Universitat Oberta de Catalunya

uoc.edu

# Análisis de Actividades Sospechosas en La Red.

**Alumno:** Jose Argenix Antigua Martinez
**Plan de Estudios:** Máster Universitario en Seguridad de Las Tecnologías de la Información y de las Comunicaciones [**MISTIC**].
Trabajo de Fin de Máster [**TFM**]. *Seguridad Empresarial MISTIC.*

**Nombre Consultor/a:** Borja Guaita Pérez.
**Nombre Profesor/a responsable de la asignatura:** Víctor García Font.

**Fecha Entrega:** 01 de junio de 2021


Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

***Agradecimientos:***

*Quiero de manera particular agradecer a* ***Mi Dios*** *por haberme permitido llegar hasta aquí y mis Pastores* ***Waldo y Zuleika*** *por haberme dado su apoyo espiritual para poder avanzar en mi carrera profesional.*

*A mi esposa* ***Cristina*** *y a mis Hijos* ***María Cristar y Cristian Argenix, a Mi Madre Norma*** *y* ***demás familiares*** *por haberme inspirado a avanzar en la vida con su apoyo.*

*A mis hermanos* ***Harom, Juan y Orlando****, quienes me animaron a dar el primer paso con este Máster y nunca me dejaron solo.*

*A* ***todos esos amigos*** *que, al tocar sus puertas, siempre estuvieron en la disponibilidad de dame una mano amiga.*

***Nunca los voy a olvidar.***

**FICHA DEL TRABAJO FINAL**

| | |
|---|---|
| **Título del trabajo:** | *Análisis de Actividades Sospechosas en La Red.* |
| **Nombre del autor:** | *Jose Argenix Antigua Martinez* |
| **Nombre del consultor/a:** | *Borja Guaita Pérez* |
| **Nombre del PRA:** | *Víctor García Font.* |
| **Fecha de entrega (mm/aaaa):** | 06/2021 |
| **Titulación:** | *Máster Universitario en Seguridad de La Tecnología de La Información y de Las Comunicaciones.* |
| **Área del Trabajo Final:** | *Análisis de actividades sospechosas en La Red. Seguridad Empresarial.* |
| **Idioma del trabajo:** | *Castellano* |
| **Palabras clave** | *NIDS, SIEM, Red* |

**Resumen del Trabajo:**


La finalidad de este trabajo de investigación e implementación consiste en identificar y desplegar aquellas tecnologías de Licenciamiento Código Libre que nos permiten analizar actividades en una red de datos y poder identificar en especial aquellas que se consideren sospechosas en base a la finalidad de su comportamiento.

En este sentido, se pretende implementar un sistema de NIDS que analice una copia de los paquetes en el tráfico de una red de datos y que este tenga la capacidad operativa de enviar esos datos recogidos a un SIEM para su posterior análisis y toma de decisiones.

Su implementación se efectuará con un NIDS que se desplegara en un entorno virtual bajo licencia de Código Libre con acceso a la red de datos que se va a analizar y un SIEM en otro entorno virtual alcanzable por red que emularía un SOC para el análisis de datos recogidos.

Dentro del contexto de su aplicación, hemos analizado que las redes de datos necesitan una constante monitorización sobre sus actividades para la toma de decisiones con respeto a la seguridad y esta implementación nos ha dado un resultado satisfactorio dado que su implementación cubre los aspectos de licenciamiento y operabilidad.

Como conclusiones, hemos determinado que es factible su implementación dado que no altera los costes por parte de licenciamiento y la carga laboral para los analistas ha sido en mejora a su rendimiento ya que les ha otorgado una herramienta que de valor agregado a sus funciones.

**Abstract:**

The purpose of this research and implementation work is to identify and deploy those Open-Source Licensing technologies that allow us to analyze activities in a data network and to be able to identify especially those that are considered suspicious based on the purpose of their behavior.

In this sense, it is intended to implement a NIDS system that analyzes a copy of the packets in the traffic of a data network and that this has the operational capacity to send this collected data to a SIEM for its subsequent analysis and decision making.

Its implementation will be carried out with a NIDS that will be deployed in a virtual environment under an Open-Source license with access to the data network to be analyzed and a SIEM in another virtual environment reachable by the network that would emulate a SOC for data analysis collected.

Within the context of its application, we have analyzed that data networks need constant monitoring of their activities to make decisions regarding security and this implementation has given us a satisfactory result since its implementation covers licensing and operability.

In our conclusions, we have determined that its implementation is feasible given that it does not alter the costs of licensing and the workload for analysts has been to improve their performance since it has given them a tool that adds value to their functions.

# Índice

# Lista de figuras

# Lista de Tablas

# 1. Introducción

## *1.1 Contexto y justificación del Trabajo*

Una red de datos [1] que se compone de una variedad de dispositivos interconectados y que de acuerdo a las características de cada equipo, van ejerciendo una función específica en la operabilidad de la red, por ejemplo un Router[2] o un Switch[3] que forman parte del funcionamiento intrínseco de esta, así como dispositivos de usuarios finales siendo por lo general los ordenadores[4] o estaciones de trabajo, los servidores [5] de la infraestructura u otros equipos como impresoras, cámaras IP, Centrales telefónicas y teléfonos IP [6], que aprovechan la disponibilidad que les otorga una red de datos para ser alcanzable por otros nodos de la misma red o más allá de esta compartir sus recursos.

En su arquitectura, podemos hoy día citar una variedad de redes de datos, definidos por el estándar [7] bajo el cual se han desarrollado como por ejemplo Token-Ring [8], IEEE 802.5, Estándar que surge en el año 1971 desarrollada por IBM [9] [10] y que se llegó a conocer por la forma en que viajaban los paquetes por el medio físico, siendo un anillo sobre hilos de cobre y que alcanzaba una velocidad de 4-16 Mbps [11]. Hoy días está en desuso.

Otra arquitectura bien conocida es la Ethernet IEEE802.3x[12] que, desde sus inicios en el año 1973 hasta la actualidad, ha evolucionado bastante y es el estándar más utilizado llegando a gestionar aproximadamente 200 Gbps 802.3cd probada en 2018, según Wikipedia[13].

Otra arquitectura que ha evolucionado bastante y se ha expandido a lo largo de todo el mundo es la Red Inalámbrica[14], específicamente el estándar 802.11x[15] (La 'x' indica que hay varios protocolos que parten del mismo estándar). La tecnología de redes inalámbricas se ha presentado como una alternativa a redes cableadas dada la facilidad de transporte utilizando como medio físico ondas radio magnéticas [16] tanto en redes empresariales, al igual que en conexiones remotas de proveedores de servicios alcanzando los más de 20 kilómetros de distancias en enlaces de radio frecuencia Punto a Punto soportados por los protocolos 802.11x.

Los despliegues de redes de datos en la actualidad se encaminan bajo un estándar que guía tanto a los fabricantes como a los integradores de redes a converger en un lenguaje común para su interoperabilidad, siendo el más conocido el modelo de Referencia OSI [17]. Este modelo, divide por capas todo el entramado de una red de datos desde que un usuario envía un dato desde una aplicación [18] en la capa 7 del modelo de referencia OSI hasta que es recibida por su el usuario receptor. En este modelo, los protocolos y estándares antes mencionados en los párrafos anteriores se corresponden a la capa 2 del modelo de referencia OSI, o capa de enlace de datos.

[1] https://es.wikipedia.org/wiki/Red_de_computadoras
[2] https://es.wikipedia.org/wiki/Router
[3] https://es.wikipedia.org/wiki/Conmutador_(dispositivo_de_red)
[4] https://es.wikipedia.org/wiki/Computadora
[5] https://es.wikipedia.org/wiki/Servidor
[6] https://es.wikipedia.org/wiki/Central_telefónica_IP
[7] https://es.wikipedia.org/wiki/Normalización
[8] https://es.wikipedia.org/wiki/Token_Ring
[9] https://es.wikipedia.org/wiki/IBM
[10] https://www.ibm.com/us-en/
[11] https://es.wikipedia.org/wiki/Megabit_por_segundo
[12] https://es.wikipedia.org/wiki/Ethernet
[13] https://es.wikipedia.org/wiki/IEEE_802.3
[14] https://es.wikipedia.org/wiki/Red_inalámbrica
[15] https://es.wikipedia.org/wiki/IEEE_802.11
[16] https://es.wikipedia.org/wiki/Medio_de_transmisión
[17] https://es.wikipedia.org/wiki/Modelo_OSI
[18] https://es.wikipedia.org/wiki/Software_de_aplicación

Esta es la primera capa de la pila de protocolos que comprende parte lógica ya que los dispositivos que en esta capa interactúan, deben tener como mínimo la capacidad de conmutar [19], es decir, de poder recibir por un canal y poder reenviar de manera inteligente la trama [20] que han recibido por otro canal relacionado a su destinatario o rechazarlo en caso de que no tenga conocimiento del destino final de esa trama.

Existen otras arquitecturas y estándares donde algunos están fuera del alcance de este TFM, por lo que su estudio quedara definido en otro trabajo que se pueda estar realizando actualmente o a futuro.

Las redes de datos, como encargadas de gestionar las interconexiones entre dispositivos de igual nomenclaturas, o aquellas que se hacen valer de dispositivos traductores intermedios para poder ser accesible a otras redes de nomenclaturas diferentes, no escapan a ataques a su seguridad y muchos de estos ataques se materializan de manera silente que solo pueden ser detectados por equipos especializados en buscar anomalías entre las diferentes PDU [21] que viajan en una Red. Es esta una de las razones que motivan a que los departamentos que gestionan la seguridad informática y seguridad de La información en las organizaciones cuenten con estas herramientas y otras herramientas y se esfuercen cada día más en su implementación para poder mitigar en el menor tiempo posible los ataques que surgen muchas veces en redes desatendidas en temas puntuales de ciberseguridad, por el simple hecho de que las redes funcionan [operan en lo más básico], obviando así que más allá de funcionar, se deben aplicar medidas y controles para salvaguardar los pilares[22] de la Integridad y la Confidencialidad dado que la disponibilidad casi siempre está cubierta por la parte operativa.

Los incidentes [23] en una red cada día indican la necesidad de poner atención a la seguridad y es por eso por lo que surge la necesidad de implementar conjunto de técnicas que, de manera inteligente y automatizada, faciliten las labores de análisis en la gestión de ciberseguridad identificando actividades sospechosas de una manera lo más transparente a la operabilidad de una red.

El tiempo de respuesta a un incidente de ciberseguridad, es una parte crítica y sobre esta puede recaer la reputación de una organización frente a los servicios que esta ofrece.

Las tecnologías con licenciamientos de uso libre, como las que son publicadas con licenciamientos de Código Libre, GPL [24] [25], por ejemplo, vienen a traer un alivio económico dado que el costo por licenciamiento para implementar estas herramientas es relativamente a cero costos, quedando un presupuesto para la adquisición de equipos físicos.

El hecho de que estas herramientas no tengan un costo económico para sus creadores o fabricantes cubierto por los usuarios no disminuye esto su calidad y su capacidad para lograr los objetivos que queremos presentar en este TFM, más bien presentamos estas herramientas con la capacidad de poder cubrir esas brechas de ciberseguridad logrando obtener resultados satisfactorios en nuestros objetivos con la facilidad que nos da el su tipo de licenciamiento y facilidad de implementación.

[19] https://es.wikipedia.org/wiki/Conmutación_(redes_de_comunicación)
[20] https://es.wikipedia.org/wiki/Capa_de_enlace_de_datos
[21] https://es.wikipedia.org/wiki/Unidad_de_datos_de_protocolo
[22] https://es.wikipedia.org/wiki/Seguridad_de_la_información
[23] https://searchdatacenter.techtarget.com/es/tutoriales/10-tipos-de-incidentes-de-seguridad-y-como-manejarlos
[24] https://es.wikipedia.org/wiki/GNU_General_Public_License
[25] https://www.gnu.org/licenses/gpl-3.0.html

## *1.2 Objetivos del Trabajo*

El objetivo de este trabajo es el de presentar el uso de herramientas de licenciamiento Código Libre para cubrir esas tareas de ciberseguridad y monitoreo que se pudiera implementar en una red de datos mostrando los mecanismos preventivos ante futuros ataques a redes en la base de monitorizar actividades sospechosas en una red.

Para lograr este objetivo, este trabajo estará centrado en subtareas tales como son:

- ✓ Un estudio de campo sobre las herramientas de las que actualmente podemos disponer y analizar sobre sacarle el mejor provecho destacando sus características.
- ✓ Mostrar las facilidades de implementación y adecuación a entornos con similitudes a ambientes lo más reales posibles y de esta manera proponer su viabilidad al ser implementado en las organizaciones.
- ✓ Identificar las ventajas que conlleva elegir este tipo de implementaciones mostrando sus resultados tanto en la detección de las actividades que se consideren sospechosas.
- ✓ Mostrar el aprovechamiento de recursos en las organizaciones para maximizar su rendimiento al integrar otras herramientas sin que estas intervengan con la infraestructura y la visión del negocio de las organizaciones.
- ✓ Mostrar los procesos de mantenimiento a los sistemas encargados de la detección de las actividades sospechosas.
- ✓ Exponer las conclusiones de este TFM y los aportes que agregamos a la ciberseguridad.

## *1.3 Enfoque y método seguido*

Para poder realizar este TFM, hemos enfocado el desarrollo de este en aquellas áreas fundamentales que nos van a permitir lograr los objetivos propuestos en los plazos de tiempos estipulados y que serán definidas en sus capítulos siguientes.

Nos hemos propuesto como objetivo implementar un sistema de detección de intrusos con reglas que puedan alertar sobre actividades identificadas como sospechosas en base a su comportamiento en la red. Para eso, debemos centrar nuestra tarea en el uso de tecnologías de licenciamiento Código Libre tanto para la parte de detección como la de visualización al igual que para los sistemas operativos y de virtualización que vamos a proponer que cuenten con estas capacidades.

Enfocaremos completa atención a la documentación que aportan los fabricantes y desarrolladores de las herramientas que vamos a proponer, dado que las estas se encontrarían accesible por cualquier organización que pretenda implementar a futuro nuestra propuesta. Es por eso, que en la fase de investigación haremos el mejor esfuerzo por hacer de estos nuestra principal guía de consulta y de materiales anexos para mantener una completa concordancia entre el objetivo del fabricante y nuestros objetivos.

Como metodología de investigación, nos acogeremos a aquellas informaciones y documentaciones oficiales y a aquellas que aporten relevancias al tema principal de este TFM, mientras que para el proceso de implementación, nos acogeremos a mostrar una manera libre donde las organizaciones que pretendan realizar esta implementaciones, pudieran incorporarlas en las infraestructuras que ya tienen funcionando pudiendo nosotros realizar recomendaciones siempre que estén detrás del objetivo de este TFM y de no aumentar costes de licenciamientos a la hora de realizar sus implementaciones.

Dado que no hemos implementado estas herramientas con los objetivos planteados en este TFM, consideramos esta es la mejor estrategia ya que nos dará la oportunidad de documentarnos lo suficiente para poder de esta manera minimizar el riesgo de que se nos llegue a escapar algún detalle, así como aumentar nuestros conocimientos al respeto.

### *1.4 Planificación del Trabajo*

Para lograr los objetivos que pretendemos con este TFM, debemos contar con recursos que nos servirán de plataforma para el despliegue en toda su entramada.

Entre los recursos que hemos destinado para este TFM, tenemos una Laptop Lenovo ® Ideapad ® Flex con sistema operativo base Fedora 32 ® [26] y la plataforma de virtualización Qemu-KVM ® [siendo este Hipervisor con características similares a un entorno de Infraestructuras como Proxmox ® [27] para virtualizar.

- Procesador Intel ® i7 7ma Generación.
- 250 GB Disco Duro SSD
- 16 GB RAM

Esta Laptop cuenta con la capacidad de virtualizar 2 servidores ejecutando GNU/Linux ® [28] [29], preferiblemente UBUNTU® [30] más no limitándonos a ninguno en particular pues deberá ser opcional a cada organización utilizar el que más se adapte a su infraestructura.

Cada estación virtualizada tendrá las siguientes características.

- 2 procesadores.
- 50 GB de Disco duro
- 2 GB RAM
- 2 tarjetas de Red. (Una operativa y otra administrativa)

El costo de esta Laptop es de aproximadamente 580,00 Euros. [31]

Una segunda Laptop marca HP ® que simularía clientes usuarios de la Red de Datos y que estaría generando los eventos tanto sospechosos como legítimos.
El costo de esta Laptop es de aproximadamente 560,00 Euros [32].

- Procesador Intel ® i5 7ma generación.
- 250 GB Disco Duro SSD.
- 16 GB RAM.

Switch CISCO ® Catalyst 1900 de 12 puertos. Este Switch presenta un costo de aproximadamente 50 Euros [33].

Los recursos humanos empelados comprenden:

[26] https://getfedora.org/es/workstation/
[27] https://www.proxmox.com/
[28] https://es.wikipedia.org/wiki/GNU/Linux
[29] https://www.gnu.org/gnu/linux-and-gnu.en.html
[30] https://ubuntu.com/
[31] https://www.ebay.com/itm/LENOVO-IDEAPAD-FLEX-5-14-FHD-Ryzen-5-4500U-16GB-256GB-SSD-81X20005US- W10/164247265907
[32] https://www.ebay.com/itm/HP-EliteBook-x360-1030-G3-13-3-i5-8350u-1-7Ghz-16gb-ram-500gb-m-2-ssd-W10-Pro/164724258259
[33] https://www.ebay.com/itm/Cisco-WS-C2950-12-Catalyst-2950-Series-24-Port-Switch/233877641320

- Alumno del Máster. Es el encargado de realizar las tareas que a continuación se describen.
- Profesor Titular. Líder del equipo del aula que estará evaluando el desarrollo del TFM.
- Profesor colaborador. Persona destinada al seguimiento al desarrollo de las tareas de acuerdo con el plan y esquema de tiempo designado por la UOC.
- Tribunal Evaluado. Es el equipo de profesionales de alto nivel encargados de la evaluación al proyecto del TFM.

Las tareas por realizar están distribuidas en un cronograma ajustado al tiempo establecido por la UOC y el plan de estudio del TFM y dividido en hitos que culminan con una entrega parcial de la Memoria final del TFM.

Como plan para elaborar este TFM, se ha definido de la siguiente manera:

**Introducción/Planificación**.

| | |
|---|---|
| 1 | **Introducción/Planificación** |
| 1.1 | Definir el Contexto y justificación del Trabajo |
| 1.1.1 | Identificar necesidades |
| 1.1.2 | Determinar relevancia y aportes a la empresa |
| 1.1.3 | Situación actual y resultado en base al aporte de esta investigación |
| 1.2 | Definir objetivos del Trabajo |
| 1.3 | Identificar el Enfoque y método seguido |
| 1.4 | Establecer una Planificación del Trabajo adecuándola al tiempo programado |
| 1.4.1 | Dividir el tiempo sobre cada tarea |
| 1.4.2 | Identificar el riesgo y posibles vías para mitigar |
| 1.5 | Describir un Breve sumario de productos obtenidos |
| 1.6 | Breve descripción de los otros capítulos de la memoria |
| 1.7 | Redacción final PEC1 |
| 1.8 | Entrega PEC1 [Primer hito con la entrega del plan de trabajo] |

Tabla 1. Tareas del tema introductorio.

Esta tarea tiene una duración de 14 días iniciando con la apertura de la docencia y culminando con la entrega de la planificación del TFM. El objetivo es el de realizar una planificación de los procesos que se van a realizar en todo el TFM. Se describen los tiempos que se le van a dedicar a las tareas y su recopilación escrita comprende el capítulo 1 de la memoria final del TFM. Se realizará en esta tarea un esquema del alcance, de los propósitos definidos en el contexto y la justificación, así como la identificación de las necesidades de la empresa con respeto a nuestra propuesta. Una breve investigación del estado actual del arte con un resumen de los recursos que se van a utilizar tanto físicos como en lógicos o aplicativos. En este conjunto de tareas también se describen los demás capítulos del resto del TFM aun pro desarrollar.

**Investigación**.

| | |
|---|---|
| 2 | **Investigación** |
| 2.1 | Investigación sobre ataques a redes y actividades sospechosas |
| 2.2 | Investigación sobre Tecnologías para Monitoreo de actividades |
| 2.2.1 | IDS/IPS bajo licenciamiento Código Libre |
| 2.2.2 | IDS/IPS bajo licenciamiento Privativos. |
| 2.2.3 | Otras alternativas a IDS/IPS |
| 2.3 | Investigación sobre Tecnologías para Visualización de actividades |
| 2.3.1 | SIEM bajo licenciamientos Código Libre |

| | |
|---|---|
| 2.3.2 | SIEM bajo licenciamientos Privativo |
| 2.3.3 | Otras alternativas para visualizar los datos recogidos |
| 2.4 | Investigación sobre la infraestructura tecnológica |
| 2.4.1 | Sistemas de Virtualización |
| 2.4.2 | Alternativas no virtualizadas |
| 2.5 | Redacción y revisión sobre la investigación y sus resultados |
| 2.6 | Entrega PEC2 [Segundo hito con la entrega de la investigación] |

Tabla 2. Tareas sobre temas de la investigación.

Esta tarea inicia justo desde que se entrega la PEC1 con una duración de 27 días. Culmina con la entrega de la PEC2.

Se pretende en este conjunto de tareas realizar una investigación sobre lo que se considera ataques a redes y lo que definiremos como actividades sospechosas.

Subsiguiente a esta, vamos a realizar una investigación sobre las tecnologías para monitoreo de redes, los llamados IDS tanto de código libre como de licencias privativas.

Luego a esto, se pretende realizar una investigación sobre herramientas para visualizar los eventos que se registren señalando la utilidad de cada herramienta que investiguemos, sus aportes al análisis de actividades sospechosas en una red y daremos una panorámica sobre los tipos de licenciamiento.

Igual se pretende realizar una investigación sobre infraestructuras tecnologías tanto donde la pretendemos instalar como en aquellas donde serviría de gran utilidad como herramienta de soporte. Trataremos en todo tiempo de mostrar las diferentes ventajas acentuando las ventajas de las infraestructuras en entornos de Código Libre. Para entrega de este apartado, vamos a redactar un recogido de todo el contenido que hemos recopilado de las diferentes fuentes de informaciones.

Una vez concluida la fase de investigación, será montada en lo que sería la segunda entrega con la investigación en sí y nuestras conclusiones acerca del producto que vamos a implementar e integrar.

**Implementación**.

| | |
|---|---|
| 3 | **Implementación** |
| 3.1 | Instalación Sistema Operativo base para IDS |
| 3.2 | Instalación sistema monitorización de Red. |
| 3.2.1 | Instalación IDS |
| 3.2.2 | Configuración IDS |
| 3.2.3 | Definiciones y pruebas de Reglas para Actividades Sospechosas. |
| 3.3 | Instalación Sistema Operativo base para SIEM |
| 3.4 | Instalación Sistema de Visualización Actividades de Red SIEM. |
| 3.4.1 | Instalación SIEM |
| 3.4.2 | Configuración SIEM |
| 3.4.3 | Integración final con IDS |
| 3.5 | Resumen y revisión de la implementación |
| 3.6 | Redacción del resultado de la implementación |
| 3.7 | Entrega PEC3 [Tercer hito con la entrega de la evidencia de implementación] |

Tabla 3. Tareas sobre el tema de la implementación.

Esta tarea tiene una duración de 28 días iniciando justo luego de la entrega de la PEC2 y culminando con la entrega de la PEC3.

Esta tarea está destinada junto a sus subtareas a implementar lo que en la fase de investigación hemos seleccionado como la solución ideal a nuestro propósito. Primero vamos a implementar una instalación fresca de los Sistemas operativos, preparando un entorno adecuado para implementar las soluciones siguientes.

Luego a esto, pretendemos realizar la implementación de la parte de monitoreo con el IDS o sistema de monitoreo de actividades sospechosas en una red para su posterior análisis. Esta fase incluirá una instalación en base a lo recomendado por el fabricante y una configuración adecuada a los propósitos del TFM.

Luego a esto, instalaremos el SIEM que hemos seleccionado previamente y lo vamos a integrar para que procese los registros que le vamos a pasar desde la parte de monitoreo para que el analista tenga una visibilidad del entorno lo más adecuado a tiempo al tiempo real.

Una vez instalado todo, integrado las diferentes plataformas y puesta en marcha, haremos un resumen de la instalación con revisiones en busca de algún punto que quede por cubrir y redactaremos un resultado que incluiremos en lo que sería la tercera entrega de este TFM.

**Presentación Memoria Final TFM.**

| | |
|---|---|
| 4 | **Presentación Memoria Final TFM** |
| 4.1 | Conclusiones del TFM sobre la propuesta inicial |
| 4.2 | Redacción memoria final. |
| 4.3 | Entrega Memoria Final [Cuarto hito con entrega del fichero de la memoria] |

Tabla 4. Tarea sobre la presentación de La Memoria.

Esta tarea cuenta con unos 35 días para ser desarrollada culminando con la entrega de La Memoria Final del TFM.

Esta tarea está dedicada a redactar las conclusiones sobre el proyecto mostrando a las organizaciones la rentabilidad de implementar este tipo de soluciones. Se pretende mostrar los resultados de nuestra propuesta para el análisis de actividades sospechosas en una red.

**Presentación del Video.**

| | |
|---|---|
| 5 | **Presentación en Video** |
| 5.3 | Elaboración video sobre la memoria del TFM |
| 5.4 | Presentación y entrega de Video sobre La Memoria del TFM |

Tabla 5. Tarea sobre la presentación del Video.

Esta tarea cuenta con 7 días para elaborar y presentar el video. Culmina cuando el video es presentado.

Se pretende en este tiempo la elaboración de un gráfico audiovisual mostrando el desarrollo de la investigación e implementación de este TFM. Se mostrará el funcionamiento a plena de la implementación mostrando los resultados en el entorno del SIEM.

**Defensa del TFM.**

| | |
|---|---|
| 6 | **Defensa del TFM** |
| | Espera resultados y calificación finales del TFM |

Tabla 6. Tarea sobre la defensa del TFM.

La defensa del TFM tiene una duración de 5. Culmina cuando con las respuestas en defensa del TFM.

En esta tarea se pretende responder a las preguntas del jurado y a la espera posterior a la calificación final.

Se ha determinado que, para la realización de este TFM, se pretende dedicar un estimado de 2 a 4 horas diarias de lunes a viernes. Se tomarán los sábados y Domingos para completar algunos retrasos que pudieran surgir durante la semana considerados en el análisis de riesgo del TFM que se va a presentar en esta obra.

## Cronograma de tareas.

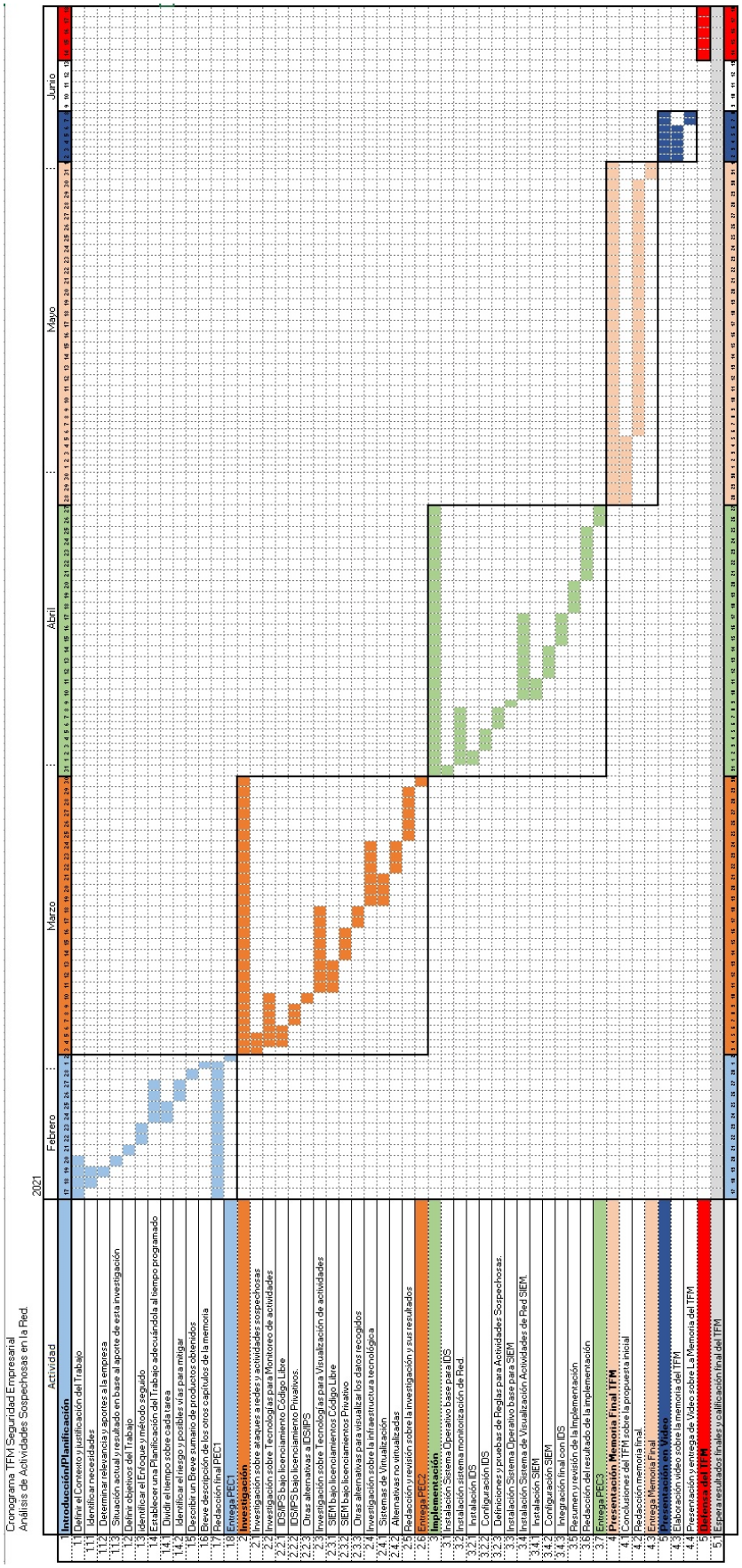



Imagen 1. Diagrama de Gantt

*1.5 Análisis de riesgo*

Durante todo el proceso en que estemos desarrollando este TFM, debemos estar claros y documentar que pueden surgir algunos imprevistos que de manera directa pusieran afectar el desarrollo de este proyecto. Luego de evaluar nuestro entorno con una mirada optimista, nos planteamos algunos puntos que ponen en riesgo el avance tal cual se ha planteado en el cronograma de actividades.

**Enfermedades hospitalarias.** Dado que estamos actualmente en un estado de alerta en el país que resido [34], La República Dominicana, dada la actual pandemia de Covid-19, nuestras autoridades sanitarias han determinado que la probabilidad de contagio aun es alta. Esto nos obliga a tomar medidas que del todo no es 100% efectiva, pero de carácter obligatoria a nivel nacional. Ante una posibilidad de contagio y pasar a severidad, supone este riesgo un punto muy desfavorable pues se tendría que dedicar un tiempo extra luego de haber recuperado nuestro estado normal de salud. Como plan de mitigación nos acogemos a los planes preventivos dispuestos por nuestras autoridades sanitarias.

**Accidentes ambulatorios**. Dado que actualmente nos encontramos laborando fuera de la ciudad que residimos, aumenta el riesgo de accidentes por desplazamientos. Esto supone un plan de contingencia preventivo en el que, como medidas mitigantes, nos comprometemos a adelantar una hora a nuestra salida desde nuestros hogares con fines de disminuir las velocidades de desplazamientos y evitar percances.

**Enfermedades de familiares cercanos**. Supone que este punto esta intrínsecamente relacionado al estado de ánimo que pudiéramos adoptar ante tales situaciones. Supondría esto una pausa de algunos 3 días antes de retomar la marcha normal del proyecto. Como medida para mitigar esta falta hemos coordinado con los integrantes del núcleo familiar continuar las mismas medidas preventivas.

**Fallas de equipos en la implementación**. Hemos definido en los recursos equipos que son los destinados para los fines del despliegue de las herramientas de este proyecto. Para los ordenadores personales, disponemos de algunos reemplazos de uso en el hogar y para fines ajenos a las actividades planteadas en la universidad. Como plan de contingencia hemos acordado que mientras no presentemos la defensa a nuestro TFM, estas estarán disponibles para fines de emergencia.

**Ajustes de tiempo de entregas**. Supone el hecho que no toda la información que necesitamos la tenemos a mano y las herramientas que vamos a desplegar no están actualmente instaladas en nuestras facilidades, esto abre la posibilidad de que los plazos planteados no se pudieran cumplir porque el alcance del proyecto no se llegue a cubrir en el tiempo planificado. Como medida de contingencia, proponemos la posibilidad de reajustar los tiempos ya determinados acortando en aquellas tareas que suponen de menor impacto para dar prioridad a aquellas que son fundamentales para la entrega.

**Fallas en las instalaciones**. Supone que algunas de estas herramientas no son de uso cotidiano por parte del estudiante, por lo que existe la posibilidad que algunas de estas herramientas resulten un poco más complejas de lo esperado. Como medida de contingencia proponemos poner más interés en las investigaciones sobre las implementaciones que afectan directamente el éxito del proyecto.

**Perdida de las informaciones**. Podría ser una posibilidad que uno de los equipos falle y las informaciones que este contiene se llegaren a perder. Para mitigar tales

[34] https://www.presidencia.gob.do/decretos/95-21

fines, hemos adoptado la disciplina de mantener sincronizado en todos los ordenadores que laboramos junto a un drive en la nube con control de versiones.

**Estado del hardware insuficiente**. Podría surgir la posibilidad de que, al momento de realizar el despliegue, el hardware seleccionado no sea el recomendado. Para tales fines, como plan de mitigación proponemos la adquisición de otros recursos al alcance financiero del proyecto. Hasta ahora hemos contemplado una distribución adecuada de los recursos por lo que la posibilidad de que se materialice esta amenaza es muy baja.

**Herramientas no adecuadas para el objetivo**. Para fines de la elaboración de este proyecto, aún no hemos probado las herramientas del mercado por lo que existe la posibilidad de que las que encontraremos no sean las adecuadas. Para mitigar este riesgo, que podría retrasar la entrega final, proponemos el uso adecuado del tiempo de investigación en contemplar por igual casos de éxitos y adquirir las fuentes de informaciones relacionadas para completar el objetivo. En caso de que estas herramientas dejaren de pertenecer a las de licenciamiento Código Libre, nos veríamos en la necesidad de optar por una variante o bifurcación de la herramienta, una versión antigua u otra herramienta con iguales características con miras a lograr los objetivos.

Hemos tratado de ser lo más meticulosos ante los riesgos inherentes al desarrollo de este TFM, por lo que de materializarse una amenaza que ponga en peligro el cumplimiento de estos objetivos, y que no le hemos ponderado, nos veremos en la necesidad de informarlo al consejo de profesores para que nos indiquen sobre alguna posibilidad de continuar y que no estemos ponderando.

### *1.6 Breve sumario de productos obtenidos*

Finalizado este TFM, el producto obtenido es una herramienta de monitoreo de actividades sospechosas en la red bajo licenciamiento de Código Libre con la capacidad de ser instalado en cualquier organización que cuente o no con un Centro de Operaciones de Ciberseguridad y Monitoreo.

Para la detección de las actividades sospechosas, hemos desplegado un NIDS, Suricata, con la capacidad de visualizar todas las tramas que se ejecuten por los puertos que estaría monitorizando o las VLANs que estén enviando una copia de las tramas que gestionan y habrá de determinar una alerta basado en las reglas descargadas o reglas que el analista habría de desarrollar en base al comportamiento que quiera monitorear.

Para poder visualizar los eventos que registre el NIDS, hemos desplegado ELK que, junto al módulo de Suricata para la ingesta de la data, podemos leer el fichero eve.json que estaría recibiendo de forma remota las actualizaciones de las actividades que han sido capturadas. Queda a opción libre del analista el construir sus Dashboards o continuar gestionando vía los que trae por defecto cuando instalamos el módulo Suricata de Filebeat.

### *1.7 Breve descripción del estado del arte.*

Aunque en la fase de investigación continuaremos con este apartado, adelantemos en esta primera entrega algunas consideraciones sobre el estado actual de estas tecnologías.

Como hemos detallado en la introducción, la intención de este trabajo es la de integrar herramientas ya elaboradas mostrando las ventajas de estas para el análisis de actividades sospechosas en una red.

Actualmente existen varias herramientas para análisis de actividades sospechosas que se encargan de la captura y análisis automatizados de datos que viajan en una red de datos [35]. Estos son ya conocidos como IDS/IPS.

**IDS/IPS/NIDS**

Estas herramientas tienen las características necesarias para determinar en base a reglas previamente definidas, el comportamiento del tráfico en una red y tomar una decisión en base las acciones configuradas en estas reglas o en los ficheros de configuración. En términos generales, podemos definir un IDS/IPS [36] como un aplicativo utilizado para detectar tráfico sospechoso y prevenir accesos no autorizados en una red o a un ordenador.

Snort ®. Es de uno de los IDS/IPS más conocido desde que iniciaron el proyecto en 1998 [37] [38]. Actualmente se sigue desarrollando por CISCO ® aunque se sigue distribuyendo ajo licencia GPLv2. De igual manera existen otras versiones bajo licenciamiento privativos y otras que se distribuyen en UTMs con el costo de licenciamiento incluido en el equipo. pfSense ® hoy NetGate ® y otras variantes, es un UTMs, que utiliza a Snort ® como IDS/IPS y de manera paralela se mantenía una versión libre que se descargaba y se instalaba con mantenimiento de las comunidades. Hoy esa herramienta ha sido bifurcada, pero continúa utilizando Snort ® como IDS/IPS.


Imagen 2. Logo de Snort ®.

Suricata ®. Ha sido la alternativa a Snort ® que, de hecho, presenta algunas características que lo hacen tener un mejor rendimiento que Snort ®. [39] [40] Este cuenta con procesamiento multi hilos procesando más que paquetes con y capacidad de analizar capas superiores a la de red, análisis de ficheros, entre otras. Aunque su arquitectura es diferente a Snort ®, comparten similitudes en cuanto al uso de las mismas firmas para la detección de intrusos y lenguaje de reglas.


Imagen 3. Logo de Suricata ®

Zeek ®. Anteriormente llamado Bro ®, es otro IDS de Código Libre basado en anomalías y en firmas [41]. Al tráfico que esta captura genera un evento o una serie de eventos. Este IDS tiene una característica que le distingue de los anteriores y es que posee un lenguaje de scripting que le facilita a quien le administre tareas de automatizaciones correlacionado con otras herramientas como antivirus para análisis de ficheros detectados como infectados.

[35] https://protegermipc.net/2018/02/22/mejores-ids-opensource-deteccion-de-intrusiones/
[36] https://es.wikipedia.org/wiki/Sistema_de_detección_de_intrusos
[37] https://en.wikipedia.org/wiki/Snort_(software)
[38] https://www.snort.org/
[39] https://en.wikipedia.org/wiki/Suricata_(software)
[40] https://suricata-ids.org/
[41] https://zeek.org/

Imagen 4. Zeek ®

Existen otros tipos de IDS/IPS que actúan en otro entorno y que están fueras del alcance de este TFM. Son los Basados en hosts [42]. Este tipo de IDS/IPS se instala en los hosts y su alcance está en las actividades propias que impactan directamente en el host que está instalado. Como nuestro caso el alcance es a la red, este tipo de herramientas de momento no nos serviría de mucha utilidad.

De igual manera, hemos mencionado que existen una variedad de IDS/IPS con licenciamientos privativos y que su uso este derivado del costo de esta licencia. El enfoque de este TFM es el uso de tecnologías Código Libre eliminando el costo de licenciamiento para las organizaciones con herramientas capaces de realizar análisis de actividades sospechosas en la red.

**Los SIEM.**

Para el objetivo de este TFM, hemos indicado que, para el analista, se le deberá proveer de una herramienta que pueda visualizar las actividades que de manera automatizada están siendo colectadas por los IDS/IPS.

Los SIEM son el acrónimo del idioma ingles [Security Information Events Management]. Son herramientas que permiten una visualización global de la seguridad de la tecnología de la información. Este se encarga de gestionar los eventos de manera centralizada y en tiempo real relativo al tiempo en que los eventos indican que un incidente se ha perpetuado [43].

Entre los principales bajo licenciamiento Código Libre, podemos encontrar a **OSSIM** ®. Esta es la versión Código Libre de AlienVault ® llegando a ser uno de los más populares [44]. Se puede integrar a otros proyectos tanto para gestión de eventos locales de servidores como para gestión de eventos de otras tecnologías como los IDS/IPS. Actualmente se está distribuyendo para las capacidades de poder gestionar descubrimiento de activos, análisis de vulnerabilidades junto con AlienVault, detección de intrusos (IDS) monitoreo de comportamiento y correlación de eventos de seguridad. Este es un SIEM con bastantes características.


Imagen 5. OSSIM ®

**ELK**. Las iniciales de (Elastic, Logstash y Kibana). Este SIEM es una combinación de varias herramientas que juntas pretenden brindar soporte a los analistas de datos relacionados a eventos de la seguridad y de amplio uso en muchas comunidades [45] [46] [47] [48]. Existen versiones de pago como es Pignus ® de la empresa Tic Defense ™

[42] https://www.clavei.es/blog/que-es-un-ids-o-intrusion-detection-system/
[43] https://sofecom.com/que-es-un-siem/
[44] https://cybersecurity.att.com/products/ossim
[45] https://es.wikipedia.org/wiki/Elasticsearch
[46] https://es.wikipedia.org/wiki/Logstash
[47] https://en.wikipedia.org/wiki/Kibana
[48] https://www.elastic.co/es/

[49] que ofrece soporte de pago a esta herramienta y algunos plugins que trae esta versión. La misma herramienta en si tiene versiones de pago con otras funcionalidades extras.

Es una herramienta de amplia aceptación y que se ha podido instalar en varias plataformas, GNU/Linux y Microsoft ® Windows.


Imagen 6. ELK. Elastic ®, Logstach ® y Kibana ®.

**GNU/Linux** ®. Es el conjunto de aplicaciones ejecutándose sobre un núcleo [Linux] con características similares a Unix® que forman un sistema operativo de uso bajo licencias GNU de la Free Software Foundation [50] y que, en la actualidad, existen varias distribuciones de características casi similares. También son conocidos como los Unix-Like. Este sistema operativo actualmente es el más usado en servidores de alto rendimiento por su amplia característica de gestión de los recursos de hardware. El kernel fue desarrollado en 1991 por Linus Torval, programador finlandés que quiso crear algo parecido a Minix®, una versión de Unix® para uso estudiantil en ese entonces en la universidad de Helsinki.

Hoy día, tiene una gran aceptación en los ordenadores personales dado la no limitante de uso de licencias para casi todas sus aplicaciones, que se pueden utilizar bajo licenciamientos de uso libre.

Para el caso en que nos compete en este TFM, este será nuestro sistema operativo base por cumplir con los requisitos que vamos a detallar en el capítulo de investigación.


