdida de paquetes.

```
sudo cat stats.log.1 | grep 'capture.kernel'
```

OUTPUT

```
capture.kernel_packets                    | Total                     | 8709926
capture.kernel_drops                      | Total                     | 51396
```

Podemos observar en el Output de la comanda que se están perdiendo alrededor de 51396 de un total de 8709926. Esta cantidad es muy pequeña, pero se puede evitar. Para ello accedemos al fichero de configuración de Suricata (suricata.yml) y aumentamos el valor de la variable **ring-size** al doble con un valor de 100000.

/var/log/suricata/fast.log

```
# Ring size will be computed with respect to "max-pending-packets" and number
# of threads. You can set manually the ring size in number of packets by setting
# the following value. If you are using flow "cluster-type" and have really network
# intensive single-flow you may want to set the "ring-size" independently of the number
# of threads:
  ring-size: 100000
```

4.2.2. Actualización Blacklists

Es importante mantener las reglas de las blacklists actualizadas de forma regular, para tener el sistema siempre alerta con las últimas amenazas. Además, permite eliminar las signaturas que ya no suponen ningún riesgo.

En este caso, el periodo de actualización lo consultamos con la documentación de cada blacklists.

En general:

- Los proyectos Abuse actualizan cada 5 minutos, de hecho, lo recomiendan.
- Los repositorios de Ming Di Leom se actualizan 2 veces al día, cada 12 horas.
- Finalmente, el proyecto ET se actualiza de forma diaria, cada 24 horas.

Vemos entonces que tendremos que crear tres scripts: *update_abuse.sh*, *update_et.sh* y *update_p_filer.sh*. Estos scripts descargarán el contenido de las páginas oficiales de cada repositorio y actualizando así la versión local. Además, serán creados con el usuario root y les daremos permisos de ejecución.

Aquí tenéis un ejemplo del script *update_p_filer.sh*

/usr/local/etc/rules/update_p_filer.sh

```
#!/bin/bash

path='/usr/local/etc/rules'

# Phishing Filter:
curl -L "https://malware-filter.gitlab.io/malware-filter/phishing-filter-suricata.rules" -o
"$path/phishing-filter-suricata.rules"

# PUP Filter:
curl -L "https://malware-filter.gitlab.io/malware-filter/pup-filter-suricata.rules" -o
"$path/pup-filter-suricata.rules"
```

1. phishing-filter-suricata.rules
2. urlhaus_suricata.rules
3. sslblacklist_tls_cert.rules
4. sslipblacklist.rules
5. emerging-compromised.suricata.rules
6. emerging-drop.suricata.rules
7. emerging-phishing.rules
8. pup-filter-suricata.rules
9. feodotracker.rules

Incluimos el conjunto de reglas anterior en el fichero de configuración de Suricata:

/usr/local/etc/suricata/suricata.yml

```
##
## Configure Suricata to load Suricata-Update managed rules.
##

default-rule-path: /usr/local/etc/rules

rule-files:
  - misreglas.rules
  - phishing-filter-suricata.rules
  - urlhaus_suricata.rules
  - sslblacklist_tls_cert.rules
  - sslipblacklist.rules
  - emerging-compromised.suricata.rules
  - emerging-drop.suricata.rules
  - emerging-phishing.rules
  - pup-filter-suricata.rules
  - feodotracker.rules
```

Finalmente, para la actualización periódica, utilizamos la herramienta Crontab, la cual ejecutará cada script en período indicado:

crontab -e

```
# m h  dom mon dow   command
  */5 * * * *       /usr/local/etc/update_abuse.sh
  0 */12 * * *      /usr/local/etc/update_p_filer.sh
  0 0 23 * *        /usr/local/etc/update_et.sh
```

4.2.3. Reglas Suricata Detección de Enlaces Maliciosos

Finalmente, procederemos a formular las reglas diseñadas anteriormente en el lenguaje empleado por Suricata. Estas reglas serán del tipo "*alert*" y las dividiremos en dos tipos: peticiones DNS y paquetes HTTP. Además, emplearemos expresiones regulares para poder detectar las características de estos enlaces o dominios.

/usr/local/etc/rules/misreglas.rules

```
# Regla 0: HTTP
alert http $HOME_NET any -> $EXTERNAL_NET 80 (msg:"REGLA 0: Protocolo HTTP sin TLS.";
classtype:sospechoso; sid:123;)

# Regla 1: Host es una IP:
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 1: Host es IP. (HTTP) [Malware]"; http.host;
pcre:"/([0-9]*[.]){3}/"; classtype:sospechoso; sid:1;)

# Regla 2: Contiene Puerto:
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 2: Contiene Puerto  [Posible Malware]";
http.host.raw; pcre:"/.+[:]+[0-9]{1,5}$/"; classtype:sospechoso; sid:2;)

# Regla 3: Longitud Host +30 caracteres:
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 3: Longitud del host +30 caracteres.
[Posible Phishing]"; http.host.raw; pcre:"/.{30,}/";  classtype:sospechoso; sid:3;)

# Regla 4: Longitud del Path = 8:
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 4: Longitud del path = 8 caracteres.
[Posible Malware]"; content:"/"; http.uri; urilen:8;  classtype:sospechoso; sid:4;)

# Regla 5: Longitud del Path = 3:
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 5: Longitud del path = 3 caracteres.
[Posible Malware]"; content:"/"; http.uri; urilen:3;  classtype:sospechoso; sid:5;)

# Regla 6: Longitud del Path = 4:
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 5: Longitud del path = 4 caracteres.
[Posible Malware]"; content:"/"; http.uri; urilen:4;  classtype:sospechoso; sid:6;)

# Regla 7: Enlace de +9 digitos:
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 7: Enlace +9 digitos. [Malware]";
content:"/"; http.uri; pcre:"/([^\d]*\d[^\d]*\d[^\d]*\d){3}/"; classtype:sospechoso; sid:7;)

# Regla 8: Enlace de +15 digitos:
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 8: Enlace +15 digitos. [Phishing]";
content:"/"; http.uri; pcre:"/([^\d]*\d[^\d]*\d[^\d]*\d){5}/"; classtype:sospechoso; sid:8;)

# Regla 9: Enlace de 4 a 6 puntos:
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 9: Enlace [4-6] puntos. [Malware]";
content:"/"; http.uri; pcre:"(/(.*[.].*){4,6}/"; classtype:sospechoso; sid:9;)

# Regla 10: Dominio/Host + 3 puntos:
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 10: Dominio +3 puntos.(HTTP) [Phishing]";
http.host; pcre:"/([a-z0-9A-Z]*[.]){3,}/"; classtype:sospechoso; sid:10;)
alert dns $HOME_NET any -> any any (msg:"REGLA 10: Dominio +3 puntos. (DNS) [Phishing]"; dns.query;
pcre:"(/([a-z0-9A-Z]*[.]){3,}/"; classtype:sospechoso; sid:100;)

# Regla 11: Simbolo "@"
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 11: Simbolo = @ (HTTP)"; content:"@";
http_uri; classtype:sospechoso; sid:11;)
# Regla 12: TLD = .81
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 12: TLD=.81 (HTTP)"; http.host;
content:".81"; endswith; classtype:sospechoso; sid:12;)
alert dns $HOME_NET any -> any any (msg:"REGLA 12: TLD=.81 (DNS)"; dns_query; content:".81"; endswith;
classtype:sospechoso; sid:122;)

# Regla 13: TLD = .234
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 13: TLD=.234 (HTTP)"; http.host;
content:".234"; endswith; classtype:sospechoso; sid:13;)
alert dns $HOME_NET any -> any any (msg:"REGLA 13: TLD=.234 (DNS)"; dns_query; content:".234";
endswith; classtype:sospechoso; sid:133;)

# Regla 14: TLD = .194
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 14: TLD=.194 (HTTP)"; http.host;
content:".194"; endswith; classtype:sospechoso; sid:14;)
alert dns $HOME_NET any -> any any (msg:"REGLA 14: TLD=.194 (DNS)"; dns_query; content:".194";
endswith; classtype:sospechoso; sid:144;)
```

```
# Regla 15: TLD = .119
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 15: TLD=.119 (HTTP)"; http.host;
content:".119"; endswith; classtype:sospechoso; sid:15;)
alert dns $HOME_NET any -> any any (msg:"REGLA 15: TLD=.119 (DNS)"; dns_query; content:".119";
endswith; classtype:sospechoso; sid:155;)

# Regla 16: TLD = .xyz
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 16: TLD=.xyz (HTTP) [Posible Phishing]";
http.host; content:".xyz"; endswith; classtype:sospechoso; sid:16;)
alert dns $HOME_NET any -> any any (msg:"REGLA 16: TLD=.xyz (DNS) [Posible Phishing]"; dns_query;
content:".xyz"; endswith; classtype:sospechoso; sid:166;)

# Regla 17: TLD = .tk
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 17: TLD=.tk (HTTP) [Posible Phishing]";
http.host; content:".tk"; endswith; classtype:sospechoso; sid:17;)
alert dns $HOME_NET any -> any any (msg:"REGLA 17: TLD=.tk (DNS) [Posible Phishing]"; dns_query;
content:".tk"; endswith; classtype:sospechoso; sid:177;)

# Regla 18: TLD = .info
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 18: TLD=.info (HTTP) [Posible Phishing]";
http.host; content:".info"; classtype:sospechoso; sid:18;)
alert dns $HOME_NET any -> any any (msg:"REGLA 18: TLD=.info (DNS) [Posible Phishing]"; dns_query;
content:".info"; endswith; classtype:sospechoso; sid:188;)

# Regla 19: TLD = .club
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 19: TLD=.club (HTTP) [Posible Phishing]";
http.host; content:"club"; endswith; classtype:sospechoso; sid:19;)
alert dns $HOME_NET any -> any any (msg:"REGLA 19: TLD=.club (DNS) [Posible Phishing]"; dns_query;
content:"club"; endswith; classtype:sospechoso; sid:199;)

# Regla 20: Keyword = admin
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 20: Keyword = admin(HTTP) [Posible
Phishing]"; content:"admin"; http_uri; classtype:sospechoso; sid:20;)

# Regla 21: Keyword = client
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"REGLA 21: Keyword = client(HTTP) [Posible
Phishing]"; content:"client"; http_uri; classtype:sospechoso; sid:21;)
```

4.2.3.1 Verificación Reglas de Detección de Enlaces Maliciosos

Después de implementar las reglas, procederemos a comprobar que detectan correctamente las signaturas. Para ello, activaremos todas las alertas para las reglas.

Para no interferir con el test, desactivamos el port mirroring y ejecutamos las comandas desde la propia máquina NIDS. Además, para el testeo de algunas reglas, hemos cambiado las cabeceras para detectar peticiones hacia nuestra red interna:

$HOME_NET any -> **$EXTERNAL_NET** ----------> $HOME_NET any -> **any**

En la **Tabla 18** podéis observar las pruebas efectuadas junto a sus resultados.

Tabla 18. Resultados set de Reglas de detección. Elaboración Propia.

#	Regla		Test	¿Ok?
1	Regla 0	HTTP sin SSL/TLS	*curl -Ikv --request GET 'http://www.google.com/"*	Ok
2			*curl -Ikv --request GET 'http://142.251.215.238/'*	Ok
3	Regla 1	Host es IP	*curl -Ikv --request GET 'http://176.32.103.205'*	Ok
4			*curl -Ikv --request GET 'http://193.144.16.62'*	Ok
5	Regla 2	Contiene Puerto	*curl -Ikv --request GET '192.168.1.17:5601/'*	Ok
6			*curl -Ikv --request GET 'google.com:80/'*	KO
7	Regla 3	Longitud Host +30 caracteres	*curl -Ikv --request GET 'http://www.lalalalalalalalalalalalalalalala.com/'*	Ok
8	Regla 4	Longitud Path = 8 caracteres	*curl -Ikv --request GET 'http://www.google.com/a2a1b34'*	Ok
9	Regla 5	Longitud Path = 3 caracteres	*curl -Ikv --request GET 'http://www.google.com/a2'*	Ok
10	Regla 6	Longitud Path = 4 caracteres	*curl -Ikv --request GET 'http://www.google.com/a1c'*	Ok
11	Regla 7	Dígitos Path+9	*curl -Ikv --request GET 'http://www.google.com/123456789'*	Ok
12			*curl -Ikv --request GET 'http://www.google.com/a1b2sa34s567s89'*	Ok
13	Regla 8	Dígitos Path +15	*curl -Ikv --request GET 'http://www.google.com/123456789101112'*	Ok
14			*curl -Ikv --request GET 'http://www.google.com/abc12s3456789a1a2a3a4a5a6a9'*	Ok
15	Regla 9	Puntos Path [4-6]	*curl -Ikv --request GET 'http://www.google.com/search?q=....'*	Ok
16			*curl -Ikv --request GET 'http://www.google.com/search?q=a.1.ds..d'*	Ok
17			*curl -Ikv --request GET 'http://www.google.com/search?q=a.1.ds..d.as.'*	Ok
18	Regla 10	Puntos Dominio +3	*curl -Ikv --request GET 'http://www.myaccount.google.com'*	Ok
19			*curl -Ikv --request GET 'http://192.168.1.17:5601/app'*	Ok
20	Regla 11	Símbolo "@"	*curl -Ikv --request GET 'http://www.google.com/search?=@'*	Ok
21			*curl -Ikv --request GET 'http://www.google.com/search?user=@test@@'*	Ok
22	Regla 12	TDL .81	*nslookup 8.8.8.8.81*	Ok
23	Regla 13	TDL .234	*nslookup 8.8.8.8.234*	Ok
24	Regla 14	TDL .194	*nslookup 8.8.8.8.194*	Ok
25	Regla 15	TDL .119	*nslookup 8.8.8.8.119*	Ok
26	Regla 16	TDL .xyz	*curl -Ikv --request GET 'http://gen.xyz/'*	Ok
27	Regla 17	TDL .tk	*curl -Ikv --request GET 'http://www.google.tk/'*	Ok
28	Regla 18	TDL .info	*curl -Ikv --request GET 'http://www.spain.info/'*	Ok
29	Regla 19	TDL .club	*curl -Ikv --request GET 'http://www.club.club/'*	Ok
30	Regla 20	Keyword = admin	*curl http://google.com/search=?admin*	Ok
31	Regla 21	Keyword = client	*curl http://google.com/search=?client*	Ok

Error regla 3: Contiene Puerto

En la **Tabla 18** observamos que la regla 3 ha fallado en el test número 4. Ha podido detectar el puerto 5600, pero no el 80 de Google. Si nos fijamos en la salida de la comanda curl, en la sección "Host" vemos que el primer caso ha incluido el puerto, por lo tanto, es detectado por Suricata, en cambio, en el segundo no.

```
root@NIDS:/usr/local/etc/rules# curl -Ikv --request GET 'http://192.168.1.17:5601/'
*   Trying 192.168.1.17:5601...
* Connected to 192.168.1.17 (192.168.1.17) port 5601 (#0)
> GET / HTTP/1.1
> Host: 192.168.1.17:5601
> User-Agent: curl/7.74.0
> Accept: */*
>
```

```
root@NIDS:/usr/local/etc/rules# curl -Ikv --request GET 'google.com:80/'
*   Trying 142.250.200.78:80...
* Connected to google.com (142.250.200.78) port 80 (#0)
> GET / HTTP/1.1
> Host: google.com
> User-Agent: curl/7.74.0
> Accept: */*
>
```

4.3. SIEM

A continuación, implementaremos y evaluaremos el Sistema de Gestión de Información y Eventos de Seguridad o SIEM. Esta sección comprende los siguientes apartados: i) Implementación de Elasticsearch; ii) Implementación de Kibana; iii) Implementación de Filebeat; iv) Implementación de Metricbeat; v) Verificación Flujo de datos; vi) Implementación de Tableros; vii) Verificación acceso al SIEM desde la red;

NOTA: para la implementación del SIEM, se ha consultado las páginas Oficiales de cada herramienta además de *DigitalOcean* [R].

