Universitat Oberta de Catalunya

uoc.edu

Implementación de un IDS de bajo coste para uso doméstico o en la pequeña empresa

Sadoht Gómez Fernández

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

Análisis de Datos

Director:
Joan Caparrós Ramírez

Responsable del área:
Helena Rifà Pous

31 de diciembre de 2019

A mamá y Juan -"los indestructibles"- por inculcarme la importancia de la educación, el inconformismo, y a no tirar nunca la toalla.

Resumen

El objetivo de este trabajo es encontrar una solución de seguridad de bajo coste, destinada al hogar y a la pequeña empresa, que permita detectar comportamientos maliciosos o anómalos en la red local, alertando al administrador y permitiendo analizar el estado de la red y el origen de los eventos.

Para ello se llevó a cabo la instalación de un punto de acceso Wifi cuyo tráfico se analiza con Suricata (sistema de detección de intrusiones), y almacena todos los eventos de seguridad en la base de datos Elasticsearch. La instalación se completó con Kibana para visualizar todos los datos almacenados y ElastAlert para el envío de notificaciones al administrador.

Una vez logrado un sistema funcional con las herramientas escogidas se estudiaron topologías alternativas, analizando el coste de implementación de cada una, junto con sus ventajas e inconvenientes.

Por otro lado, se desarrolló un bot de Telegram para ayudar a relacionar las direcciones IP de cada evento con su respectivo dispositivo y usuario. Con estos datos se generó un panel de usuario en Kibana que muestra la información de seguridad y su uso de la red a fin de ayudar a detectar comportamientos anómalos.

Finalmente se hicieron algunas pruebas para demostrar las capacidades de detección de Suricata, y observar el funcionamiento general de sistema, visualizando los datos en Kibana y recibiendo las notificaciones pertinentes.

Abstract

The goal of this project is to find a low-cost security solution, aimed at the family home and small businesses, which allows the detection of malicious or anomalous behavior on the local network, alerting the administrator and serving to analyze the state of the network and the origin of events.

For this purpose, a Wifi access point was installed, whose traffic is analyzed by Suricata (Intrusion Detection System), storing all security events in the Elasticsearch database. The installation was completed with Kibana to visualize all the stored data and ElastAlert to send notifications to the administrator.

Once a functional system was achieved with the chosen tools, alternative topologies were studied, analyzing the cost of implementing each one, along with its advantages and disadvantages.

Furthermore, a Telegram bot was developed to help linking the IP addresses of each event with their respective device and user. With this data, a dashboard was generated in Kibana to show the security information and network usage, in order to help in detecting anomalous behavior.

Finally, some tests were made to demonstrate the detection capabilities of Suricata, and to observe the general functionality of the system, visualizing the data in Kibana and receiving the relevant notifications.

Índice

Índice de figuras

1. Introducción

1.1 Explicación detallada del problema a resolver

Contexto del problema

Los entornos domésticos y las pequeñas empresas cada vez cuentan con más dispositivos conectados.

En el caso del hogar, la tendencia hacia el Internet de las Cosas (IoT, por sus siglas en inglés) conlleva tener cada vez más electrodomésticos y aparatos conectados a internet. Neveras, cafeteras o robots de limpieza empiezan a compartir cada vez más las mismas redes domésticas con el resto de ordenadores del hogar. Estos dispositivos están ya tan avanzados que son perfectamente susceptibles de ataques informáticos e infecciones de malware como cualquier computadora. Más si cabe, son especialmente vulnerables, ya que no suelen disponer de mecanismos para la detección de ataques o infecciones. Las redes domésticas suman, además, cada día más teléfonos móviles, tablets, ordenadores, etc. de cada miembro de la familia.

Por otro lado, las pequeñas empresas son líderes en la adopción de políticas BYOD (Bring Your Own Device), que permiten a los empleados llevar sus propios dispositivos (teléfonos, portátiles, tablets, etc.) al lugar de trabajo y utilizarlos para realizar sus tareas. Esta tendencia presenta nuevos retos para la gestión de la seguridad, ya que los administradores de los sistemas informáticos tienen limitada su gestión de vulnerabilidades al no poder controlar el software de los dispositivos posiblemente infectados de los empleados.

Por ello, un análisis del uso que todos estos dispositivos hacen de la red a la que están conectados podría ser muy beneficioso.

Solución que se plantea

Se plantea encontrar una solución de seguridad que permita analizar el uso de la red por parte de los distintos dispositivos, a fin de detectar comportamientos anómalos o maliciosos. Esta solución alertará al administrador cada vez que se detecte algún evento sospechoso, y proveerá de herramientas que permitan analizar en profundidad el estado de la red y el origen de los eventos detectados.

Para implementar un sistema con estos requisitos, se propone instalar un punto de acceso Wifi (AP) de forma que se tenga acceso a todo el tráfico. Este tráfico será analizado con algún sistema de detección de intrusiones (IDS), que será el

responsable de detectar los comportamientos anómalos o maliciosos y registrar tales eventos. Para que el administrador pueda analizar la información generada por el IDS se propone la instalación de un sistema de almacenamiento y análisis de eventos (SIEM, Security Information and Event Management), que le permitirá generar visualizaciones a partir de la información recibida desde el IDS a fin de ayudarle a comprender el origen y naturaleza de las amenazas, y tomar las decisiones necesarias para evitar posibles daños.

Junto con el sistema de gestión de eventos se configurará un sistema de alertas que monitorizará la información almacenada y enviará al administrador un aviso en caso de que algo debiera ser analizado en profundidad.

Teniendo en cuenta que el enfoque del proyecto es el hogar y la pequeña empresa, el coste de implantación de la solución final deberá ser lo más reducido posible. Para lograrlo, se priorizará el uso de software de código abierto y licencias gratuitas, y la utilización de hardware de bajo coste.

1.2 Objetivos del proyecto

A continuación, se describen los objetivos del proyecto divididos en dos categorías. Los objetivos principales garantizan un sistema final funcional y usable en el entorno doméstico y de pequeña empresa.

Los objetivos específicos son pruebas que buscan optimizar la solución final, y explorar las diferentes configuraciones, tanto de software como hardware, que podrían ser desplegadas en función del presupuesto o de las necesidades de seguridad del usuario final.

Objetivos principales:

- Investigar el estado del arte en cuanto a herramientas disponibles para instalar un AP, IDS, SIEM y sistema de alertas.
- Hacer una selección de herramientas, compatibles entre ellas, que minimicen el coste final de la implementación.
- Aprender a utilizar las herramientas escogidas.
- Instalar y configurar las herramientas escogidas, de modo que se demuestre que todo el sistema puede funcionar en su conjunto.
- Comprobar que el administrador es alertado al realizar una conexión sospechosa desde un dispositivo conectado al punto de acceso configurado.

- Comprobar que los eventos quedan debidamente registrados y son accesibles y visualizables con la herramienta SIEM escogida.

- Elaborar un informe que incluya la experiencia obtenida con las diferentes herramientas y las posibles soluciones de bajo coste que hayan sido encontradas.

Objetivos específicos:

- Estudiar más en profundidad las posibilidades de detección que ofrece el software de IDS escogido, así como su consumo de recursos en función del número de reglas aplicadas.
- Estudiar las diferentes topologías (de bajo coste) en las que se podrían desplegar las herramientas seleccionadas, y elaborar un informe con las posibles restricciones encontradas.
- Explorar distintas herramientas y opciones de alertado (e-mail, Telegram, Slack, etc.).
- Explorar las posibilidades de visualización y análisis de los eventos de seguridad con la herramienta SIEM seleccionada.
- Estudiar las posibilidades en cuanto a creación de perfiles de usuario, con la finalidad de mejorar las alertas por detección de anomalías.

1.3 Descripción de la metodología

El desarrollo del sistema se llevará a cabo de forma iterativa e incremental. Se realizarán tres iteraciones correspondientes con los tres hitos de entrega especificados por la asignatura. En cada iteración se estudiará cada componente del sistema y las posibles mejoras. Al final de cada hito se actualizará el documento de la memoria para reflejar los avances, enumerar las pruebas realizadas, justificar las decisiones tomadas e informar de posibles problemas inesperados.

La primera iteración consistirá en una fase de investigación. Durante esta etapa, no se pretende alcanzar un sistema funcional, sino estudiar cada uno de los componentes, hacer pruebas por separado, y tomar las decisiones en cuanto a herramientas finales, recursos necesarios y topología de la red.

La segunda fase pretende implementar las decisiones tomadas en la iteración anterior. Se trata de conseguir que los distintos componentes funcionen juntos, aunque la solución no sea la óptima para un despliegue real. Se comprobará que se ha conseguido una integración satisfactoria de todas las herramientas y que cada una hace su trabajo. Se estudiará el consumo de recursos de cada parte con la finalidad de proponer cambios que ayuden a encontrar una solución final usable en los entornos objetivo: hogar y pequeña empresa.

Durante la tercera y última fase se experimentará con diferentes topologías de bajo presupuesto, razonables para la puesta en producción del proyecto. También se intentará mejorar la configuración del IDS para detectar más y mejor. Se realizarán pruebas con las distintas reglas de detección, y se estudiará la creación de perfiles de usuario que permitan la detección de comportamientos anómalos.

1.4 Listado de las tareas a realizar

A continuación se enumeran las tareas planeadas para cada fase del proyecto.

1.4.1 Fase de investigación

Durante esta primera fase se pretende acabar de estudiar el estado del arte y tomar las decisiones en cuanto a las herramientas de software a utilizar. Se divide en las tareas descritas a continuación. Cabe mencionar que la ejecución de estas tareas se solapará en el tiempo, ya que el objetivo es encontrar un conjunto que integre bien todas las partes. La fase de investigación incluirá las siguientes tareas:

Elección de herramienta IDS. Durante la fase de planificación se encontraron dos opciones interesantes: Suricata y Zeek. Durante la fase de investigación se estudiarán en profundidad para acabar de tomar una decisión. Se prestará especial atención al consumo de recursos de cada opción para determinar si es posible que el IDS comparta el hardware con el AP.

Elección de herramienta SIEM. Un estudio breve del estado del arte destaca las herramientas basadas en Elasticsearch. Las opciones más interesantes, Kibana y Graylog, serán analizadas en esta etapa. Se deben estudiar ambas opciones en detalle, prestando especial atención a la integración de cada una de ellas con el resto de herramientas que conformarán el sistema final.

Elección de sistema de alertas. Esta tarea se realizará de forma conjunta con la anterior, ya que depende en gran medida de la decisión que se tome en cuanto al SIEM. En el caso de decantarse por Graylog, el sistema de alertas ya viene integrado. En caso de escoger Kibana habría que buscar herramientas de terceros.

Diseño de la topología inicial. En base al estudio de las herramientas que serán utilizadas, y los recursos que cada una necesite, se propondrá un diseño inicial de la topología de red. Este diseño inicial será utilizado en la fase de implementación, y podrá ser modificado en la fase de optimización.

Redacción del primer entregable. Se expondrán las herramientas y topología escogidas y los razonamientos que han llevado a tales decisiones.

1.4.2 Fase de implementación

El objetivo de esta fase es conseguir una instalación completa de las herramientas que conformarán el sistema final, y comprobar que todas las piezas se integran correctamente. Las tareas para esta fase son:

Configuración del punto de acceso. Se instalará y configurará el software que cree el punto de acceso Wifi, junto con el servicio DHCP para la asignación automática de IPs. Se realizará la instalación sobre una Raspberry Pi, lo que implica también la instalación del sistema operativo, y las configuraciones para acceder a ella de forma remota y segura. Se realizará una prueba de conexión desde otro dispositivo para comprobar que funciona correctamente y se tiene salida a internet.

Instalación y configuración del IDS. En función de la topología de red decidida en la fase anterior, se instalará el IDS y se configurará para que analice el tráfico enrutado por el AP. Se hará una primera configuración sencilla para que analice y detecte conexiones sospechosas realizadas a IPs en listas negras. Para comprobar que funciona correctamente se escogerá alguna de las IPs contenidas en la lista negra configurada, se hará un intento de conexión desde algún dispositivo conectado al AP, y se observará si el IDS ha generado el evento correctamente.

Instalación y configuración del SIEM. En base a la topología de red decidida en la fase anterior, se instalará la herramienta SIEM escogida. En esta tarea se deberá hacer la integración del IDS con el SIEM, de modo que los eventos generados por el primero sean transmitidos y almacenados por el segundo. Una vez configurado, se repetirá la prueba de conexión a una IP de la lista negra, y se comprobará que el evento es correctamente recibido y almacenado en el SIEM.

Experimentación con las opciones de visualización. Una vez se tengan almacenados los eventos del IDS en la base de datos de la solución SIEM se estudiarán las posibilidades que la herramienta ofrece en cuanto a análisis y visualización de los mismos.

Configuración del sistema de alertas. En caso de haber escogido Kibana como SIEM, esta tarea incluirá la instalación de las herramientas de terceros seleccionadas en la fase anterior. Se considerarán distintas opciones para alertar al administrador del sistema, siendo Telegram e E-mail las opciones preferentes, y se repetirá de nuevo la prueba de conexión a una IP de la lista negra, esta vez para comprobar que el sistema funciona correctamente en su conjunto y el administrador recibe las alertas pertinentes.

Redacción del segundo entregable. Se redactará la experiencia obtenida durante la fase de implementación.

1.4.3 Fase de optimización

La fase de optimización incluirá las siguientes tareas:

Creación de perfiles de usuario. Investigar posibles soluciones para la elaboración de perfiles de usuario para poder detectar comportamientos anómalos que no serían considerados sospechosos de otra manera. Alertar al administrador para un posterior análisis.

Estudio de las posibilidades del IDS. Aumentar el número de reglas activas en el IDS pueda llegar a agotar los recursos del hardware que lo opera. Observar también cómo esto afecta al consumo de recursos del SIEM, al tener que empezar a almacenar y procesar un mayor volumen de datos.

Propuesta de topologías reales. Considerando que el enfoque del proyecto es la economía doméstica y de pequeña empresa, se valorarán distintas topologías de red y su coste asociado a la hora de implantar el sistema final. Se estudiarán opciones como la instalación completa del sistema en la red local usando sólo hardware de bajo coste (Raspberry Pi), y la opción de mover algunos de los servicios necesarios a la nube.

Redacción de la memoria final. Añadir al documento con los hallazgos encontrados en esta fase, y completarlo con las conclusiones y referencias pertinentes.

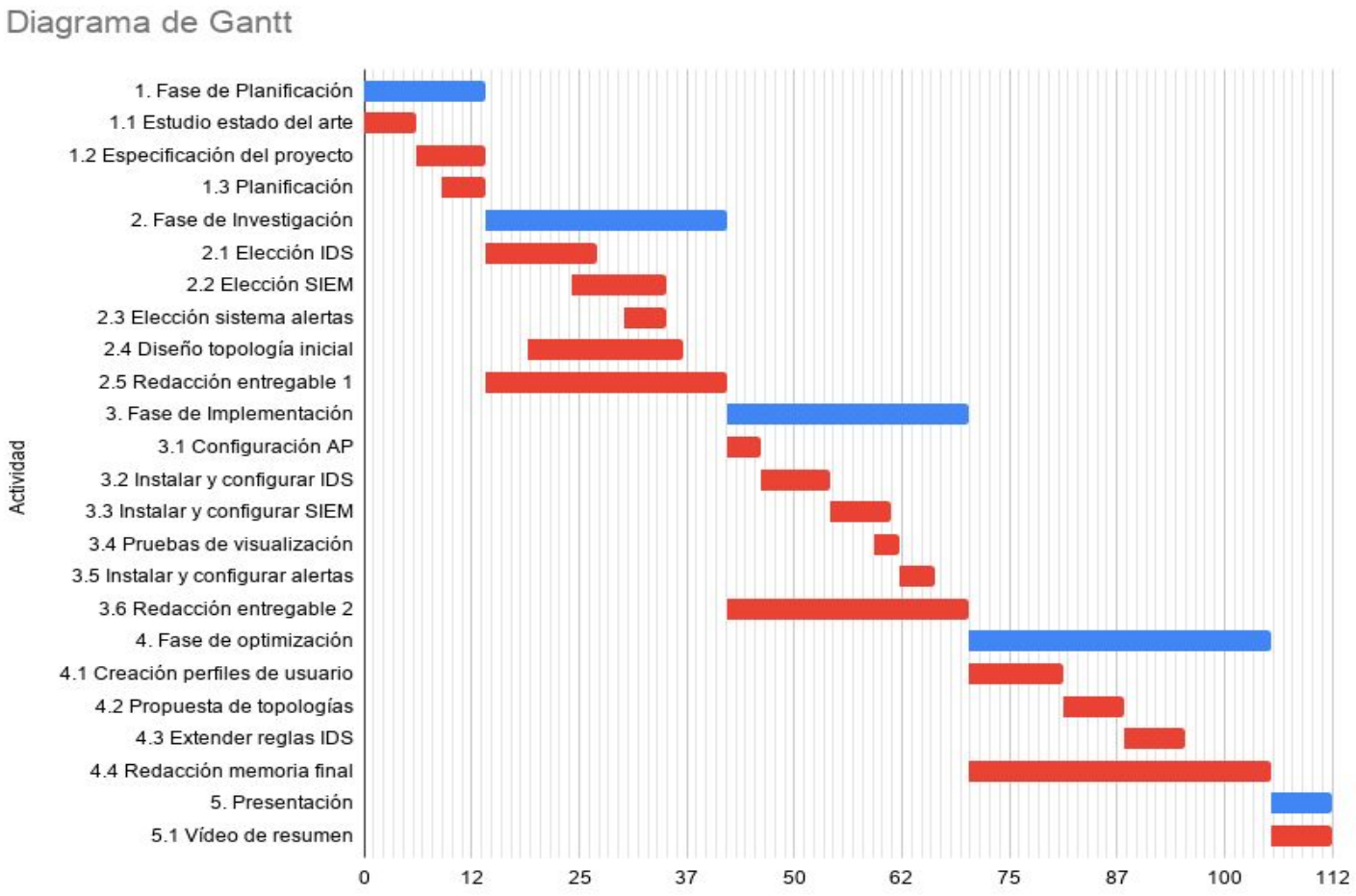
Presentación en vídeo. Crear un vídeo de unos 15 minutos con una síntesis del trabajo y una demostración del funcionamiento del sistema.

1.5 Planificación temporal detallada

Detalle por tareas

Actividad	Fecha Inicio	Fecha Fin	Duración
1. Fase de Planificación			
1.1 Estudio estado del arte	18/9/19	23/9/19	6
1.2 Especificación del proyecto	24/9/19	1/10/19	8
1.3 Planificación	27/9/19	1/10/19	5
2. Fase de Investigación			
2.1 Elección IDS	2/10/19	14/10/19	13
2.2 Elección SIEM	12/10/19	22/10/19	11
2.3 Elección sistema alertas	18/10/19	22/10/19	5
2.4 Diseño topología inicial	7/10/19	24/10/19	18
2.5 Redacción entregable 1	2/10/19	29/10/19	28
3. Fase de Implementación			
3.1 Configuración AP	30/10/19	2/11/19	4
3.2 Configuración IDS	3/11/19	10/11/19	8
3.3 Configuración SIEM	11/11/19	17/11/19	7
3.4 Pruebas de visualización	16/11/19	18/11/19	3
3.5 Configuración alertas	19/11/19	22/11/19	4
3.6 Redacción entregable 2	30/10/19	26/11/19	28
4. Fase de optimización			
4.1 Creación perfiles de usuario	27/11/19	7/12/19	11
4.2 Propuesta de topologías	8/12/19	14/12/19	7
4.3 Extensión reglas IDS	15/12/19	21/12/19	7
4.4 Redacción memoria final	27/11/19	31/12/19	35
5. Presentación			
5.1 Vídeo de resumen	1/1/20	7/1/20	7

Nótese que festivos y fines de semana no han sido excluidos de la planificación de forma intencionada debido a la situación laboral del alumno durante el periodo establecido para el proyecto.



1.6 Revisión del estado del arte

Para la planificación del proyecto se ha hecho una breve investigación sobre las herramientas disponibles para cada apartado.

Hardware de bajo coste

En principio se considera principalmente Raspberry Pi, que es un ordenador de placa reducida muy popular y de bajo coste. La elección de este hardware viene condicionado por el gran número de entusiastas y la extensa comunidad que han hecho de la Raspberry Pi una de las computadoras más vendidas de la historia[1].