Imagen 7. GNU/Linux.

**Proxmox** ®. Es un sistema de virtualización para infraestructuras empresariales lanzado en 2008 que viene a ser una alternativa Código Libre a otros sistemas para virtualizar como lo es VMware ®. Está basado en Debian ® gestionado por una consola web, y tiene a KVM [Kernel Virtual Machine] como hipervisor y Open VZ ® para gestión de contenedores GNU/Linux [51] [52].

Soporta máquinas virtuales Microsoft ®, GNU/Linux ®, Solaris ® en arquitecturas i386 y x86_64. Es ideal para entornos de pruebas, entidades con presupuestos limitados para licenciamientos, organizaciones de crecimiento constante soportando almacenamientos en red, en arreglos de discos y en sistemas de gestión de volúmenes lógicos LVM [53].

[49] https://www.ticdefense.com/
[50] https://www.gnu.org/gnu/linux-and-gnu.en.html
[51] https://www.proxmox.com/
[52] https://es.wikipedia.org/wiki/Proxmox_Virtual_Environment
[53] https://es.wikipedia.org/wiki/Logical_Volume_Manager_(Linux)

Imagen 8. Proxmox ®.

### *1.8 Breve descripción de los otros capítulos de la memoria*

**Capítulo 2. La investigación de campo.**

Es un capítulo destinado a la investigación de las tecnologías que vamos a implementar. Esta investigación estará centrada en elegir investigar y elegir los sistemas de detección, el SIEM y los Sistemas Operativos que vamos a tomar para este TFM, tomando en cuenta sus características, requisitos, recomendaciones, adaptación y completar aquellos conocimientos que al momento de iniciar este TFM, pudiera ser desconocido para el Estudiante al igual que los aplicativos o servicios que vamos a necesitar para cumplir con nuestra propuesta manteniendo el enfoque en las documentaciones de los fabricantes, recomendaciones de profesionales expertos cuando así se considere y mantener una higiene en las citas que vamos a incluir como referencias a la hora de entregar la memoria final y sus hitos previos.

**Capítulo 3. La implementación técnica.**

Es un capítulo completamente técnico, donde pondremos énfasis en la implementación de las tecnologías que hemos designados para cubrir este TFM. Pretendemos en este capítulo no solo demostrar la viabilidad técnica del proyecto, también a eso presentar una guía documentada de cómo se implementaría este proyecto en una organización tomando en cuenta los factores vinculantes al logro de nuestros objetivos.

Estaremos mostrando de manera gráfica como se realizarían las implementaciones acentuando los pasos a seguir y de manera más directa, para un caso más amplio al alcance de este TFM, citando las fuentes originales de los fabricantes y las diferentes documentaciones que nos van a proveer.

**Capítulo 4. La conclusión.**

Se pretende en este capítulo sacar las conclusiones sobre la investigación e implementación de esta propuesta de TFM, mostrando los alcances obtenidos y las conclusiones técnicas que permitirán recomendar esta implementación en una organización viendo las ventajas en el aspecto económico, técnico y los resultados que se esperarán al implementar una solución de igual característica.

**Capítulo 5.**

Este capítulo mostraremos un glosario detallado de los términos y acrónimos más relevantes.

**Capítulo 6.**

Una bibliografía con las referencias que hemos tomado para este TFM.

**Capítulo 7.**

Documentos anexos que no se pudieran incluir en los demás capítulos, pero se consideran necesarios ser presentados.

# 2. La investigación de campo

*2.1 Ataques a redes de datos.*

Como indicamos en el capítulo 1, las redes de datos no están exentas de ataques a su seguridad poniendo en riesgo la Confidencialidad, Integridad y Disponibilidad, o como se le conoce, *los 3 pilares de la seguridad*.

Muchos de esos ataques son materializados en diferentes vectores de ataques sin dejar a un lado cuando estos son provocados intencionalmente con fines de violar la seguridad de las redes y materializado por uno que otro individuo con malas intenciones.

Los integradores de redes, muchas veces carentes de unas experticias en ciberseguridad, configuran equipos que componen la red de una manera que permite que los atacantes tengan un mejor escenario para materializar sus ataques a las redes.

De cara a la internet, los equipos de perímetro dedicados a proteger el tráfico entre la internet y las redes internas se ven afectados por vulnerabilidades que permiten que estos ataques se materialicen, siendo una de la más común el dejar configuraciones que vienen por defecto de fábrica, en el peor de los casos, no ponen ninguna seguridad.

En nuestra experiencia como consultor de ciberseguridad y analista de respuesta a incidentes, hemos podido evidenciar que muchos de estos ataques se llegan a materializar por estos mismos casos antes mencionados. Técnicos que han dejado contraseñas débiles en servicios expuestos permitiendo que herramientas automatizadas puedan romper con esa débil seguridad, entre otros casos.

Otro de los indicadores que hemos podido identificar son los sistemas desatendidos y por ende, muchas veces con versiones obsoletas y vulnerables en sus sistemas principales, lo que permite un camino expuesto a vectores de ataques con poca limitantes dado que cuando estas vulnerabilidades son hechas públicas, muchas veces hacen publico algunas herramientas para poder evidenciar la posibilidad que se materialice un ataque no siendo estos hechos con intenciones malintencionadas más bien, con la finalidad de que se realicen las pruebas de concepto de lugar para poder evidenciar la presencia de una vulnerabilidad. Estas herramientas, al estar disponibles para su descarga libre, los ciberatacante se hacen de las mismas y materializan ataques a redes impactando a organizaciones que aún no han solucionado sus debilidades.

A todo esto, encontramos algo muy típico ya y es que todo equipo conectado a una red es de una manera u otra es vulnerable a algún tipo de ataque. Estos ataques, bien pueden variar de acuerdo con su comportamiento, su objetivo, a la capa del modelo OSI donde se ejecuta, a la capacidad técnica de quien materializa un ataque, así como a comportamiento de herramientas que suelen automatizar los atacantes, y es esto lo que a partir de este punto podemos indicar entre lo que son actividades sospechosas en una red.

Definiremos una actividad sospechosa en una red como un comportamiento no legitimo o no usual que sucede en una red y que difiere en para el cual ha sido desplegada en una red de datos a lo que, como consecuencia, se puede o se podría estar materializando un ataque o un incidente a la seguridad.

### *2.1.1 Actividades sospechosas.*

Las actividades sospechosas en una red van de la mano con los ataques ya que, aunque bien puede estar materializándose por vía de un programa maligno instalado de manera intencional por un atacante o un ataque dirigido. Un ejemplo podría ser un intento de forzar el acceso a un nodo de la red al cual no estaría autorizado o algún que un equipo de la red ha sido comprometido por una infección de BotNet [54].

En una red de datos, los paquetes se mueven de un nodo a otro y este tráfico se puede monitorizar de manera automática, que es el objetivo de este TFM, pudiendo dar con la posibilidad de aislar aquellas actividades que consideremos sospechosas de aquellas identificadas como legitimas.

Entre los diferentes ataques que podemos identificar en una red y que el portal techtarget.com lo agrupa entre los 10 principales, están los siguientes:

- **Intentos no autorizados a sistemas**. Son actividades donde un atacante intenta dar con el acceso a un sistema donde no tiene las credenciales y posiblemente tampoco la autorización. Estos ataques bien pueden materializarse con métodos manuales ya sea adivinando o introduciendo cadenas de caracteres hasta ver si logran dar con el acceso o, con herramientas automatizadas que intentan forzar la entrada al sistema de una manera más eficiente. Este tipo de ataque como comportamiento sospechoso, se podría evidenciar identificando esos patrones que poseen esos paquetes de datos cuando viajan por una red de datos. Los ataques a contraseñas están ligados a este tipo de intentos ya que estas forman parte de las credenciales de acceso a la mayoría de los sistemas.

- **Movimientos laterales y escalada de privilegios.** Cuando un atacante ha logrado acceder a un sistema con unas credenciales que no necesariamente poseen privilegios administrativos, un atacante querría tener más control sobre ese sistema, pero por igual, obtener control sobre otros nodos de la red a la que pertenece el sistema ya comprometido, es otra metodología que utilizan los atacantes. Este tipo de ataque lo que busca es poseer el mayor control en una red y en sus nodos.

- **Amenazas internas.** Se constituyen en un peligro casi desapercibido ya que este tipo de ataque viene de algún individuo que en un momento tuvo o actualmente tiene acceso a recursos de la red. Por lo general, este tipo de amenaza no es muy fácil de detectar ya que no son predecibles del todo y suelen ser explotadas mediante actividades legitimas.

- **Ataques de phishing.** Creciente hoy día y cada vez mucho mejor elaborado, consiste en obtener las credenciales de usuarios legítimos mediante el uso de sistemas fraudulentos. Estos ataques tienen dos factores de detección. La concienciación donde se le dan entrenamientos a los usuarios para identificar estos ataques y así prevenirlos, y el otro es dando seguimiento a portales de dudosa reputación. Por lo general este vector de ataque se materializa vía correos electrónicos que aparentan ser legítimos, incitando a que los usuarios accedan a enlaces que tienen como objetivo capturar las credenciales de sistemas reales con portales de características visuales idénticas a los legítimos. Otra manera de perpetuar este tipo de ataque es cuando el atacante empela otras vías para dar con informaciones de los usuarios y aportar credibilidad al cebo y de esta manera tener mejor éxito. Llamadas telefónicas

[54] https://es.wikipedia.org/wiki/Botnet

haciéndose pasar por individuos que solicitan informaciones puede ser otro ejemplo.

- **Los ataques de malwares**. Este tipo de ataque es muy variante y depende del tipo de programa maligno que se ha alojado en los nodos de la red, por lo general en estaciones de trabajo y en servidores, aunque podría ejecutarse en otro dispositivo de la red. Por sus características, se podrían identificar estas actividades sospechosas y que, en su variedad, su comportamiento es distinto. Un gusano, por ejemplo, que se propaga en una red de un nodo a otro, podría este generar un patrón de conducta al momento de auto propagarse. Un backdoor, que podría estar generando tráfico ilegitimo mientras las estaciones no deberían estar trabajando, es otra manera de poder identificar estos. Son solo ejemplos que nos servirían más adelante con la identificación de actividades sospechosas en una red.

- **Ataque de denegación de servicios**. Suelen ser muy frecuentes a servicios expuestos de cara a la internet como servicios web, o a equipos periféricos, y que tampoco deja a un lado que dentro de una red interna se pudiera materializar un ataque de este tipo. Estos ataques consisten en dejar deshabilitado estos servicios al que se tiene alcance. Suelen ser identificados por la gran cantidad de tráficos de paquetes que cargan los sistemas y los hacen colapsar. Estas actividades sospechosas suelen ser mitigadas con equipos intermedios que detectan y bloquean el tráfico entrante, por lo que, en una red interna, el proceso estaría centrado en la identificación de aquellos nodos que aun siendo de confianza, actuarían de manera sospechosa evidenciando mediante su comportamiento que no son legítimos.

- **Man in the middle, MitM [Hombre en el medio].** Suele ser uno de los ataques más difíciles de detectar ya que todo el tráfico se origina de manera legitima y aun cuando pasa por el dispositivo intermedio, es un tráfico de comportamiento aparentemente legítimo. Consiste en un sistema que intercepta el tráfico luego que sale de su origen, tiene la posibilidad de leer el mensaje inclusive alterándolo antes de llegar a su destinatario. Atenta directamente a la confidencialidad y a la integridad dado que interceptaría un paquete, lo leería violando la confidencialidad y lo podrías alterar violando la integridad. Este tipo de ataques algunas veces suele identificarse bajando a capas inferiores y verificando que los equipos intermedios son legítimos verificando la legitimidad de sus direcciones de interfaces físicas. Como método preventivo se recomiendan otras capas de seguridad de niveles superiores que impidan que estos paquetes aun sean interceptados, no puedan ser violentados y al llegar a su destinatario, no sufran alteraciones. La suplantación de equipos es parte de su patrón de acción a este tipo de ataques.

- **Snifing.** Es una técnica que suele ser de mucha utilidad cuando es empelada con fines de monitoreo, pero cuando es un atacante que intenta ver que sucede en la red, entonces suele ser un incidente que afecta la confidencialidad de la red. Son actividades sospechosas que se suelen materializar en entornos donde no se tiene control de los equipos que forman parte de una red de datos. En este tipo de redes, el acceso al medio no tiene seguridad por lo que identificar estas anomalías suele hacerse verificando si los nodos que se conectan están debidamente autorizados. Consiste en capturar paquetes en la red e intentar ver su contenido.

Estos ataques junto a otros que más adelante se pudieran definir, vienen a ser parte de los más puntuales a la hora de identificar actividades sospechosas en una red de datos. Un analista de monitoreo de ciberseguridad deberá contar con herramientas que automaticen la búsqueda constante de estas actividades en las redes y brinden una manera eficiente de poder actuar con suficiente anticipación antes de que estos ataques se llegaren a materializar o evitar que el impacto sea mucho mayor para las organizaciones si se ha llegado a materializar un ataque.

Este tipo de herramientas son por lo general sistemas dedicados a la búsqueda constante de patrones que difieran de lo que se podría considerar como tráfico normal. Muchos de este tráfico se disfrazan como legítimo, por lo que se requiere una capacidad de inspección de esos paquetes más a profundidad lo que suma una carga de procesamiento en algunos casos y, por lo tanto, se le deben dedicar recursos suficientes para que sus labores sean ejecutadas de manera transparente y eficiente en las redes de datos.

## *2.2 Tecnologías para monitoreo de actividades sospechosas.*

En el entorno de los sistemas que detectan actividades sospechosas, se han desarrollado con el paso del tiempo algunas herramientas que han resultado eficientes en muchos aspectos. El mantenimiento de sus configuraciones, los soportes tanto de organizaciones privadas como de comunidades de uso libre, así como también en su implementación en entornos dado que estas herramientas no se limitan a las configuraciones propias de las organizaciones, sino a ser completamente elásticos y adaptables a diferentes estructuras organizacionales han permitido que estas herramientas hoy puedan ser una realidad. Para nuestro caso, vamos a identificar y definir los que son de licencia de Código Libre y los de licencias privativas.

Para el caso preciso de este TFM, nos centraremos en la utilización de los **IDS/IPS** como herramientas para la identificación de anomalías que vamos a considerar como actividades sospechosas en una red de datos, en especial las de código Libre y lo justificaremos en su apartado correspondiente.

**IDS/IPS**, son las iniciales del inglés *Intrusion Detection System / Intrusion Prevention System.* Su traducción seria: *Sistemas de Detección de Intrusos / Sistema de prevención de Intrusos.* Estas son herramientas que tienen la capacidad de analizar el tráfico de la red con sensores que pueden identificar basado en reglas predefinidas ataques conocidos, comportamientos sospechosos inclusive, puede analizar el contenido de los paquetes. [55]

Estas herramientas se han desarrollado en diferentes arquitecturas. Están las basadas en Host. Estas herramientas analizan los tráficos desde y hacia los hosts o estaciones donde están instalados. Tienen la particularidad que su instalación y su alcance se limita al sistema donde se ha instalado. Para nuestro alcance, estas herramientas no nos serian de mucha utilidad porque solo centrarían su rango de acción al host donde han sido instalados. Su utilidad es más dedicada a servicios críticos que requieren de una atención dedicada.

### *2.2.1 IDS/IPS de licenciamiento Código Libre.*

Entre los IDS/IPS con licenciamiento de Código Libre, encontramos varios que ya hemos mencionado en el capítulo 1. **Snort** ®, **Suricata** ®, entre otros. Estos los podemos descargar de sus portales destinados para su descarga e instalarlos y situarlos en las organizaciones sin costos de licenciamientos. Su uso es muy potente y

[55] https://es.wikipedia.org/wiki/Sistema_de_detección_de_intrusos

efectivo cuando se trata de detección de intrusos y un monitoreo de redes. Traen la posibilidad de poder detener un intruso o de alertar sobre alguna anomalía.

Los IDS/IPS que pretendemos implementar no solo son de licenciamiento de Código Libre, también son catalogados como NIDS [56]. Estos tienen la particularidad que se encuentran ubicado en un punto de la red desde donde pueden realizar capturas del tráfico de manera transparente y poder analizar los paquetes capturados para luego tomar una decisión de acuerdo con la regla o la configuración que se le ha implementado.

Entre los IDS/IPS de código abierto con estas características, encontramos a**:**

- **Snort** ® [57]. Es un sistema de detección de intrusos con licenciamiento Código libre, que podemos instalar en sistemas operativos tanto GNU/Linux como Microsoft ®. Es multi plataforma, creado inicialmente como APE, un aplicativo para GNU/Linux para el año 1998. Fue escrito en Lenguaje C [58] por Marty Roesch. Actualmente está publicado bajo licencia GPLv2 [59], aunque CISCO ® tiene parte en el desarrollo actual de este sistema.

  Snort ® tiene la capacidad de almacenar los eventos de su labor de detección en diferentes formatos, ya sea en ficheros de texto plano, o integrado ya que Igual se puede realizar una implementación en una base de datos. Es muy potente como detector de eventos tales como ataque y escaneo de puertos.

  Snort ® tiene algunas limitantes ante otros NIDS, por ejemplo, es que su procesamiento es diferente. No es multihilo [60] y es una desventaja notable ante otros NIDS.

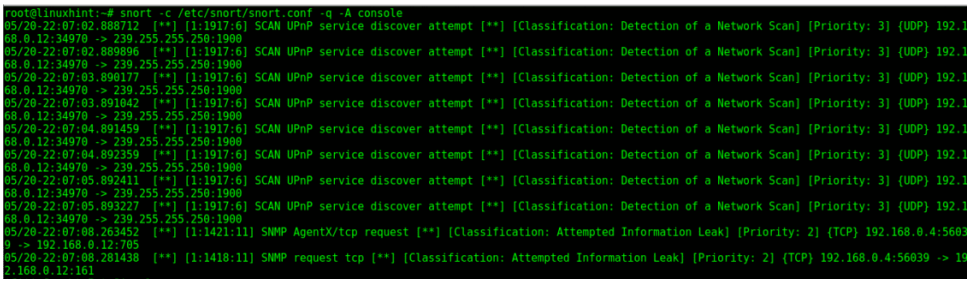


Imagen 9. Consola Snort de alertas.

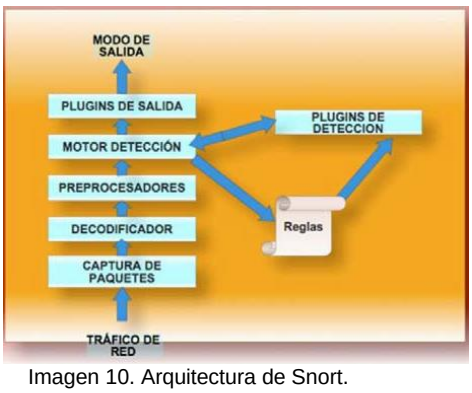


Imagen 10. Arquitectura de Snort.

[56] https://es.wikipedia.org/wiki/NIDS
[57] https://es.wikipedia.org/wiki/Snort
[58] https://es.wikipedia.org/wiki/ C_(lenguaje_de_programacion)
[59] https://es.wikipedia.org/wiki/ GNU_General_Public_License#Versión_2
[60] https://es.wikipedia.org/wiki/Multihilo

Snort ® su parte más crítica es el saber configurar reglas que puedan detectar actividades sospechosas en una red. Para eso, veamos cómo se compone una regla de Snort.


Imagen 11. Reglas Snort.

Las acciones, indican que hará Snort ® cuando se dispare la regla.

Origen y Destino indican las direcciones de capa 3 que influyen en la regla.
Puertos indican los puertos lógicos de capa 4 al que se está estableciendo la conexión o a que servicio se está solicitando u originando la conexión.
dirección indican el flujo de la comunicación.

```
alert tcp $HOME_NET 21 -> any any (msg: “Error autenticación FTP”; content: “Login or password incorrect”; sid:1000001; rev:1;)
```

Esta es una regla de ejemplo que indica que se emitirá una alerta cuando un usuario intente desde la red local, realizar una conexión FTP y las credenciales no están correctas.

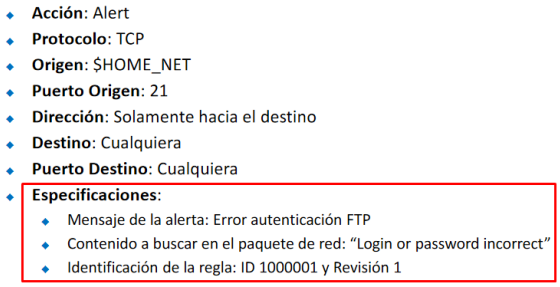



Imagen 12. Reglas Snort (ejemplo).

Se deberá prestar atención a las especificaciones ya que de esto depende que logremos filtrar la regla cuando esta abra un paquete pues deberá buscar una cadena de caracteres muy específica ubica en el campo “content:”. Antes del filtro está el mensaje de alerta, que será una cadena de caracteres que, al leerse, el analista podrá identificar el evento y para el motor de Snort, siguen el ID de la regla y su revisión.

Ya en el proceso de implementación vamos a ver con mejor detalle esta parte de la configuración de las reglas.

- **Suricata** ® [61]. Desarrollado por la *Open Information Security Fundación (OISF)* [62] en 2009 cuando sale si primera versión beta, es un potente NIDS multiplataforma que se distribuye bajo licencia GNU GPL [63]. Tiene una similitud con Snort ® en las reglas lo que le permite realizar una inspección a paquetes en una red actuando tanto de IPS o IDS, pero Suricata aprovecha más el hardware que, de hecho, si es multihilo lo que le permite realizar tareas

[61] https://suricata-ids.org/
[62] https://en.wikipedia.org/wiki/Open_Security_Foundation
[63] https://en.wikipedia.org/wiki/GNU_General_Public_License

de forma paralelas y no secuenciales únicamente. Igual junto a Snort ® pueden trabajar de manera fuera de línea [ofline] analizando ficheros “pcaps” [64].

Este NIDS utiliza el estándar de entrada y salida ya conocido YAML [65] y JSON [66] y su integración con SIEMs es muy común por lo que su escalabilidad es una gran ventaja para las organizaciones que deciden utilizarlo.

Otra ventaja que podemos acentuar en Suricata ® es su poderoso sistema de scripting [67] lo que le diferencia en la automatización de las detecciones con otros NIDSs de su clase. También podemos ver que es posible la extracción de ficheros mapeados a ciertas reglas lo que le facilita el proceso de análisis.

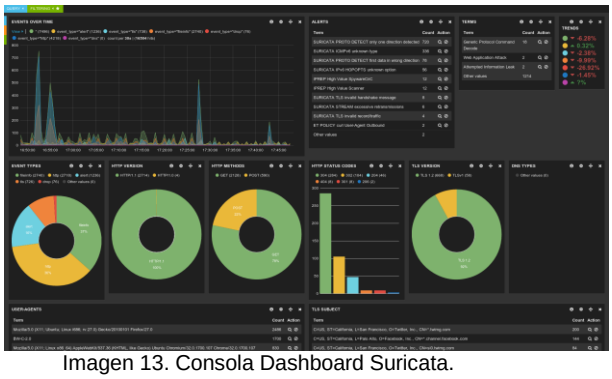


Imagen 13. Consola Dashboard Suricata.

Para el caso de Suricata ®, como hemos dicho este comparte cierta similitud con Snort ® en la estructura de sus reglas. Siguen un patrón idéntico por lo que se podría hasta cierto punto reutilizar las reglas de Snort ® en Suricata ®.

```
alert tcp 192.168.65.133 any -> any !80 (app-layer-protocol: http; msg: "HTTP but not port 80"; sid:3; rev:1;)
```

Imagen 14. Regla Suricata.

Como se puede observar, esta regla es casi similar en su estructura a la de Snort ®.

Esta regla lo que hace es que desde la ip origen 192.168.65.133 identifica cuando la conexión es diferente al puerto 80 a cualquier destino utilizando el protocolo de aplicación HTTP generando un mensaje que la conexión es HTTP pero que no es al puerto 80.

Ya en el proceso de implementación del siguiente capítulo, vamos a ver con mejor detalle esta parte de la configuración de las reglas.

- **Zeek** ® [68] Desarrollado por Vem Pazson [69] en 1994 originalmente como **Bro**, es un IDS/IPS bajo licencia BSD ® [70] y es bien utilizado como monitor de redes de datos. Tiene una amplia disponibilidad de analizar extensivas actividades de redes, aunque su alcance se puede apreciar desde las conexiones cableadas hasta las capas superiores como la de aplicación.

  Por defecto, Zeek envía la información que colecta a ficheros JSON, pero se puede integrar a bases de datos y a sistemas SIEM. Este NIDS tiene

[64] https://es.wikipedia.org/wiki/Pcap_(interfaz)
[65] https://es.wikipedia.org/wiki/YAML
[66] https://es.wikipedia.org/wiki/JSON
[67] https://es.wikipedia.org/wiki/Script
[68] https://docs.zeek.org/
[69] https://en.wikipedia.org/wiki/Vern_Paxson
[70] https://en.wikipedia.org/wiki/BSD_licenses

capacidad de análisis en capas de aplicaciones como la detección de malwares, al igual que detección de intentos de fuerza bruta en servicios de conexiones remotas como SSH o la verificación de cadenas de conexiones SSL.

Al igual que otros IDS/IPS, cuenta con módulos y funcionabilidad de scripting para automatización de tareas de análisis de los eventos capturados y posteriores notificaciones.

Zeek ® es bastante completo y no es de dudar que muchas grandes organizaciones lo estén utilizando y sacando el mejor de los provechos.

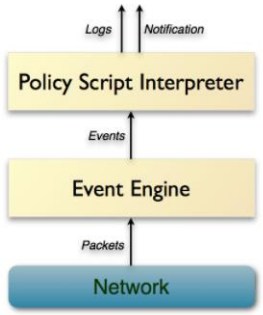



Imagen 15. Arquitectura de Zeek ®.

- **OpenWIPS-ng** ® [71] es otro IDS/IPS que pudimos identificar en nuestra tarea de investigación. Es un NISD enfocado a redes inalámbricas. Cuenta con un sensor para colectar la data y ser enviadas, un servidor para análisis de datos y una interfaz para visualizar los eventos. Cada instalación comprende un sensor que se ejecuta como pequeños CPU lo que limita bastante.

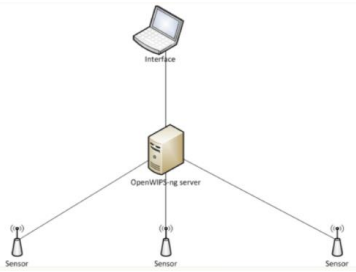


Imagen 16. Arquitectura de OpenWISP-ng.

- **Sguil** ® [72] [73]. Es más bien un monitor de redes con capacidad de IDS. Funciona mejor integrado con otros IDS como Snort ® y es preferible instalarlo como clientes en los nodos de la red para que genere las alertas de los incidentes. Se puede implementar bajo licencia GPLv3 [74].

No pretendemos con este TFM abordar todos los IDS/IPS de licenciamiento Código Libre que actualmente están disponibles, pero si aquellos que podrían formar parte de la mejor opción para presentar este proyecto.

[71] https://openwips-ng.org/index.html
[72] https://en.wikipedia.org/wiki/Sguil
[73] http://bammv.github.io/sguil/index.html
[74] https://en.wikipedia.org/wiki/GNU_General_Public_License#Version_3

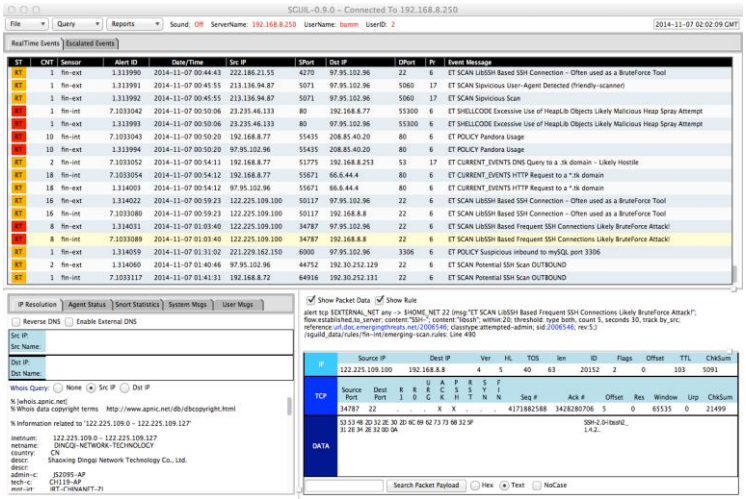


Imagen 17. Consola de eventos de Sguil.

### *2.2.2 IDS/IPS de licenciamiento Privativo.*

De la misma manera que existen IDS/IPS con licenciamiento de Código Libre, igual podemos encontrar otros con otro tipo de licenciamiento que podría tener algún costo para su uso e implementación o, que pudiera tener un soporte de pago para su uso. Estas herramientas son las que llamamos bajo licenciamientos privativos, ya que su licencia trae limitaciones de uso y como ya mencionamos, los usuarios deben incurrir en pagos para poder hacer uso de estas herramientas.

Entre las más comunes bajo licenciamiento privativo están:

- **Secure IPS (NGIPS) ®** [75]. Es una solución del gigante CISCO ® que ofrece las funcionalidades necesarias para poder monitorear una red de manera automática y a la vez, poder tener una acción proactiva ante amenazas detectadas. Trae integrado una interfaz para visualizar los eventos y para la gestión de la herramienta bastante madura. Sobre las configuraciones de las reglas, al ser un soporte de pago recibe cada cierto tiempo actualizaciones que se implementan de inmediato. Este IDS/IPS viene en dispositivos ya preinstalados que se sitúan en un ligar estratégico de la red y casi de inmediato la organización puede visualizar el tráfico y realizar análisis o tomar decisiones en base al tráfico que captura. Los appliances o dispositivos son la versión Firepower ™ Series anterior ASA ™ de CISCO ®. También se puede instalar una máquina virtual lista para VMWare ® con suscripciones de pago para recibir las actualizaciones.

  Por los costes de licenciamientos y el tipo de dispositivo, entendemos que este equipo no está a nuestro alcance ya que la visión y el objetivo que llevamos es la del uso de herramientas de código libre aprovechando la ventaja del no costo y libertad de implementación para las organizaciones. Como sistema bajo licenciamiento privativo, este no nos da una opción que podamos implementar sin cubrir el costo de la licencia de su uso e implementación.

[75] https://www.cisco.com/c/en/us/products/security/ngips/index.html


Imagen 18. Catalogo de productos CISCO Firepower.

- **Fidelis Network** ® [76]. Fidelis ® es una empresa dedicada a la producción de equipos para la seguridad de las empresas. Entre su catálogo de productos, tiene a Fidelis Network ™. Es un dispositivo o appliance con capacidad para detectar amenazas en la red y con la capacidad de responder a amenazas. Mediante el uso de Machine-learning, estos dispositivos escanean los puertos de la red en busca de amenazas para acumulando metadatos para su posterior análisis. Su escalabilidad es amplia y se adapta a entornos empresariales ofreciendo un rendimiento muy aceptable.

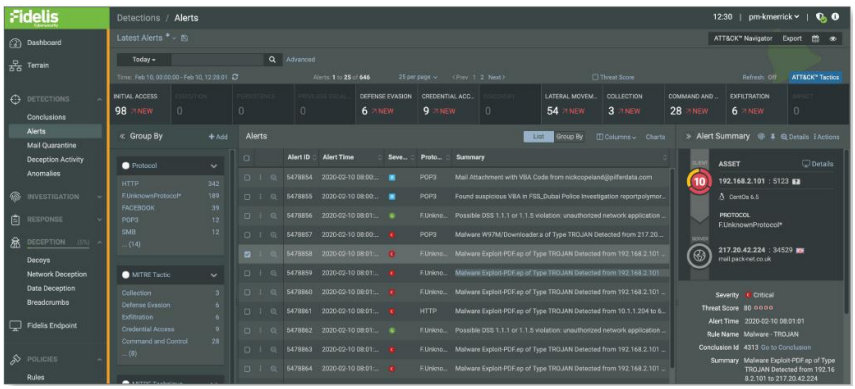



Imagen 19. Consola administrativa de Fidelis ®

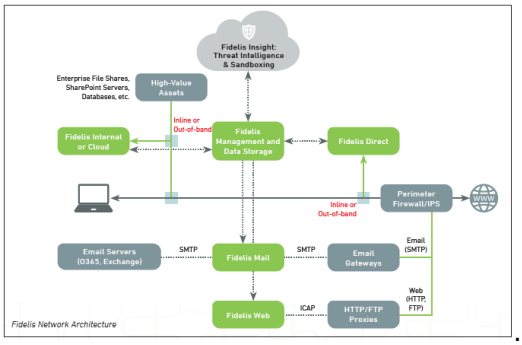



Imagen 20. Arquitectura de Fidelis Network

Su soporte es por suscripción aparte al costo de adquisición de los equipos.

Como se ha podido observar, existen muchas soluciones en el mercado privativo y que ofrecen soluciones muy robustas a la hora de realizar una implementación de este tipo. Se podría considerar una de estas si la organización dispone de suficientes

[76] https://fidelissecurity.com/products/network/

recursos, aunque estas tecnologías, igual se pueden implementar con otras de licenciamientos Código Libre.

Entre aquellas que hemos indicado que son de licenciamiento de Código Libre, algunas organizaciones han confiado en su poder y la han adoptado en sus catálogos de productos pudiendo comercializar el soporte. Casos como el de pfSense ® [77], hoy bifurcado con NetGate ® [78]. Estos son Firewall perimetrales que traen integrado Snort ®. Aunque soportan una comunidad de usuarios, sus servicios se venden tanto en dispositivos o appliances, así como servicios en la nube. Se distribuye bajo licencia Apache 2.0 [79] y su sistema base es BSD ®.

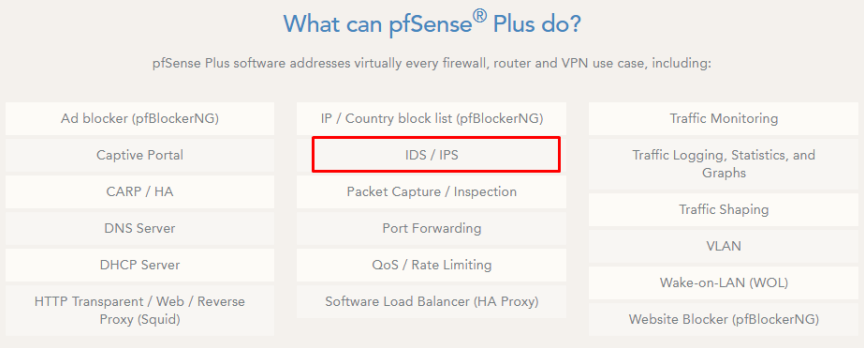


Imagen 21. Caracteristicas de pfSense ® y NetGate ®

Estos dispositivos tienen objetivos diferentes a los nuestros que, aunque podríamos instalar pfSense ® bajo licenciamiento BSD ® [80] y Apache 2.0 [81], sus objetivos son muy diferentes a los nuestros ya que no son ampliamente dedicados el tema IDS/IPS. Estas son características agregadas como equipos perimetrales que estos ofrecen.

### *2.2.3 Otras alternativas a IDS/IPS.*

Existen otras herramientas [82] como los analizadores de paquetes [83] que nos podrían dar uno que otro resultado en otro modo operacional, estas herramientas se ejecutan en un modo un poco más abstracto y por lo general capturan tramas, PDUs de la capa 2 del modelo OSI. En capas superiores igual se pueden encontrar herramientas que hacen el trabajo de analizador de protocolos como paquetes de Red, Cifrados, cabeceras HTTP, entre otros. Los más comunes son los siguientes:

- **Wireshark ®**. [84] Es una herramienta de código libre que captura tramas, paquetes y datos en una red. Es muy útil porque su interfaz es muy amigable y ofrece muchas posibilidades de ver el contenido de las PDUs, aunque si el usuario no conoce bien como se componen los PDUs de cada capa del modelo OSI, se le puede complicar ver el contenido y saber interpretarlo.

77 https://www.pfsense.org/
78 https://www.netgate.com/solutions/pfsense-plus/
79 https://en.wikipedia.org/wiki/Apache_License
80 https://es.wikipedia.org/wiki/Licencia_BSD
81 https://www.apache.org/licenses/
82 http://www.mundocisco.com/2009/08/que-es-un-sniffer.html
83 https://es.wikipedia.org/wiki/Analizador_de_paquetes
84 https://www.wireshark.org/

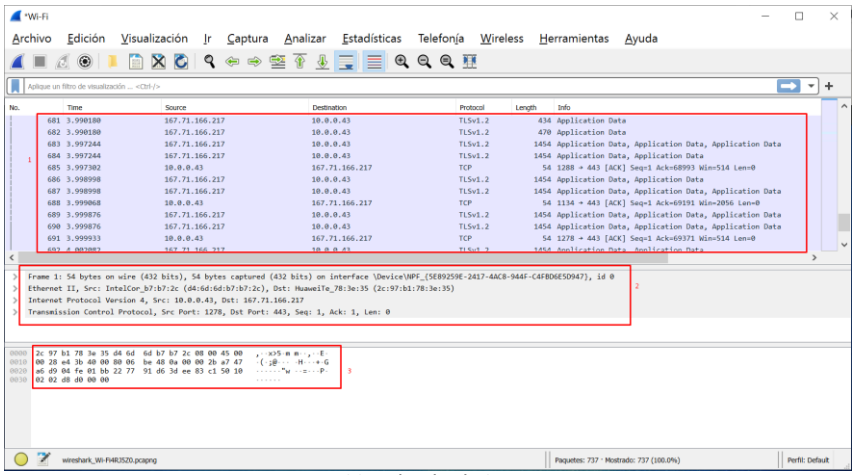



Imagen 22. Wireshark ®

En el recuadro 1, Se puede visualizar cada captura que se va realizando en tiempo real. Cuando elegimos una de esa captura, en el recuadro 2 podemos ver su contenido separado por cada PDU y en el recuadro 3, podemos ver el valor hexadecimal de la captura que queremos leer.

- Ettercap. [85] Otras herramientas para análisis de trafico de red. Es más bien un Interceptor de paquetes. Su configuración por lo general es para situarla como hombre en el medio e interceptar y redirigir los paquetes a otro destino con fines de ser suplantados. Para los que gustan de la línea de comando, Bettercap [86] es la solución. Es más utilizada en ataques de hombre en el medio.

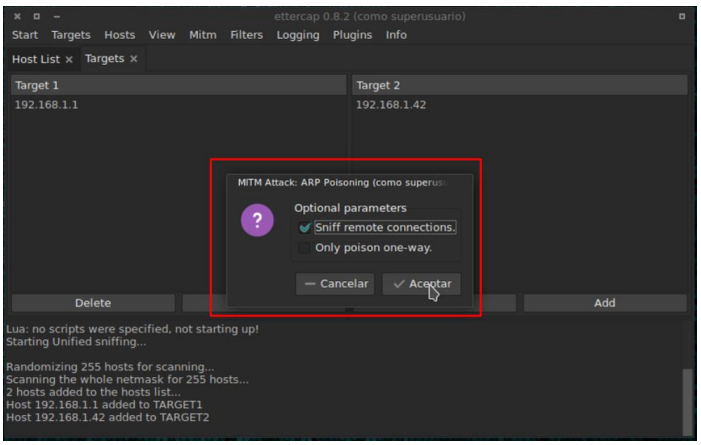



Imagen 23. Ettercap

Estas y otras herramientas, aunque son de licenciamiento de código abierto, solo las vamos a ver por encima ya que su uso es muy específico y un poco fuera del alcance de este TFM. Con los IDS/IPS entendemos conseguimos un mejor resultado.

## *2.3 Tecnologías para La Visualización de actividades sospechosas en la red.*

Hasta ahora, hemos estudiado el tema de captura del tráfico que se genera cuando existen actividades en una red, en especial aquellas donde se ha generado una actividad sospechosa. El proceso de análisis va un poco más allá de solamente capturar los datos. El proceso de análisis necesita de herramientas que permitan a los

[85] https://www.ettercap-project.org/
[86] https://www.bettercap.org/

analistas visualizar esa data que se ha extraído y presentarla de una manera amigable y entendible para la toma de decisiones ante un posible incidente.

Estas herramientas deben tener la capacidad de recibir estos datos, procesarlos, enriquecerlos de una manera donde se pudiera mapear con otras fuentes de informaciones y que se pueda inclusive poder utilizar como soporte para permitir evidenciar un incidente.

Las herramientas que hemos presentado para la recogida de informaciones traen casi todas sus consolas que permiten visualizar las capturas hasta en tiempo real, pero nos damos con la disyuntiva que cada una está desarrollada posiblemente en diferentes tecnologías y posiblemente, en todas no podríamos visualizar los datos de una manera estandarizada puesto que cada una provee informaciones en base al proceso y motor de recogida que ejecuta y con la visión que se ha desarrollado. Por este punto, entre otros, creemos que se necesita de una herramienta que visualice esos datos y que los pueda inclusive cruzar con otras informaciones permitiendo enriquecer esa data cruda, por una más acabada.

Actualmente nos podemos encontrar con Centros de Operaciones de Seguridad o **SOCs** al igual que en los Centros de Respuestas a Incidentes de Ciberseguridad o CSIRT, que ya contienen herramientas que les permite procesar datos desde diferentes fuentes y enriquecerlos para facilitar las tareas de los analistas.

Estas herramientas, las podemos encontrar con licenciamiento de Código Libre, lo que nos seria de mucha ayuda ya que se comprende a los objetivos de este TFM.

### *2.3.1 SIEM de licenciamiento Código Libre.*

Los **SIEMs**, como lo definimos en el capítulo 1, son el acrónimo del idioma ingles [Security Information Events Management]. Son herramientas que permiten una visualización global de la seguridad de la tecnología de la información. Este se encarga de gestionar los eventos de manera centralizada y en tiempo real relativo al sello de tiempo en que los eventos indican que un incidente se ha ejecutado [87] y de diferentes fuentes.

Los objetivos del SIEM serían:

- Recolectar una variedad de eventos de distintas fuentes
- Establecer un contexto
- Implementar las capacidades un análisis en tiempo real
- Implementar las capacidades para un análisis histórico.

La versatilidad de un SIEM implica la capacidad de poder recibir datos de diferentes fuentes de informaciones y poder correlacionarlas para brindar al analista un panorama más certero de la realidad de un ataque al momento de que este se esté materializando. En consecuencia, pasa a ser una de las herramientas de mayor utilidad en sus labores por lo que debe esta ser de suma confianza a la hora de ser puesta en producción.