4.3.1. Implementación Elasticseach

A continuación, implementamos y configuramos Elasticsearch en la Raspberry Pi destinada al SIEM.

Primeramente, instalamos el Elasticsearch desde el repositorio de Debian junto a algunos paquetes necesarios.

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

A continuación, modificamos al fichero de configuración de Elasticsearch situado en el directorio */etc/elasticsearch*. Este le agregamos la IP de nuestra red, lo configuramos como nodo único y aceptamos algunas medidas de seguridad.

/etc/elasticsearch/elasticsearch.yml

```
# ---------------------------------- Network -----------------------------------
#
# By default Elasticsearch is only accessible on localhost. Set a different
# address here to expose this node on the network:
#
network.host: 192.168.1.17
#
# By default Elasticsearch listens for HTTP traffic on the first free port it
# finds starting at 9200. Set a specific HTTP port here:
#
http.port: 9200
#
# --------------------------------- Discovery ----------------------------------
#
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
discovery.seed_hosts: ["192.168.1.17"]

# Enable security features
xpack.security.enabled: true
```

Por otro lado, agregamos reglas al Firewall para asegurar que Elasticsearch es accesible desde la red privada.

```
sudo ufw allow in on eth0
sudo ufw allow out on eth0
sudo systemctl start elasticsearch.service
```

Finalmente, iniciamos el servicio:

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

4.3.1.1. Configuración de Credenciales

Este apartado es importante, ya que configuramos los datos de acceso al SIEM.

Mediante la comanda *elasticsearch-setup-passwords* interactive, situada en el directorio */usr/share/elasticsearch/bin*, indicamos las contraseñas:

```
cd /usr/share/elasticsearch/bin
sudo ./elasticsearch-setup-passwords interactive
```

Para comprobar que funciona y se encuentra disponible, accedemos a Elastic desde cualquier dispositivo conectado a la red **Figura 53**. Hemos verificado que al acceder nos ha solicitado las contraseñas anteriormente establecidas.

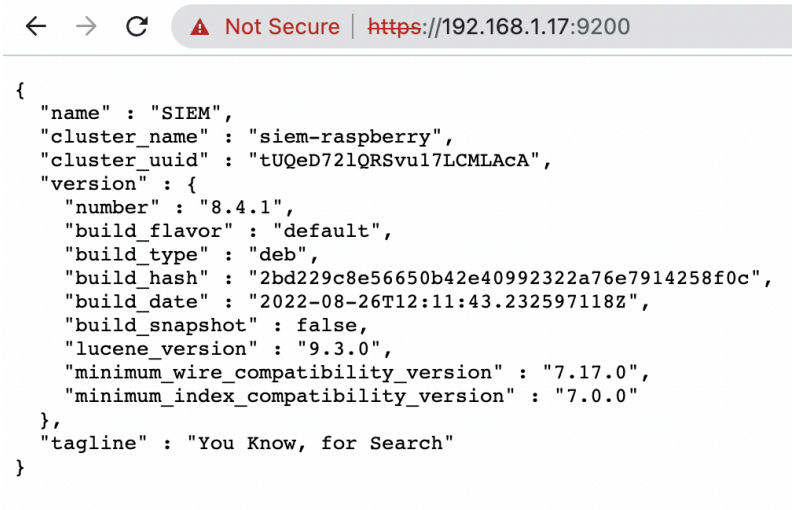


Figura 53. Acceso a Elasticsearch. Elaboración Propia.

Finalmente, generaremos los certificados autosignados con la comanda *elasticsearch-certgen*, también situada en el directorio */usr/share/elasticsearch/bin*

4.3.2. Implementación Kibana

A continuación, mostramos el proceso de implementación de Kibana.

Primeramente, instalamos Kibana:

```
sudo apt update
sudo apt install kibana
```

A continuación, generaremos algunas claves de cifrado que utiliza Kibana para almacenar datos de sesión, paneles, cookies, entre otros.

```
cd /usr/share/kibana/bin/
sudo ./kibana-encryption-keys generate -q
```

La anterior comanda nos generará el siguiente Output que debemos almacenar:

```
xpack.encryptedSavedObjects.encryptionKey: f84fd1ad91ccda0c5c563b69ec3c0502
xpack.reporting.encryptionKey: 216cf71f081c38b9de6f078e699efa13
xpack.security.encryptionKey: 216a0cfcef32ae55a68e38e13b5379cf
```

A continuación, iniciamos con el proceso de configuración. Para ello accedemos al fichero *kibana.yml* situado en el directorio */etc/kibana*. En este fichero indicamos nuestra IP local, el certificado y la IP de Elasticsearch, las credenciales de acceso a Elastic y la claves de cifrado generadas anteriormente.

/etc/Kibana/Kibana.yml

```
# ==================== System: Kibana Server ====================
# Kibana is served by a back end server. This setting specifies the port to use.
server.port: 5601

# Specifies the address to which the Kibana server will bind. IP addresses and host names are
both valid values.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: "192.168.1.17"
# Specifies the public URL at which Kibana is available for end users. If
# `server.basePath` is configured this URL should end with the same basePath.
server.publicBaseUrl: http://192.168.1.17:5601

…

# ==================== System: Elasticsearch ====================
# The URLs of the Elasticsearch instances to use for all your queries.
elasticsearch.hosts: ["https://192.168.1.17:9200"]

# If your Elasticsearch is protected with basic authentication, these settings provide
# the username and password that the Kibana server uses to perform maintenance on the Kibana
# index at startup. Your Kibana users still need to authenticate with Elasticsearch, which
# is proxied through the Kibana server.
elasticsearch.username: "kibana_system"
elasticsearch.password: "********"
elasticsearch.ssl.certificateAuthorities: [ "/etc/kibana/certs/http_ca.crt" ]

pack.encryptedSavedObjects.encryptionKey: f84fd1ad91ccda0c5c563b69ec3c0502
xpack.reporting.encryptionKey: 216cf71f081c38b9de6f078e699efa13
xpack.security.encryptionKey: 216a0cfcef32ae55a68e38e13b5379cf
```

Finalmente, iniciamos el servicio y verificamos que se ejecuta correctamente. Para ello accedemos a la dirección http://192.168.1.17:5601/. Como observamos en la **Figura 54**, efectivamente, no pide las credenciales de acceso configuradas anteriormente.

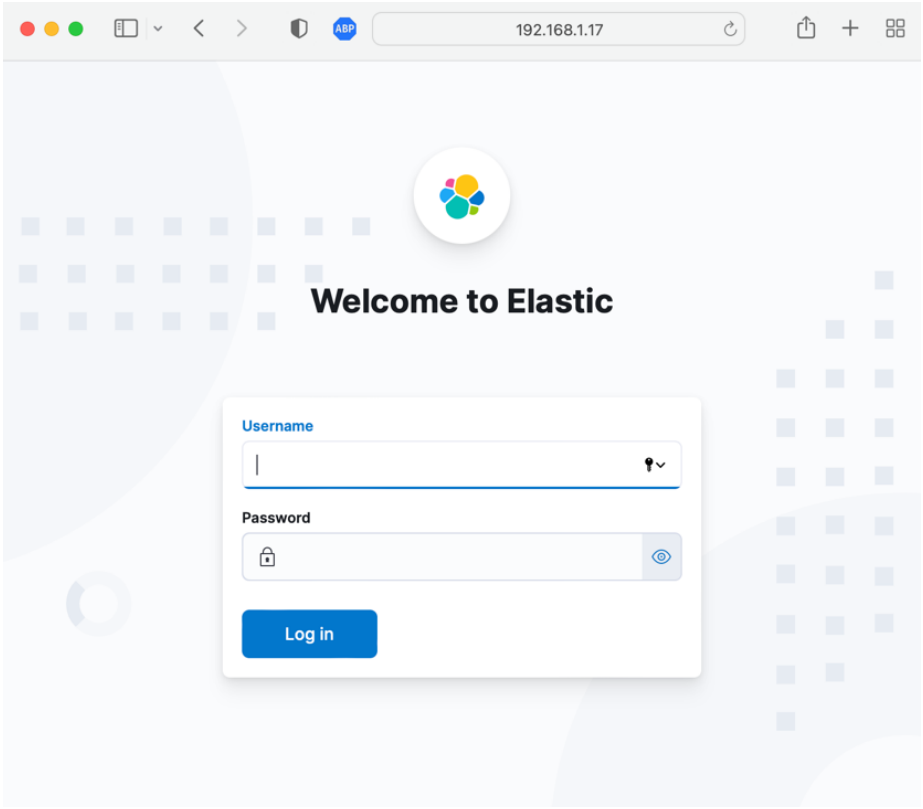


Figura 54. Inicio de sesión Kibana. Elaboración Propia.

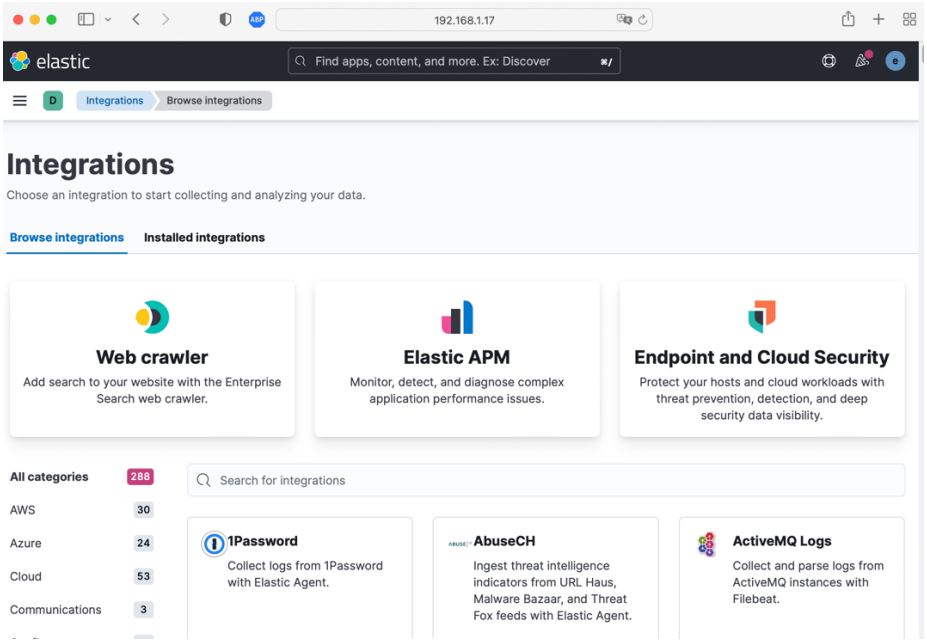


Figura 55. Panel Kibana. Elaboración Propia.

4.3.3. Implementación Filebeat

Después de instalar Kibana y Elastic, es momento de configurar el Filebeat para enviar logs del NIDS al SIEM, de esta forma podremos visualizar los datos. Esta instalación se realizará desde la Raspberry pi NIDS.

Iniciaremos la instalación del paquete con las siguientes comandas:

```
sudo apt update
sudo apt install filebeat
sudo systemctl status filebeat
sudo systemctl enable filebeat
sudo systemctl stop filebeat
```

Una vez verificado que está instalado correctamente, iniciaremos la configuración del servicio a través del fichero *filebeat.yml*. Este se encuentra en el directorio */etc/filebeat*.

En este caso tendremos que incluir las configuraciones de Elastic y Kibana implementadas anteriormente en el SIEM.

Configuración sección Elasticsearch:

/etc/filebeat/filebea.yml

```
# ---------------------------- Elasticsearch Output ----------------------------
output.elasticsearch:

  # Array of hosts to connect to.
  hosts: ["192.168.1.17:9200"]
  ssl.certificate_authorities: ["/etc/filebeat/certs/http_ca.crt"]

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

  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  username: "elastic"
  password: "*********"
```

Configuración sección Kibana:

/etc/filebeat/filebea.yml

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

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

  # Kibana Host
  # Scheme and port can be left out and will be set to the default (http and 5601)
  # In case you specify and additional path, the scheme is required: http://localhost:5601/path
  # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
  host: "192.168.1.17:5601"
```

Una vez terminada la configuración, debemos indicar la localización de los logs de suricata que vamos a transferir al SIEM. Para ello accedemos al módulo *suricata.yml* situado en el directorio */etc/filebeat/modules.d*. Como podéis observar, transferimos el fichero *eve.json* de Suricata, este contiene todos los eventos y alertas.

/etc/filebeat/modules.d/suricata.yml

```
# Module: suricata
# Docs: https://www.elastic.co/guide/en/beats/filebeat/8.3/filebeat-module-suricata.html

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

    # Set custom paths for the log files. If left empty,
    # Filebeat will choose the paths depending on your OS.
    var.paths: [ "/var/log/suricata/eve.json" ]
```

Después de configurar el módulo, lo activamos con la siguiente comanda:

```
sudo filebeat modules enable suricata
```

Una vez finalizada toda la configuración,ejecutamos la comanda test para verificar que todo está correcto y se puede alcanzar el SIEM.

```
sudo filebeat test config
sudo filebeat test output
```

OUTPUT

```
Config OK

elasticsearch: https://192.168.1.17:9200...
  parse url... OK
  connection...
    parse host... OK
    dns lookup... OK
    addresses: 192.168.1.17
    dial up... OK
  TLS...
    security: server's certificate chain verification is enabled
    handshake... OK
    TLS version: TLSv1.3
    dial up... OK
  talk to server... OK
  version: 8.3.3
```

Finalmente, activamos filebeat y reiniciamos el servicio:

```
sudo filebeat setup
```

OUTPUT

```
Overwriting ILM policy is disabled. Set `setup.ilm.overwrite: true` for enabling.

Index setup finished.
Loading dashboards (Kibana must be running and reachable)
Loaded dashboards
Loaded Ingest pipelines
```

```
sudo systemctl start filebeat
sudo systemctl status filebeat
```

4.3.4. Implementación Metricbeat

Por último, implementaremos la herramienta Metricbeat, esta debe ser instalada en ambos sistemas SIEM y NIDS.

```
sudo apt update
sudo apt install metricbeat
sudo systemctl status metricbeat
sudo systemctl enable metricbeat
sudo systemctl stop metricbeat
```

Al igual que Filebeat, le indicamos las configuraciones de Elasticsearch y Kibana:

/etc/metricbeat/metricbeat.yml

```
# ---------------------------- Elasticsearch Output ----------------------------
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["192.168.1.17:9200"]

  ssl.certificate_authorities: ["/etc/metricbeat/certs/http_ca.crt"]
  # Protocol - either `http` (default) or `https`.
  protocol: "https"

  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  username: "elastic"
  password: "*****"
```

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

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

  # Kibana Host
  # Scheme and port can be left out and will be set to the default (http and 5601)
  # In case you specify and additional path, the scheme is required: http://localhost:5601/path
  # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
  host: "192.168.1.17:5601"
```

Además, le indicaremos un periodo de actualización de 10 segundos:

/etc/metricbeat/metricbeat.yml

```
# Period on which files under path should be checked for changes
  reload.period: 10s
```

Después de terminar de configurar el fichero anterior, seleccionamos los parámetros del sistema que deseamos evaluar. Para ello accedemos al fichero *system.yml*, situado en el directorio */etc/metricbeat/modules.d*.