[1] https://magpi.raspberrypi.org/articles/raspberry-pi-sales

En junio de 2019 se lanzó el modelo 4 de Raspberry, que incluye una versión que cuenta con hasta 4 GB de RAM. Este modelo más potente, aunque un poco más caro, da ciertas garantías de éxito en la implantación del sistema. De todas formas, se evaluará la posibilidad de usar las versiones más antiguas, a fin de ofrecer a los entusiastas de estos dispositivos una segunda vida para los modelos anteriores que ya tengan en sus casas.

También se podría considerar la posibilidad de utilizar hardware virtual. Existen muchos servicios que alquilan servidores virtuales de bajo coste. Una de las opciones favoritas es DigitalOcean, que ofrece, por ejemplo, un servidor de 4 Gb de RAM por 20 € al mes. A largo plazo sería una decisión más costosa que el despliegue en Raspberries, pero esta opción también ofrece otras ventajas, como el fácil redimensionado de recursos para adaptarlos a los cambios de las necesidades. Por ejemplo, en caso de necesitar más RAM para el mantenimiento de un servicio, bastaría con un par de clicks y el reinicio del servidor virtual. Obviamente esto afectará a la factura de cada mes.

En caso de que la ejecución del IDS junto con el AP en el mismo dispositivo ralentice demasiado la conexión a internet, se considerará una solución basada en un switch con port-mirroring. El TP-Link TL-SG105E de 5 puertos es una buena opción, a un precio de 22 €. En caso de encontrar una solución software para el problema, se necesitará un switch igualmente para la comunicación con otros dispositivos; el TL-SG105 de 5 puertos (mismo modelo, pero sin port-mirroring) cuesta la mitad, 11 euros.

Sistemas de detección de intrusos (IDS)

Los IDS de código abierto que destacan en la actualidad son Snort y Suricata. Snort inició su desarrollo en 1998, y desde entonces se ha convertido casi en el estándar de facto. Suricata es un proyecto más reciente (la primera versión fue publicada en 2009) y fue diseñado desde un principio para aprovechar las CPUs modernas multinúcleo con procesamiento paralelo. Recientemente, Snort ha empezado a soportar también procesamiento en paralelo, pero Suricata es un proyecto más maduro en cuanto a este aspecto.

La gran ventaja de Suricata respecto a otros IDS modernos es que es compatible con el lenguaje de reglas de Snort, y por tanto se beneficia de todas esas reglas de detección de intrusiones desarrolladas durante años.

Se ha estudiado también el estado de ambos proyectos en cuanto a desarrollo y mantenimiento actuales y Suricata parece un proyecto mucho más activo durante el último año.

Otra opción, algo menos popular pero con un desarrollo y una comunidad muy activos es Zeek (antiguamente Bro). Zeek es un IDS con una perspectiva diferente a la de Snort y Suricata: se centra principalmente en la detección de anomalías y es instalado

frecuentemente en Raspberries. Durante la fase de investigación se llevará a cabo un estudio más en profundidad de esta opción.

Sistemas de monitorización y análisis de eventos

Se estudiaron muchas alternativas para el análisis y visualización de los eventos, pero parece que las mejores opciones actualmente, para este caso de uso, serán las basadas en la base de datos Elasticsearch.

Elasticsearch es uno de los proyectos de código abierto mantenidos actualmente por la empresa Elastic, junto con Kibana, Logstash y Beats. Todas estas herramientas se integran muy bien y suelen desplegarse juntas en lo que se conoce como *ELK Stack* [2]. Logstash y Beats recogen y procesan los datos que se guardan en Elasticsearch, mientras que Kibana se instala para servir como motor de visualización de los datos almacenados.

El problema de Kibana es que no tiene sistema de alertas integrado en su licencia Básica. Una licencia de Elastic que incluya el paquete de alertas tiene un precio prohibitivo tanto para la pequeña empresa como para la economía doméstica, por lo que es una opción que queda descartada para este proyecto. Afortunadamente hay otras opciones gratuitas y de código abierto que ofrecen la funcionalidad del sistema de alertas y son fácilmente integrables con ElasticSearch y Kibana. La alternativa más popular y activa es ElastAlert, desarrollada por Yelp, que cuenta además con un plugin, desarrollado por BitSensor, para la integración en Kibana.

Existen además otras alternativas, como Sentinl (desarrollado por Siren), pero son proyectos menos activos y menos populares. En caso de que ElastAlert no satisficiese las necesidades de este proyecto se podrían estudiar más a fondo estas alternativas.

Otra opción SIEM interesante basada en ElasticSearch es Graylog. La gran ventaja de Graylog es que tiene integrado el sistema de alertas y reportes. Tiene también otras funcionalidades interesantes como la gestión de usuarios, que Kibana incluye sólo en su licencia de pago. En cuanto a herramientas de visualización, Graylog parece que va un paso por detrás de Kibana, pero tiene un desarrollo muy activo y ha mejorado notablemente durante el último año. Como punto negativo, Graylog añade MongoDB como dependencia para guardar las configuraciones del usuario. Esto implica una instalación un poco más compleja que la de Kibana.

Los sucesivos análisis contemplados en la fase de investigación permitirán la elección de la herramienta más adecuada.

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

1.7 Recursos y presupuesto del proyecto

Dependiendo de los resultados que vaya arrojando la investigación de las diferentes tecnologías, los recursos y el presupuesto del proyecto pueden variar.

Cabe resaltar la diferencia entre el presupuesto para la investigación y desarrollo de este proyecto, y el presupuesto para implementar la solución final. Pudiera ser que el planteamiento inicial de desplegar el sistema en 3 Raspberry Pi´s 3 fuese erróneo, y fuese necesario adquirir una cuarta, o otros modelos más avanzados (Modelo 4, por ejemplo). En ese caso, el presupuesto para el desarrollo de la investigación incluiría todos los gastos, incluido el hardware que al final no haría falta en la implementación final. El presupuesto para la implementación final sólo incluiría, en este ejemplo, los modelos avanzados.

El objetivo del proyecto es obtener un sistema final de bajo coste, aunque la investigación no sea necesariamente de bajo coste. A continuación se describen los máximos presupuestados para cada recurso, en caso de llegar a aplicarse cada uno.

Ordenador personal Para realizar la investigación sobre el estado del arte, redactar la memoria, instalar y configurar las herramientas en las Raspberry Pi´s.	600 €
Raspberry Pi 3 La predicción inicial es de 3 o 4 unidades para el despliegue de las distintas herramientas.	35 € / unidad
Raspberry Pi 4 de 4 GB En el peor de los casos se prevé que harán falta dos de estos dispositivos, para la base de datos de eventos (Elastic), y otra para el IDS.	65 € / unidad
Switch básico TP-Link TL-SG105 de 5 puertos. Para la interconexión de las Raspberries.	11 €
Switch con port-mirroring TP-Link TL-SG105E de 5 puertos. En caso de que el IDS ralentice la conexión a internet, y se opte por explorar la solución con port-mirroring.	22 €

Conexión a internet Salida a internet que será monitorizada. También necesaria para la etapa de documentación e investigación. Se considerará una duración de 4 meses, para el cálculo del presupuesto total.	30 €/mes
Cables Ethernet Pack de 5 cables ethernet, de categoría 6 y 1 metro de largo cada uno, para las conexiones de las Raspberries.	10 €
Antena Wifi USB TP-Link TL-WN722N, de mayor alcance que la antena integrada en la Raspberry, y con capacidad de trabajar en modo AP.	8 €

De este modo, considerando también los gastos de otros elementos menores como cables USB, adaptadores o tarjetas de memoria, el presupuesto total para el desarrollo del proyecto variará entre 850 € y 1000 € aproximadamente.

Asimismo, se prevé un presupuesto para la implantación del sistema final entre 130 € y 250 € aproximadamente, sobre una conexión a internet ya existente (no incluida en el presupuesto).

1.8 Análisis de riesgos

A continuación, se identifican los mayores riesgos que podrían desviar el proyecto de la planificación especificada en este documento.

Riesgo 1. Objetivo demasiado extenso. Es posible que al especificar el proyecto se haya sobreestimando la cantidad de tiempo disponible o infravalorado la complejidad del problema.

Mitigación: Seguir la planificación temporal de la forma más ajustada posible, y replantearse los objetivos menores si empieza a retrasarse el desarrollo, a fin de completar por lo menos los objetivos principales.

Riesgo 2. Demasiada falta de conocimiento inicial. Puede ser que la falta de experiencia del alumno en el área del proyecto implique una necesidad de tiempo mayor a la prevista a la hora de comprender los conceptos nuevos y las tecnologías que serán utilizadas.

Mitigación: Priorizar el estudio práctico y de los conceptos esenciales para el desarrollo del proyecto. Prestar especial atención a la planificación temporal durante la fase de investigación.

Riesgo 3: Herramientas con curvas de aprendizaje demasiado altas. Las herramientas finalmente escogidas podrán ser las más adecuadas para la solución del problema, pero pueden requerir una cantidad de tiempo inesperado a la hora de aprender su funcionamiento y de configurarlas correctamente.

Mitigación: Informarse de la curva de aprendizaje antes de tomar decisiones sobre las herramientas a utilizar.

Riesgo 4: Conflictos inesperados de integración. Aún cuando las herramientas escogidas sean correctas, es posible que presenten problemas imprevistos a la hora de actuar como un conjunto, o que no permitan respetar las limitaciones de infraestructura impuestas por los objetivos del proyecto.

Mitigación: Realizar pruebas de integración entre las herramientas en la etapa más temprana posible. Considerar alternativas en caso de que una herramienta pueda demostrarse problemática.

Riesgo 5: Hardware demasiado limitado. Es posible que las limitaciones impuestas en la infraestructura para mantener el coste reducido imposibiliten la ejecución del sistema.

Mitigación: Considerar primero topologías con más capacidades y recursos, donde el sistema es funcional, y explorar después las alternativas de menor coste y mayor riesgo.

2. Fase de Investigación

Durante esta fase se llevó a cabo un estudio del estado del arte, llevando a cabo la selección de las herramientas que serán utilizadas en la implementación del sistema final.

2.1 Elección de la herramienta IDS

Durante la fase de planificación se hizo un breve estudio del estado del arte concluyendo que las opciones más interesantes para cumplir los objetivos del proyecto son Zeek y Suricata. Dado que es importante que el IDS se sitúe en la red local para

poder procesar los datos de la red, se comenzó instalando Zeek y Suricata en una Raspberry Pi 3 con la intención de analizar el tráfico local de la misma.

En la Raspberry se instaló Raspbian Buster, que es la versión de Debian Buster preparada para Raspberry, publicada en julio de 2019. Se optó por la versión Lite, que es más ligera ya que no incluye entorno de escritorio, y para el objetivo del proyecto no es necesario.

En los repositorios de Raspbian hay versiones antiguas de Zeek y Suricata. La instalación desde los repositorios es más fácil y rápida, pero las versiones disponibles datan de finales de 2018 y ambos son proyectos bastante activos[3] [4] y han cambiado mucho desde entonces[5] [6]. En el caso de Zeek, en 2018 todavía se llamaba Bro, y la última versión de Suricata incluye opciones muy interesantes para los objetivos de este proyecto. Por ello se escogió instalar las dos herramientas compilando la últimas versiones y siguiendo las instrucciones de instalación del sitio web de cada una. La principal razón para optar por estas últimas versiones es la integración de las reglas y listas negras más actualizadas.

Tras la instalación, y sin tener que configurar casi nada, ambas herramientas son opciones muy válidas y dan bastante información útil del tráfico que pasa por la interfaz. En los dos casos se obtienen logs fácilmente legibles con todas las consultas DNS, peticiones HTTP, intercambio de certificados SSL y más, clasificables por protocolos, y que permiten fácilmente tener una perspectiva de lo que está ocurriendo en la red a nivel de protocolo de aplicación.

Un punto importante a tener en cuenta es la posibilidad y facilidad con la que se pueden integrar listas externas de IPs y dominios relacionados con malware. Existen multitud de este tipo de listas, algunas más actualizadas y completas que otras, que son mantenidas por diversas organizaciones. Algunas de estas organizaciones ofrecen una versión de pago de sus listas, que están más actualizadas y depuradas (detectan menos falsos positivos) que sus respectivas opciones gratuitas. Para la integración de estas listas, existen herramientas que obtienen las listas de IPs, dominios, hashes, etc., de las organizaciones que las mantienen, y las transforman en reglas de detección utilizables directamente por Zeek o Suricata. Algunas de estas herramientas incluso eliminan posibles reglas duplicadas provenientes de diferentes listas. Para Zeek existen herramientas de terceros, como IntelStack[7], que bajo registro ofrece un cliente de descarga de estas listas. Por otro lado, con las últimas versiones de Suricata ya se instala suricata-update[8], que cumple dicha función. Suricata cuenta además con

[3] https://github.com/zeek/zeek/graphs/contributors
[4] https://github.com/OISF/suricata/graphs/contributors
[5] https://github.com/zeek/zeek/releases
[6] https://github.com/OISF/suricata/releases
[7] https://intelstack.com/
[8] https://suricata-update.readthedocs.io/en/latest/

alternativas como pulled-pork[9] o oinkmaster[10] si por cualquier razón suricata-update no cumpliese con los requisitos.

En cuanto a las posibilidades de detección de cada herramienta, Zeek cuenta con su propio lenguaje de programación para expresar cualquier tipo de análisis sobre el tráfico de la red y ofrece una plataforma[11] para que la comunidad pueda compartir sus scripts. Un ejemplo de las posibilidades de análisis de Zeek es la detección de conexiones a servidores CnC (Command and Control) en base al comportamiento del cliente. Un dispositivo conectado a una botnet[12] envía mensajes de baliza al servidor CnC periódicamente para obtener nuevas instrucciones. Estas balizas suelen ser paquetes del mismo tamaño enviados en ciclos de una determinada duración. Con ello, un script de Zeek podría analizar el tráfico de la red y detectar flujos de paquetes con baja varianza en sus tamaños y dirigidos a una misma IP en un rango de tiempo determinado. Esta clase de análisis permite una detección del malware basada en su comportamiento, sin depender de que haya sido incluido algún hash o IP que lo relacione con alguna lista negra.

El lenguaje de reglas de Suricata[13], se acerca más al modelo de detección basado en firmas. Las reglas se escriben con la intención de encontrar patrones en el propio contenido de los paquetes de red. Por ejemplo, la siguiente regla se usa para encontrar una determinada secuencia de bytes en el contenido de un paquete HTTP, y serviría para detectar conexiones establecidas con la herramienta de desarrollo y ejecución de exploits Metasploit[14]:

```
alert http any any -> any any (msg:"msfconsole powershell response";
flow:established; content:!"<html>"; content:!"<script>"; content:"|70
6f 77 65 72 73 68 65 6c 6c 2e 65 78 65|"; http_server_body; content:"|46
72 6f 6d 42 61 73 65 36 34 53 74 72 69 6e 67|"; http_server_body;
classtype:exploit-kit; sid:3016005; rev:1;)[15]
```

Otro ejemplo que demuestra las capacidades de análisis y detección de Suricata es el siguiente par de reglas, que están relacionadas entre sí (mediante una bandera, o flowbit), de modo que la segunda regla no generará una alerta si la primera no fue activada antes:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg: “Registro diciendo
Blah”; content:”userlogin”; flowbit:set, userlogin; flowbit:noalert;)