[87] https://sofecom.com/que-es-un-siem/

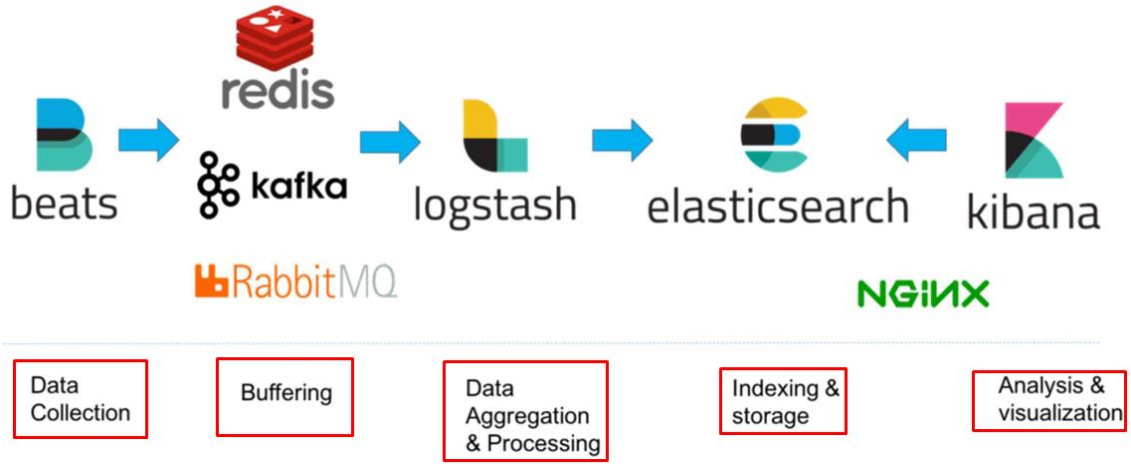



Imagen 24. Componentes basicos de un SIEM.

Como se puede ver en la imagen anterior, una estructura básica de un SIEM se compone de un data *collector*, que es el que recibe la data de diferentes fuentes, un *buffering* para la redistribución oportuna de los datos en crudo, luego el data *aggregation & processing* quien procesa los datos y luego se lo pasa al *indexing & storage* como motor de indexado y almacenamiento y ya, por último, el *Analysis & Visualization* que es la parte donde interactúa el analista para visualizar los eventos.

Bajo licenciamiento Código Libre, podemos encontrar a **OSSIM** ™. Esta es la versión Código Libre de AlienVault ® llegando a ser uno de los más populares [88]. Esta herramienta es más bien una bifurcación o una versión libre y que trae algunas limitantes porque las funciones extras son de pago. Son muchas las herramientas que presentan este tipo de situaciones, donde una versión para la comunidad es permitida ser desplegada mientras que la versión de pago va ofreciendo las mejores funcionalidades.

Si realizamos una comparativa entre las dos versiones, vemos que sus ventajas como Código abierto no es tan rentable.

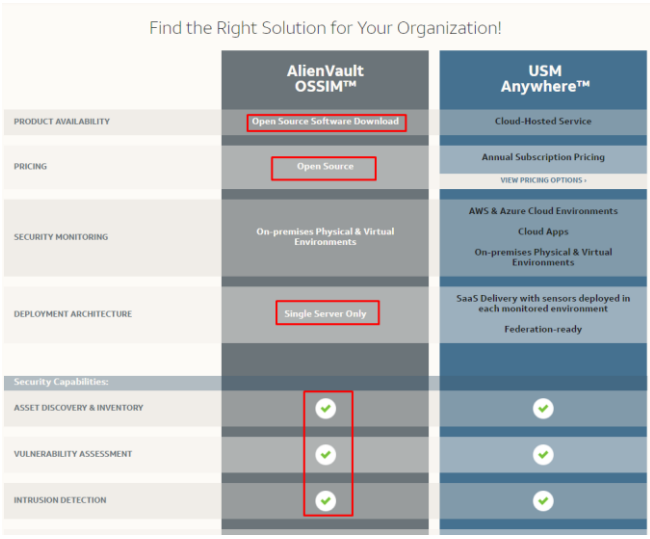



Imagen 25. Comparativa de licencias.

Esto supone ser un problema ya que la versión soporte más adelante en el mismo portal nos identifica algunas limitantes a la hora de integrarla con otras herramientas. Esta trae su propio IDS OSSEC, identificado como un basado en host IDS. y su

[88] https://cybersecurity.att.com/products/ossim

instalación es algo limitada. Una organización que se embarca en este tipo de herramientas con esas limitantes podría a futuro tener problemas al querer crecer o subsistir en el tiempo dado el surgimiento de nuevas demandas. La escalabilidad debe ser algo que lo garantice la tecnología, no que la limite.

**Apache Metron** ™ [89]. Este proyecto nace en OpenSOC ® [90] de CISCO ® en 2016 por lo que se ve que es relativamente nuevo. Esta herramienta centra sus funciones en recopilar, transmitir y procesar datos de seguridad. Su sonda de recopilación de datos, Apache NIFI recopila datos desde las diferentes fuentes para luego ser analizados y finalmente procesados y normalizados en formato .json. Este proyecto se vale de otras herramientas ya en el mercado y mucho más maduras como es Elasticsearch ® para la indexación y enriquecimiento al igual que otras herramientas y para visualizar, utiliza Kibana ®.

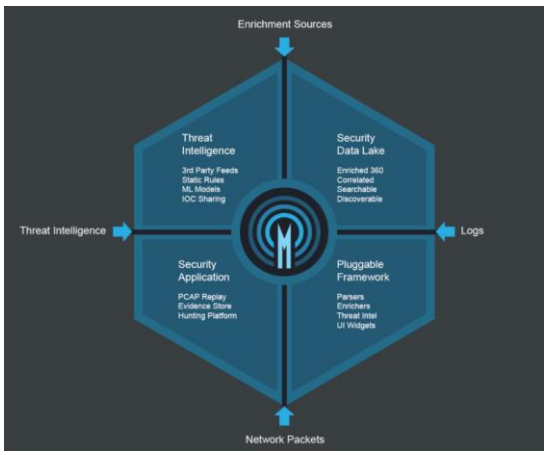


Imagen 26. Arquitectura simple de Apache Metron.

Su distribución está bajo licencia Apache [91] y se puede instalar con asistencias de aprovisionamientos o directamente en máquinas virtuales tanto para Sistemas operativos de la familia RedHat ® o Ubuntu corriendo contenedores Docker ®.

Su documentación [92] está muy bien estructurada por lo que una instalación no sería nada complicado si se siguen las indicaciones. Su documentación oficial tiene buen material y ara una persona con experiencias en Sistemas Operativos antes mencionados, no le resultaría muy difícil realizar esta implementación.

**Security Onion ® [93].** Es una de las suites más completas puesto que trae ya en una distro preinstalada IDS Network y Host based, gestión de inventarios, Alertas, SIEM entre otras utilidades. Es una imagen que se descarga de su portal y se instala como un servidor en la red. Como trae ya consigo varias herramientas, es super escalable ya que se expande por toda la red su alcance ofreciendo una gama de servicios integrados ya con una sola instalación.

[89] https://metron.apache.org/about/
[90] https://opensoc.github.io/
[91] http://www.apache.org/licenses/LICENSE-2.0
[92] https://cwiki.apache.org/confluence/display/METRON/Documentation
[93] https://securityonionsolutions.com/software

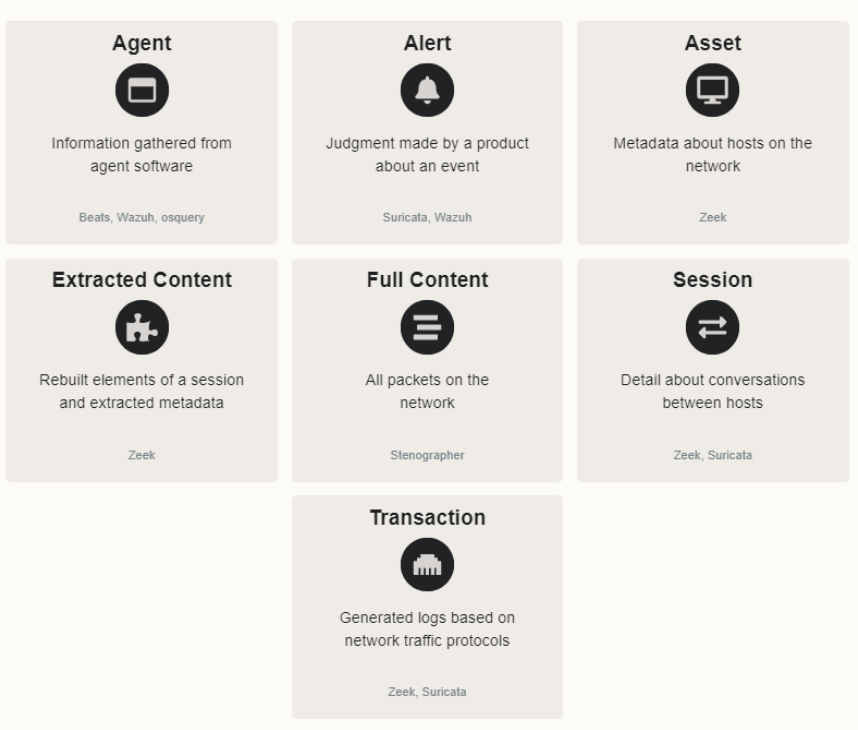



Imagen 27. Estructura y componentes de Security Onion.

Caso aparte de que es multiherramientas, ofrece la oportunidad de elegir entre varias herramientas para objetivos específicos. Esto brinda una oportunidad de poder adaptarse tanto a la organización como a la base de conocimientos de sus integradores.

**ELK**. Las iniciales de (Elasticsearch ®, Logstash ® y Kibana ®) [94] [95] [96] [97]. Este SIEM es una combinación de varias herramientas que juntas pretenden brindar soporte a los analistas de datos relacionados a eventos de la seguridad y de amplio uso en muchas comunidades. Hoy por hoy, casi en todos los SOCs o en los CERTs o CSIRTs, es una de sus herramientas principales. Multiplataforma, ya que se puede instalar en GNU/Linux y Microsoft Windows ® y su reputación le precede. Es bien potente y de mucha utilidad en el análisis de grandes volúmenes de datos de diferentes fuentes.

Se puede integrar a otras herramientas para la colección de datos donde ELK se encarga de renderizarlos brindando al analista una herramienta muy madura.

Logstash ®, tiene la capacidad de recopilar datos desde casi cualquier fuente, al mismo tiempo que puede filtrar y procesarlos correlacionándolos debidamente. Para almacenarlos, se hace valer de Elasticsearch ® como gestor de base de datos. Este motor de base de datos NoSQL [98] indexa los datos para poder realizar las búsquedas mientras que Kibana ®, se encarga de mostrarlo. Otro componente del stack es Beats que se encarga de cargar una serie de registros relativamente pequeños y enviarlos a la pila vía Logstash ®.

Algunas limitantes que aún están en desarrollo para ser solventadas es la capacidad de poder emitir alertas o informes, por lo que es necesario utilizar otras herramientas para cubrir esta parte. Hasta hace poco, el tema de la seguridad era algo distante puesto que las reglas de seguridad no eran permitidas. En versiones más recientes, se

[94] https://es.wikipedia.org/wiki/Elasticsearch
[95] https://es.wikipedia.org/wiki/Logstash
[96] https://en.wikipedia.org/wiki/Kibana
[97] https://www.elastic.co/es/
[98] https://es.wikipedia.org/wiki/NoSQL

ha podido ver que es posible ya integrar módulos de autenticación para poder acceder al sistema. Hoy día ya se puede integrar con sistemas de autenticación como LDAP o con el módulo nativo de Elasticsearch ®.

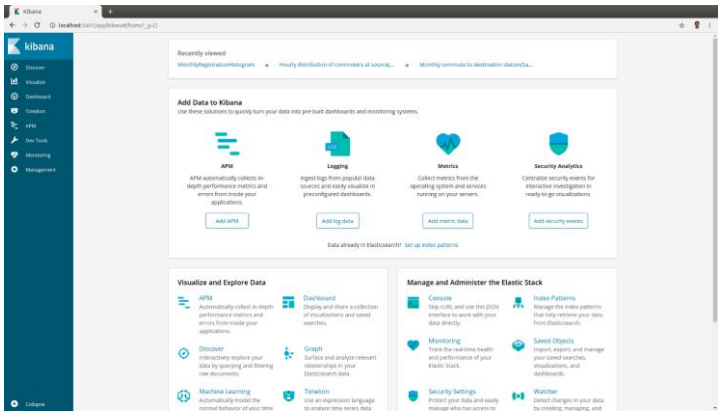


Imagen 28. Pagian de inicio de Kibana.

Existen de seguro otros que se pueden obtener e implementar bajo el mismo licenciamiento, y que podrían ser de mayor utilidad. Hasta el momento, estos son los que hemos identificado como los más utilizado.

### *2.3.2 SIEM de licenciamiento Privativo.*

Aunque no es el objetivo de este TFM el implementar soluciones bajo licenciamiento distinto a Código Libre, entendemos que en el mercado existen algunas soluciones que pueden ser de utilidad y darían un resultado muy satisfactorio siempre y la organización cuente con un presupuesto que cubra estas necesidades.

De las mismas soluciones que hemos presentado anteriormente como soluciones de código Libre, podemos identificar que algunas de ellas tienen soportes de pago y que, de esta manera, se puede acceder a funcionalidades que una versión comunitaria no las trae, o que no se tiene una línea de soporte que garantice un tiempo de respuesta por parte del fabricante ante un eventual hecho que requiera de un especialista.

**ELK**. Que ya hemos detallado parte de sus funcionalidades, tiene unas versiones de paga que podría ser costeada por una organización que tenga el presupuesto disponible, aunque con la ventaja de que pudiera realizar la implementación de manera libre y luego emplear soporte de pago.

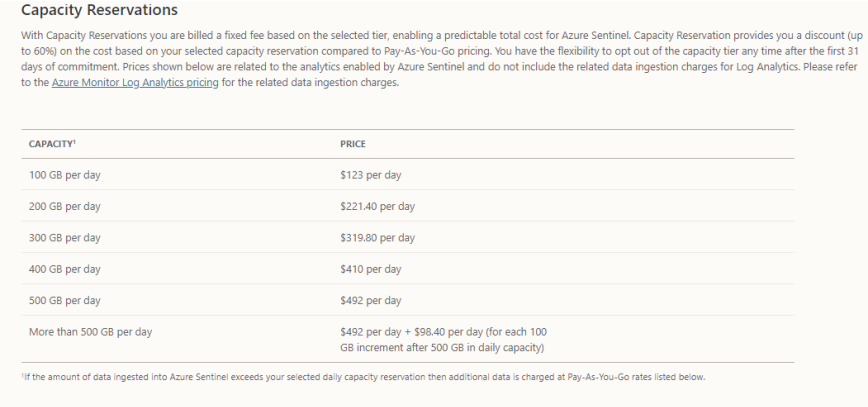
Imagen 29. Precios de Elastic.

Como se puede observar, Elasticsearch ® se puede adquirir por vía de pago y emplear soporte del fabricante más otros beneficios que posiblemente no tenga en la versión comunitaria. Aun así, Elasticsearch ® se destaca por ser de Código Libre. Casos como estos, muchas veces se especifican que lo que se paga es el soporte.

Una de las ventajas más acentuadas es la versión Cloud, que permite aprovechar la nube y sus beneficios.

**Azure Sentinel** ® [99]. Es la versión de Microsoft ® que ofrece para este tipo de tecnologías. Tiene capacidad de colectar datos, hacer inteligencia detectando amenazas e investigación de amenazas y se puede integrar con otras herramientas para poder dar respuestas.

Su precio es por capacidad de almacenamiento procesado por día y que puede variar dependiendo de la cuenta de abonados que se tiene con el fabricante.



Capacity Reservations

With Capacity Reservations you are billed a fixed fee based on the selected tier, enabling a predictable total cost for Azure Sentinel. Capacity Reservation provides you a discount (up to 60%) on the cost based on your selected capacity reservation compared to Pay-As-You-Go pricing. You have the flexibility to opt out of the capacity tier any time after the first 31 days of commitment. Prices shown below are related to the analytics enabled by Azure Sentinel and do not include the related data ingestion charges for Log Analytics. Please refer to the Azure Monitor Log Analytics pricing for the related data ingestion charges.

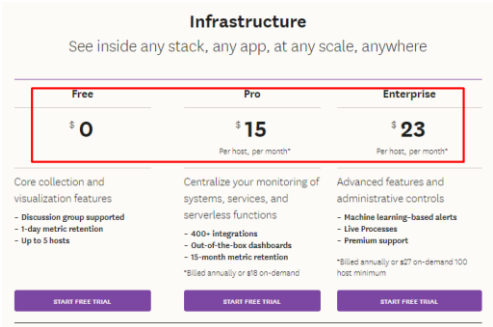
| CAPACITY[1] | PRICE |
|---|---|
| 100 GB per day | $123 per day |
| 200 GB per day | $221.40 per day |
| 300 GB per day | $319.80 per day |
| 400 GB per day | $410 per day |
| 500 GB per day | $492 per day |
| More than 500 GB per day | $492 per day + $98.40 per day (for each 100 GB increment after 500 GB in daily capacity) |

[1]If the amount of data ingested into Azure Sentinel exceeds your selected daily capacity reservation then additional data is charged at Pay-As-You-Go rates listed below.

Imagen 30. Precios de Azure Sentinel.

Actualmente es bien utilizado en gobiernos para SIEM de SOCs y de Centros de Respuestas a incidentes.

**Datadog ® [100].** Es otro SIEM de pago que hemos visto ser utilizado en algunas organizaciones, aunque su integración con otras herramientas no se tiene mucha documentación, si cuenta con una API que integra la recepción de datos con otras herramientas de terceros. Su fuerte es la recolección de datos sobre el comportamiento de aplicaciones o servidores que por lo general están en la nube. Es una herramienta buena para centralizar eventos, investigación de incidentes y respuesta temprana a amenazas. Permite realizar alertas ante eventos críticos y permite compartir la data que esta recopila.



Infrastructure

See inside any stack, any app, at any scale, anywhere

| Free | Pro | Enterprise |
|---|---|---|
| $0 | $15 Per host, per month* | $23 Per host, per month* |
| Core collection and visualization features<br>- Discussion group supported<br>- 1-day metric retention<br>- Up to 5 hosts | Centralize your monitoring of systems, services, and serverless functions<br>- 400+ integrations<br>- Out-of-the-box dashboards<br>- 15-month metric retention<br>*Billed annually or $18 on-demand | Advanced features and administrative controls<br>- Machine learning-based alerts<br>- Live Processes<br>- Premium support<br>*Billed annually or $27 on-demand 100 host minimum |
| START FREE TRIAL | START FREE TRIAL | START FREE TRIAL |

Imagen 31. Precios de Datadog.

[99] https://azure.microsoft.com/en-us/services/azure-sentinel/#product-overview
[100] https://www.datadoghq.com/

### *2.3.3 Otras alternativas para visualizar datos recogidos.*

Entre las herramientas que investigamos para la detectar actividades sospechosas, vimos los IDS/IPS. Snort ®, Suricata ®, Zeek ®, son herramientas que no trabajan a ciegas, tienen la capacidad de mostrar mediante una consola la salida de los datos en tiempo real. El detalle con estas opciones es que el analista debería estar en la misma localidad donde se estarían generando los eventos. En una organización donde los eventos estarían realizándose en una localidad remota, se necesita de un visor que permita ver esos datos y proceder a su análisis.

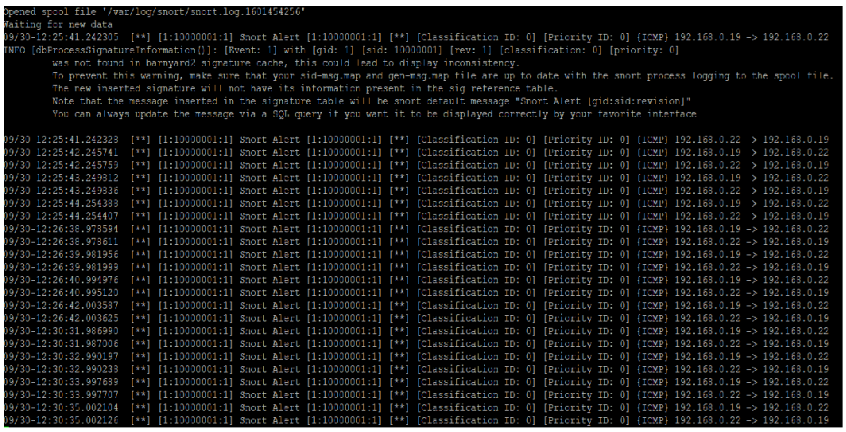



Imagen 32. Logs de Snort.

En una opción donde se tenga acceso al host donde se está ejecutando el IDS, como bien podría ser Snort ®, este va guardando los eventos en el fichero de logs */var/log/snort/snort.log.####* y accediendo a él, podríamos ver los eventos de una forma poco amigable. Esta parte no ofrece ninguna analítica de las que ofrece un SIEM.

En el caso de Suricata, también se podría ver los Logs directamente en el host donde se ha instalado el IDS. Es el mismo caso anterior donde para poder ver esta consola, debemos estar alcanzable a este nodo. Preferiblemente en la misma red, pero igual si se tiene alcance de manera remota, se puede llegar sin problemas con las credenciales correctas.

En el tema de las versiones de licencias privativas, las que hemos mostrado no son las únicas del mercado. Son muchas más, pero como no está en nuestro alcance, preferimos no abundar más allá de lo que hemos incluido en este TFM.

## *2.4 Infraestructura tecnológica para este TFM.*

Para poder realizar esta implementación, vamos a necesitar desplegar ciertas tecnologías que permitirán su despliegue tanto a los recolectores y detectores de actividades sospechosas, como a los SIEMs que van a recolectar y procesar las informaciones que se van a estar generando, entre ellas, los sistemas de virtualización.

### *2.4.1 Alternativas para virtualizar la infraestructura.*

Las organizaciones vienen buscando las maneras de abaratar costes, para eso, la reutilización de sus recursos ha venido a ser un alivio bien grande. Una de las partes donde más se aprovechan los recursos es en la virtualización [101]. Con este procedimiento, se crean máquinas virtuales que ejecutan computación como si fueran en hardware físicos. Se valen del aprovechamiento de los recursos por medio de un

[101] https://en.wikipedia.org/wiki/Virtualization

Hipervisor que hace las funciones de gestionar los recursos y ponerlos a la disposición de las virtuales que vamos a incluir en la infraestructura.

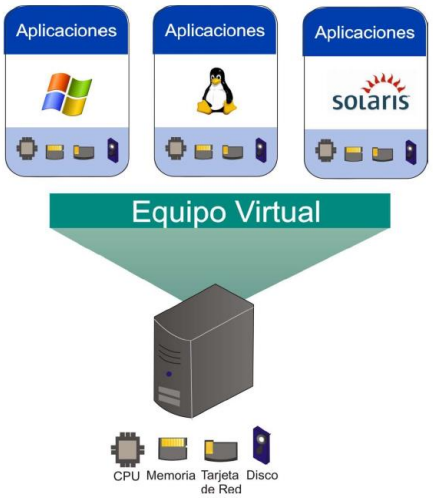



Imagen 33. Entorno virtualizado.

Para la propuesta de este TFM, hemos pensado en virtualizar las instalaciones y proponer al final a las organizaciones que pretendan ya que se reutilizarían recursos ya existentes y no incurrir en gastos extras.

Entre los sistemas para virtualizar más utilizados actualmente, podemos encontrar a **VMWare** ® [102]. Es de licenciamiento privativo y bastante costoso. Tiene uno de los soportes más respetados por lo que grandes organizaciones tienen sus plataformas de virtualización en premisa en este sistema de virtualización. Existen versiones gratuitas para entornos no comerciales y que se puede desplegar en diferentes plataformas, Microsoft ® o GNU/Linux. Su hipervisor es VMWARE ESXi ™[103].

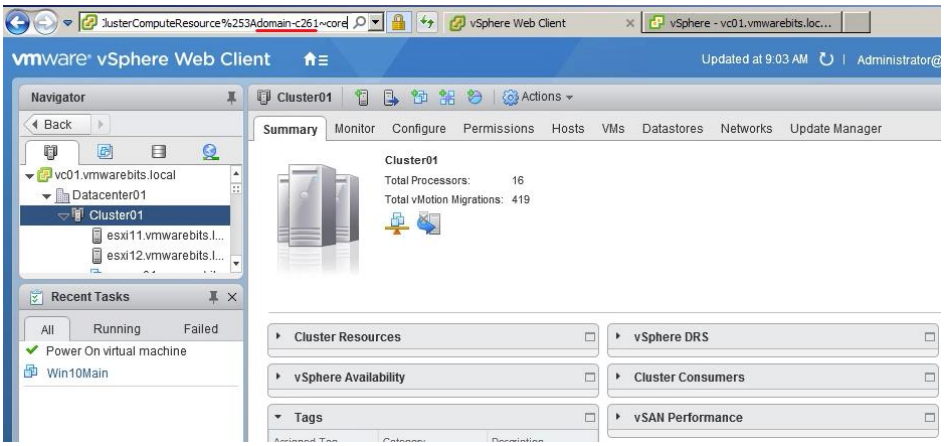



Imagen 34. Consola web VMWare ®.

Desde su consola, se pueden gestionar la administración de toda la infraestructura y sus recursos de apoyo como bien podría ser sistemas de almacenamiento externo o un clúster con alta disponibilidad. En la actualidad, VMWare ® es propiedad de Dell-EMC ® [104] [105] desde diciembre 2003.

[102] https://www.vmware.com/
[103] https://en.wikipedia.org/wiki/VMware_ESXi
[104] https://es.wikipedia.org/wiki/Dell_EMC
[105] https://www.delltechnologies.com/en-us/converged-infrastructure/index.htm

Microsoft ® tiene su sistema de virtualización también. Se le conoce como Hyper-V ® [106] el cual se activa como una función del sistema operativo y se puede utilizar en su funcionabilidad de acuerdo con la licencia con que se esté ejecutando el Sistema Operativo Microsoft ® ya que trae algunas limitaciones por el licenciamiento.

Otras limitaciones es que no funciona en todo hardware por lo que se recomienda antes de ser implementado, leer las especificaciones para poder aprovechar el rendimiento.

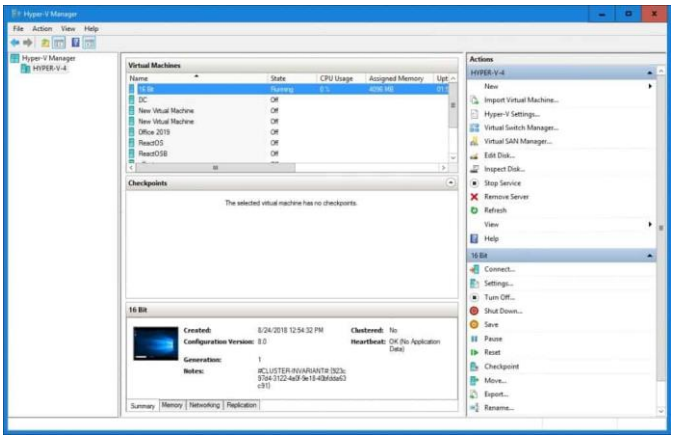


Imagen 35. Consola de Gestion Hyper-V.

Otro sistema de virtualización que actualmente podemos utilizar y este, fuera de licenciamiento privativo ya que se puede desplegar bajo licencia de Código Abierto, es **Proxmox** ®. Igual que otros proyectos de Código Libre, podemos adquirir soporte de pago ya sea en su portal o en distribuidores de cada país.

Proxmox ® ha venido a ser una solución para aquellas organizaciones que no cuentan con un presupuesto elevado para virtualizar su infraestructura y al ser muy robusto, es muy confiable. Su sistema base esta sobre Debian [107] que es una distribución de GNU/Linux y el hipervisor es KVM. Con Proxmox ®, se puede virtualizar sistemas de diferentes plataformas al igual que otros como VMWare ® o Hyper-V ®.

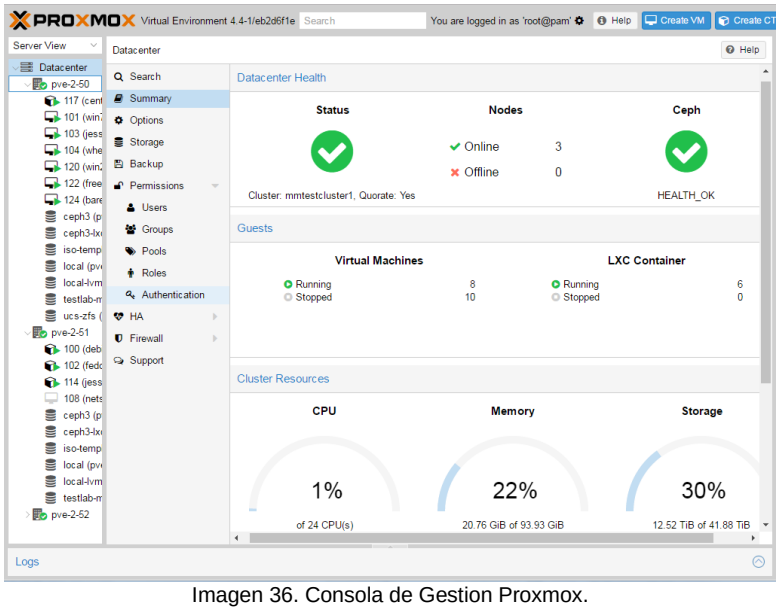



Imagen 36. Consola de Gestion Proxmox.

106 https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/
107 https://www.debian.org/

En nuestro despliegue para presentar este TFM, pretendemos levantas máquinas virtuales en Qemu-KVM que es un sistema para virtualizar que trae consigo la mayoría de las distribuciones GNU\Linux. Es muy similar a Proxmox solo que no trae una consola web, sino un cliente que se llama Virt-Manager-GUI.

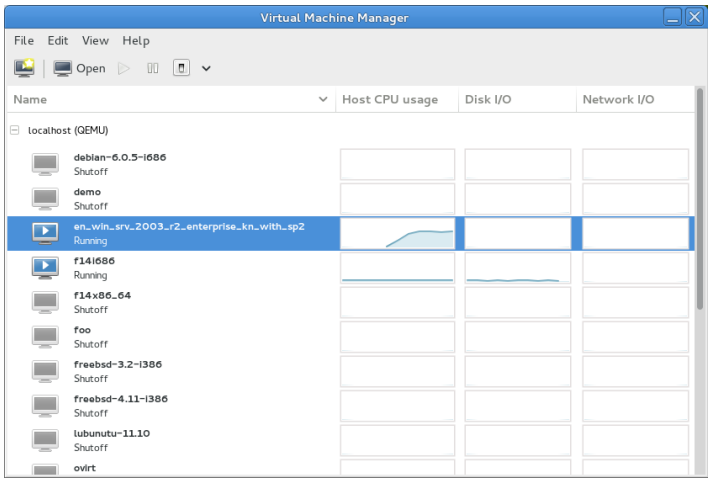


Imagen 37. Consola de Virtual Machine Manager.

Con esta plataforma, podemos inclusive migrar de ella a Proxmox ® ya que los discos que utiliza es en formato Qcow2. Es el soportado por ambos por defecto.

### *2.4.2 Alternativas no virtualizada.*

Una alternativa no virtualizada, es la de utilizar equipos físicos con las capacidades de poder ejecutar ya sea el IDS/IPS o el SIEM para la recogida de los datos y poder visualizarlo.

Dependiendo del IDS/IPs que se pretenda utilizar, se debe considerar las especificaciones del fabricante al igual que cuente con tarjetas de redes suficientes para operar y para poder ser gestionado. Esto, de una manera tangible aumenta el costo del proyecto y solo sería desplegado por una organización que tenga los recursos y que necesariamente se deba implementar de esa manera.

### *2.4.3 Sistemas Operativos.*

Un sistema operativo es un conjunto de aplicaciones que se ejecutan sobre un núcleo y en conjunto, pasan a ser el software principal con la capacidad de gestionar los recursos físicos de los ordenadores y la capacidad de poder brindar una interfaz a los usuarios [108].

Todo dispositivo electrónico que tiene un microprocesador necesita de un sistema operativo para poder funcionar.

Con el pasar de los tiempos, estos han evolucionado en función de las necesidades que han cubierto con el desarrollo de nuevas tecnologías viniendo desde 1954, cuando estos eran grandes consolas para que los programadores operen, luego en los años 1955-1965, llegaron los compiladores [109], ensambladores [110] y cargadores [111] para el manejo de dispositivos y desarrollo de nuevas aplicaciones.

[108] https://es.wikipedia.org/wiki/Sistema_operativo
[109] https://es.wikipedia.org/wiki/Compilador
[110] https://es.wikipedia.org/wiki/Ensamblador
[111] https://es.wikipedia.org/wiki/Cargador_de_programas

A finales de los 80s, se incorporaron nuevas tecnologías como la inclusión de la multimedia. En los años 90s, fue notable el auge de sistemas que hoy día se disputan los mercados enfrentándose los de diferentes licenciamientos y es en 1991 cuando un desarrollador de la universidad de Helsinki de nombre Linus B. Torval [112], anunciaba la creación de un núcleo muy parecido a Unix ® y nace uno de los sistemas operativos paralelizado en su desarrollo ya que aunque comparten el mismo núcleo, tienen diferentes sabores y estilos únicos que denotan la comunidad que les da soporte por sobre todo, nos referimos a GNU/Linux [113].

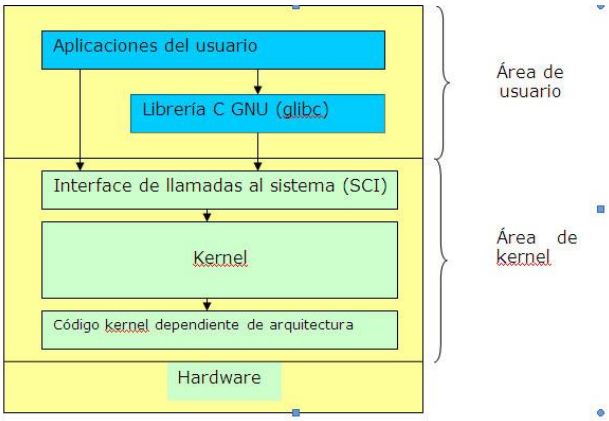



Imagen 38. Arquitectura de GNU/Linux.

Este sistema operativo cumple con la mayoría de los requisitos que necesitamos para este TFM. Es un sistema robusto, demostrado que es bien estable, se ha compilado para diferentes arquitecturas de hardware y lo más importante, se distribuye bajo licencia de Código Libre con la ya conocida GPL [114].

Distribuciones como RedHat ® se distribuyen bajo licenciamiento semiprivatizo, ya que su soporte es de pago, ofrece algunas alternativas como Fedora ® [115], el ya anunciado que desaparecería CentOS ® [116] o CentOS Stream ® [117] que se prevé como la versión comunitaria de RedHat ® luego de anunciar el fin de desarrollo de CentOS [118] y que si se pueden utilizar bajo licenciamiento de Código Libre.

Otra distribución bien conocida es Debian [119]. Es una distribución de GNU/Linux muy versátil, soportada por una comunidad bastante extendida por todo el mundo y que ha permitido que otras distribuciones se desarrollen bajo su raíz y lo más importante es su gran aceptación.

Una de las distribuciones que emergieron bajo esta distribución es UBUNTU ® [120], desarrollado por Canonical en 2004 y distribuida bajo licencia de GNU GPL y una gran variedad de soporte privativo que le han permitido darse a conocer como una alternativa bastante robusta. Su interfaz es bien amigable para quienes quisieran migrar de entornos privativos como Microsoft ®.

Para los fines de este TFM, pretendemos realizar nuestra implementación en este sistema operativo dado que es de amplio uso en organizaciones como en centros de enseñanzas y, al igual que sus iguales, se puede descargar libremente.

[112] https://es.wikipedia.org/wiki/Linus_Torvalds
[113] https://es.wikipedia.org/wiki/GNU/Linux
[114] https://es.wikipedia.org/wiki/GNU_General_Public_License
[115] https://getfedora.org/
[116] https://www.centos.org/
[117] https://www.centos.org/centos-stream/
[118] https://www.muylinux.com/2020/12/09/centos-8-stream-descontinuado-rolling-release/
[119] https://www.debian.org/
[120] https://ubuntu.com/

## *2.5 configuración del Proyecto*

La configuración de este proyecto se propone desde el principio de una manera que se pudiera implementar en cualquier organización reutilizando los recursos de las organizaciones que pretendan desplegar esta implementación.

Por parte del SOC que también podría ser un CSIRT, este se compone de una estructura de análisis y respuesta a incidentes. Este proyecto pretende cubrir la parte del análisis brindando una herramienta a los analistas para poder visualizar actividades en una red y poder analizar actividades sospechosas.

### *2.5.1 Diagrama de Red.*

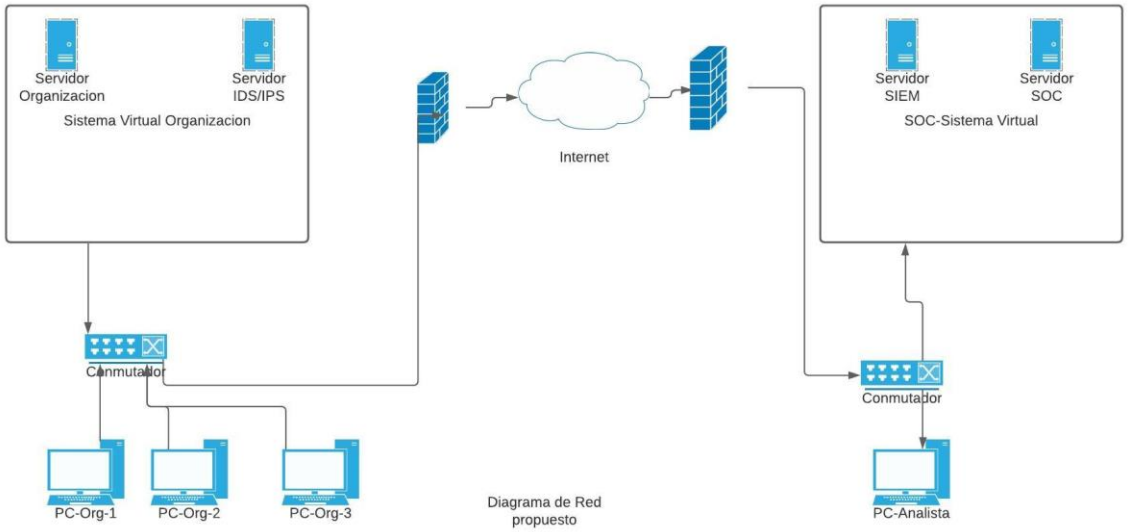



Imagen 39. Diagrama de Red.

Este diagrama de red está compuesto por dos localidades distantes que no necesariamente deberían estar en el mismo edificio pudiendo inclusive estar subcontratado.

Cada localidad con su sistema de servicio de virtualización debidamente conectado a la internet con los servicios alcanzables.

Cada localidad contaría con una estructura de red interna para el uso compartido de recursos de la organización.

Cada localidad contara con equipos periféricos que controlarían los accesos a las localidades remotas permitiendo la conexión segura entre los IDS/IPS y los SIEM.

Las estaciones de trabajo como los servicios a lo interno de la organización en sus respectivos servidores, no se verían afectados por la implementación de esta herramienta siendo la misma desplegada en una configuración y ubicación completamente transparente.

Para la configuración de cada IDS/IPS, se necesitará mínimo dos tarjetas de red. Una que estaría inspeccionando la red interna y otra con fines de conectarse a los SIEM.

*2.5.2 Inventario de equipos.*

| Equipo/Software | Descripción | Utilidad |
|---|---|---|
| Laptop Lenovo® Ideapad ® Flex | Simulador de los servidores. | Instalación de Sistemas para virtualizar. |
| Laptop marca HP ® | Simulando los usuarios | Instalación de sistemas usuarios |
| Sistema operativo Fedora ® | Host anfitrión | Instalación de los servidores. |
| Sistema operativo Microsoft Windows 10 ® | Simulación de usuarios de red corporativa | Pruebas de actividades sospechosas. |
| Qemu-KVM | Sistema para virtualizar los servidores | Se estaría utilizando en nuestro entorno de pruebas y desarrollo. |
| Suricata | Analizador de Red IDS/IPS | Detector de intrusos y actividades sospechosas. |
| ELK. | Sistema para visualizar Eventos. | SIEM. |
| Switch CISCO ® | Conmutador de red para simular la organización. | Conexión equipos. |
| Ubuntu 18.04 | Sistema Operativo | Sistema Operativo para las tecnologías elegidas. |

Tabla 7. Inventario útil.

Este inventario se utilizaría tal cual fue descrito en el capítulo primero.

# 3. Implementación

## *3.1 instalación Sistemas operativos [Ubuntu 18.04].*

Para la implementación de este TFM y las tecnologías que hemos elegido (Suricata, ELK y Ubuntu 18.04), vamos a instalar primero el sistema operativo para esta implementación. Descargamos Ubuntu Server desde la página web oficial [121]

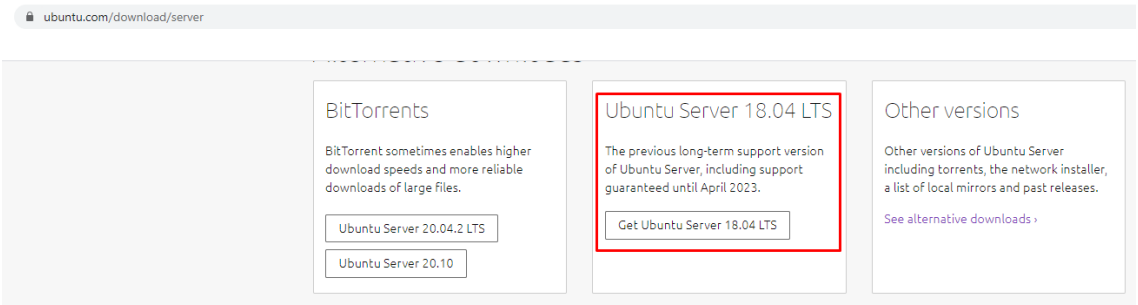



Imagen 40. Descarga Ubuntu Server 18.04 LTS.

Vamos a descargar e instalar esta versión ya que aún mantiene soporte hasta 2023 y, casi todos sus paquetes han sido probados desde su lanzamiento.

Una vez descargado el fichero en formato ISO, lo procedemos a instalar en Qemu-KVM ya instalado previamente en Fedora 32 ®. La instalación la realizamos utilizando como guía la documentación oficial de Ubuntu [122]

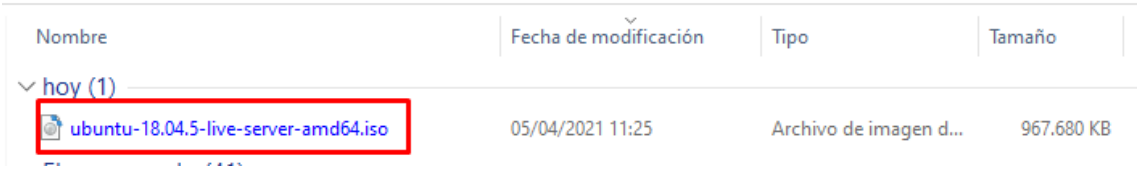



Imagen 41. Imagen de Ubuntu Server 18.04 LTS.

Ya con estos ficheros y esta documentación podemos realizar la instalación.

Qemu-KVM viene instalado en mi distribución de escritorio que he utilizado durante todo el Máster. Esta instalación de Fedora es la versión 32 ®.