A continuación, podéis ver los parámetros seleccionados:

/etc/ metricbeat /modules.d/system.yml

```
- module: system
  period: 10s
  metricsets:
    - cpu
    - load
    - memory
    - network
    - process
    - process_summary
    - socket_summary
    #- entropy
    - core
    - diskio
    #- socket
    #- service
    #- users
```

Al terminar de configurar el módulo sistema lo habilitamos con al comanda:

```
sudo metricbeat modules enable system
```

OUTPUT

```
Config OK

elasticsearch: https://192.168.1.17:9200...
  parse url... OK
  connection...
    parse host... OK
    dns lookup... OK
    addresses: 192.168.1.17
    dial up... OK
  TLS...
    security: server's certificate chain verification is enabled
    handshake... OK
    TLS version: TLSv1.3
    dial up... OK
  talk to server... OK
  version: 8.3.3
```

Al igual que Filebeat, comprobamos que la configuración esté correcta con las comandas.

```
sudo metricbeat test config
sudo metricbeat test output
```

Y finalmente activamos el servicio:

```
sudo metricbeat setup
sudo systemctl start metricbeat
sudo systemctl status metricbeat
```

4.3.5. Verificación Flujo de Datos

Finalmente, después de implementar todas las herramientas necesarias, procedemos a comprobar si el flujo de datos ocurre de la manera esperada, es decir, que los logs del NIDS se envían correctamente al SIEM. Para realizar esta verificación, basta con comprobar que desde el escritorio de Kibana se han creado dos índices de datos: uno para Filebeat que contiene datos de Suricata y otro de Metricbeat con las métricas del sistema tanto del NIDS como del SIEM.

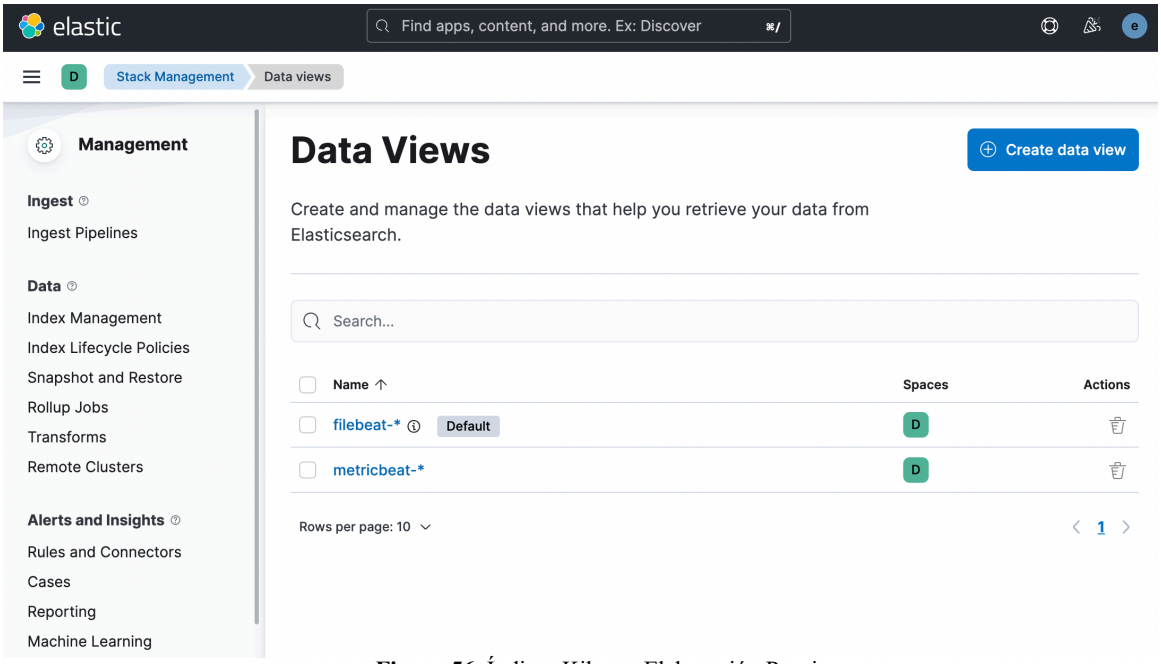


Figura 56. Índices Kibana. Elaboración Propia.

Es más, podemos acceder a la sección *Discover* de Kibana y filtrar los datos recibidos. En la **Figura 57** se ha filtrado los datos provenientes de Fileebat según el tipo de eventos, en este caso se ha seleccionado alerta. Además, mostramos solamente unos campos en específico como es la IP origen, IP destino y la signatura de la alerta.

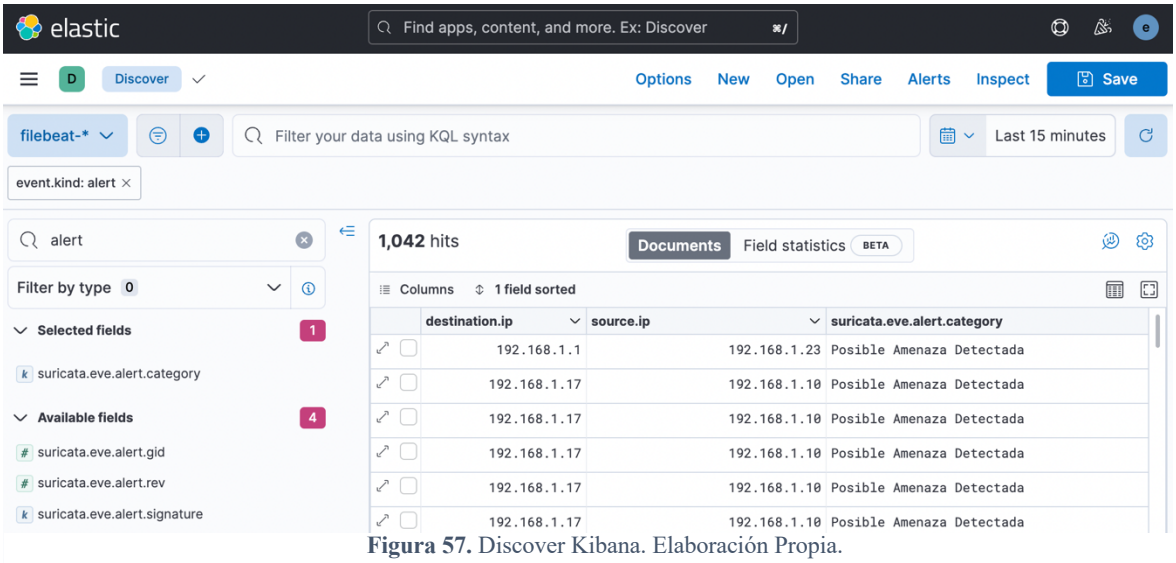


Figura 57. Discover Kibana. Elaboración Propia.

4.3.6. Implementación Tableros

Finalmente, después de instalar las herramientas y comprobar su correcto funcionamiento. Procederemos a diseñar los paneles en Kibana.

En general, en el diseño hemos especificado siete paneles: dos para la visualización de métricas, dos para la visualización de eventos de la red, dos para la visualización de alertas y un panel como centro de unión de todos los tableros de visualización.

Por otro lado, a pesar de que en Kibana podemos encontrar tableros por defecto para Suricata y Metricbeat, estos no poseen toda la información que necesitamos para diseñar el SIEM (Diseño Paneles). Por lo tanto, aprovecharemos algunos elementos que aparecen en estos tableros y diseñaremos unos propios para conseguir el resultado deseado.

4.3.6.1. Elementos comunes

Dentro de cada panel existen elementos comunes que comparten todos los tableros. Estos están relacionados con la movilidad dentro del sistema o simplemente información del panel que se está visualizando.

En la **Figura 58,** podéis observar estos tres elementos:

- Cabecera con información básica sobre el prototipo.
- Un menú de navegación principal.
- Una sección que describe el panel que se está visualizando y enlaces hacia las dos opciones de visualización (Resumen-Avanzado).

El panel de la **Figura 58,** es el de métricas del sistema, modo resumido.

Figura 58. Cabecera SIEM. Elaboración Propia.

NOTA: En los siguientes apartados, se omiten las cabeceras en las figuras adjuntas por motivos espaciales del documento (no significa que estén eliminadas).

4.3.6.2. Tablero General

En la **Figura 58**,podéis observar el panel general del SIEM, este sirve como punto de unión hacia el resto de los paneles. Básicamente se trata de un panel textual con enlaces hacia los tableros correspondientes.

Figura 59. Panel Principal SIEM. Elaboración Propia.

4.3.6.3. Tablero Métricas

El tablero de métricas ha sido uno de los más sencillos de implementar, dado que la mayoría de los elementos gráficos que hemos utilizado los ofrecía el panel por defecto de Metricbeats. Por lo tanto, hemos redistribuido estos elementos y organizado según el modo de visualización.

4.3.6.3.1. Métricas: Vista Resumen

La vista resumen está compuesta por estos seis elementos **Figura 60**.Esta vista engloba información tanto del NIDS como del SIEM. En general, podemos observar el uso de CPU, de la RAM y del disco.

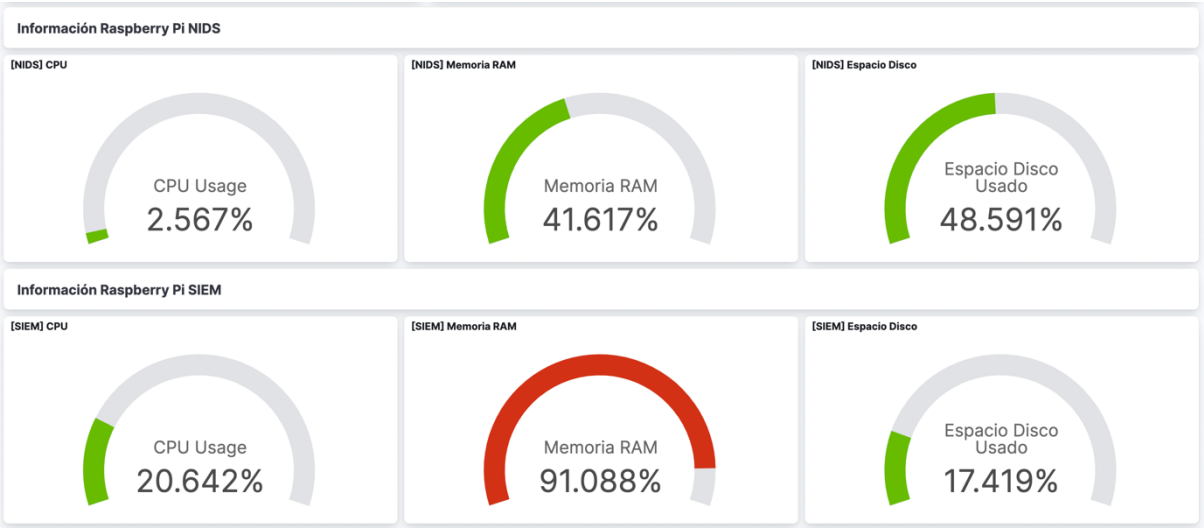


Figura 60.Métricas Vista Resumen. Elaboración Propia.

4.3.6.3.1. Métricas: Vista Avanzada

En cuanto a las métricas avanzadas, para no duplicar tableros, hemos creado el siguiente filtro **Figura 61** para poder pasar de un sistema a otro.

Figura 61. Tablero Métricas Vista avanzada. Elaboración Propia.

En este último tablero, la información se organiza en 3 secciones:

- *Métricas CPU*: % de uso, numero de procesos, Top procesos que consumen más CPU y promedio de uso de la CPU a lo largo del día.
- *Métricas RAM*: % memoria RAM y memoria Swap, top procesos que consumen más RAM y promedio de uso de la memoria a lo largo del día.
- *Métricas Disco*: cantidad disponible y tota de espacio, cantidad de lecturas y escrituras en el disco.

Métricas CPU

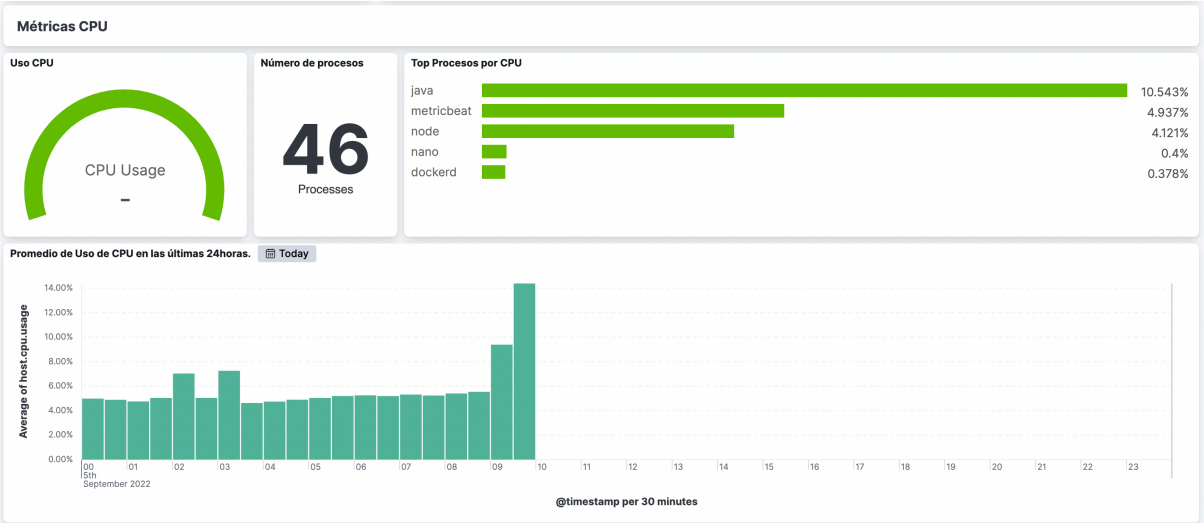


Figura 62. Métricas CPU. Elaboración Propia.

Métricas Memoria RAM

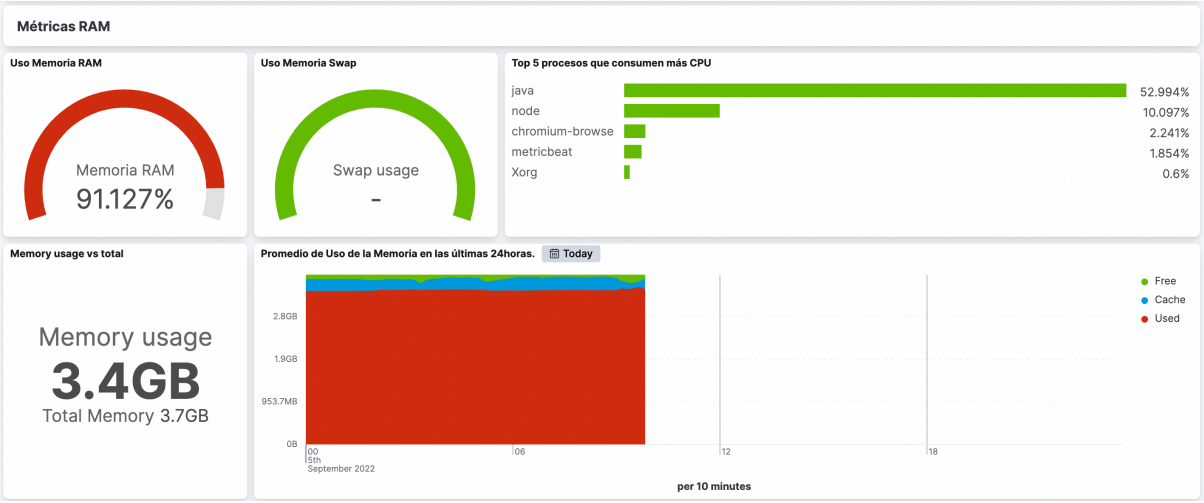


Figura 63. Métricas RAM. Elabroación Propia.