alert http $HOME_NET any -> $EXTERNAL_NET any (msg: “Registro diciendo
Blah”; flowbit:isset, userlogin; content:”blah”;)
```

[9] https://github.com/shirkdog/pulledpork
[10] http://oinkmaster.sourceforge.net/about.shtml
[11] https://packages.zeek.org/
[12] https://es.wikipedia.org/wiki/Botnet
[13] https://suricata.readthedocs.io/en/suricata-5.0.0/rules/index.html
[14] https://www.metasploit.com/
[15] https://github.com/suricata-rules/suricata-rules/blob/master/Metasploit/metasploit.rules

Con esto se ve que el sistema de reglas de Suricata es también bastante versátil, y además más fácil de aprender que el lenguaje de Zeek. Por otro lado el éxito de Suricata depende más de que las reglas de detección se mantengan actualizadas.

El 18 de octubre de 2019, durante la fase de investigación de este proyecto, fue publicada la versión 5 de Suricata[16]. Esta nueva versión, además de otras muchas mejoras, cuenta con dos nuevas funciones especialmente interesantes. Por un lado cuenta con soporte para JA3S, integrado por defecto. JA3 y JA3S[17] son métodos de fingerprinting (identificación de rastros) del proceso de negociación que ocurre al establecer conexiones seguras SSL/TLS. La combinación de estos dos métodos da la posibilidad de reconocer de forma fiable una conexión cifrada entre un cliente y un servidor específicos, y con ello detectar la comunicación entre software conocido de clientes o servidores de malware, independientemente de las IPs o dominios que utilicen para la conexión. Por otro lado, aunque aún en fase experimental, Suricata 5 facilita la integración de fuentes de datos externas permitiendo añadir nuevas reglas con las palabras clave “dataset” y “datarep”. Estas reglas se pueden utilizar para buscar coincidencias en listas de IPs o de hashes que estén en formato CSV[18]. Esto mejora notablemente las posibilidades de mantener actualizadas las capacidades de detección de Suricata, que sería su punto más débil con respecto a Zeek.

La conclusión es que en general Zeek parece una herramienta más potente y con más posibilidades, pero también más difícil de aprender y configurar que Suricata. Ambas herramientas son perfectamente válidas para la solución del problema, pero considerando que uno de los riesgos analizados durante la planificación del proyecto son las curvas de aprendizaje elevadas, Suricata parece la opción que mejor evitará posibles retrasos.

2.2 Elección de la herramienta SIEM y sistema de alertas

Durante la fase de planificación se determinó que las mejores soluciones SIEM para cumplir con los objetivos del proyecto son Graylog y Kibana, ambas basadas en la base de datos Elasticsearch. Se estudió el estado de cada proyecto en función de su popularidad, facilidad para encontrar documentación y ejemplos, y nivel de soporte y mantenimiento de la comunidad con el proyecto. En todos estos puntos Kibana lleva ventaja sobre Graylog, por lo que se comenzó la investigación orientada a esta herramienta.

La solución de Kibana requiere la instalación de las siguientes herramientas:

Kibana. Servicio web, que permite la búsqueda y creación de visualizaciones sobre los datos almacenados en Elasticsearch.

[16] https://suricata-ids.org/2019/10/15/announcing-suricata-5-0-0/
[17] https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967
[18] https://suricata.readthedocs.io/en/suricata-5.0.0-rc1/rules/datasets.html

Elasticsearch. Base de datos donde se almacenarán todos los datos generados por el IDS. En la fase de optimización podría también considerarse recoger datos de más fuentes, como ficheros de log del sistema operativo.

Logstash. Herramienta mantenida por la empresa Elastic, que recoge, procesa y envía los mensajes del log del IDS a Elasticsearch. Se puede configurar para que complete cada mensaje con información adicional, como geolocalización de cada IP. En caso de necesitar aliviar la carga de trabajo en el nodo del IDS, podría incluso delegarse la tarea de analizar listas negras de IPs y dominios a Logstash.

ElastAlert. Existen varios proyectos con licencia gratuita para completar Kibana con un sistema de alertas, pero ElastAlert parece el más completo y activo de todos. ElastAlert funciona como un proceso independiente que hace consultas periódicas en Elasticsearch. Tiene muchas opciones de configuración para detectar anomalías, como alertas en caso de incrementos bruscos en el volumen de datos, variación de algún valor particular, o aparición de nuevos términos (que podría usarse para detectar nuevas IPs de origen o nuevos puertos usados por un determinado dispositivo). Además ElastAlert tiene soporte para Slack, Telegram y Mail, que eran las opciones preferentes al inicio del proyecto.

Un inconveniente de ElastAlert es que las reglas de detección se configuran en ficheros que deben copiarse al servidor, complicando un poco su mantenimiento. Existe un plugin[19] para Kibana, mantenido por Bitsensor, para poder gestionar estas reglas desde la interfaz web, pero requiere la instalación de su propia versión del servicio de ElastAlert y, en el momento de escribir esto, no está actualizado para ser compatible con las últimas versiones de Elastic (7.x).

Beats. Es otra herramienta mantenida por la empresa Elastic. Sirve para transmitir los mensajes de logs a Elasticsearch, pero sin la capacidad de procesado de Logstash. La ventaja de Beats es que consume menos recursos, por lo que es mejor opción para transmitir el log del IDS al servidor de Elasticsearch. También puede configurarse para enviar los datos a Logstash, y que este haga el procesado antes de guardarlos en la base de datos.

Se consideran dos posibles escenarios: realizar la instalación de estas herramientas en un servidor virtual de DigitalOcean o en una Raspberry Pi 4. Ambas opciones tienen sus ventajas en inconvenientes.

La instalación en DigitalOcean permitiría acceder a los datos a través de Internet, lo cual puede ser ventajoso en caso de recibir una alerta y no estar el administrador conectado a la red local para analizar el incidente. Además permite redimensionar fácilmente el servidor virtual para adaptarlo a los recursos necesarios. Por otro lado, esta solución requiere transmitir los datos del IDS, que contienen mucha información personal, a través de internet. Esto plantea nuevas amenazas de seguridad. Asimismo, uno de los problemas de Kibana es que no tiene ningún tipo de gestión de usuarios en su licencia básica. Tener Kibana disponible a través de internet implica tener que

[19] https://github.com/bitsensor/elastalert-kibana-plugin

configurar por lo menos autenticación básica HTTP, y evitar enviar la contraseña en texto plano mediante TLS. Ambos problemas pueden solucionarse mediante la instalación de un servicio web como Nginx[20], que tiene funciones de proxy reverso, y con ello la capacidad de proteger Kibana haciendo de intermediario ante Internet. Puede configurarse con un certificado auto-firmado (o con uno gratuito expedido por Let's Encrypt[21]) para proteger las comunicaciones.

En el caso de la instalación en la Raspberry, al permanecer los datos en la red local, existe cierto margen de error para poder hacer una primera instalación funcional en un entorno menos hostil, e iterar posteriormente sobre los distintos componentes para solucionar los problemas de seguridad de cada uno. Por el contrario, los recursos en la Raspberry son limitados y esto podría llegar a ser un problema. En caso de encontrarse con que los recursos de la Raspberry no son suficientes existen múltiples soluciones. Por ejemplo, dado que no es necesario que todos los servicios se ejecuten en el mismo hardware, podría adaptarse la topología, instalando Kibana en una Raspberry independiente, o el en mismo PC desde el que se realizan las consultas.

Otro problema con la instalación en Raspberries es que el procesador es de arquitectura ARM, que es mucho menos común que la x86 de la mayoría de las computadoras personales. Esto puede ser conflictivo a la hora de encontrar software compatible o de configurar las aplicaciones para usar los intérpretes adecuados.

Instalación en DigitalOcean

DigitalOcean es actualmente uno de los proveedores de servidores virtuales más populares, y destaca principalmente por su bajo coste y facilidad de uso.

Se creó una máquina virtual de 2 GB de RAM, con Ubuntu 18.04 LTS, en una cuenta de DigitalOcean para uso personal. A continuación se siguieron las guías proporcionadas por Elastic para la instalación de Elasticsearch[22], Kibana[23] y Logstash[24] desde su repositorio oficial. La instalación de las 3 herramientas resultó rápida y sencilla.

Completar la instalación básica de Kibana en DigitalOcean requiere la configuración de Nginx, implementando la autenticación básica de HTTP[25]. Para evitar que la contraseña viaje en texto plano a través de internet, se configuró Nginx para usar el protocolo HTTPS con certificados gratuitos de Let's Encrypt[26].

Para comprobar el funcionamiento general del sistema, se configuró Logstash para leer los logs del servicio de Nginx instalado, y completar cada mensaje con datos de geolocalización de las IPs clientes. Se comprobó a través de Kibana que esos datos se

[20] https://nginx.org/en/
[21] https://letsencrypt.org/
[22] https://www.elastic.co/guide/en/elasticsearch/reference/7.x/deb.html
[23] https://www.elastic.co/guide/en/kibana/7.x/deb.html
[24] https://www.elastic.co/guide/en/logstash/7.x/installing-logstash.html
[25] https://nginx.org/en/docs/http/ngx_http_auth_basic_module.html
[26] https://letsencrypt.org/

almacenan correctamente en Elasticsearch, y se realizaron gráficos y dashboards sencillos.

Por último, se hicieron pruebas con ElastAlert. La instalación de esta herramienta fue mucho más problemática, principalmente a causa de conflictos entre dependencias y versiones incompatibles de python y python-pip. Una vez resueltos todos estos problemas, siguiendo la documentación[27] pudo configurarse una alerta que detecte cuando una nueva IP se conecte a Nginx. Una vez activada la regla se utilizó un servicio de VPN para hacer una conexión desde una IP diferente, y el funcionamiento fue satisfactorio: ElastAlert registra el evento en Elasticsearch y puede consultarse a través de Kibana. Se considerará parte de la fase de implementación la configuración del envío de notificaciones (via email, Slack o Telegram) a partir de las alertas generadas.

Cabe mencionar la prueba realizada con la versión de ElastAlert de Bitsensor[28] para la integración con Kibana. Se pudo comprobar que efectivamente, a fecha de escribir esto, el proyecto está demasiado desactualizado para poder incluirse en la solución.

Instalación en Raspberry Pi 4

El 23 de octubre de 2019, durante la fase de investigación de este proyecto, Elastic publicó la versión 7.4 de Elasticsearch, Kibana y Logstash. Se intentó instalar el entorno usando esta última versión pero no fue posible a causa de incompatibilidades entre las herramientas y las versiones de Java necesarias para poder ejecutarlas en arquitectura ARM.

Finalmente se optó por instalar la versión 7.3.2, la misma versión instalada en el servidor virtual, y que fué publicada pocas semanas antes[29].

La instalación en la Raspberry no resultó tan sencilla, principalmente a causa de las incompatibilidades esperadas con el procesador ARM. Afortunadamente la comunidad de Elastic es bastante activa y se pudo encontrar algún artículo[30] con soluciones para la instalación. A grandes rasgos, la solución pasa por instalar manualmente una versión de Java para ARM, forzar la instalación de Elasticsearch (por defecto se bloquea al detectar un procesador incompatible), y cambiar la configuración para que Elasticsearch use la versión correcta de Java.

La instalación de Kibana en la Raspberry no está exenta de conflictos, resueltos también gracias a soluciones encontradas en los foros de Elastic[31]. En este caso se

[27] https://elastalert.readthedocs.io/en/latest/ruletypes.html#new-term
[28] https://github.com/bitsensor/elastalert
[29] https://www.elastic.co/downloads/past-releases#elasticsearch
[30] https://medium.com/hepsiburadatech/setup-elasticsearch-7-x-cluster-on-raspberry-pi-asus-tinker-board-6a307851c801
[31] https://discuss.elastic.co/t/installing-kibana-on-a-raspberry-pi-4-using-raspbian-buster/202612/6

requiere recompilar algunas de las dependencias de Kibana y sustituir esa compilación por los módulos que se instalan por defecto.

En cuanto a Logstash, los conflictos causados por las incompatibilidades de la versión 7.4 también se resolvieron al cambiar a la versión 7.3.2. Además, los cambios realizados en el sistema operativo para arreglar la instalación de Kibana en la Raspberry, solucionan también la instalación de Logstash.

Conocidas las versiones necesarias de python y python-pip, y la lista completa de dependencias requeridas por ElastAlert, la instalación de esta herramienta en la Raspberry se pudo realizar sin mayor problema.

Instalación de Graylog

Aunque Graylog trabaja con Elasticsearch, necesita también MongoDB para el guardado de las configuraciones. Esto hace la instalación algo más compleja y con más necesidad de RAM en el sistema, que es uno de los recursos más limitados.

Se instalaron MongoDB y Graylog en el servidor virtual, intentando reutilizar la instalación de Elasticsearch utilizada para las pruebas con Kibana. Surgieron nuevamente conflictos con la versión de Java necesaria, que pudieron solucionarse con relativa facilidad gracias a la experiencia obtenida durante las pruebas con Kibana. Lamentablemente, al iniciar Graylog se pudo comprobar que no soporta Elastic 7.x. Habría que sustituir la instalación con Elasticsearch 6.8, publicada en Octubre de 2018, para hacerlo funcionar, lo que implicaría llevar a cabo una investigación totalmente paralela para Graylog. Este cambio haría necesario adaptar el resto de herramientas para encontrar una combinación de versiones funcional (que ha sido también el principal problema con Kibana). Por ejemplo, habría que instalar otra versión de Java, y otra versión de Logstash que fuese compatible con la versión 6.x de Elasticsearch. Adicionalmente, si se considera realizar la instalación en las Raspberries, surgen aún más dudas que requerirían todavía más pruebas. Seguir por este camino implicaría invertir más tiempo en la fase de investigación e incumplir la planificación establecida.

La solución con Kibana cumple con el objetivo de esta fase, que era encontrar una serie de herramientas que pudiesen trabajar juntas para obtener un sistema funcional con los requisitos iniciales. Graylog sigue siendo una opción muy interesante, ya que al integrar sistema de alertas (incluyendo interfaz web para su configuración) y gestión de usuarios, podría ahorrar problemas instalando el acceso con autenticación básica HTTP y la instalación de ElastAlert, necesarios en la solución con Kibana.

2.3 Topología inicial

La experiencia obtenida durante la investigación muestra que la instalación en la Raspberry tiende a ser más conflictiva que en el servidor virtual, pero tiene la gran

ventaja de permitir hacer pruebas en un entorno más seguro, ya que los datos permanecen en la red local. Considerando que estos datos incluirán gran cantidad de información personal (historial de sitios web visitados, servicios utilizados, etc.) evitar estos problemas de seguridad es especialmente importante.

En un primer momento se consideró realizar la instalación completa del sistema en el modelo 4 de Raspberry, pero las mediciones del consumo de recursos realizadas durante la investigación intuyen que aún este modelo más potente podría no ser suficiente al recibir la carga de trabajo de un entorno real. Esta sería una opción interesante ya que permite ahorrar aún más los costes de despliegue, además de simplificar en gran medida la configuración. Se considerará de nuevo esta topología en la fase de optimización.

De este modo, para comenzar la instalación del entorno real se plantea la topología ilustrada en la siguiente figura:

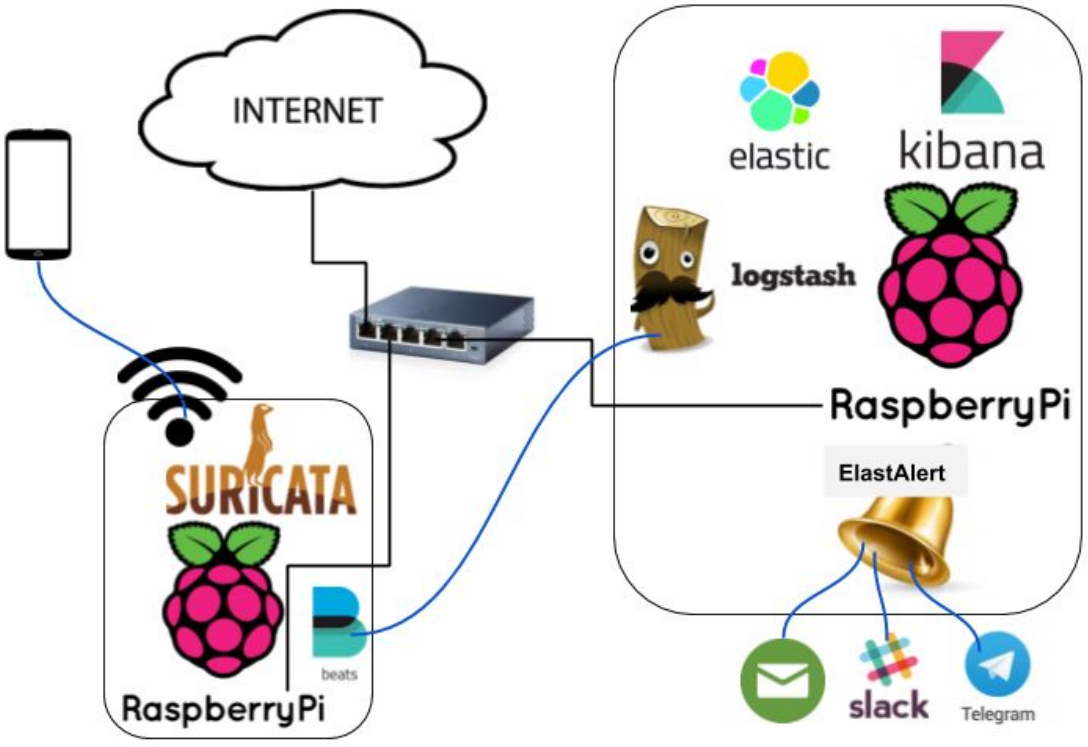


Figura 1: Topología inicial

Se pretende obtener un primer sistema funcional usando dos Raspberries. Se empleará una Raspberry Pi 3 para la instalación del AP, Suricata y Elastic Beats. Como se observa en el diagrama, esta Raspberry tendrá su interfaz ethernet conectada a un router con salida a Internet. Suricata “escuchará” el tráfico de esta interfaz y generará su log de eventos de seguridad. Este log será enviado por Beats a la segunda Raspberry. Logstash recibirá los logs, los procesará, y los almacenará en Elasticsearch, instalado en el mismo hardware. Para esta segunda Raspberry se usará el modelo 4B de 4 GB de RAM, al que además se le ha añadido una SD de 128 GB

que debería satisfacer las necesidades de almacenamiento. Este terminal contendrá también Kibana y ElastAlert, y estará conectado vía ethernet al mismo router.

En caso de que la primera topología se demuestre inviable por falta de recursos, incompatibilidades imprevistas con el procesador ARM, o cualquier otra razón, se plantea una segunda topología que ofrece una mayor garantía de éxito, ilustrada en la siguiente figura:

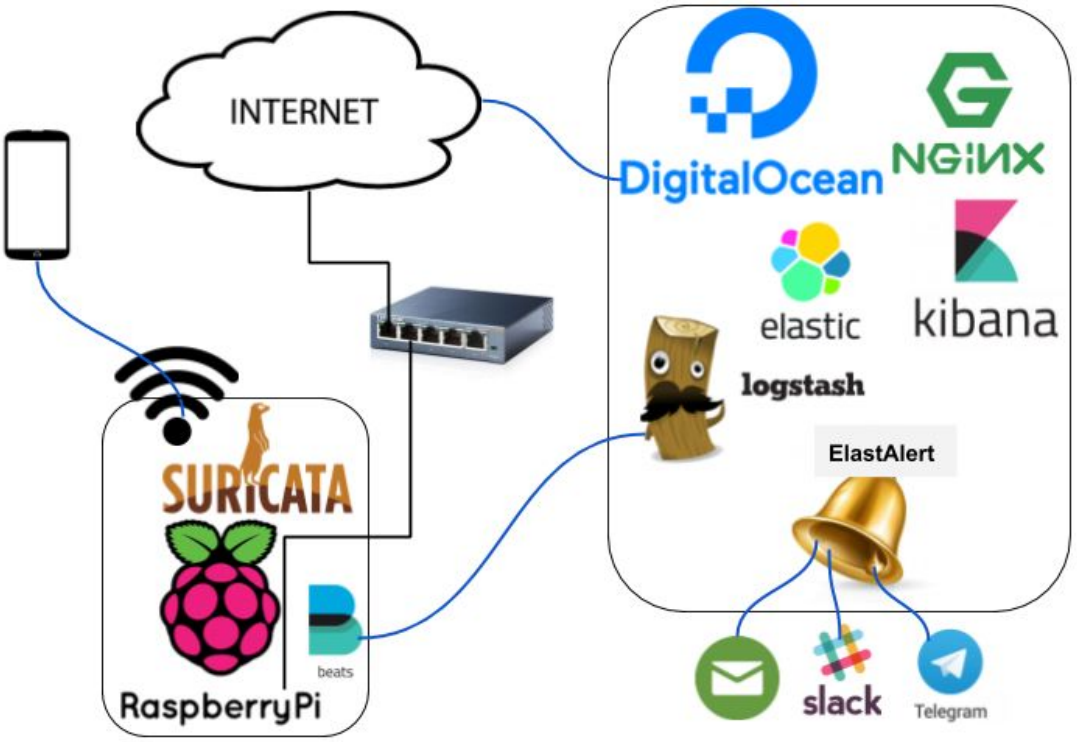


Figura 2: Topología inicial alternativa

En este caso se prescinde de la segunda Raspberry, migrando todos sus servicios a un servidor virtual de DigitalOcean. En caso de falta de recursos con la Raspberry del IDS, también podrá reemplazarse por el modelo 4B, que quedaría vacante en esta solución.

Esta solución tiene tres ventajas evidentes:

- Elimina los conflictos a causa del procesador ARM de Raspberry, facilitando la instalación y configuración de las herramientas.
- En caso de necesitar más recursos, el servidor virtual puede redimensionarse fácilmente sin tener que reinstalar ni configurar nada.
- Kibana queda disponible a través de internet, permitiendo consultar los datos almacenados sin necesidad de estar conectado a la red local. Esto puede lograrse también en la solución con Raspberry, pero requiere una configuración

más compleja y, generalmente, peor calidad del servicio. DigitalOcean garantiza una disponibilidad del 99,99% del tiempo[32].

Por otro lado, entre los problemas de esta solución, habría que garantizar que las conexiones entre la Raspberry con el IDS y el servidor virtual, que ahora se harán a través de internet, se hagan de forma segura (configuración de conexiones cifradas y certificadas). Esto se debe aplicar no sólo a las conexiones web con Kibana, sino también al envío de logs desde Beats a Logstash.

Otro inconveniente es que esta topología tiene un coste más elevado a medio y largo plazo. Un servidor virtual en DigitalOcean de 4 GB de RAM cuesta cerca de 20 euros/mes[33], incluyendo peor CPU y menos almacenamiento. La primera topología habría amortizado los costes de hardware a partir del cuarto o quinto mes de uso.

2.4 Requisitos legales y éticos de la solución propuesta

Desde el 25 de mayo de 2018 se aplica en Europa el Reglamento General de Protección de Datos (RGPD), que es complementado en España con la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (LOPD-GDD)[34]. Este nuevo marco legal de protección de datos plantea gran cantidad de desafíos técnicos para cumplir dicha ley.

Es relevante que la RGPD considera como información personal cualquier dato del usuario que le pueda identificar como individuo, directa o indirectamente. De este modo, las IPs, incluso si son dinámicas, son datos personales si pueden llegar a utilizarse para identificar al usuario, que es el caso de este proyecto ya que pueden relacionarse con sus dispositivos personales. Una web que no tenga medios legales para pedir al ISP la identificación del usuario detrás de la IP, no tendría por qué considerar esa IP como dato personal.

El sistema almacenará de forma temporal datos personales de los usuarios. Cumplir con los estipulado en el RGPD implica:

- Que los usuarios puedan informarse, antes de utilizar el sistema, sobre qué información será recogida, durante cuánto tiempo, cómo será procesada, dónde y cómo será almacenada, y posibles cesiones a terceros.
- Derecho al olvido. El sistema debe incluir mecanismos, o un contacto en la organización, para que los usuarios puedan solicitar el borrado de todos sus datos personales.
- Los usuarios deben tener mecanismos para presentar quejas, dudas, o consultar la información personal almacenada que les concierne.

[32] https://www.digitalocean.com/docs/platform/droplet-policies/
[33] https://www.digitalocean.com/pricing/
[34] https://www.aepd.es/prensa/2018-11-23.html

- La organización debe comunicar a los usuarios cualquier brecha de seguridad en un plazo de 72 horas desde el momento del suceso.
- Los datos que no sean estrictamente necesarios para la prestación del servicio deberán ser eliminados.

La solución propuesta en este proyecto requeriría ciertas mejoras para llegar a respetar los puntos incluidos. La solución con los datos en el servidor virtual complica aún más el cumplimiento de la normativa[35].

Desde un punto de vista ético, podría considerarse que el punto más importante a solucionar sería el de asegurar que los usuarios, antes de comenzar a utilizar el sistema, sean conscientes de que parte de la información de sus comunicaciones será almacenada temporalmente. Para ello se debería presentar al usuario un formulario, en papel o digital (un portal cautivo sería la opción ideal), donde el usuario pueda informarse del uso que se hará de sus datos, y pueda dar su consentimiento.

Otra posible mejora que aumentaría la protección de los datos de los usuarios sería la anonimización o pseudonimización de la información almacenada. Particularmente, podrían reemplazarse los datos de IPs, direcciones MAC, nombres de dispositivos, etc., por sus respectivos hashes o pseudónimos, que eviten relacionar esos datos con sus individuos en caso de brecha de seguridad.

3. Fase de implementación

Durante esta fase se instalaron todas las herramientas escogidas y se configuraron para que cooperen, resultando en un sistema final funcional. A continuación se detalla la instalación y configuración de cada una de las partes.

3.1 Instalación del punto de acceso

La salida a internet del sistema se hace a través de un router con su propio servicio DHCP. Para conectar el punto de acceso con Suricata a la red del router se debe conectar la Raspberry al router mediante un cable ethernet, y se utiliza la interfaz wifi integrada en el hardware de la Raspberry para lanzar el punto de acceso. Después se crea un puente[36] entre ambas interfaces para permitir la comunicación de los dispositivos conectados al punto de acceso con el resto de la red. Para llevar esto a cabo se siguieron los siguientes pasos:

[35] https://strathprints.strath.ac.uk/63134/1/Weir_etal_BAFA2017_Cloud_accounting_systems_the_audit_trail_forensics_and_the_EU.pdf
[36] https://wiki.linuxfoundation.org/networking/bridge

1. Instalación de dependencias:

   ```
   # apt install hostapd bridge-utils
   ```

2. Configuración del punto de acceso. Se debe editar el archivo de configuración del AP `/etc/hostapd/hostapd.conf` añadiendo las siguientes reglas, modificando el nombre de la wifi y la contraseña:

   ```
   interface=wlan0
   bridge=br0
   ssid=Suricata
   hw_mode=g
   channel=7
   wmm_enabled=0
   macaddr_acl=0
   auth_algs=1
   ignore_broadcast_ssid=0
   wpa=2
   wpa_passphrase=suricata_password
   wpa_key_mgmt=WPA-PSK
   wpa_pairwise=TKIP
   rsn_pairwise=CCMP
   ```

   La elección del canal puede llevarse a cabo con la herramienta airodump-ng poniendo la interfaz previamente en modo monitor, para escoger el canal con menos ruido dentro del vecindario.

3. Creación y configuración del puente:

   ```
   # brctl addbr br0
   # brctl addif br0 eth0
   ```

4. Configuración del cliente DHCP para evitar que la interfaz ethernet, conectada al router, obtenga su propia IP, ya que ello imposibilitaría la creación del puente:

   ```
   # echo “\ndenyinterfaces eth0” >> /etc/dhcpcd.conf
   ```

5. Configuración de la interfaz puente (`br0`) en systemd. Para ello se requiere crear los siguientes tres archivos con sus correspondientes contenidos:

   a.
   ```
   # cat /etc/systemd/network/bridge-br0.netdev
   [NetDev]
   Name=br0
   Kind=bridge
   ```

   b.
   ```
   # cat /etc/systemd/network/bridge-br0.network
   [Match]
   Name=br0