Para fines de este TFM, solo la utilizaremos simulando un entorno virtual de una organización haciendo la salvedad que en caso de que la organización donde se esté implementando, si tienen implementado Proxmox ®, la migración seria casi al instante ya que ambos utilizan KVM como Hipervisor y el formato de discos es qcow2 en ambos casos.

Una vez situada la imagen en el equipo con Fedora, procedemos con la instalación de Ubuntu.

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

122 https://help.ubuntu.com/18.04/serverguide/installing-from-cd.html

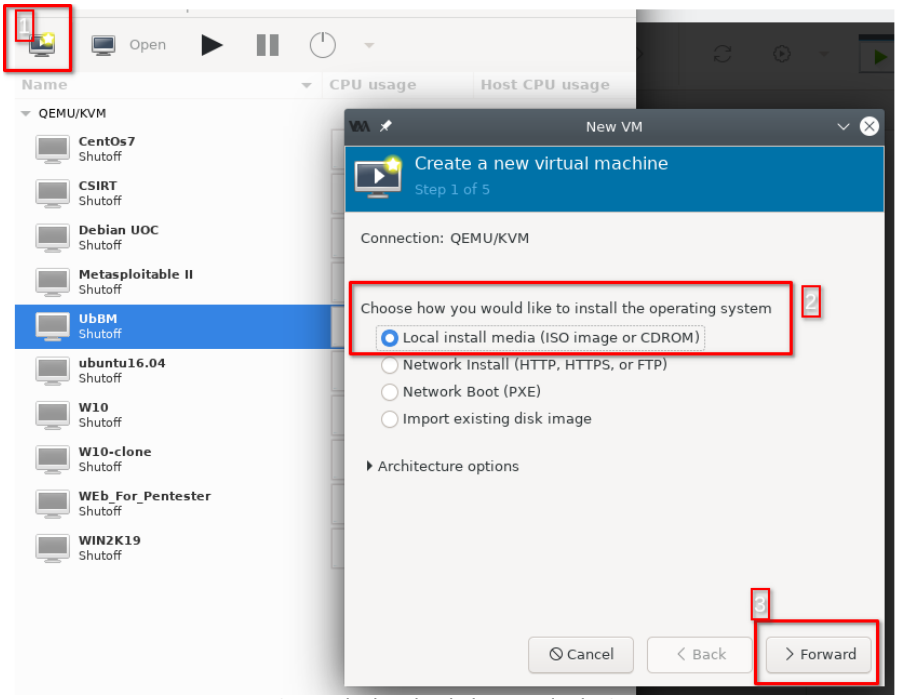



Imagen 42. Instalacion desde la consola de Qemu-KVM.

Una vez lanzamos la consola de virtualización, le damos a nueva máquina. Acto seguido elegimos *Local install media (ISO image or CDROM)* y le damos a *Forward.*

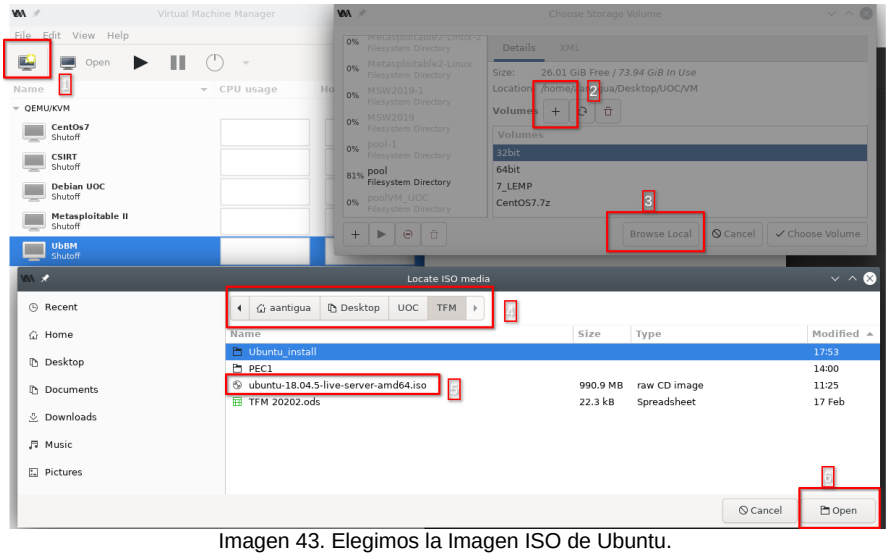



Imagen 43. Elegimos la Imagen ISO de Ubuntu.

En la siguiente ventana, elegimos la imagen ISO de Ubuntu buscando nuevo volumen en búsqueda local y eligiendo el fichero en el directorio donde lo hemos guardado.

Algunas veces nos solicita cambiarle los permisos al fichero .ISO, para que el hipervisor pueda tener acceso a este y poder montarlo en la consola virtual. Otras veces no es necesario dado que lo situamos en un directorio que pertenece al dominio y control de este hipervisor.

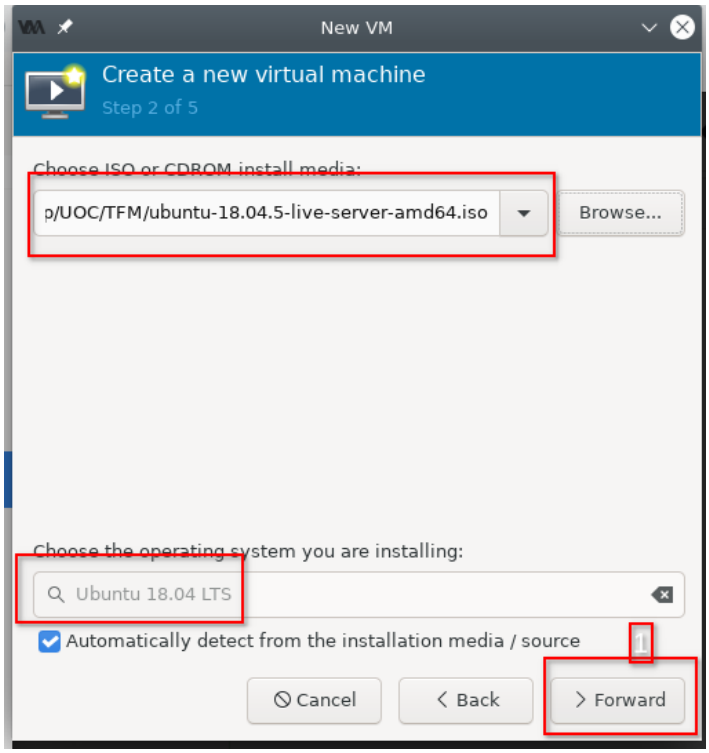



Imagen 44. Avanzamos con la ISO de Ubuntu.

Una vez tenemos la ISO que necesitamos, el mismo hipervisor de manera automática sabe qué sistema Operativo es que va a instalar por lo que solo es dar click en *Forward* y avanzamos a la siguiente ventana.

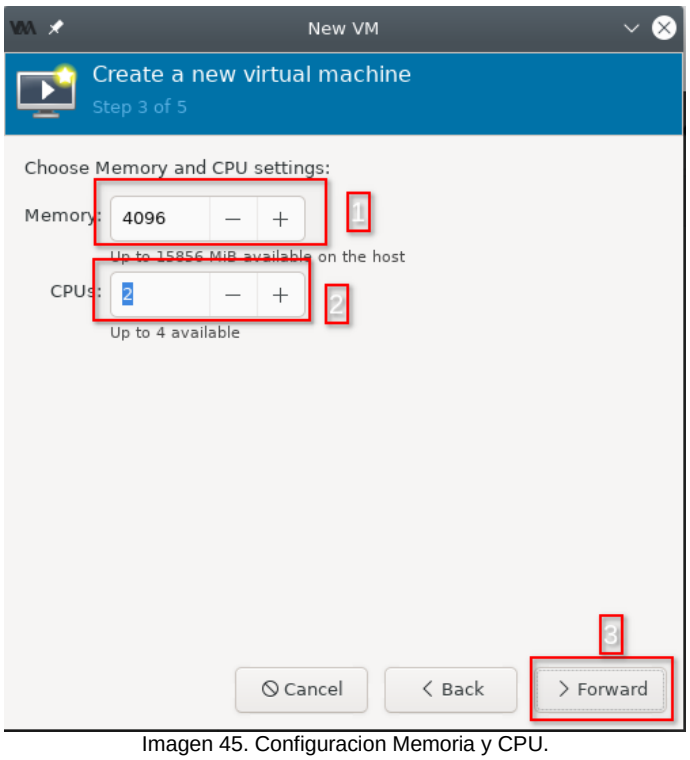



Imagen 45. Configuracion Memoria y CPU.

Para nuestro caso, elegimos en esta ventana la cantidad de memoria y cuantos CPU vamos a utilizar. Elegimos 2 de 4 procesadores y 4 GB de memoria de 16 GB disponible.

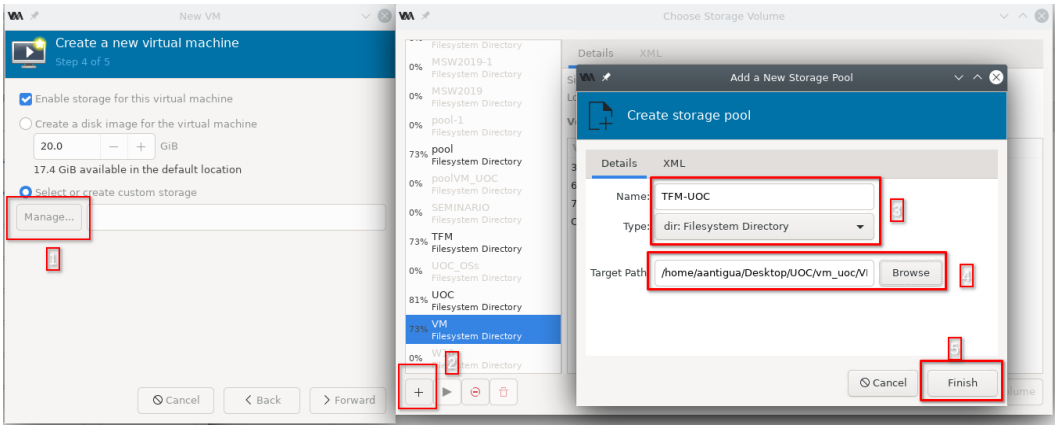



Imagen 46. Configuracion Pool de almacenamiento.

En la siguiente ventana, configuramos un Pool de discos virtuales para alojar nuestro almacenamiento. Lo nombramos y luego elegimos el directorio donde estarán los discos virtuales. Luego que terminamos, le damos a *Finish* para avanzar.

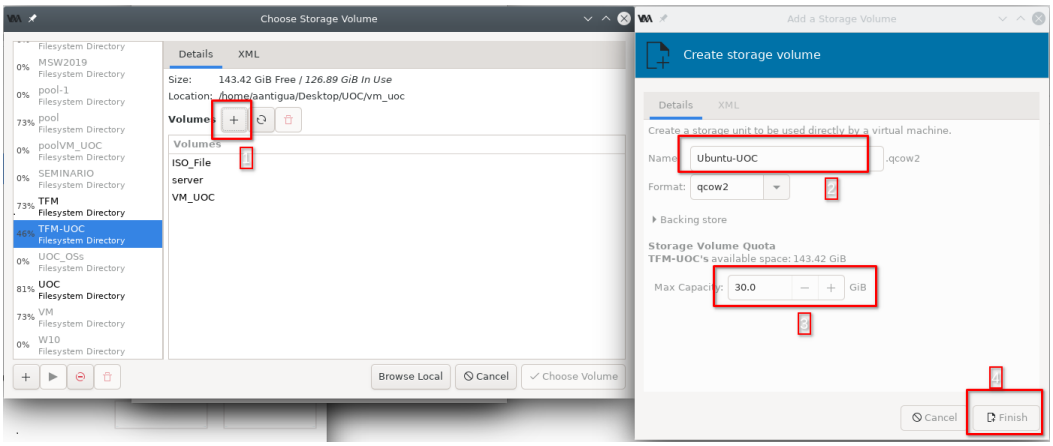



Imagen 47. Configuracion de almacenamiento.

En la siguiente ventana, creamos nuestro disco virtual donde se va a alojar el sistema operativo. Damos a + en Volume y luego en la siguiente ventana, renombramos y asignamos la capacidad máxima ya que será de almacenamiento dinámico. Una vez asignado los valores, le damos click a *Finish"* y avanzamos a la siguiente ventana.

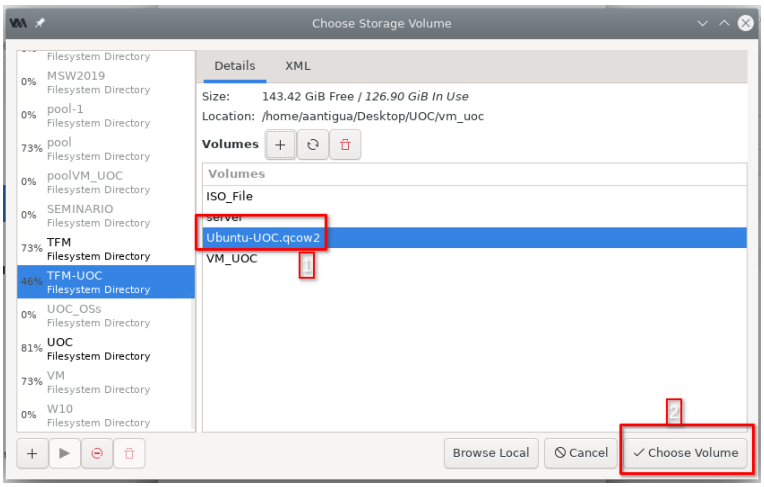



Imagen 48. Eleccion de Disco virtual.

En la siguiente ventana, elegimos el disco recién creado como destino de almacenamiento principal.

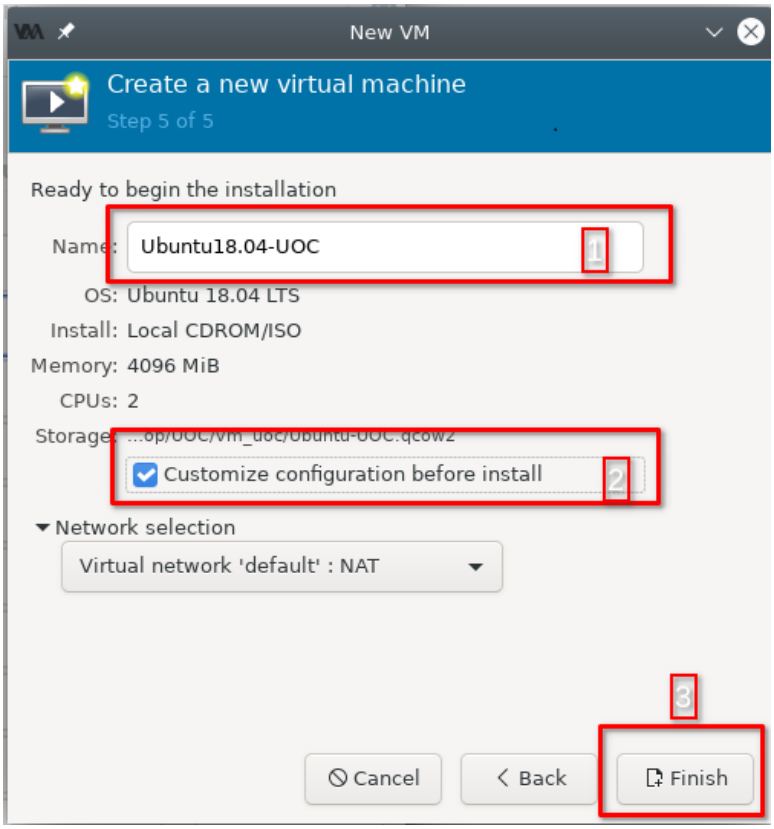



Imagen 49. Finalizacion.

En la siguiente ventana, finalizamos la parametrización de la máquina virtual que queremos. Como no le pudimos agregar una segunda tarjeta de red, luego de renombrar la maquina le damos click para elegir customizar la configuración y al dar click a *finish*, este nos lleva a la ventana de configuración avanzada.

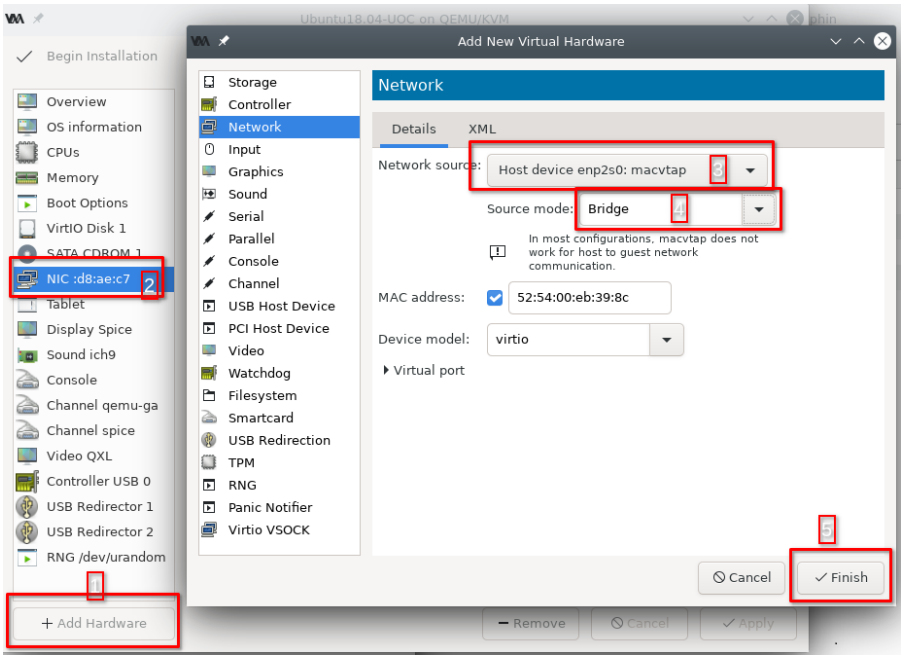



Imagen 50. Configuracion avanzada.

En esta ventana, damos click a *Add hardware* y elegimos a *Network* para que nos agregue otra tarjeta de red. La asignamos como Bridge de la tarjeta de red Ethernet. Damos click a *Finish* y listo. Luego a esto, procedemos con la instalación del sistema operativo.

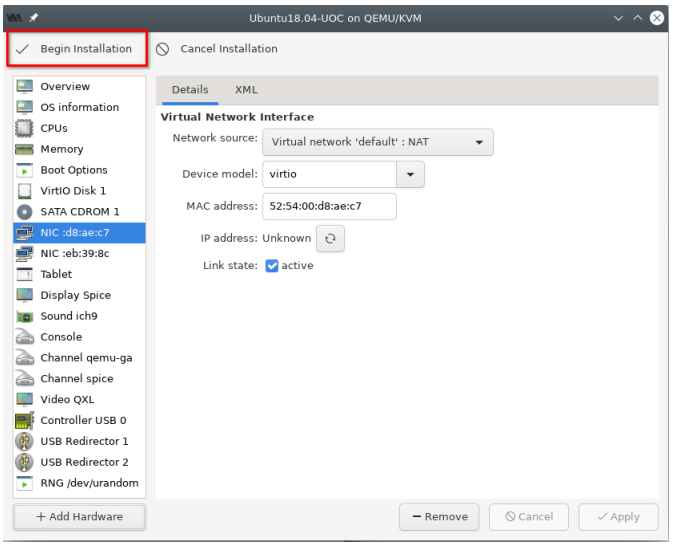


Imagen 51. Inicio instalacion Sistema Operativo.

Una vez finalizada la configuración avanzada, en la parte superior izquierda localizamos a *Begin Installation* y al dar click, nos enciende la máquina virtual e inicia la instalación llamando la ISO que hemos asignado.

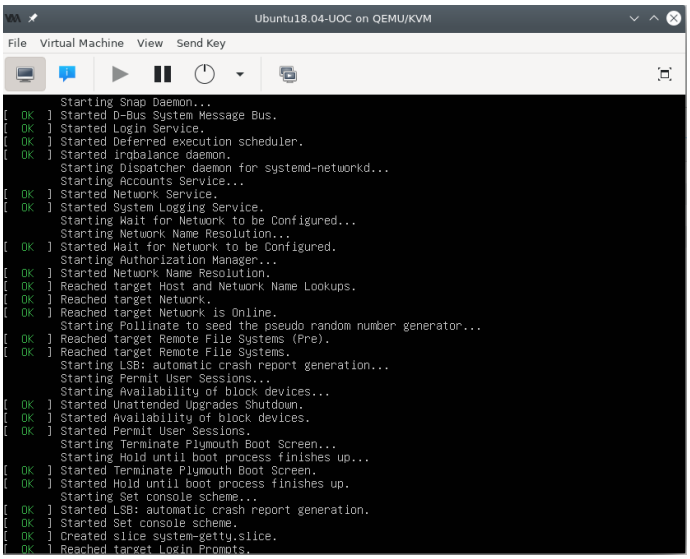


Imagen 52. Instalacion Sistema Operativo.

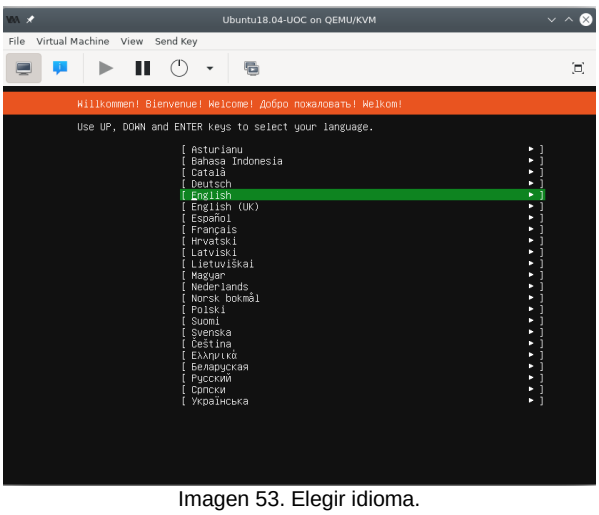


Imagen 53. Elegir idioma.

Una vez inicie el Sistema de instalación, este nos presenta los procesos que van cargando previo a la instalación. Acto seguido, nos presenta la opción para elegir el idioma.

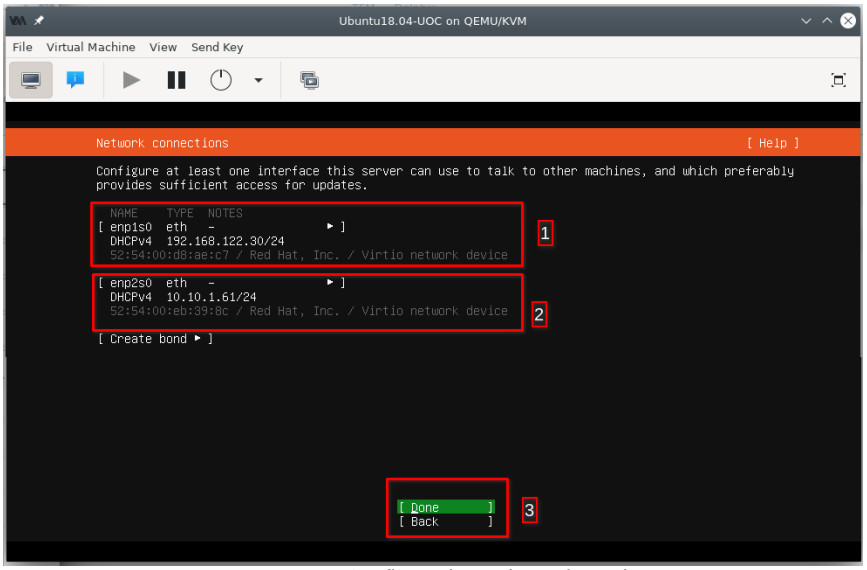



Imagen 54. Configuracion tarjetas de Red.

Ya luego del idioma, elegimos la configuración de la tarjeta de red. Por ahora, ambas tarjetas la ponemos DHCP.

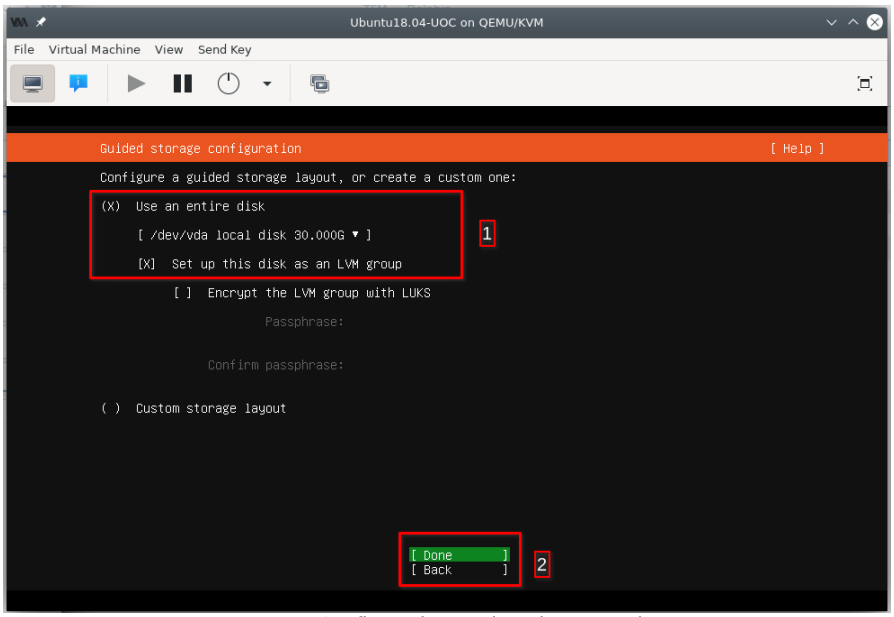



Imagen 55. Configuracion Basica Almacenamiento.

En la siguiente ventana, vamos a configurar los discos. Para nuestro caso, vamos a realizar configuraciones por defecto sin cifrado.

Para este caso, elegimos todo el disco y lo dejamos en LVM por si más adelante queremos gestionar el particionamiento ya sea agregando más almacenamiento por algún crecimiento de una base de datos.

Ya en la siguiente ventana, confirmamos las particiones que vamos a tener en nuestra instalación.

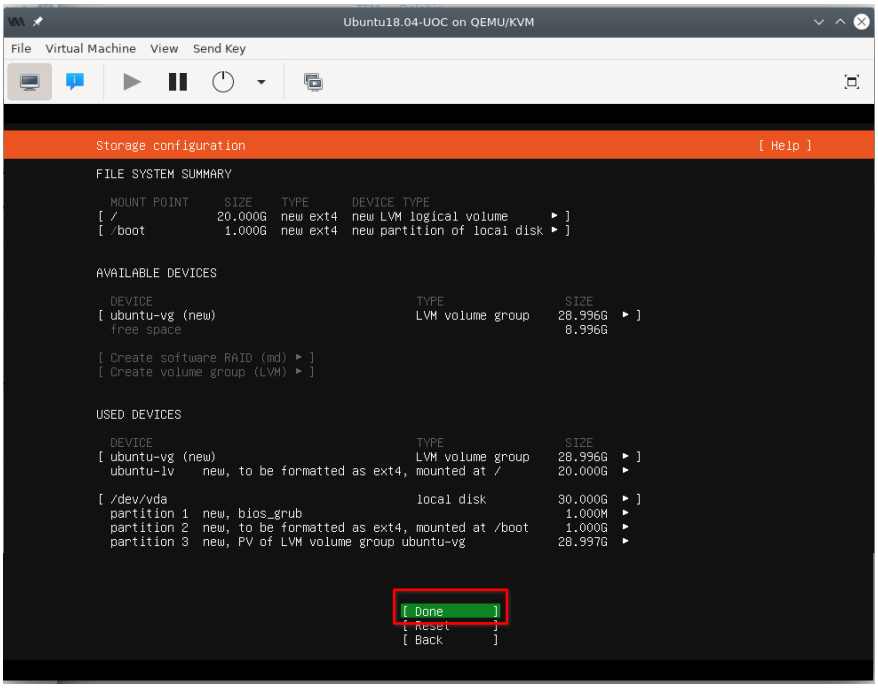



Imagen 56. Configuracion de Almacenamiento.

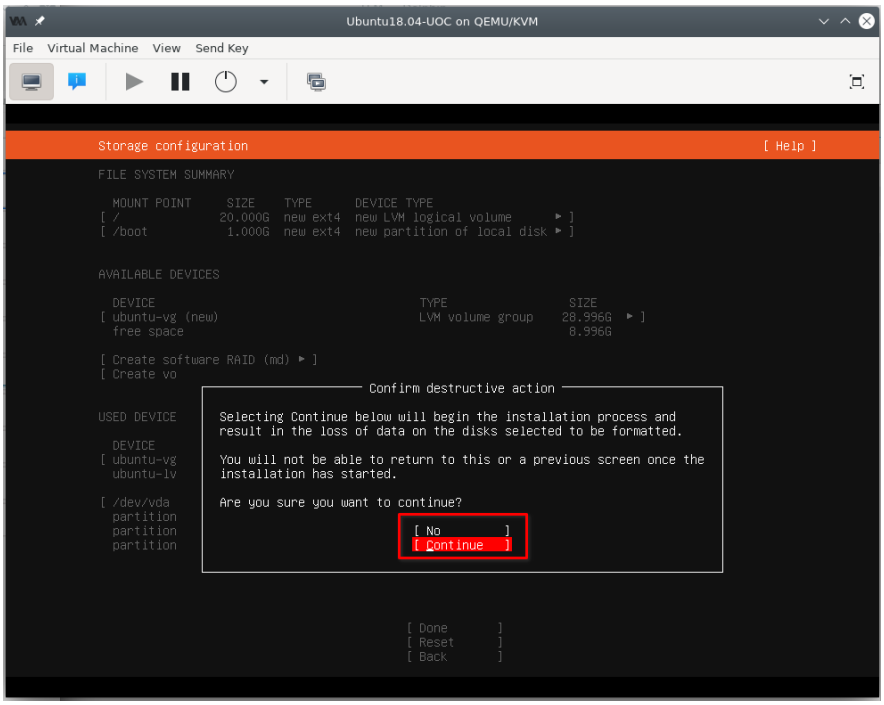



Imagen 57. Configuracion borrado disco duro.

Una siguiente ventana nos indica que las particiones serán creadas y no habrá manera de revertir ese proceso. Ya con esta información, el proceso de instalación tiene definido como seria la instalación del sistema operativo. Las siguientes ventanas, ya son las que gestionarían la parte de gestión de credenciales y acceso al Sistema Operativo.

Definimos el usuario, la contraseña y el nombre del servidor. Una vez realizado este proceso iniciaría el proceso de instalación final del sistema operativo.

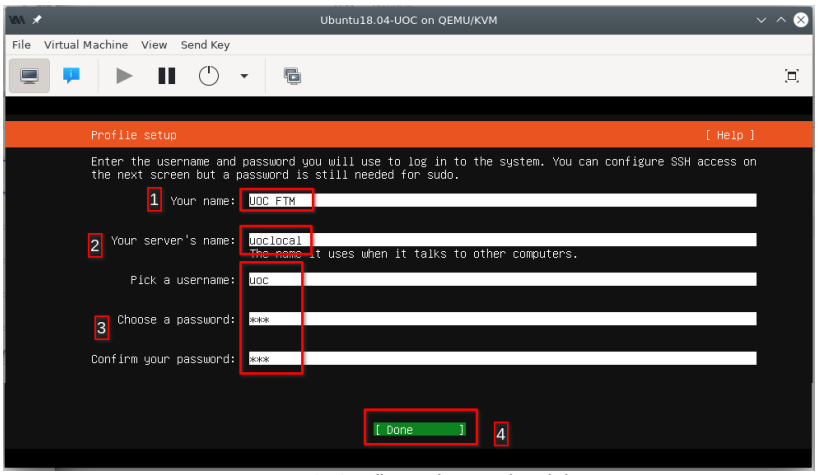



Imagen 58. Configuracion credenciales.

Una vez culminado el proceso de instalación, nos pedirá reiniciar.

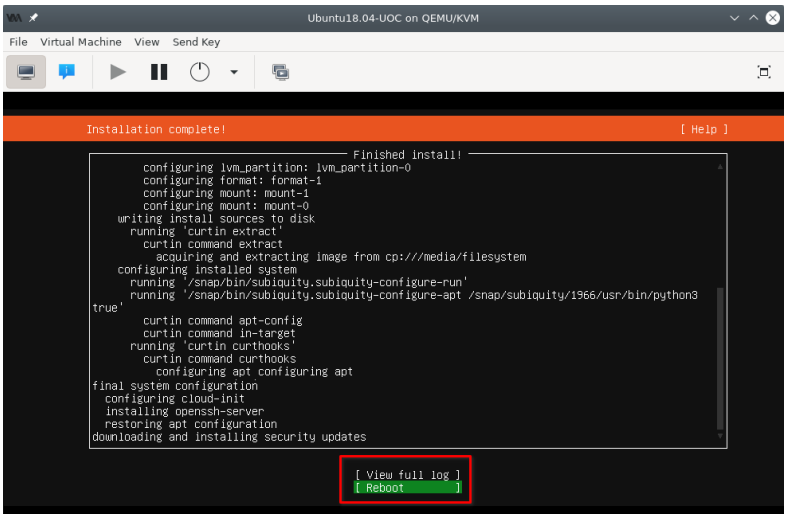



Imagen 59. Reinicio post-instalacion.

Una vez que reiniciamos la máquina virtual, nos pedirá las credenciales para ingresar y listo. La virtual ha sido instalada.

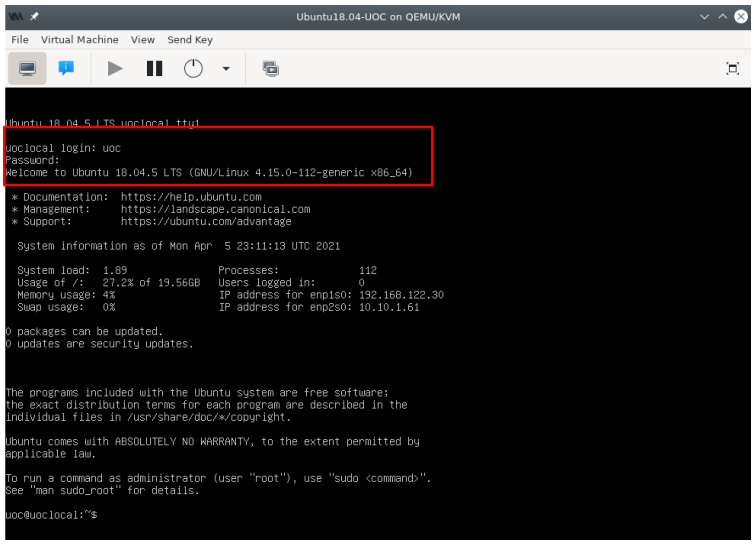



Imagen 60. Inicio del Sistema Operativo.

Una vez culminada la instalación, procedemos a realizar un clonado de la máquina virtual como medida de contención y para evitar tener que realizar nuevas instalaciones.

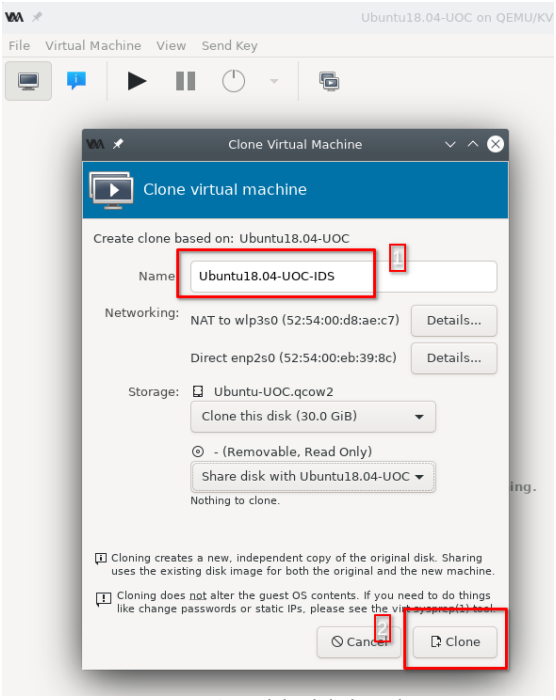



Imagen 61. Inicio del clonado.

Al dar click derecho sobre la maquina apagada, damos *Clone* y nos sale esta ventana. Luego en esta ventana, renombramos y luego damos al botor *Clone* y nos envía a otra ventana donde esperamos que el proceso culmine.

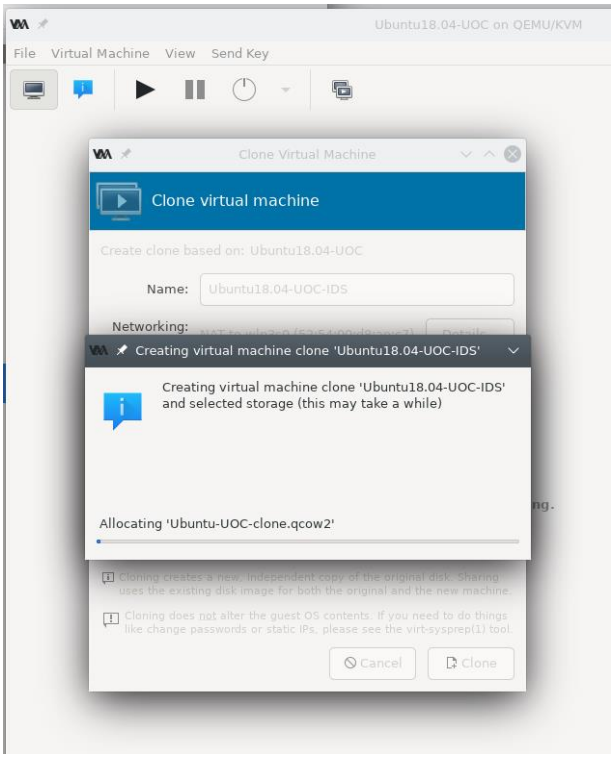



Imagen 62. Proceso de clonado.

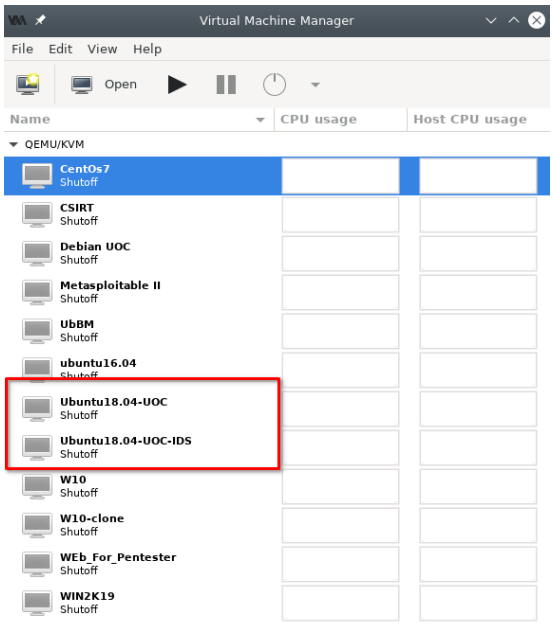



Imagen 63. Consola de Maquinas.

Ya en esta ventana, vemos las dos máquinas virtuales listas para utilizarse.

## *3.2 instalación Sistemas de Monitorización de red.*

Para este apartado, vamos a visualizar la manera en que se instalaría un sistema NISD como Suricata.

Vamos a instalar este sistema con la capacidad de visualizar las tramas que pasen por ciertos puertos de nuestra red y que el motor de IDS identifique los paquetes de las capas superiores

### *3.2.1 Instalación IDS [Suricata].*

Para la instalación del IDS, hemos tomado como referencia la documentación oficial [123] y hemos procedido de la siguiente manera.

Actualizamos los repositorios de la siguiente manera.

```
uoc@uoclocal:~$ su -
Password:
root@uoclocal:~# apt-get update
Hit:1 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://archive.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... Done
root@uoclocal:~#
```

Imagen 64. Actualizacion repositorios.

Una vez actualizado los repositorios, procedemos a instalar las dependencias primero.

Estos son los prerrequisitos que necesita Suricata para ejecutarse.

123 https://suricata.readthedocs.io/en/latest/install.html

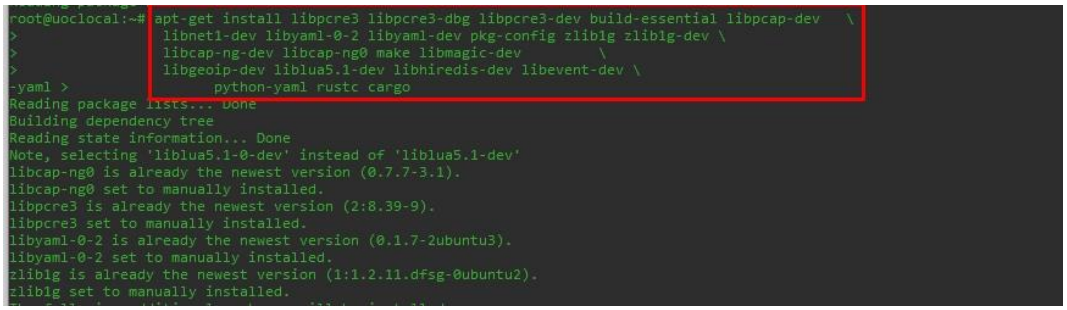


Imagen 65. Instalacion prerequisitos.

De esta manera, ya tenemos los paquetes que son dependencias del servicio. Acto seguido procedemos a agregar los repositorios de Suricata.

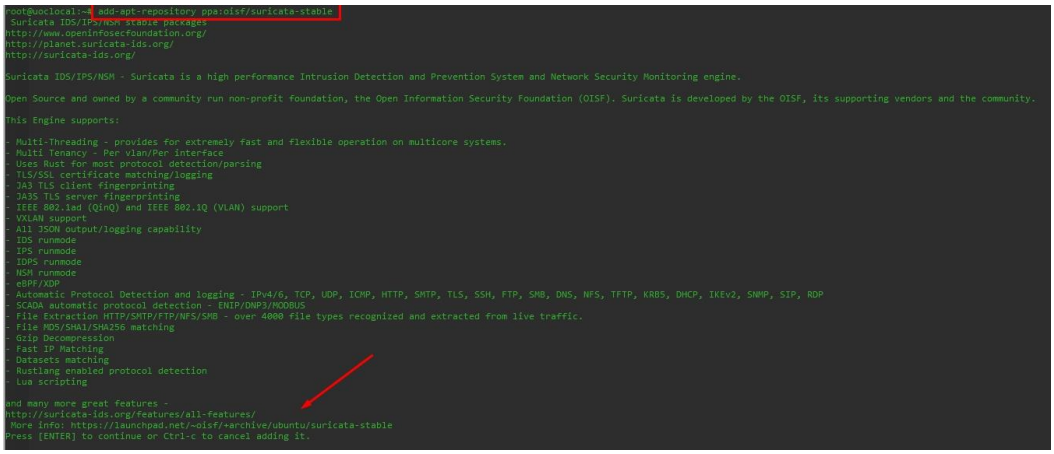


Imagen 66. Instalacion prerequisitos.