Métricas Disco

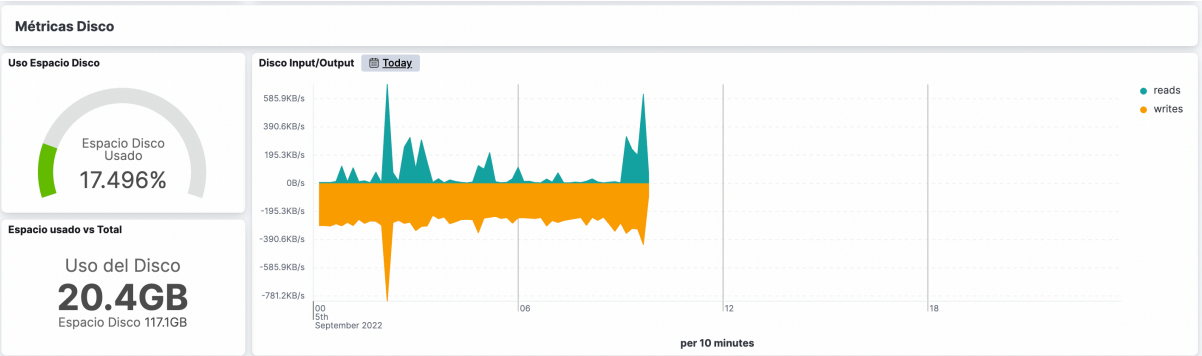


Figura 64. Métricas Disco. Elaboración Propia.

4.3.6.4. Tablero Eventos

Este tablero, a diferencia del anterior, la mayoría de los elementos han sido implementados por nosotros.

4.3.6.4.1. Eventos: Vista Resumen

La vista resumen se compone de dos secciones: información general de los protocolos y paquetes e información de los hosts conectados a la red local.

Información General

La vista general muestra los siguientes paneles **Figura 65**.Los elementos TOP protocolos y Top Protocolos de Transporte se han adquirido del panel por defecto de Suricata. El resto de los elementos han sido creados por nosotros, los filtros han sido los siguientes:

- Paquetes Capturados: se han contabilizado todas las entradas de suricata con el filtro *event.module:suricata*
- Cantidad de Paquetes: se ha utilizado el parámetro kernel_drops del fichero stats.log de Suricata: *suricata.eve.stats.capture.kernel_drops*
- HTTP vs HTTPS: se ha delimitado según el puerto destino: *destination.port : 80* o *433*.

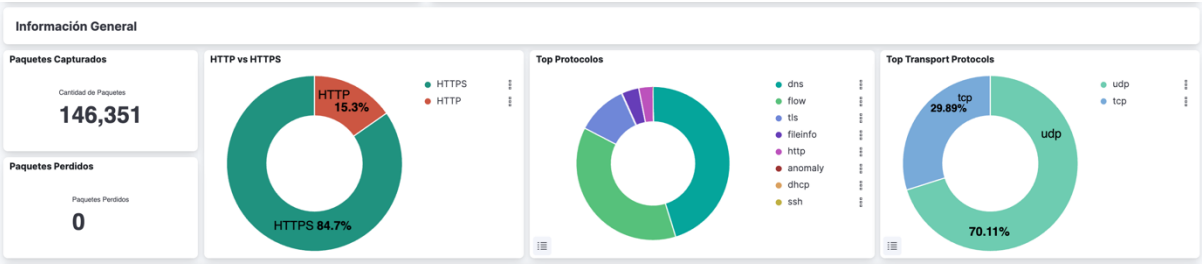


Figura 65. Tablero Eventos Información General. Elaboración Propia.

Información Hosts

En esta segunda sección, hemos aplicado muchos más filtros, en los cuales cada *ip.destino* la hemos sustituido por una cadena String (Ordenador 1, Tablet 1, Móvil 2, etc.) con tal de reconocer los dispositivos. Los elementos creados son:

- *Hosts Conectados*: se reconocen los hosts activos según la cantidad de paquetes que han transmitido en los últimos 10 min. Por ejemplo, vemos que el Ordenador 1 está apagado, por lo tanto, no transmite paquetes.
- *Hosts Mayor Actividad*: como su nombre indica, muestra que hosts transmiten más paquetes.
- *Hosts Acceso a páginas Inseguras*: esta última gráfica indica un TOP de hosts que acceden a sitios inseguros debido a que la información no está cifrada con el TLS.

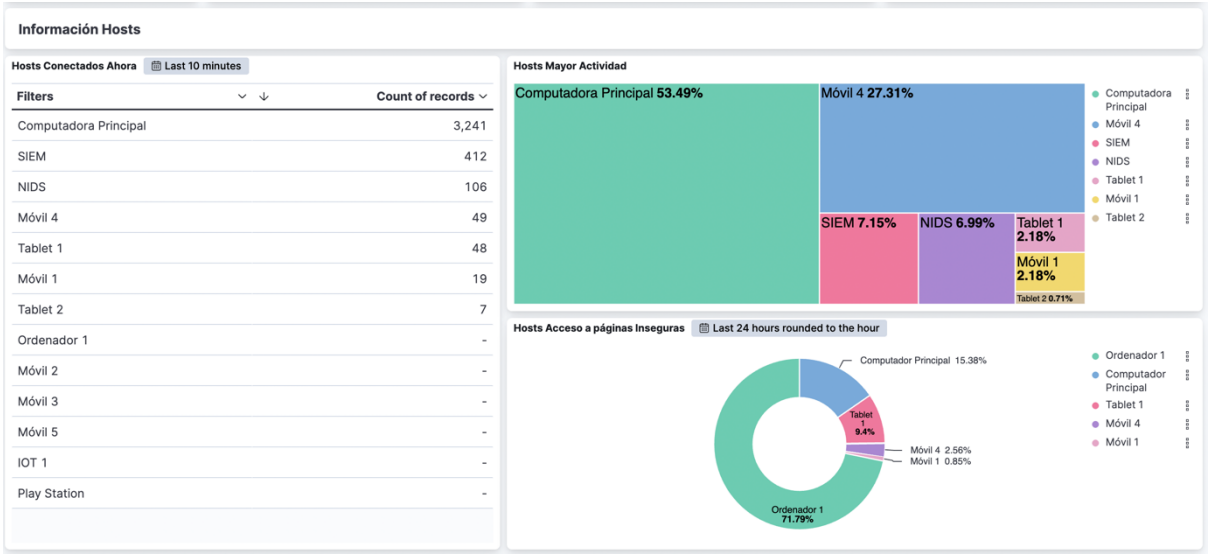


Figura 66. Tablero Eventos Información Hosts. Elaboración Propia.

4.3.6.4.2. Eventos: Vista Avanzada

Esta vista avanzada es una extensión de la vista resumen, encontramos estos 4 bloques:

- *Información general*: además de lo anterior, mostramos métricas en relación con las últimas 24 horas y el Top países destino.
- Información Hosts: añadimos la actividad del host a lo largo del día para conocer cuando es más activo.
- *Consultas HTTP sin TLS:* incluimos una lista con los dominios accedidos.
- *Registro*: incluimos un registro extenso que muestra información detallada de:
    - comunicaciones HTTP
    - todos los paquetes y eventos capturados.

Información General Paquetes

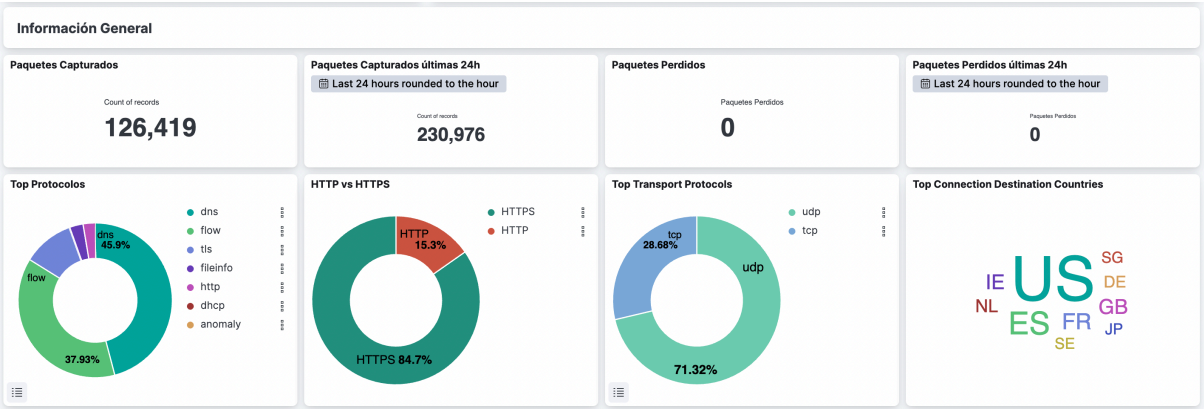


Figura 67. Información General Paquetes. Elaboración Propia.

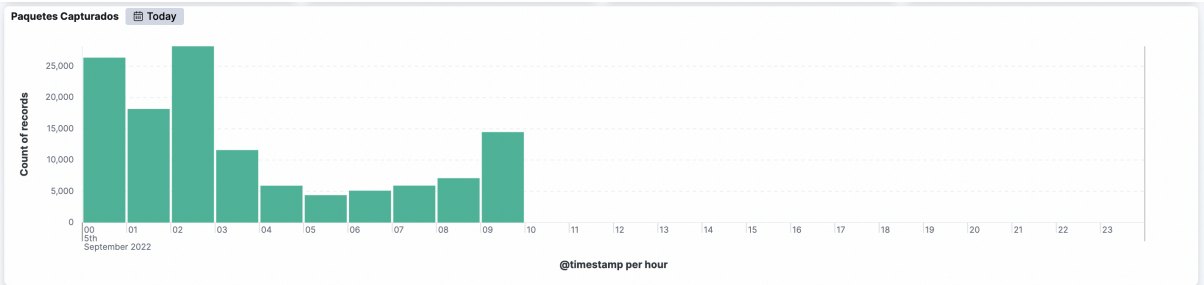


Figura 68. Paquetes capturados a lo largo del día. Elaboración Propia.

Información Hosts

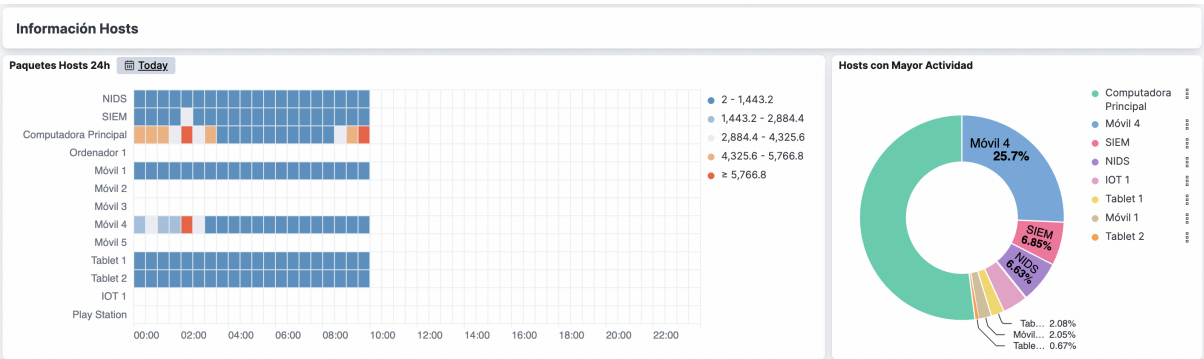


Figura 69.Información Hosts. Elaboración Propia.

Consultas HTTP sin TLS

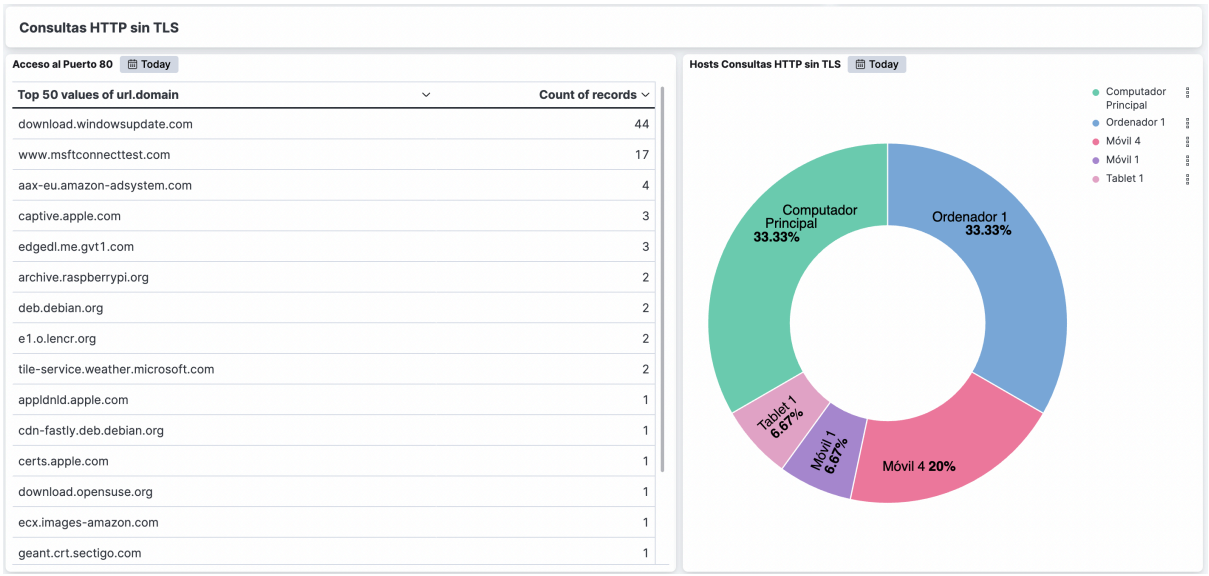


Figura 70. consultas HTTP sin TLS. Elaboración Propia.

Registro Consultas HTTP sin TLS

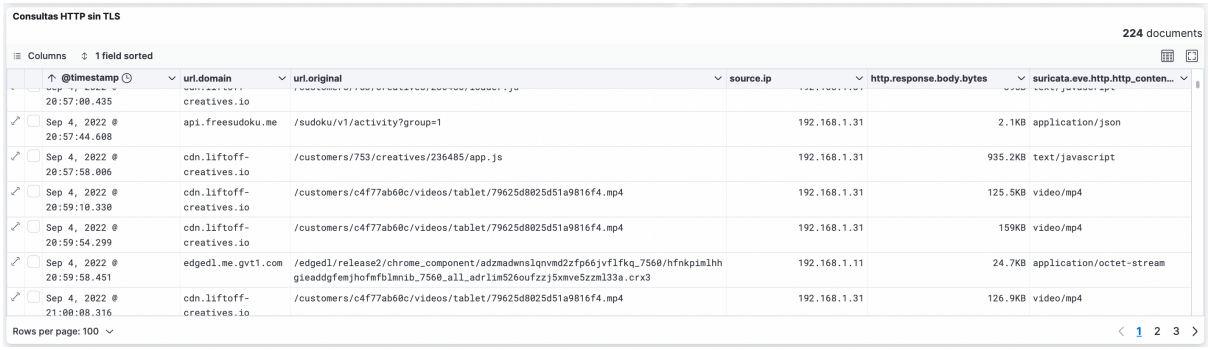


Figura 71. Regitro consultas HTTP. Elaboración Propia.