   [Network]
   Address=192.168.1.10/24
   Gateway=192.168.1.1
   ```

```
DNS=1.1.1.1
```

c.
```
# cat /etc/systemd/network/uplink.network
[Match]
Name=eth0

[Network]
Bridge=br0
```

6. Activación del servicio de red de systemd:

```
# systemctl enable systemd-networkd
```

7. Activación del servicio hostapd:

```
# systemctl unmask hostapd
# systemctl enable hostapd
# systemctl start hostapd
```

Una vez reiniciada la Raspberry, queda disponible el punto de acceso con el nombre y contraseña especificados en la configuración.

Para un acceso más cómodo al sistema operativo, se activó el servicio SSH, y se realizaron cambios en la configuración (prohibir acceso de root, permitir autenticación únicamente mediante clave privada autorizada y cambiar el puerto por defecto) a fin de fortalecer la seguridad básica del sistema.

3.2 Instalación del IDS y envío de eventos

Dado que la versión de Suricata disponible en los repositorios de Debian es demasiado antigua, se optó por compilar la última versión disponible (Suricata 5). Para ello se siguieron las instrucciones especificadas en la documentación oficial[37], instalando todas las dependencias necesarias antes de ejecutar el proceso de compilación. Además de las dependencias especificadas en la documentación resultó útil la instalación de Rust antes de iniciar la compilación, ya que permite realizar una instalación completa de Suricata, incluyendo suricata-update, para la gestión de reglas.

```
# wget https://www.openinfosecfoundation.org/download/suricata-5.0.0.tar.gz
# tar -zxvf suricata-5.0.0.tar.gz
# cd suricata-5.0.0/
# apt -y install libpcre3 libpcre3-dbg libpcre3-dev build-essential
autoconf automake libtool libpcap-dev libnet1-dev libyaml-0-2 libyaml-dev
zlib1g zlib1g-dev libmagic-dev libcap-ng-dev libjansson-dev pkg-config
libnspr4-dev libnss3-dev liblz4-dev rustc cargo python-distutils-extra
# ./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var
# make
```

[37] https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Debian_Installation

```
# make install-full
```

Una vez completada la compilación se creó un servicio de systemd, para que el sistema operativo inicie Suricata al reiniciar la Raspberry. Este servicio debe configurarse para que ejecute el siguiente comando:

```
/usr/bin/suricata -c /etc/suricata/suricata.yaml -i br0
```

El comando anterior especifica el archivo en el que Suricata debe encontrar la configuración (opción -c), y la interfaz en la que debe ponerse en modo monitor para analizar su tráfico (opción -i).

Con la configuración por defecto, Suricata almacena los eventos en el fichero `/var/log/suricata/eve.json`. En este fichero se guarda en formato JSON una nueva línea por cada evento generado. Para enviar estos datos a Elasticsearch se instaló Filebeat, que es una aplicación muy ligera, desarrollada y mantenida por Elastic, que garantiza que todos los datos quedan almacenados en la base de datos, y en caso de interrupción puede continuar a partir del último evento enviado. Su reducido consumo de recursos lo convierte en la opción ideal para ser instalado en la Raspberry junto con Suricata.

Filebeat no tiene soporte oficial para Raspberry debido a su procesador ARM, pero puede instalarse igualmente si se compila explícitamente para esta arquitectura. El proceso es bastante complejo pero existe un proyecto[38] que automatiza la instalación de las dependencias requeridas. Después, descarga, compila e instala Filebeat, Metricbeat, Packetbeat y Auditbeat en su versión 7.3.2 (la necesaria para la integración con el resto de las herramientas de Elastic instaladas). De este modo, utilizando este proyecto, la instalación se limita a descargar, descomprimir y ejecutar el script incluido:

```
# wget https://github.com/josh-thurston/easyBEATS/raw/master/beats_arm.zip
# unzip beats_arm.zip
# cd beats_arm/
# bash easyBEATS-7.3.2_arm
```

Una vez instalado, se debe modificar la configuración de Filebeat para transmitir el log de eventos de Suricata a Logstash, instalado en la otra Raspberry. Se editó el archivo `/etc/filebeat/filebeat.yml` para que incluya el siguiente contenido:

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
   - /var/log/suricata/eve.json

output.logstash:
  hosts: ["pi4:5044"]
```

[38] https://github.com/josh-thurston/easyBEATS

Nótese que el host de Logstash se identifica como “pi4”. La resolución de este nombre para obtener la IP correspondiente se hace a través del archivo `/etc/hosts`, que contiene la siguiente entrada:

```
192.168.1.102    pi4
```

La configuración de Logstash para recibir y almacenar los datos enviados por Filebeat se trata en el apartado 3.3.

Para finalizar la lista de herramientas instaladas en la Raspberry con Suricata, se configuró Metricbeat para el envío de métricas del sistema y poder consultar en Kibana el consumo de recursos. Para ello, la única modificación necesaria consiste en editar el archivo `/etc/metricbeat/metricbeat.yml` especificando el host y puerto de Logstash:

```
output.logstash:
  hosts: ["pi4:5045"]
```

Después, para que el sistema operativo inicie Metricbeat con cada reinicio de la Raspberry se ejecuta el siguiente comando:

```
# systemctl enable metricbeat
```

Una vez obtenido un punto de acceso funcional, con el tráfico siendo analizado por Suricata, se llevaron a cabo algunas pruebas para comprobar la velocidad de la conexión. Se realizaron varias mediciones usando la web fast.com, que es un servicio creado por Netflix para comprobar la velocidad de la conexión. En la siguiente tabla se muestra la media de los valores obtenidos para cada métrica observada:

Configuración	Descarga (Mbps)	Subida (Mbps)	Latencia sin carga (ms)	Latencia con carga (ms)
Conexión directa al router	31,5	7,4	10,5	135
Conexión al AP sin Suricata	9,1	7,3	11	290
Suricata con reglas ET	10,1	7,3	10,5	590
Suricata con 3 fuentes de reglas activadas	9,1	6,3	12	592

Se puede concluir que el punto de acceso es significativamente más lento que una conexión directa al router. Aunque esto podría estar causado por la calidad de la señal wifi. De todas formas el punto de acceso ofrece anchos de banda de hasta 12 Mbps, que es una velocidad aceptable para un uso doméstico. También puede observarse que al aplicar las reglas de detección, la latencia bajo carga de trabajo aumenta a más del doble.

3.3 Almacenamiento de los eventos en Elasticsearch

En el punto anterior se configuró Filebeat para que envíe los eventos registrados por Suricata a la Raspberry 4, donde se instalaron las tres herramientas de Elastic para el procesado, almacenamiento y visualización de los datos. A continuación, se detalla la instalación y configuración de cada una de las herramientas.

Instalación y configuración de Elasticsearch

Dado que las herramientas de Elastic no tienen soporte para ARM, el proceso de instalación en Raspberry requiere configuraciones adicionales. Elasticsearch se instala con su propio entorno de Java ya incluido, pero este entorno por defecto no funciona en ARM, y es necesario sustituirlo por la versión adecuada, así que se debe empezar instalando la versión necesaria:

```
# apt install openjdk-11-jdk
```

A continuación, se debe descargar y forzar la instalación del paquete oficial de Elasticsearch, ya que por defecto el sistema detectará que no existe una versión compatible con ARM, y detendrá la instalación.

```
# wget
https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.3.2-no-j
dk-amd64.deb
# dpkg -i --force-all --ignore-depends=libc6
elasticsearch-7.3.2-no-jdk-amd64.deb
```

Esto deja el sistema es un estado inconsistente y no permite instalar nada más. Para arreglar la situación se debe editar el archivo `/var/lib/dpkg/status` y encontrar la línea "`Package: elasticsearch`". Se debe cambiar la línea de Status que indica que el estado está a "medio configurar" (`half-configured`), y sustituirlo por "`installed`". La línea debe quedar como sigue:

```
Status: install ok installed
```

También se debe cambiar la línea de dependencias (`Depends`) para que indique lo siguiente:

```
Depends: bash (>= 4.1), lsb-base (>= 4), adduser, coreutils (>=
8.4)
```

Se puede comprobar que todo funciona bien ejecutando:

```
# apt upgrade
```

Después hay que indicar a Elasticsearch la versión de Java que debe utilizar. Para ello se debe modificar el archivo `/etc/default/elasticsearch`, y añadir la siguiente línea:

```
JAVA_HOME=/usr/lib/jvm/java-11-openjdk-armhf
```

Para acabar, se debe dar permiso de escritura de grupo en el directorio de configuración de Elasticsearch para evitar conflictos. Este cambio se realiza con el siguiente comando:

```
# chmod g+w /etc/elasticsearch
```

Existen algunos módulos incompatibles que crean conflictos al iniciar Elasticsearch. Dado que no se necesitan estos módulos para el caso de uso de este proyecto, se optó por desactivarlos. Para ello se debe editar el archivo de configuración de Elasticsearch `/etc/elasticsearch/elasticsearch.yml`, y añadir al final las siguientes líneas:

```
xpack.ml.enabled: false
bootstrap.system_call_filter: false
```

Solo queda iniciar el servicio, y configurar el sistema operativo para que inicie Elasticsearch al reiniciar la Raspberry:

```
# systemctl enable elasticsearch
# systemctl start elasticsearch
```

Se puede comprobar que todo está funcionando correctamente haciendo una consulta HTTP al puerto de Elasticsearch con el siguiente comando:

```
# curl localhost:9200/
```

Instalación y configuración de Logstash

Durante la fase de investigación se llevó a cabo la instalación de esta herramienta usando la versión 11 de Java. A la hora de configurar y ejecutar el servicio esta versión se tornó conflictiva, por lo que se tuvo que repetir la instalación usando Java 8.

Del mismo modo que con el resto de herramientas de Elastic, Logstash no tiene soporte para Raspberry, por lo que deben solucionarse varios conflictos e incompatibilidades durante el proceso de instalación, descrito a continuación.

Se inicia el proceso instalando la version 8 de Java y descargando e instalando la versión 7.3.2 de Logstash de la web oficial:

```
# apt install openjdk-11-jdk
# wget https://artifacts.elastic.co/downloads/logstash/logstash-7.3.2.deb
# dpkg -i logstash-7.3.2.deb
```

Si se inicia Logstash en este momento dará el siguiente error de JFFI:

```
load error: ffi/ffi -- java.lang.NullPointerException
```

FFI (Foreign Function Interface) es un mecanismo que permite a un lenguaje de programación invocar funciones escritas en otro lenguaje distinto. En este caso, Logstash necesita hacer llamadas a funciones escritas en Ruby, desde el lenguaje de programación Java.

Para solucionar este problema en Raspberry hace falta llevar a cabo las siguientes acciones:

1. Compilar Java Foreign Function Interface (JFFI) para Raspberry:

   ```
   # git clone https://github.com/jnr/jffi.git
   # cd jffi/
   # ant jar && ant archive-platform-jar
   ```

2. Reemplazar la nueva compilación por la antigua:

   ```
   # cp -f ~/jffi/build/jni/libjffi-1.2.so
   /usr/share/logstash/vendor/jruby/lib/jni/arm-Linux/libjffi-1.2.so
   ```

3. Completar el archivo JRuby para que reconozca arm-linux como una plataforma válida. Esta es una solución temporal a un error de JRuby que el equipo de desarrollo del proyecto tiene intención de solucionar en las próximas versiones[39].

   Con las siguientes instrucciones se desempaqueta el archivo .jar de JRuby, se crea el archivo `platform.conf`, ausente para la plataforma arm-linux, y se vuelve a empaquetar el .jar original:

   ```
   # JARDIR="/usr/share/logstash/logstash-core/lib/jars"
   # JAR="jruby-complete-9.2.7.0.jar"
   # JRUBYDIR="${JAR}-dir"
   # PLATDIR=
   "META-INF/jruby.home/lib/ruby/stdlib/ffi/platform/arm-linux"

   # cd ${JARDIR}
   # unzip -d ${JRUBYDIR} ${JAR}
   # cd "${JRUBYDIR}/${PLATDIR}"
   # cp -n types.conf platform.conf
   # cd "${JARDIR}/${JRUBYDIR}"
   # zip -r jruby-complete-9.2.7.0.jar *
   ```

   Finalmente se vuelve de dejar el nuevo .jar en la ruta original, y con los permisos originales:

   ```
   # mv -f jruby-complete-9.2.7.0.jar ..
   ```

[39] https://github.com/elastic/logstash/issues/10888

```
# chown logstash:logstash jruby-complete-9.2.7.0.jar
```

Una vez completado el proceso de instalación, el siguiente comando configura el sistema operativo para iniciar Logstash al reiniciar la Raspberry:

```
# systemctl enable logstash
```

A continuación se explica la configuración realizada para que Logstash reciba los datos enviados por Filebeat y Metricsbeat desde la Raspberry con Suricata, y los almacene en Elasticsearch. Los cuatro bloques de configuración detallados a continuación componen el archivo `/etc/logstash/conf.d/beats-pipeline.conf`.

En primer lugar, se configura Logstash para escuchar en el puerto 5044, donde recibe la conexión y transmisión de los datos de Filebeat. Dado que el log de Suricata está en formato JSON, se indica a Logstash que debe “parsear” el JSON antes de almacenarlo en Elasticsearch, y marcar esos registros con el tipo “SuricataIDPS”:

```
input {
  beats {
    port => 5044
    codec => json
    type => "SuricataIDPS"
  }
}
```

Dado que en la Raspberry con Suricata se configuró Metricbeat para el envío de métricas de sistema, también se debe configurar Logstash para aceptar la conexión y transmisión de los datos de este proceso. Considerando que los datos enviados por Metricbeat no requieren la descodificación de JSON ni la etiqueta SuricataIDPS, se escogió un puerto distinto para recibir estos datos:

```
input {
  beats {
    port => 5045
  }
}
```

Durante la implementación del sistema se observó que Suricata no tiene soporte para la incorporación de las direcciones MAC en los eventos. Esto imposibilitaba relacionar cada evento con cada dispositivo conectado al punto de acceso. Este imprevisto pudo solucionarse de la siguiente manera: Los eventos de asignación de IP mediante DHCP contienen tanto la IP asignada como la dirección MAC del dispositivo y, a partir de la asignación de la IP, los eventos incluyen la nueva IP en los campos `src_ip` o `dest_ip`. Con esto, se modificó Logstash para que haga, por cada evento recibido de Suricata que contenga los campos `src_ip` o `dest_ip`, una búsqueda en Elasticsearch del último evento DHCP registrado para esa IP. En caso de éxito, se añade la MAC encontrada al nuevo evento:

```
filter {
  if [src_ip] {
    elasticsearch {
      index => "filebeat*"
      query => "dhcp.assigned_ip:\"%{[src_ip]}\""
      fields => {
        "[dhcp][client_mac]" => "device_mac"
      }
    }
  }

  if [dest_ip] {
    elasticsearch {
      index => "filebeat*"
      query => "dhcp.assigned_ip:\"%{[dest_ip]}\""
      fields => {
        "[dhcp][client_mac]" => "device_mac"
      }
    }
  }
}
```

En el último bloque se configura el host y puerto de Elasticsearch al que Logstash debe enviar los eventos recibidos. Se indica también el nombre del índice bajo el que deben ser almacenados:

```
output {
  elasticsearch {
    hosts => "localhost:9200"
    manage_template => false
    index =>
"%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
  }
}
```

Por último, para iniciar Logstash:

```
# systemctl start logstash
```

Instalación y configuración de Kibana

Igual que con el resto de herramientas de Elastic, Kibana no tiene soporte oficial para Raspberry, por lo que la instalación requiere recompilar ciertas dependencias para hacerlo funcionar.

En primer lugar, debe descargarse la versión 7.3.2 desde la web oficial, y forzar la instalación:

```
# wget
https://artifacts.elastic.co/downloads/kibana/kibana-7.3.2-amd64.deb
# dpkg -i --force-all kibana-7.3.2-amd64.deb
```

A continuación, deben compilarse las dependencias @elastic/nodegit y ctags para la plataforma ARM, y reemplazarlas por las versiones amd64 incluidas por defecto. Para ello, deben descargarse e instalarse las siguientes dependencias necesarias para la compilación:

```
# apt install nodejs npm libkrb5-dev
# cd ~/ && git clone https://github.com/nodegit/nodegit.git && cd
nodegit
# wget
https://github.com/fg2it/phantomjs-on-raspberry/releases/download/v2.1.1-wh
eezy-jessie-armv6/phantomjs
```

Los siguientes comandos realizan la compilación de nodegit para ARM:

```
# export PATH=$PATH:~/nodegit
# chmod -R 777 ~/nodegit
# npm install
```

Una vez compilado, debe sustituirse por la versión por defecto:

```
# cp -rf ~/nodegit/build/Release
/usr/share/kibana/node_modules/@elastic/nodegit/build
# cp -f ~/nodegit/dist/enums.js
/usr/share/kibana/node_modules/@elastic/nodegit/dist
```

A continuación se compila ctags, y se reemplaza por la versión por defecto:

```
# cd ~/nodegit
# npm install ctags
# cp -f ~/nodegit/node_modules/ctags/build/Release/ctags.node
/usr/share/kibana/node_modules/@elastic/node-ctags/ctags/build/ctag
s-node-v64-linux-arm
```

Por último, debe configurarse Kibana para que use la nueva instalación de Node.js. Para ello debe editarse el archivo `/usr/share/kibana/bin/kibana` y añadirse la siguiente línea:

```
NODE="/usr/bin/node"
```

Finalmente, se configura el sistema operativo para iniciar Kibana al reiniciar la Raspberry:

```
# systemctl enable kibana
```

Kibana necesita un par de minutos para iniciarse en la Raspberry, y después queda accesible a través del puerto 5601.

3.4 Pruebas de visualización

Una vez instaladas las tres herramientas de Elastic, puede accederse a Kibana para comprobar que los datos enviados desde Filebeat y Metricbeat se reciben y almacenan correctamente.

