Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e implementación de infraestructura NIDS (Network Instrussion Detection System) para PIMES.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Asier Agra Monte

Tutor: José Ismael Ripoll Ripoll

2016 - 2017

Resumen

La ciberseguridad es un elemento clave para el desarrollo económico. La protección frente a las amenazas (introducción de código dañino en sistemas, ataques a páginas web, exfiltración de información, fraude electrónico…) y el fomento de la seguridad constituyen factores esenciales para el desarrollo de la economía de Internet.

Una de las soluciones más empleadas es la que se conoce como NIDS. Se podría entender como la evolución del concepto de "antivirus". Este esquema de defensa permite detectar más tipos de ataques, con mayor antelación y se pueden tomar contramedidas más efectivas.

El trabajo consiste en seleccionar, configurar e integrar todos los elementos hardware, software y organizativos para conseguir una solución operativa NIDS contra ataques informáticos. Se empleará para ello dispositivos hardware de bajo coste y software libre.

El desarrollo del proyecto requiere de:

1. Trabajar con equipos de red y pequeños dispositivos (como Raspberry Pi).
2. Estudiar, evaluar y configurar IDS.
3. Gestionar motores de bases de datos.
4. Seleccionar y configurar aplicaciones de interfaz para logs y eventos.

Palabras clave: NIDS, IDS, PYME, seguridad, raspberry, suricata, elasticsearch.

Abstract

Cybersecurity is a key element for economic growth. Protection against threats (introduction of malicious code in systems, webssite attacks, information exfiltration, electronic fraud…) and security encouragement are essential factors for the Internet economy growth.

One of the most commonly used solutions is known as NIDS. It can be understood as the evolution of "antivirus" concept. This defence scheme allows to detect more types of attacks, faster and allows to take more effective countermeasures.

The Project involves selecting, configuring and integrating all hardware, software and organizative elements to get an operational NIDS solution against computer attacks. There will be used low-cost hardware devices and free software.

The development of the project requires:

1. Working with network and small devices (like Raspberry Pi).
2. Study, evaluate and configure IDS.
3. Manage database engines.
4. Select and configure interface application for logs and events.

Keywords : NIDS, IDS, PYME, security, raspberry, suricata, elasticsearch.

Resum

La ciberseguretat es un element clau per al desenvolupament econòmic. La protecció enfront de les amenaces (introducció de códi nociu en sistemes, atacs a pàginas web, exfiltració de informació, frau electrònic…) i el foment de la seguretat constitueixen factors essencials per al desenvolupament de l'economia d'Internet.

Una de les solucions més empradess és la que es coneix com NIDS. Es podria entendre com l'evolució del concepte de "antivirus". Aquest esquema de defensa permet detectar més tipus de atacs, amb més antelació i se poden prendre contramesures més efectives.

El traball consisteix en seleccionar, configurar e integrar tots els elements hardware, software y organitzatius per aconseguir una solució operativa NIDS contra atacs informàtics. Es farà servir per a això dispositius hardware de baix cost y software lliure.

El desenvolupament del projecte requereix de:

1. Treballar amb equips de xarxa i petits dispositius (com Raspberry Pi).
2. Estudiar, avaluar i configurar IDS.
3. Gestionar motors de bases de dades.
4. Seleccionar i configurar aplicacions d'interficie per logs i esdeveniments.

Paraules clau: NIDS, IDS, PYME, seguretat, raspberry, suricata, elasticsearch.

Agradecimientos

Antes de comenzar con la memoria del proyecto, me gustaría dedicar unas líneas a las personas que han sido relevantes en estos últimos años, y que poco a poco han sido desencadenantes y participes de mi evolución hasta ser la persona que soy ahora.

En primer lugar, me gustaría agradecer a mi madre. Es un referente a nivel profesional y personal. Es una persona que siempre ha creído en mí y gracias a su apoyo he podido estudiar una carrera universitaria.

En segundo lugar, me gustaría agradecer al equipo docente de la Universidad Politécnica de Valencia. Gracias a muchos de ellos despierta en los alumnos el interés por algún campo en especial. En mi caso personal, quiero agradecer a mi tutor, José Ismael Ripoll, pues su énfasis al contar noticias de Ciberseguridad me inspiró a investigar por mi cuenta, descubriendo mi actual orientación profesional.

Por último, pero no por ello menos importante, quiero agradecer a todos los compañeros que, a lo largo de esta etapa han influido en mayor o menor medida en mí. Gracias a ellos por todas las horas de estudio, trabajo en equipo, conocimientos compartidos y, sobretodo, tiempo de ocio. Sin ellos la experiencia de estudiar una carrera no hubiera sido la misma.

Tabla de contenidos

1. Introducción

El principal objetivo de este trabajo, como se describe en el resumen es "seleccionar, configurar e integrar todos los elementos hardware, software y organizativos para conseguir una solución operativa NIDS contra ataques informáticos enfocado a PYMES" pero ¿Cuál es la motivación que me lleva como autor a realizar este proyecto?

1.1. Motivación

En un mundo en el que la tecnología ha tomado mando, y los sistemas de la información tienen el conocimiento y control de la mayoría de procesos de una empresa, la ciberseguridad se antoja cada día más importante, llegando a ser clave en algunos casos.

Este concepto parece quedar claro en países tecnológicamente avanzados, como Estados Unidos, Alemania, Inglaterra... Pero resulta relativamente nuevo en territorios como el nuestro.

Analizando la situación de nuestro país, puedo decir por mi experiencia profesional que, las empresas de mayor peso ya toman cartas en el asunto, invirtiendo parte de su presupuesto en materia de ciberseguridad. El problema reside en las pequeñas y medianas empresas que bajo argumentos como "¿Quién puede desearme el mal?" o "¿Qué pueden querer de mí?" descuidan la protección de su infraestructura por el importante desembolso de presupuesto que puede suponer para estas.

Así como las empresas parecen tener clara la importancia de tener "una puerta y cuatro paredes" vigiladas protegiendo su empresa, no trasladan este concepto al ámbito de las tecnológico. Por esta razón, muchos empresarios son reacios a invertir en ciberseguridad. Con este proyecto deseo simplificar las medidas para llevar a cabo esta tarea y su coste. De esta manera, se pretende impulsar a alguna PYME a comenzar su inversión en materia de ciberseguridad.

1.2. Apartados de la memoria

A lo largo de esta memoria, se tratan diversos puntos que nos llevan desde el análisis de las bases del proyecto a su implementación, pasando por la explicación de las elecciones tomadas.

En un primer lugar se definen los conceptos necesarios para entender el proyecto. Estos explican qué es una intrusión, un sistema de detección o los diferentes tipos que existen.

A continuación, se explican los diferentes elementos que componen un sistema de detección de intrusos basado en red. Esta definición a alto nivel ayuda al lector a entender el funcionamiento y el flujo de datos de este tipo de sistemas.

Una vez definida la arquitectura, se pasa a la comparación de las diferentes opciones presentes en el mercado que mejor se ajustan a las necesidades planteadas. Esto se centra en conseguir un sistema de sencilla implementación, bajo mantenimiento y un coste contenido, acorde con las posibilidades de una PYME.

Tras esto, se describe la instalación del sistema para un entorno genérico con un servidor y una única sonda, explicando las configuraciones más relevantes del proyecto para, en caso de ser necesario, conocer cómo escalar el sistema en horizontal, desplegando multitud de nodos en paralelo.

En las últimas páginas de la memoria, se encuentra un apartado con conclusiones sobre el sistema una vez desplegado en una infraestructura de pruebas, proporcionada por un cliente anónimo para la prueba del rendimiento del sistema.

También acompaña a estas un breve apartado en el que se menciona los trabajos futuros que surjan de observaciones una vez evaluado el proyecto desplegado.

2. Conceptos básicos

2.1. Intrusión

La Real academia de la lengua española define una ***intrusión*** como: "apropiarse, sin razón ni derecho, de un cargo, una autoridad, una jurisdicción, etc". Esta definición es muy amplia y poco precisa, por lo que vamos a centrarnos en lo tocante a la seguridad de un sistema de la información o de control.

La seguridad en estos sistemas viene marcada por tres principios básicos:

Confidencialidad:

Es la propiedad que impide la divulgación de información a individuos, entidades o procesos no autorizados. Esta asegura el acceso a la información y control únicamente a aquellas personas que cuenten con la debida autorización.

Un ataque de este tipo podría hacer pública información sensible o filtrar información acerca del proceso de negocio-fabricación.

Integridad:

Este principio busca mantener los datos libres de modificaciones no autorizadas, evitando que la información sea manipulada.

La modificación de datos puede llevar a estados inconsistentes que deriven en productos de peor calidad, desgaste de máquinas o toma de malas decisiones, entre otras.

Disponibilidad:

Esta característica responde a la cualidad del sistema de encontrarse operativo y a disposición de ser utilizado por las personas o procesos autorizados a ello.

Este tipo de ataques podrían dejar inservible una parte de la cadena de producción o imposibilitar el acceso a la página web corporativa, impidiendo ventas, servicios, etc.

Aplicando este concepto a los sistemas de información o de control, podemos definir intrusión como "un conjunto de acciones que intentan comprometer la integridad, confidencialidad o disponibilidad de un sistema".

2.2. IDS o Sistema de Detección de Intrusos

Un IDS o Sistema de Detección de Intrusos es una herramienta de seguridad que nos proporciona la posibilidad de monitorizar el estado y detectar eventos que ocurren en una infraestructura de tecnologías de la información (en adelante TI) o de control (en adelante OT). Los IDS buscan patrones anómalos y previamente definidos que pueden implicar actividad sospechosa en dicha infraestructura.

Un IDS es un sistema pasivo diseñado para la prevención y alerta de eventos, por lo que no evitará el ataque detectado, relegando esta tarea al analista de seguridad encargado de la monitorización. El analista de seguridad será, por tanto, el que tome las medidas reactivas necesarias para gestionar y solucionar el incidente. Estas herramientas se basan en los siguientes patrones para la detección:

- Actividad de los usuarios del sistema.
- Configuraciones y vulnerabilidades del sistema.
- Reconocimiento de modelos de ataques conocidos.
- Análisis estadístico de modelos de actividad.

2.3. Necesidad de un IDS

La evolución de la tecnología y en particular la irrupción de internet ha provocado un gran cambio en el paradigma de la sociedad. La información se procesa, almacena y transmite sin restricciones de distancia, tiempo ni volumen. Este nuevo entorno tiene una gran trascendencia en las empresas y los mercados se han transformado en globales y digitales en poco tiempo.

En este entorno, la seguridad cobra un sentido especial. Todas las propiedades del mercado digital (Velocidad, capacidad, movilidad…) son aprovechadas y explotadas por aquellos que pretenden obtener beneficio de manera fraudulenta.

Por consiguiente, es de vital importancia entender (y posteriormente prevenir) las pérdidas que puede causar un ataque informático como los anteriormente mencionados en una empresa, independientemente de su tamaño.

Es importante tener presente las ventajas y beneficios que un IDS pueden suponer sobre el coste tienen para la organización. En empresas con un incipiente nivel de madurez se utilizan argumentos tales como “¿Quién va a estar interesado en atacarme?”. Lo cierto es que como objetivo premeditado, ya sea por la competencia o un usuario descontento, o simplemente por azar en campañas de malware a gran escala, todas las empresas y particulares son objetivos potenciales.

2.4. Network-based IDS o IDS basado en red

Los Network-based IDS o IDS basados en red, son un subgrupo dentro de los IDS que se caracterizan por analizar eventos del tráfico de la red en la que se instalan y clasificarlos adecuadamente dependiendo de su contenido. Un NIDS realiza un examen exhaustivo de los paquetes y segmentos de la red que monitoriza, buscando coincidencias con patrones previamente definidos como sospechosos.

Normalmente están formado por un conjunto de sensores, también llamados sondas, localizados estratégicamente en diferentes puntos de la red. Estos sensores se encuentran configurados para actuar de manera transparente a la comunicación, por lo que no requiere ningún tipo de modificación de la infraestructura a monitorizar.

La principal ventaja de este tipo de sistemas es que, gracias a los sensores, es toda la infraestructura de red y no dispositivos por separado los que son monitorizados. Esto permite detectar un mayor número de ataque y relacionar acciones entre si.

Esta transparencia también tiene implicaciones positivas en el ámbito de la seguridad, pues resulta difícil para un atacante detectar la presencia de un NIDS.

Existen dos modos de configuración para estos dispositivos sonda:

Inline

En esta configuración, el IDS se encuentra en un host que actúa como proxy de red. Esta sonda analizará todo el tráfico que atraviese sus interfaces de red. Es de vital importancia analizar los requerimientos de hardware necesarios para esta configuración, pues el análisis podría introducir retardos en las comunicaciones. Este tipo de retardos puede ser inadmisible en algunas redes, por ejemplo, en infraestructuras de control industrial.

Modo Tab

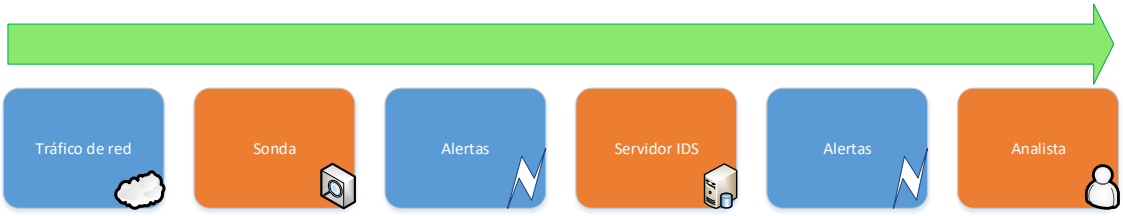
Para llevar a cabo esta arquitectura, se selecciona un dispositivo de red y se realiza un “port mirroring” o “puerto espejo” del tráfico que llega a sus interfaces. Esta técnica, reenvía todo el tráfico, además de a su destino final, por una interfaz seleccionada para tal fin, consiguiendo en ella una copia de este. A esta interfaz se conecta un dispositivo sonda cuya interfaz de adquisición se encuentra en modo promiscuo, aceptando todos los paquetes de red sin importar destino, para su posterior análisis.

La elección tendrá en cuenta la infraestructura de la red a monitorizar, seleccionando la opción más eficaz o eficiente para cada caso, ya sea monitorizar la máxima área posible de la infraestructura o, por el contrario, centrándose en puntos estratégicos de importancia.

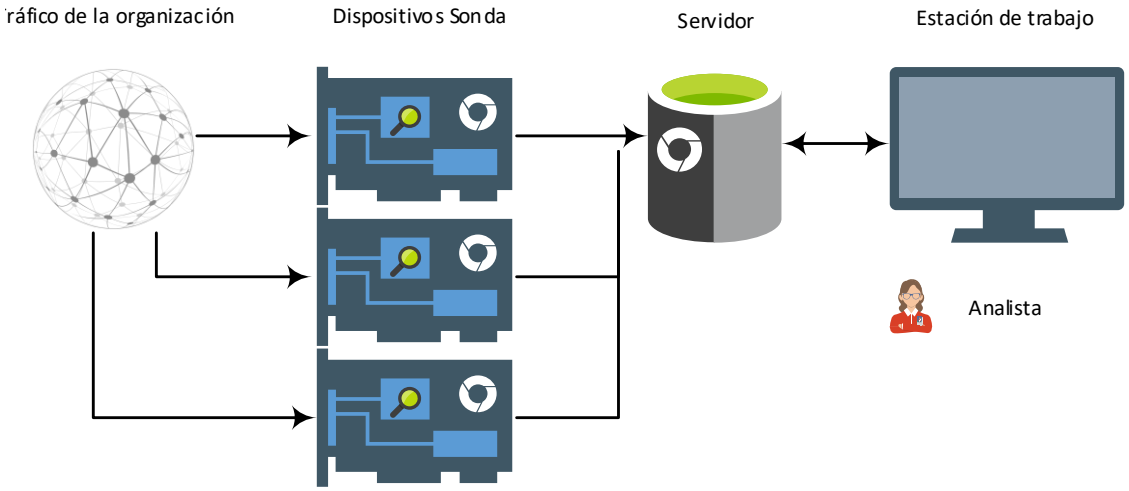
3. Elementos de un IDS

Ahora que el concepto de IDS basado en red ha sido definido, queda por conocer las piezas que lo forman, así como el flujo y trato que recibe la información a su paso por este.