Registro Total de Eventos

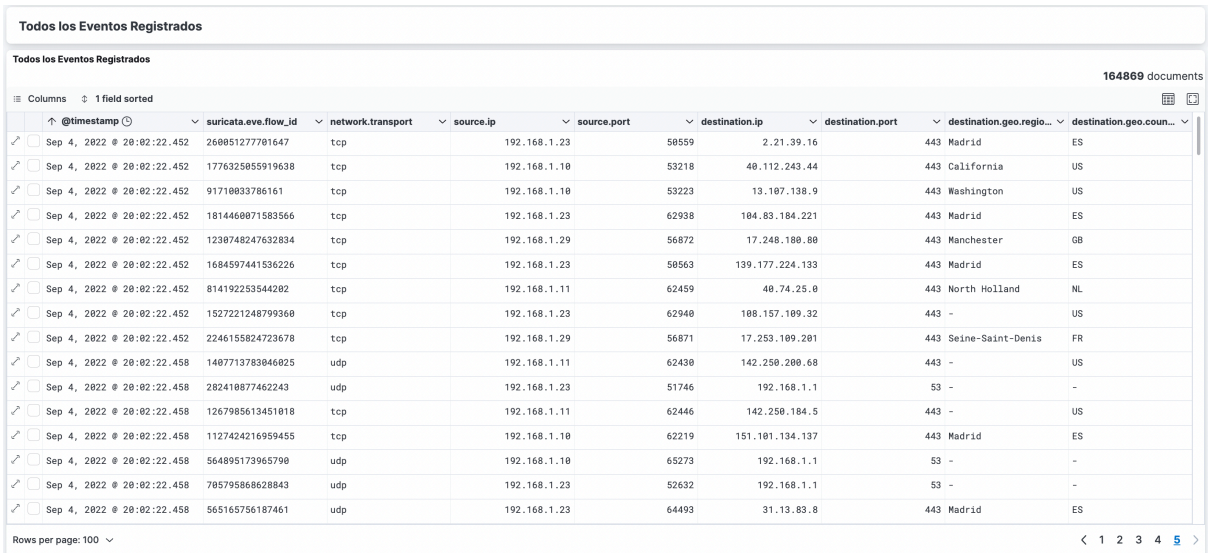


Figura 72. Registro todos los eventos. Elaboración Propia.

4.3.6.5. Tablero Alertas

Finalmente, el último tablero configurado es el de las Alertas. Este tablero principalmente muestra información en relación con:

- *Informaciones General Alertas*: cantidad de alertas y tipología, además de los hosts que más han activado las alertas.
- *Información Reglas*: alertas generadas por el set de reglas diseñado.
- *Información Blacklists*: alertas generadas por las blacklists.

4.3.6.5.1. Alertas Blacklists

En cuanto a las blacklists han sido agrupadas por: *Phishing Filter*, *Pup Filter*, *URlhaus*, *SSLBL* y *ET*. Agrupamos los proyectos de esta manera para facilitar la visualización. Dentro de Kibana, estas blacklists las hemos diferenciado por el inicio de la signatura.

Por ejemplo, todas las reglas provenientes de SSLBL la signatura inicia por "*SSLBL*":

/usr/local/etc/rules/sslblacklist_tls_cert.rules

```
alert tls $EXTERNAL_NET any -> $HOME_NET any (msg:"SSLBL: Malicious SSL certificate detected
(AsyncRAT C&C)"; tls_cert_fingerprint;
content:"98:9c:73:49:42:f4:82:91:75:00:9b:ed:75:3c:9d:4c:1d:29:d7:8e"; reference:url,
sslbl.abuse.ch/ssl-certificates/sha1/989c734942f4829175009bed753c9d4c1d29d78e/; sid:903204979;
rev:1;)
```

Entonces el filtro aplicado sería: *suricata.eve.alert.signature : SSLBL**

Además, debido a que (por fortuna) las reglas de las blacklists no han sido detectadas, hemos implementado un set de reglas sencillas (puerto 80, TLD .com, etc) para poder comprobar las visualizaciones.

4.3.6.5.2. Alertas: Vista Resumen

En este panel podemos observar una vista resumida de las alertas. Esta muestra información general de:

- Cantidad de alertas generadas.
- Cantidad de alertas generadas por cada Host.
- Diferencia entre las Alertas generadas por las Reglas y por las Blacklists.
- TOP de alertas provenientes de las Blacklists.
- TOP de reglas de “posibles amenezas”.

Como hemos mencionado anteriormente, las alertas de Blacklists que aparecen en la **Figura 73**, son tests de reglas. No se trata de alertas reales.

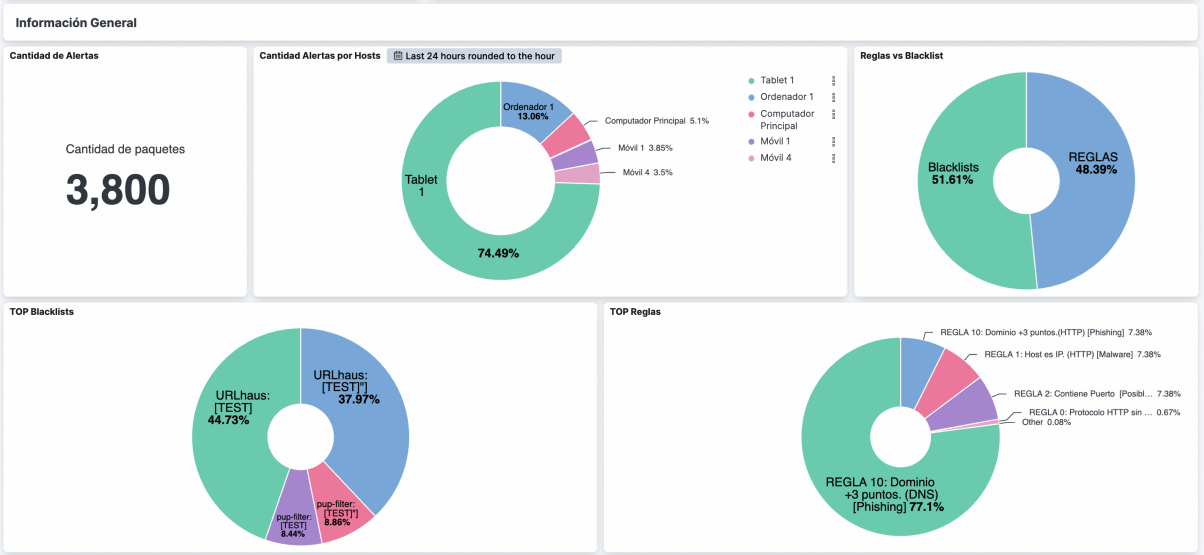


Figura 73. Tablero Alertas vista resumen. Elaboración Propia.

4.3.6.5.23. Alertas: Vista Avanzada

Finalmente, la vista avanzada contiene información más específica en relación con las alertas.

Información General

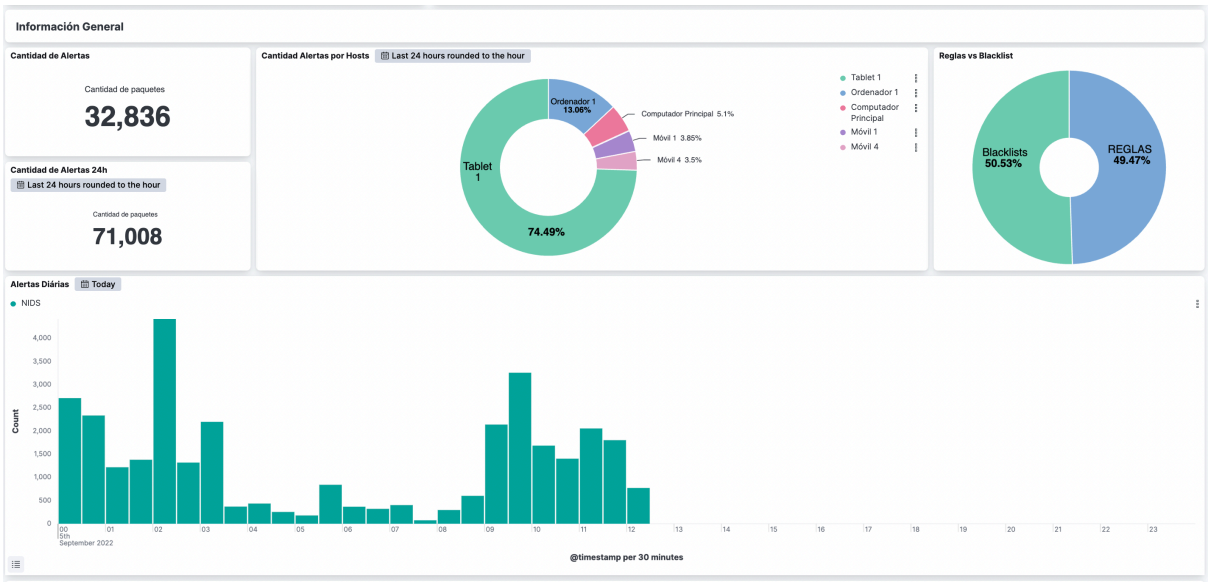


Figura 74. Tablero Alertas Avanzado. Elaboración Propia.

Información sobre “posibles amenazas”

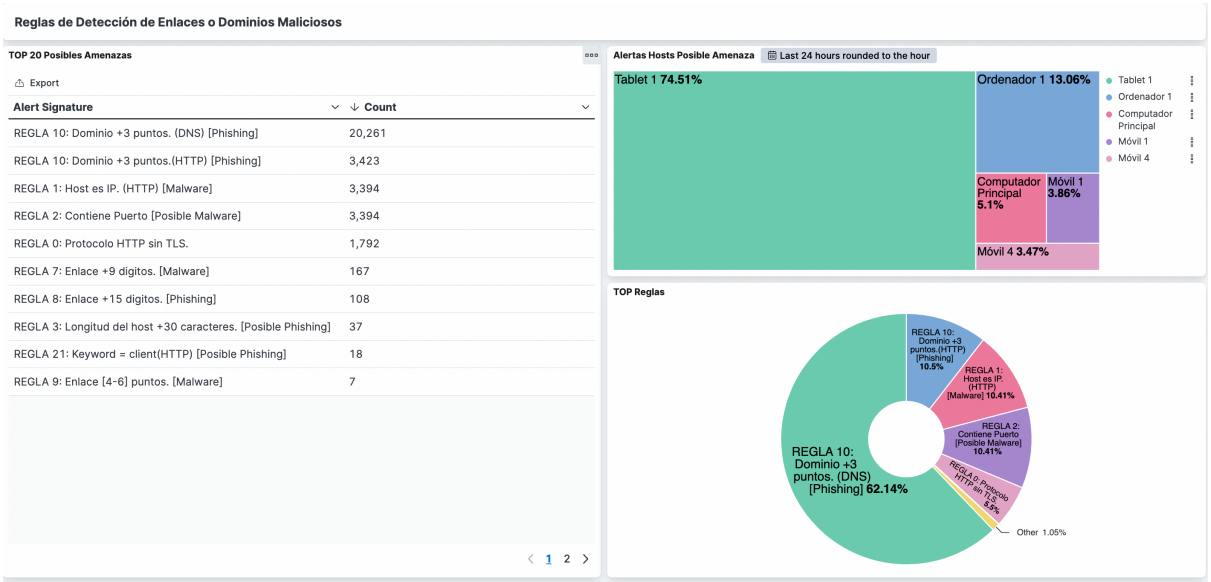


Figura 75. Tablero Alertas con Posibles Amenazas.

Información sobre Blacklists

Podéis observar las signaturas de las Blacklists como aparece la palabra TEST indicando que son reglas propias y no reales.

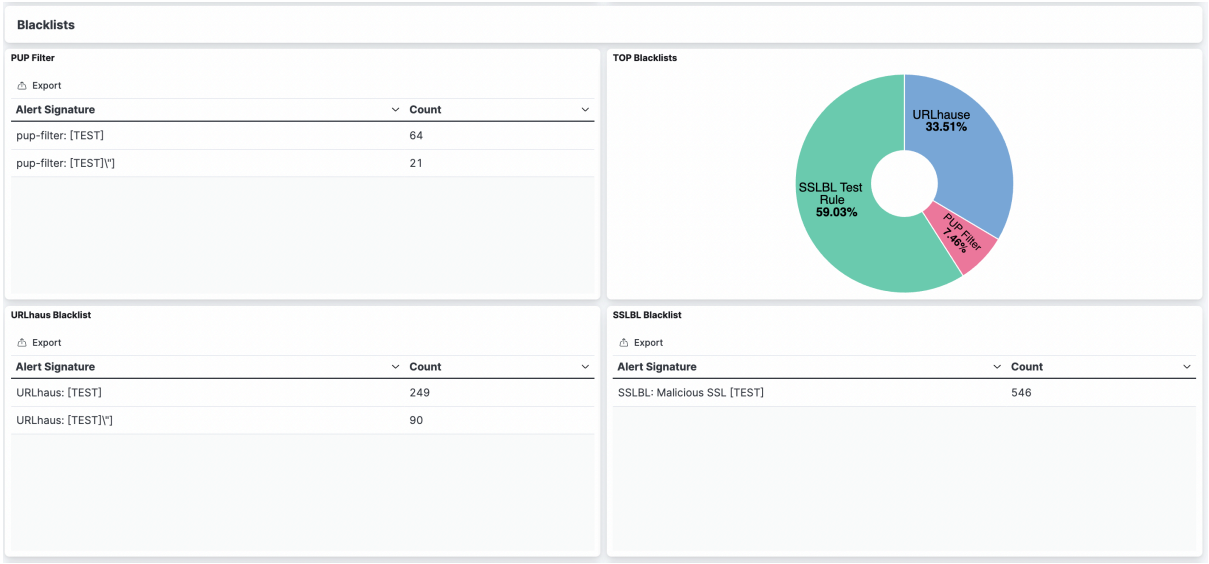


Figura 76. Tablero alertas Blacklists. Elaboración Propia.

4.3.7. Verificación acceso al SIEM

A continuación, vamos a comprobar el acceso al SIEM desde cualquier dispositivo conectado y con las credenciales establecidas, de tal forma que se puedan visualizar los tableros correctamente.

En la figura 77, podéis observar cómo se ha podido acceder al SIEM desde 3 diferentes dispositivos: ordenador, móvil y tablet. Además, el panel es responsive, lo que significa que los elementos se adaptan al tamaño de la pantalla.