En la pestaña “Discover” de Kibana pueden consultarse los documentos que están siendo almacenados. Como se observa en la siguiente imagen, en el índice de Filebeat se están almacenando hasta 50 documentos por minuto:

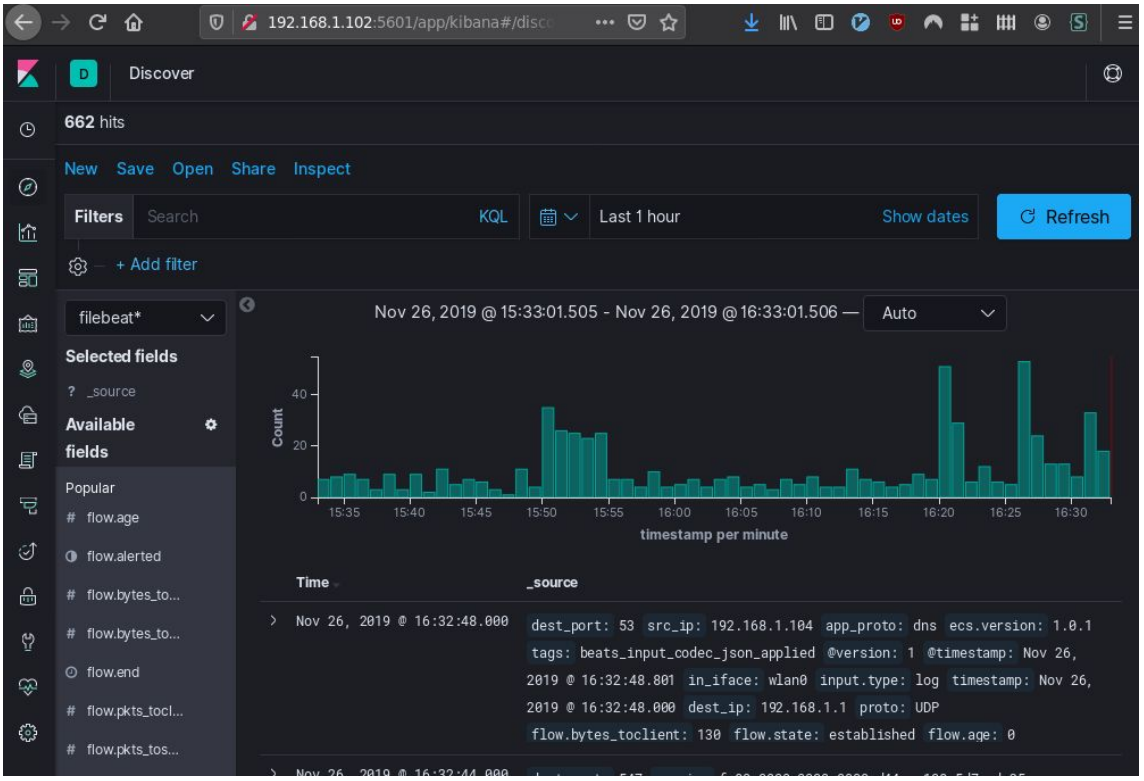


Figura 3: Kibana - Discover

Además se puede verificar que los documentos recibidos contienen el campo `device_mac`, configurado en Logstash para identificar el dispositivo relacionado con cada evento. En la siguiente imagen se observan las dos direcciones MAC de los dos dispositivos utilizados en la pruebas (un teléfono móvil y un ordenador portátil). El gráfico representa el porcentaje de documentos que incluye cada una de las direcciones:

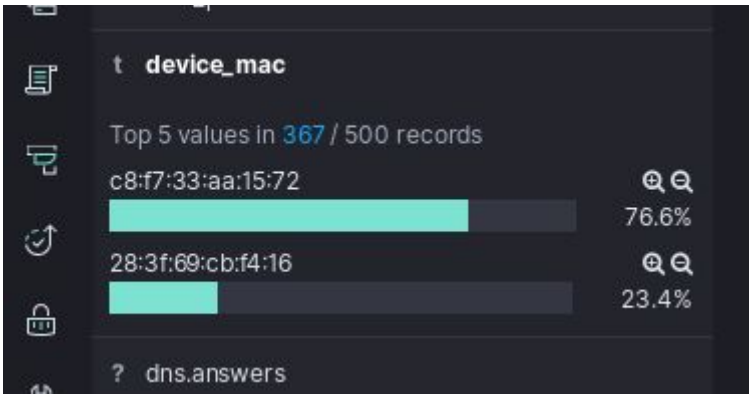


Figura 4: Kibana - Discover :>>device_mac

Con el objetivo de explorar las posibilidades de visualización de datos de Kibana, se elaboraron algunas visualizaciones. En primer lugar se creó una tabla que muestra las alertas de seguridad reportadas por Suricata. Se configuró la visualización para que las alertas se muestren ordenadas por gravedad (Severity), creando una tabla independiente para los datos de cada dispositivo (dirección MAC):

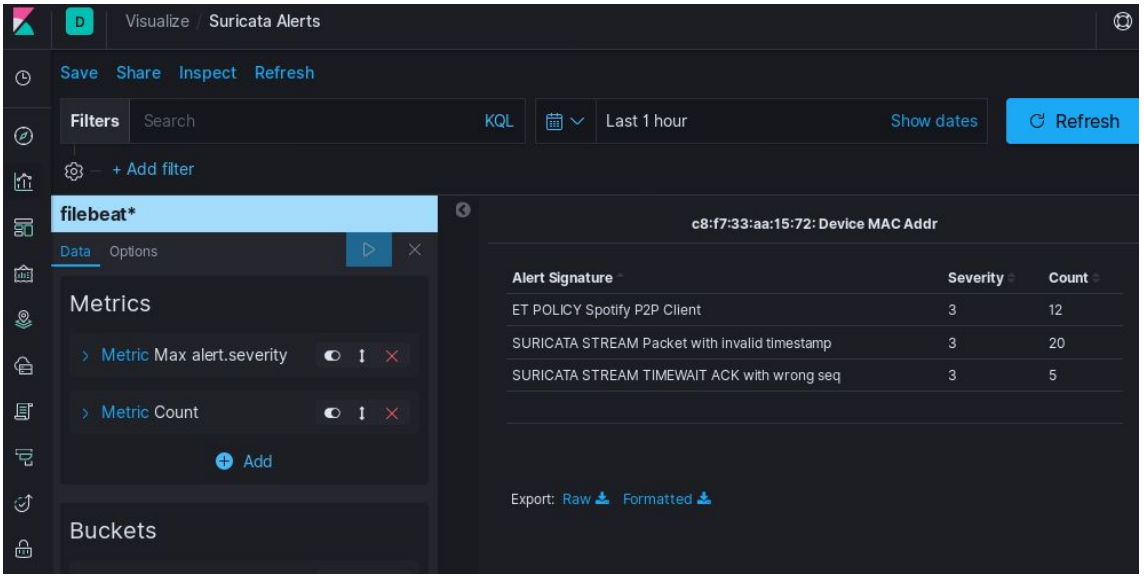


Figura 5: Kibana - Visualización de las alertas de Suricata

También se realizaron pruebas con los datos de estado del sistema, recibidos desde Metricbeat. A continuación, se muestra otra visualización creada a partir de los datos de red almacenados. El gráfico representa la velocidad media de transmisión de entrada (azul) y salida (verde) de la Raspberry con Suricata. Estos datos no expresan necesariamente la actividad de los dispositivos conectados al punto de acceso, ya que incluye también el uso de red del sistema en esa Raspberry:

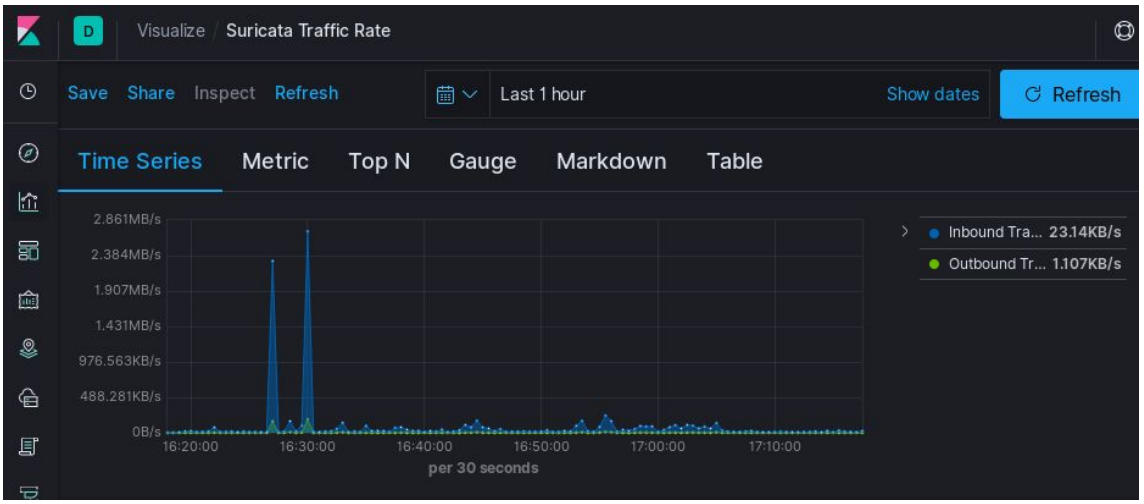


Figura 6: Kibana - Visualización del tráfico en la Raspberry de Suricata

La siguiente figura muestra el panel de configuración de una visualización que descompone el tráfico enviado y recibido (número de bytes) para cada dispositivo conectado (dirección MAC):

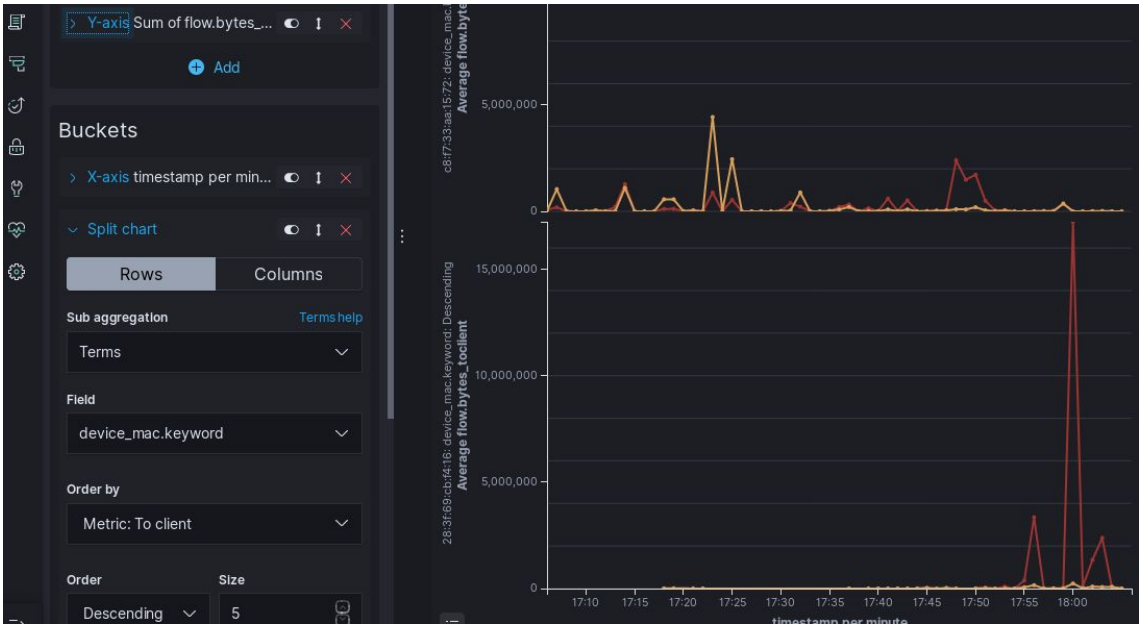


Figura 7: Kibana - Visualización del tráfico de red por dispositivo conectado

Finalmente, se compuso un *dashboard* utilizando algunas de las visualizaciones creadas:

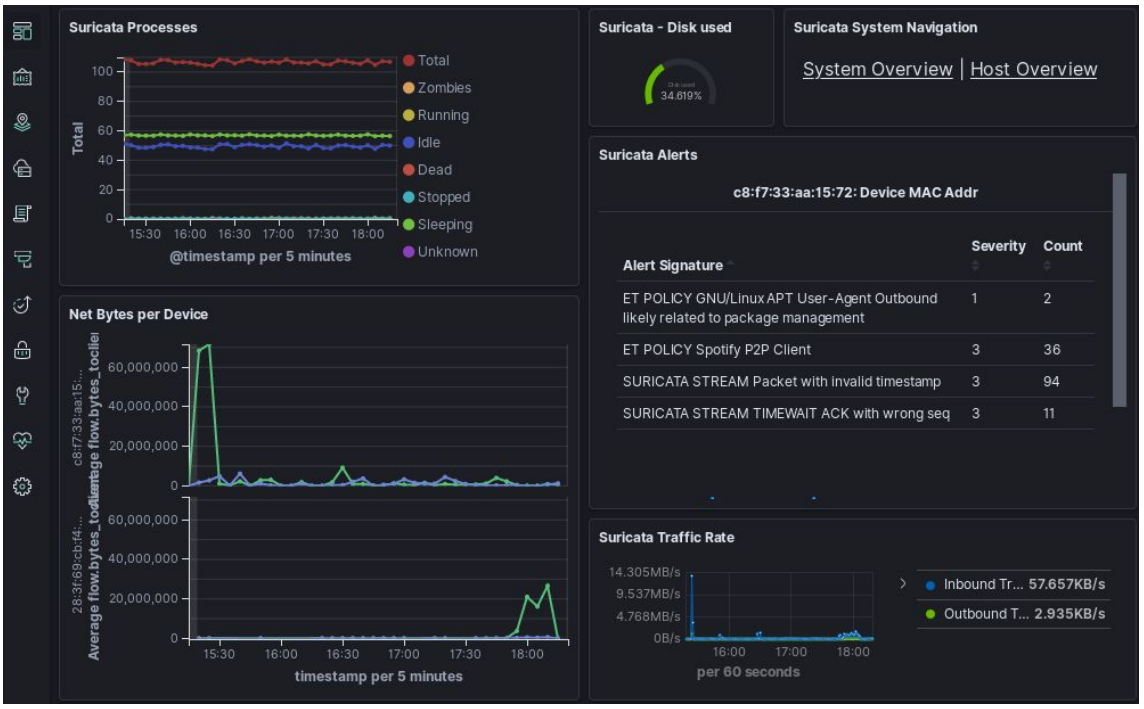


Figura 8: Kibana - Dashboard 1

Como conclusión, Kibana es una herramienta muy completa y ofrece una cantidad enorme de posibilidades. El lado negativo es que muchas de las opciones no resultan muy evidentes, y la curva de aprendizaje es más elevada de lo esperado.

3.5 Sistema de alertas

Se instaló ElastAlert como sistema de alertas complementario a las herramientas de Elastic. Además se crearon un par de reglas de detección sencillas, y se configuró el envío de notificaciones por email y Telegram. A continuación se detalla cada uno de los pasos.

Instalación y configuración de ElastAlert

La instalación de ElastAlert en Raspberry es bastante sencilla, y no difiere de la instalación en el servidor virtual. Es importante asegurar que se usa pip3 para la instalación de setuptools, y Python 3.6 para ejecutar el script de instalación.

Los siguientes cinco comandos instalan ElastAlert y sus dependencias:

```
# apt install build-essential libpq-dev libssl-dev openssl
libffi-dev zlib1g-dev python3 python3-setuptools python3-pip
# pip3 install "setuptools>=11.3"
# git clone https://github.com/Yelp/elastalert.git
# cd elastalert/
# python3.6 setup.py install
```

Una vez instalado ElastAlert, se debe crear el archivo `config.yaml` a partir del archivo de ejemplo `config.yaml.example`. Deben editarse las siguientes directivas:

```
rules_folder: rules
es_host: localhost
```

ElastAlert guarda información y metadatos en Elasticsearch sobre las consultas y las alertas generadas, permitiendo auditarlas, además de dar la posibilidad a ElastAlert de reiniciarse y continuar desde el último registro analizado. El siguiente comando crea los índices de Elasticsearch que ElastAlert necesita para gestionar las alertas. Es importante que el archivo de configuración esté en el directorio actual, para evitar que se abra el diálogo que vuelve a pedir los valores ya especificados en ese fichero.

```
# elastalert-create-index
```

Para comprobar el funcionamiento de ElastAlert, se creó una regla de detección que comprueba cada minuto si existe algún nuevo evento almacenado en Elasticsearch con una MAC no registrada en las últimas 24 horas. De este modo, cada día que se conecte un nuevo dispositivo, se debería enviar una alerta vía email y Telegram. Para ello se generó el archivo `rules/new-mac.yaml`, en el directorio de ElastAlert, con el siguiente contenido:

```
name: New MAC
type: new_term
index: filebeat*
fields:
 - "agent"
terms_window_size:
  days: 1
filter:
- term:
   type: SuricataIDPS
alert:
- "email"
- "telegram"
email:
- "sadoht@gmail.com"
telegram_bot_token: "830731765:FAkETOkEnExaMPleeebFJJioahDl04_HchA"
telegram_room_id: "535453583"
```

La acciones necesarias para hacer funcionar las notificaciones vía email y vía Telegram, configuradas en el bloque anterior, se detallan a continuación.

Configuración de notificaciones vía email

Para el envío de las notificaciones vía email se decidió configurar postfix y enviar los emails a través de Gmail, y con ello evitar mejor los filtros de SPAM. Para lograr esto, debe crearse una cuenta de Gmail, y aplicar autenticación de doble factor. Esto es

necesario para poder obtener códigos de autenticación de aplicación, que son requeridos para la autenticación de postfix con los servidores de Gmail.

Deben instalarse las siguientes dependencias:

```
# apt install postfix mailutils libsasl2-2 ca-certificates
libsasl2-modules
```

A continuación, el archivo `/etc/postfix/main.cf` debe contener las siguientes directivas para la correcta comunicación con los servidores de Gmail:

```
inet_protocols = ipv4
relayhost = [smtp.gmail.com]:587
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_CApath = /etc/ssl/certs
smtpd_tls_CApath = /etc/ssl/certs
smtp_use_tls = yes
```

Como se indica en la directiva `smtp_sasl_password_maps`, las credenciales se especifican en el archivo `/etc/postfix/sasl_passwd`. Se añade el siguiente contenido, que especifica el código de aplicación y la dirección de email obtenidos al crear la cuenta de Gmail:

```
[smtp.gmail.com]:587  bot.elastalert@gmail.com:fake pass goes here
```

Finalmente, se asignan los permisos adecuados al archivo con la contraseña, se carga la configuración en la tabla interna de postfix y se reinicia el servicio:

```
# chmod 400 /etc/postfix/sasl_passwd
# postmap /etc/postfix/sasl_passwd
# systemctl restart postfix
```

Para comprobar que funciona correctamente, se puede ejecutar el siguiente comando, que envía un email a la dirección especificada, con el asunto y mensaje indicados:

```
# echo "Esto es una prueba" | mail -s "Hola" sadoht@gmail.com
```

Configuración de notificaciones vía Telegram

Para el envío de notificaciones a través de Telegram es necesario crear un bot en la plataforma[40]. Para ello basta con buscar el bot The BotFather en la aplicación, y enviarle el mensaje “/newbot”. BotFather preguntará por el nombre que se le quiere dar al nuevo bot, y generará un token de autenticación. Este valor es el que se debe especificar en la configuración de ElastAlert en el campo `telegram_bot_token`.

[40] https://core.telegram.org/bots

Para que el bot pueda comunicarse con un usuario, el usuario debe enviar primero un mensaje al bot. Se llevó a cabo el desarrollado una pequeña aplicación en Node.js para la comunicación con el bot. De esta forma, para cada mensaje enviado al bot, se muestra por pantalla, entre otros datos, el número de identificación del canal entre el bot y el usuario. Este ID es el segundo valor necesario para la configuración del ElastAlert: `telegram_room_id`.

La posibilidad de interactuar con el bot a través de la aplicación creada abre la opción de enviar comandos a través de Telegram cada vez que se recibe una alerta y decidir si, por ejemplo, se quiere bloquear el acceso a internet del dispositivo afectado. Es una opción interesante que podría considerarse como trabajo futurible.

Se configuró otra regla para el envío de notificaciones por cada alerta de seguridad que genera Suricata, pero debido a la cantidad, y a la poca importancia, de los eventos recibidos se optó por eliminar la regla. En la fase de optimización se pretende mejorar la configuración de Suricata para que genere sólo alertas relevantes.

4. Fase de optimización

Durante esta última fase se exploraron diferentes alternativas y mejoras a la implementación obtenida en la fase anterior. A continuación se detalla cada prueba estudiada para cada componente de la solución original.

4.1 Topologías alternativas

Durante esta última fase se exploraron otras dos topologías que ofrecen ciertas ventajas e inconvenientes con respecto a la topología inicial. Se detallan ambas opciones a continuación.

Topología con ELK en la nube.