A continuación, se presenta un diagrama del sentido del flujo de datos (de izquierda a derecha) y se detalla los diferentes puntos:



En este diagrama de alto nivel del flujo de datos, puede ser traducido a su vez en un diagrama basado en las distintas piezas del puzzle que forman el sistema, tomando como factor diferenciador el lugar en el que se alojan. Dicho diagrama quedaría como a continuación se expone:



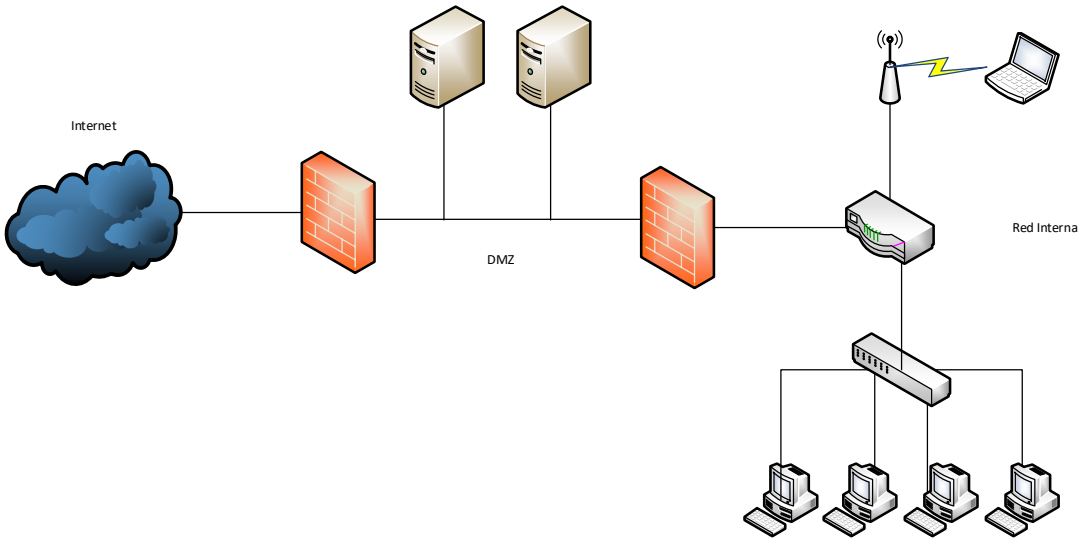
3.1. La Sonda

En primer lugar, en todo sistema de detección de intrusos basado en red tenemos los dispositivos Sonda. Las sondas son los encargados de la captura y procesado del tráfico de red en bruto.

Estas sondas contienen el motor de correlación que, en base a unas reglas predefinidas y mediante una comparación optimizada, emite alertas indicando posibles comunicaciones sospechosas.

Estas reglas han sido creadas en base a muestras de tráfico reportadas, extrayendo sus cualidades comunes como: Direcciones IP, dominios, puertos específicos, contenido dentro del mensaje y/o un conjunto de las anteriores.

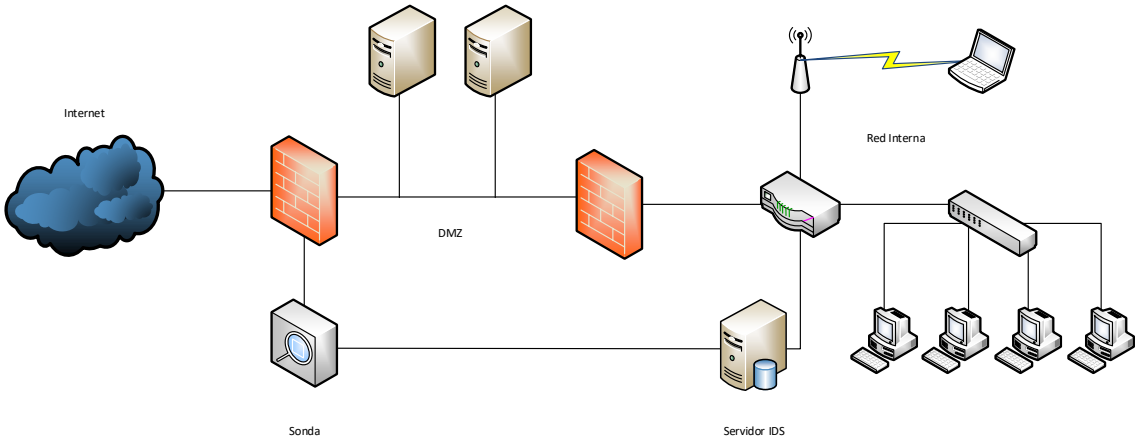
El número de dispositivos de este tipo y su localización pueden adoptar multitud de configuraciones. Esto permite obtener infinidad de escenarios de monitorización. Para ejemplificarlo, se ha diseñado un diagrama de la red de una pequeña empresa:



En ella se puede observar que la empresa tiene instalado un primer firewall (Muro izquierdo) que protege la zona desmilitarizada o DMZ del exterior. A continuación, tras otro firewall encontramos la red interna con redes cableadas y wifi.

3.1.1. Sonda situada en el firewall externo

Una primera aproximación sería colocar la sonda adquiriendo el tráfico que atraviesa el firewall externo:

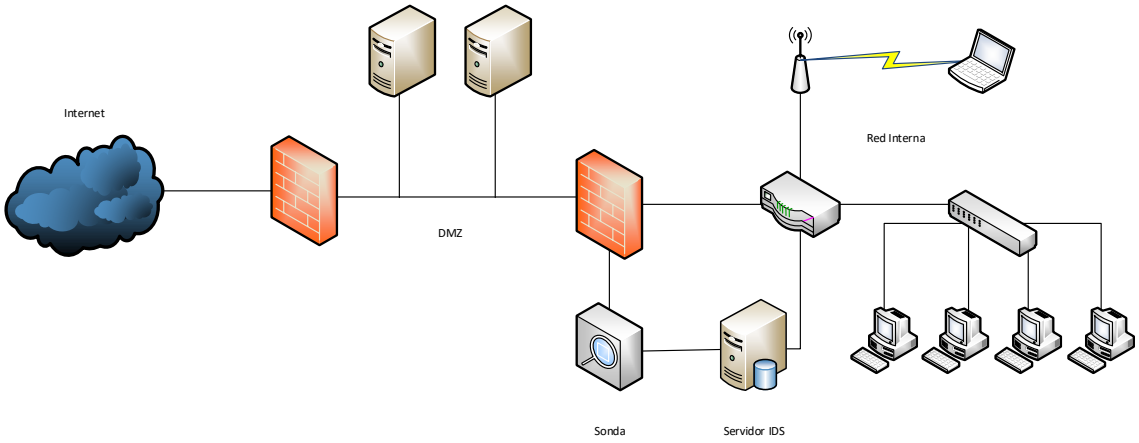


De esta configuración obtendríamos todas las alertas relacionadas con la infraestructura de la red que es capaz de penetrar el primer firewall.

En contrapartida, este primer firewall será mucho más permisivo que el segundo, por lo que no tendremos seguridad de que un ataque que ha penetrado dentro de la zona desmilitarizada, pase a red interna.

3.1.2. Sonda situada en el firewall interno

Como segunda opción podríamos situar la sonda en el firewall interno, como se muestra en la siguiente imagen:



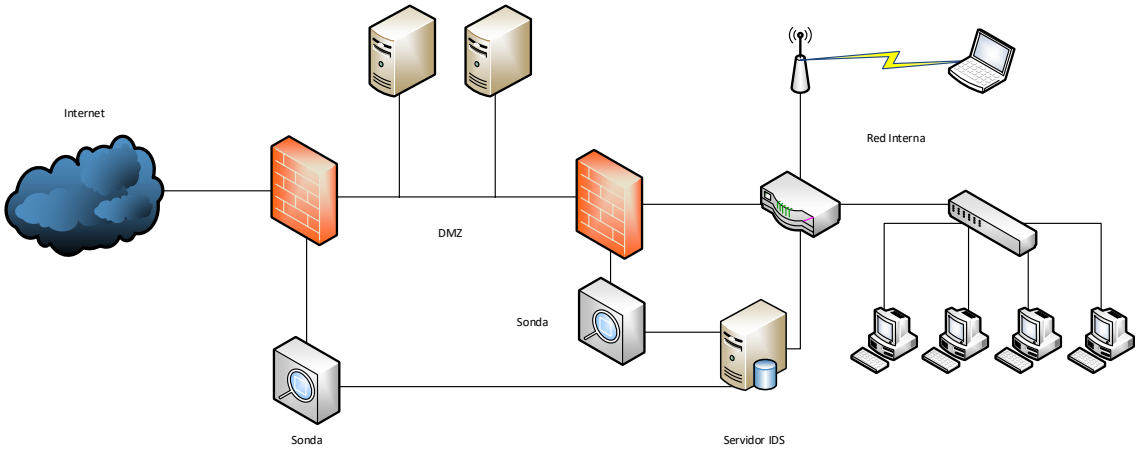
En este caso el tráfico analizado si ha atravesado el segundo dispositivo de seguridad, por tanto, nos encontramos con mayor fiabilidad a la hora de generar alertas que en el caso anterior.

Además, el volumen de tráfico que se procesa es considerablemente inferior, requiriendo un menor consumo de recursos para su análisis.

Como punto negativo en esta implementación, se deja fuera el tráfico que afecta a la zona desmilitarizada, por lo que según la infraestructura puede no ser recomendable.

3.1.3. Sondas situadas en ambos firewalls

Como se ha comentado al principio, existen infinidad de combinaciones posibles. Por tanto, ¿sería posible controlar los dos puntos anteriores a la vez?



En esta ocasión se han colocado las dos sondas, cada una analizando el tráfico de los diferentes niveles. Con esta solución se elimina la problemática de dejar zonas fuera de la cobertura del sistema de detección de intrusos, así como la incertidumbre ante la efectividad de un ataque contra la red interna.

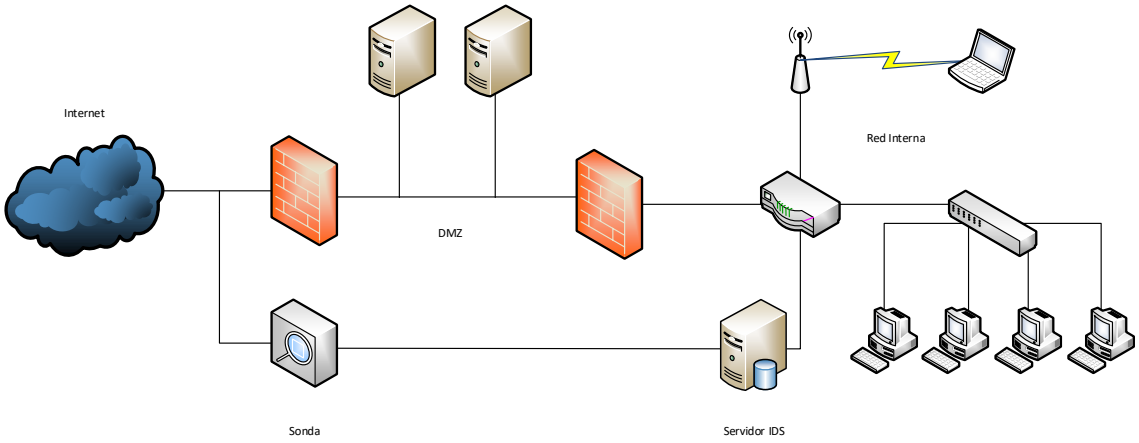
Pero esta solución tampoco es milagrosa. Con ella aumenta considerablemente el volumen de tráfico analizado y las alertas generadas, por lo que la inversión será considerablemente mayor.

También es realmente importante analizar duplicidades en el tráfico recibido por los dispositivos sonda, pues toda comunicación que se realice a la red interna será evaluada dos veces.

Para ello es conveniente configurar el firewall externo (o dispositivo que se encargue del enrutamiento) para enviar a la sonda solamente el tráfico que se dirige a los servidores de la propia zona desmilitarizada, dejando el resto para su análisis en el siguiente nivel.

3.1.4. Sonda situada fuera de la organización

Por último, existe otra aproximación qué, a pesar de tener una menor fiabilidad a la hora de emitir alertas puede tener otros fines:



En esta ocasión la sonda se sitúa fuera del firewall externo. Esta táctica puede parecer de escasa utilidad, pues muchos de los ataques que se realizaran son parados por los dispositivos de seguridad activa de la infraestructura.

Adicionalmente, el volumen de tráfico será indudablemente superior al expuesto en los casos anteriores.

¿Qué ventajas nos proporciona esta implementación con respecto al resto?

En primer lugar, nos permite estudiar a los posibles atacantes de una organización, sus técnicas, detectar si somos un objetivo habitual, si los ataques son automáticos, o si hay alguien intentando entrar por todos los medios posibles.

En segundo lugar, el tráfico que puede resultar realmente útil por su fiabilidad es el de salida de la organización. En caso de detectar una comunicación saliente realizada por un código malicioso o malware, sabremos que no solo se ha infectado el equipo, sino que ningún dispositivo lo ha parado, con lo que la comunicación con el punto de control ha sido efectiva.

3.2. Persistencia de datos

Una vez se han generado las alertas y de camino a ser procesadas por un usuario observamos que estas pueden ser generadas a cualquier hora, y a una velocidad vertiginosa, desbordando las capacidades humanas de los operarios.

Por esta razón se ve necesario almacenar las alertas generadas en una base de datos. Además del gran volumen de información, esta existe en una gran variedad de datos que pueden ser representados como orígenes, destinos, fechas, horas, acciones o el contenido de la comunicación.

Pero a pesar de la cantidad de volumen generado, el acceso a la información ha de ser ágil. Por tanto, en este tipo de sistemas es necesario recurrir a uno de los conceptos "de moda" en la actualidad: el Big Data.

3.2.1. Big Data

Denominamos Big Data a la gestión y análisis de enormes volúmenes de datos que no pueden ser tratados de manera convencional, ya que superan los límites y capacidades de las herramientas de software habitualmente utilizadas para la captura, gestión y procesamiento de datos.

¿Cuál es entonces la diferencia entre los clásicos modelos de gestión y los nuevos conceptos de Big Data? Las diferencias se asocian a tres palabras, las tres 'Vs' del Big Data: **Volumen**, **Variedad** y **Velocidad**.

Hablamos de Big Data cuando los **volúmenes** superan la capacidad del software habitual para ser manejados y gestionados. Cuando hablamos de grandes volúmenes nos referimos a tratamientos de información (p.e. alertas de IDS) que la tecnología convencional no permitía procesarlos cómodamente.

En el concepto de **variedad** nos referimos a la inclusión de multitud de fuentes de datos. Esto permite gestionar diversos puntos de una infraestructura al completo, e incluso permite al usuario relacionar sucesos sobre diferentes activos.

Por último, el concepto de velocidad se refiere a la **rapidez** con que los datos se reciben, se procesan y se toman decisiones a partir de ellos. A la mayoría de los sistemas tradicionales les es imposible analizar de forma inmediata los grandes volúmenes de datos que les llegan, sin embargo, incorporar este concepto permite un análisis ágil, mejorando los tiempos de detección y respuesta ante incidentes.

3.3. Presentación

La última fase del flujo de datos corresponde a la interacción del analista con el sistema de detección de intrusos. La interfaz gráfica es la parte del sistema informático que actúa como punto de comunicación entre este y el usuario, utilizando un conjunto de objetos gráficos para representar la información y acciones disponibles.

El objetivo de Big Data, anteriormente descrito, es convertir el Dato en información que facilita la toma de decisiones. Por esto cuando tratamos con grandes cantidades de información, una interfaz gráfica es primordial.

La capacidad de un analista para relacionar un grupo de alertas, trazar una línea temporal y, por último, emitir un diagnóstico y solución dependerá de las opciones que dicha interfaz ofrezca. Por consiguiente, entre las necesidades de esta se encontrará la aplicación de múltiples filtros como: IP origen, IP destino, puerto origen, puerto destino, fecha, hora, tipo de alerta, protocolo y campos dentro de la comunicación (payload).

A pesar de que este nodo se encuentra diferenciado en el flujo de datos del sistema, es común encontrarlo físicamente implementado en una de las máquinas del sistema encomendadas a la persistencia de datos. Con esta estrategia se disminuye la cantidad de tráfico enviado entre estos dos puntos, reduciendo la latencia en las consultas realizadas.