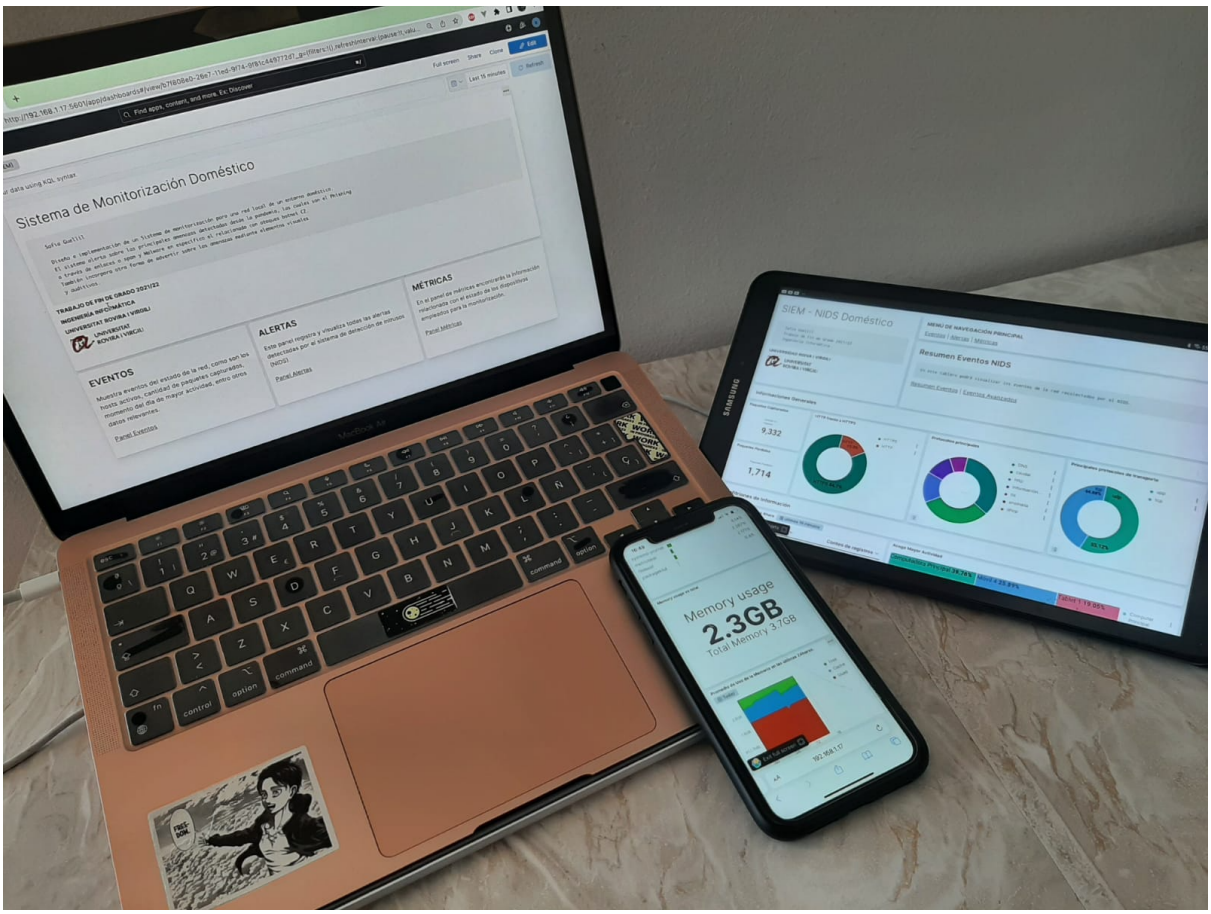


Figura 77. Acceso al SIEM desde diferentes dispositivos. Elaboración Propia.

4.4. Sistema de Alertas

Finalmente, implementamos el sistema de Alertas utilizando los pines GPIO de la Raspberry Pi. Estos nos sirven para enviar señales eléctricas que activarán los leds y buzzer cuando se detecte una entrada en el fichero de alertas: *fast.log*.

La longitud temporal de estas alertas será la siguiente:

- Los leds parpadearán 3 veces.
- El buzzer sonará durante 10 segundos.

4.4.1 Implementación Circuito

Primeramente, montamos el circuito con una protoboard, tres resistencias (una para cada led) y cables que unirán la placa con los pines GPIO de la Raspberry Pi. Los pines que hemos seleccionado han sido: 7 (led rojo), 13 (led amarillo), 15 (led verde) en el modo BOARD. Para el Buffer utilizamos el modo BCM: 26 (buzzer).

Estos pines fueron seleccionados básicamente por la localización de los cables.

En la **Figura 2**, podéis ver la disposición de estos elementos y como se han unido a la Raspberry Pi.

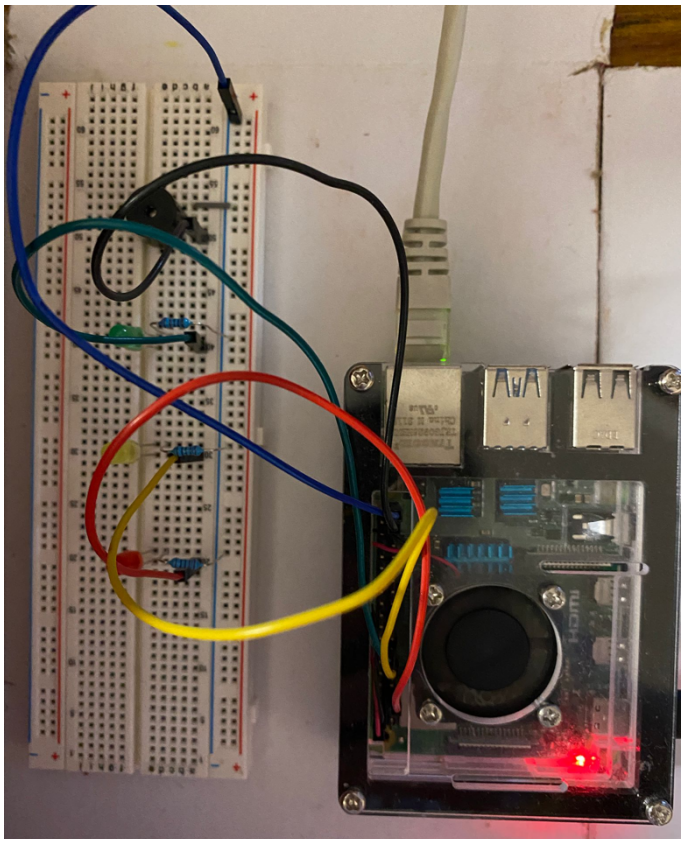


Figura 78. Implementación Sistema de Alerma. Elaboración Propia.

4.4.2 Verificación Circuito

Antes de iniciar con la implementación del servicio, es importante comprobar que el circuito funciona correctamente. Para ello realizaremos dos pruebas una para los leds y otra para el buzzer.

4.4.2.1 Verificación Leds

Primeramente, verificamos que los leds funcionan, para ello ejecutaremos el siguiente código. Este hará que los leds parpadeen 3 veces:

/etc/systemd/system/test_led.service

```
from time import sleep
import RPi.GPIO as GPIO

def encender(pin):
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(pin, GPIO.OUT)
        n = 3
        while n>0:
                GPIO.output(pin, True)
                time.sleep(1)
                GPIO.output(pin, False)
                time.sleep(1)
                n= n-1
        GPIO.cleanup()

encender(7)
encender(13)
encender(15)
```

En las siguientes figuras podéis observar cómo se han encendido correctamente las luces, además han parpadeado 3 veces como estaba establecido.

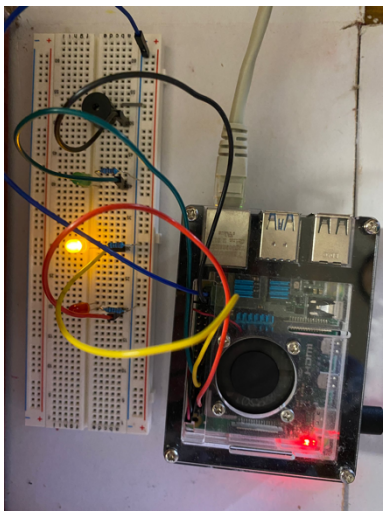


Figura 79. Evaluación led amarillo. Elaboración Propia.

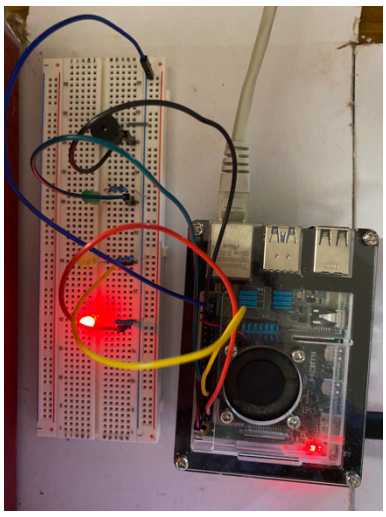


Figura 80. Evaluación led rojo. Elaboración Propia.

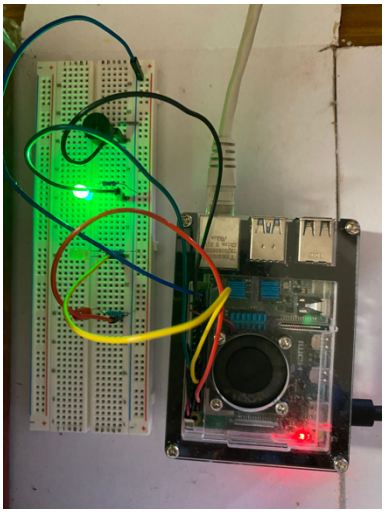


Figura 79. Evaluación led verde. Elaboración Propia.

4.4.2.1 Verificación Buzzer

La segunda comprobación, será la del buzzer o alarma. Para ello, utilizaremos el siguiente código que hará saltar la alarma durante 10 segundos:

/etc/systemd/system/test_buzzer.service

```
import time
from gpiozero import Buzzer

def buzzer():
    fi = True
    buzzer = Buzzer(26)
    while fi:
        buzzer.on()
        sleep(10)
        fi = False

buzzer()
```

Resultados: se ha escuchado correctamente y en el tiempo establecido.

4.4.3. Implementación Sistema de Alertas

Después de comprobar que el circuito funciona correctamente, crearemos un fichero en Python utilizando el código anterior para los leds y buzzer. Este nuevo programa estará pendiente de leer las nuevas escrituras en el fichero *fast.log*, el cual es dónde Suricata escribe las amenazas detectadas.

Hemos creado el siguiente código en Python[65]:

/usr/local/etc/alert/alert.py

```
import time
from gpiozero import Buzzer
from time import sleep
import RPi.GPIO as GPIO

path = '/var/log/suricata/fast.log'

GPIO.setmode(GPIO.BOARD)

def checkLine(last_line):
    return last_line[last_line.find("Priority: ")+10]

def encender(pin):
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(pin, GPIO.OUT)
        n = 3
        while n>0:
                GPIO.output(pin, True)
                time.sleep(1)
                GPIO.output(pin, False)
                time.sleep(1)
                n= n-1
        GPIO.cleanup()

def buzzer():
    fi = True
    buzzer = Buzzer(26)
    while fi:
        buzzer.on()
        sleep(10)
        fi = False

if __name__ == '__main__':

    f = open(path, 'r')

    while True:
        line = f.readline()
        if not line:
            time.sleep(1)
            print(Nada Nuevo')
        else:
            print('Nueva Alerta: ', line)
            num = checkLine(line)
            if (num == "1"):
                encender(7)
                buzzer(26)
            elif (num == "2"):
                encender(13)
            else:
                encender(15)
```

[65] https://stackoverflow.com/questions/182197/how-do-i-watch-a-file-for-changes

4.4.4 Implementación Servicio de Alertas

Finalmente, implementamos este sistema en un servicio, de esta manera podremos iniciarlo o detenerlo en cualquier momento:

/etc/systemd/system/alertas.service

```
[Unit]
Description=Alertas Suricata
After=suricata.service
Requires=suricata.service

[Service]
Type=simple
ExecStart= python3 /usr/local/etc/alert/alert.py
ExecReload=/bin/kill -HUP $MAINPID


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

Comprobamos que funciona el servicio correctamente:

```
sudo systemctl enable alertas
sudo systemctl start alertas
sudo systemctl status alertas
```

```
● alertas.service - Alertas Suricata
     Loaded: loaded (/etc/systemd/system/alertas.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-09-06 12:52:33 CEST; 4min 39s ago
   Main PID: 19479 (python3)
      Tasks: 1 (limit: 4164)
        CPU: 219ms
     CGroup: /system.slice/alertas.service
             └─19479 python3 /usr/local/etc/alert/alert.py