Con estos repositorios listos en nuestro sistema, procedemos a actualizar nuevamente para que los repositorios se sincronicen y podamos tener una lista de paquetes disponibles.

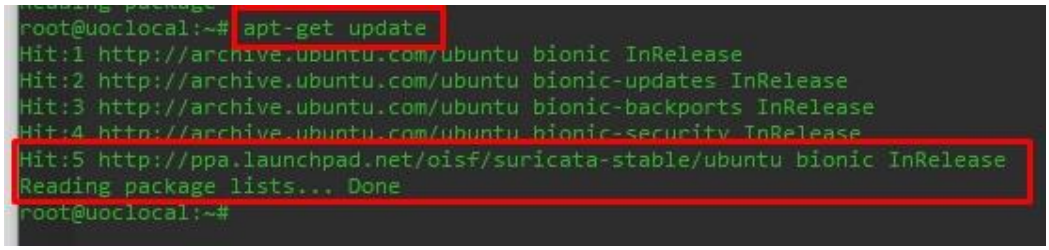


Imagen 67. Actualizacion repositorios.

De esta manera, ya nuestro sistema tiene listo estos repositorios y podemos instalar el paquete principal de nuestro servicio.

Imagen 68. Instalacion Suricata.

De esta manera lo que hacemos es instalar Suricata desde los repositorios oficiales que hemos agregado en nuestro sistema operativo. Luego verificamos que el servicio

estaría listo para ejecutarse y lo agregamos a que se ejecute al inicio del sistema operativo.

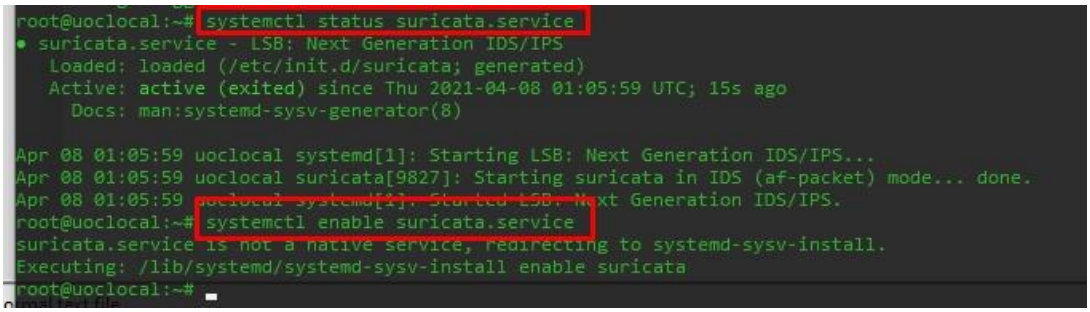


Imagen 69. Configuracion inicio servicio.

### *3.2.2 Configuración IDS [Suricata]*

Como no tenemos tiempo para la creación de reglas personalizadas, nos vamos a socorrer del portal de emergingthreats.net [124] y vamos a utilizar el aplicativo OinkMaster para mantener estas reglas actualizadas. Procedemos a instalarlo.


Imagen 70. Instalacion OinkMaster.

Luego a esto, debemos configurar el fichero de OinkMaster para poder descargar las reglas del portal antes mencionado.

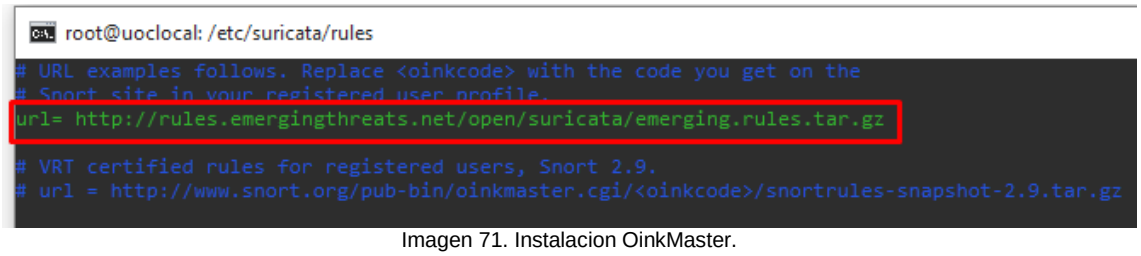


Imagen 71. Instalacion OinkMaster.

Ya con esta configuración, cada vez que ejecutemos *oinkmaster*, este va a actualizar lo que según la url que le hemos asignado.

Imagen 72. OinkMaster en ejecucion.

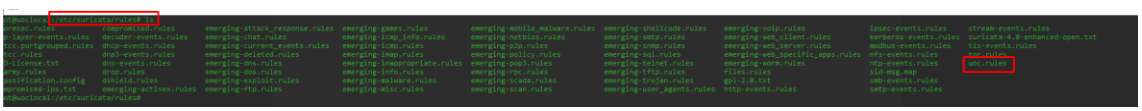


Imagen 73. Lista de reglas.

124 https://rules.emergingthreats.net/open/suricata/

Al verificar el directorio /etc/suricata/rules, podemos ver que nos ha descargado ya varias reglas que podemos utilizar en nuestras pruebas y en nuestra puesta en producción. Son reglas ya bien utilizadas por lo que podemos utilizar a confianza. Como no vamos a necesitar todas esas reglas, he creado un fichero para agregar las reglas que estaré necesitando. uoc.rules es el fichero que he creado. En este, vamos a estar agregando nuestras reglas. Ya por último estaré modificando la configuración de Suricata para que me lea esa regla y proceso a ejecutarlo.

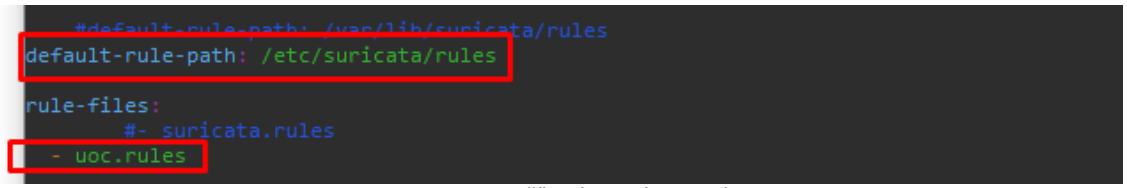



Imagen 74. Modificacion suricata.yml.

Iniciamos el servicio de Suricata.

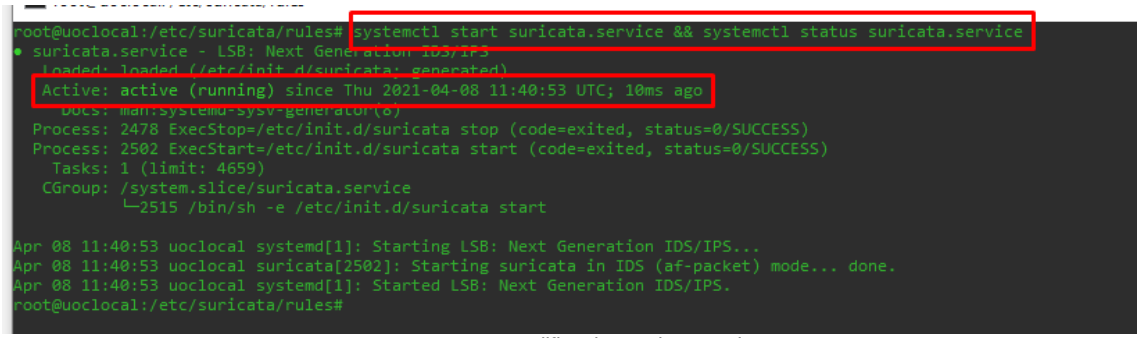



Imagen 75. Modificacion suricata.yml.

Ya de esta manera, tenemos nuestro IDS funcionando. Lo probamos de la siguiente manera agregando unas reglas básicas (Detectar ICMP o Ping) al fichero /etc/suricata/rules/uoc.rules.

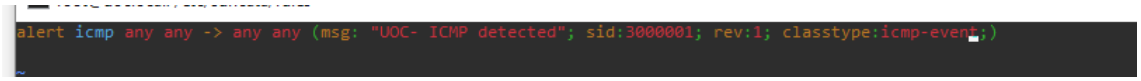



Imagen 75. Modificacion uoc.rules.

Ahora reiniciamos el sistema operativo y probamos.

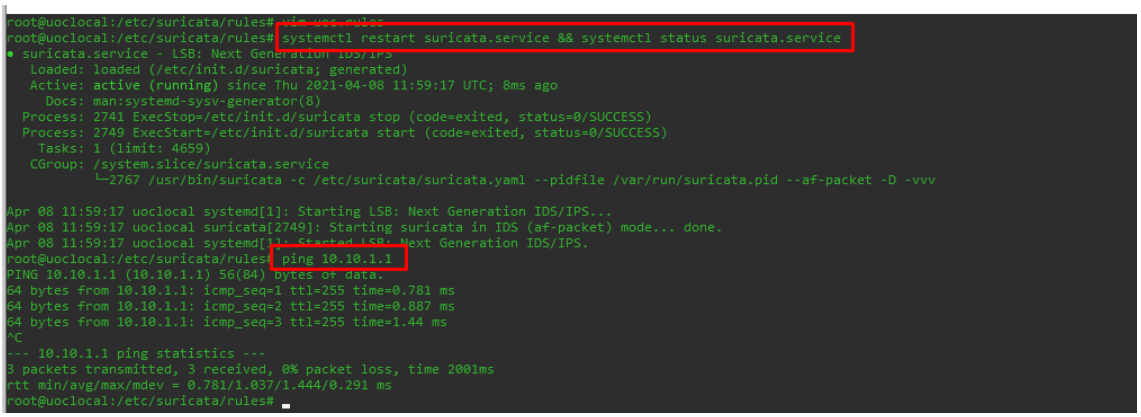



Imagen 76. Reiniciar servicio.

Ahora verificamos los logs de Suricata y listo.

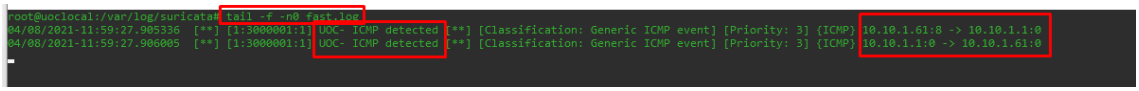



Imagen 77. Log de Suricata.

Para que nuestro IDS funcione de la manera que queremos, debemos realizar algunos ajustes en los Switches que vamos a estar conectados. Es necesario que el puerto físico donde se conecte la Suricata tenga la capacidad lógica de poder escuchar los demás puertos ya que, como función principal de un Switch, es que cada puerto sea independiente uno del otro evitando de esta manera que surjan colisiones de tramas [125]. Para este objetivo, configuramos la tarjeta física en modo Passthrough que sería algo parecido al modo promiscuo y configuramos un puerto que será el que este escuchando o recibiendo una copia de las tramas que pasan por otro puerto. En el Switch que hemos dedicado para este proyecto es un Catalyst 2950 y su configuración es bien simple. Tomamos como referencia estas indicaciones para ponerlo en modo Spam [126]

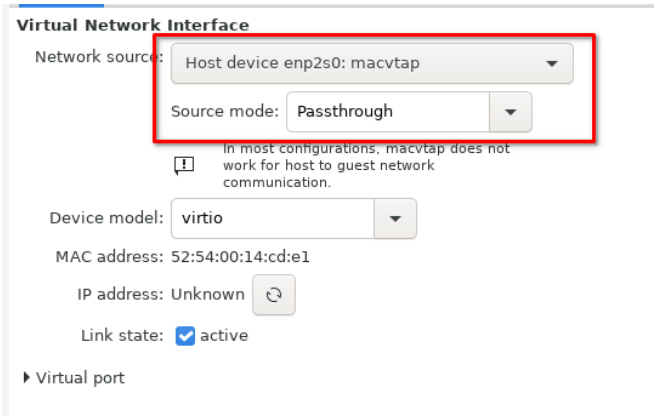



Imagen 78. Tarjeta Fisica.

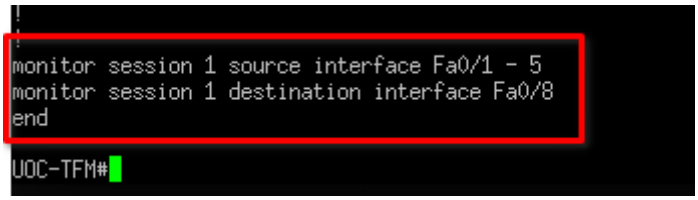



Imagen 79. Configuracion del Switch.

Como se puede ver en la imagen anterior, la tarjeta física de Suricata estaría conectada en el puerto Fa0/8 del Switch y los equipos que vamos a monitorear estarían conectados en los puertos Fa0/1 – 5. De esta manera capturamos las tramas y al conectar Suricata en este puerto, este tendrá una copia de las tramas que pasan por cada puerto que está en declaradas como “source”.

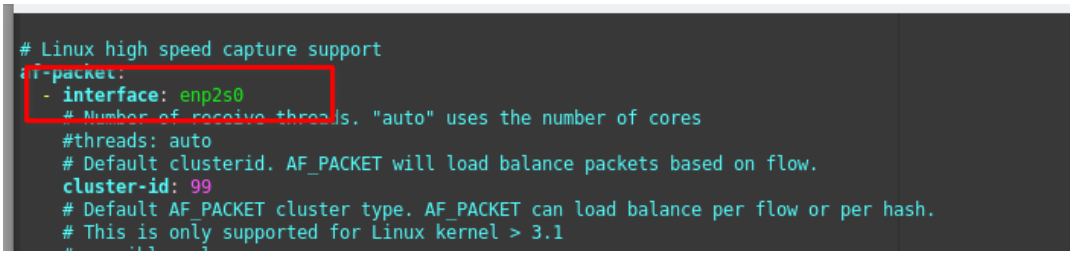



Imagen 80. Configuracion del Switch.

125 https://es.wikipedia.org/wiki/Dominio_de_colision

126 https://www.cisco.com/c/en/us/support/docs/switches/catalyst-6500-series-switches/10570-41.html

De esta manera dejamos definido que tarjeta estaría escuchando nuestro NIDS.

### *3.2.3 Definición y pruebas de Reglas para Actividades sospechosas de la Red.*

Para la gestión de las reglas, hemos optado por utilizar las reglas de **Emergin Threats** [127] como hemos indicado en el subapartado anterior dado que no es necesario pagar una suscripción y se actualizan casi a diario y mantenemos el objetivo de utilizar siempre Tecnología de Código Abierto liberando la organización de tener que costear por pagos de suscripción y licenciamiento.

En dado caso que se quiera configurar una regla aparte a las que descarga Oinkmaster, se puede realizar creando un fichero con extensión .rules en /etc/suricata/rules/ y luego ingresar el nombre del fichero tal cual lo hicimos y evidenciamos en la imagen 74.

Para trabajar con las reglas de Emergin Threats, ejecutamos el aplicativo Oinkmaster de la siguiente manera.

```
root@uoclocal ~# oinkmaster -C /etc/oinkmaster.conf -o /etc/suricata/rules
Loading /etc/oinkmaster.conf
Downloading file from http://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz...
```

Imagen 81. Descarga reglas Emergin Threats.

Una vez descargadas las reglas y aplicadas por Oinkmaster, las reglas quedarían de esta manera.

```
default-rule-path: /etc/suricata/rules
rule-files:
  - botcc.rules
  - ciarmy.rules
  - compromised.rules
  - drop.rules
  - dshield.rules
# - emerging-activex.rules
  - emerging-attack_response.rules
  - emerging-chat.rules
  - emerging-current_events.rules
  - emerging-dns.rules
  - emerging-dos.rules
  - emerging-exploit.rules
  - emerging-ftp.rules
# - emerging-games.rules
  - emerging-icmp_info.rules
# - emerging-icmp.rules
  - emerging-imap.rules
# - emerging-inappropriate.rules
  - emerging-malware.rules
  - emerging-misc.rules
  - emerging-mobile_malware.rules
  - emerging-netbios.rules
  - emerging-p2p.rules
  - emerging-policy.rules
  - emerging-pop3.rules
  - emerging-rpc.rules
  - emerging-scada.rules
  - emerging-scan.rules
# - emerging-shellcode.rules
  - emerging-smtp.rules
  - emerging-snmp.rules
  - emerging-sql.rules
  - emerging-telnet.rules
  - emerging-tftp.rules
  - emerging-trojan.rules
  - emerging-user_agents.rules
  - emerging-voip.rules
  - emerging-web_client.rules
  - emerging-web_server.rules
# - emerging-web_specific_apps.rules
  - emerging-worm.rules
  - tor.rules
  - uoc.rules
```

Imagen 82. Configuracion reglas Emergin Threats.

Luego a esto, verificamos el fichero /var/log/suricata/eve.json y vemos que está capturando las tramas de acuerdo con las reglas declaradas.

127 https://rules.emergingthreats.net/open/suricata/rules/

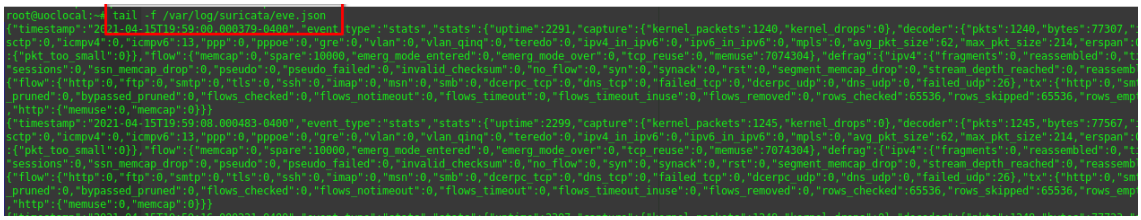


Imagen 83. Fichero eve.json.

Siempre será necesario tener en cuenta que la estructura de las reglas deberá cumplir con una nomenclatura especifica [128]. Para mejor facilidad, copiar una regla existente y modificarle los parámetros es la manera más factible. Una vez que apliquemos nuevas reglas, debemos reiniciar el servicio para cargar la nueva configuración.

## *3.3 Instalación Sistemas Operativo para SIEM.*

Este apartado no presenta diferencia con el 3.1 De hecho, solo hemos copiado la misma máquina virtual y le hemos dejado una de las tarjetas de red para este propósito en particular. De esta manera procedemos a instalar nuestro SIEM recordando que este puede variar de acuerdo con la configuración de la organización donde se va a instalar. Para nuestra propuesta, asumimos que le SIEM está en una localidad que no necesariamente tiene que ser la misma donde está instalado el NIDS y que se dispone de una estructura para virtualizar este servidor.

## *3.4 Instalación Sistemas de visualización de actividades sospechosas de Red SIEM.*

Tal cual definimos en el capítulo 2.5.2 y en su tabla 7, vamos a instalar ELK para poder dejar una herramienta a que el analista pueda visualizar los eventos que pudiera recoger nuestro NIDS.

### *3.4.1 instalación SIEM [ELK]*

Para proceder con esta instalación, vamos a agregar los repositorios necesarios para instalar los paquetes desde ELK. Como guía, vamos a utilizar a Digital Ocean [129]

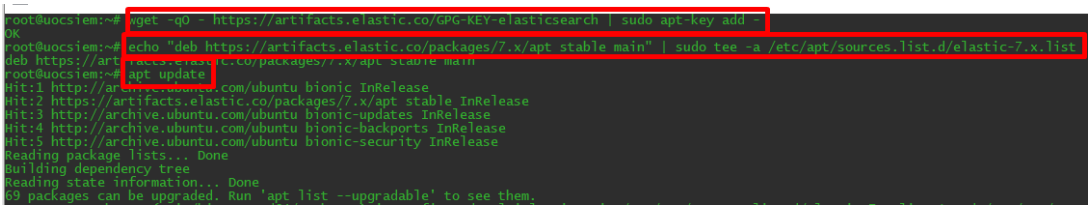


Imagen 84. Repositorios de ELK.

Una vez hemos instalado los repositorios, procederemos a instalar Elastic Search de primero.

128 https://suricata.readthedocs.io/en/latest/rules/intro.html

129 https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-18-04

```
root@siemlocal:~# apt install elasticsearch
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  elasticsearch
0 upgraded, 1 newly installed, 0 to remove and 140 not upgraded.
Need to get 328 MB of archives.
After this operation, 546 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 elasticsearch amd64 7.12.0 [328 MB]
Fetched 328 MB in 1min 27s (3,766 kB/s)
Selecting previously unselected package elasticsearch.
(Reading database ... 67102 files and directories currently installed.)
Preparing to unpack .../elasticsearch_7.12.0_amd64.deb ...
Creating elasticsearch group... OK
Creating elasticsearch user... OK
Unpacking elasticsearch (7.12.0) ...
Setting up elasticsearch (7.12.0) ...
Created elasticsearch keystore in /etc/elasticsearch/elasticsearch.keystore
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for systemd (237-3ubuntu10.42) ...
root@siemlocal:~#
```

Imagen 85. Instalacion Elasticsearch.

Una vez instalado, lo que procedemos es a configurar el servicio antes de ejecutarlo donde primero vamos a modificar el fichero de configuración /etc/elasticsearch/elasticsearch.yml. Este lo hemos modificado de la siguiente manera.

```
cluster.name: UOC-TFM
#
# ------------------------------------ Node ------------------------------------
#
# Use a descriptive name for the node:
#
node.name: uoc-node
#
# Add custom attributes to the node:
#
#node.attr.rack: r1
#
# ----------------------------------- Paths ------------------------------------
#
# Path to directory where to store the data (separate multiple locations by comma):
#
path.data: /var/lib/elasticsearch
#
# Path to log files:
#
path.logs: /var/log/elasticsearch
#
# ----------------------------------- Memory -----------------------------------
```

Imagen 86. Configuracion Elasticsearch [1].

```
# ---------------------------------- Network -----------------------------------
#
# By default Elasticsearch is only accessible on localhost. Set a different
# address here to expose this node on the network:
#
#network.host: 192.168.0.1
network.host: 0.0.0.0
#
# By default Elasticsearch listens for HTTP traffic on the first free port it
# finds starting at 9200. Set a specific HTTP port here:
#
http.port: 9200
#
# For more information, consult the network module documentation.
#
# --------------------------------- Discovery ----------------------------------
#
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
#discovery.seed_hosts: ["host1", "host2"]
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
#cluster.initial_master_nodes: ["node-1", "node-2"]
#
# For more information, consult the discovery and cluster formation module documentation.
#
# ---------------------------------- Various -----------------------------------
#
# Require explicit names when deleting indices:
#
#action.destructive_requires_name: true
#
discovery.type: single-node
```

Imagen 87. Configuracion Elasticsearch [2].

Estos son los parámetros mínimos.
El nombre del Clúster.
El nombre del nodo del Clúster
El Network Host
El Puerto HTTP y
El Tipo de Discovery del Nodo.

Luego de esto, ejecutamos el servicio y lo dejamos fijo para autoinicio.


Imagen 88. Iniciar Elasticsearch

Luego de instalado Elasticsearch, procedemos a instalar Kibana.

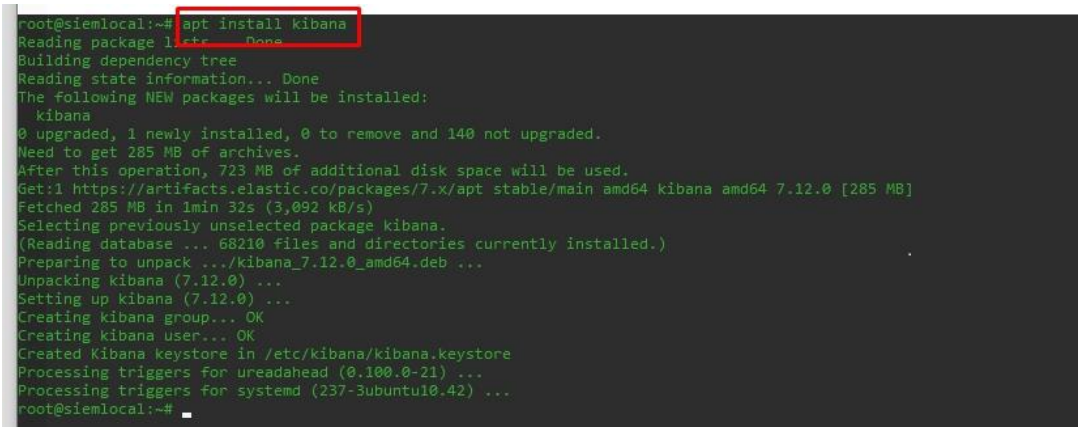



Imagen 89. Instalar Kibana

Kibana necesita un servidor Web para poder funcionar. En este caso vamos a instalar Nginix [130] [131].

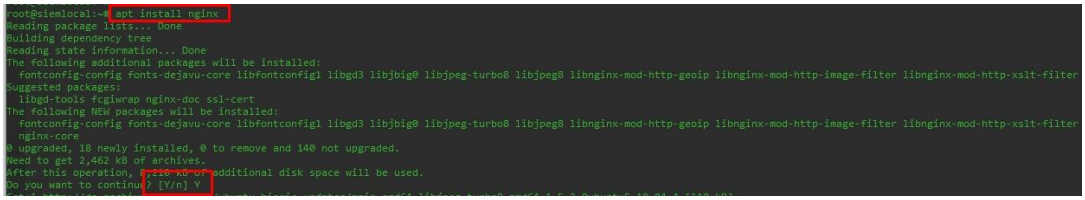



Imagen 90. Instalar Nginix

Luego iniciamos Nginix como servicio.

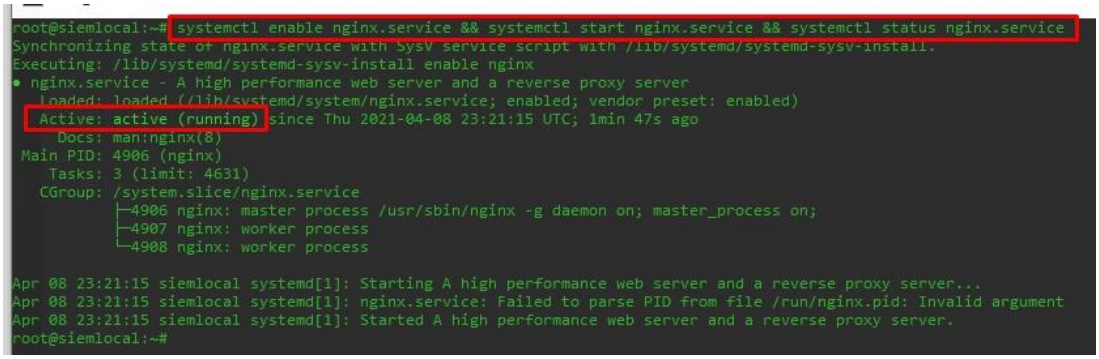



Imagen 91. Iniciar Nginix

130 https://www.nginx.com/

131 https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-18-04

Una vez instalado y ejecutando Nginix, procedemos a iniciar Kibana.

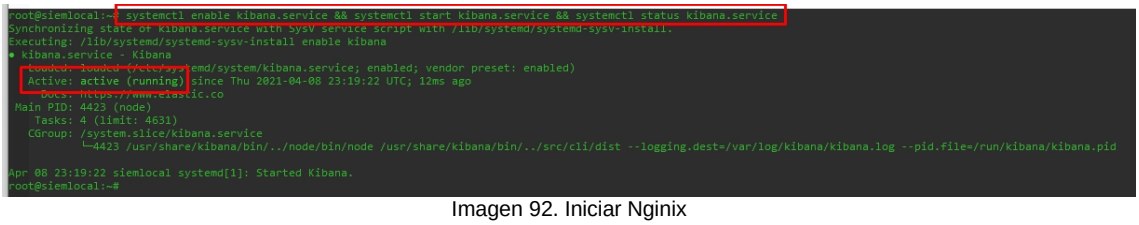



Imagen 92. Iniciar Nginix

Ya de esta manera podemos ir verificando que estas instalaciones están en funcionando de manera correcta.

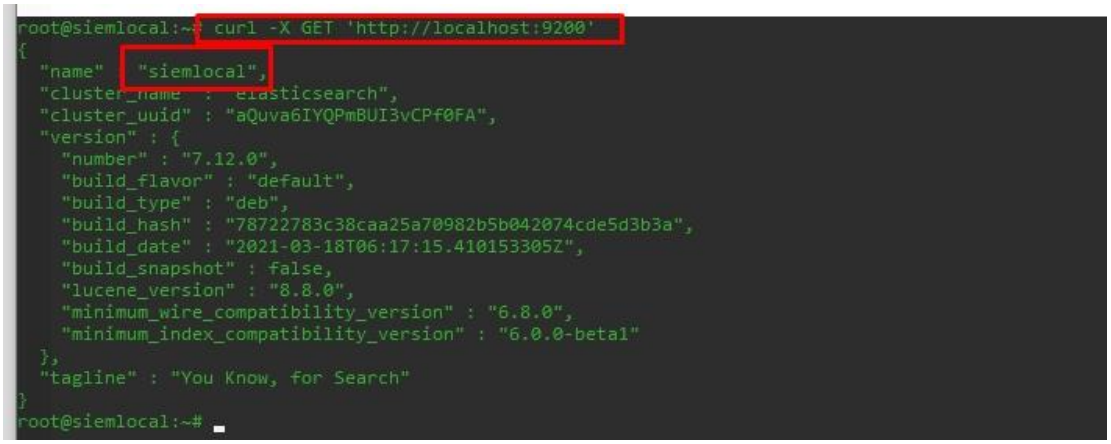



Imagen 93. Verificamos Elasticsearch

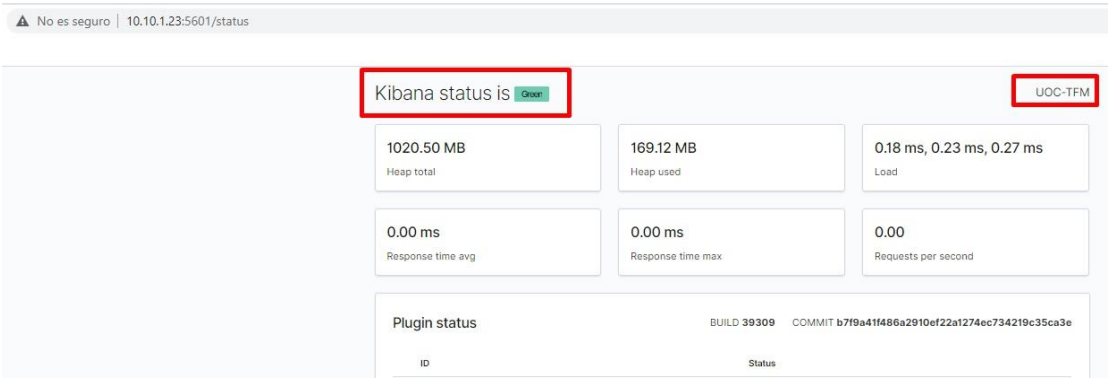



Imagen 94. Verificamos Kibana

Ya de esta manera, tenemos en ejecución una instalación virgen de Elasticsearch y de Kibana y demás componentes.

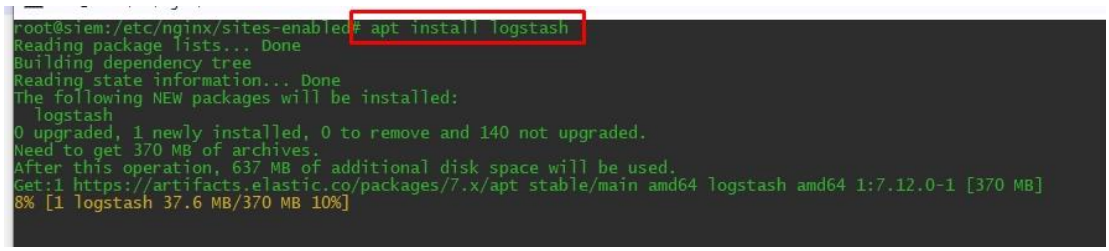



Imagen 95. Instalamos Logstash

Como vamos a trabajar con Suricata, necesitaremos un módulo que comprenda el fichero eve.json donde este NIDS guarda los eventos en un formato estándar.

### *3.4.2 Configuración SIEM [ELK]*

La configuración de ELK se realizó en el momento de la instalación y puesta en marcha. Las imágenes 86 y 87 presentan las configuraciones que hicimos a Elasticsearch. Establecimos el nombre del clúster, el nombre del nodo, la ubicación de las data con que va a trabajar Elasticsearch y donde estará gestionando los logs del sistema. Igual configuramos los parámetros de Red [Abierto a toda la red] y los parámetros del puerto http [9200] que va a conectar a Elasticsearch con la demás parte que integran ELK.

Para el caso de Kibana fue modificada la configuración del fichero /etc/kibana/kibana.yml.

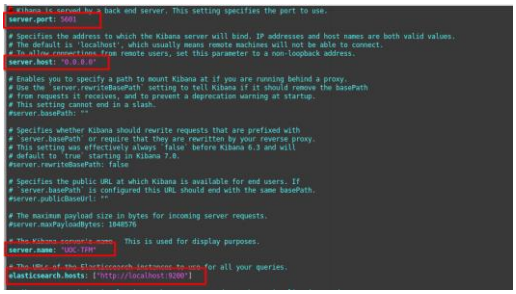



Imagen 96. Consfiguracion Kibana

Se especifico el puerto al que nos vamos a conectar en Kibana, siendo el por defecto 5601, de la misma manera un nombre para fines de visualización y la url de Elasticsearch con su puerto especificado [9200].

Para poder visualizar los logs de Suricata, hemos elegido el módulo Suricata de Filebeat [132] y hemos dejado por defecto su configuración dado que el asume que Suricata está instalado en el mismo equipo. Para esos fines, hemos creado el directorio /var/suricata/ y hemos alojado en este directorio el fichero eve.json donde vamos a depositar los diferentes logs de Suricata siendo este u otro que estemos recibiendo en el mismo SIEM.

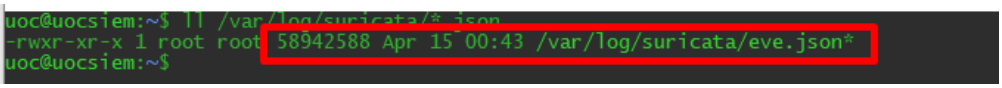



Imagen 97. Localizacion fichero eve.json

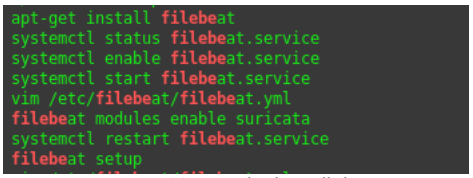



Imagen 98. Instalacion Filebeat

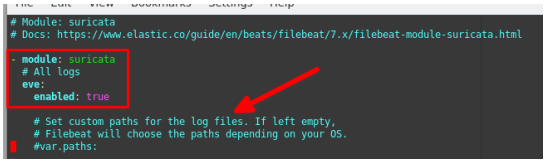



Imagen 99. Instalacion Filebeat

Logstash solo ha sido instalado y dejado tal cual. Ahora verificamos que los Dashboard han sido creados y que están conectados al fichero que estaremos alimentando con los eventos de Suricata.

132 https://www.elastic.co/guide/en/beats/filebeat/6.8/filebeat-module-suricata.html

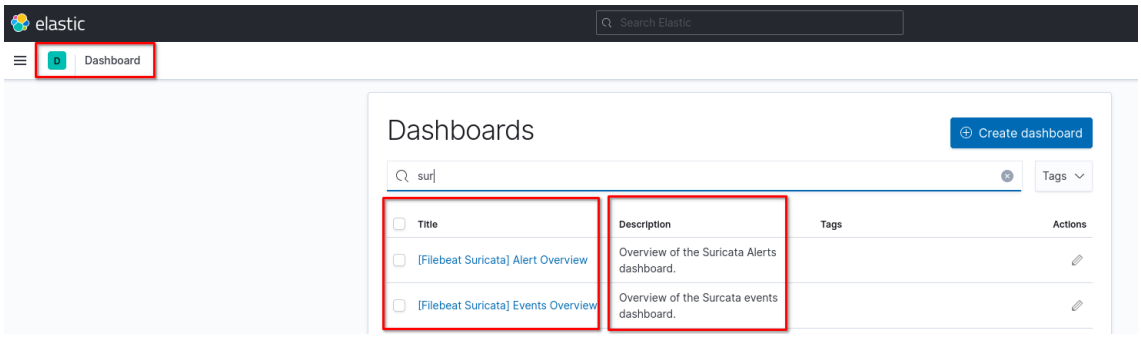



Imagen 100. Dashboard Filebeat-Suricata

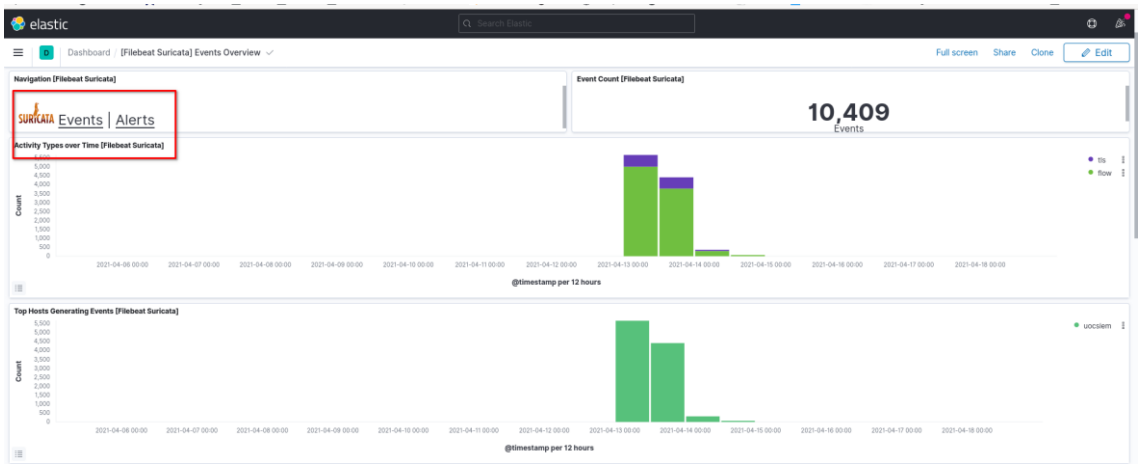



Imagen 101. Dashboard [Events|Alerts] Filebeat-Suricata

En estos Dashboard, Podemos ya visualizar los registros que hemos Engestado desde el NIDS Suricata y podemos tomar y modificar a la necesidad del analista o de la organización. Esto no está sujeto a ningún licenciamiento que obligue a la organización a regirse únicamente por el que trae por defecto, pues, de hecho, se pueden importar de otras fuentes o crearse estos sin problemas.

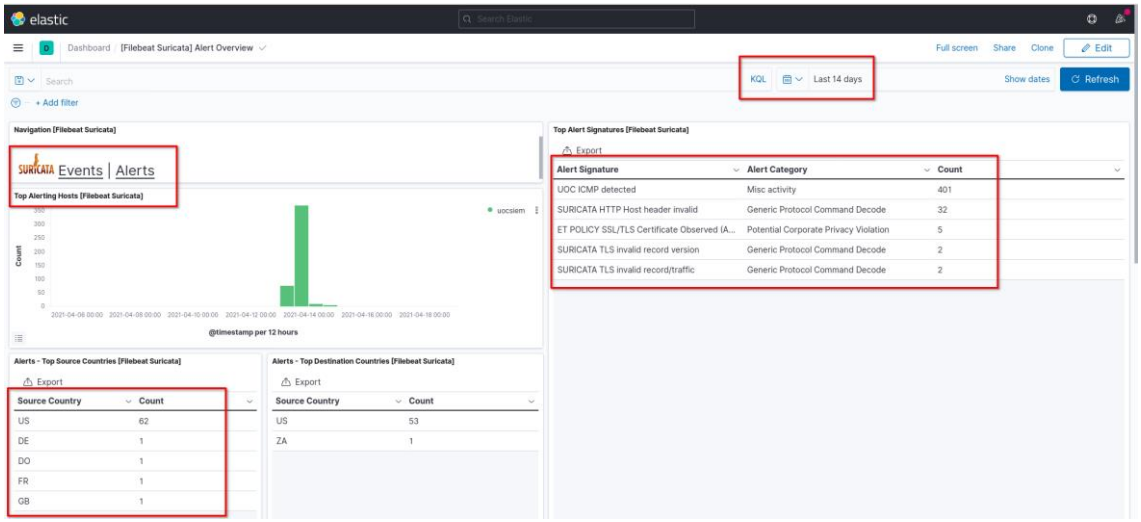



Imagen 102. Dashboard [Events|Alerts] Filebeat-Suricata

En esta imagen 100, Podemos ver ya las Top-Alerts de los datos hasta ahora recogidos. En la parte superior podemos ver los datos que recoge la regla que identificamos en la configuración de prueba de Suricata.

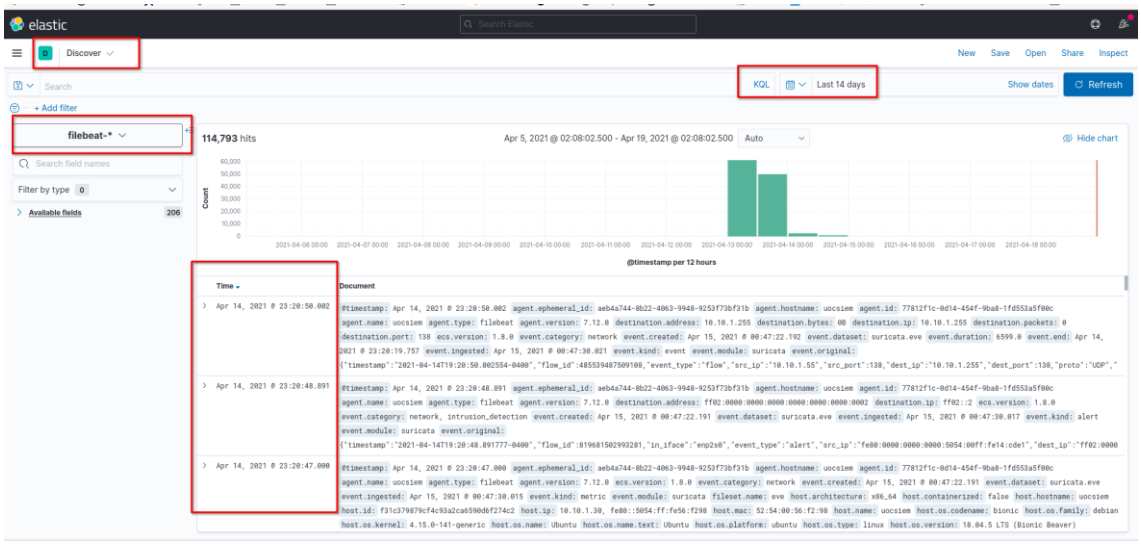



Imagen 103. Discover Filebeat-Suricata.

De esta manera, podemos ver el despliegue de las alertas que se van capturando en el NIDS.