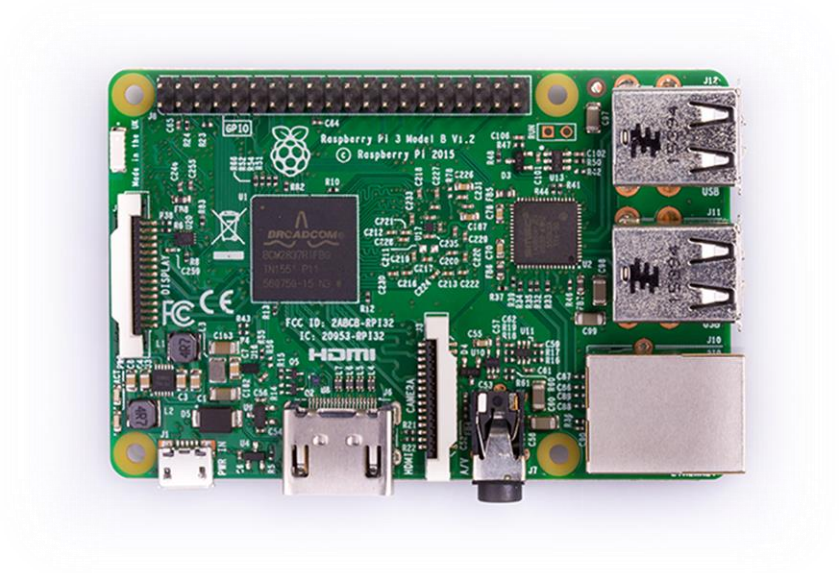
4. Elección hardware y software

A lo largo de este capítulo se exponen y justifican las elecciones tomadas respecto a las diferentes piezas de esta infraestructura. En primer lugar, antes de entrar en materia, se expone una lista con la relación de elementos analizados en el capítulo.

Dispositivo sonda
 Raspberry PI 3
Motor de detección
Snort Suricata
Base de datos
Elasticsearch
Recolección de datos
Rsyslog Logstash Beats
Interfaz gráfica
Graylog 2 Kibana
Sistemas operativos
Raspbian Ubuntu Arch Fedora Windows

4.1. Dispositivo sonda

La base elegida para albergar el software de detección de intrusos basado en red es la Raspberry PI 3 Modelo B [1]. La elección de este hardware como base fue la idea precursora de este proyecto.



Gracias a la mejora de capacidad de cómputo y conectividad que se le otorgó con la salida de este modelo y revisión de los modelos anteriores, se vio factible su uso en redes reducidas, como las que se puede encontrar en la mayoría de pequeñas y medianas empresas.

La Raspberry Pi fue diseñada por la **Fundación Raspberry Pi** con el objetivo de estimular el interés por la informática a nivel escolar, como hizo en los años 80 la Acorn con el BBC Micro.

En el año 2012 se comercializaron las primeras placas de Raspberry Pi modelo B, que tuvieron una gran acogida entre los aficionados a la programación, vendiendo más de medio millón de unidades en sus primeros 6 meses.

La primera placa Pi modelo B tenía una capacidad de computo reducida, 512 MiB de memoria SDRAM y una conectividad reducida. Con el éxito obtenido, se decidió sacar modelos más reducidos, como el modelo A, y con el tiempo, revisiones y evoluciones que terminarían por aumentar la capacidad de computo e incluso incluir conectividad vía Ethernet, Wifi y Bluetooth.

En el anexo I se incluye una tabla con los principales modelos de Raspberry PI, con las características más relevantes para este tipo de proyectos.

La fuerte acogida que ha tenido esta familia de dispositivos en el mercado se debe principalmente al reducido coste que poseen frente a la infinidad de posibilidades que

ofrece, respaldado por cantidad de proyectos y documentación generados por la comunidad.

Esta ventaja económica ha sido el segundo factor que ha empujado a que esta sea la base elegida para este tipo de dispositivos sonda, reduciendo el coste y aumentando la cantidad de dispositivos de este tipo que se pueden distribuir en la infraestructura, permitiendo configuraciones realmente interesantes como las explicadas en el apartado 3.1.

4.2. Motor de detección

Entre el amplio elenco de soluciones de detección de intrusos basados en red, cabe destacar dos productos sobre el resto: Snort y Suricata.

Snort ha sido el motor de IDS de facto durante años. Este motor tiene una enorme comunidad de usuarios soportándolo y, un abanico aún mayor y en aumento de suscriptores a sus reglas.

Por otro lado, aunque Suricata no tiene una vida tan larga en comparación con Snort, ha estado reclamando mercado como una respuesta evolucionada o alternativa a este, basando sus argumentos principalmente en sus capacidades de proceso multi-hilo.

4.2.1. Snort

Hay infinidad de opciones a la hora de elegir motores IDS disponibles para automatizar y simplificar el proceso de detección de intrusiones, y Snort [2] es una de las mejores opciones. Snort se ha convertido en la tecnología de prevención y detección más ampliamente extendida y confiable del mundo. La revista SC Magazine afirmó que el éxito de Snort se debe al hecho de que los usuarios de la comunidad de seguridad de código abierto de todo el mundo pueden detectar y responder ante incidentes de seguridad de forma más rápida y eficiente que con otros motores IDS de la competencia.

Además, hay una amplia variedad de guías de referencia disponibles para instalar, configurar, implementar y administrar sensores de Snort.

Pero, ¿Cuáles son las claves por las que Snort tiene ese éxito como monitor de tráfico de red?

Snort utiliza un lenguaje basado en reglas que combina beneficios de inspección basados en firmas, protocolos y anomalías. Con su velocidad, potencia y rendimiento, Snort ganó impulso rápidamente. Con casi 4 millones de descargas hasta la fecha, Snort se ha convertido en la tecnología de detección de intrusos más utilizada del mundo.

El principal trabajo de Snort es escuchar el tráfico de red y buscar firmas en el flujo de datos que puedan indicar un incidente de seguridad en la infraestructura a monitorizar. Las reglas están configuradas para realizar una determinada acción. Esta acción puede ser pasiva, alertando del incidente, o activa, aplicando contramedidas en tiempo real.

Las organizaciones pueden aprovechar la aplicación de conjuntos de reglas existentes proporcionadas por la comunidad de Snort, así como escribir o modificar sus propias reglas de acuerdo con requisitos específicos de la red. Resulta posible escribir reglas complejas para identificar casi cualquier tipo de tráfico que atraviesa la red.

Gracias a la comunidad, las reglas de IDS se encuentran bajo una continua revisión y mejora con el fin de detectar las últimas amenazas de seguridad conocidas.

4.2.2. Suricata

Entonces ¿Qué es Suricata y que beneficios ofrece en contrapartida a Snort?

Suricata [3] es un sistema de detección de intrusos (al igual que Snort) basado en código abierto con el fin de construir la próxima generación El objetivo de OISF es aportar nuevas ideas de seguridad e innovaciones tecnológicas a la industria de detección de intrusiones.de IDS de código abierto.

Con la ayuda financiera del Departamento de Seguridad Interna de los Estados Unidos, se creó una alternativa multiproceso a Snort para ayudar a proteger las redes contra intrusiones de seguridad avanzadas.

La arquitectura de múltiples hilos de Suricata es única, ya que puede soportar sistemas multi-core y multiprocesador de alto rendimiento. Los principales beneficios de un diseño de múltiples hilos es que ofrece mayor velocidad y eficiencia en el análisis de tráfico de red y también puede ayudar a dividir la carga de trabajo de IDS / IPS en función de las necesidades de procesamiento. Además de la aceleración de hardware (con limitaciones de hardware y de tarjeta de red), el motor está diseñado para utilizar la mayor potencia de procesamiento ofrecida por los últimos chips de CPU multi-núcleo.

Suricata se ha desarrollado con la idea de una fácil implementación, acompañado de documentación de inicio paso-a-paso y un potente manual de usuario. El motor también ha sido desarrollado en C, pensado desde los inicios para escalar. Adicionalmente, para facilitar la migración a este producto, Suricata emplea las mismas reglas y formatos de salida que Snort.

Aunque Suricata sigue siendo un producto joven y menos extendido que su competencia, esta tecnología está ganando impulso entre los usuarios y empresas. El aumento del rendimiento, la compatibilidad con IPv6 nativa, modelos estadísticos de detección de anomalías y aceleración GPU son, entre otras, los principales puntos a favor de este motor.

4.2.3. Elección

Ambos motores IDS han probado su efectividad y son realmente populares. En la siguiente tabla [4] se presentan algunas de las principales características de los dos motores:

	Snort	**Suricata**
Desarrollador	Sourcefire, Inc.	Open Information Security Foundation (OISF)
Disponibilidad	Desde 1998	Desde 2009
Lenguaje de codificación	C	C
Sistema Operativo	Cross-platform	Cross-platform
Versión estable	2.9.8.3 (22 Junio 2016)	3.0 (27 enero 2016)
Hilos	Hilo único	Multi-hilo
Soporte IPv6	Si	Si
Soporta reglas de Snort (VRT)	Si	Si
Soporta reglas de Emerging Threats	Si	Si
Formato de Logging	Unified2	Unified2
Compatible con Aanval	Si	Si

Tras un análisis de los motores de búsqueda, a pesar de encontrarse a la par en tecnología, Suricata parece situarse ligeramente adelantado respecto a Snort. En lo que respecta al proyecto, se decide emplear Suricata como motor de detección dado que su capacidad multi-hilo y escalabilidad aprovechan mejor las características hardware del dispositivo Raspberry Pi 3 Model B del que disponemos. Se recuerda que este dispone de un procesador ARM de cuatro núcleos que podrán ser utilizados de manera concurrente por este motor.

4.3. Base de datos

En lo tocante a la persistencia de datos y consultas parece no haber dudas, un producto domina el mercado del software libre: ElasticSearch.

4.3.1. ElasticSearch

ElasticSearch [5] es una solución de base de datos u servidor de búsqueda basado en Lucene. Este producto provee un motor de búsqueda distribuido y con capacidad multi-tenencia.

El término multi-tenencia se refiere a una arquitectura de software en la que una única instancia de software se ejecuta en un servidor y sirve a varios inquilinos. Los inquilinos son un grupo de usuarios que comparten un acceso común con privilegios específicos a la instancia de software. Con una arquitectura multi-tenencia, una aplicación de software está diseñada para proporcionar a cada inquilino una parte dedicada de la instancia, incluyendo sus datos, configuración, gestión de usuarios, funcionalidad individual y propiedades no funcionales. multi-tenencia contrasta con arquitecturas multi-instancia, donde las instancias de software por separado operan en nombre de los diferentes inquilinos.

Por otra parte, Leucene es una interfaz de programación de aplicaciones, o API, de código abierto pensada para la recuperación de información. Esta está orientada a cualquier aplicación que requiera indexado y búsqueda a texto completo.

El centro de la arquitectura lógica de Lucene se encuentra el concepto de Documento (Document) que contiene Campos (Fields) de texto. Esta flexibilidad permite a Lucene ser independiente del formato del fichero.

Esta solución emplea una interfaz RESTful. Esto es un estilo de arquitectura software para sistemas hipermedia distribuido y describe una interfaz entre sistemas que emplea HTTP para obtener datos o ejecutar operaciones sobre estos sin abstracciones adicionales.

Los sistemas REST, y por tanto nuestra elección, cumplen con una serie de fundamentos clave:

- Un protocolo Cliente-Servidor sin estado, es decir, cada mensaje contiene toda la información necesaria para comprender la petición. Por consiguiente, ninguna de las dos partes requiere recordar ningún estado.
- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de la información.
- Una sintaxis universal para identificar los recursos. En los sistemas REST cada recurso es direccionable únicamente a través de su URI.

Por último, los ficheros de datos emplean el formato JSON. Acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos. Una de las principales ventajas de este formato es la simplicidad de implementar un analizador sintáctico.

Pero no todo acaba aquí. La velocidad de proceso de ElasticSearch radica en su escalabilidad horizontal.

4.3.2. Cluster ElasticSearch

Otra de las principales ventajas de este producto. Es posible la ejecución de varias instancias en paralelo, trabajando de manera concurrente para dar un servicio más rápido a la hora de ejecutar búsquedas y, más confiable, manteniendo copias de la base de datos que asegurarían cierto grado de resiliencia en caso de caída parcial de los nodos de la infraestructura.

Para llevar esta tarea a cabo, un cluster de ElasticSearch dispone de diversos nodos que pueden asumir uno o varios roles dentro de los tres disponibles: nodos de datos, nodos de ingesta y nodos maestros.

En primer lugar, tenemos los ***nodos de datos.*** Estos son los nodos que almacenan la información. Entre ellos, se reparten fragmentos de la base de datos. Para cada fragmento del sistema existirá una copia principal y, al menos, una copia de respaldo ubicada en otro nodo diferente.

Esta repartición de fragmentos de la base de datos permite, además de añadir robustez al sistema, realizar consultas distribuidas. Esto disminuye de manera notable el tiempo de las consultas sobre bases de datos de gran tamaño, al haber más de una maquina realizando consultas en local sobre una parte reducida de los datos.

Para alimentar estos e introducir información en la base de datos, aparece la figura de los ***nodos de ingesta***. El trabajo de estos nodos es tratar la información y enviarla al nodo del cluster al que le corresponda almacenar la información.

Por último, tenemos los ***nodos maestros***. Estos son los encargados de repartir la carga de trabajo y volumen de información que tratarán los diferentes nodos.

Dado que la lógica del conjunto reside en estos, es conveniente disponer de más de un nodo elegible a asumir este papel. Ante una caída o segmentación de la red, en caso de no encontrarse un nodo maestro, se elegirá uno entre los elegibles para este cargo, creando sub-clusters que, en el momento de recuperar el estado óptimo, se encargaran de actualizar las partes que no estuvieran correctamente actualizadas.

4.3.3. Elección

En este caso, existiendo únicamente un producto, la elección reside en el modo de configuración, entre una única instancia de elasticsearch o en modo cluster. Esta elección depende de la cantidad de información que se desee almacenar y el presupuesto del que se disponga.

4.4. Recolección de alertas

En este apartado se analiza las diferentes opciones para enviar las alertas generadas por Suricata desde nuestro dispositivo sonda a la base de datos de elasticsearch.

4.4.1. Rsyslog

Syslog es uno de los estándares más usados en el reenvió de alertas y registros. Rsyslog [6] es una utilidad de software de código abierto utilizada en sistemas informáticos UNIX y similares a Unix para reenviar mensajes de registro en una red IP, incluyendo la Ubuntu basada en ARM elegida para este proyecto.

Implementa el protocolo syslog básico, lo extiende con filtrado basado en contenido, capacidades de filtrado enriquecidas, opciones de configuración flexibles y añade características como el uso de TCP para el transporte.

Este cliente puede ser configurado para enviar los registros de un único fichero y resulta extremadamente liviano tanto a nivel de red como de computo de máquina.

En contrapartida, Esta entrada sólo admite syslog RFC3164 con algunas pequeñas modificaciones. Por ejemplo, la fecha también permite el formato ISO8601. Por tanto, las opciones de envió en este cliente serán más reducidas si no emplea este estándar.

4.4.2. Logstash

El segundo cliente que se expone es Logstash [5] . Logstash es un motor de recolección de datos de código abierto desarrollado por Elastic, creadores de ElastiSearch. Este puede unificar dinámicamente datos de fuentes dispares y normalizar los datos en destinos de su elección.

Si bien Logstash originalmente impulsó la innovación en la recopilación de registros, sus capacidades se extienden mucho más allá de ese caso de uso. Cualquier tipo de evento puede ser enriquecido y transformado con una amplia gama de entradas, filtros y plugins de salida, con muchos códecs nativos simplificando aún más el proceso de la ingesta.

Esto permite modificar los registros para almacenar toda la información necesaria, en el formato que más nos interese y de la forma más óptima.

La aplicación se encuentra basada en jRuby y requiere de una “Java virtual Machine” para funcionar. Gracias a esto la herramienta es multiplataforma, pudiendo ser ejecutada en cualquier sistema operativo moderno como Linux, Mac OS X o Windows.

Otra ventaja de Logstash respecto a sistemas basados en syslog es que ha sido creada por el mismo desarrollador que la base de datos. Esto asegura que no habrá incompatibilidad entre las diferentes piezas de la implementación.

El punto negativo de este cliente, es que no se encuentra disponible oficialmente en arquitecturas ARM, y las versiones beta disponibles no se encuentran suficientemente pulidas en términos de estabilidad por su elevado uso de computo.

4.4.3. Beats

Por último, se presenta Beats [5] , una serie de agentes ligeros de reenvío, de código abierto, desarrollados también por Elastic como una alternativa más ligera a Logstash. Originalmente, para la ingesta de datos era necesaria la inclusión de Logstash como nodo puente en la arquitectura. Desde la versión 5.0 en adelante, ElasticSearch admite la entrada directamente desde Beats, simplificando el diseño de esta.

Packetbeat, Filebeat, Metricbeat y Winlogbeat son algunos ejemplos de Beats. Packetbeat es un analizador de paquetes de red que envía información sobre las transacciones intercambiadas entre sus servidores de aplicaciones. Filebeat envía archivos de registro desde sus servidores. Metricbeat es un agente de monitoreo de servidores que periódicamente recopila métricas de los sistemas operativos y servicios que se ejecutan en sus servidores. Y Winlogbeat envía registros de eventos de Windows.

Para el caso que atañe al proyecto, el cliente correcto sería Filebeat. Este, a diferencia de su hermano mayor, se encuentra implementado en versiones estables para arquitecturas con procesador ARM. En contrapartida, no ofrece la potencia de procesado que ofrece Logstash.