Durante la fase anterior se llevó a cabo una instalación de las herramientas de Elastic (Elasticsearch, Logstash y Kibana) en un servidor virtual en DigitalOcean. Durante esta fase se configuró temporalmente el entorno real para el envío de los datos desde la Raspberry con Suricata a este servidor virtual.

El principal inconveniente de esta topología era lograr un envío seguro de datos de carácter personal a través de internet. Particularmente para este caso de uso, se debe cifar la conexión entre Elastic Beats, (Filebeat y Metricbeat) y Logstash. Para ello se consideraron las siguientes soluciones:

1. **Uso de túneles SSH**. Los túneles SSH permiten configurar las distintas herramientas para realizar las conexiones a un puerto en localhost, y enlazarlo de forma transparente con otro puerto en otro dispositivo a través de una conexión SSH. Además de su simplicidad, la ventaja de seguridad de esta solución es que evita exponer puertos adicionales a internet.

   Se requieren dos túneles: uno para Filebeat y otro para Metricbeat. Para garantizar que cada túnel permanezca activo se utiliza la herramienta autossh, que realiza una reconexión automática al interrumpirse la conexión activa. Para que autossh pueda restablecer el túnel, debe poder conectarse al servidor sin solicitar una contraseña, por lo que es necesario realizar la conexión utilizando un par de claves.

   Una vez comprobado que se tiene acceso SSH al servidor virtual desde la Raspberry y utilizando la clave RSA, deben ejecutarse los siguientes comandos para crear los túneles:

   ```
   autossh -L 5044:localhost:5044 pi4
   autossh -L 5045:localhost:5045 pi4
   ```

   Estos comandos deben añadirse a un servicio de systemd para que se inicien los túneles al reiniciar la Raspberry. Después, una vez creados los túneles, debe modificarse la configuración de Filebeat y Metricbeat para que estas herramientas envíen los datos a través del túnel creado en localhost en vez de conectarse a la Raspberry Pi 4. Para ello debe modificarse el archivo `/etc/filebeat/filebeat.yml` para que la sección `output` quede de la siguiente forma:

   ```
   output.logstash:
     hosts: ["localhost:5044"]
   ```

   Y del mismo modo para `/etc/metricbeat/metricbeat.yml`, pero cambiando el puerto destinado en Logstash para metricbeat:

   ```
   output.logstash:
     hosts: ["localhost:5045"]
   ```

   Por cuestiones de rendimiento también se consideró la implementación de los túneles usando WireGuard[41], pero finalmente se optó por SSH al ser algo más sencillo de configurar.

   Otra ventaja de la solución basada en túneles es su versatilidad. Por ejemplo, si la conexión al servidor virtual se hiciese únicamente desde dispositivos con clientes SSH con acceso al servidor, podría hacerse otro túnel para poder acceder a Kibana desde localhost. Esto evita exponer Kibana constantemente a internet, además de simplificar la implementación del sistema al no necesitar

[41] https://www.wireguard.com/

configurar ni Nginx ni los certificados de Let's Encrypt utilizados en las fases anteriores.

2. **Autenticación mutua con SSL**. Logstash y Elastic Beast ofrecen soporte para autenticación mutua con certificados. Para ello deben generarse una autoridad certificadora, certificados y claves pública y privada para cada una de las partes. Después deben copiarse los archivos generados a los dispositivos y modificar las configuraciones de las herramientas para generar conexiones seguras.

   Aunque esta solución ofrece un buen nivel de seguridad para la comunicación entre Beats y Logstash, se desestimó para la implementación por ser menos versátil y más difícil de configurar con respecto a la opción basada en túneles.

Como se estudió anteriormente, esta topología tiene las ventajas de poder incrementar fácilmente los recursos disponibles en función de las necesidades, y de estar más fácilmente accesible a través de internet, pero a un coste más elevado a medio y largo plazo.

Topología con Suricata en Raspberry Pi 4

Analizando el consumo de recursos en la topología inicial (con dos Raspberries), se observó que la Raspberry con Suricata tiene su funcionalidad y rendimiento limitados por la cantidad de memoria RAM. Por otro lado, la Raspberry Pi 4, que aloja las herramientas de Elastic, dispone de más memoria libre que toda la cantidad disponible en el modelo 3.

Para esta última topología estudiada se instalaron el punto de acceso y Suricata en la Raspberry 4 de la misma forma que en la fase anterior, y se configuró Logstash para acceder directamente al log de Suricata. De este modo, no se requiere abrir los puertos para Filebeat y Metricbeat, y se especifica un solo `input` para el fichero de log de la siguiente forma:

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

Mover Suricata y el punto de acceso a la Raspberry Pi 4 tiene varias ventajas:

- Reducción del coste de la implementación, al poder prescindir de la mitad del hardware (Raspberry 3, tarjeta SD, alimentación y cable de red). Es una solución especialmente interesante para los casos de uso donde el presupuesto de la implementación juega un papel importante.

- Simplificación de la configuración, al disponer todas las herramientas en localhost, eliminando además algunos de los problemas de seguridad que surgen de las comunicaciones en red.
- Simplificación de la puesta en producción ya que sólo es necesario conectar un dispositivo.
- Reducción de las necesidades generales de recursos. En este escenario no se requiere Filebeat, ya que Logstash puede acceder directamente el log de Suricata. Metricbeat tampoco es necesario ya que las métricas de sistema pueden obtenerse desde Kibana. También se ahorran los recursos destinados al sistema operativo de la Raspberry que deja de usarse.

A continuación se muestra un análisis de la velocidad de conexión, tal y como se hizo con la topología inicial:

Configuración	Descarga (Mbps)	Subida (Mbps)	Latencia sin carga (ms)	Latencia con carga (ms)
Conexión directa al router	35	29	7	345
Conexión al AP sin Suricata	8,9	8,1	7	590
Suricata con reglas ET/JA3	8,2	6,8	7	604

Se observa que los valores son bastante similares a los obtenidos en la fase anterior, por lo que esta solución ofrece casi las mismas prestaciones con un coste más reducido. Además, la Raspberry todavía cuenta con un 25% de memoria disponible, que podría utilizarse para la activación de más reglas de detección de Suricata.

El principal inconveniente encontrado para esta solución es que Logstash dispara el consumo de CPU, haciendo el rendimiento algo más inestable.

4.2 Relación entre dispositivo y usuario

En la fase anterior se implementó una solución que permite incluir las direcciones MAC de los dispositivos conectados en los eventos generados por Suricata. Esto permite clasificar y filtrar los distintos eventos para cada dispositivo. Uno de los problemas de esa solución es que las direcciones MAC no son fáciles de recordar para que el administrador pueda asociar cada MAC con su dispositivo y con el usuario de este. En esta fase se implementó una mejora a esa solución para poder añadir un nombre de dispositivo y un nombre de usuario a cada MAC detectada.

Para llevar esto a cabo, se continuó con la implementación en Node.js del bot de Telegram para que haga lo siguiente:

1. Conectarse al índice de eventos de Suricata almacenados en Elasticsearch, y hacer una petición cada minuto que compruebe si existen eventos registrados en el último minuto con una dirección MAC, pero se desconozca el nombre del dispositivo o del usuario.

2. En caso de detectar una MAC desconocida, comprueba que no se haya enviado una notificación relacionada con esa MAC en los últimos 15 minutos. De no ser así, realiza una consulta HTTP a una API de terceros para identificar el fabricante del dispositivo a partir de la MAC. Después envía un mensaje a través del bot de Telegram al canal establecido con el administrador. Este mensaje contiene la dirección MAC detectada, el nombre del fabricante, y dos botones: uno para indicar el nombre del dispositivo y el otro para indicar el nombre del usuario. En la siguiente figura se observa un ejemplo de estos mensajes en los que el bot ha detectado dos dispositivos conectados:

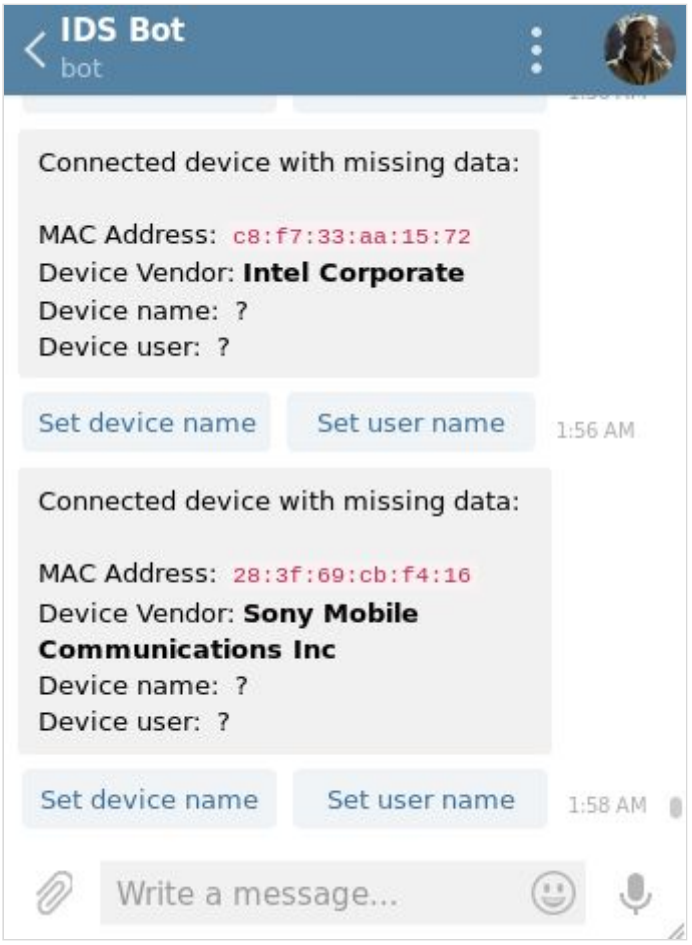


Figura 9: Mensajes recibidos al detectar dispositivos desconocidos

3. Cuando el administrador recibe el mensaje puede seleccionar una de las opciones disponibles. Entonces Telegram notificará al bot la opción escogida y se envía de nuevo un mensaje al administrador pidiendo el nombre de usuario

o de dispositivo que se quiere asociar con la MAC en cuestión. La siguiente figura muestra un ejemplo de esta funcionalidad:

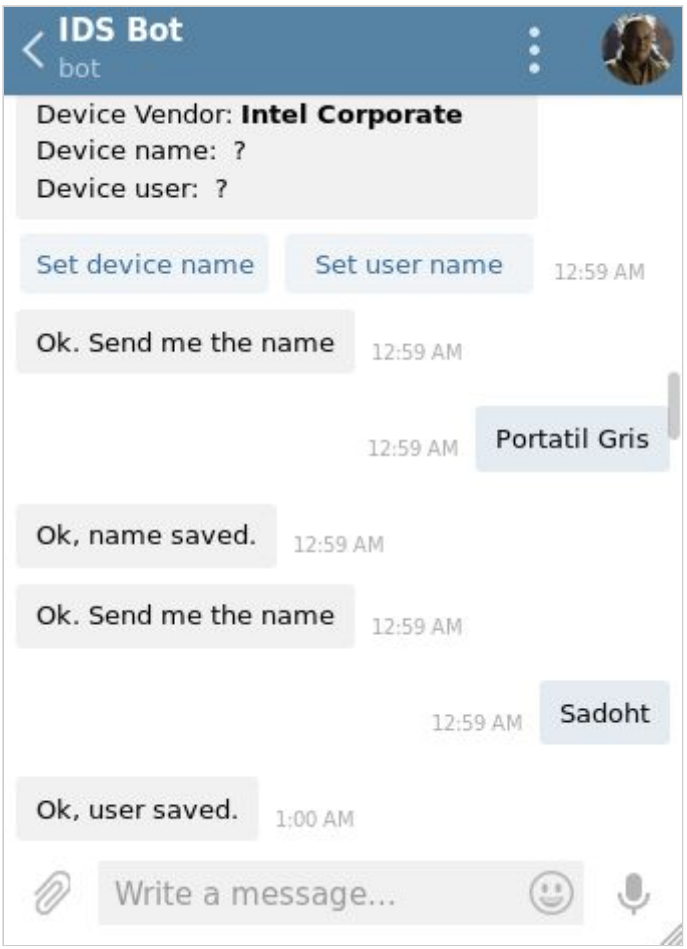


Figura 10: Bot de Telegram guardando nombres de usuario y dispositivo

4. Cuando el bot recibe un nombre de usuario o de dispositivo para una dirección MAC guarda esa relación en un índice de Elasticsearch creado especialmente para estos datos. Esto permite cambiar la configuración de Logstash para que haga una consulta a este índice por cada evento de Suricata que contenga una dirección MAC, y complete los datos antes de almacenar el evento en Elasticsearch.

Dado que cualquier usuario de Telegram puede enviar mensajes a cualquier bot de Telegram se comprueba que todos los mensajes recibidos provengan del canal con el administrador, en caso contrario también se reportará el mensaje recibido.

Todo esto permite mejorar las visualizaciones de forma que se muestre el nombre del dispositivo en vez de la dirección MAC, permitiendo reconocer fácilmente el dispositivo. Además en caso de tener varios dispositivos por usuario, se podrán hacer otros análisis del uso de la red de un individuo a través de todos sus dispositivos, ayudando a la detección de anomalías.

Con este fin se compuso el siguiente dashboard de Kibana, que permite visualizar las alertas de Suricata y el uso del punto de acceso que hace un usuario, incluyendo todos sus dispositivos. La siguiente figura muestra un esquema general del panel, que se compone de las siguientes visualizaciones:

1. Filtro de usuario. Se almacena como parte del panel, lo que permite crear un panel para cada usuario simplemente cambiando el filtro.
2. Tráfico por dispositivo. Muestra un histórico de la cantidad de bytes enviados y recibidos a lo largo del tiempo configurado.
3. Conexiones por Traffic ID. Se presenta el número de conexiones realizadas a cada identificador reconocido en las reglas Traffic ID de Suricata a lo largo del tiempo.
4. Alertas de Suricata. Tabla con las alertas generadas por Suricata, indicando el número de ocurrencias de cada uno.
5. Conexiones TLS. Tabla con el número de conexiones TLS a cada SNI[42].
6. Consultas DNS. Gráfica de pastel que muestra los nombres de dominio más solicitados vía DNS.
7. Tráfico por IP. Gráfica de pastel que muestra las direcciones IP con más tráfico (en bytes).

Figura 11: Esquema del dashboard de usuario

[42] https://en.wikipedia.org/wiki/Server_Name_Indication

Nótese que todas las visualizaciones crean un gráfico o tabla independiente para cada dispositivo asociado al usuario seleccionado, lo que facilita mucho el reconocimiento de conexiones anómalas.

Por razones de legibilidad se incluyen las dos siguientes figuras, que muestran el panel en más detalle:

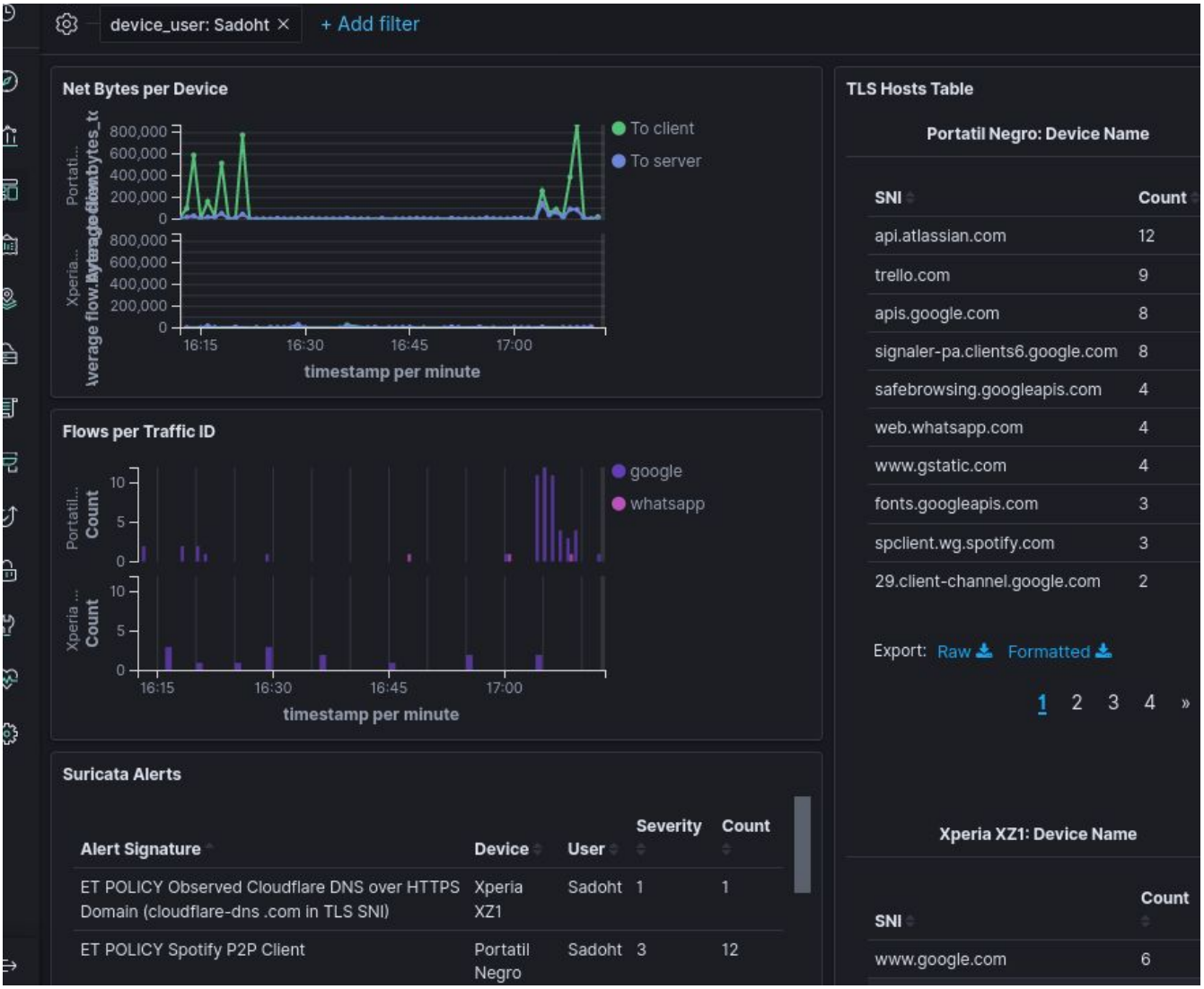


Figura 12: Dashboard con perfil de usuario (parte 1)

Figura 13: Dashboard con perfil de usuario (parte 2)

En las figuras puede observarse un usuario (Sadoht) con dos dispositivos registrados (Portatil y Xperia XZ1). Para cada dispositivo se muestra el uso que hace de la red, direcciones IP y dominios que más conexiones reciben, y la cantidad de alertas de cada tipo generadas por Suricata.

Este dashboard puede clonarse para cada usuario nuevo al que se permita el uso del punto de acceso (compartiendo con él la contraseña), aplicando a cada dashboard el filtro que relaciona los eventos mostrados con el usuario asociado.