### *3.4.3 Integración con NIDS [ELK-Suricata]*

Para la integración de estas herramientas, dado el hecho que asumimos que no necesariamente están en el mismo local tanto el NIDS como el SIEM, se deben tener en cuenta algunas consideraciones.

De las dos tarjetas de redes que tiene la virtual del NIDS [Ver imagen 51], una de las tarjetas estará operando para comunicarse con el SIEM. Dependiendo de la arquitectura de la red, la distancia que se pueda considerar se deberá ponderar que esta tarjeta tendrá que alcanzar el SIEM. Ya sea por una VPN, por una VLAN administrativa o que el SIEM este en la nube, al que igual se considera acceder vía VPN, estas dos interfaces de red deberán ser alcanzables. En especial los puertos SSH.
Para realizar el envío del eve.json desde el NIDS hasta el SIEM, consideramos enviarlo vía SSH. Como el SIEM considera que Suricata está instalado en el mismo lugar que ELK, vamos a enviar de manera íntegra el fichero que genera Suricata y lo vamos a actualizar en el fichero que hemos preparado en donde tenemos instalado ELK. Lo haremos de la siguiente manera.

```
#!/bin/bash

#echo "Sincronizar NIDS con SIEM"

rsync -avzhe ssh /var/log/suricata/eve.json  uoc@192.168.122.48:/home/uoc/SuriLog
```

Imagen 104. Script Bash NIDS -> SIEM.

Con este script, lo que hacemos es cargar en el SIEM una copia sincronizada del fichero eve.json que genera Suricata. Para mantenerlo lo más actualizado posible, lo hacemos cada 1 minuto con una tarea de que se ejecute sola.

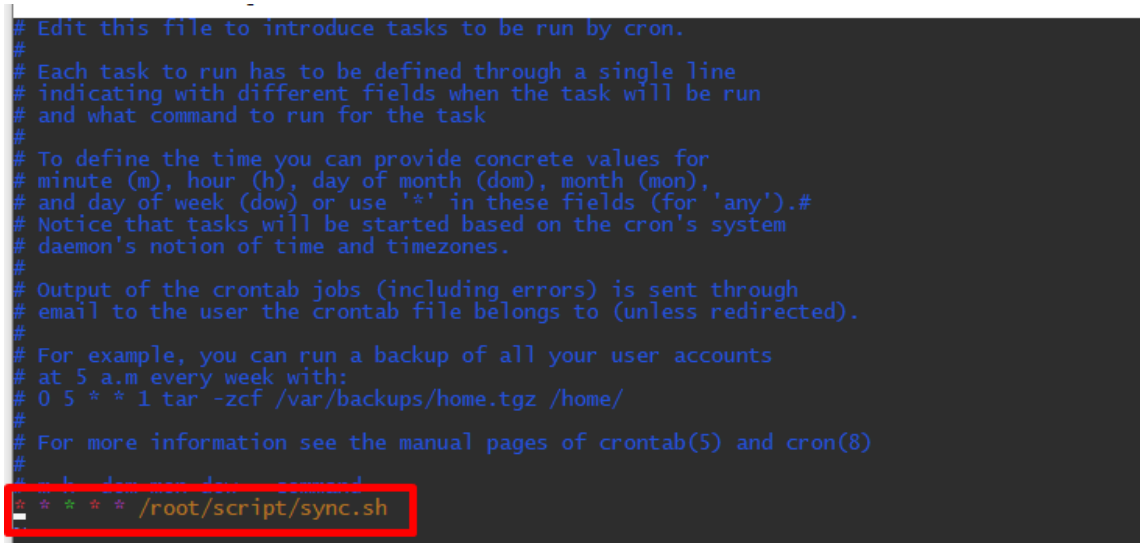



Imagen 105. Crontab.

Una vez ese fichero lo hemos sincronizado en el SIEM, lo que hacemos es otro Script que se encargue de leer este fichero y cargarlo integro al fichero que lee Filebeat. Esto se realiza en una sola vía para evitar que se accidente la data.

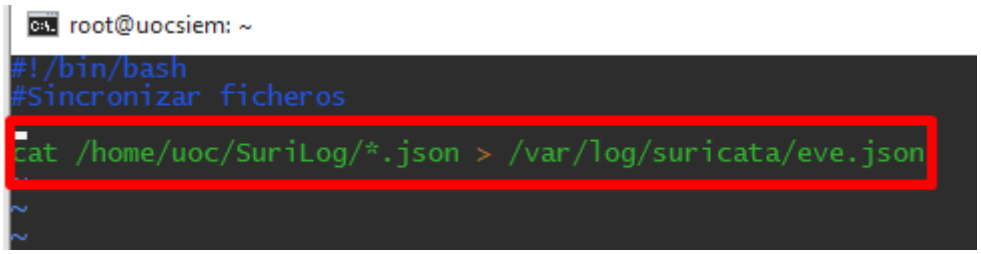



Imagen 106. Script Bash SIEM.

Este Script lo ponemos en una tarea programada cada 5 minutos y listo.

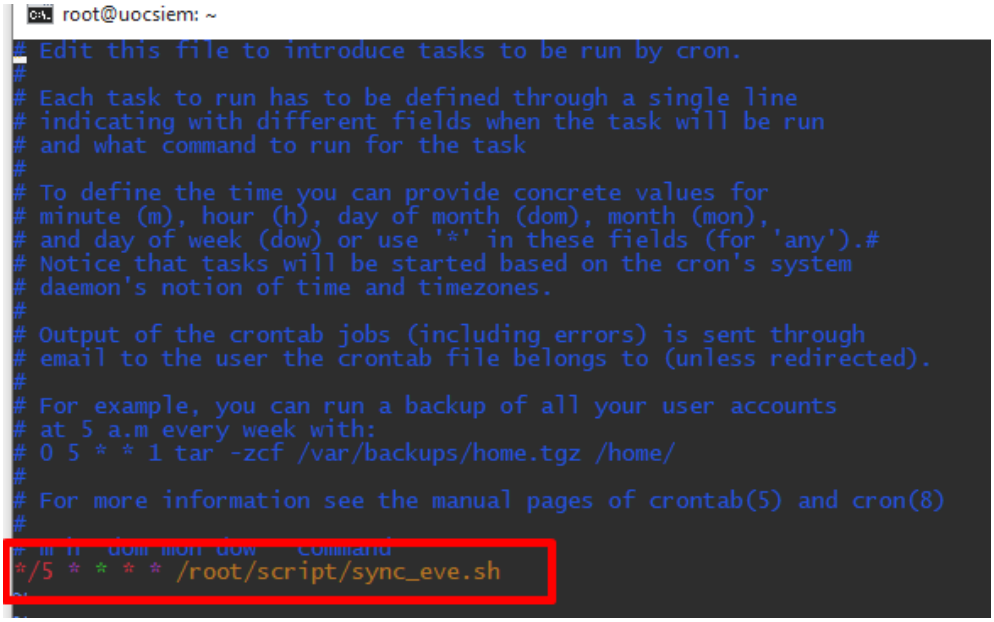



Imagen 107. Crontab SIEM.

De esta manera y como ya hemos configurado en el apartado 3.4.2 Filebeat con el módulo Suricata, cada 15 minutos carga este fichero eve.json que está en /var/log/suricata.

De esta manera no importa lo x-distante que se encuentre el SIEM del NIDS. Solo una de las tarjetas del NIDS deberá tener alcance al SIEM para enviar los datos de manera recurrente. Como estos datos se conservan porque se sincronizan de una sola vía, en caso de fallar la comunicación, este continuaría una vez se establezcan las conexiones nuevamente y el Analista tendrá como continuar con sus labores.

### *3.5 Resumen instalación e integración.*

Hasta el momento, hemos detallado como ha sido la instalación de nuestro sistema para análisis de actividades sospechosas en la red.

Hemos instalado y configurado con éxito una unidad de Suricata. Con la capacidad de ver las tramas que pasan por los puertos que tienen en modo escucha en el Switch dedicado.

Hemos instalado ELK con la capacidad de dar una visión más amigable al analista.

Hemos integrado ambas partes con una comunicación de una sola vía para que el SIEM tenga datos al menos cada 5 minutos en Kibana. Esto podría variar dependiendo del volumen que gestionen y la calidad de la comunicación entre ambos nodos.

Hemos configurado las reglas en el NIDS dejando al analista la posibilidad de agregar nuevas reglas en el fichero /etc/suricata/rules/uoc.rules, bajo la consideración que se tiene que respetar la nomenclatura exigida por Suricata. Ambos sistemas se comunican por el protocolo SSH donde intercambiar configuraciones.

### *3.6 Revisión de la implementación.*

Una vez hemos culminado con la integración, vamos a verificar que las configuraciones están correctas y que hemos podido desplegar una herramienta con la finalidad de que un analista de monitoreo pueda monitorear las actividades de una red y en especial, aquellas que considere sospechosas.

Para verificar que Suricata está funcionando, lo podemos hacer de varias maneras siendo las más sencilla verificar los logs del sistema.

Si verificamos los logs de Suricata, en especial /var/log/suricata/eve/json, esto es lo que tenemos.

Imagen 108. Log de Suricata.

Como se puede apreciar, esteevento queda registrado y es detectado por el NIDS en base a la regla que hemos creado.

Una vez confirmado que está generando eventos y que estos se están guardando en este fichero, verificamos que este fichero se esté enviando al SIEM via SSH en una tarea programada cada 1 minuto. Esta tarea solo envía las ultimas actualizaciones o más bien, los últimos eventos generados desde la última vez que fue enviado.

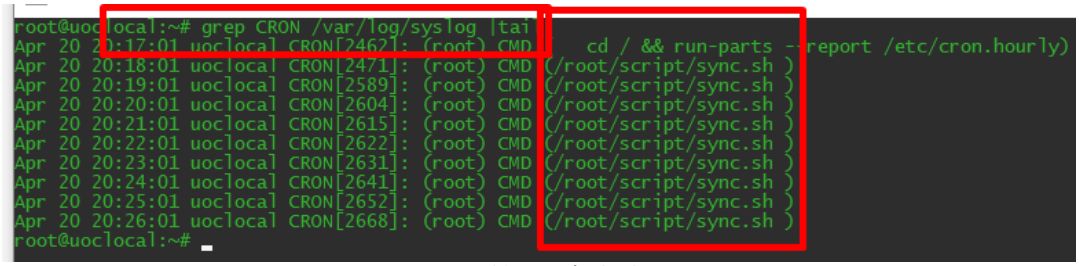


Imagen 109. Log de CROND.

Como ya estamos enviando de manera recurrente el contenido de este fichero, ahora verificamos en el SIEM que estos eventos se están recibiendo.

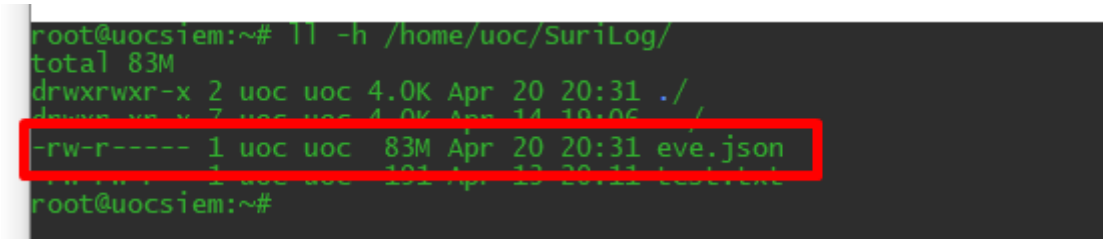


Imagen 110. Fichero eve.json recibido.

Una vez en la estación remota que tiene instalado nuestro SIEM ELK, verificamos que este fichero está siendo actualizado.

Como este SIEM podría recibir de varias fuentes, dejando la posibilidad de que sea utilizado en un SOC, lo que hacemos es recibirlo en este directorio y se crearía uno nuevo para otro NIDS que tengamos desplegado.

Una vez tenemos este y la posibilidad de otros ficheros, lo que hacemos es correr otro script [imagen 106] que mantenga actualizado el fichero maestro de /var/log/suricata/eve.log que es idéntico al que tiene el NIDS, solo que de manera local.

Una vez este Script se ha ejecutado, verificamos el fichero maestro /var/log/suricata/eve.log de nuestro SIEM y vemos que está siendo actualizado.

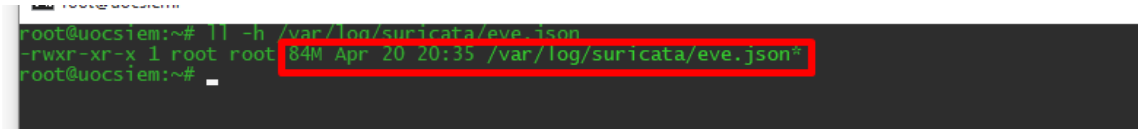


Imagen 111. Fichero eve.json actualizado.

Ya de esta manera, tenemos nuestro fichero listo para ser presentado en ELK

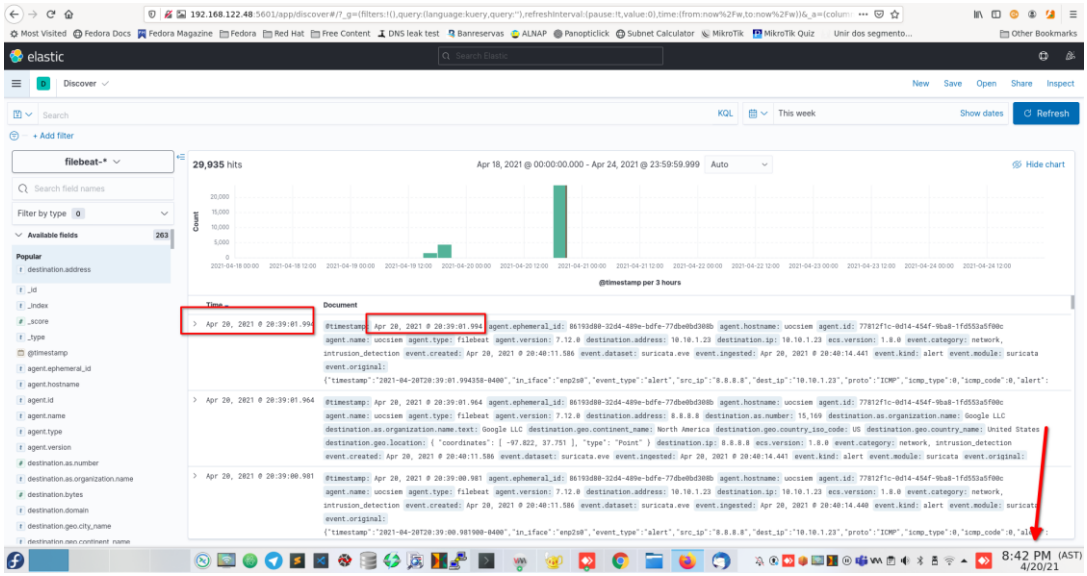



Imagen 112. Kibana mostrando datos de Suricata.

Como se pudo evidenciar en el punto 3.4.2, configuramos nuestro SIEM para que reciba estos datos mediante el módulo *Suricata* que nos brinda esta herramienta. Esto permite que con los índices de Filebeat creados por defecto, podamos ya poder visualizar estos eventos casi de inmediato. La tasa de retardo que presentamos actualmente es de 5 minutos como máximo ya que es el tiempo que tarda en sincronizar los ficheros que recibimos de los NIDS con el fichero maestro.

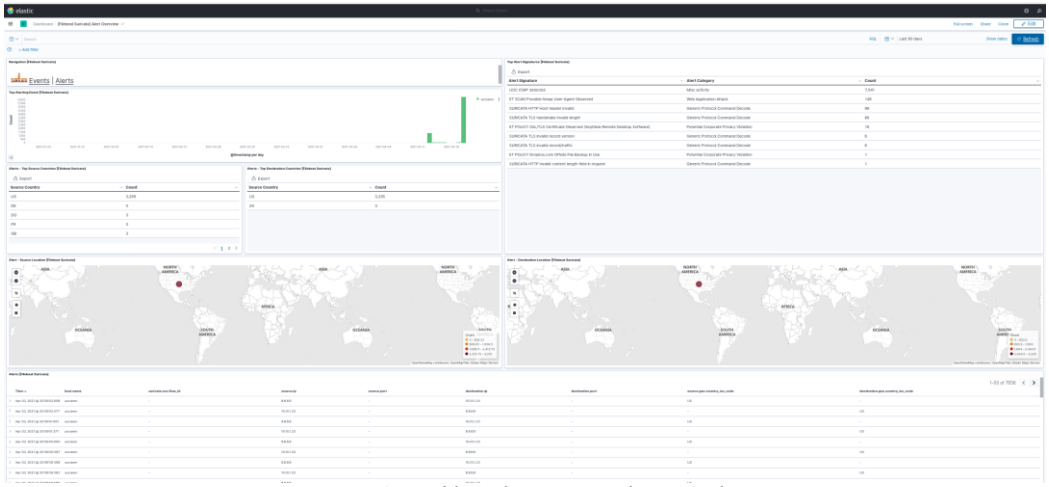


Imagen 113. Dashboard Eventos y Alertas Suricata.

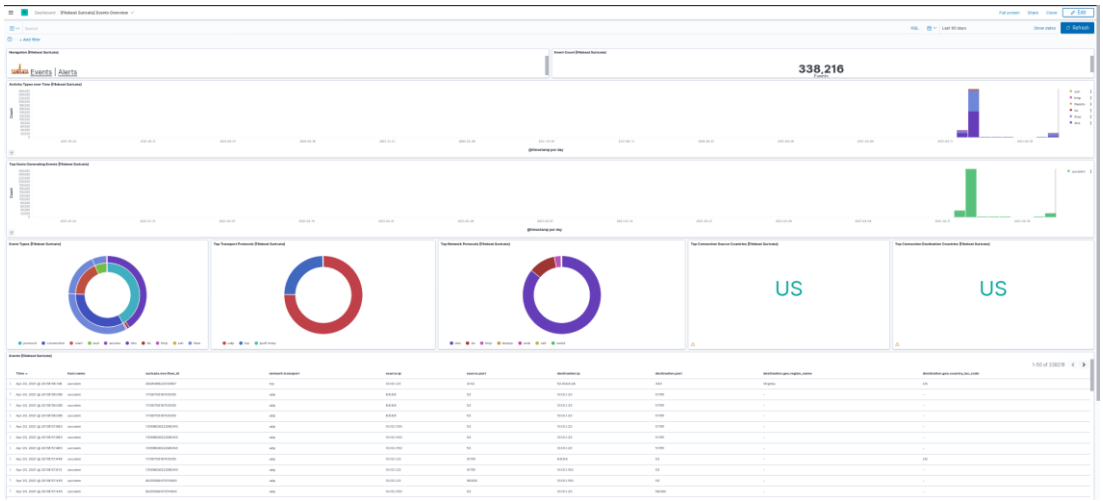



Imagen 114. Dashboard [2] Eventos y Alertas Suricata.

Como es un SIEM que hemos elegido por el tipo de licenciamiento, una versión de paga podría implementar algunas otras funcionalidades propias de un SIEM en un SOC.

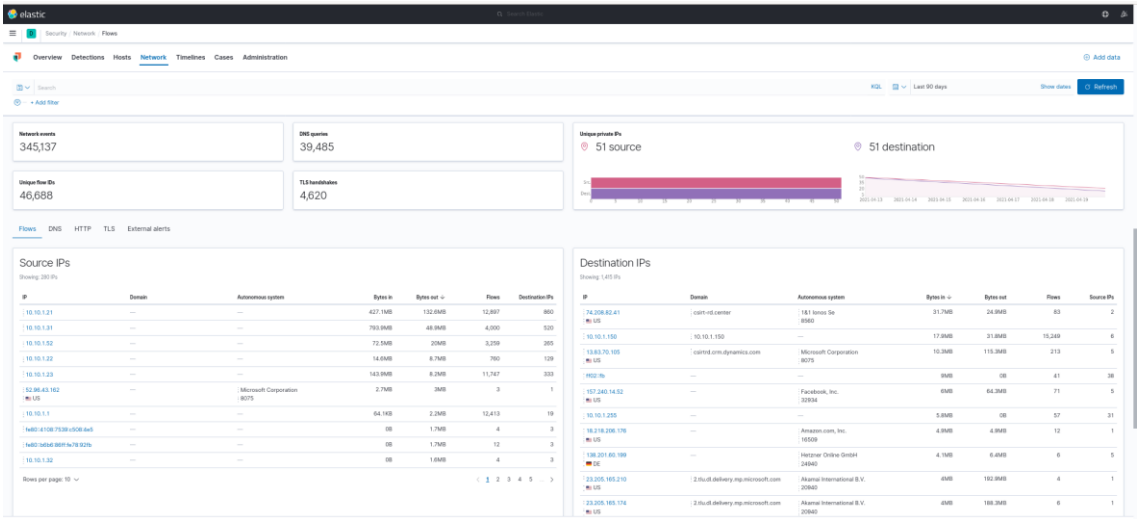



Imagen 115. Dashboard Security SIEM.

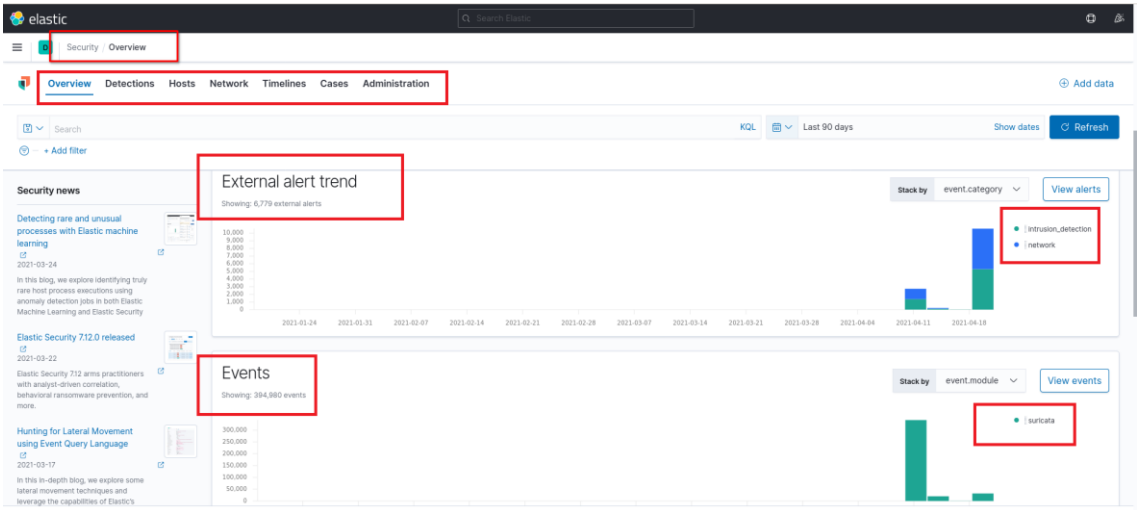



Imagen 116. Dashboard Overview SIEM.

De esta manera podemos verificar que todo funciona tal cual ha sido planteado.

# 4. Conclusiones Finales

## *4.1 Conclusiones finales sobre la propuesta inicial.*

Llegado a este punto, podemos ya tener una conclusión sobre nuestra propuesta inicial donde si recapitulamos los requerimientos, nos damos con los siguientes:

### *4.1.1 Sobre la capacidad de análisis de actividades sospechosas en la Red.*

Desde el inicio de este trabajo, centramos nuestro objetivo en la necesidad de aquellas herramientas automatizadas que nos permitan ver de manera constante todo lo que sucede en una red de datos. Tener esta capacidad operativa la consideramos de vital importancia ya que nuestra propuesta era la de desplegar esas herramientas en una organización y en especial aquellas con licenciamiento de Código Libre para evitar incurrir en elevados presupuestos.

Hemos concluidos que las herramientas que hemos elegido, NIDS Suricata ® para la captura de las tramas y posterior envío a un SIEM para su análisis ha sido de resultados satisfactorios dado que nos brinda la libertad de poder integrar reglas ya sea creadas por el analista o quien le estaría dando mantenimiento a las reglas o continuar con aquellas que se pueden descargar de manera libre de costos.

Para esta implementación, encontramos cero obstáculos dado que elegimos un Sistema Operativo, UBUNTU ® 18.04, que cuenta en sus repositorios los paquetes que se necesitan para esta implementación de este NIDS.

Sobre los gastos incurridos para esta parte, en líneas futuras es de cero costes para la organización. Estos sistemas se pueden descargar de manera libre desde sus portales como describimos en la investigación e implementación.

### *4.1.2 Sobre la capacidad de visualizar actividades sospechosas en la Red*

Consideramos que, para este punto, la herramienta que hemos elegido, ELK ®, cumple con los requisitos propuesto inicialmente. Su licenciamiento nos permite realizar el despliegue sin costos adicionales o que los tenga a posterior que cubrir la organización que desea implementar nuestra propuesta. Esta herramienta SIEM tiene la capacidad operativa de integrarse con el NIDS Suricata ® que hemos elegido para este despliegue. De hecho, cuenta con unos módulos integrados que se instalaron en el punto 3.4.2, lo que nos facilitó bastante este proceso de integración NIDS-SIEM dado la madurez con que este módulo viene ya.

Para esta implementación, encontramos cero obstáculos dado que elegimos un Sistema Operativo, UBUNTU ® 18.04, que cuenta en sus repositorios los paquetes que se necesitan para esta implementación de este SIEM.

Sobre los gastos incurridos para esta parte, en líneas futuras es de cero costes para la organización. Estos sistemas se pueden descargar de manera libre desde sus portales como describimos en la investigación e implementación.

Tal cual como fue propuesto en el punto 1.2 de este TFM, concluimos que su despliegue se realizó sin inconvenientes.

Concluimos en base a las pruebas realizadas y documentadas que esta propuesta inicial cuenta con la capacidad de visualizar las actividades sospechosas de una red capturadas por el NIDS propuesto en este TFM.

*4.1.3 Sobre la Metodología propuesta.*

Nos propusimos que toda implementación seria sobre tecnologías de licenciamiento de Código Libre, propuesta que se mantuvo hasta el final. Nos propusimos acogernos a aquellas documentaciones oficiales y a aquellas que nos aporten informaciones relevantes a la hora de mantener una investigación con una higiene depurable. En este punto, los obstáculos fueron muy mininos dado que nuestra fuente principal de búsqueda estaba en la Internet, en especial en la documentación que ponían los fabricantes en sus portales y en documentos de carácter generales que nos daban informaciones pertinentes y adecuadas al objetivo de este TFM. Concluimos que, cada segundo la internet como fuente de documentación crece poniendo a nuestra disponibilidad fuentes de informaciones cada vez más adaptada y actualizada.

Estas conclusiones dejan un camino abierto a aquellas organizaciones que quieran en una línea futura implementar esta solución y tengan la documentación a disponible.

*4.1.4 Sobre la planificación de trabajo y los recursos iniciales.*

Nuestra planificación tal como fue presentada en el inicio de este TFM, se respetó en todo momento dado que no surgieron contratiempos que pudieran alterar el curso de lo propuesto.

Concluimos que esta implementación se podría realizar en una organización en un periodo comprendido de la siguiente manera.

- ✓ Análisis de la infraestructura donde se implementaría, un periodo de una semana.
- ✓ Tiempo necesario para planificar el despliegue, una semana.
- ✓ Tiempo de despliegue total, una semana.
- ✓ Tiempo de pruebas e integraciones finales, un mes. Tomando en cuenta que se estarían acondicionando las reglas descargadas y aquellas que el analista estaría necesitando para sus labores de análisis de actividades sospechosas en una red.

*4.1.5 Sobre la factibilidad económica.*

Durante toda la investigación e implementación, hemos sostenido la sostenibilidad económica para las organizaciones al momento de optar por esta implementación.

Hasta el momento, hemos sostenido que esta implementación no supone un costo de licenciamiento y que se encuentre con toda libertad de poder desplegar esta solución.

El único coste adicional que podría incurrir la organización estaría en la nómina de empleados dado que, si no existe un analista con la experiencia, tendría que verse en la necesidad de contratar un personal nuevo con las capacidades de:

- ✓ Experiencia en Monitoreo de Seguridad.
- ✓ Experiencia en NIDS.
- ✓ Experiencia en Infraestructuras y Redes.
- ✓ Experiencia en administración de sistemas GNU/Linux..

## *4.2 Prueba de Concepto.*

Para validar nuestras conclusiones, evidenciamos como prueba de concepto que la herramienta que hemos propuesto para el análisis de actividades sospechosas puede ser de utilidad para una organización.

Iniciamos ya en lo que presentamos en la instalación e integración donde pudimos validar una regla que detectaba tráfico **ICMP**, y que estos los podía más adelante visualizar en el SIEM que hemos desplegado. Luego a esa integración, validamos algunas otras reglas que mostramos a continuación.

Imagen 117. Reglas para PoC.

En estas reglas, podemos identificar tráfico ICMP, Conexiones SSH, Conexiones RDP, y escaneos con herramientas como NMAP.

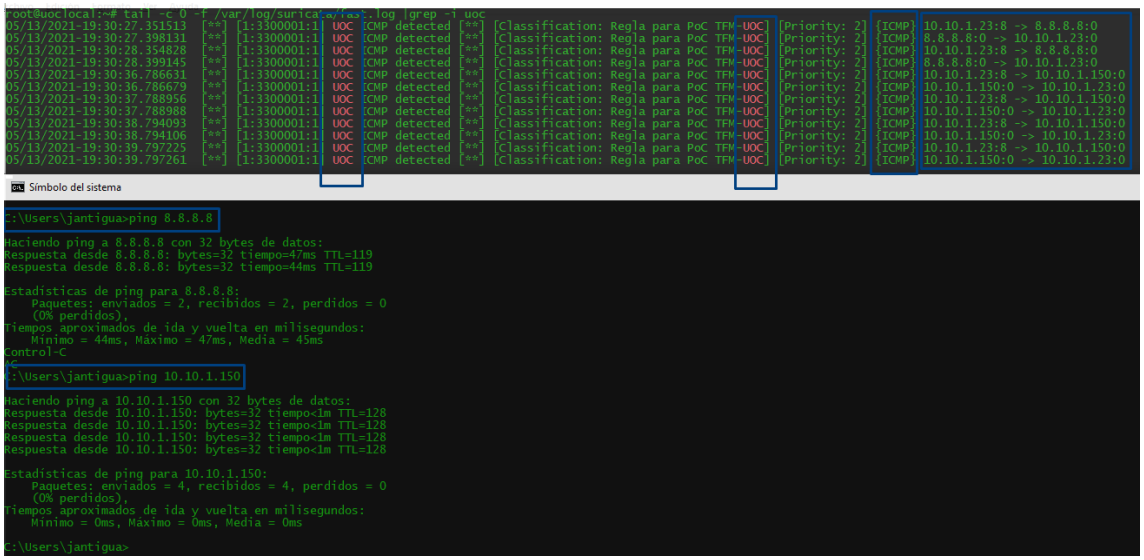



Imagen 118. Captura ICMP.

Esta es una regla sencilla, pero nos da una idea de como seria la nomenclatura de una regla de acuerdo a como lo especifica la documentacion oficial [133]

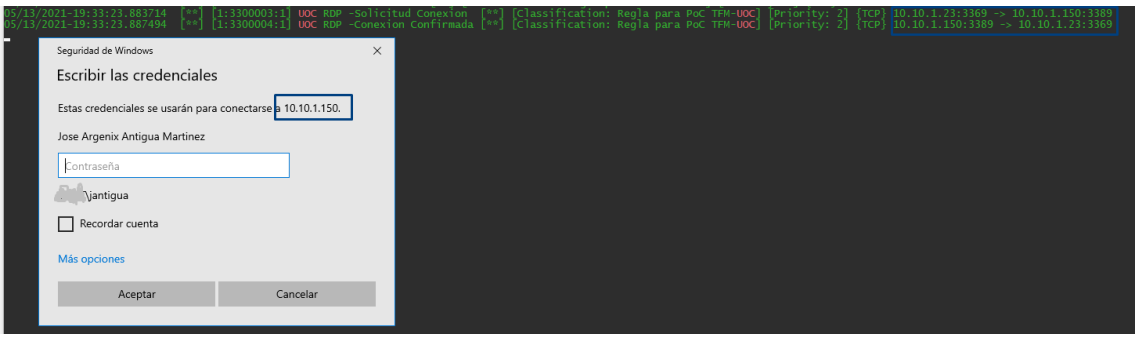



Imagen 119. Captura RDP.

133 https://suricata.readthedocs.io/en/suricata-6.0.0/rules/index.html

Esta regla que presentamos en la imagen 119, podemos evidenciar la posibilidad de poder realizar capturas a conexiones RDP.

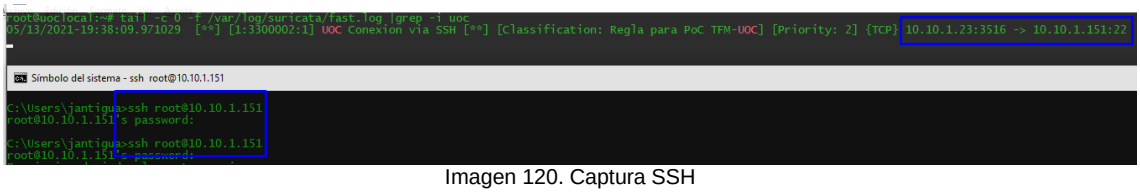



Imagen 120. Captura SSH

En la imagen anterior, evidenciamos la posibilidad de capturar tráfico destinados a conexiones SSH. En esta ocasión necesitaríamos como en otros servicios que queremos estar monitoreando tener en cuenta los puertos TCP/UDP que han sido configurados para no obviar estos posibles incidentes.

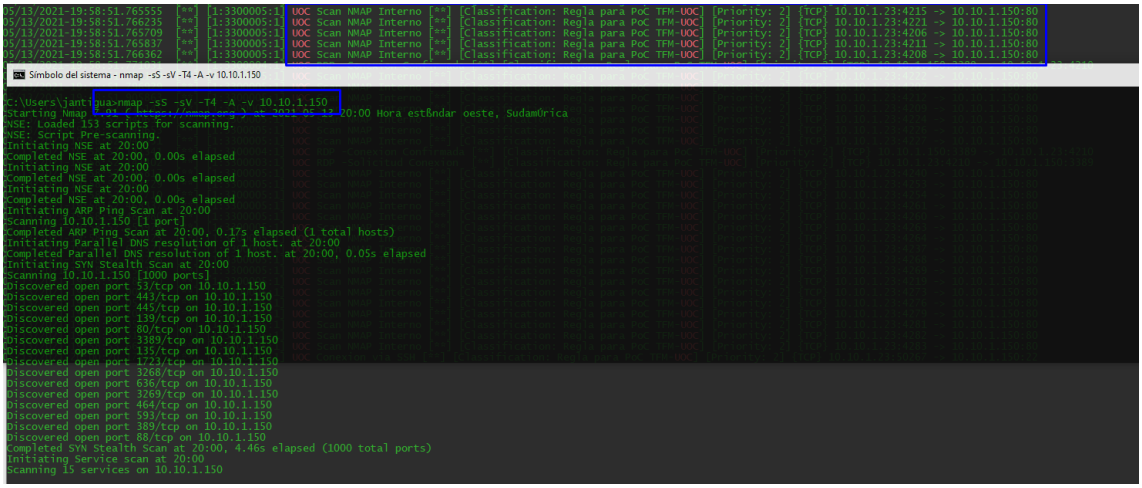



Imagen 121. Captura ejecucion NAMP [A lo interno].

Como se puede apreciar en la imagen 121, un analista puede configurar una regla con un poco más de complejidad como lo es detectar escaneos de red, una actividad muy inusual y sospechosa más allá de que la esté realizando alguien de manera legitima.

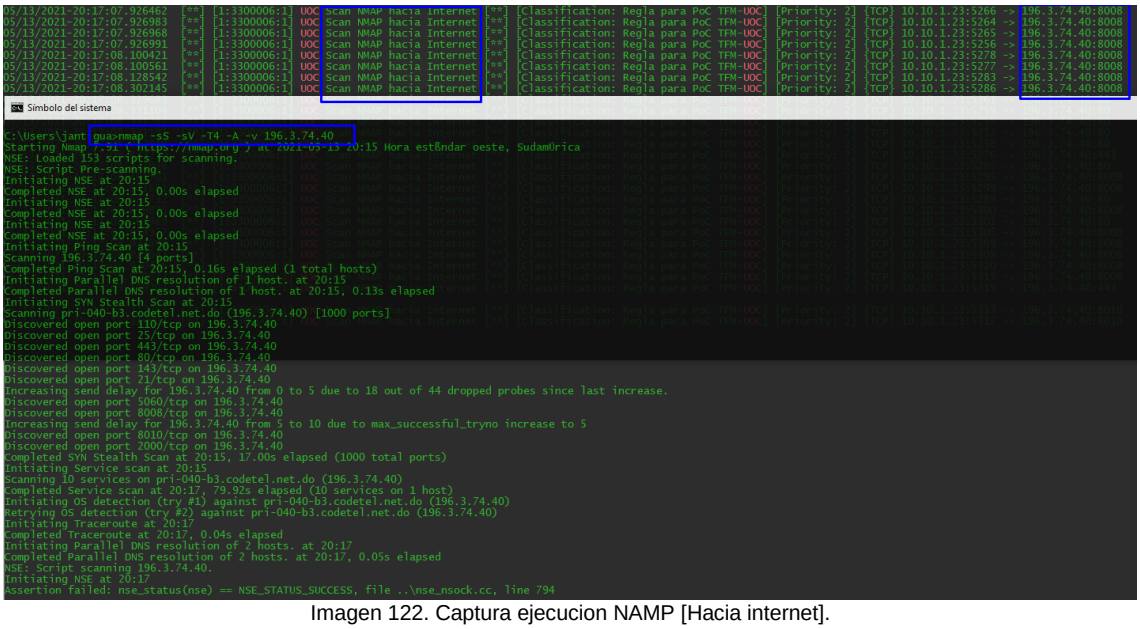



Imagen 122. Captura ejecucion NAMP [Hacia internet].

Con esta imagen 122, verificamos que podemos diferenciar por el destino si estamos realizando un escaneo a lo interno o hacia la internet. En este caso, capturamos el tráfico hacia la internet.

Cabe resaltar el hecho que cada regla dependerá de que considere la organización y su equipo de monitoreo como actividad sospechosa para ponerla bajo vigilancia y que estas reglas aún se pueden afinar más para un escaneo más profundo inclusive.

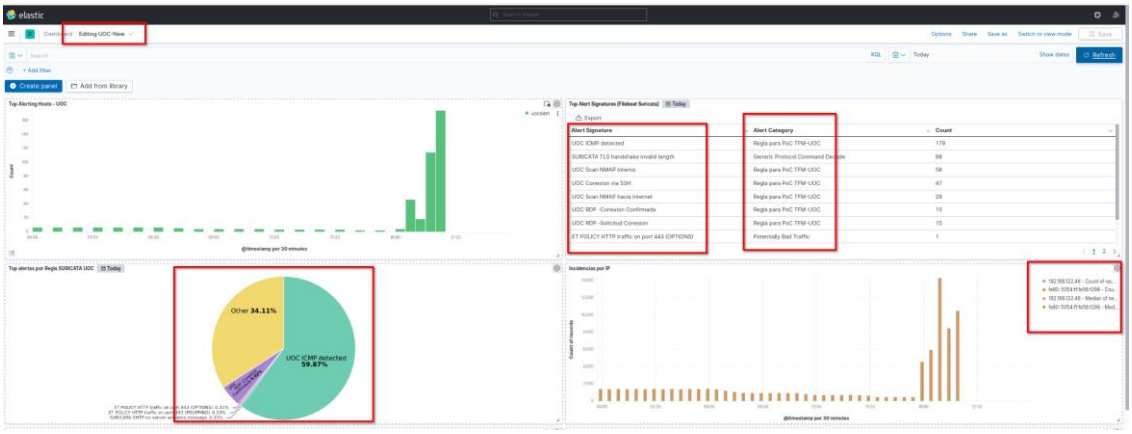



Imagen 123. Dashboard Personalziado.

Ya en este dashboard personalizado que presentamos en esta imagen 123, podemos ver o tener una idea de cómo el analista estaría visualizando las actividades que se puedan recoger en el NIDS caso aparte a aquellos dashboards que vienen integrados en la plataforma por defecto.

Otra prueba de concepto que queremos mostrar es la posibilidad de mantener una constante verificación de tráfico sospechoso a actividades como la BotNet donde si cargamos una lista de reputación con mantenimiento diario, el analista tendrá la tarea de verificar estas reglas y agregarlas a Suricata ® aunque se podría realizar de forma automática. Esta lista de reputación la podemos descargar ya como una lista de regla desde https://sslbl.abuse.ch/blacklist/sslipblacklist_aggressive.rules y modificamos algunas cosas que explico a continuación.

Como este proyecto es con posibilidad abierta de poder instalar varios NIDS, es muy importante que las reglas en el mensaje “msg:”, **contenga un identificador del área que estamos monitoreando**. En nuestro caso, hemos modificado como se ve a continuación para identificar la regla en el Dashboard.

Imagen 124. Reglas para BotNet.

Una vez tenemos esta regla lista, la agregamos en el ‘*/etc/suricata/suricata.yml*’ en el área de las reglas. Se puede evidenciar que, en el fichero de reglas, hemos agregado una de manera manual en entorno controlado para no tener que comprometer nuestra conectividad y salir en una lista negra nuestra IP publica, lo que hemos hecho es utilizar un portal de prueba y un puerto que hemos configurado manual para otros propósitos y lo hemos declarado una IP sospechosa junto al puerto.

Procedemos a conectarnos vía web a la ip y al portal y desde que establecemos conexiones o descargamos algo de este nos debe arrojar una alerta.

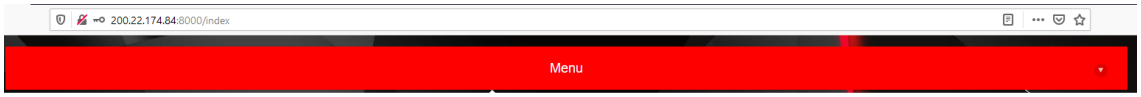



Imagen 125. Conexión a portal sospechoso.

De inmediato Suricata ® nos detecta la conexión y registra e levento.

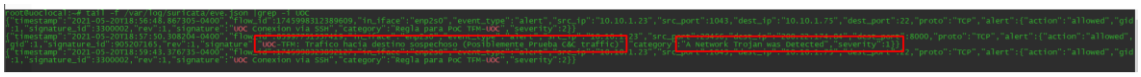


Imagen 126. Registro a conexión posible botNet.

Luego que se actualizan los ficheros eve.json, el analista entonces tiene una visualizacion inmediata e identificada para el caso que nos compete.

Inclusive si descargamos algo desde esta ip, la alerta se registra.

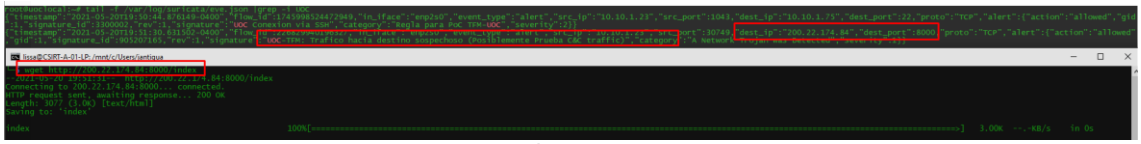


Imagen 127. Registro a conexión con descarga desde posible botNet.