4.4.4. Elección

Tras evaluar los pros y contras de los tres productos, se decide emplear Filebeat. Este producto, en comparación con Rsyslog, ofrece más posibilidades de interpretación de los eventos, edición de campos, etc.

En comparación con Logstash, ofrece una versión más estable en arquitecturas con procesador ARM como la que se ha seleccionado para el dispositivo sonda. Así mismo, este cliente ligero ofrece una capacidad de interpretación menor, pero suficiente para los campos de los mensajes de alerta de Suricata.

Esta última razón también deriva en un uso de recursos bastante menor, lo cual también beneficia al dispositivo sonda, que recordamos tiene unas prestaciones contenidas.

4.5. Interfaz gráfica

Por último, queda por debatir la capa más externa de nuestra arquitectura, la cual estará en contacto con el usuario: La interfaz gráfica.

La elección de esta influirá en gran medida en la agilidad y capacidad de interpretar los eventos de seguridad que se han detectado en fases anteriores. Una mayor claridad en la exposición de los datos y posibilidad a la hora de aplicar filtros jugarán a favor de la eficiencia a la hora de investigar incidentes de seguridad.

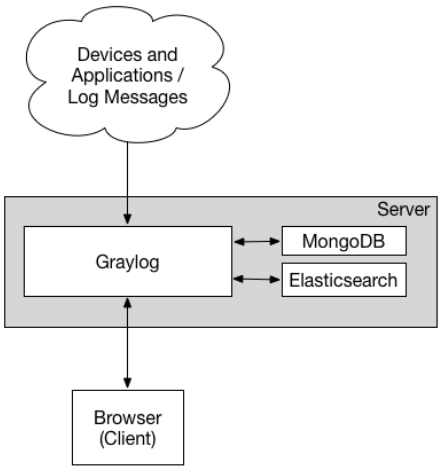
A continuación, se comentan las dos herramientas más populares en este ámbito:

4.5.1. Graylog

Graylog [7] es una solución para visualizar registros almacenados en una base de datos Elasticsearch. Graylog permite realizar consultas avanzadas sobre los datos almacenados, crear tablones con los resultados de las mismas o incluso generar alarmas ante determinados datos o ausencia de ellos.

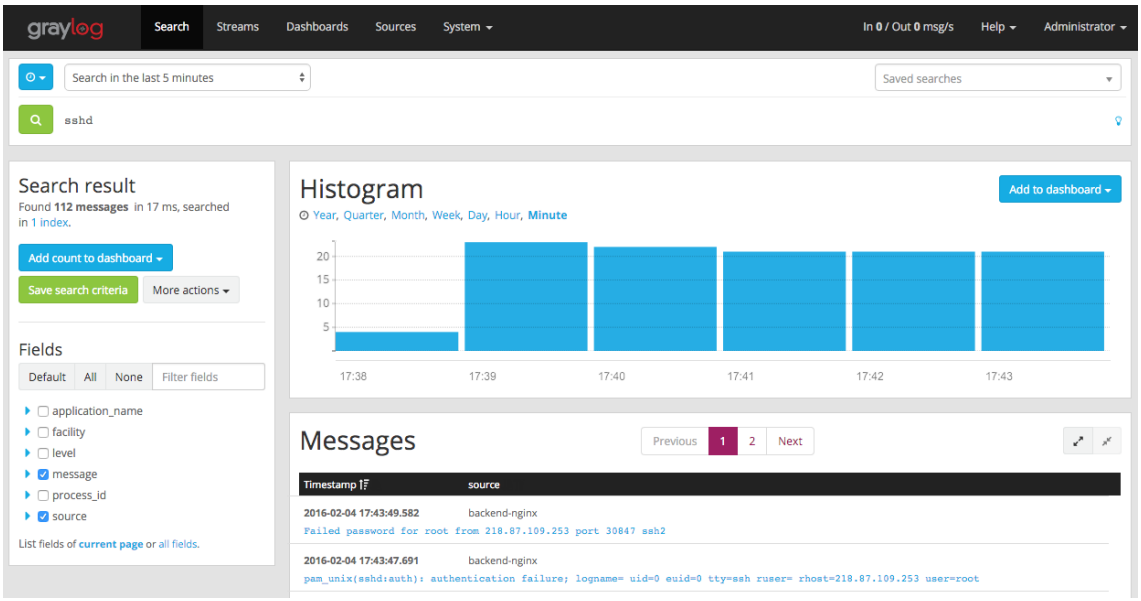
El acceso se realiza vía navegador y dispone autenticación de usuarios para poder gestionar el acceso a la aplicación. La parte servidor solo tiene versiones RPM y DEB, por lo que únicamente funcionará sobre plataformas Red Hat, Debian y similares.

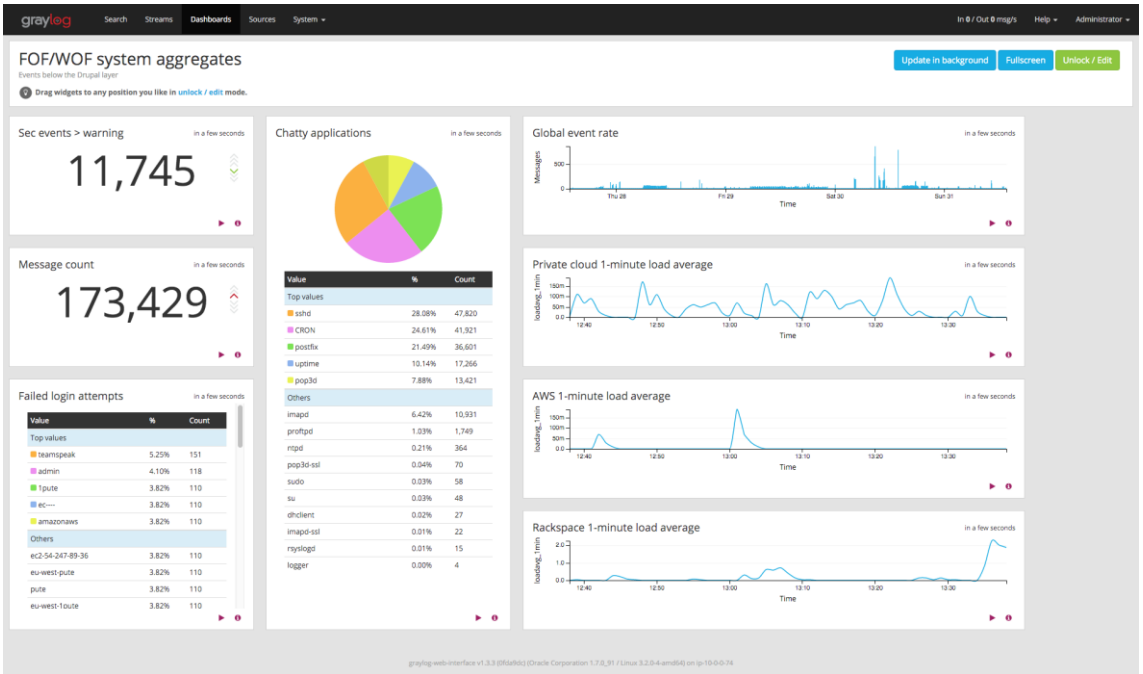
A pesar de emplear la base de datos Elasticsearch, graylog necesita de una base de datos adicional MongoDB o MariaDB para almacenar su configuración, usuarios, etc.



Una de las principales desventajas es que Graylog no es compatible con la última versión de Elasticsearch 5.x, perdiendo el gran potencial que ofrece esta base de datos actualmente.

En las siguientes imágenes se observa la interfaz gráfica de Graylog:





4.5.2. Kibana

Kibana [5] es la alternativa a Graylog propuesta por Elastic. Esta solución es una potente herramienta pensada para el análisis masivo de datos almacenados en una base de datos Elasticsearch. Gracias a esta concepción, enfocada al Big Data, Kibana ofrece una infinidad de nuevas posibles representaciones gráficas para los eventos, como mapas de calor, localizaciones, etc.

Como su alternativa, Kibana ofrece un servicio accesible vía navegador. Si bien la opción de acceso mediante usuarios no está disponible en la versión básica, se puede implementar añadiendo el modulo X-Pack o un proxy Nginx para tal fin.

El software servidor no tiene dependencias del sistema operativo en el que se encuentra dado que, al igual que el resto de software de elastic, se monta sobre la Máquina Virtual Java (JVM).

Otra de las principales ventajas de Kibana es su completa integración con el resto de productos elegidos. Al ser un stack pensado y recomendado por el propio fabricante, no existirán incompatibilidades entre versiones al actualizar el software.

A continuación, se presentan capturas de múltiples posibles interfaces diseñadas en Kibana:

4.5.3. Elección

Las dos herramientas resultan muy efectivas para este cometido. Las posibilidades de personalización juegan a favor de Kibana, pues es mucho más personalizable que Graylog.

A nivel de seguridad Graylog ofrece de manera nativa la autenticación por usuarios, pero no resulta una ventaja diferenciadora dado que existe la posibilidad de añadirlo mediante módulos adicionales.

Pero el último factor determinante es la compatibilidad con la base de datos. En estos momentos, solo Kibana permite trabajar con la versión 5.x de Elasticsearch, lo cual decanta definitivamente la balanza hacia este producto.

4.6. Sistema operativo del dispositivo sonda

Una vez seleccionado el dispositivo hardware que se empleará como base del dispositivo y el software que se encargará de generar y enviar alertas, llega el momento de analizar cuál será el sistema operativo más adecuado como base.

Este dispositivo será el encargado de alojar el IDS Suricata y el agente ligero Beats, por lo que el sistema base deberá ser capaz de ejecutar estos servicios.

Con la creciente popularidad de los dispositivos Raspberry, han proliferado una gran variedad de sistemas operativos para estas placas. A continuación, exponemos los más representativos para un proyecto de estas características.

4.6.1. Raspbian

Raspbian [8] es una distribución del sistema operativo Linux, libre y basado en Debian. Nació como un *port* no oficial de Debian para dar soporte a las CPUs basadas en ARM que empleaba este tipo de dispositivos. Para ello fueron modificados multitud de paquetes, mejorando el rendimiento respecto a la versión oficial en este tipo de plataformas. A pesar de esto, sigue empleando el sistema de paquetes DEB clásico de los sistemas Debian.

Debian por su parte es una de las distribuciones más extendidas basadas en Linux. Pertenece al "Proyecto Debian", una comunidad conformada por desarrolladores y usuarios que mantiene el sistema operativo que tratamos.

Nació en 1993 de la mano de Ian Murdock como una apuesta por separar el software libre del que no lo es. Aunque el proyecto tiene apoyo por parte de empresas, no persigue ningún fin comercial, poniendo su código a disposición de cualquier usuario en internet. Las empresas son libres de modificar y distribuir el software, siempre que respeten la licencia de "Software Libre".

Debian 0.01 fue lanzado el 15 de septiembre de 1993 pero no fue hasta 1996 cuando se lanzó la primera versión estable.

A pesar de no ser una distribución oficial de debían, existe un gran soporte por parte de la comunidad de Raspberry, que hicieron de este el sistema operativo referencia para sus proyectos.

4.6.2. Ubuntu

Ubuntu [9] es una distribución del sistema Linux basado en Debian. Está orientadoal usuario promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia del usuario.

Ubuntu es una bifurcación del código base del proyecto Debian. El objetivo inicial era hacer de Debian una distribución más fácil de usar y entender para los usuarios finales, corrigiendo varios errores de este y haciendo más sencillas algunas tareas como la gestión de programas. Su primer lanzamiento fue el 20 de octubre de 2004.

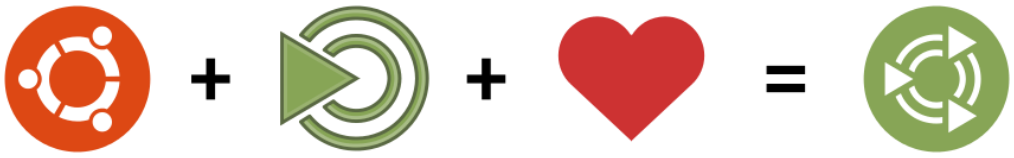
Ubuntu usa primariamente software libre, haciendo excepciones en el caso de varios controladores privativos (además de firmware y software). Antes de cada lanzamiento, se lleva a cabo una importación de paquetes, desde Debian, aplicando las modificaciones específicas de Ubuntu. Un mes antes del lanzamiento, comienza un proceso de congelación de importaciones, ayudando a que los desarrolladores puedan asegurar que el software sea suficientemente estable.

Desde el inicio del proyecto, Shuttleworth proporcionó el soporte económico gracias a los beneficios obtenidos después de vender su empresa Thawte a VeriSign, por unos 575 millones de dólares estadounidenses.

El 8 de julio de 2005, Shuttleworth anunció la creación de la Fundación Ubuntu y aportaron 10 millones de dólares como presupuesto inicial. El propósito de la fundación es el de asegurar soporte y desarrollo para todas las futuras versiones de Ubuntu.

A partir de la versión 9.04, se empezó a ofrecer soporte extraoficial para procesadores ARM. El 3 de junio de 2010, Mark Shuttleworth anuncia el trabajo en conjunto con el proyecto Linaro y su desarrollo de código abierto para Linux en procesadores con tecnología ARM. A fines de septiembre se da a conocer antes del lanzamiento de Ubuntu 10.10, que esta versión incluiría un mejor y más estable soporte para procesadores ARM.

Otra de las particularidades de Ubuntu es la cantidad de distribuciones que proporciona, diferenciándose principalmente por el entorno de escritorio y aplicaciones de base.



Para el uso sobre dispositivos Raspberry 2 y 3 destaca Ubuntu Mate [10]. Este proyecto fue cofundado por Martin Wimpress y Alan Pope. Martin es desarrollador del entorno de escritorio Mate y líder del proyecto de Ubuntu Mate. Alan por su parte, es desarrollador de Canonical en el proyecto de Ubuntu.

Entre el resto del equipo se observan multitud de desarrolladores de Debian, Ubuntu y Mate para entornos no ARM. Esta es una de las razones por la cual esta distribución toma una clara ventaja respecto a otras no oficiales, disponiendo de software muy actualizado y estable.

4.6.3. Fedora

Fedora [11, 12] es una distribución Linux para propósitos generales que también basada en software libre. A diferencia de las anteriores emplea paquetes basados en RPM, heredado de Red Hat, empresa que respalda y promociona esta distribución.

El Proyecto Fedora no solo busca incluir software libre y de código abierto, sino ser el líder en ese ámbito tecnológico. Algo que hay que destacar es que los desarrolladores de Fedora prefieren hacer cambios en las fuentes originales en lugar de aplicar los parches específicos en su distribución, de esta forma se asegura que las actualizaciones estén disponibles para todas las variantes de Linux.

El Proyecto Fedora nació a finales de 2003 cuando Red Hat Linux fue descontinuado. La empresa Red Hat prefirió continuar con el desarrollo de Red Hat Enterprise Linux (RHEL), relegando Fedora a un proyecto comunitario. A pesar de esto, las ramas de RHEL derivan de las versiones de Fedora que han demostrado su estabilidad en este entorno.

Fedora destaca por su seguridad, siendo de las distribuciones pioneras en incorporar SELinux. Este es un módulo de seguridad para el kernel Linux que proporciona el mecanismo para soportar políticas de seguridad para el control de acceso, incluyendo controles de acceso obligatorios como los del Departamento de Defensa de Estados Unidos. Se trata de un conjunto de modificaciones del núcleo y herramientas de usuario que pueden ser agregadas a diversas distribuciones Linux. Su arquitectura se enfoca en separar las decisiones de las aplicaciones de seguridad de las políticas de seguridad mismas y racionalizar la cantidad de software encargado de las aplicaciones de seguridad. Los conceptos clave que soportan SELinux pueden ser trazados a diversos proyectos previos de la Agencia de Seguridad Nacional de Estados Unidos.

El soporte para los dispositivos Raspberry 2 y 3 llegó a partir de la versión de Fedora 25, lanzada el 22 de noviembre de 2016. Por tanto, a pesar de tener un fuerte apoyo por parte de Red Hat y la comunidad, es posible que aún existan paquetes no optimizados para esta arquitectura.

4.6.4. Arch Linux

Arch [13, 14] es otra distribución basada en Linux orientada a usuarios avanzados, también caracterizada por su énfasis en el software libre y de código abierto. A diferencia de las distribuciones anteriores, el soporte es íntegramente comunitario, no habiendo entidades empresariales tras su mantenimiento de manera oficial.

El enfoque del desarrollo sigue el principio KISS, "Keep It Simple, Stupid!", que podría traducirse como "¡Hazlo sencillo, estúpido!". Esto da como resultado código elegante, exacto, minimalista y sencillo.