Sep 06 12:57:04 NIDS python3[19479]: Nueva Alert:  09/06/2022-00:02:17.393170  [**] [1:1999119:0] pup-filter: [TEST]
Sep 06 12:57:04 NIDS python3[19479]: Nueva Alert:  09/06/2022-00:02:19.086713  [**] [1:100:0] REGLA 10: Dominio +3 pu
Sep 06 12:57:04 NIDS python3[19479]: Nueva Alert:  09/06/2022-00:02:19.086713  [**] [1:19999:0] URLhaus: [TEST] [**]
Sep 06 12:57:04 NIDS python3[19479]: Nueva Alert:  09/06/2022-00:02:19.088147  [**] [1:100:0] REGLA 10: Dominio +3 pu
Sep 06 12:57:04 NIDS python3[19479]: Nueva Alert:  09/06/2022-00:02:19.088147  [**] [1:19999:0] URLhaus: [TEST] [**]
Sep 06 12:57:04 NIDS python3[19479]: Nueva Alert:  09/06/2022-00:02:20.128425  [**] [1:100:0] REGLA 10: Dominio +3 pu
Sep 06 12:57:04 NIDS python3[19479]: Nueva Alert:  09/06/2022-00:02:20.128425  [**] [1:19999:0] URLhaus: [TEST] [**]
Sep 06 12:57:04 NIDS python3[19479]: Nueva Alert:  09/06/2022-00:02:21.484756  [**] [1:1999119:0] pup-filter: [TEST]
Sep 06 12:57:04 NIDS python3[19479]: Nueva Alert:  09/06/2022-00:02:22.071640  [**] [1:19999:0] URLhaus: [TEST] [**]
```

Figura 80. Output Servicio Alertas. Elaboración Propia.

En la figura 82, podemos ver como el servicio funciona correctamente y lee las diferentes alertas generadas por Suricata. Además, visualmente se activan los leds y el buzzer.

4.5. Evaluación Sistema Final

En la figura 83, se puede observar el resultado final de todo el prototipo. En esta sección, evaluaremos el rendimiento del sistema en nuestro entorno local.

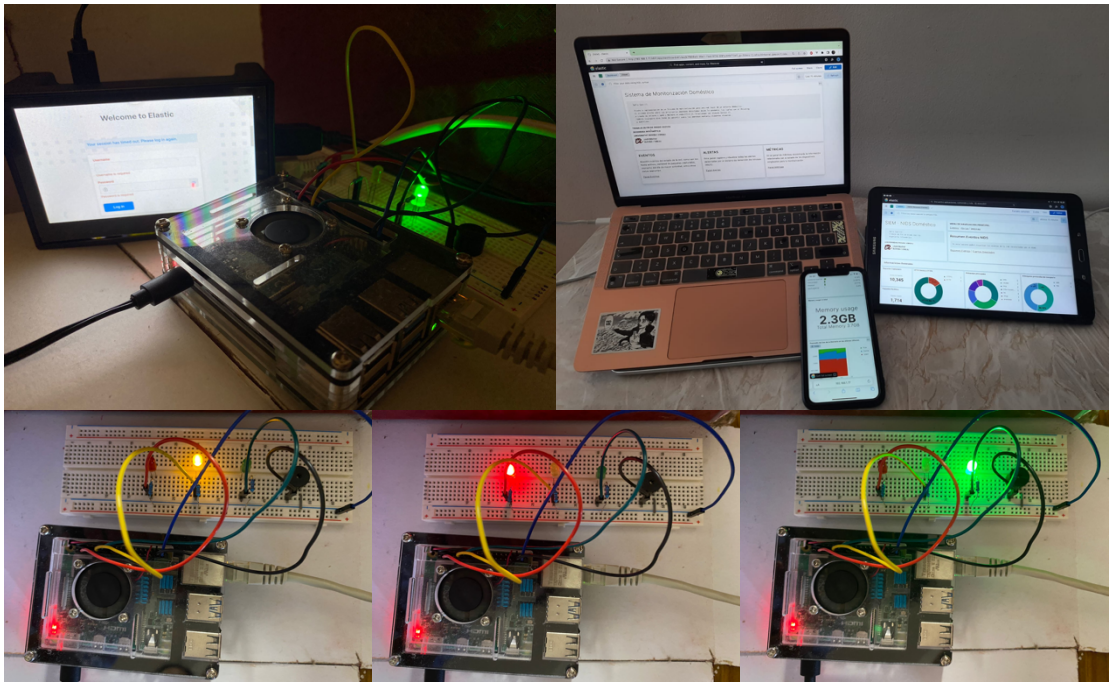


Figura 81. Resultado Final. Elaboración Propia.

A lo largo de la implementación del proyecto, hemos ido realizando diferentes pruebas de cada una de las partes individuales del mismo. Como hemos indicado en la planificación, se ha seguido una metodología *Waterfall* (o en cascada) para cada uno de los subsistemas. De tal manera, que no iniciamos la implementación de la siguiente fase hasta verificar el correcto funcionamiento de la anterior. Este procedimiento ha sido necesario, ya que al tratarse de tres sistemas que deben funcionar en conjunto, es importante que cada elemento individual ejerza correctamente sus funciones.

En esta sección, nos centramos en evaluar todo el conjunto, es decir, verificar si el prototipo cumple con sus funciones como sistema de monitorización y detección.

Principalmente, verificamos las siguientes características:

1. **Arquitectura**: comprobar si la arquitectura del proyecto es correcta, si el flujo de paquetes transcurre de la manera adecuada y se recoge correctamente por el SIEM.
2. **Hardware**: analizar si el hardware seleccionado es el adecuado.
3. **Detección de Amenazas**: comprobar si se detectan y clasifican las amenazas correctamente.

El sistema de Alertas no será evaluado, ya que se ha comprobado su correcto funcionamiento en una sección anterior.

4.5.1. Evaluación de la Arquitectura

Para la evaluación de la arquitectura emplearemos el panel Eventos del SIEM.

Primeramente, mediante la vista Resumen podemos observar estadísticas generales de la red. Este panel estará configurado con el rango temporal de "Today", el cual mostrará estadísticas desde las 00.00h hasta las 16.00h, momento del día en el que se ha realizado el test.

En la figura 83, observamos que se han recibido alrededor de 162.919 paquetes, lo cual, al ser un valor tan alto, indica que el sistema está funcionando correctamente recolectando los paquetes que producen los distintos dispositivos.

Por otro lado, también observamos que Suricata está perdiendo alrededor de 6.334 paquetes que representa aproximadamente 0,4% del total. Este último valor al ser tan diminuto no representa un gran problema.

Finalmente, vemos que se están recibiendo diferentes tipos de paquetes correspondientes a los distintos protocolos.

Figura 82. Tablero de Eventos. Elaboración Propia.

Por otro lado, también podemos visualizar la información relacionada con los hosts. En este caso, observamos: los hosts conectados (últimos 10 minutos), los hosts con mayor actividad y un ranking de los hosts que acceden a páginas sin certificado SSL. Esta información, nos verifica que se están conectando correctamente los dispositivos a la red y que se están recibiendo correctamente sus datos. También observamos el comportamiento que tienen en relación con el acceso a sitios HTTP, como es el caso del Ordenador 1.

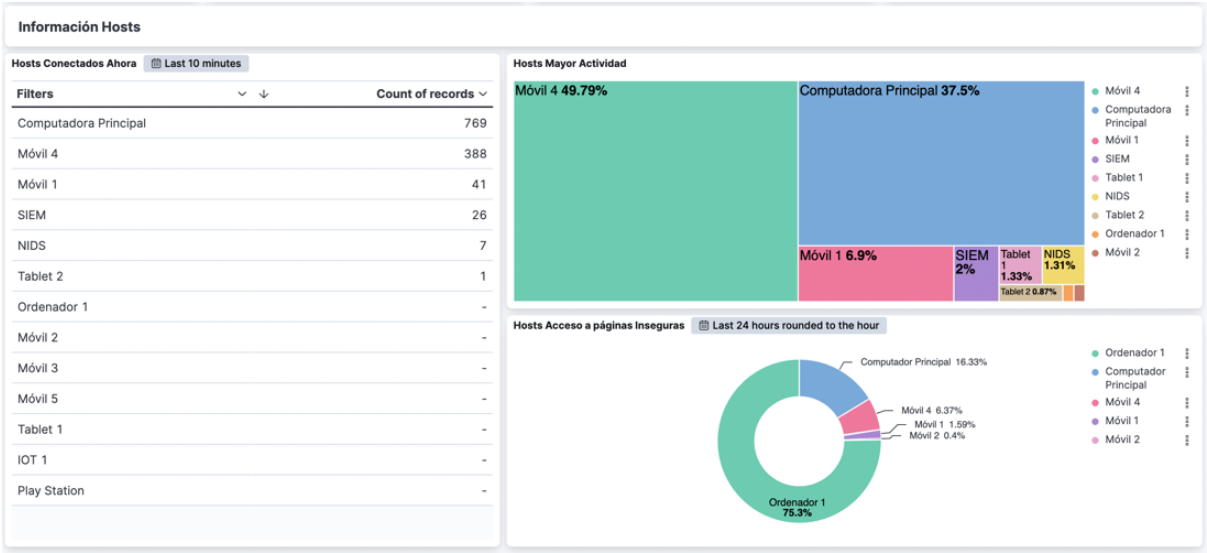


Figura 83. Información de Hosts. Elaboración Propia.

Finalmente, con el panel de Eventos Avanzado, podemos observar más métricas como son la cantidad de paquetes capturados en las últimas 24h o la actividad de los hosts. En este caso, el móvil 4 es el de mayor actividad, seguido por la Computadora principal.

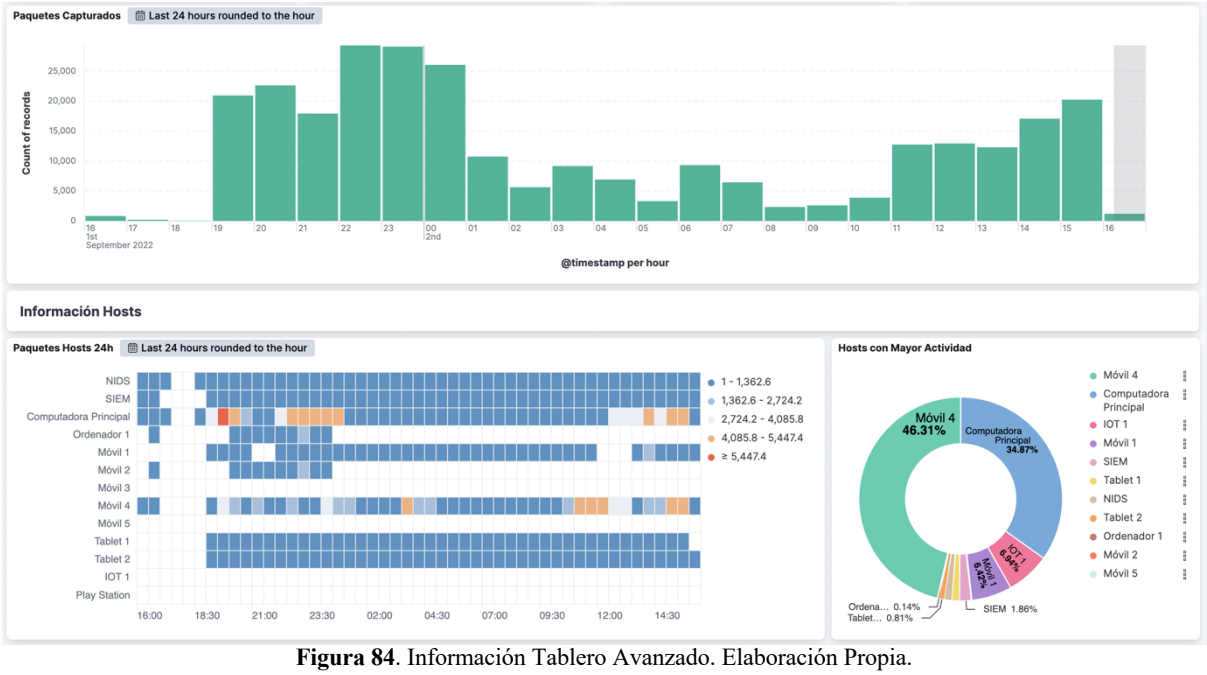


Figura 84. Información Tablero Avanzado. Elaboración Propia.

En general, concluimos que se están recibiendo de forma correcta los paquetes de los distintos dispositivos conectados en nuestra red local. Además de ser adecuadamente procesados, evaluados y visualizados por el sistema de monitorización.

4.5.2. Evaluación del Hardware

En este apartado determinamos si el Hardware seleccionado, Raspberry Pi 4B de 4 GB de RAM, es adecuado para la implementación del prototipo en nuestro entorno. Para esta evaluación se ha configurado el parámetro temporal a "*Today*" y se ha empleado el tablero Métricas del SIEM.

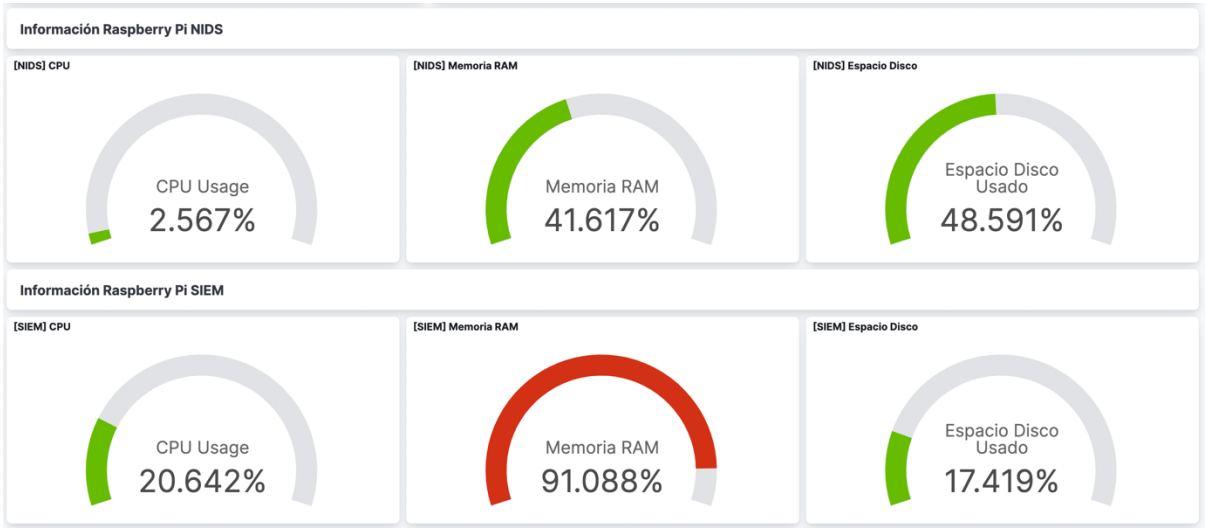


Figura 85. Tablero Métricas. Elaboración Propia.

Métricas CPU

En general, el uso de la CPU en cuanto al dispositivo NIDS, ha sido bastante bajo alrededor del 2%. Por otro lado, el SIEM ha utilizado la CPU hasta un 20%, siendo Elasticsearch el primer consumidor seguido de Metricbeat.

En cualquier caso, se trata de valores bastante pequeños, lo cual indica que la CPU no se está aprovechando en toda su capacidad.

Métricas Memoria RAM

Por lo general, observamos que la máquina del SIEM utiliza más recursos, sobre todo la RAM, alrededor de 3,4Gb de un total de 3,7Gb. De hecho, ha estado alrededor del 88-92% durante las últimas 24 horas. Esto se explica básicamente porque Elasticsearch es un motor de búsqueda que en generalmente consume muchos recursos. Si accedemos al panel avanzado y observamos las métricas de la RAM, efectivamente vemos que el proceso llamado "java" que se trata del set Elastic ocupa alrededor del 64%.

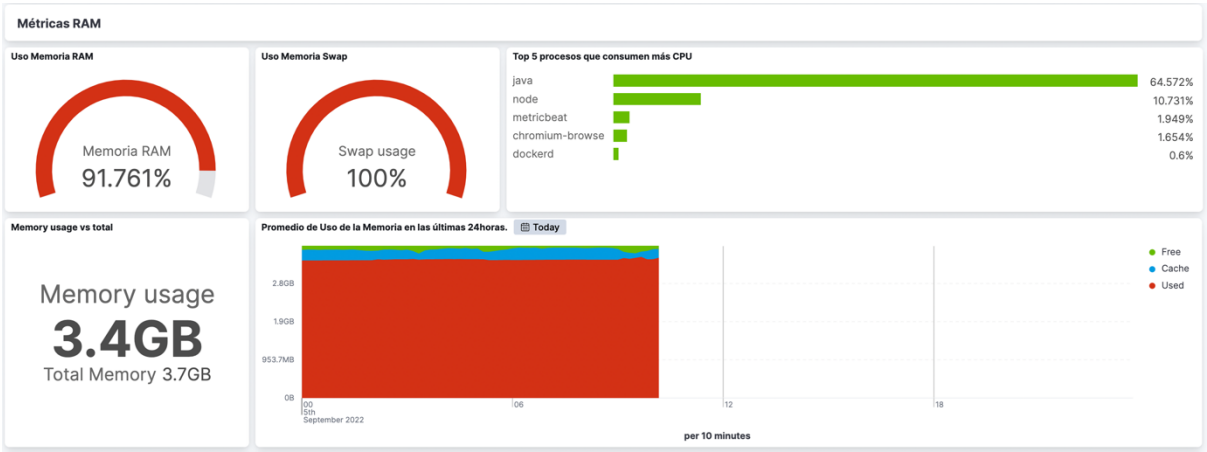


Figura 86. Información RAM del SIEM. Elaboración Propia.

De hecho, la máquina SIEM tiene instalado un SO con Escritorio, además de incorporar una pequeña pantalla como dispositivo periférico. Esta pantalla ha sido probada en otras ocasiones y funciona correctamente, pero al estar utilizando el sistema un promedio de 90% de RAM, presenta retardos y bloqueos continuamente, lo cual dificulta bastante la visualización.

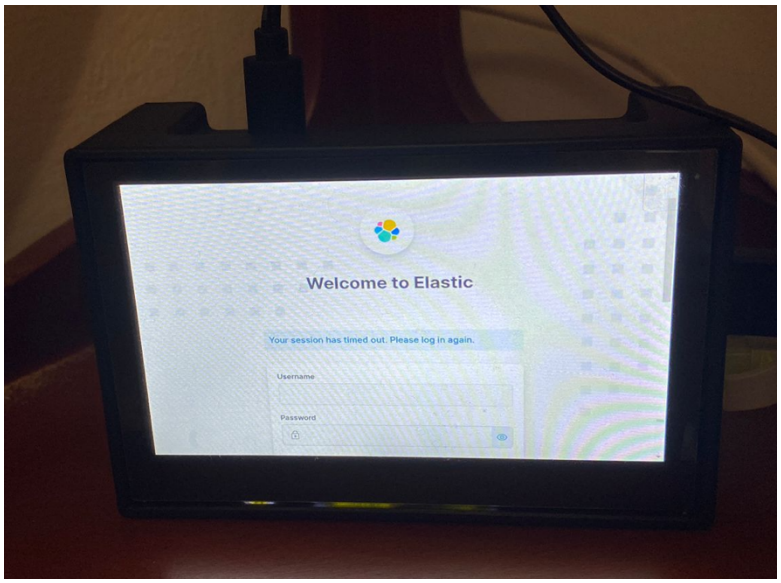


Figura 87. Pantalla incorporada a la Raspberry Pi del SIEM. Elaboración Propia.

Por otro lado, en el caso del NIDS, observamos que apenas llega al 40%, con un consumo promedio de 2Gb, lo cual es menos de la mitad de la capacidad.

En resumen, en el caso de la Raspberry Pi del SIEM, en nuestro entorno, necesitaría más RAM, ya que 4GB es bastante poco. O también, utilizar un SO sin Escritorio y sin dispositivos periféricos para reducir el consumo. Por otro lado, el NIDS podríamos reducir la capacidad a 3 GB y seguiría correctamente funcional (en nuestro entorno).

Métricas Disco

Finalmente, en cuanto al disco, se ha utilizado una tarjeta de 64GB para el NIDS y 128GB para el SIEM. Este último se le ha otorgado más memoria dado que debe almacenar más datos como son las métricas de ambos sistemas.

El prototipo se ha mantenido trabajando alrededor de un mes, de hecho, debido a algunas pruebas, en algunas ocasiones no se han registrado datos. Aun así, en un periodo de un mes, la tarjeta SD del NIDS está casi al 50%. Este dato podría estar relacionado con el servicio logrotate, el cual se ha establecido el parámetro rotate a 30, que supone almacenar 30 ficheros de log. Se ha establecido este parámetro para registrar una gran cantidad de datos pasados, pero es un valor bastante alto y lo vemos reflejado en la ocupación del disco.

4.5.3. Evaluación del sistema de Detección de Amenazas

En general, las reglas de las blacklists no han sido detectadas, de hecho, las observadas en las visualizaciones de las secciones anteriores, han sido reglas pensadas para que se detecten. De esta manera podamos verificar la detección de signaturas por parte del SIEM. En todo caso, es correcto, puesto que significa que ningún usuario de nuestra red doméstica ha accedido a sitios maliciosos.

Por otro lado, en cuanto a las reglas, en un rango temporal de 24h, han saltado más de 60.000 alertas, lo que significa que hay una gran tasa de falsos positivos. De hecho, alrededor del 50% han sido debido a la regla 10: *dominio +3 puntos* [HTTP], seguida de la regla 0 con alrededor del 43%.

Estas dos reglas deben ser eliminadas o refinadas, ya que están generando una cantidad enorme de "*posibles amenazas*", hasta tal punto que resulta muy complicado consultar su origen para establecer si es una amenaza real o no.

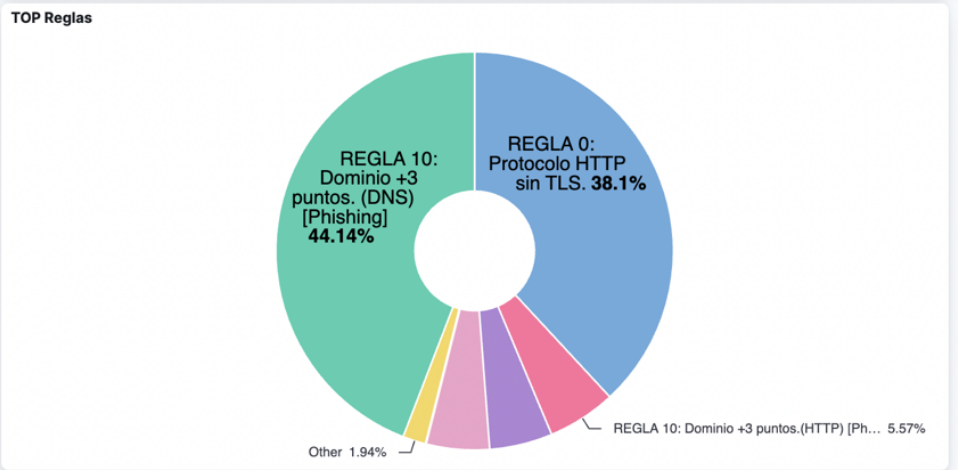


Figura 88. TOP reglas de detección. Elaboración Propia.

TOP 20 Posibles Amenazas

Export

Alert Signature	Count
REGLA 10: Dominio +3 puntos. (DNS) [Phishing]	31,144
REGLA 0: Protocolo HTTP sin TLS.	26,881
REGLA 10: Dominio +3 puntos.(HTTP) [Phishing]	3,929
REGLA 1: Host es IP. (HTTP) [Malware]	3,614
REGLA 2: Contiene Puerto [Posible Malware]	3,613
REGLA 7: Enlace +9 digitos. [Malware]	584
REGLA 8: Enlace +15 digitos. [Phishing]	474