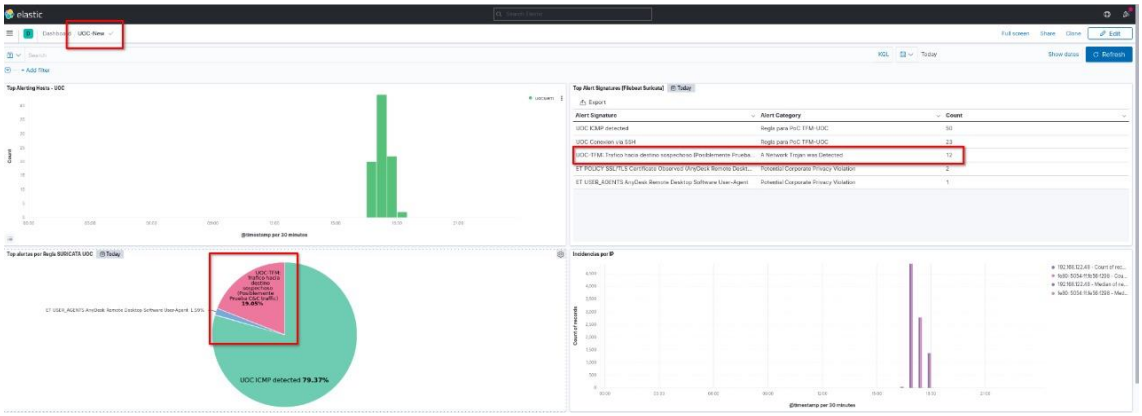



Imagen 128. Dashboard mostrando conexion botNet.

Con estas pruebas de concepto, validamos el funcionamiento de nuestra propuesta y su funcionamiento al igual que la capacidad de agregar más detalles o afinar las reglas ya existentes.

### *4.3 Líneas de trabajo a futuro.*

Dado el hecho que estas son herramientas que integramos, aunque su origen es de diferente fabricante, se tomaría un poco más de tiempo el poder explorar otros proyectos que se pudieran integrar y dar aún más valor agregado a todo este despliegue, podríamos acentuar a continuación las que consideramos más relevantes.

- ✓ Capacidad de integrar el SIEM con un servidor de correos para que las alertas se desplieguen de acuerdo con la criticidad que se le asigne vía correo electrónico.

- ✓ Poder integrarlo con otro proyecto donde el objetivo **no** sea únicamente la red, más bien los servicios críticos y para esto, se utilice un HIDS.

- ✓ Capacidad de gestionar otras listas de reputación con capacidad de alertas dado que las posibilidades inmediatas resultan un poco tortuosas porque su complejidad aumenta en cada integración y de acuerdo con las necesidades de la organización. Esto estaría mirando más allá de anomalías en una red dado que el alcance definido estaría sujeto de forma inicial a lo interno de una organización. Para este lineamiento pendiente, proponemos ver más allá cuando el Indicador de Compromiso es externo. Para nuestro caso en particular, este módulo que gestiona las listas de reputaciones no es del todo compatible con la versión actual de SURICATA ® que hemos desplegado. Quedaría pendiente para cuando este impase este solucionado por parte del fabricante.

- ✓ Estudiar la capacidad de que el analista tenga como configurar reglas en base a comportamientos que pudieran identificar que no necesariamente estén en las reglas que hemos integrado. Bien podría ser en base a firmas, a otros patrones como las descargas, movimientos laterales sospechosos, ejecución de script, Etc.

- ✓ Definir las políticas de los mensajes de las reglas para cuando se tenga más de un NIDS, el analista tenga la capacidad de identificar a que segmento estarían vinculadas las alertas. Se ha recomendado en la PoC que al momento de configurar el “msg” en las reglas, identificar cada segmento.

- ✓ Explorar la posibilidad integrar a un servicio de tickets y de enviar alertas de manera automática a estos sistemas para mantener organizada la acción de respuestas e investigaciones de incidentes.

# 5. Glosario

**Analista [De Ciberseguridad]:** Es un experto en la seguridad informática de las empresas y organizaciones. Se encarga de la formulación de planes para salvaguardar archivos informáticos, Análisis y detección de amenazas de seguridad y desarrollo de técnicas de prevención.

**Análisis:** El análisis es el proceso de dividir un tema complejo o sustancia en partes más pequeñas para obtener una mejor comprensión de él.

**API:** [*Del inglés, Application Programming Interface*] Interfaz de Programación de Aplicaciones, es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción

**Aplicaciones:** Programa informático diseñado como una herramienta para realizar operaciones o funciones específicas.

**Arquitecturas [I386, X86_64]:** Es el diseño conceptual y la estructura operacional fundamental de un sistema de computadoras. Es decir, es un modelo y una descripción funcional de los requerimientos y las implementaciones de diseño para varias partes de una computadora, con especial interés en la forma en que la unidad central de proceso (CPU) trabaja internamente y accede a las direcciones de memoria.

**Ataque De Denegación De Servicios [DDoS]:** (*Del inglés, Denial of Service*), es un ataque a un sistema de computadoras o red que causa que un servicio o recurso sea inaccesible a los usuarios legítimos.

**Ataques A la Seguridad:** [Ciberataques] Son un mecanismo por el que un agente de amenazas explota una vulnerabilidad para causar un impacto de seguridad informática que afecte a la confidencialidad, integridad o disponibilidad de un sistema de información

**Botnet:** Es un término que hace referencia a un conjunto o red de robots informáticos o bots, que se ejecutan de manera autónoma y automática.

**Brechas:** Es un incidente que permite el acceso no autorizado a datos informáticos, aplicaciones, redes o dispositivos. Es decir, permite acceder sin autorización a información.

**Bridge:** [*Traducción de Puente del Inglés*]. IEEE 802.1D Es el dispositivo de interconexión de redes de computadoras que opera en la capa 2 (nivel de enlace de datos) del modelo OSI. Interconecta segmentos de red (o divide una red en segmentos) haciendo la transferencia de datos de una red hacia otra con base en la dirección física de destino de cada paquete.

**Buffer:** Memoria de almacenamiento temporal de información que permite transferir los datos entre unidades funcionales con características de transferencia diferentes.

**Cargadores** [*De Aplicaciones*] Es la parte del núcleo del sistema operativo cuya función es cargar programas en memoria desde los ejecutables.

**Centro De Operaciones De Ciberseguridad Y Monitoreo. [SOC]**: Es un centro de seguridad informática que previene, monitorea y controla la seguridad en las redes y en Internet.

**CERT:** [*Del inglés* ***Computer Emergency Response Team***] Equipo de Respuesta ante Emergencias Informáticas es un centro de respuesta para incidentes de seguridad en tecnologías de la información.

**CIA:** Estas tres letras significan confidencialidad, integridad y accesibilidad, también conocidas como la triada de la CIA o triada de la Seguridad de La Información.

**Ciberataque:** Es un conjunto de acciones ofensivas contra sistemas de información como bases de datos, redes computacionales, etc. hechas para dañar, alterar o destruir instituciones, personas o empresas.

**Ciberatacante:** Individuo que realiza un ciberataque o una acción en contra de un sistema informático.

**Ciberseguridad:** Es el área relacionada con la informática y la telemática que se enfoca en la protección de la infraestructura computacional y todo lo vinculado con la misma, y especialmente la información contenida en una computadora o circulante a través de las redes de computadoras.

**Cifrados:** Que está escrito con letras, símbolos o números que solo pueden comprenderse si se dispone de la clave necesaria para descifrarlos.
Clonar: Hacer una copia exacta de un sistema físico o lógico.

**Clúster**: Sistemas distribuidos de granjas de computadoras unidos entre sí normalmente por una red de alta velocidad y que se comportan como si fuesen un único servidor.

**Colisiones**: Ocurre cuando dos o más dispositivos intentan transmitir datos a través de una red al mismo tiempo.

**Collector**: Es un mecanismo implícito de gestión de memoria implementado en algunos lenguajes de programación de tipo interpretado o semi interpretado.

**Compilador**: Es un tipo de traductor que transforma un programa entero de un lenguaje de programación a otro. Usualmente el lenguaje objetivo es código máquina, aunque también puede ser traducido a un código intermedio o a texto.

**Confidencialidad**: Es la propiedad de la información, por la que se garantiza que está accesible únicamente a personal autorizado a acceder a dicha información.

**Conmutar**: Se considera como la acción de establecer una vía, un camino, de extremo a extremo entre dos puntos, un emisor (Tx) y un receptor (Rx) a través de nodos o equipos de transmisión.

**Consola**: Es un método, interfaz o dispositivo que permite a las personas dar instrucciones a algún programa informático

**Consultor** [*De Ciberseguridad*] Es un profesional de La Seguridad de la Información que ayuda a las empresas a identificar y administrar los riesgos en todos los niveles de la organización.

**Controles** [*De ciberseguridad*]: Son un conjunto de acciones, priorizadas, ampliamente analizadas y de efectividad probada que pueden ser tomadas por las organizaciones para mejorar su nivel de ciberseguridad.

**CPU**: [*Del Inglés Central Processor Unit*]. Unidad Central de Procesamiento. Es el hardware dentro de una computadora u otros dispositivos programables. Su trabajo es interpretar las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y externas.

**CSIRT**: [*Del inglés* ***Computer Security Incident Response Team***] Equipo de Respuesta ante Incidencias de Seguridad Informáticas.

**Código Abierto**: Es un modelo de desarrollo de software basado en la colaboración abierta.

**Código Libre**: Concepto de las 4 libertades. Libertad 0: Poder usar el programa con cualquier propósito. Libertad 1: Poder estudiar cómo funciona el programa y poder modificarlo. Libertad 2: Poder distribuir copias del programa. Libertad 3: Poder mejorar el programa y poder compartir dichas mejoras para beneficio de todos.

**Datos**: Es una representación simbólica (numérica, alfabética, algorítmica, espacial, etc.) de un atributo o variable cuantitativa o cualitativa.

**Dependencias**: Es una aplicación o una biblioteca requerida por otro programa para poder funcionar correctamente.

**DHCP**: [*Del Ingles Dynamic Host Configuration Protocol*] es un conjunto de reglas para asignar direcciones IP y opciones de configuración a ordenadores y estaciones de trabajo en una red.

**Disponibilidad**: Se trata de la capacidad de un servicio, un sistema o una información, a ser accesible y utilizable por los usuarios o procesos autorizados cuando éstos lo requieran.

**Distribución**: [*GNU/Linux*] Es una distribución en si o una versión de software basada en el núcleo Linux que incluye determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios, dando así origen a ediciones domésticas, empresariales y para servidores.

**Docker**. Es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

**Ensambladores**: Se refiere a un tipo de programa informático que se encarga de traducir un fichero fuente escrito en un lenguaje ensamblador, a un fichero objeto que contiene código máquina, ejecutable directamente por el microprocesador.

**Escáner**: Herramienta que actúa con la finalidad de localizar todos los dispositivos conectados en su red (desde que tengan un número de IP).

**Estándar**: Que sirve de patrón, modelo o punto de referencia para medir o valorar cosas de la misma especie.

**Ethernet**: IEEE 802.3. Estándar de redes de área local para computadoras, por sus siglas en español Acceso Múltiple con Escucha de Portadora y Detección de Colisiones. Su nombre procede del concepto físico de éter.

**FTP**: El Protocolo de transferencia de archivos es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP, basado en la arquitectura cliente-servidor.

**GB**: Es una unidad de almacenamiento de información cuyo símbolo es el GB, equivalente a $10^9$ de bytes.

**GPL**: Es una licencia de derecho de autor ampliamente usada en el mundo del software libre y código abierto, y garantiza a los usuarios finales la libertad de usar, estudiar, compartir y modificar el software.

**Gusano**: Programa maligno que se replica para propagarse a otras computadoras.

**Hardware**: Se refiere a las partes físicas, tangibles, de un sistema informático, sus componentes eléctricos, electrónicos, electromecánicos y mecánicos.

**HIDS**: [*Del inglés HostIDS*] Es también conocido como Sistema de detección de intrusos en un Host. Busca detectar anomalías que indican un riesgo potencial, revisando las actividades en la máquina. Puede tomar medidas protectoras.

**Hipervisor**: Es una capa de software para realizar una virtualización de hardware que permite utilizar, al mismo tiempo, diferentes sistemas operativos en una misma computadora.

**Host** [*Anfitrión*] Se usa en informática para referirse a las computadoras u otros dispositivos conectados a una red que proveen y utilizan servicios de ella.

**HTTP**: [*Hypertext* Transfer Protocol], Protocolo de transferencia de hipertextos que se utiliza en algunas direcciones de internet.

**IDS**: [*Del inglés Intrusion Detection System*] Es un programa de detección de accesos no autorizados a un computador o a una red.

**IEEE**: Asociación mundial de ingenieros dedicada a la normalización y el desarrollo en áreas técnicas.

**Incidentes**: [*Ciberseguridad*] Es un evento o serie de eventos inesperados o no deseados, que tienen una probabilidad significativa de comprometer las operaciones del negocio; provocando una pérdida o uso indebido de información, interrupción parcial o total de los Sistemas

**Infraestructura**: Agrupa y organiza el conjunto de elementos tecnológicos que integran un proyecto, soportan las operaciones de una organización o sustentan una operación.

**Integridad**: Es la propiedad de la información, por la que se garantiza la exactitud de los datos transportados o almacenados, asegurando que no se ha producido su alteración, pérdida o destrucción, ya sea de forma accidental o intencionada, por errores de software o hardware o por condiciones medioambientales.

**ISO** [*Image*]: Es un archivo informático donde se almacena una copia o imagen exacta de un sistema de archivos. Se rige por el estándar ISO 9660, que le da nombre.

**.json**: Es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera un formato independiente del lenguaje.

**Kernel**: [*Núcleo*] Es un software que constituye una parte fundamental del sistema operativo, y se define como la parte que se ejecuta en modo privilegiado y se encarga de conceder el acceso al hardware de forma segura para todo el software que lo solicita.

**KVM**: [*Del inglés: Kernel-Based Virtual Machine Máquina Virtual basada en Núcleo*] Es una tecnología de virtualización de Código Abierto integrada a Linux.

**LDAP**: Es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

**Logs**: [Registro, Eventos, Historial] Grabación secuencial en un archivo o en una base de datos de todos los acontecimientos que afectan a un proceso particular. De esta forma constituye una evidencia del comportamiento del sistema.

**LVM**: Es una implementación de un gestor de volúmenes lógicos para el núcleo Linux.

**Malwares**: [*Programa Malicioso*] Cualquier tipo de software que realiza acciones dañinas en un sistema informático de forma intencionada y sin el conocimiento del usuario.

**MITM:** [*Man In The Middle u Hombre en el medio*] Es un ataque en el que se adquiere la capacidad de leer, insertar y modificar a voluntad. El atacante debe ser capaz de observar e interceptar mensajes entre las dos víctimas y procurar que ninguna de las víctimas conozca que el enlace entre ellos ha sido violado.

**Microprocesador**: Es el circuito integrado central más complejo de un sistema informático; a modo de ilustración, se le suele llamar por analogía el «cerebro» de un ordenador.

**Mitigar**: Es la reducción de la vulnerabilidad, es decir, la atenuación de los daños potenciales sobre la vida y los bienes causados por un evento geológico, como un sismo o tsunami; hidrológico, como una inundación o sequía; o sanitario o por un ciberataque.

**Modo Promiscuo:** Es aquel en el que una computadora conectada a una red compartida, tanto la basada en cable de cobre como la basada en tecnología inalámbrica, captura todo el tráfico que circula por ella.

**Monitoreo**: Procesos de recolección de registros y eventos de seguridad en los sistemas para ser correlacionado con métricas de mediciones definidas con miras a prevenir incidentes en la ciberseguridad.

**Multi Hilos**: Capacidad de poder ejecutar varios procesos o subprocesos de manera simultaneas de acuerdo con la disponibilidad del procesador.

**Multi Plataforma**: Es un atributo conferido a programas informáticos o métodos y conceptos de cómputo que son implementados, y operan internamente en múltiples plataformas informáticas.

**Multimedia**: Hace referencia a cualquier objeto o sistema que utiliza múltiples medios de expresión físicos o digitales electrónicas para presentar o comunicar información.

**Módulo**: Es una porción de un programa de ordenador. De las varias tareas que debe realizar un programa para cumplir con su función u objetivos, un módulo realizará, comúnmente, una de dichas tareas (o varias, en algún caso).

**NIDS**: [*Del inglés NetworkIDS*] También conocido como Sistema de detección de intrusos en una Red. Busca detectar anomalías que indiquen un riesgo potencial, tales como ataques de denegación de servicio, escáneres de puertos o intentos de entrar en un ordenador, analizando el tráfico en la red en tiempo real.

**Nodo**: Es un punto de intersección, conexión o unión de varios elementos que confluyen en el mismo lugar.

**Nomenclaturas**: Es el conjunto de términos que conforman un área del conocimiento.

**Nosql**: Es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas.

**Nube**: Es un paradigma que permite ofrecer servicios de computación a través de una red, que usualmente es internet.

**Ordenadores**: Es una máquina digital programable que ejecuta una serie de comandos para procesar los datos de entrada, obteniendo convenientemente información que posteriormente se envía a las unidades de salida.

**OSI**: [*Del inglés Open Systems Interconnection*] Es un modelo de referencia para los protocolos de la red, creado en el año 1980 por la Organización Internacional de Normalización.

**OSSIM**: [*Del inglés Open Source Security Information Management*] Es una colección de herramientas bajo la licencia GPL, diseñadas para ayudar a los administradores de red en la seguridad de las computadoras, detección de intrusos y prevención.

**P2P**: [*Peer to Peer*] Es un tipo de conexión con una arquitectura destinada a la comunicación entre aplicaciones. Esto permite a las personas o a los ordenadores compartir información y archivos de uno a otro sin necesidad de intermediarios.

**Patrones**: Es un tipo de tema de sucesos u objetos recurrentes con una serie de variables constantes, identificables dentro de un conjunto mayor de datos.

**Patrón De Conducta**: Se denomina patrón de conducta, a una forma de conducta que hace las veces de modelo.

**PCAPS**: Es una interfaz de una aplicación de programación para captura de paquetes.

**PDU**: [*Del inglés Protocol Data Unit*] Unidades de Datos de Protocolo también llamadas PDU, se utilizan para el intercambio de datos entre unidades disparejas, dentro de una capa del modelo OSI.

**PEC**: Iniciales de Practica de evaluación Continua.

**Phishing**: Conjunto de técnicas que persiguen el engaño a una víctima ganándose su confianza haciéndose pasar por una persona, empresa o servicio de confianza, para manipularla y hacer que realice acciones que no debería realizar.

**Plataformas**: Es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible. Dicho sistema está definido por un estándar alrededor del cual se determina una arquitectura de hardware y una plataforma de software.

**Procesador**: Componente electrónico donde se realizan los procesos lógicos.

**Protocolos**: Es un sistema de reglas que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellas para transmitir información por medio de cualquier tipo de variación de una magnitud física.

**Puerto**. [***Físico***] Es una interfaz a través de la cual se pueden enviar y recibir los diferentes tipos de datos. [***Lógico***] Es el valor que se usa, en el modelo de la capa de transporte, para distinguir entre las múltiples aplicaciones que se pueden conectar al mismo host, o puesto de trabajo.

**Qcow2**: Es uno de los formatos de imagen de disco apoyado por el emulador QEMU. Es una representación de un dispositivo del tamaño de bloque fijo en un archivo.

**Qemu**: Es un emulador de procesadores basado en la traducción dinámica de binarios (conversión del código binario de la arquitectura fuente en código entendible por la arquitectura huésped).

**Red**: Es un conjunto de equipos nodos y software conectados entre sí por medio de dispositivos físicos o inalámbricos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos, con la finalidad de compartir información, recursos y ofrecer servicios.

**Repositorios**: Es un grupo o conjunto de aplicaciones alojadas en un servidor o varios y que a través de un gestor de aplicaciones podremos instalar.

**Riesgo**: Es una medida de la magnitud de los daños frente a una situación peligrosa.

**Router**: [*Traducción: encaminador*] Es un dispositivo que permite interconectar computadoras que funcionan en el marco de una red. Su función es la de establecer la ruta que destinará a cada paquete de datos dentro de una red informática.

**Seguridad De La Información**: La seguridad de la información es el conjunto de medidas preventivas y reactivas de las organizaciones y sistemas tecnológicos que permiten resguardar y proteger la información buscando mantener la confidencialidad, la disponibilidad e integridad de datos.

**Seguridad Informática:** Es el área relacionada con la informática y la telemática que se enfoca en la protección de la infraestructura computacional y todo lo vinculado con la misma, y especialmente la información contenida en una computadora o circulante a través de las redes de computadoras.

**Servidores**: Es un conjunto de computadoras capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia. Los servidores se pueden ejecutar en cualquier tipo de computadora que cumpla con los requisitos del sistema a instalar.

**SIEM**: [*Del inglés Security Information and Event Management*] Un sistema de Gestión de Eventos e Información de Seguridad es un sistema que centraliza el almacenamiento y la interpretación de los datos relevante de seguridad.

**Sistema Operativo**: Es el software principal o conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación de software, ejecutándose en modo privilegiado respecto de los restantes.

**Sniffer**: Es un programa de captura de las tramas de una red de computadoras.

**SSD**: [*Del Ingles Solid State Disk*] Es un tipo de dispositivo de almacenamiento de datos que utiliza memoria no volátil, como la memoria flash, para almacenar datos, en lugar de los platos o discos magnéticos de las unidades de discos duros (HDD) convencionales.

**SSH**: [*De las ingles Secured Shell*] Es el nombre de un protocolo y del programa que lo implementa cuya principal función es el acceso remoto a un servidor por medio de un canal seguro en el que toda la información está cifrada.

**SSL**: [*Del inglés Secured Socket Layer*] Son protocolos criptográficos, que proporcionan comunicaciones seguras por una red, comúnmente Internet.

**Switch**: [*Conmutador*] Es el dispositivo digital lógico de interconexión de equipos que opera en la capa de enlace de datos del modelo OSI.

**Tarjeta De Red Ethernet**: [*También denominadas adaptadores de red, tarjetas de interfaz de red o NIC*] Constituye el elemento de conexión del equipo terminal a la red, es decir, conforma la interfaz entre un nodo y el cable de red, respetando el protocolo Ethernet.

**TFM**: Siglas en español de **Trabajo final de Máster**.

**Token-Ring**: Es una arquitectura de red desarrollada por IBM en los años 1970 con topología lógica en anillo y técnica de acceso de paso de testigo, usando una trama de 3 bytes llamado testigo que viaja alrededor del anillo. Token Ring queda definido en el estándar IEEE 802.5

**Trama**: [*Del inglés Frame*]. Es una unidad de envío de datos de capa 2. Es una serie sucesiva de bits, organizados en forma cíclica, que transportan información y que permiten en la recepción extraer esta información. Viene a ser el equivalente de paquete de datos o Paquete de red, en el Nivel de red

**Tráfico**: Hace referencia a los datos que se desplazan por una red en un momento determinado.

**Unix-Like**: Es un sistema que se comporta de manera similar a un sistema Unix, aunque no es necesario que sea certificado en ninguna versión de la *Single Unix Specification*.

**Usuario**: Es una persona o entidad que utiliza una computadora o un servicio de red.

**UTM**: [*Del inglés Unified Threat Management*] Gestión Unificada de Amenazas es un término que se refiere a un dispositivo de red único con múltiples funciones, las funcionalidades básicas que debe tener son: Antivirus. Firewall (cortafuegos) Sistema de detección/prevención de intrusos.

**Vectores De Ataques**: Es un método que utiliza una amenaza para atacar un sistema.

**Virtualización**: Es la creación a través de software de una representación de algún recurso tecnológico. Por tanto este software tiene la función de simular la existencia del recurso tecnológico que se quiere virtualizar.

**VLAN**: [*Del inglés Virtual LAN*]. Es un método para crear redes lógicas independientes dentro de una misma red física. Varias VLAN pueden coexistir en un único conmutador físico o en una única red física.

**VPN**: [*Del inglés Virtual Private Network*] Es una tecnología de red de ordenadores que permite una extensión segura de la red de área local sobre una red pública o no controlada como Internet.

**Vulnerabilidades**: Esta puede considerarse como la debilidad en los procedimientos de seguridad de un sistema de información.

**Web**: Conjunto de información que se encuentra en una dirección determinada de internet.

**YAML**: Es un formato de serialización de datos legible por humanos inspirado en lenguajes como XML, C, Python, Perl, así como el formato para correos electrónicos especificado en RFC 2822.

Nota: El origen de estas definiciones son de libre búsqueda en la Internet.

# 6. Bibliografía

## *6.1 Referencias bibliográficas:*

### *6.1.1 Investigación de Campo*

DON MURDOCH. [***Blue Team Handbook, Incident Response Edition***]
Version 2.2, October 2016. ISBN-10: 1500734756 & ISBN-13: 978-1500734756

## *6.2 Referencias Webs:*

### *6.2.1 Introducción*

1- **Wikipedia**: Redes de Computadoras
https://es.wikipedia.org/wiki/Red_de_computadoras
19 febrero 2021

2- **Wikipedia:** El Router
https://es.wikipedia.org/wiki/Router
19 febrero 2021

3- **Wikipedia**: El Switch.
https://es.wikipedia.org/wiki/Conmutador_(dispositivo_de_red)
19 febrero de 2021

4- **Wikipedia**: El ordenador
https://es.wikipedia.org/wiki/Computadora
19 febrero de 2021

5- **Wikipedia**: Servidores.
https://es.wikipedia.org/wiki/Servidor
19 febrero de 2021

6- **Wikipedia**: Centrales Telefónicas IP
https://es.wikipedia.org/wiki/Central_telefónica_IP
19 febrero de 2021

7- **Wikipedia**: Estándares.
https://es.wikipedia.org/wiki/Normalización
20 febrero de 2021

8- **Wikipedia**: Token Ring
https://es.wikipedia.org/wiki/Token_Ring
20 febrero de 2021

9- **Wikipedia**: IBM
https://es.wikipedia.org/wiki/IBM
20 febrero de 2021

10- **IBM**: IBM
https://www.ibm.com/us-en/
20 febrero de 2021

11- **Wikipedia**: Mbps
https://es.wikipedia.org/wiki/Megabit_por_segundo
20 febrero de 2021

12- **Wikipedia**: Ethernet
https://es.wikipedia.org/wiki/Ethernet
20 febrero de 2021

13- **Wikipedia**: 802.3
https://es.wikipedia.org/wiki/IEEE_802.3
22 febrero de 2021

14- **Wikipedia**: Redes inalámbricas

https://es.wikipedia.org/wiki/Red_inalámbrica
22 febrero de 2021

15- **Wikipedia**: 802.11
https://es.wikipedia.org/wiki/IEEE_802.11
22 febrero de 2021

15- **Wikipedia**: Medios de transmisión
https://es.wikipedia.org/wiki/Medio_de_transmisión
22 febrero de 2021

16- **Wikipedia**: Modelo de Referencia OSI.
https://es.wikipedia.org/wiki/Modelo_OSI
22 febrero de 2021

17- **Wikipedia**: Aplicación
https://es.wikipedia.org/wiki/Software_de_aplicación
22 febrero de 2021

18- **Wikipedia**: Conmutación
https://es.wikipedia.org/wiki/Conmutación_(redes_de_comunicación)
22 febrero de 2021

19- **Wikipedia**: Capa de enlace Modelo OSI.
https://es.wikipedia.org/wiki/Capa_de_enlace_de_datos
22 febrero de 2021

21- **Wikipedia**: PDU
https://es.wikipedia.org/wiki/Unidad_de_datos_de_protocolo
22 febrero de 2021

22- **Wikipedia**: Seguridad de la información.
https://es.wikipedia.org/wiki/Seguridad_de_la_información
24 febrero de 2021

23- **Techtarget**: Tipos de incidentes de seguridad.
https://searchdatacenter.techtarget.com/es/tutoriales/10-tipos-de-incidentes-de-seguridad-y-como-manejarlos
24 febrero de 2021

24- **Wikipedia**: GNU-GPL.
https://es.wikipedia.org/wiki/GNU_General_Public_License
24 febrero de 2021

25- **GNU Org:** GNU-GPL
https://www.gnu.org/licenses/gpl-3.0.html
24 febrero de 2021

26- **Fedora:** Fedora Workstations.
https://getfedora.org/es/workstation/
24 febrero de 2021

27- **Proxmox:** Proxmox PE
https://www.proxmox.com/
24 febrero de 2021

28- **Wikipedia**: GNU/Linux
https://es.wikipedia.org/wiki/GNU/Linux
24 febrero de 2021

29- **GNU Org:** Linux and GNU
https://www.gnu.org/gnu/linux-and-gnu.en.html
24 febrero de 2021

30- **UBUNTU:** Ubuntu Server.
https://ubuntu.com/
24 febrero de 2021

31- **Ebay:** Lenovo Ideapad Flex Price.
https://www.ebay.com/itm/LENOVO-IDEAPAD-FLEX-5-14-FHD-Ryzen-5-4500U-16GB-256GB-SSD-81X20005US- W10/164247265907
24 febrero de 2021

32- **Ebay:** HP Elite Book Price.
https://www.ebay.com/itm/HP-EliteBook-x360-1030-G3-13-3-i5-8350u-1-7Ghz-16gb-ram-500gb-m-2-ssd-W10-Pro/164724258259
24 febrero de 2021

33- **Ebay:** Cisco 2950 Catalyst Price.
https://www.ebay.com/itm/Cisco-WS-C2950-12-Catalyst-2950-Series-24-Port-Switch/233877641320
24 febrero de 2021

34- **Presidencia Republica Dominicana:** Decreto 95-21 sobre el Estado de Emergencias
https://www.presidencia.gob.do/decretos/95-21
24 febrero de 2021

35- **Protegermipc:** Mejores IDS
https://protegermipc.net/2018/02/22/mejores-ids-opensource-deteccion-de-intrusiones/
24 febrero de 2021

36- **Wikipedia**: IDS
https://es.wikipedia.org/wiki/Sistema_de_detección_de_intrusos
24 febrero de 2021

37- **Wikipedia**: Snort
https://en.wikipedia.org/wiki/Snort_(software)
24 febrero de 2021

38- **Snort:** Snort.
https://www.snort.org/
24 febrero de 2021

39- **Wikipedia**: Suricata
https://en.wikipedia.org/wiki/Suricata_(software)
24 febrero de 2021

40- **Suricata:** Suricata
https://suricata-ids.org/
24 febrero de 2021

41- **Seek:** Seek
https://zeek.org/
24 febrero de 2021

42- **Clavei:** Que es un IDS
https://www.clavei.es/blog/que-es-un-ids-o-intrusion-detection-system/
25 febrero de 2021

43- **Sofecom**: SIEM
https://sofecom.com/que-es-un-siem/
24 febrero de 2021

44- **Att:** OSSIM
https://cybersecurity.att.com/products/ossim
24 febrero de 2021

45- **Wikipedia**: Elasticsearch
https://es.wikipedia.org/wiki/Elasticsearch
24 febrero de 2021

46- **Wikipedia**: Logstash
https://es.wikipedia.org/wiki/Logstash
24 febrero de 2021

47- **Wikipedia**: Kibana
https://en.wikipedia.org/wiki/Kibana
24 febrero de 2021

48- **Elasticsearch:** ELK
https://www.elastic.co/es/
24 febrero de 2021

49- Tic **Defense**: Pignius
https://www.ticdefense.com/
24 febrero de 2021

50- **GNU Org:** Linux and GNU
https://www.gnu.org/gnu/linux-and-gnu.en.html
24 febrero de 2021

51- **Proxmox:** Proxmox
https://www.proxmox.com/
26 febrero de 2021

52- **Wikipedia**: Proxmox
https://es.wikipedia.org/wiki/Proxmox_Virtual_Environment
26 febrero de 2021

53- **Wikipedia**: LVM
https://es.wikipedia.org/wiki/Logical_Volume_Manager_(Linux)
26 febrero de 2021

### *6.2.2 Investigación de Campo*

54- **Wikipedia**: BotNet
https://es.wikipedia.org/wiki/Botnet
03 marzo de 2021

55- **Wikipedia**: IDS
https://es.wikipedia.org/wiki/Sistema_de_detección_de_intrusos
03 marzo de 2021

56- **Wikipedia**: NIDS
https://es.wikipedia.org/wiki/NIDS
03 marzo de 2021

57- **Wikipedia**: Snort
https://es.wikipedia.org/wiki/Snort
03 marzo de 2021

58- **Wikipedia**: Lenguaje C
https://es.wikipedia.org/wiki/ C_(lenguaje_de_programacion)
04 marzo de 2021

59- **Wikipedia**: GPL versión 2
https://es.wikipedia.org/wiki/ GNU_General_Public_License#Versión_2
04 marzo de 2021

60- **Wikipedia**: Multihilo
https://es.wikipedia.org/wiki/Multihilo
04 marzo de 2021

61- **Suricata**: Suricata
https://suricata-ids.org
04 marzo de 2021

62- **Wikipedia**: Open Security Fountation
https://en.wikipedia.org/wiki/Open_Security_Foundation
05 marzo de 2021

63- **Wikipedia**: GPL
https://en.wikipedia.org/wiki/GNU_General_Public_License
05 marzo de 2021

64- **Wikipedia**: PCAP
https://es.wikipedia.org/wiki/Pcap_(interfaz)
05 marzo de 2021

65- **Wikipedia**: YAML
https://es.wikipedia.org/wiki/YAML
06 marzo de 2021

66- **Wikipedia**: JSON
https://es.wikipedia.org/wiki/JSON
06 marzo de 2021

67- **Wikipedia**: Script
https://es.wikipedia.org/wiki/Script
07 marzo de 2021

68- **Zeek**: IDS Zeek
https://docs.zeek.org/
07 marzo de 2021

69- **Wikipedia**: Vern Pazson
https://en.wikipedia.org/wiki/Vern_Paxson
07 marzo de 2021

70- **Wikipedia**: Licencia BSD
https://en.wikipedia.org/wiki/BSD_licenses

08 marzo de 2021

71- **Open Wips**: OpenWips
https://openwips-ng.org/index.html
08 marzo de 2021

72- **Wikipedia**: Sguil
https://en.wikipedia.org/wiki/Sguil
08 marzo de 2021

73- **GitHub**: Sguil
http://bammv.github.io/sguil/index.html
08 marzo 2021

74- **Wikipedia**: GPL versión 3
https://en.wikipedia.org/wiki/GNU_General_Public_License#Version_3
09 marzo de 2021

75- **Cisco**: NGIPS
https://www.cisco.com/c/en/us/products/security/ngips/index.html
09 marzo de 2021

76- **Fidelis Security**: Fidelis Network
https://fidelissecurity.com/products/network/
09 marzo de 2021

77- **PFSENSE**: pfsense Firewall
https://www.pfsense.org/
09 marzo de 2021

78- **NetGate**: pfsense NetGate
https://www.netgate.com/solutions/pfsense-plus/
09 marzo de 2021

79- **Wikipedia**: Licencia Apache
https://en.wikipedia.org/wiki/Apache_License
09 marzo de 2021

80- **Wikipedia**: Licencia BSD
https://es.wikipedia.org/wiki/Licencia_BSD
09 marzo de 2021

81- **Apache**: Licencia Apache
https://www.apache.org/licenses/
10 marzo de 2021

82- **Mundo Cisco**: Sniffer
http://www.mundocisco.com/2009/08/que-es-un-sniffer.html
10 marzo de 2021

83- **Wikipedia**: Analizador de Paquetes
https://es.wikipedia.org/wiki/Analizador_de_paquetes
10 marzo de 2021

84- **WireShark**: Wireshark
https://www.wireshark.org/
10 marzo de 2021

85- **Ettercap Project**: EtterCap
https://www.ettercap-project.org/
10 marzo de 2021

86- **BetterCap Project**: Bettercap
https://www.bettercap.org
10 marzo de 2021

87- **Safecom**: SIEM
https://sofecom.com/que-es-un-siem/
11 marzo de 2021

88- Cybersecurity ATT: OSSIM
https://cybersecurity.att.com/products/ossim
11 marzo de 2021

89- **Apache**: Metron
https://metron.apache.org/about/
11 marzo de 2021

90- **GitHub**: Open Soc
https://opensoc.github.io/
12 marzo de 2021

91- **Apache**: Licencia Apache 2.0
http://www.apache.org/licenses/LICENSE-2.0
12 marzo de 2021

92- **Apache**: Metron
https://cwiki.apache.org/confluence/display/METRON/Documentation
12 marzo de 2021

93- **Security Onion**: Security Onion ISO
https://securityonionsolutions.com/software
13 marzo de 2021

94- **Wikipedia**: Elasticsearch
https://es.wikipedia.org/wiki/Elasticsearch
13 marzo de 2021

95- **Wikipedia**: Logstash
https://es.wikipedia.org/wiki/Logstash
14 marzo de 2021

96- **Wikipedia**: Kibana
https://en.wikipedia.org/wiki/Kibana
14 marzo de 2021

97- **Elastic**: ELK
https://www.elastic.co/es/
14 marzo de 2021

98- **Wikipedia**: NoSQL
https://es.wikipedia.org/wiki/NoSQL
15 marzo de 2021

99- **Microsoft**: Sentinel
https://azure.microsoft.com/en-us/services/azure-sentinel/#product-overview
15 marzo de 2021

100- **Data Dog**: Datadog
https://www.datadoghq.com/
16 marzo de 2021

101- **Wikipedia**: Virtualization
https://en.wikipedia.org/wiki/Virtualization
20 marzo de 2021

102- **VMWare**: VMWare
https://www.vmware.com/
21 marzo de 2021

103- **Wikipedia**: ESXi
https://en.wikipedia.org/wiki/VMware_ESXi
21 marzo de 2021

104- **Wikipedia**: Dell EMC
https://es.wikipedia.org/wiki/Dell_EMC
21 marzo de 2021

105- **Dell Technologies**: VMWare
https://www.delltechnologies.com/en-us/converged-infrastructure/index.htm
21 marzo de 2021

106- **Microsoft**: Hyper-V
https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/
22 marzo de 2021

107- **Debian**: Debian
https://www.debian.org/
22 marzo de 2021

108- **Wikipedia**: Sistemas Operativos
https://es.wikipedia.org/wiki/Sistema_operativo
23 marzo de 2021

109- **Wikipedia**: Compiladores.
https://es.wikipedia.org/wiki/Compilador
23 marzo de 2021

110- **Wikipedia**: Ensambladores
https://es.wikipedia.org/wiki/Ensamblador
23 marzo de 2021
111- **Wikipedia**: Cargadores
https://es.wikipedia.org/wiki/Cargador_de_programas
23 marzo de 2021

112- **Wikipedia**: Linus B. Torbal
https://es.wikipedia.org/wiki/Linus_Torvalds
23 marzo de 2021

113- **Wikipedia**: GNU/Linux
https://es.wikipedia.org/wiki/GNU/Linux
23 marzo de 2021

114- **Wikipedia**: Licencia GPL
https://es.wikipedia.org/wiki/GNU_General_Public_License
24 marzo de 2021

115- **Fedora**: Fedora OS
https://getfedora.org/
24 marzo de 2021

116- **CentOS ORG**: CentOS OS
https://www.centos.org/
24 marzo de 2021

117- **CentOS ORG**: CentOS Stream OS
https://www.centos.org/centos-stream/
24 marzo de 2021

118- **My Linux**: CentOS Stream OS
https://www.muylinux.com/2020/12/09/centos-8-stream-descontinuado-rolling-release/
24 marzo de 2021

119- **Debian**: Debian OS
https://www.debian.org/
24 marzo de 2021

120- **Ubuntu**: Ubuntu OS
https://ubuntu.com/
24 marzo de 2021

### *6.2.3 Implementación*

121- **Ubuntu**: Ubuntu OS Download
https://ubuntu.com/download/server
31 marzo de 2021

122- **Ubuntu**: Ubuntu OS Install
https://help.ubuntu.com/18.04/serverguide/installing-from-cd.html
31 marzo de 2021

123- **Suricata**: Suricata Install
https://suricata.readthedocs.io/en/latest/install.html
01 abril de 2021

124- **Emergin Threats**: Suricata Rules
https://rules.emergingthreats.net/open/suricata/
04 abril de 2021

125- **Wikipedia**: Dominio de colisión
https://es.wikipedia.org/wiki/Dominio_de_colision
04 abril de 2021

126- **CISCO**: Catalyst 6500
https://www.cisco.com/c/en/us/support/docs/switches/catalyst-6500-series-switches/10570-41.html
05 abril de 2021

127- **Emergin Threats**: Suricata Rules

https://rules.emergingthreats.net/open/suricata/
05 abril de 2021

128- **The Red Docs**: Suricata Rules
https://suricata.readthedocs.io/en/latest/rules/intro.html
05 abril de 2021

129- **Digital Ocean**: ELK Install on UBUNTU
https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-18-04
11 abril de 2021

130- **Nginix**: Nginix
https://www.nginx.com/
11 abril de 2021

131- **Digital Ocean**: Nginix Install on UBUNTU
https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-18-04
11 abril de 2021

132- **Elastic**: Filebeat Suricata Module
https://www.elastic.co/guide/en/beats/filebeat/6.8/filebeat-module-suricata.html
11 abril de 2021

### *6.2.4 Conclusiones finales*

133- **Suricata**: Reglas
https://suricata.readthedocs.io/en/suricata-6.0.0/rules/index.html
05 mayo de 2021

# 7. Anexos

*7.1. Configuración Red NIDS [Ubuntu 18.04].*

```
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp1s0:
      dhcp4: true

    enp2s0:
          #dhcp4: false
      addresses: [10.10.1.61/24]
      #gateway4: 10.10.1.1
      #nameservers:
      #search: [reinterna.local]
      #addresses: [10.10.1.1]
  version:
```