Arch Linux emplea un modelo de "Rolling reléase", es decir de liberación continua. Esto se refiere a sistemas en constante desarrollo y que no requieren reinstalarse sobre la versión anterior para mantenerse actualizados. En vez de esto, todos los binarios son actualizados sobre el sistema a medida que se actualizan. Esta distribución emplea su propio sistema de paquetes, PKG, los cuales son gestionados por su Packman, gestor escrito específicamente para Arch Linux.

4.6.5. Windows IoT core

Por último tenemos Windows IoT [15], originalmente Windows Embedded. Es una familia de sistemas operativos de la empresa Microsoft diseñada para su uso en sistemas embebidos y de entornos IoT. Microsoft tiene actualmente tres subfamilias diferentes de sistemas operativos para dispositivos integrados dirigidos a un amplio mercado, que van desde dispositivos de espacio reducido en tiempo real hasta dispositivos de punto de venta (PDV) como quioscos.

A diferencia de los entornos Linux, Windows IoT no dispone de una fuerte comunidad, por lo que la documentación disponible para este tipo de proyectos es bastante menor.

Así mismo, en ocasiones resulta difícil encontrar los binarios funcionales para esta plataforma, por su incipiente aparición.

4.6.6. Elección

Tras haber analizado y probado los sistemas previamente citados destacan particularmente dos: Ubuntu y Fedora.

Como primera comparativa Windows vs Linux, El primero parece encontrarse en un punto menos maduro e incipiente en comparación con el segundo. Para este tipo de dispositivos, con funcionalidades a medio camino entre IoT y TI convencionales, parece quedar atrás. Adicionalmente no se han encontrado algunos de los productos seleccionados en un estado funcional, como beats, eliminando la posibilidad de emplear estas piezas en el sistema.

Dentro de las distribuciones Linux, Arch Linux resulta una opción muy llamativa por su política de "rolling reléase" y la posibilidad de programar actualizaciones y olvidarse del mantenimiento de versiones del dispositivo. Por desgracia, el alto nivel de base que se requiere para la gestión de esta distribución, puede ser negativo ante una actuación sobre el dispositivo, pues no siempre se contará con usuarios de un nivel realmente avanzado en el tipo de empresas que buscamos proteger. Por este motivo se descarta Arch Linux para este proyecto.

Raspbian, a pesar de ser una de las distribuciones más extendidas en este tipo de proyectos, no cuenta con el respaldo de una comunidad u organización tan influyente como Canonical, Red Hat o la propia Debian. A pesar de no ser un problema en el momento actual, es posible que a la larga no disponga de un soporte tan eficiente como Ubuntu o Fedora, decantando la balanza por estas dos.

Por último, la decisión entre Ubuntu y Fedora. Estas dos distribuciones tienen un funcionamiento estable, cantidad de documentación, soporte por parte de comunidad y organizaciones, etc. Entonces ¿Por qué decantarse por una u otra?

En esta ocasión se toma como Sistema operativo Ubuntu. Las razones son de antigüedad en la plataforma ARM de Raspberry. Mientras que Fedora fue lanzado recientemente, a finales de 2016, Ubuntu lleva considerablemente más tiempo portado de manera oficial. Adicionalmente, su similitud con Raspbian hace que haya disponible multitud de documentación y binarios.

4.7. Sistema operativo de los servidores

La última pieza del sistema que queda por seleccionar y al mismo tiempo la que menos dependencias genera. Este sistema será el encargado de alojar en su interior la base de datos Elasticsearch y la interfaz gráfica Kibana.

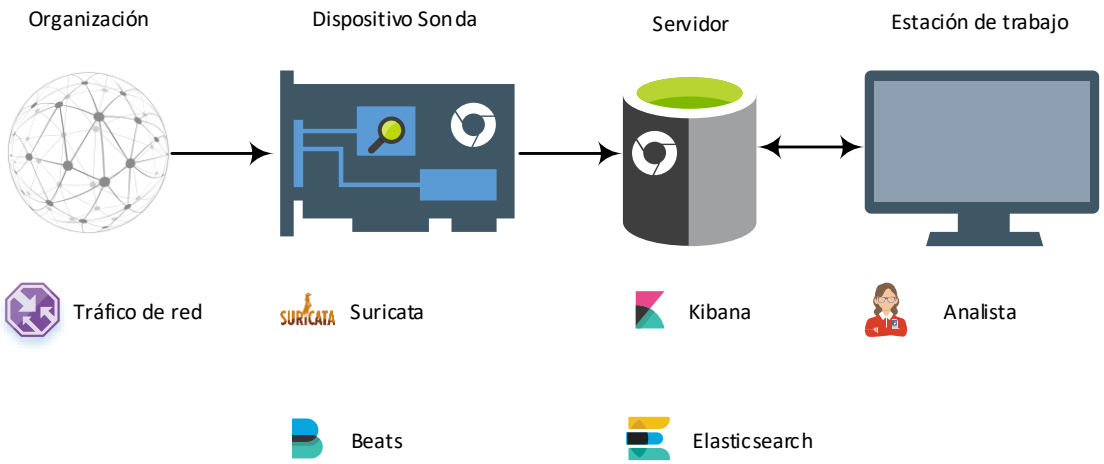
Dados los requisitos del software elegido, que resulta ser multiplataforma gracias a la capa intermedia ofrecida por la Máquina Virtual de Java, es posible emplear la mayoría de los sistemas operativos de uso común del mercado.

Por tanto, se elige una distribución basada en Debian/Ubuntu. Estas distribuciones son el equivalente a la seleccionada en el dispositivo sonda, por lo que no requiere conocimientos en diferentes sistemas por parte de los usuarios que gestionen los servidores. Así mismo son sistemas orientados a usuarios noveles en entornos Linux.

5. Arquitectura y flujo de datos

A lo largo de este apartado este apartado, se describe la arquitectura para el sistema de detección de intrusos basado en red resultante de la toma de decisiones a lo largo de los puntos anteriores.

A continuación, se adjunta una imagen con un único dispositivo por tipo. Cabe mencionar que esta arquitectura es únicamente a nivel explicativo y que, como se ha comentado antes, es posible multiplicar el número de nodos de cada tipo según las necesidades de cada infraestructura.



En primer lugar, tenemos el dispositivo sonda. Este recibirá el tráfico de red de la organización por su interfaz configurada en modo promiscuo. Para ello es necesario realizar el port mirror de los puntos de la infraestructura que se desee monitorizar, tal y como se explicado en el punto 3.1.

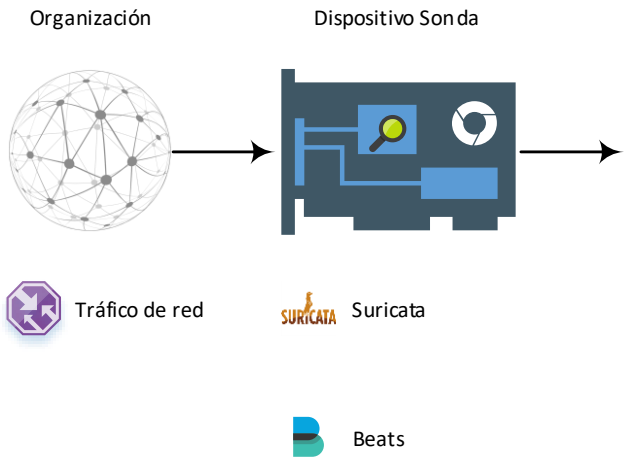
En el dispositivo sonda, el modo promiscuo de la interfaz de red acepta todo el tráfico entrante sin realizar filtrado por destino. Esto permite al motor de búsqueda de Suricata recibir todo el tráfico sin importar a quien va dirigido, siempre que se encuentre en la copia mandada a esta interfaz.

Como se ha comentado con anterioridad, se configurará Suricata para aprovechar al máximo las capacidades que ofrece un procesador con múltiples núcleos. Esto permite que el motor pueda levantar un hilo de ejecución por cada uno de ellos. El hilo principal, hará las veces de balanceador de carga, enviando cada alerta al nodo menos ocupado en cada instante de ejecución. Tendremos por tanto un sistema de análisis de tráfico concurrente que se estima efectivo en entornos de pequeñas y medianas empresas, para los que está orientado este proyecto.

Los reportes de alertas de Suricata únicamente permiten la salida a un fichero del sistema. Suricata permite varios formatos de salida. Para este cometido, se elige un formato JSON, pues se trata del más completo de todos. Otros formatos como Syslog, eliminan información y añaden la necesidad de interpretar el fichero compartido por otros procesos del sistema. Por otro lado, Unified2 se ve conveniente únicamente por su compatibilidad con barnyard2, indispensable en caso de haber elegido Graylog.

Una vez Suricata se encuentra volcando sus resultados en el fichero JSON será necesario enviarlo a la base de datos. Para ello, se ha elegido el agente ligero Beats, que se encargará de la ingesta en el nodo / cluster de Elasticsearch. La principal ventaja de este cliente ligero es que apenas interferirá en el rendimiento de los hilos de Suricata que se encuentren corriendo en ese mismo dispositivo sonda.

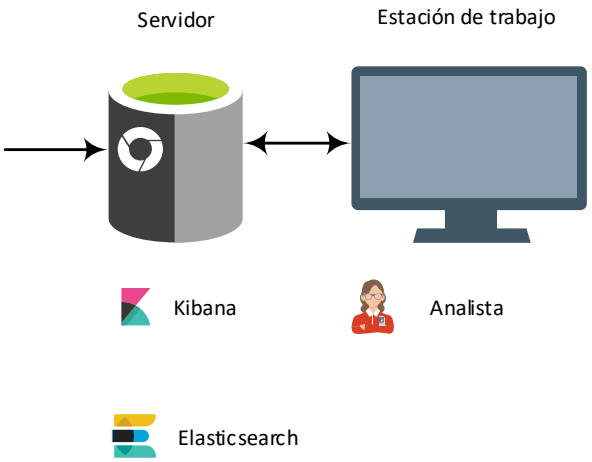
Dada la escasa capacidad de almacenamiento que tendremos en la Raspberry y para evitar duplicidades innecesarias en la información (Recordamos que la base de datos ya tiene mecanismos de copia de seguridad), se aplicará una política de rotado del fichero de alertas. Este rotado se generará de manera diaria, empaquetando alertas por días. El número de copias históricas que se almacenen dependerá de los requisitos de la organización, capacidad de almacenamiento y volumen de tráfico.



A continuación, se explica el funcionamiento de Elasticsearch. Tras recibir las alertas desde los diferentes dispositivos sonda, las instancias de Beats se encargan de procesar la información recibida y enviarla a la base de datos. En caso de que el sistema sea un cluster, el nodo maestro indicará el destino del fragmento original y la copia de respaldo.

Cuando se reciba una consulta, el nodo maestro de Elasticsearch se encargará de distribuir la carga de trabajo entre los diferentes nodos, consiguiendo otra vez un sistema concurrente de trabajo. A diferencia del paralelismo obtenido en los nodos sonda, Elasticsearch obtiene esta concurrencia gracias a nodos situados en diferentes máquinas. En caso de tener únicamente un nodo, este paralelismo no ocurrirá.

Por último, en el desempeño por parte del analista de su tarea de gestión de alertas, este accederá a Kibana. La interfaz gráfica Kibana será el software encargado de realizar las consultas necesarias a la base de datos para representar la información. Esta interfaz deberá ser correctamente configurada para exprimir el potencial de la herramienta. Para ello, se deberá tomar en consideración las necesidades de la organización, así como las del analista.



6. Implementación

Una vez definida la arquitectura del sistema de detección de intrusos basado en red en el apartado anterior, se realizará un despliegue de la arquitectura mínima de la plataforma.

Con este despliegue se desea obtener un laboratorio del cual extraer valiosa información del funcionamiento del sistema más allá del permisivo papel, donde todo funciona correctamente.

De este entorno se obtendrá un manual de despliegue probado, con puntos a tener en consideración tras instalaciones fallidas, deficientes o mejorables.

A lo largo de este apartado se describen los diferentes puntos de la instalación llevada a cabo.

6.1. Sistema operativo del dispositivo sonda

En primer lugar, se instalará el sistema operativo del dispositivo sonda, la raspberry Pi 3 Model B. Para ello, descargamos el sistema operativo de la página oficial de Ubuntu:

http://www.finnie.org/software/raspberrypi/ubuntu-rpi3/ubuntu-16.04-preinstalled-server-armhf+raspi3.img.xz

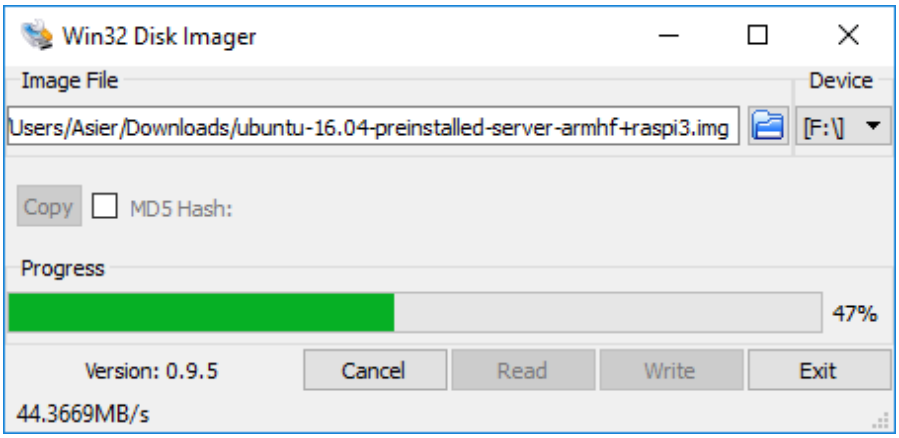
Este tipo de imágenes preinstaladas se encuentran optimizadas para este tipo de dispositivos. Para instalarla desde un sistema Windows, necesitaremos hacer uso de un software adicional para grabar el sistema en la tarjeta microSD que hace la función de almacenamiento en los dispositivos Raspberry.

El software elegido es Win32diskimager y es posible obtenerlo de la siguiente dirección:

https://sourceforge.net/projects/win32diskimager/

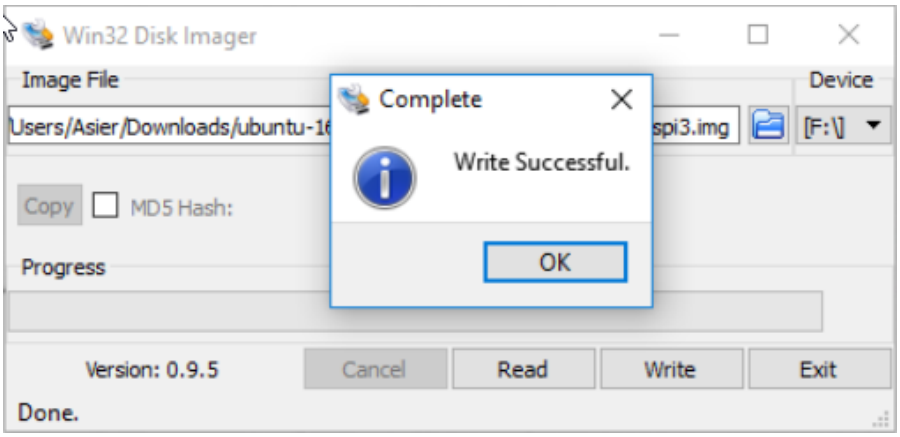
Tras realizar la instalación, aparecerá una ventana como la que se muestra a continuación de este párrafo. Para grabar la imagen, es necesario seleccionar la imagen descargada previamente. Para ello, se seleccionará el icono con la carpeta azul. En el dialogo que se abre, se debe elegir la ruta a la imagen.

A continuación, se debe seleccionar el punto de montaje donde se encuentra la tarjeta microSD. Bastará simplemente con seleccionar la letra en el menú desplegable que aparece a la derecha del icono antes mencionado.



Una vez se han seleccionado las opciones anteriores, se procede a grabar la imagen haciendo click sobre el botón “Write”. Este proceso puede tardar unos minutos, dependiendo del tamaño de la imagen y la tarjeta microSD.

Cuando el proceso termine, aparecerá un dialogo confirmándonos que el proceso ha terminado de manera satisfactoria.



Si en un futuro se desea hacer una copia del sistema, es posible realizarlo con la opción “Read” de esta misma herramienta. Esto puede ser de utilidad si se prevé que es posible corromper el sistema tras alguna acción.

Tras iniciar el sistema, se procederá a la configuración de este. La primera configuración a tener en consideración desde el punto de vista de este dispositivo sonda será el nombre del dispositivo. Esta configuración permitirá en el futuro identificar la fuente de las alertas generadas. Para realizar este cambio de manera permanente se deberá modificar los ficheros "/etc/hostname" y "/etc/hosts". El nombre debe ser identificativo, evitando problemas a la hora de diferenciar las fuentes. El elegido para este proyecto será "SRV-Ubuntu-Sonda1".