Para reflejar ese comportamiento se conectó un nuevo dispositivo al punto de acceso. Al recibir la notificación a través del bot, se le asignaron nombre de dispositivo (Portátil) y usuario (Elena). La siguiente figura muestra la conversación con el bot:

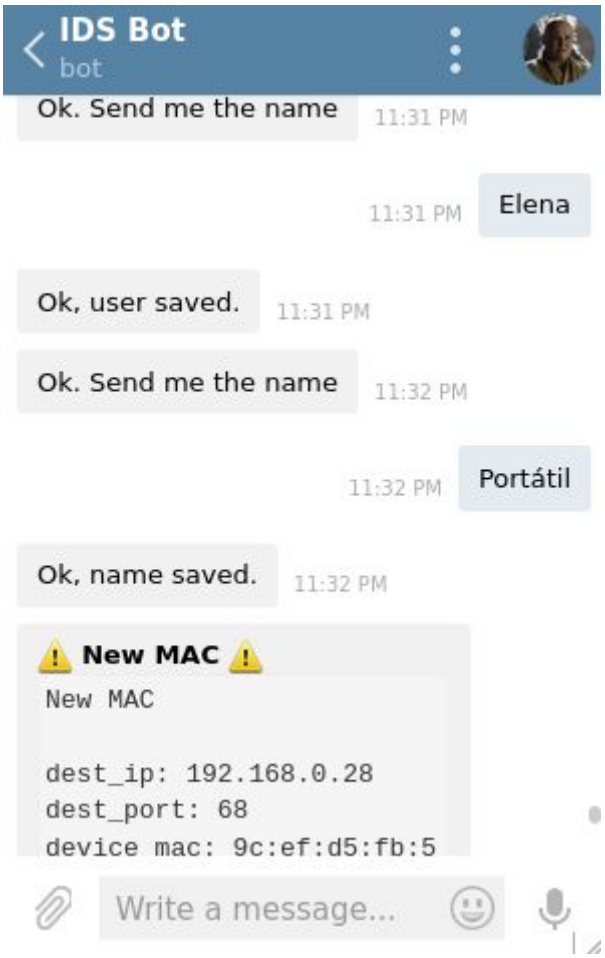


Figura 14: Bot registrando un usuario nuevo

A continuación se clonó el dashboard de usuario *Sadoht* y se cambió el filtro de los eventos para mostrar la información del usuario Elena. A continuación se muestra como ambos dashboards quedan disponibles en las sección *Dashboards* de Kibana:

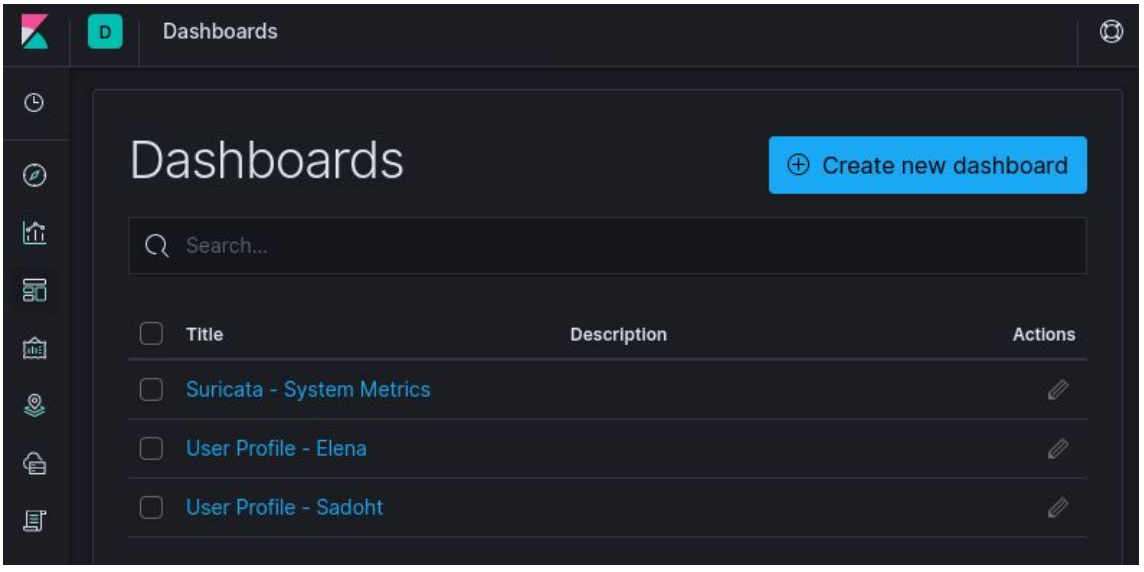


Figura 15: Directorio de dashboards creados en Kibana

La siguiente figura muestra el nuevo dashboard con su filtro correspondiente:

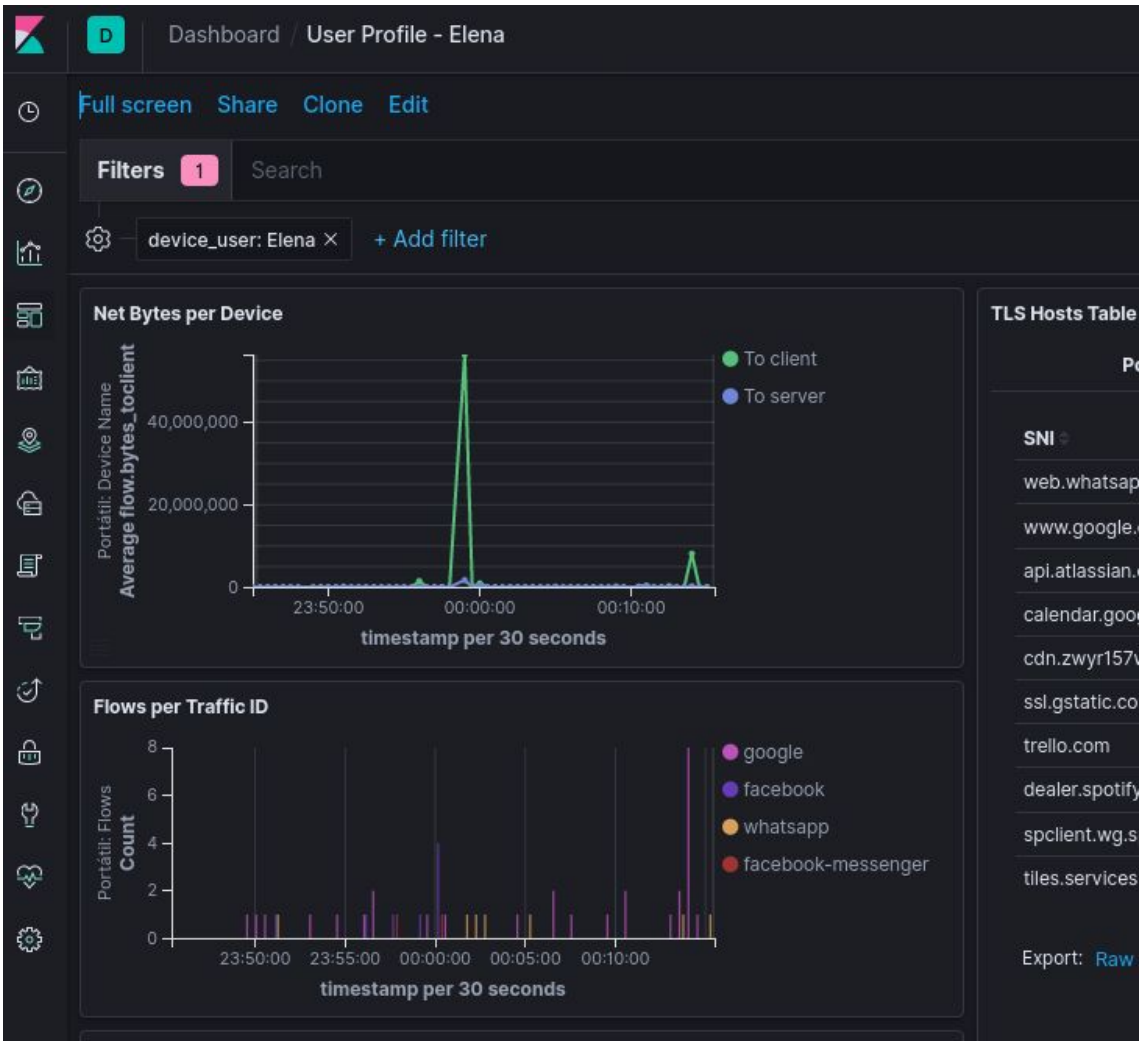


Figura 16: Dashboard con filtro para el nuevo usuario

El bot se ejecuta en la Raspberry Pi 4, donde ya está instalado Node.js para Kibana y el sistema operativo fue configurado para iniciar el bot cada vez que se reinicie la Raspberry. El código fuente del bot puede encontrarse adjunto en el *Anexo 1* de este documento.

4.3 Optimización de las reglas activas en Suricata

Durante esta fase se estudiaron más en profundidad las distintas reglas disponibles para Suricata, y se realizaron algunas pruebas para comprobar su funcionamiento. Este estudio se detalla a continuación.

Selección de las reglas

Como se vió durante la fase anterior, la herramienta suricata-update ofrece la posibilidad de activar y actualizar fácilmente conjuntos de reglas de distintas fuentes. Los recursos limitados de la Raspberry Pi 3 impedían la activación simultánea de varios de estos conjuntos. Durante esta fase, la implementación de la topología alternativa donde Suricata se ejecuta en la Raspberry Pi 4, al disponer de más memoria RAM, permitió la activación de más reglas.

Se realizó un estudio de los distintos conjuntos de reglas, y se decidió una combinación que se adapta a los recursos disponibles. Cabe mencionar que suricata-update integra algunos conjuntos de reglas con licencia comercial, que se desestimaron por razones de presupuesto. A continuación se enumeran los conjuntos de reglas activados y la razón por la que se consideran beneficiosos para este caso de uso:

- Emerging Threats Open Ruleset. Es el conjunto de reglas que se activa por defecto, además de ser el más completo dentro de las opciones que no requieren licencia de pago. Incluye reglas de detección muy variadas, desde detección de escaneos, conexiones a IPs de la red Tor, o tráfico generado por malware conocido.

- Suricata Traffic ID ruleset. Es un pequeño conjunto de reglas que identifica conexiones a servicios populares (Google, Twitter, Netflix), y añade etiquetas a los eventos con el identificador de cada uno.[43]

- Abuse.ch Suricata JA3 Fingerprint Ruleset. Lista actualizada cada cinco minutos que incluye firmas JA3 para la detección de *handshakes SSL* maliciosos.[44]

- Abuse.ch SSL Blacklist. Actualizada también cada cinco minutos, mantiene una lista de firmas de certificados SSL distribuidos por servidores maliciosos, principalmente servidores Command&Control pertenecientes a botnets.[45]

- Positive Technologies Attack Detection Team ruleset. Estas reglas detectan ataques que explotan vulnerabilidades nuevas y *0-days*. El equipo de Positive Technologies mantiene un repositorio con las reglas de detección organizadas por código CVE.[46]

Por otro lado, dada la gran cantidad de alertas generadas de tipo SURICATA STREAM, se optó por desactivar estas reglas. Para desactivar reglas con suricata-update debe editarse el archivo `/etc/suricata/disable.conf`, y añadir el siguiente contenido:

[43] https://github.com/OISF/suricata-trafficid/blob/master/rules/traffic-id.rules
[44] https://sslbl.abuse.ch/blacklist/ja3_fingerprints.rules
[45] https://sslbl.abuse.ch/blacklist/sslblacklist.rules
[46] https://github.com/ptresearch/AttackDetection

```
group:stream-events.rules
```

Después debe ejecutarse de nuevo `suricata-update`, para regenerar el listado de reglas activas, y finalmente reiniciar Suricata para que aplique el nuevo listado.

Pruebas de detección

Una vez activadas las reglas seleccionadas, se hicieron algunas pruebas de acciones que podrían resultar sospechosas para comprobar la detección de Suricata.

En primer lugar se utilizó la herramienta nmap para hacer un escaneo de dispositivos y puertos abiertos en la red local. Si se examina el archivo de reglas de detección de escaneos[47], se observa que la mayoría de las alertas se activan únicamente si la conexiones provienen de la red externa. Para poder generar alertas haciendo el escaneo desde la red local debe modificarse la configuración de Suricata de modo que incluya la siguiente declaración:

```
vars:
  address-groups:
   HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
   EXTERNAL_NET: "any"
```

Después puede usarse el siguiente comando para hacer un escaneo completo, intentado detectar el sistema operativo y versiones de los servicios de los equipos detectados:

```
# nmap -A -T4 192.168.0.1/24
```

La siguiente figura muestra las alertas generadas por Suricata tras la finalización del escaneo:

[47] https://rules.emergingthreats.net/open/suricata/rules/emerging-scan.rules

Alert Signature	User	Severity	Count
ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	Sadoht	1	68
ET SCAN Possible Nmap User-Agent Observed	Sadoht	1	68
ATTACK [PTsecurity] Apache2 <2.2.34 <2.4.27 Optionsbleed (CVE-2017-9798) Attempt	Sadoht	2	1
ET SCAN NMAP OS Detection Probe	Sadoht	2	43
ET SCAN Potential SSH Scan	Sadoht	2	2
ET SCAN Potential SSH Scan OUTBOUND	Sadoht	2	9
ET SCAN Potential VNC Scan 5800-5820	Sadoht	2	2
ET SCAN Potential VNC Scan 5900-5920	Sadoht	2	3
ET SCAN Suspicious inbound to MSSQL port 1433	Sadoht	2	5
ET SCAN Suspicious inbound to Oracle SQL port 1521	Sadoht	2	5
ET SCAN Suspicious inbound to PostgreSQL port 5432	Sadoht	2	5
ET SCAN Suspicious inbound to mySQL port 3306	Sadoht	2	5
GPL RPC portmap listing TCP 111	Sadoht	2	2
ET POLICY Spotify P2P Client	Sadoht	3	12
GPL NETBIOS SMB-DS IPC$ share access	Sadoht	3	3

Figura 17: Tabla de alertas de Suricata tras escaneo con nmap

Se pueden observar numerosas alertas de tráfico sospechoso a los puertos por defecto de servicios populares como PostgreSQL o VNC, detección del *User-Agent* de nmap analizando los puertos de servicios web, intentos de detección del sistema operativo, un posible ataque a una vulnerabilidad conocida, etc.

A continuación se comprobó el nivel de actualización de las listas de IPs de la red Tor, y si Suricata alerta de conexiones realizadas a través de dicha red. El siguiente comando hace una petición HTTP a través de la red Tor:

```
torsocks curl -I https://www.uoc.edu
```

Se realizaron múltiples peticiones a diferentes sitios web. Como se puede observar en la siguiente figura, Suricata reporta alguna conexión, pero la mayoría pasaron inadvertidas:

Alert Signature	User	Severity	Count
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 360	Sadoht	2	1

Figura 18: Detectando conexiones a Tor con torsocks

Después se realizaron conexiones a través de torchat y torbrowser, que ya resultaron más ruidosas. La siguiente figura muestra las alertas generadas tras abrir ambas herramientas:

Alert Signature	User	Severity	Count
ET POLICY TLS possible TOR SSL traffic	Sadoht	3	2
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 660	Sadoht	2	1
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 551	Sadoht	2	1
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 548	Sadoht	2	1
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 509	Sadoht	2	1
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 348	Sadoht	2	1
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 310	Sadoht	2	1
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 301	Sadoht	2	1
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 293	Sadoht	2	1
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 209	Sadoht	2	1
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 152	Sadoht	2	1
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 149	Sadoht	2	1
ET TOR Known Tor Relay/Router (Not Exit) Node Traffic group 125	Sadoht	2	1

Figura 19: Detectando conexiones a Tor con torchat y torbrowser

Durante las pruebas realizadas en esta fase, pudo observarse que las reglas de Suricata necesitan actualizarse cada dos o tres días para mantenerse relevantes, (especialmente las reglas dependientes de listas de IPs). Para ello se modificó el archivo `/etc/crontab` añadiendo la siguiente línea para actualizar las reglas durante la noche:

```
30 6     * * *   root/usr/bin/suricata-update
```

De este modo suricata-update será ejecutado todos los días a las 6:30am.

Por último se hizo una visualización que muestra el número de conexiones realizadas agrupadas por cada identificador incluido en las reglas de `Suricata Traffic ID`:

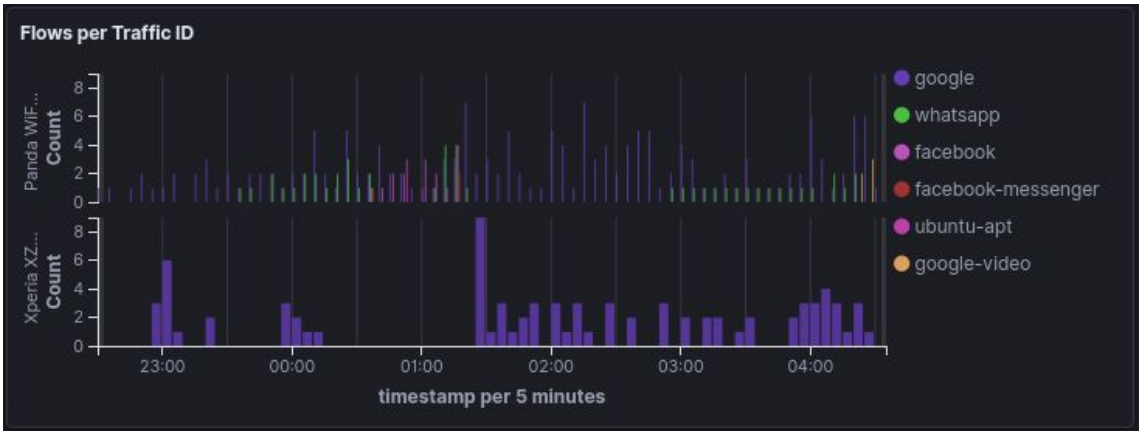


Figura 20: Conexiones para cada Traffic ID

Se puede observar como el primer dispositivo (un ordenador portátil) realiza conexiones a varios de los servicios reconocidos por las reglas de Traffic ID, mientras que el segundo dispositivo (teléfono Android), se mantiene realizando conexiones periódicamente a Google.

4.4 Mejora de las notificaciones

En la fase anterior se configuraron un par de reglas de ElastAlert para comprobar que las notificaciones vía email y Telegram funcionan correctamente. En esta fase se implementaron otras alertas interesantes y se exploraron un poco más las posibilidades de esta herramienta.

Modificación del contenido de las notificaciones

En primer lugar, se formateó el contenido de las alertas creadas en la fase anterior. Por defecto, las alertas de ElastAlert contienen todos los campos del documento almacenado en Elasticsearch, incluyendo metadatos como nombres de host o identificadores internos que son irrelevantes para la notificación. Para ello se utilizó el módulo de mejoras que incluye ElastAlert, que permite definir una función en Python que recibe los datos de la alerta, y modifica las alertas antes de ser enviadas.

Concretamente, para modificar la alerta de nueva MAC detectada eliminando los campos irrelevantes se creó el archivo `clean_new_mac_event.py` con el siguiente contenido:

```
from elastalert.enhancements import BaseEnhancement

class CleanNewMACEvent ( BaseEnhancement ) :

   def process (self, match):
         match.pop('_type', None)
         match.pop('_id', None)
         match.pop('_index', None)
         match.pop('@timestamp', None)
         match.pop('@version', None)
         match.pop('type', None)
         match.pop('event_type', None)
         match.pop('tags', None)
         match.pop('input', None)
         match.pop('agent', None)
         match.pop('ecs', None)
         match.pop('host', None)
         match.pop('log', None)
```

Es importante crear también el archivo `__init__.py`, sin ningún contenido, para que ElastAlert pueda reconocer el nuevo módulo de mejora. Finalmente se deben añadir las siguientes líneas al archivo `rules/new_mac.yaml`, que contiene la especificación de la alerta:

```
match_enhancements:
  - "my_enhancements.clean_new_mac_event.CleanNewMACEvent"
```

Una vez reiniciado el servicio de ElastAlert, las notificaciones recibidas ya no incluirán los datos eliminados en el script.

La capacidad de modificar las alertas con scripts de Python es muy interesante, ya que también presenta la posibilidad de completar las alertas con cualquier tipo de información, ya sean datos derivados de los campos de la alerta original, consultas a otros documentos de Elasticsearch, o información extraída de otras bases de datos o de la web. También podrían componerse URLs donde encontrar más información sobre la alerta, como el dashboard del usuario afectado o una página con la especificación de la regla activada o de la vulnerabilidad explotada si procede.