REGLA 3: Longitud del host +30 caracteres. [Posible Phishing]	282
REGLA 21: Keyword = client(HTTP) [Posible Phishing]	18
REGLA 9: Enlace [4-6] puntos. [Malware]	7

Figura 89. TOP 20 reglas de detección. Elaboración Propia.

5. Conclusiones

El prototipo diseñado e implementado ha podido ser adaptado correctamente en la arquitectura local, permitiendo así el análisis del tráfico que transcurre por toda la red doméstica. El NIDS ha logrado captar una gran parte de los paquetes (alrededor del 99%) que han sido examinados en busca de posibles amenazas. Además, las métricas generadas por este último han sido visualizadas y gestionadas correctamente desde el SIEM, el cual es accesible desde cualquier dispositivo conectado a la red. De igual modo, el sistema de alertas ha estado pendiente de las detecciones del NIDS para activarlas cuando sea necesario.

Sin embargo, también hemos observado que la arquitectura seleccionada no era la más adecuada para nuestro entorno local, puesto que existe una gran diferencia entre el uso del hardware por parte del NIDS y del SIEM. Por un lado, vemos que con una Raspberry Pi de 3 Gb de RAM sería suficiente para el NIDS, mientras que el SIEM necesitaría (algo más de memoria) alrededor del 6GB. Además, en ambos casos no aprovechaban toda la capacidad de la CPU.

En cuanto al sistema de detección, no hemos detectado alertas reales, debido a que los usuarios de nuestro entorno no han cometido ninguna acción peligrosa que comprometa el sistema. Aunque hemos comprobado y testeado las signaturas de las reglas de detección.

Finalmente, cabe destacar que se ha padecido muchas dificultades en adquirir las Raspberry Pi, dado que actualmente, debido a la escasez de chips y la alta demanda, son bastante difíciles de conseguir.

5.1. Trabajo Futuro

En un futuro, nos gustaría implementar un sistema aún más robusto como sería un IPD (Sistema de Prevención de Intrusos), el cual no solamente detecta las amenazas, sino que también las previene bloqueando los paquetes maliciosos hacia nuestros dispositivos.

Por otro lado, en el sistema de alertas, nos gustaría incorporar envíos al correo electrónico, de tal manera que también podamos ser alertados estando fuera del hogar.

Finalmente, nos gustaría diseñar reglas de detección que consideren más características (como es el contenido de la página HTML o la procedencia del dominio) e incorporar firmas de diferentes tipos de IOC como sería el JA3.

Referencias

[1]. *2022 State of the Phish | Proofpoint ES*. (2022, 2 junio). Proofpoint. https://www.proofpoint.com/es/resources/threat-reports/state-of-phish

[2]. 6.1. Rules Format — Suricata 6.0.5 documentation. (s. f.). Suricata. https://suricata.readthedocs.io/en/suricata-6.0.5/rules/intro.html

[3]. ¿Qué es la gestión de eventos y de la información de seguridad (SIEM)? | IBM. (s. f.). IBM. https://www.ibm.com/es-es/topics/siem

[4]. *¿Qué es Elasticsearch?* (s. f.). Elastic. https://www.elastic.co/es/what-is/elasticsearch

[5]. *abuse.ch - Figthing malware and botnets*. (s. f.). https://abuse.ch/

[6]. Amat, A. (2021, 16 marzo). *Brecha digital en la educación, una consecuencia que deja la Covid-19*. La Vanguardia. https://www.lavanguardia.com/vida/formacion/20210310/6272400/brecha-digital-educacion-colegios-pandemia-desigualdad.html

[7]. Belcic, I. (2021, 8 octubre). *Definición de botnet: ¿Qué es una botnet?* Avast. https://www.avast.com/es-es/c-botnet

[8]. Camisso, J. (2022, 25 enero). How To Build A SIEM with Suricata and Elastic Stack on Debian 11. DigitalOcean Community. https://www.digitalocean.com/community/tutorials/how-to-build-a-siem-with-suricata-and-elastic-stack-on-debian-11

[9]. CCN-CERT. (2020, septiembre). Ciberamenazas y Tendencias (Edición 2020). https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/5377-ccn-cert-ia-13-20-ciberamenazas-y-tendencias-edicion-2020/file.html

[10]. colaboradores de Wikipedia. (s. f.). *Anexo:Puertos de red+ - Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Anexo:Puertos_de_red

[11]. colaboradores de Wikipedia. (2022, 1 marzo). Indicador de compromiso. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Indicador_de_compromiso

[12]. colaboradores de Wikipedia. (2022b, junio 14). Ingeniería social (seguridad informática). Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Ingenier%C3%ADa_social_(seguridad_inform%C3%A1tica)

[13]. colaboradores de Wikipedia. (2022c, agosto 6). Phishing. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Phishing

[14]. colaboradores de Wikipedia. (2022b, mayo 16). Raspberry Pi. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Raspberry_Pi

[15]. CrowdStrike. (2022, 23 marzo). IOA vs IOC: Defining & Understanding The Differences | CrowdStrike. Crowdstrike.Com. https://www.crowdstrike.com/cybersecurity-101/indicators-of-compromise/ioa-vs-ioc/

[16]. Davinci Group. (2020, 16 julio). *¿Qué son los Beats de elastic? + Ejemplo de uso de Filebeat*. https://www.davincigroup.es/beats-elastic-ejemplo-filebeat/

[17]. *Feodo Tracker | Browse Botnet C&Cs*. (s. f.). https://feodotracker.abuse.ch/browse/

[18]. *Filebeat: Lightweight Log Analysis & Elasticsearch*. (s. f.). Elastic. https://www.elastic.co/beats/filebeat

[19]. Herrero, H. (2021, 11 agosto). Visualizando los registros de Suricata en Grafana o Kibana | Blog Bujarra.com. Blog Bujarra.com | Blog IT de Héctor Herrero. https://www.bujarra.com/visualizando-los-registros-de-suricata-en-grafana-o-kibana/

[20]. Hoover, C. (2022, mayo). Comparative Study of Snort 3 and Suricata Intrusion Detection Systems. ScholarWorks@UARK. https://scholarworks.uark.edu/csceuht/105/

[21]. Ikwu, R. E. (2022, 30 marzo). *Extracting Feature Vectors From URL Strings For Malicious URL Detection*. Medium. https://towardsdatascience.com/extracting-feature-vectors-from-url-strings-for-malicious-url-detection-cbafc24737a
[22]. ITACA. (s. f.). U4.6 Ataques pasivos vs ataques activos. Escuela Especializada en Ingeniería ITCA FEPADE. https://virtual.itca.edu.sv/Mediadores/cms/u46_ataques_pasivos_vs_ataques_activos.html
[23]. Juliana, V. A. P. B. (2021, 7 junio). *Spot suspicious activity on your local network with Suricata Intrusion Detection System (IDS) on Raspberry Pi*. Juliana Fajardini. https://jufajardini.wordpress.com/2021/02/15/suricata-on-your-raspberry-pi/#required-materials
[24]. Klein, A. O. (2021, enero). *El impacto del COVID-19: la digitalización como bien común.* Real Instituto Elcano. https://www.realinstitutoelcano.org/wp-content/uploads/2021/01/dt1-2021-ortega-el-impacto-del-covid-19-digitalizacion-como-bien-comun.pdf
[25]. Kumar, A., & Shrivastava, S. (2018, 30 octubre). *Comparative Study on Network Vulnerabilities and Intrusion Detection System*. ijrar. http://ijrar.com/upload_issue/ijrar_issue_20542796.pdf
[26]. Malwarebytes. (2019, 8 mayo). Malware Emotet: una introducción al troyano bancario. https://es.malwarebytes.com/emotet/
[27]. Martínez López, M. A. (2010, 13 agosto). Diseño y simulación de un sistema para detección de intrusos en redes de comunicaciones utilizando OMNET++. http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/martinez_l_ma/
[28]. *Metricbeat: Lightweight Shipper for Metrics*. (s. f.). Elastic. https://www.elastic.co/beats/metricbeat
[29]. Not sure whether to use Logstash or Beats | Filebeat Reference [8.4]. (s. f.). Elastic. https://www.elastic.co/guide/en/beats/filebeat/current/diff-logstash-beats.html
[30]. OISF. (s. f.). Home. Suricata. https://suricata.io/
[31]. R. (2022, 1 junio). Las mejores prácticas para implementar una estrategia SIEM. CIBERSEGURIDAD .blog. https://ciberseguridad.blog/las-mejores-practicas-para-implementar-una-estrategia-siem/
[32]. R. (2018, 18 noviembre). Por qué las herramientas SOAR revitalizarán el ecosistema SIEM. CIBERSEGURIDAD .blog. https://ciberseguridad.blog/por-que-las-herramientas-soar-revitalizaran-el-ecosistema-siem/
[33]. R. (2018a, junio 10). Soluciones open source para la gestión de logs en ciberseguridad. CIBERSEGURIDAD .blog. https://ciberseguridad.blog/soluciones-open-source-para-la-gestion-de-logs-en-ciberseguridad/
[34]. Rabie A. Ramadan, Bassam W. Aboshosha, Jalawi Sulaiman Alshudukhi, Abdullah J. Alzahrani, Ayman El-Sayed, Mohamed M. Dessouky, "Cybersecurity and Countermeasures at the Time of Pandemic", *Journal of Advanced Transportation*, vol. 2021, Article ID 6627264, 19 pages, 2021. https://doi.org/10.1155/2021/6627264
[35]. Randstad Research, CEOE. (2021, octubre). *¿Qué ha cambiado con el COVID-19? Transformación y adaptación, nuevos retos y soluciones RRHH.* https://www.randstadresearch.es/que-ha-cambiado-con-el-covid-19-transformacion-y-adaptacion-nuevos-retos-y-soluciones-rrhh/
[36]. Segu-Info. (2022, 23 febrero). ¿Qué es CyberSecurity Mesh Architecture (CSMA) y por qué es el futuro? https://blog.segu-info.com.ar/2022/02/que-es-cybersecurity-mesh-architecture.html?m=0

[37]. Sistema Estadístico de Criminalidad. (2022, agosto). Informe sobre Cibercriminalidad en España (Edición 2021). Ministerio del Interio. https://www.interior.gob.es/opencms/pdf/prensa/balances-e-informes/2021/Informe-Cibercriminalidad-2021.pdf
[38]. Swanagan, M. C. (2021, 21 octubre). Intrusion Detection VS Prevention Systems: What's The Difference? PurpleSec. https://purplesec.us/intrusion-detection-vs-intrusion-prevention-systems/#IDSTypes
[39]. Suricata User Guide — Suricata 6.0.6 documentation. (s. f.). Suricata. https://suricata.readthedocs.io/en/suricata-6.0.6/
[40]. *URL 2016 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. (s. f.-b). de https://www.unb.ca/cic/datasets/url-2016.html
[41]. *What is Kibana?* (s. f.). Elastic. https://www.elastic.co/what-is/kibana