La otra tarea que se deberá llevar a cabo es la configuración de las interfaces de red. Como se ha comentado anteriormente, este dispositivo necesita la configuración de una interfaz de red en modo promiscuo, así como otra interfaz de red que se encargará de las comunicaciones generales y garantizar el acceso al dispositivo.

En este punto se detalla la configuración de la interfaz promiscua, encargada de ingestar el tráfico de la red. Por requisitos hardware, la interfaz encargada de esta tarea será la cableada, pues es la única de las interfaces que puede ponerse en modo promiscuo. Para ello, detectamos el nombre de la interfaz y modificamos el fichero "/etc/network/interfaces" para configurar esta de la siguiente manera:

```
auto XXX
iface XXX inet manual
  up ifconfig XXX promisc up
  down ifconfig XXX promic down
```

Dado que no es el objetivo del documento, el resto de configuraciones que no tienen impacto en el desarrollo del proyecto son omitidas por razones de espacio.

6.2. Sistema operativo del servidor

En este apartado se explica la instalación del sistema operativo huésped de la base de datos y la capa de presentación de nuestra infraestructura. Dado que se trata de una instalación reducida y buscando un bajo coste, se empleará una única máquina virtual para este ejemplo.

En primer lugar, se descargará la imagen elegida previamente. En el siguiente enlace se pueden encontrar las imágenes de Ubuntu Server. Se ha elegido esta versión por su soporte extendido hasta abril de 2021.

https://www.ubuntu.com/download/server

Una vez descargada, se procede a crear la máquina virtual. La asignación de recursos a esta máquina ha sido determinada por la cota superior de anteriores proyectos e instalaciones de sistemas similares. Otro factor influyente ha sido la máquina en la que se aloja. En cualquier caso, estos valores pueden variar dependiendo de las necesidades del cliente.

Procesador:	Intel Core i5-7400 (4/8 cores)
RAM:	8 GB
ROM:	40 GB SSD

Al crear la máquina, se le asigna un nombre identificativo. En este caso el nombre será “SRV-Ubuntu-NIDS”. Esto indica que se trata de un servidor, que funciona sobre un sistema Ubuntu y que aloja servicios de nuestro sistema de detección de intrusos basado en red.

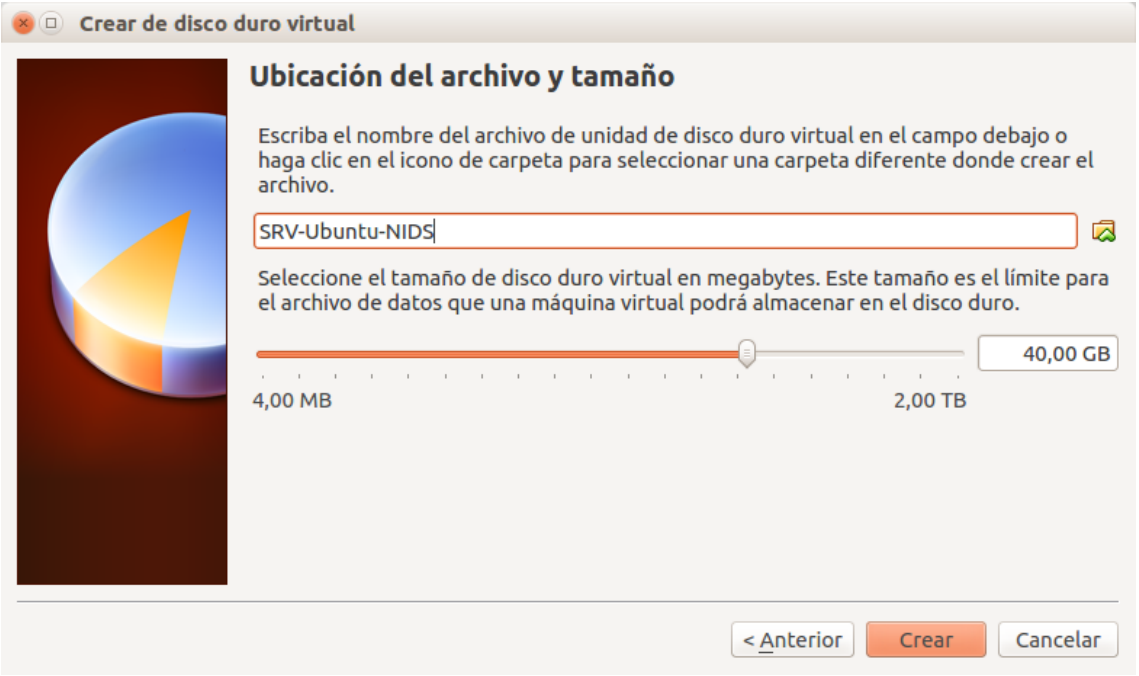
La nomenclatura elegida en una empresa real deberá ser, como en el caso de los dispositivos sonda, elegida teniendo en consideración la nomenclatura que pueda tener la organización para identificar sus sistemas.

En el siguiente paso, se selecciona la memoria RAM. Es importante remarcar que los sistemas que ejecutan consultas sobre bases de datos de tamaño considerable, tienden a ser muy dependientes de este recurso.

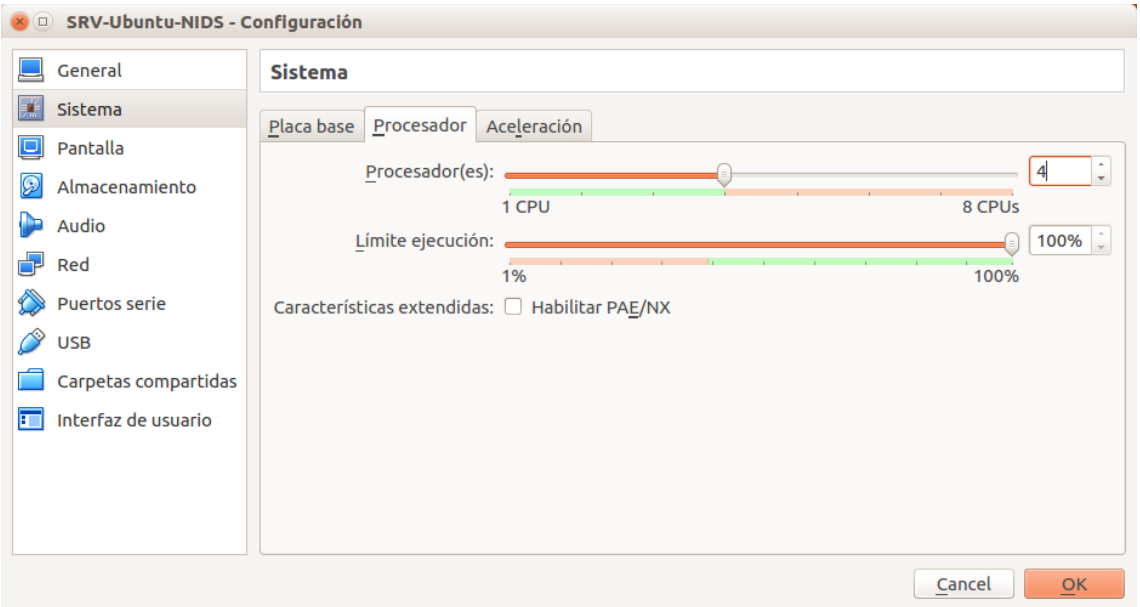
Adicionalmente, estos sistemas sufren una penalización de rendimiento importante cuando la base de datos desborda la memoria volátil y pasa a ser parcial o totalmente almacenada en la partición swap. El acceso a esta partición que se encuentra en el disco duro es considerablemente más lento, reduciendo drásticamente la respuesta de la aplicación.

Por esta razón, se ve oportuno otorgar 8 Gigabytes de memoria RAM a este servidor.

A continuación, se creará el disco duro de la máquina. El volumen elegido es bastante reducido, 40 Gigabytes, seleccionado para esta instalación por su carácter de investigación.

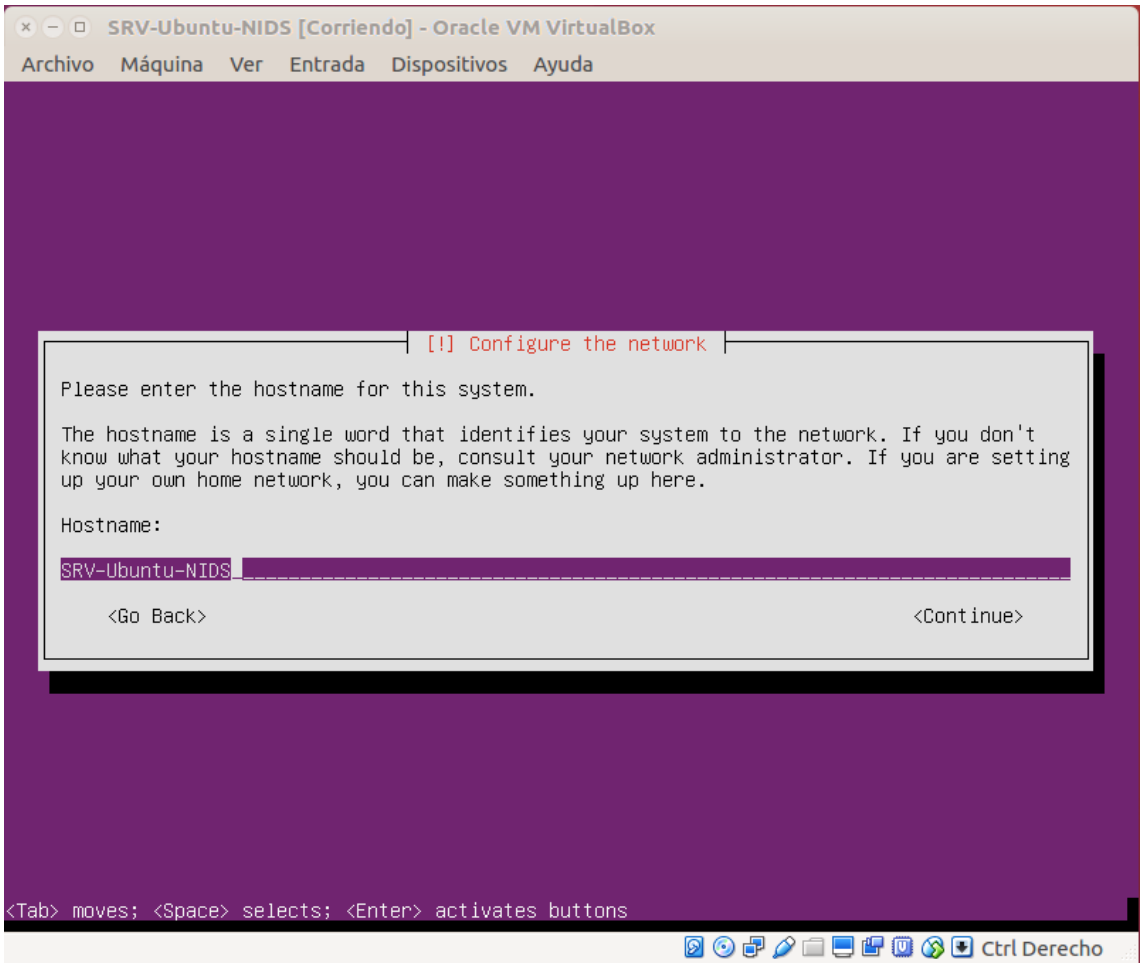


Por último, se accede a la configuración de la máquina. En el apartado “sistema”, se accede a la pestaña “procesador” y añadimos 4 CPUs. Esto permitirá el funcionamiento de nuestra aplicación con suficiente fluidez.



Una vez creada la máquina y asignados los recursos, se procede a la instalación del sistema operativo. Por razones obvias, solo se detallarán los puntos de la instalación que se consideren importantes en el proceso, evitando puntos que no repercutirían en el sistema como selecciones de idiomas, etc.

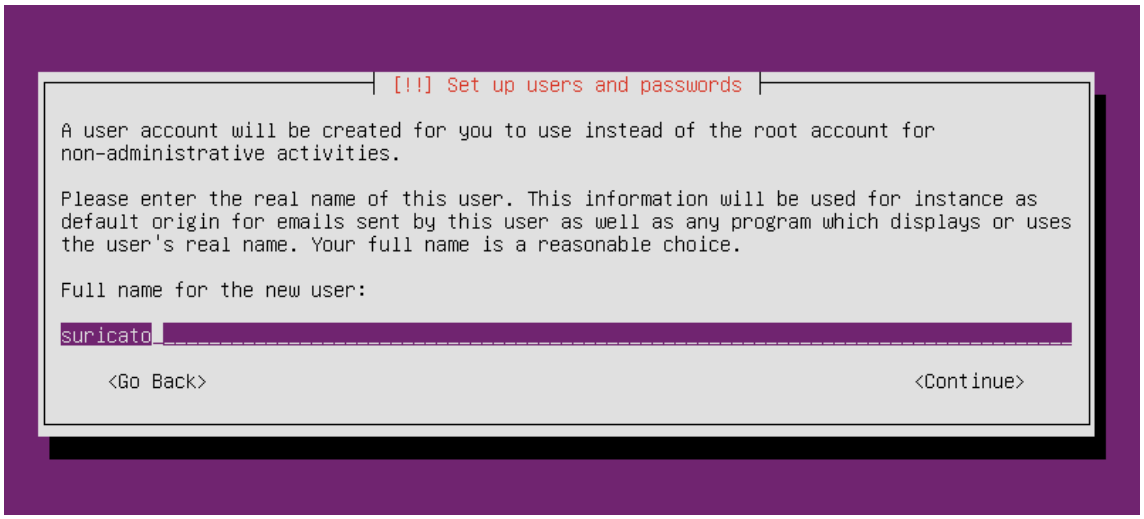
El primer punto de interés que se debe abordar es el nombre de la máquina. Dado que ya ha sido elegido anteriormente, nombramos el nodo como “SRV-Ubuntu-NIDS”.

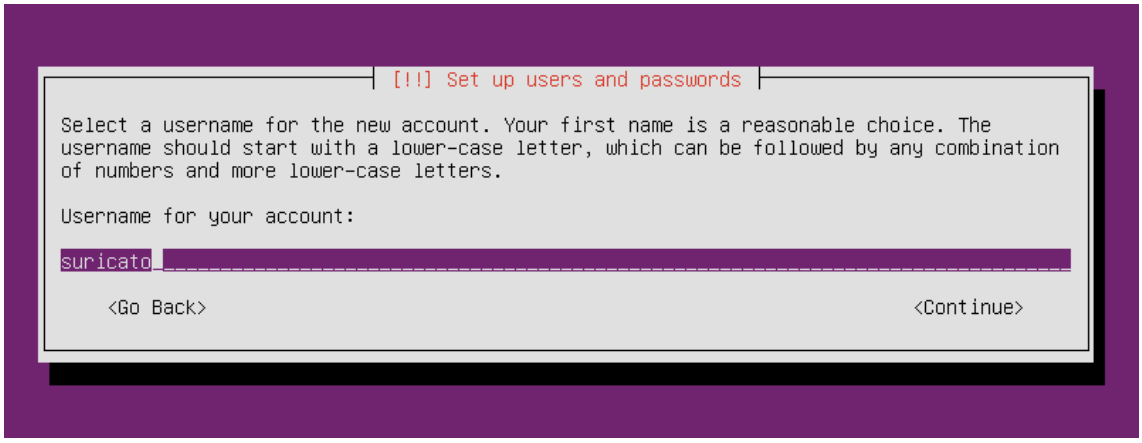


En el siguiente punto a tratar, crearemos un usuario diferente de “root” para acceder al sistema. Es importante no emplear “root” como usuario por defecto para todas las tareas, evitando su uso en la medida de lo posible. Esto se debe a que cualquier acción errónea realizada por este supe usuario puede tener grabes repercusiones sobre el sistema y, además, puede ser realizada sin necesidad de confirmación. En su lugar emplearemos ordenes con el prefijo “sudo”, e introduciremos la credencial necesaria.

Otro punto a tener en cuenta es que, en caso de ocurrir un fallo de seguridad en cualquiera de los módulos implementados en el sistema, un atacante remoto podría ejecutar código arbitrario con privilegios de super usuario.

En este caso, se crea el usuario “suricato” que tendrá los permisos previamente mencionados.





Uno de los puntos importantes a tener en cuenta desde el punto de sistemas es el particionado del disco duro. A continuación, se detalla cada partición y las razones por las que se ha elegido dimensionar con ese tamaño para que, en caso de tener un sistema de características diferentes, se tenga una noción de los tamaños a elegir.