Notificación de alertas de Suricata

Durante la fase anterior se configuró ElastAlert para enviar una notificación por cada alerta reportada por Suricata. Esta regla se acabó desactivando dada la gran cantidad de notificaciones recibidas. Durante esta fase se realizaron dos modificaciones importantes a esa regla con el objetivo de reducir el número de notificaciones:

En primer lugar se definió un nuevo script para el módulo de mejoras que, además de formatear el contenido, evita enviar ciertas notificaciones demasiado ruidosas. De este modo, aunque algunas alertas ya fueron desactivadas en la configuración de Suricata, esta solución permite almacenar en Elasticsearch otras alertas que podría interesar auditar, pero sin necesidad de enviar las notificaciones. A continuación se incluye el script contenido en el archivo `my_enhancements.clean_suricata_alerts.py`:

```
from elastalert.enhancements import BaseEnhancement
from elastalert.enhancements import DropMatchException

class CleanSuricataAlert ( BaseEnhancement ) :

   def process (self, match):
         disabled = [
               'ET POLICY Spotify P2P Client',
               'ET POLICY GNU/Linux APT User-Agent Outbound'
               + 'likely related to package management'
               ]

         if (
               (match['alert'] is not None)
               and (match['alert']['signature'] is not None)
               and (match['alert']['signature'] in disabled)
```

```
            ):
        raise DropMatchException()

        match.pop('_type', None)
        match.pop('_id', None)
        match.pop('_index', None)
        match.pop('@timestamp', None)
        match.pop('@version', None)
        match.pop('type', None)
        match.pop('event_type', None)
        match.pop('tags', None)
        match.pop('input', None)
        match.pop('agent', None)
        match.pop('ecs', None)
        match.pop('host', None)
        match.pop('log', None)
```

Y en segundo lugar se aplicaron las opciones `realert` y `exponential_realert`, que evitan el reenvío de notificaciones frecuentes, de modo que una determinada notificación no pueda repetirse más de una vez cada 15 minutos, y en caso de que se continúe generando la misma alerta, la ventana se extendería consecutivamente en 15 minutos por cada notificación enviada. A continuación se muestra la configuración final de esta regla, almacenada en el archivo `rules/suricata_alerts.yaml`:

```
name: Suricata Alert
description: Notify every Suricata Alert
index: filebeat*
type: any
filter:
- term:
   _type: "_doc"
- term:
   event_type.keyword: "alert"

alert:
- "telegram"
telegram_bot_token: "830731765:FAkETOkEnExaMPleeebFJJioahDl04_HchA"
telegram_room_id: "535453583"

realert:
  minutes: 15
query_key:
  - device_mac
  - alert.signature
exponential_realert:
  minutes: 15

match_enhancements:
  - "my_enhancements.clean_suricata_alert.CleanSuricataAlert"
```

5. Conclusiones

Se ha demostrado que existen múltiples soluciones que permiten instalar un sistema de detección de intrusiones de bajo coste en el entorno doméstico o en la pequeña empresa. Durante este proyecto se han estudiado algunas topologías y herramientas que sirven perfectamente para este propósito, pero hay más alternativas interesantes (Snort, Zeek, Graylog, servicios en la nube, etc.) que podrían adaptarse mejor dependiendo de las necesidades.

Para hacer realmente efectivo el sistema, se comprobó que la especificación de reglas de detección, y de las notificaciones que deben ser enviadas, es un proceso largo que requiere experiencia y muchas iteraciones. Resulta crítico lograr una configuración adaptada al caso de uso, que detecte correctamente las amenazas minimizando los falsos positivos para evitar que el administrador empiece a ignorar las alertas.

Entre las herramientas utilizadas, cabe resaltar Kibana como una alternativa muy potente, aunque con una curva de aprendizaje bastante elevada. A pesar de que durante el desarrollo del proyecto se exploraron muchas de sus opciones de visualización, no se han llegado a demostrar la mayoría de sus posibilidades. Además, tanto Kibana como el resto del software mantenido por Elastic son proyectos frecuentemente actualizados que mejoran a gran velocidad. Durante el desarrollo de este proyecto Elastic llegó a publicar 5 versiones nuevas.

También el resto de herramientas estudiadas, como Snort o Suricata, muestran una gran actividad, publicando cambios constantemente en sus repositorios oficiales. Esta rápida evolución hace bastante atractivo el futuro de proyectos como este.

El bot de Telegram desarrollado inicialmente para relacionar los eventos con sus dispositivos y usuarios demostró tener también un gran potencial. Algunas de sus posibles aplicaciones se plantean en el siguiente apartado: Trabajo futurible.

En cuanto a la infraestructura utilizada, se resalta la Raspberry Pi 4 como una pieza de hardware sorprendentemente potente. Resulta impresionante que este dispositivo de bajo coste sea capaz de ejecutar todas las herramientas seleccionadas sin necesidad de limitar los recursos de ningún proceso. Por otro lado, se observó que la instalación de herramientas en Raspberry es bastante conflictiva, principalmente por la falta de soporte oficial para arquitecturas ARM. Del mismo modo, gracias a la comunidad de cada proyecto pudieron encontrarse soluciones a todos los problemas encontrados.

Para finalizar, se han cumplido satisfactoriamente todos los objetivos principales establecidos al inicio del proyecto: estudiar el estado del arte, seleccionar una combinación de herramientas y lograr un sistema totalmente funcional y de bajo coste a partir de dicha selección. También se han logrado cumplir los objetivos específicos: estudio de topologías alternativas, y exploración de las opciones de detección, visualización y notificación de los eventos de seguridad. Además, el análisis de riesgos

ayudó a tomar decisiones correctas que evitaron retrasos, y con ello cumplir adecuadamente con la planificación establecida al inicio del proyecto.

6. Trabajo futurible

Durante el desarrollo de este proyecto surgieron otras ideas interesantes que no llegaron a implementarse debido a la restricción del tiempo disponible. No obstante, se comentan a continuación como posible trabajo futurible.

Detección y notificación de conexiones anómalas

Se estudiaron diversas opciones con el objetivo de detectar anomalías en los patrones de comportamiento de cada dispositivo o usuario.

En primer lugar se contemplaron soluciones basadas en *Machine Learning* para la detección de anomalías, pero las opciones encontradas con una implementación suficientemente sencilla (como la solución de ML incluída en las últimas versiones de Kibana), requieren el pago de licencias que suspenden el objetivo del coste reducido del proyecto. Otras soluciones basadas en librerías de código abierto requerirían una implementación con un coste de tiempo no disponible.

También se consideró la detección de tráfico sospechoso basado en su comportamiento, tal y como se vió en la fase de investigación para algún módulo de zeek, intentando detectar conexiones repetitivas en el tiempo y con payloads de baja varianza.

Por último, se estudió crear otra regla de ElastAlert para la detección de conexiones a puertos inusuales y notificar, por ejemplo, que un dispositivo empieza a realizar conexiones al puerto SSH, siendo ello un comportamiento anómalo para ese dispositivo. Para ello podría crearse otra regla de tipo *new term,* filtrando por eventos de conexiones nuevas (`flow.state: "new"`).

Pruebas de detección de intrusiones

Sería interesante comprobar las capacidades de detección de intrusiones de Suricata llevando a cabo una intrusión real. Para ello se planteó la instalación de algún servicio vulnerable y explotable con metasploit[48], para confirmar que en caso de ataque el sistema reportaría este tipo de alertas.

[48] https://www.metasploit.com/

Con la misma finalidad también se estudió la posibilidad de instalar en algún dispositivo conectado al punto de acceso alguna herramienta popular de creación de puertas traseras como TheFatRat [49].

Notificación en picos de alertas

La configuración final de ElastAlert ignora ciertas alertas ruidosas que, aunque individualmente no sean importantes, todavía sería interesante recibir un aviso en caso de dispararse el número de notificaciones de estos tipos en un determinado rango de tiempo, que podría resultar sospechoso, y digno de investigación.

ElastAlert incluye el tipo de regla *spike* que detecta este tipo de anomalía. Habría que especificar los rangos de tiempo que serían comparados y el ratio necesario entre ambos para la activación de la alerta.

Explotación de las posibilidades del bot de Telegram

Se plantean las siguientes opciones que aprovecharían mejor las posibilidades que ofrece el bot de Telegram:

- Solicitar un listado dispositivos conectados, en el momento de la petición o durante un periodo de tiempo determinado (las últimas 3 horas, por ejemplo). Además, el bot podría ejecutar un escaneo de la red, pudiendo detectar no sólo los dispositivos conectados al punto de acceso, sino también los conectados al router detrás de la interfaz del puente.
- Desconectar o bloquear un dispositivo. Al recibir alguna alerta de seguridad, podría ser útil poder decidir inmediatamente si se prohíbe el uso de la red a ese dispositivo.
- Silenciar un determinado tipo de alerta para un dispositivo. La propuesta es poder consultar los tipos de alertas enviados recientemente, y dar la opción de silenciar cada tipo durante un tiempo determinado; o directamente desactivar la regla pertinente en la configuración de Suricata.

Alternativa para la configuración de ElastAlert

La instalación final requiere conectarse vía SSH a la Raspberry Pi, y editar o crear los archivo de reglas para la gestión de las notificaciones.

Durante el estudio del estado del arte realizado en la primera fase se consideró la versión de ElastAlert desarrollada por Bitsensor, que dispone de un plugin para poder gestionar las reglas desde Kibana. Lamentablemente esta versión no resolvió sus incompatibilidades con las últimas versiones de Elasticsearch antes de finalizar este proyecto.

[49] https://github.com/Screetsec/TheFatRat

Otra alternativa sería el uso de Graylog como motor de visualización, aprovechando su gestión integrada de alertas y notificaciones. Esta opción fue considerada en la fase de investigación, y desestimada por su incompatibilidad con las últimas versiones de Elasticsearch. Este conflicto tampoco logró resolverse antes de la finalización de este proyecto, pero con más tiempo podría considerarse realizar una instalación de esta herramienta sobre la versión soportada de Elasticsearch.

Mejora del rendimiento de Logstash

Al realizar muchas conexiones generando muchos eventos de Suricata en poco tiempo, como ocurre al realizar un escaneo de la red local, Logstash parece colapsar durante unos minutos, haciendo que los eventos se guarden con bastante retraso y las alertas se envíen mucho tiempo después de que ocurran. Aunque esto no afecta demasiado a la conectividad con internet ya que el proceso de Logstash se ejecuta con una prioridad baja, sería interesante estudiar cuál es la causa de que Logstash no sea capaz de procesar los eventos de Suricata suficientemente rápido.

7. Bibliografía

Documentación oficial y guías de usuario

- Suricata. Guía de usuario.
  https://suricata.readthedocs.io/en/suricata-5.0.0/
- Zeek. Manual de usuario.
  https://docs.zeek.org/en/stable/
- Elastic. Documentación de las herramientas Elastic.
  https://www.elastic.co/guide/index.html
- Elastic. Repositorio de Kibana.
  https://github.com/elastic/kibana
- Graylog. Documentación de Graylog 3.1.
  https://docs.graylog.org/en/3.1/
- Graylog. Repositorio de Graylog2-Server.
  https://github.com/Graylog2/graylog2-server
- Let's Encrypt. Documentación.
  https://letsencrypt.org/docs/
- Nginx. Documentación.
  https://nginx.org/en/docs/
- ElastAlert. Documentación.
  https://elastalert.readthedocs.io/en/latest/
- SirenSolutions. Repositorio de Sentinl.
  https://github.com/sirensolutions/sentinl
- Snort. Documentación y Guías.
  https://www.snort.org/documents
- Raspberry Pi Foundation. Guía de instalación de Raspbian en Linux.
  https://www.raspberrypi.org/documentation/installation/installing-images/linux.md
- Bitsensor. Repositorio de ElastAlert Kibana Plugin.
  https://github.com/bitsensor/elastalert-kibana-plugin
- Bitsensor. Repositorio del Servidor para ElastAlert Kibana Plugin.
  https://github.com/bitsensor/elastalert
- OISF. Repositorio de Suricata-Update.
  https://github.com/OISF/suricata-update

- Python. Manual de instalación de módulos. https://docs.python.org/3/installing/index.html
- Documentación de Filebeat. Secure communication with Logstash. https://www.elastic.co/guide/en/beats/filebeat/current/configuring-ssl-logstash.html

Vídeos

- Graylog. Sección de videos de la librería de recursos. https://www.graylog.org/resources#videos
- Corelight, Inc. An Introduction to Threat Hunting With Zeek (Bro). https://www.youtube.com/watch?v=R5mnIvjQn-g
- Zeek. Bro Befriends Suricata by Michal Purzynski. https://www.youtube.com/watch?v=_ObW8ZS0K5k
- Corelight, Inc. The power of Zeek/Bro and why you should include it in your security infrastructure. https://www.youtube.com/watch?v=Nb6DRvAKHCw
- Robbie Corley. BriarIDS suricata demo. https://www.youtube.com/watch?v=IjxnDSJ_SEM
- Lawrence Systems / PC Pickup. Suricata Network IDS/IPS System Installation, Setup and How To Tune The Rules & Alerts on pfSense. https://www.youtube.com/watch?v=KRlbkG9Bh6I
- Jesse K. Test Case: Suricata VS Snort IDS. https://www.youtube.com/watch?v=9FZEaqUAcUs
- Just me and Opensource. Elasticsearch Email alerting using Elastalert https://www.youtube.com/watch?v=vhRxUrg2OxM

Artículos, Blogs y Foros

- George Weir, Andreas Aßmuth and Mark Whittington. Cloud Accounting Systems, the Audit Trail, Forensics and the EU GDPR: How Hard Can It Be? https://strathprints.strath.ac.uk/63134/1/Weir_etal_BAFA2017_Cloud_accounting_systems_the_audit_trail_forensics_and_the_EU.pdf
- Proofpoint. Proofpoint ET Pro Ruleset. https://www.proofpoint.com/sites/default/files/pfpt-us-ds-et-pro-ruleset.pdf
- Reddit. ELK Stack vs Graylog. https://www.reddit.com/r/devops/comments/4jsuo2/elk_stack_vs_graylog/

- Sam Bocetta. 5 useful open source log analysis tools. https://opensource.com/article/19/4/log-analysis-tools
- Foro de Elastic. Installing Kibana on a Raspberry Pi 4 using Raspbian Buster. https://discuss.elastic.co/t/installing-kibana-on-a-raspberry-pi-4-using-raspbian-buster/202612
- Foro de Elastic. Installing Elasticsearch 7.4 on a Raspberry Pi 4 (Raspbian Buster). https://discuss.elastic.co/t/installing-elasticsearch-7-4-on-a-raspberry-pi-4-raspbian-buster/202599
- Eray Arslan. Setup Elasticsearch 7.x Cluster on Raspberry Pi / ASUS Tinker Board. https://medium.com/hepsiburadatech/setup-elasticsearch-7-x-cluster-on-raspberry-pi-asus-tinker-board-6a307851c801
- Cytopia. SSH Tunneling for fun and profit: Autossh. https://www.everythingcli.org/ssh-tunnelling-for-fun-and-profit-autossh/
- Santiago Basset. Mejorar la analítica de Seguridad con el Elastic Stack, Wazuh e IDS. https://www.elastic.co/es/blog/improve-security-analytics-with-the-elastic-stack-wazuh-and-ids
- Ahmad Darwich. Monitoring and detection of anomalies with ELK. https://blog.invivoo.com/monitoring-and-detection-of-anomalies-with-elk/
- Kumud Dwivedi. Realización de detección de intrusiones en la red con Azure Network Watcher y herramientas de código abierto. https://docs.microsoft.com/es-es/azure/network-watcher/network-watcher-intrusion-detection-open-source-tools

8. Anexos

8.1 Anexo 1: Código fuente del bot de Telegram

Archivo `/ids_bot/index.js`:

```
const telegram = require('telegram-bot-api')

const es = require('./es')
const macVendorLookup = require('./macVendorLookup')

const token = '89572364:FAKEToKENYILuzUhDwu5aiBdkhDl48_FchT'
const legitChatId = 537743183
const api = new telegram({token, updates: { enabled: true }})

const checkInterval = 1000 * 60
setInterval(() => {
  console.log('Checking for missing data...')
  reportPartialDevices()
  .catch(console.error)
}, checkInterval)


async function reportPartialDevices () {
  const reportable = await es.getReportable()

  for (const mac in reportable) {

   // Compose keyboard from reportable
   const inlineKeyboard = {
   inline_keyboard: [[
         {
           text: 'Set device name',
           callback_data: `set_name_${mac}`,
         },
         {
           text: 'Set user name',
           callback_data: `set_user_${mac}`,
         },
   ]]
   }

   const macVendor = await macVendorLookup(mac)

   console.log(`Reporting ${mac} - ${macVendor}`)
   api.sendMessage({
     chat_id: legitChatId,
     text: 'Connected device with missing data:\n'
```

```
        + `\nMAC Address: \`${mac}\``
        + `\nDevice Vendor: *${macVendor}*`
        + `\nDevice name: ${reportable[mac].device_name || ' ? '}`
        + `\nDevice user: ${reportable[mac].device_user || ' ?
'}`,
   reply_markup: JSON.stringify(inlineKeyboard),
   parse_mode: 'Markdown',
   })
   .then(msg => es.saveReportedMAC(mac))
   .catch(console.error)
  }
}


let msgIsInput = ''
api.on('message', function(message) {
  if (!isLegitChannel(message)) return

  if (msgIsInput && msgIsInput.match(/^set_/)) {
   const [ , field, mac ] = msgIsInput.split('_')
   const value = message.text
   es.saveMACField({mac, field, value})
   .then(() => {
         api.sendMessage({
         chat_id: legitChatId,
         text: `Ok, ${field} saved.`
         })
   })
   .catch(console.error)
  }
})


api.on('inline.callback.query', function(message) {
  if (!isLegitChannel(message.message)) return

  const data = message.data
  api.sendMessage({
   chat_id: legitChatId,
   text: 'Ok. Send me the name'
  })
  .then(msg => msgIsInput = data)
  .catch(console.error)
})


api.on('inline.query', function(message) {
  console.log('inline.query!')
  if (!isLegitChannel(message)) return

  // Received inline query
```

```
  console.log('inline query:', message);
})

api.on('inline.result', function(message) {
  console.log('inline result!')
  if (!isLegitChannel(message)) return

  // Received chosen inline result
  console.log('inline result:', message);
})


/**
 * If the message channel is not the legit channel, notifies the
 * legit channel, and throws an error to stop execution
 */
function isLegitChannel (message) {
  const chatId = message.chat.id

  if (chatId === legitChatId) return true

  api.sendMessage({
   chat_id: legitChatId,
   text: 'A stranger is talking to me:\n'
         + message.text,
  })

  return false
}

console.log(‘IDS Bot is online')
```

Archivo `/ids_bot/es.js`:

```
const elasticsearch = require('elasticsearch')

Object.assign(module.exports, {
  getReportable,
  saveReportedMAC,
  saveMACField,
})


const client = new elasticsearch.Client({
  host: 'localhost:9200',
  //log: 'debug',
  apiVersion: '7.3', // Same version of Elasticsearch instance
})
```

```
const conf = {

  // Min time between reports for  one MAC
  reportInterval: 1000 * 60 * 15,

  // Every minute
  checkInterval: 1000 * 60,

  // Track of reported MACs
  reportedMACs: [],
}


/**
 * Upserts a document for the given MAC in the deviced index,
 * adding the given field and value
 */
async function saveMACField ({mac, field, value}) {
  await client.update({
   index: 'devices',
   type: 'device',
   id: mac,
   body: {
   script: `ctx._source.device_${field} = '${value}'`,
   upsert: {
         device_mac: mac,
         [`device_${field}`]: value,
         created: new Date(),
   }
   }
  })
}


async function getPartialDevices () {

  // Get a current timestamp for local time
  const now = new Date()
  const offset = 0
  //const offset = now.getTimezoneOffset() * 60 * 1000
  const to = new Date(now - offset)
  const from = new Date(to - 1000 * 60 * 5) // last 5 min

  try {
   const data = await client.search({
   index: 'filebeat*',
   body: {
         query: {
         bool: {
         filter: [
               { exists: { field: 'device_mac' }},
```

```
                { range: { timestamp: { gte: from, lte: to }}}
          ],
          must_not: [
                { bool: { should: [
                { exists: { field: 'device_name' }},
                { exists: { field: 'device_user' }},
                ]}},
          ],
          },
          },
    },
    })
    const hits = data.hits.hits
    const devices = hits.reduce((macs, hit) => {
    const mac = hit._source.device_mac
    macs[mac] = macs[mac] || {}
    const devName = hit._source.device_name
    const devUser = hit._source.device_user
    if (devName) macs[mac].devName = devName
    if (devUser) macs[mac].devUser = devUser
    return macs
    }, {})
    return devices
  } catch (err) {
    console.error(err)
  }
}


async function getReportable () {
  const partialMACs = await getPartialDevices()

  // Clean report registry
  const now = new Date()
  conf.reportedMACs = conf.reportedMACs
                      .filter(r => r.time > now -
conf.reportInterval)

  // Ignore MACs that were reported recently
  for (const mac in partialMACs) {
    if (conf.reportedMACs.find(r => r.mac === mac)) {
    delete partialMACs[mac]
    }
  }

  return partialMACs
}


function saveReportedMAC (mac) {
  conf.reportedMACs.push({
```

```
    time: new Date(),
    mac,
  })
}
```

Archivo `/ids_bot/macVendorLookup.js`:

```
const axios = require('axios')

module.exports = macVendorLookup

async function macVendorLookup (mac) {
  const vendorBytes = mac.toString().slice(0,8)

  // Throttle reqs, to avoid hitting API limit
  await wait(2000)
  const url = `https://api.macvendors.com/${vendorBytes}`
  const { data } = await axios.get(url)
  return data
}

function wait (ms) {
  return new Promise(resolve => setTimeout(resolve, ms))
}
```

Archivo `/ids_bot/package.json`:

```
{
  "dependencies": {
    "@elastic/elasticsearch": "7.3.0",
    "axios": "^0.19.0",
    "elasticsearch": "^16.5.0",
    "telegram-bot-api": "^1.3.3"
  }
}
```