Partición swap

En primer lugar, tenemos la partición de intercambio o swap. Esta partición es empleada cuando el sistema considera que es necesario un extra de memoria RAM. El sistema operativo envía parte de la memoria RAM al disco duro para liberar espacio.

Como se ha comentado en el apartado de dimensionamiento de la RAM, este tipo de sistemas de acceso rápido a base de datos sufren grabes penalizaciones de rendimiento con el acceso a información almacenada en esta partición. Por esta razón, se configurará el despliegue para no hacer uso del área de intercambio por parte de estas aplicaciones.

La partición, por tanto, se dimensiona únicamente con 2 Gigabytes de capacidad, pues solo el sistema operativo y la aplicación Kibana tendrán la posibilidad de emplear esta área de la memoria.

Partición /boot

Este directorio contiene todo lo necesario para que funcione el proceso de arranque del sistema. “/boot” almacena los datos que se utilizan antes de que el kernel comience a ejecutar programas en modo usuario.

Por diversas razones como seguridad y estabilidad, es conveniente tener una partición separada para este directorio.

No se ve necesario, en ningún caso, tener una partición superior a 1 Gigabyte de capacidad para esta clase de instalaciones. Por tanto, el tamaño será fijo en cualquier servidor de estas características.

Partición /

Partición principal del sistema. En esta partición se almacenarán los binarios del sistema y aplicaciones del despliegue. Una vez el despliegue ha finalizado, esté tipo de sistemas no deberían ocupar más de 5 Gigabytes de memoria.

También se encuentran los directorios personales de los usuarios. Estos no deberían tener un impacto sobre la capacidad disponible, dado que no se esperan usuarios trabajando en esta máquina.

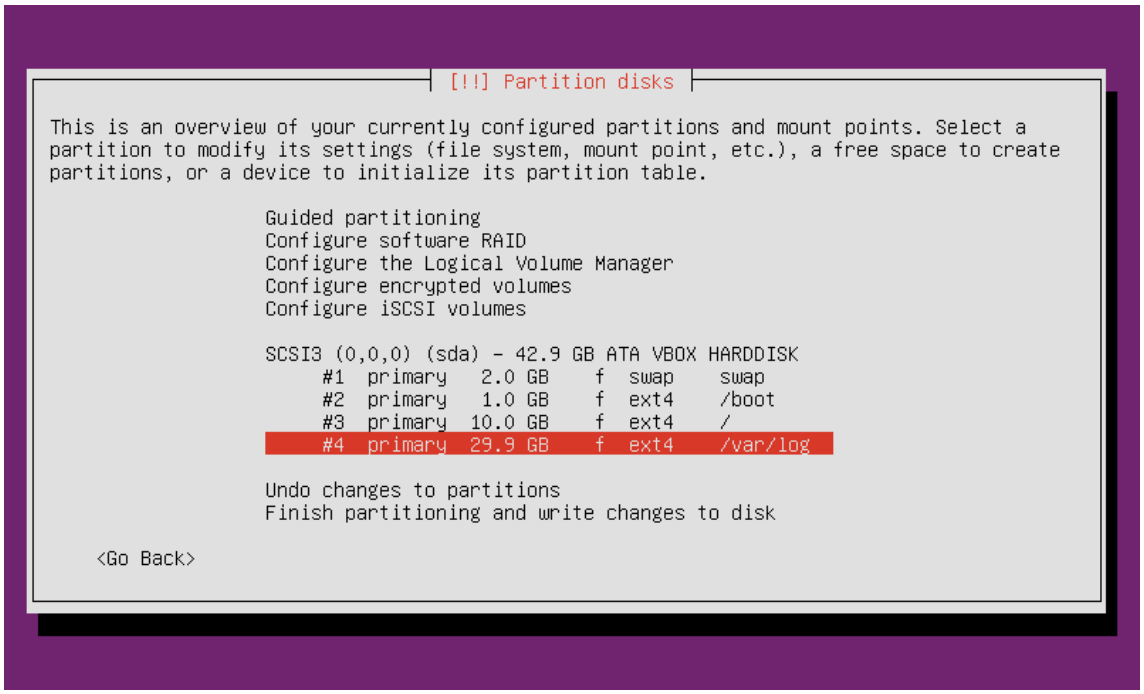
Por esta razón, se ha decidido otorgar 10 Gigabytes de capacidad a esta partición.

Partición /var/log

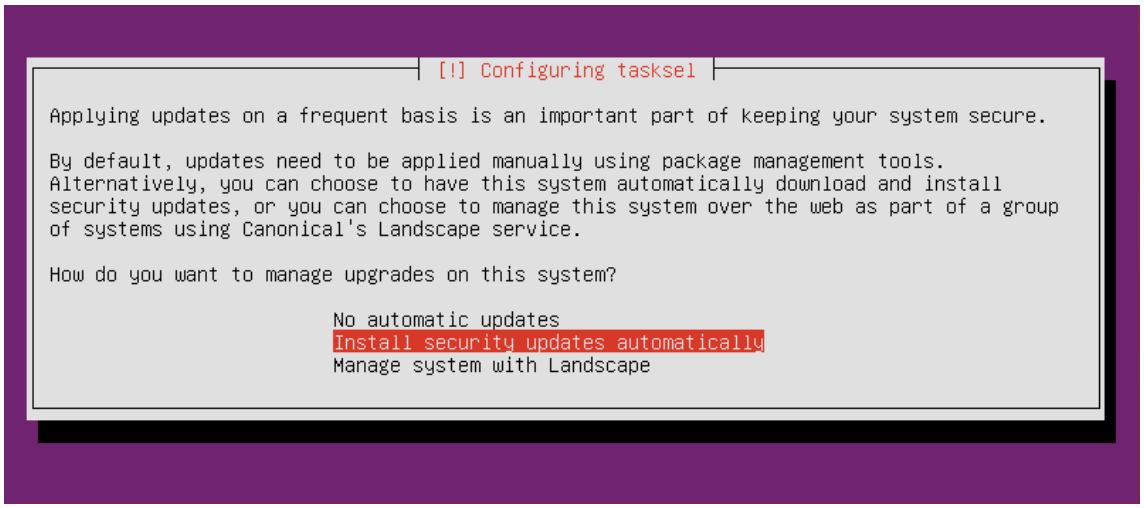
Por último, tenemos la partición "/var/log". Esta partición suele montarse separada del resto en sistemas cuyo principal objetivo es el almacenado y procesado de registros. La compartimentación de este directorio evitará problemas en caso de un volumen excesivo de registros.

En nuestro caso, la aplicación elasticsearch emplea este directorio como almacén de registros. Por esta razón, se dimensiona esta partición con 30 Gigabytes de capacidad. En futuros despliegues, esta partición deberá ser objeto de estudio según las necesidades de retención de eventos del cliente y el volumen de alertas que genere el sistema.

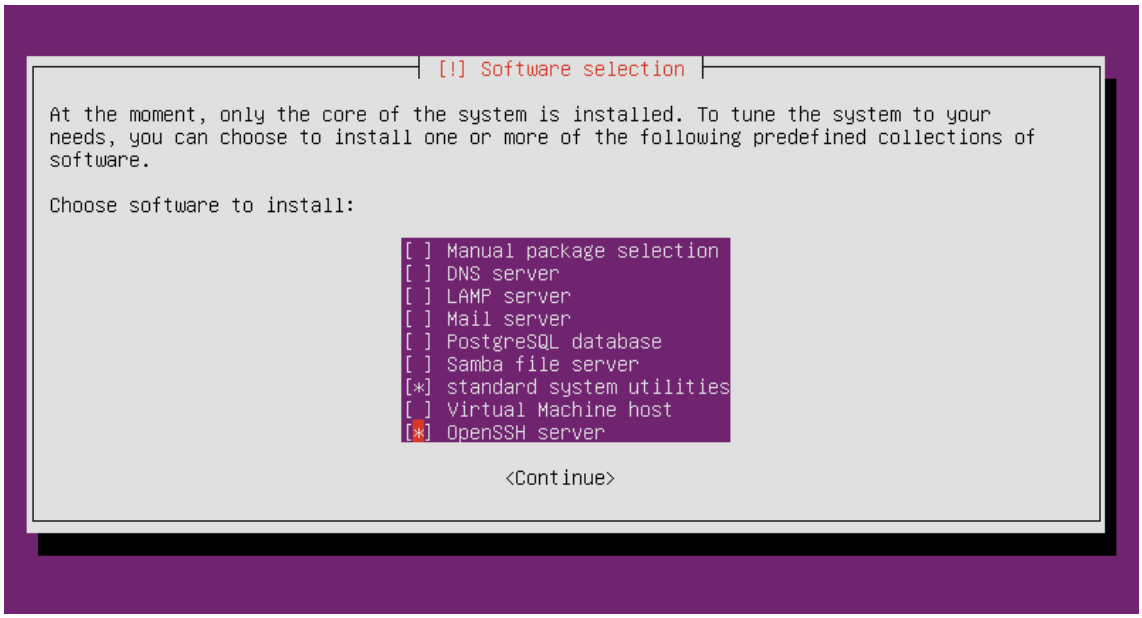
El resultado de las particiones debería quedar así:



Para concluir, y dada las características del sistema, se ve oportuno activar las actualizaciones de seguridad automáticas en este servidor.



Como servicios, únicamente necesitaremos añadir el servidor OpenSSH con el fin de facilitar las tareas administrativas sobre esta máquina, sin necesidad de acceso físico a ella. Tras estos pasos, terminamos la instalación del sistema con normalidad.



Para concluir con la instalación, solo quedará la tarea de asignar una dirección IP fija a la interfaz del servidor que vayamos a emplear para la comunicación con los dispositivos sonda.

6.3. Elasticsearch

A continuación, con el sistema operativo del servidor instalado, se procederá con el despliegue de la herramienta Elasticsearch. En primer lugar, se ha de subsanar la necesidad del software que se pretende desplegar en esta máquina, la JVM o Máquina Virtual Java.

Viendo la documentación de elastic, se recomienda hacer uso de la versión 1.8 de Java. Para obtenerla [17] realizamos los siguientes pasos:

En primer lugar, añadimos el repositorio de Oracle donde se encontrará el paquete:

```
sudo add-apt-repository ppa:webupd8team/java
```

Acto seguido, actualizamos los repositorios e instalamos el paquete necesario:

```
sudo apt update
sudo apt install oracle-java8-installer
```

Para comprobar que la instalación ha sido correcta, introducimos el siguiente comando:

```
java – versión
```

Esta orden debe indicar la versión de Java 1.8.

Una vez subsanado el prerrequisito, se continua con la instalación de Elasticsearch.

En primer lugar, es necesario importar la clave pública de Elastic. Para ello, introducimos las siguientes órdenes:

```
# wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
# echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" | sudo tee -a
# sudo apt update
# sudo apt -y upgrade
```

Y acto seguido instalamos el paquete de elasticsearch:

```
# sudo apt -y install elasticsearch
```

Las siguientes ordenes lanzaran el demonio y se encargaran de que inicie con el arranque del sistema:

```
# systemctl start elasticsearch
# systemctl enable elasticsearch.service
```

Una vez instalado, pasamos a la configuración del servicio. Esta configuración se realiza mediante la edición del fichero "/etc/elasticsearch/elasticsearch.yml".

```
// Editamos la variable cluster.name. Todos los nodos del cluster tendran el mismo
// nombre.


cluster.name: elasticStohis


// En la variable network.host insdicamos la dirección IP del nodo actual.


network.host: X.X.X.X


// La variable node.name será el nombre que reciba el host dentro del cluster.


node.name: ${HOSTNAME}


// Los siguientes campos sirven para el descubrimiento de otros nodos, indicando su
// nombre o dirección IP (segun tengamos DNS apuntando a ellos o no).


discovery.zen.ping.unicast.hosts: ["X.X.X.X", "X.X.X.X"]

discovery.zen.ping.unicast.hosts: ["node01", "node02", "node03"]


// Por último definimos la variable que hace referencia al descubrimiento de nodos
// maestro, recordando que esta variable debe ser (master_eligible_nodes / 2) + 1


discovery.zen.minimum_master_nodes
```

Tras realizar los cambios de configuración, podemos reiniciar el servicio:

```
# systemctl restart elasticsearch
```

Los siguientes comandos pueden resultar útiles para realizar comprobaciones de estado. El primero realiza una petición de datos almacenados. El segundo, nos servirá para comprobar el estado de los nodos dentro del cluster:

```
# curl -XGET 'http://192.168.204.131:9200/_search?pretty=true&q=*:*'
```

```
# curl -XGET 'http://X.X.X.X:9200/_cluster/state?pretty'
```

6.4. Suricata

Una vez instalada la base de datos, se comenzará la instalación de la parte del sistema que alimentará esta.

De vuelta en el dispositivo sonda, se procede a la instalación del motor de análisis de tráfico de red, suricata. Para esto, los sistemas Linux basados en Devian tienen un paquete en sus repositorios, por lo que es tan sencillo como ejecutar:

```
sudo apt -y install suricata
```

Y se añadirá la siguiente línea en el fichero "/etc/rc.local" para iniciar el demonio suricata al inicio del sistema, siendo XXX la interfaz que previamente se ha configurado en modo promiscuo. Esta permitirá que suricata se ejecute con las opciones de configuración que deseemos.

```
suricata -c /etc/suricata/suricata-debian.yaml -i XXX -l /var/log/suricata/ -D
```

Una vez instalado, será momento de pasar a la configuración básica de suricata [16]. Editamos el fichero "/etc/suricata/suricata.yaml", que como el anterior, sigue las directrices del formato yml. En este fichero se buscará y modificará las siguientes variables según nuestras necesidades:

```
// en el apartado file:
enabled: yes
filename: /var/log/suricata/suricata.log

// en af-packet escribimos nuestra interfaz promiscua:
interface: XXX

// Buscamos la variable HOME_NET e introducimos la red que queremos monitorizar
HOME_NET: "[X.X.X.X/XX, X.X.X.X/XX]"
```

Pero Suricata trae consigo una serie de necesidades extras que deberán ser cubiertas para un correcto funcionamiento del sistema. El primero de ellos que se debe abordar es la gestión de firmas del motor de detección. Estas firmas, como si de un antivirus se tratase, deben ser actualizadas para incluir las relacionadas con las últimas amenazas. Para ello, se empleará un gestor de firmas, Oinkmaster [18]. Para incluir este en nuestro sistema se procederá como a continuación se indica:

```
sudo apt -y install oinkmaster
```

Este gestor apunta a un repositorio que se encargará de distribuir, modificar o eliminar estas firmas de detección. Para este proyecto, se ha decidido implementar las reglas gratuitas de Emerging Threats [19], que vienen por defecto configuradas para Suricata. Para ello, nos dirigimos al fichero /etc/oinkmaster/oinkmaster.conf" y comentamos las líneas que hacen referencia a los repositorios de Snort que vienen configurados por defecto. En su lugar, incluimos la siguiente línea:

```
url = http://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz
```

Adicionalmente, se comprobará la existencia de las líneas citadas a continuación. En caso negativo, se incluirán en dicho fichero:

```
classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
```

Para comprobar el correcto funcionamiento de la herramienta o en caso de querer lanzarla de forma manual, se deberá introducir la siguiente orden en la línea de comandos:

```
oinkmaster -C /etc/oinkmaster.conf -o /etc/suricata/rules
```

Para asegurar un correcto análisis del tráfico, las firmas deberán estar actualizadas diariamente. El mantenimiento de estas se antoja difícil si no imposible en escenarios con múltiples sondas, además de suponer un gasto de tiempo en el administrador de los sistemas. Por ello se configurará una tarea en el sistema que se ejecutará diariamente a las 4:00 AM. En primer lugar abrimos el editor de tareas Crontab:

```
crontab -e
```

Una vez en el editor, introducimos la siguiente línea:

```
0 4 * * * /etc/oinkmaster/oinkmaster.pl -C /etc/oinkmaster/oinkmaster.conf -o
/etc/suricata/rules
```

Por último, comprobamos las tareas programadas en el sistema:

```
crontab -l
```

6.5. Beats

Llega la hora de integrar la última pieza de este dispositivo sonda. A diferencia del resto, esta pieza no tiene disponibilidad en repositorios para plataformas ARM. Por tanto, se deberá de descargar e instalar esta utilidad manualmente.

La primera de las tareas será determinar la última versión publicada en el repositorio de desarrolladores. El directorio donde se almacenan se encuentra en la URL "https://beats-nightlies.s3.amazonaws.com/" Este repositorio tiene versiones estables desde hace meses, pero es conveniente leer comentarios de la versión previamente para asegurarnos de que no tiene ningún fallo detectado. Concretamente se escogerá la última versión cuyo nombre termina en "filebeat-linux-arm":

```
wget https://beats-nightlies.s3.amazonaws.com/jenkins/filebeat/XXX/filebeat-linux-arm
```

A continuación, se prepararán los directorios necesarios para la instalación:

```
sudo mkdir /opt/filebeat
sudo mkdir /etc/filebeat
sudo mv filebeat-linux-arm /opt/filebeat/
sudo chmod +x /opt/filebeat/filebeat-linux-arm
```

En algunos casos, se requiere de la descarga de ficheros adicionales para su funcionamiento. Como solución se accederá al repositorio de Github de elastic. En este caso se descargaron y añadieron los siguientes ficheros a "/opt/filebeat/:

```
https://github.com/elastic/beats/blob/master/filebeat/filebeat.template.json
https://github.com/elastic/beats/blob/master/filebeat/filebeat.template-es2x.json
```

Dada que esta instalación no se realiza con medios automáticos, debemos crear el fichero para lanzar la utilidad a mano. Para esto creamos el fichero "/lib/systemd/system/filebeat.service" con el siguiente contenido:

```
[Unit]
Description=filebeat
Documentation=https://www.elastic.co/guide/en/beats/filebeat/current/index.html
Wants=network-online.target
After=network-online.target

[Service]
ExecStart=/opt/filebeat/filebeat-linux-arm -c /etc/filebeat/filebeat.yml
Restart=always

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

Tras esto, la ejecución de la herramienta quedará como otra cualquiera, ejecutando las siguientes órdenes para activarlo al inicio y lanzar el servicio, respectivamente:

```
systemctl enable filebeat
systemctl start filebeat
```

Para completar la integración del dispositivo con la infraestructura, debemos indicar a Beats que reenvíe la información que genera Suricata en "/var/log/suricata/eve.json", si no se le cambia el nombre, a nuestro clúster de Elasticsearch. Para llevar a cabo esta tarea crearemos el siguiente fichero de configuración, "/etc/filebeat/filebeat.yml":

```
filebeat:
  prospectors:
    -
      paths:
        - /var/log/suricata/xxx.json
      input_type: json
      document_type: suricata
      scan_frequency: 1s

    -
      paths:
        - /var/log/syslog
      input_type: log
      document_type: syslog
      scan_frequency: 1s

output:
  elasticsearch:
    hosts: ["elk-hostname:9200"]
    index: "filebeat"
```

Tras esto, puede ser interesante emplear los comandos indicados para depurar en el apartado de Elasticsearch con el fin de observar si los datos son ingestados correctamente en la base de datos.

6.6. Kibana

De vuelta en el servidor, en este apartado se describe la instalación de la última pieza de software de este, y del proyecto hasta el punto que se ha decidido presentar. Como el resto de aplicaciones que pertenecen a elastic, podemos instalar Kibana desde el repositorio configurado anteriormente para Elasticsearch. En caso de no haberlo añadido anteriormente, se escribirá:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt -y install kibana
```

Acto seguido se ejecutará y se añadirá al arranque del sistema:

```
systemctl start kibana
systemctl enable kibana.service
```

La configuración se encuentra en el fichero "/etc/kibana/kibana.yml" en el que tendrá que editar las siguientes variables para apuntar al servidor de Elasticsearch, que en este caso es el mismo en el que se hallará kibana:

```
// Se detallan el host y puerto donde escucha kibana y el servidor de elastic donde
// realizará las consultas. El puerto indicado es el que se define por defecto.
server.host: "X.X.X.X"
server.port: "5601"
elasticsearch.url: "http://X.X.X.X:9200"
```

Una vez tenemos la aplicación instalada, es hora de cargar los “dashboards”, o presentaciones, que se encargaran de mostrar de una manera amigable los datos recogidos en la base de datos.

Para ello, en primer lugar, nos descargamos un paquete prediseñado de elastic:

```
curl -L -O https://download.elastic.co/beats/dashboards/beats-dashboards-1.1.0.zip
```

A continuación, desempaquetamos el fichero descargado y ejecutamos el script para cargarlos que se encuentra en su interior:

```
unzip beats-dashboards-*.zip
cd beats-dashboards-*
./load.sh
```

Lo único que quedará por hacer es seleccionarlo en la interfaz gráfica como vista por defecto.

7. Evaluación

Hasta este punto únicamente se ha explicado el funcionamiento de la herramienta y su despliegue. En este capítulo se explicará los resultados obtenidos de implementar esta solución en dos escenarios diferenciados: Un laboratorio de pruebas y un cliente.

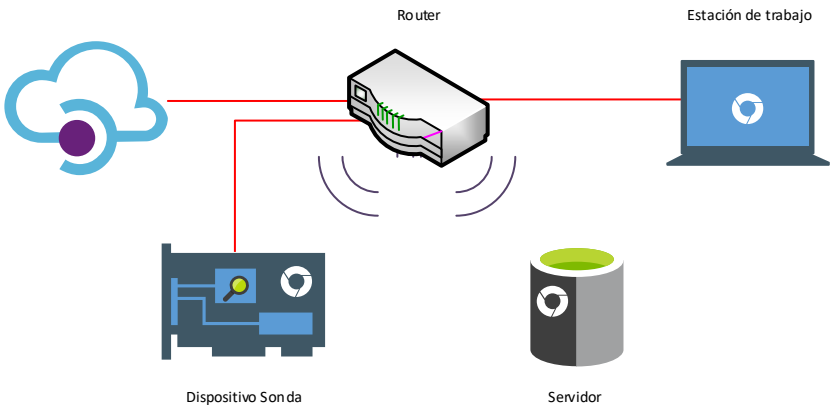
Para el primer caso se detallará la arquitectura empleada, pues se trata de un laboratorio montado para tal fin. Respecto a la implementación en cliente, no se ha permitido el uso de información identificativa de este por motivos de confidencialidad. A pesar de esto, si nos permite revelar las conclusiones extraídas de este montaje.

7.1. Escenario 1: Laboratorio

En este apartado se describirá el primer escenario, el laboratorio, y los resultados obtenidos de esta experiencia.

En primer lugar, se desplegó un sencillo laboratorio que consistía en una única estación de trabajo y un router con capacidad de port mirroring. El objetivo es que todo el tráfico generado por la estación de trabajo fuera copiado por otra salida, ya fuera entrante o saliente. Estas comunicaciones indicadas en rojo fueron cableadas.

Sobre esta pequeña arquitectura, se integró la infraestructura más básica posible de nuestro sistema de detección de intrusos basado en red: un servidor y un único dispositivo sonda. Estas comunicaciones fueron implementadas de manera inalámbrica y separadas de la red anterior.



Inicialmente se realizaron pruebas básicas: Contacto con dominios maliciosos. Estas pruebas sirvieron para demostrar el correcto funcionamiento de la infraestructura.

Posteriormente, fueron empleadas capturas de tráfico malicioso diseñadas para probar estos sistemas. Estas capturas [20], llamadas Pcaps, pueden ser descargadas libremente de internet en páginas como “http://www.netresec.com/?page=PcapFiles”.

Mediante la herramienta Tcpreplay se enviaron desde la estación de trabajo estas capturas con el objetivo de probar el sistema con volúmenes mayores.

```
tcpreplay --intf1=eth0 XXX.pcap
```

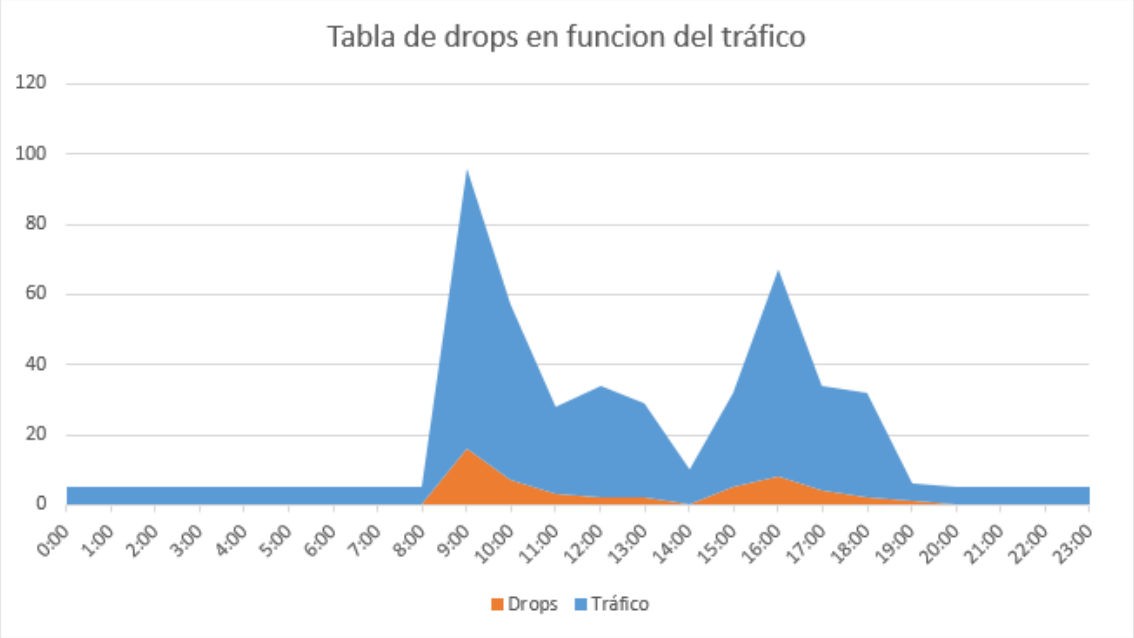
Tras múltiples pruebas, el resultado fue francamente positivo, encontrando en todo momento un drop de 0% respecto a los paquetes analizados.

7.2. Escenario 2: Infraestructura de cliente

Posteriormente, gracias a la relación mantenida con el departamento de tecnologías de la información de una empresa, fue posible desplegar un piloto en su infraestructura. Se trata de una pequeña empresa, con un volumen inferior a 50 empleados, del sector primario alimenticio.

El piloto desplegado, tenía las mismas características que el anterior: una sonda y un servidor. Este piloto fue monitorizado durante un mes para obtener estos datos promedio, empleando únicamente días laborales.

A continuación, se puede observar un gráfico del porcentaje de drops respecto al tráfico reenviado por parte del router encargado del port mirroring. En azul se indica porcentaje de tráfico de red que llega frente al límite teórico que la tarjeta de red del dispositivo puede procesar. En naranja observamos el porcentaje de drops que ofrece el sistema frente al tráfico analizado.



Únicamente se detectan dos picos de tráfico en los cuales una única sonda resulta insuficiente para abarcar todo el tráfico de la organización.

El primer pico se sitúa de 9:00 a 10:00, coincidiendo con el inicio de la jornada laboral llegando a un 16% de drops sobre el tráfico enviado. El segundo ocurre tras el descanso para comer de medio día, empezando poco antes de las 16:00 y terminando media hora después. Esta vez la tasa disminuye a la mitad, un 8%.

Por lo general, el resto del tiempo la tasa de drops es inferior al 3% en todos los casos, siendo esta cifra despreciable.

7.3. Resultado de la evaluación

Tras analizar el resultado obtenido de las pruebas realizadas en los dos escenarios elegidos para tal fin, llega el momento de la conclusión. Si bien es cierto que un único dispositivo sonda puede tener problemas para monitorizar el tráfico de una organización en hora punta, el bajo coste de estos permite desplegar varios dispositivos. Este crecimiento horizontal permite suplir estos problemas de rendimiento por un precio de aproximadamente 40€ por dispositivo, muy inferior al que puede costar una sonda empleada en una gran empresa.

Por tanto, este proyecto cumple con las necesidades planteadas en su origen: ofrece un sistema de detección de intrusos basado en red, de bajo coste de implantación, sencillo de desplegar y capaz de mejorar las capacidades en materia de ciberseguridad de una PYME.

8. Conclusiones y trabajo futuro

Con este proyecto, se ha demostrado que en la actualidad es posible implementar sistemas de seguridad enfocados a PYMES con un coste asequible para estas. Raspberry es una de las plataformas más extendidas para el desarrollo de este tipo de proyectos a nivel doméstico. Gracias a este tipo de proyectos se ve la viabilidad de usar estos dispositivos en entornos empresariales reducidos.

Otro de los objetivos del proyecto ha sido analizar las mejores opciones que ofrece el mercado, en software libre, para llevar implementar un sistema de detección de intrusos basado en red con logs centralizados. Mediante el estudio realizado, se ha llegado a la conclusión de que el mejor motor de análisis de tráfico basado en firmas en estos momentos es Suricata.

También se determina que la conocida pila de elastic formada por Beats, Elasticsearch y Kibana son la mejor opción para abordar los apartados de persistencia de los datos y presentación de la información. Esto se debe a las prestaciones del sistema y la multitud de presentaciones de la información posibles. A esto se le añade la simplicidad de integración entre los componentes, dado que son trabajados por el mismo grupo de desarrolladores.

A lo largo de la memoria se detallan los pasos necesarios para implementar este sistema. En estos apartados se abordan las configuraciones básicas. Gracias a esto, cualquier PYME podrá montar su propio sistema de detección de intrusos basado en red, adaptándolo según sus necesidades de escalabilidad horizontal.

Por último, se ha probado el rendimiento de este sistema con un único dispositivo sonda y una máquina virtual haciendo el papel de servidor centralizado. El resultado de esta prueba es satisfactorio, permitiendo el análisis de diversas capturas de tráfico con una tasa de drop de 0%.

Como segunda prueba de rendimiento, se ha monitorizado el tráfico de una PYME anónima. Un único dispositivo sonda ha sido suficiente para esta tarea durante la mayor parte de la jornada laboral, viéndose únicamente afectado en dos picos de tráfico ocurridos a la entrada de los trabajadores.

Tras haber analizado el cumplimiento de la infraestructura implementada, se han detectado algunas mejoras que, a pesar de quedar fuera del alcance del proyecto inicialmente propuesto, se ve necesario continuar.

El primero de los puntos a mejorar es la posibilidad de implementar un acceso identificado mediante credenciales personales a la interfaz web de Kibana. Si bien en la comparativa con Graylog comentábamos que esto era uno de los puntos a favor de esta segunda herramienta, no era decisivo. Esto se debe a que existen métodos alternativos de gestionar las credenciales.

En el futuro, se analizará la posibilidad de implementar un proxy inverso para controlar el login de los analistas en la interfaz de Kibana. Previsiblemente, y a falta de estudiar alternativas, se empleará el proxy Nginx.

Otro punto a tener en cuenta será la securización de las comunicaciones entre diferentes nodos de la infraestructura. Para el despliegue de este modelo, se han empleado entornos de laboratorio aislados en los que la seguridad de la infraestructura no corría ningún peligro. Como trabajo futuro se pretende securizar, mediante el uso de canales cifrados, las comunicaciones existentes entre nodos.

Por último, un trabajo a tener en cuenta al hacer frente a una organización con una mayor envergadura serán los sets de reglas a analizar [16]. Por desgracia, esta configuración ha de depender de los requerimientos de la organización y no de los recursos de este dispositivo sonda, por lo que variará en función del escenario.

9. Bibliografía

[1] Página web de Raspberry:

https://www.raspberrypi.org/

[2] Página web de Snort:

https://www.snort.org/

[3] Página web de Suricata:

https://suricata-ids.org/

[4] Comparativa entre motores Snort y Suricata:

http://wiki.aanval.com/wiki/Snort_vs_Suricata

[5] Página web de Elastic:

https://www.elastic.co/

[6] Página web de Rsyslog:

http://www.rsyslog.com/

[7] Página web de Graylog:

https://www.graylog.org/

[8] Página web de Raspbian:

https://www.raspbian.org/

[9] Página web de Ubuntu:

https://www.ubuntu.com/

[10] Página web de Ubuntu Mate:

https://ubuntu-mate.org/raspberry-pi/

[11] Wiki de Fedora, sección sobre ARM:

https://fedoraproject.org/wiki/Architectures/ARM/Raspberry_Pi?rd=Raspberry_Pi

[12] Página web de Fedora ARM:

https://arm.fedoraproject.org/

[13] Página web de Arch Linux:

https://www.archlinux.org/

[14] Wiki de Arch Linux ARM:

https://wiki.archlinux.org/index.php/Raspberry_Pi

[15] Página de desarrolladores de Windows IOT Core:

https://developer.microsoft.com/es-es/windows/iot

[16] Manual de "Suricata Extreme Performance Tunning":

https://github.com/pevma/SEPTun

[17] Manual de instalación de Java JDK8:

http://www.webupd8.org/2012/09/install-oracle-java-8-in-ubuntu-via-ppa.html

[18] Página web de Oinkmaster:

http://oinkmaster.sourceforge.net/

[19] Página web de Emergingthreats:

http://doc.emergingthreats.net/

[20] Página web con capturas de tráfico:

http://www.netresec.com/?page=PcapFiles