*7.2. Configuración Suricata [Ubuntu 18.04].*

```
%YAML 1.1
---

# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricatayaml

##
## Step 1: inform Suricata about your network
##

vars:
  # more specifc is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12,10.10.1.0/24]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

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

    HTTP_SERVERS: "$HOME_NET"
    SMTP_SERVERS: "$HOME_NET"
    SQL_SERVERS: "$HOME_NET"
    DNS_SERVERS: "$HOME_NET"
    TELNET_SERVERS: "$HOME_NET"
    AIM_SERVERS: "$EXTERNAL_NET"
    DNP3_SERVER: "$HOME_NET"
    DNP3_CLIENT: "$HOME_NET"
    MODBUS_CLIENT: "$HOME_NET"
```

```
    MODBUS_SERVER: "$HOME_NET"
    ENIP_CLIENT: "$HOME_NET"
    ENIP_SERVER: "$HOME_NET"

  port-groups:
    HTTP_PORTS: "80"
    SHELLCODE_PORTS: "!80"
    ORACLE_PORTS: 1521
    SSH_PORTS: 22
    DNP3_PORTS: 20000
    MODBUS_PORTS: 502


##
## Step 2: select the rules to enable or disable
##

default-rule-path: /etc/suricata/rules
rule-files:
 - botcc.rules
 - ciarmy.rules
 - compromised.rules
 - drop.rules
 - dshield.rules
# - emerging-activex.rules
 - emerging-attack_response.rules
 - emerging-chat.rules
 - emerging-current_events.rules
 - emerging-dns.rules
 - emerging-dos.rules
 - emerging-exploit.rules
 - emerging-ftp.rules
# - emerging-games.rules
 - emerging-icmp_info.rules
# - emerging-icmp.rules
 - emerging-imap.rules
# - emerging-inappropriate.rules
 - emerging-malware.rules
 - emerging-misc.rules
 - emerging-mobile_malware.rules
 - emerging-netbios.rules
 - emerging-p2p.rules
 - emerging-policy.rules
 - emerging-pop3.rules
 - emerging-rpc.rules
 - emerging-scada.rules
 - emerging-scan.rules
# - emerging-shellcode.rules
 - emerging-smtp.rules
 - emerging-snmp.rules
 - emerging-sql.rules
 - emerging-telnet.rules
 - emerging-tftp.rules
 - emerging-trojan.rules
 - emerging-user_agents.rules
 - emerging-voip.rules
```

```
 - emerging-web_client.rules
 - emerging-web_server.rules
# - emerging-web_specific_apps.rules
 - emerging-worm.rules
 - tor.rules
 - uoc.rules
 - uoc_ipre.rules
- uoc_botnet.rules
# - decoder-events.rules # available in suricata sources under rules dir
# - stream-events.rules  # available in suricata sources under rules dir
 - http-events.rules    # available in suricata sources under rules dir
 - smtp-events.rules    # available in suricata sources under rules dir
 - dns-events.rules     # available in suricata sources under rules dir
 - tls-events.rules     # available in suricata sources under rules dir
# - modbus-events.rules  # available in suricata sources under rules dir
# - app-layer-events.rules  # available in suricata sources under rules dir
# - dnp3-events.rules       # available in suricata sources under rules dir
#classification-file: /etc/suricata/classification.config
classification-file: /etc/suricata/rules/classification.config
reference-config-file: /etc/suricata/reference.config
# threshold-file: /etc/suricata/threshold.config


##
## Step 3: select outputs to enable
##

# The default logging directory.  Any log or output file will be
# placed here if its not specified with a full path name. This can be
# overridden with the -l command line parameter.
default-log-dir: /var/log/suricata/

# global stats configuration
stats:
  enabled: yes
  # The interval field (in seconds) controls at what interval
  # the loggers are invoked.
  interval: 8

# Configure the type of alert (and other) logging you would like.
outputs:
  # a line based alerts log similar to Snort's fast.log
  - fast:
      enabled: yes
      filename: fast.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

  # Extensible Event Format (nicknamed EVE) event log in JSON format
  - eve-log:
      enabled: yes
      filetype: regular #regular|syslog|unix_dgram|unix_stream|redis
      filename: eve.json
      #prefix: "@cee: " # prefix to prepend to each log entry
      # the following are valid when type: syslog above
      #identity: "suricata"
```

```
#facility: local5
#level: Info ## possible levels: Emergency, Alert, Critical,
            ## Error, Warning, Notice, Info, Debug
#redis:
#  server: 127.0.0.1
#  port: 6379
#  mode: list ## possible values: list (default), channel
#  key: suricata ## key or channel to use (default to suricata)
# Redis pipelining set up. This will enable to only do a query every
# 'batch-size' events. This should lower the latency induced by network
# connection at the cost of some memory. There is no flushing implemented
# so this setting as to be reserved to high traffic suricata.
#  pipelining:
#    enabled: yes ## set enable to yes to enable query pipelining
#    batch-size: 10 ## number of entry to keep in buffer
types:
  - alert:
      # payload: yes             # enable dumping payload in Base64
      # payload-buffer-size: 4kb # max size of payload buffer to output in eve-log
      # payload-printable: yes   # enable dumping payload in printable (lossy) format
      # packet: yes              # enable dumping of packet (without stream segments)
      http: yes               # enable dumping of http fields
      tls: yes                # enable dumping of tls fields
      ssh: yes                # enable dumping of ssh fields
      smtp: yes               # enable dumping of smtp fields
      dnp3: yes               # enable dumping of DNP3 fields

      # Enable the logging of tagged packets for rules using the
      # "tag" keyword.
      tagged-packets: yes

      # HTTP X-Forwarded-For support by adding an extra field or overwriting
      # the source or destination IP address (depending on flow direction)
      # with the one reported in the X-Forwarded-For HTTP header. This is
      # helpful when reviewing alerts for traffic that is being reverse
      # or forward proxied.
      xff:
        enabled: no
        # Two operation modes are available, "extra-data" and "overwrite".
        mode: extra-data
        # Two proxy deployments are supported, "reverse" and "forward". In
        # a "reverse" deployment the IP address used is the last one, in a
        # "forward" deployment the first IP address is used.
        deployment: reverse
        # Header name where the actual IP address will be reported, if more
        # than one IP address is present, the last IP address will be the
        # one taken into consideration.
        header: X-Forwarded-For
  - http:
      extended: yes     # enable this for extended logging information
      # custom allows additional http fields to be included in eve-log
      # the example below adds three additional fields when uncommented
      #custom: [Accept-Encoding, Accept-Language, Authorization]
  - dns:
      # control logging of queries and answers
      # default yes, no to disable
```

```
        query: yes     # enable logging of DNS queries
        answer: yes    # enable logging of DNS answers
        # control which RR types are logged
        # all enabled if custom not specified
        #custom: [a, aaaa, cname, mx, ns, ptr, txt]
    - tls:
        extended: yes     # enable this for extended logging information
    - files:
        force-magic: no   # force logging magic on all logged files
        # force logging of checksums, available hash functions are md5,
        # sha1 and sha256
        #force-hash: [md5]
    #- drop:
    #    alerts: yes      # log alerts that caused drops
    #    flows: all       # start or all: 'start' logs only a single drop
    #                     # per flow direction. All logs each dropped pkt.
    - smtp:
        #extended: yes # enable this for extended logging information
        # this includes: bcc, message-id, subject, x_mailer, user-agent
        # custom fields logging from the list:
        #  reply-to, bcc, message-id, subject, x-mailer, user-agent, received,
        #  x-originating-ip, in-reply-to, references, importance, priority,
        #  sensitivity, organization, content-md5, date
        #custom: [received, x-mailer, x-originating-ip, relays, reply-to, bcc]
        # output md5 of fields: body, subject
        # for the body you need to set app-layer.protocols.smtp.mime.body-md5
        # to yes
        #md5: [body, subject]

    - ssh
    - stats:
        totals: yes       # stats for all threads merged together
        threads: no       # per thread stats
        deltas: no        # include delta values
    # bi-directional flows
    - flow
    # uni-directional flows
    #- netflow
    #- dnp3

# alert output for use with Barnyard2
- unified2-alert:
    enabled: no
    filename: unified2.alert

    # File size limit.  Can be specified in kb, mb, gb.  Just a number
    # is parsed as bytes.
    #limit: 32mb

    # Sensor ID field of unified2 alerts.
    #sensor-id: 0

    # Include payload of packets related to alerts. Defaults to true, set to
    # false if payload is not required.
    #payload: yes
```

```
    # HTTP X-Forwarded-For support by adding the unified2 extra header or
    # overwriting the source or destination IP address (depending on flow
    # direction) with the one reported in the X-Forwarded-For HTTP header.
    # This is helpful when reviewing alerts for traffic that is being reverse
    # or forward proxied.
    xff:
      enabled: no
      # Two operation modes are available, "extra-data" and "overwrite". Note
      # that in the "overwrite" mode, if the reported IP address in the HTTP
      # X-Forwarded-For header is of a different version of the packet
      # received, it will fall-back to "extra-data" mode.
      mode: extra-data
      # Two proxy deployments are supported, "reverse" and "forward". In
      # a "reverse" deployment the IP address used is the last one, in a
      # "forward" deployment the first IP address is used.
      deployment: reverse
      # Header name where the actual IP address will be reported, if more
      # than one IP address is present, the last IP address will be the
      # one taken into consideration.
      header: X-Forwarded-For

  # a line based log of HTTP requests (no alerts)
  - http-log:
      enabled: no
      filename: http.log
      append: yes
      #extended: yes     # enable this for extended logging information
      #custom: yes       # enabled the custom logging format (defined by customformat)
      #customformat:  "%{%D-%H:%M:%S}t.%z  %{X-Forwarded-For}i  %H  %m  %h  %u
%s %B %a:%p -> %A:%P"
      #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

  # a line based log of TLS handshake parameters (no alerts)
  - tls-log:
      enabled: no  # Log TLS connections.
      filename: tls.log # File to store TLS logs.
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'
      #extended: yes # Log extended information like fingerprint

  # output module to store certificates chain to disk
  - tls-store:
      enabled: no
      #certs-log-dir: certs # directory to store the certificates files

  # a line based log of DNS requests and/or replies (no alerts)
  - dns-log:
      enabled: no
      filename: dns.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

  # Packet log... log packets in pcap format. 3 modes of operation: "normal"
  # "multi" and "sguil".
  #
  # In normal mode a pcap file "filename" is created in the default-log-dir,
```

```
  # or are as specified by "dir".
  # In multi mode, a file is created per thread. This will perform much
  # better, but will create multiple files where 'normal' would create one.
  # In multi mode the filename takes a few special variables:
  # - %n -- thread number
  # - %i -- thread id
  # - %t -- timestamp (secs or secs.usecs based on 'ts-format'
  # E.g. filename: pcap.%n.%t
  #
  # Note that it's possible to use directories, but the directories are not
  # created by Suricata. E.g. filename: pcaps/%n/log.%s will log into the
  # per thread directory.
  #
  # Also note that the limit and max-files settings are enforced per thread.
  # So the size limit when using 8 threads with 1000mb files and 2000 files
  # is: 8*1000*2000 ~ 16TiB.
  #
  # In Sguil mode "dir" indicates the base directory. In this base dir the
  # pcaps are created in th directory structure Sguil expects:
  #
  # $sguil-base-dir/YYYY-MM-DD/$filename.<timestamp>
  #
  # By default all packets are logged except:
  # - TCP streams beyond stream.reassembly.depth
  # - encrypted streams after the key exchange
  #
  - pcap-log:
      enabled: no
      filename: log.pcap

      # File size limit.  Can be specified in kb, mb, gb.  Just a number
      # is parsed as bytes.
      limit: 1000mb

      # If set to a value will enable ring buffer mode. Will keep Maximum of "max-files" of
size "limit"
      max-files: 2000

      mode: normal # normal, multi or sguil.

      # Directory to place pcap files. If not provided the default log
      # directory will be used. Required for "sguil" mode.
      #dir: /nsm_data/

      #ts-format: usec # sec or usec second format (default) is filename.sec usec is
filename.sec.usec
      use-stream-depth: no #If set to "yes" packets seen after reaching stream inspection
depth are ignored. "no" logs all packets
      honor-pass-rules: no # If set to "yes", flows in which a pass rule matched will
stopped being logged.

  # a full alerts log containing much information for signature writers
  # or for investigating suspected false positives.
  - alert-debug:
      enabled: no
      filename: alert-debug.log
```

```
    append: yes
    #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

# alert output to prelude (http://www.prelude-technologies.com/) only
# available if Suricata has been compiled with --enable-prelude
- alert-prelude:
    enabled: no
    profile: suricata
    log-packet-content: no
    log-packet-header: yes

# Stats.log contains data from various counters of the suricata engine.
- stats:
    enabled: yes
    filename: stats.log
    totals: yes       # stats for all threads merged together
    threads: no       # per thread stats
    #null-values: yes  # print counters that have value 0

# a line based alerts log similar to fast.log into syslog
- syslog:
    enabled: no
    # reported identity to syslog. If ommited the program name (usually
    # suricata) will be used.
    #identity: "suricata"
    facility: local5
    #level: Info ## possible levels: Emergency, Alert, Critical,
                ## Error, Warning, Notice, Info, Debug

# a line based information for dropped packets in IPS mode
- drop:
    enabled: no
    filename: drop.log
    append: yes
    #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

# output module to store extracted files to disk
#
# The files are stored to the log-dir in a format "file.<id>" where <id> is
# an incrementing number starting at 1. For each file "file.<id>" a meta
# file "file.<id>.meta" is created.
#
# File extraction depends on a lot of things to be fully done:
# - file-store stream-depth. For optimal results, set this to 0 (unlimited)
# - http request / response body sizes. Again set to 0 for optimal results.
# - rules that contain the "filestore" keyword.
- file-store:
    enabled: no       # set to yes to enable
    log-dir: files    # directory to store the files
    force-magic: no   # force logging magic on all stored files
    # force logging of checksums, available hash functions are md5,
    # sha1 and sha256
    #force-hash: [md5]
    force-filestore: no # force storing of all files
    # override global stream-depth for sessions in which we want to
    # perform file extraction. Set to 0 for unlimited.
```

```
    #stream-depth: 0
    #waldo: file.waldo # waldo file to store the file_id across runs

  # output module to log files tracked in a easily parsable json format
  - file-log:
      enabled: no
      filename: files-json.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

      force-magic: no   # force logging magic on all logged files
      # force logging of checksums, available hash functions are md5,
      # sha1 and sha256
      #force-hash: [md5]

  # Log TCP data after stream normalization
  # 2 types: file or dir. File logs into a single logfile. Dir creates
  # 2 files per TCP session and stores the raw TCP data into them.
  # Using 'both' will enable both file and dir modes.
  #
  # Note: limited by stream.depth
  - tcp-data:
      enabled: no
      type: file
      filename: tcp-data.log

  # Log HTTP body data after normalization, dechunking and unzipping.
  # 2 types: file or dir. File logs into a single logfile. Dir creates
  # 2 files per HTTP session and stores the normalized data into them.
  # Using 'both' will enable both file and dir modes.
  #
  # Note: limited by the body limit settings
  - http-body-data:
      enabled: no
      type: file
      filename: http-data.log

  # Lua Output Support - execute lua script to generate alert and event
  # output.
  # Documented at:
  # https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Lua_Output
  - lua:
      enabled: no
      #scripts-dir: /etc/suricata/lua-output/
      scripts:
      #   - script1.lua

# Logging configuration.  This is not about logging IDS alerts/events, but
# output about what Suricata is doing, like startup messages, errors, etc.
logging:
  # The default log level, can be overridden in an output section.
  # Note that debug level logging will only be emitted if Suricata was
  # compiled with the --enable-debug configure option.
  #
  # This value is overriden by the SC_LOG_LEVEL env var.
  default-log-level: notice
```

```
  # The default output format.  Optional parameter, should default to
  # something reasonable if not provided.  Can be overriden in an
  # output section.  You can leave this out to get the default.
  #
  # This value is overriden by the SC_LOG_FORMAT env var.
  #default-log-format: "[%i] %t - (%f:%l) <%d> (%n) -- "

  # A regex to filter output.  Can be overridden in an output section.
  # Defaults to empty (no filter).
  #
  # This value is overriden by the SC_LOG_OP_FILTER env var.
  default-output-filter:

  # Define your logging outputs.  If none are defined, or they are all
  # disabled you will get the default - console output.
  outputs:
  - console:
      enabled: yes
      # type: json
  - file:
      enabled: yes
      level: info
      filename: /var/log/suricata/suricata.log
      # type: json
  - syslog:
      enabled: no
      facility: local5
      format: "[%i] <%d> -- "
      # type: json


##
## Step 4: configure common capture settings
##
## See "Advanced Capture Options" below for more options, including NETMAP
## and PF_RING.
##

# Linux high speed capture support
af-packet:
  - interface: enp2s0
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99
    # Default AF_PACKET cluster type. AF_PACKET can load balance per flow or per
hash.
    # This is only supported for Linux kernel > 3.1
    # possible value are:
    #  * cluster_round_robin: round robin load balancing
    #  * cluster_flow: all packets of a given flow are send to the same socket
    #  * cluster_cpu: all packets treated in kernel by a CPU are send to the same socket
    #   * cluster_qm: all packets linked by network card to a RSS queue are sent to the
same
    #  socket. Requires at least Linux 3.14.
```

```
  #  * cluster_random: packets are sent randomly to sockets but with an equipartition.
  #  Requires at least Linux 3.14.
  #   * cluster_rollover: kernel rotates between sockets filling each socket before moving
  #  to the next. Requires at least Linux 3.10.
  # Recommended modes are cluster_flow on most boxes and cluster_cpu or cluster_qm on system
  # with capture card using RSS (require cpu affinity tuning and system irq tuning)
  cluster-type: cluster_flow
  # In some fragmentation case, the hash can not be computed. If "defrag" is set
  # to yes, the kernel will do the needed defragmentation before sending the packets.
  defrag: yes
  # After Linux kernel 3.10 it is possible to activate the rollover option: if a socket is
  # full then kernel will send the packet on the next socket with room available. This option
  # can minimize packet drop and increase the treated bandwidth on single intensive flow.
  #rollover: yes
  # To use the ring feature of AF_PACKET, set 'use-mmap' to yes
  #use-mmap: yes
  # Lock memory map to avoid it goes to swap. Be careful that over suscribing could lock
  # your system
  #mmap-locked: yes
  # Use experimental tpacket_v3 capture mode, only active if use-mmap is true
  #tpacket-v3: yes
  # Ring size will be computed with respect to max_pending_packets and number
  # of threads. You can set manually the ring size in number of packets by setting
  # the following value. If you are using flow cluster-type and have really network
  # intensive single-flow you could want to set the ring-size independently of the number
  # of threads:
  #ring-size: 2048
  # Block size is used by tpacket_v3 only. It should set to a value high enough to contain
  # a decent number of packets. Size is in bytes so please consider your MTU. It should be
  # a power of 2 and it must be multiple of page size (usually 4096).
  #block-size: 32768
  # tpacket_v3 block timeout: an open block is passed to userspace if it is not
  # filled after block-timeout milliseconds.
  #block-timeout: 10
  # On busy system, this could help to set it to yes to recover from a packet drop
  # phase. This will result in some packets (at max a ring flush) being non treated.
  #use-emergency-flush: yes
  # recv buffer size, increase value could improve performance
  # buffer-size: 32768
  # Set to yes to disable promiscuous mode
  # disable-promisc: no
  # Choose checksum verification mode for the interface. At the moment
  # of the capture, some packets may be with an invalid checksum due to
  # offloading to the network card of the checksum computation.
  # Possible values are:
  #  - kernel: use indication sent by kernel for each packet (default)
  #  - yes: checksum validation is forced
  #  - no: checksum validation is disabled
```

```
    #  - auto: suricata uses a statistical approach to detect when
    #  checksum off-loading is used.
    # Warning: 'checksum-validation' must be set to yes to have any validation
    #checksum-checks: kernel
    # BPF filter to apply to this interface. The pcap filter syntax apply here.
    #bpf-filter: port 80 or udp
    # You can use the following variables to activate AF_PACKET tap or IPS mode.
    # If copy-mode is set to ips or tap, the traffic coming to the current
    # interface will be copied to the copy-iface interface. If 'tap' is set, the
    # copy is complete. If 'ips' is set, the packet matching a 'drop' action
    # will not be copied.
    #copy-mode: ips
    #copy-iface: eth1

  # Put default values here. These will be used for an interface that is not
  # in the list above.
  - interface: default
    #threads: auto
    #use-mmap: no
    #rollover: yes
    #tpacket-v3: yes

# Cross platform libpcap capture support
pcap:
  - interface: eth0
    # On Linux, pcap will try to use mmaped capture and will use buffer-size
    # as total of memory used by the ring. So set this to something bigger
    # than 1% of your bandwidth.
    #buffer-size: 16777216
    #bpf-filter: "tcp and port 25"
    # Choose checksum verification mode for the interface. At the moment
    # of the capture, some packets may be with an invalid checksum due to
    # offloading to the network card of the checksum computation.
    # Possible values are:
    #  - yes: checksum validation is forced
    #  - no: checksum validation is disabled
    #  - auto: suricata uses a statistical approach to detect when
    #  checksum off-loading is used. (default)
    # Warning: 'checksum-validation' must be set to yes to have any validation
    #checksum-checks: auto
    # With some accelerator cards using a modified libpcap (like myricom), you
    # may want to have the same number of capture threads as the number of capture
    # rings. In this case, set up the threads variable to N to start N threads
    # listening on the same interface.
    #threads: 16
    # set to no to disable promiscuous mode:
    #promisc: no
    # set snaplen, if not set it defaults to MTU if MTU can be known
    # via ioctl call and to full capture if not.
    #snaplen: 1518
  # Put default values here
  - interface: default
    #checksum-checks: auto

# Settings for reading pcap files
pcap-file:
```

```
  # Possible values are:
  #  - yes: checksum validation is forced
  #  - no: checksum validation is disabled
  #  - auto: suricata uses a statistical approach to detect when
  #  checksum off-loading is used. (default)
  # Warning: 'checksum-validation' must be set to yes to have checksum tested
  checksum-checks: auto

# See "Advanced Capture Options" below for more options, including NETMAP
# and PF_RING.


##
## Step 5: App Layer Protocol Configuration
##

# Configure the app-layer parsers. The protocols section details each
# protocol.
#
# The option "enabled" takes 3 values - "yes", "no", "detection-only".
# "yes" enables both detection and the parser, "no" disables both, and
# "detection-only" enables protocol detection only (parser disabled).
app-layer:
  protocols:
    tls:
      enabled: yes
      detection-ports:
        dp: 443

      # Completely stop processing TLS/SSL session after the handshake
      # completed. If bypass is enabled this will also trigger flow
      # bypass. If disabled (the default), TLS/SSL session is still
      # tracked for Heartbleed and other anomalies.
      #no-reassemble: yes
    dcerpc:
      enabled: yes
    ftp:
      enabled: yes
    ssh:
      enabled: yes
    smtp:
      enabled: yes
      # Configure SMTP-MIME Decoder
      mime:
        # Decode MIME messages from SMTP transactions
        # (may be resource intensive)
        # This field supercedes all others because it turns the entire
        # process on or off
        decode-mime: yes

        # Decode MIME entity bodies (ie. base64, quoted-printable, etc.)
        decode-base64: yes
        decode-quoted-printable: yes

        # Maximum bytes per header data value stored in the data structure
        # (default is 2000)
```

```
    header-value-depth: 2000

    # Extract URLs and save in state data structure
    extract-urls: yes
    # Set to yes to compute the md5 of the mail body. You will then
    # be able to journalize it.
    body-md5: no
  # Configure inspected-tracker for file_data keyword
  inspected-tracker:
    content-limit: 100000
    content-inspect-min-size: 32768
    content-inspect-window: 4096
imap:
  enabled: detection-only
msn:
  enabled: detection-only
smb:
  enabled: yes
  detection-ports:
    dp: 139
# smb2 detection is disabled internally inside the engine.
#smb2:
#  enabled: yes
dns:
  # memcaps. Globally and per flow/state.
  #global-memcap: 16mb
  #state-memcap: 512kb

  # How many unreplied DNS requests are considered a flood.
  # If the limit is reached, app-layer-event:dns.flooded; will match.
  #request-flood: 500

  tcp:
    enabled: yes
    detection-ports:
      dp: 53
  udp:
    enabled: yes
    detection-ports:
      dp: 53
http:
  enabled: yes
  # memcap: 64mb

  # default-config:           Used when no server-config matches
  #   personality:            List of personalities used by default
  #   request-body-limit:     Limit reassembly of request body for inspection
  #                           by http_client_body & pcre /P option.
  #   response-body-limit:    Limit reassembly of response body for inspection
  #                           by file_data, http_server_body & pcre /Q option.
  #   double-decode-path:     Double decode path section of the URI
  #   double-decode-query:    Double decode query section of the URI
  #   response-body-decompress-layer-limit:
  #                           Limit to how many layers of compression will be
  #                           decompressed. Defaults to 2.
  #
```

```
# server-config:            List of server configurations to use if address matches
#   address:                List of ip addresses or networks for this block
#   personalitiy:           List of personalities used by this block
#   request-body-limit:     Limit reassembly of request body for inspection
#                           by http_client_body & pcre /P option.
#   response-body-limit:    Limit reassembly of response body for inspection
#                           by file_data, http_server_body & pcre /Q option.
#   double-decode-path:     Double decode path section of the URI
#   double-decode-query:    Double decode query section of the URI
#
#   uri-include-all:        Include all parts of the URI. By default the
#                           'scheme', username/password, hostname and port
#                           are excluded. Setting this option to true adds
#                           all of them to the normalized uri as inspected
#                           by http_uri, urilen, pcre with /U and the other
#                           keywords that inspect the normalized uri.
#                           Note that this does not affect http_raw_uri.
#                           Also, note that including all was the default in
#                           1.4 and 2.0beta1.
#
#   meta-field-limit:       Hard size limit for request and response size
#                           limits. Applies to request line and headers,
#                           response line and headers. Does not apply to
#                           request or response bodies. Default is 18k.
#                           If this limit is reached an event is raised.
#
# Currently Available Personalities:
#   Minimal, Generic, IDS (default), IIS_4_0, IIS_5_0, IIS_5_1, IIS_6_0,
#   IIS_7_0, IIS_7_5, Apache_2
libhtp:
  default-config:
    personality: IDS

    # Can be specified in kb, mb, gb.  Just a number indicates
    # it's in bytes.
    request-body-limit: 100kb
    response-body-limit: 100kb

    # inspection limits
    request-body-minimal-inspect-size: 32kb
    request-body-inspect-window: 4kb
    response-body-minimal-inspect-size: 40kb
    response-body-inspect-window: 16kb

    # response body decompression (0 disables)
    response-body-decompress-layer-limit: 2

    # auto will use http-body-inline mode in IPS mode, yes or no set it statically
    http-body-inline: auto

    # Take a random value for inspection sizes around the specified value.
    # This lower the risk of some evasion technics but could lead
    # detection change between runs. It is set to 'yes' by default.
    #randomize-inspection-sizes: yes
    # If randomize-inspection-sizes is active, the value of various
    # inspection size will be choosen in the [1 - range%, 1 + range%]
```

```
      # range
      # Default value of randomize-inspection-range is 10.
      #randomize-inspection-range: 10

      # decoding
      double-decode-path: no
      double-decode-query: no

    server-config:

      #- apache:
      #    address: [192.168.1.0/24, 127.0.0.0/8, "::1"]
      #    personality: Apache_2
      #    # Can be specified in kb, mb, gb.  Just a number indicates
      #    # it's in bytes.
      #    request-body-limit: 4096
      #    response-body-limit: 4096
      #    double-decode-path: no
      #    double-decode-query: no

      #- iis7:
      #    address:
      #      - 192.168.0.0/24
      #      - 192.168.10.0/24
      #    personality: IIS_7_0
      #    # Can be specified in kb, mb, gb.  Just a number indicates
      #    # it's in bytes.
      #    request-body-limit: 4096
      #    response-body-limit: 4096
      #    double-decode-path: no
      #    double-decode-query: no

# Note: Modbus probe parser is minimalist due to the poor significant field
# Only Modbus message length (greater than Modbus header length)
# And Protocol ID (equal to 0) are checked in probing parser
# It is important to enable detection port and define Modbus port
# to avoid false positive
modbus:
  # How many unreplied Modbus requests are considered a flood.
  # If the limit is reached, app-layer-event:modbus.flooded; will match.
  #request-flood: 500

  enabled: no
  detection-ports:
    dp: 502
  # According to MODBUS Messaging on TCP/IP Implementation Guide V1.0b, it
  # is recommended to keep the TCP connection opened with a remote device
  # and not to open and close it for each MODBUS/TCP transaction. In that
  # case, it is important to set the depth of the stream reassembling as
  # unlimited (stream.reassembly.depth: 0)

  # Stream reassembly size for modbus. By default track it completely.
  stream-depth: 0

# DNP3
dnp3:
```

```
    enabled: no
    detection-ports:
      dp: 20000

  # SCADA EtherNet/IP and CIP protocol support
  enip:
    enabled: no
    detection-ports:
      dp: 44818
      sp: 44818

# Limit for the maximum number of asn1 frames to decode (default 256)
asn1-max-frames: 256


##########################################################################
#########
##
## Advanced settings below
##
##########################################################################
#########

##
## Run Options
##

# Run suricata as user and group.
#run-as:
#  user: suri
#  group: suri

# Some logging module will use that name in event as identifier. The default
# value is the hostname
#sensor-name: suricata

# Default pid file.
# Will use this file if no --pidfile in command options.
#pid-file: /var/run/suricata.pid

# Daemon working directory
# Suricata will change directory to this one if provided
# Default: "/"
#daemon-directory: "/"

# Suricata core dump configuration. Limits the size of the core dump file to
# approximately max-dump. The actual core dump size will be a multiple of the
# page size. Core dumps that would be larger than max-dump are truncated. On
# Linux, the actual core dump size may be a few pages larger than max-dump.
# Setting max-dump to 0 disables core dumping.
# Setting max-dump to 'unlimited' will give the full core dump file.
# On 32-bit Linux, a max-dump value >= ULONG_MAX may cause the core dump size
# to be 'unlimited'.

coredump:
  max-dump: unlimited
```

```
# If suricata box is a router for the sniffed networks, set it to 'router'. If
# it is a pure sniffing setup, set it to 'sniffer-only'.
# If set to auto, the variable is internally switch to 'router' in IPS mode
# and 'sniffer-only' in IDS mode.
# This feature is currently only used by the reject* keywords.
host-mode: auto

# Number of packets preallocated per thread. The default is 1024. A higher number
# will make sure each CPU will be more easily kept busy, but may negatively
# impact caching.
#
# If you are using the CUDA pattern matcher (mpm-algo: ac-cuda), different rules
# apply. In that case try something like 60000 or more. This is because the CUDA
# pattern matcher buffers and scans as many packets as possible in parallel.
#max-pending-packets: 1024

# Runmode the engine should use. Please check --list-runmodes to get the available
# runmodes for each packet acquisition method. Defaults to "autofp" (auto flow pinned
# load balancing).
#runmode: autofp

# Specifies the kind of flow load balancer used by the flow pinned autofp mode.
#
# Supported schedulers are:
#
# round-robin       - Flows assigned to threads in a round robin fashion.
# active-packets    - Flows assigned to threads that have the lowest number of
#                     unprocessed packets (default).
# hash              - Flow alloted usihng the address hash. More of a random
#                     technique. Was the default in Suricata 1.2.1 and older.
#
#autofp-scheduler: active-packets

# Preallocated size for packet. Default is 1514 which is the classical
# size for pcap on ethernet. You should adjust this value to the highest
# packet size (MTU + hardware header) on your system.
#default-packet-size: 1514

# Unix command socket can be used to pass commands to suricata.
# An external tool can then connect to get information from suricata
# or trigger some modifications of the engine. Set enabled to yes
# to activate the feature. In auto mode, the feature will only be
# activated in live capture mode. You can use the filename variable to set
# the file name of the socket.
unix-command:
  enabled: yes
  filename: /var/run/suricata-command.socket

# Magic file. The extension .mgc is added to the value here.
#magic-file: /usr/share/file/magic
#magic-file:

legacy:
  uricontent: enabled
```

```
##
## Detection settings
##

# Set the order of alerts bassed on actions
# The default order is pass, drop, reject, alert
# action-order:
#   - pass
#   - drop
#   - reject
#   - alert

# IP Reputation
reputation-categories-file: /etc/suricata/iprep/categories.txt
default-reputation-path: /etc/suricata/iprep
reputation-files:
 - reputation.list
 - uoc.list

# When run with the option --engine-analysis, the engine will read each of
# the parameters below, and print reports for each of the enabled sections
# and exit.  The reports are printed to a file in the default log dir
# given by the parameter "default-log-dir", with engine reporting
# subsection below printing reports in its own report file.
engine-analysis:
  # enables printing reports for fast-pattern for every rule.
  rules-fast-pattern: yes
  # enables printing reports for each rule
  rules: yes

#recursion and match limits for PCRE where supported
pcre:
  match-limit: 3500
  match-limit-recursion: 1500

##
## Advanced Traffic Tracking and Reconstruction Settings
##

# Host specific policies for defragmentation and TCP stream
# reassembly. The host OS lookup is done using a radix tree, just
# like a routing table so the most specific entry matches.
host-os-policy:
  # Make the default policy windows.
  windows: [0.0.0.0/0]
  bsd: []
  bsd-right: []
  old-linux: []
  linux: []
  old-solaris: []
  solaris: []
  hpux10: []
  hpux11: []
  irix: []
  macos: []
  vista: []
```

```
  windows2k3: []

# Defrag settings:

defrag:
  memcap: 32mb
  hash-size: 65536
  trackers: 65535 # number of defragmented flows to follow
  max-frags: 65535 # number of fragments to keep (higher than trackers)
  prealloc: yes
  timeout: 60

# Enable defrag per host settings
#  host-config:
#
#    - dmz:
#        timeout: 30
#        address: [192.168.1.0/24, 127.0.0.0/8, 1.1.1.0/24, 2.2.2.0/24, "1.1.1.1", "2.2.2.2",
"::1"]
#
#    - lan:
#        timeout: 45
#        address:
#          - 192.168.0.0/24
#          - 192.168.10.0/24
#          - 172.16.14.0/24

# Flow settings:
# By default, the reserved memory (memcap) for flows is 32MB. This is the limit
# for flow allocation inside the engine. You can change this value to allow
# more memory usage for flows.
# The hash-size determine the size of the hash used to identify flows inside
# the engine, and by default the value is 65536.
# At the startup, the engine can preallocate a number of flows, to get a better
# performance. The number of flows preallocated is 10000 by default.
# emergency-recovery is the percentage of flows that the engine need to
# prune before unsetting the emergency state. The emergency state is activated
# when the memcap limit is reached, allowing to create new flows, but
# prunning them with the emergency timeouts (they are defined below).
# If the memcap is reached, the engine will try to prune flows
# with the default timeouts. If it doens't find a flow to prune, it will set
# the emergency bit and it will try again with more agressive timeouts.
# If that doesn't work, then it will try to kill the last time seen flows
# not in use.
# The memcap can be specified in kb, mb, gb.  Just a number indicates it's
# in bytes.

flow:
  memcap: 128mb
  hash-size: 65536
  prealloc: 10000
  emergency-recovery: 30
  #managers: 1 # default to one flow manager
  #recyclers: 1 # default to one flow recycler thread

# This option controls the use of vlan ids in the flow (and defrag)
```

```
# hashing. Normally this should be enabled, but in some (broken)
# setups where both sides of a flow are not tagged with the same vlan
# tag, we can ignore the vlan id's in the flow hashing.
vlan:
  use-for-tracking: true

# Specific timeouts for flows. Here you can specify the timeouts that the
# active flows will wait to transit from the current state to another, on each
# protocol. The value of "new" determine the seconds to wait after a hanshake or
# stream startup before the engine free the data of that flow it doesn't
# change the state to established (usually if we don't receive more packets
# of that flow). The value of "established" is the amount of
# seconds that the engine will wait to free the flow if it spend that amount
# without receiving new packets or closing the connection. "closed" is the
# amount of time to wait after a flow is closed (usually zero). "bypassed"
# timeout controls locally bypassed flows. For these flows we don't do any other
# tracking. If no packets have been seen after this timeout, the flow is discarded.
#
# There's an emergency mode that will become active under attack circumstances,
# making the engine to check flow status faster. This configuration variables
# use the prefix "emergency-" and work similar as the normal ones.
# Some timeouts doesn't apply to all the protocols, like "closed", for udp and
# icmp.

flow-timeouts:

  default:
    new: 30
    established: 300
    closed: 0
    bypassed: 100
    emergency-new: 10
    emergency-established: 100
    emergency-closed: 0
    emergency-bypassed: 50
  tcp:
    new: 60
    established: 600
    closed: 60
    bypassed: 100
    emergency-new: 5
    emergency-established: 100
    emergency-closed: 10
    emergency-bypassed: 50
  udp:
    new: 30
    established: 300
    bypassed: 100
    emergency-new: 10
    emergency-established: 100
    emergency-bypassed: 50
  icmp:
    new: 30
    established: 300
    bypassed: 100
    emergency-new: 10
```

```
    emergency-established: 100
    emergency-bypassed: 50

# Stream engine settings. Here the TCP stream tracking and reassembly
# engine is configured.
#
# stream:
#   memcap: 32mb              # Can be specified in kb, mb, gb.  Just a
#                             # number indicates it's in bytes.
#   checksum-validation: yes    # To validate the checksum of received
#                             # packet. If csum validation is specified as
#                             # "yes", then packet with invalid csum will not
#                             # be processed by the engine stream/app layer.
#                             # Warning: locally generated trafic can be
#                             # generated without checksum due to hardware offload
#                             # of checksum. You can control the handling of checksum
#                             # on a per-interface basis via the 'checksum-checks'
#                             # option
#   prealloc-sessions: 2k       # 2k sessions prealloc'd per stream thread
#   midstream: false            # don't allow midstream session pickups
#   async-oneside: false        # don't enable async stream handling
#   inline: no                # stream inline mode
#   max-synack-queued: 5        # Max different SYN/ACKs to queue
#   bypass: no                # Bypass packets when stream.depth is reached
#
#   reassembly:
#     memcap: 64mb              # Can be specified in kb, mb, gb.  Just a number
#                             # indicates it's in bytes.
#     depth: 1mb                # Can be specified in kb, mb, gb.  Just a number
#                             # indicates it's in bytes.
#     toserver-chunk-size: 2560 # inspect raw stream in chunks of at least
#                             # this size.  Can be specified in kb, mb,
#                             # gb.  Just a number indicates it's in bytes.
#                             # The max acceptable size is 4024 bytes.
#     toclient-chunk-size: 2560 # inspect raw stream in chunks of at least
#                             # this size.  Can be specified in kb, mb,
#                             # gb.  Just a number indicates it's in bytes.
#                             # The max acceptable size is 4024 bytes.
#       randomize-chunk-size: yes # Take a random value for chunk size around the
specified value.
#                             # This lower the risk of some evasion technics but could lead
#                             # detection change between runs. It is set to 'yes' by default.
#       randomize-chunk-range: 10 # If randomize-chunk-size is active, the value of
chunk-size is
#                                       # a random value between (1 - randomize-chunk-
range/100)*toserver-chunk-size
#                               # and (1 + randomize-chunk-range/100)*toserver-chunk-size and
the same
#                             # calculation for toclient-chunk-size.
#                             # Default value of randomize-chunk-range is 10.
#
#     raw: yes                # 'Raw' reassembly enabled or disabled.
#                             # raw is for content inspection by detection
#                             # engine.
#
#     chunk-prealloc: 250       # Number of preallocated stream chunks. These
```

```
#                       # are used during stream inspection (raw).
#    segments:              # Settings for reassembly segment pool.
#      - size: 4            # Size of the (data)segment for a pool
#        prealloc: 256      # Number of segments to prealloc and keep
#                       # in the pool.
#    zero-copy-size: 128    # This option sets in bytes the value at
#                       # which segment data is passed to the app
#                       # layer API directly. Data sizes equal to
#                       # and higher than the value set are passed
#                       # on directly.
#
stream:
  memcap: 64mb
  checksum-validation: yes      # reject wrong csums
  inline: auto                  # auto will use inline mode in IPS mode, yes or no set it statically
  reassembly:
    memcap: 256mb
    depth: 1mb                  # reassemble 1mb into a stream
    toserver-chunk-size: 2560
    toclient-chunk-size: 2560
    randomize-chunk-size: yes
    #randomize-chunk-range: 10
    #raw: yes
    #chunk-prealloc: 250
    #segments:
    #  - size: 4
    #    prealloc: 256
    #  - size: 16
    #    prealloc: 512
    #  - size: 112
    #    prealloc: 512
    #  - size: 248
    #    prealloc: 512
    #  - size: 512
    #    prealloc: 512
    #  - size: 768
    #    prealloc: 1024
    #  - size: 1448
    #    prealloc: 1024
    #  - size: 65535
    #    prealloc: 128
    #zero-copy-size: 128

# Host table:
#
# Host table is used by tagging and per host thresholding subsystems.
#
host:
  hash-size: 4096
  prealloc: 1000
  memcap: 32mb

# IP Pair table:
#
# Used by xbits 'ippair' tracking.
```

```
#
#ippair:
#  hash-size: 4096
#  prealloc: 1000
#  memcap: 32mb


##
## Performance tuning and profiling
##

# The detection engine builds internal groups of signatures. The engine
# allow us to specify the profile to use for them, to manage memory on an
# efficient way keeping a good performance. For the profile keyword you
# can use the words "low", "medium", "high" or "custom". If you use custom
# make sure to define the values at "- custom-values" as your convenience.
# Usually you would prefer medium/high/low.
#
# "sgh mpm-context", indicates how the staging should allot mpm contexts for
# the signature groups.  "single" indicates the use of a single context for
# all the signature group heads.  "full" indicates a mpm-context for each
# group head.  "auto" lets the engine decide the distribution of contexts
# based on the information the engine gathers on the patterns from each
# group head.
#
# The option inspection-recursion-limit is used to limit the recursive calls
# in the content inspection code.  For certain payload-sig combinations, we
# might end up taking too much time in the content inspection code.
# If the argument specified is 0, the engine uses an internally defined
# default limit.  On not specifying a value, we use no limits on the recursion.
detect:
  profile: medium
  custom-values:
    toclient-groups: 3
    toserver-groups: 25
  sgh-mpm-context: auto
  inspection-recursion-limit: 3000
  # If set to yes, the loading of signatures will be made after the capture
  # is started. This will limit the downtime in IPS mode.
  #delayed-detect: yes

  prefilter:
    # default prefiltering setting. "mpm" only creates MPM/fast_pattern
    # engines. "auto" also sets up prefilter engines for other keywords.
    # Use --list-keywords=all to see which keywords support prefiltering.
    default: mpm

  # the grouping values above control how many groups are created per
  # direction. Port whitelisting forces that port to get it's own group.
  # Very common ports will benefit, as well as ports with many expensive
  # rules.
  grouping:
    #tcp-whitelist: 53, 80, 139, 443, 445, 1433, 3306, 3389, 6666, 6667, 8080
    #udp-whitelist: 53, 135, 5060

  profiling:
```

```
    # Log the rules that made it past the prefilter stage, per packet
    # default is off. The threshold setting determines how many rules
    # must have made it past pre-filter for that rule to trigger the
    # logging.
    #inspect-logging-threshold: 200
    grouping:
      dump-to-disk: false
      include-rules: false      # very verbose
      include-mpm-stats: false

# Select the multi pattern algorithm you want to run for scan/search the
# in the engine.
#
# The supported algorithms are:
# "ac"      - Aho-Corasick, default implementation
# "ac-bs"   - Aho-Corasick, reduced memory implementation
# "ac-cuda" - Aho-Corasick, CUDA implementation
# "ac-ks"   - Aho-Corasick, "Ken Steele" variant
# "hs"      - Hyperscan, available when built with Hyperscan support
#
# The default mpm-algo value of "auto" will use "hs" if Hyperscan is
# available, "ac" otherwise.
#
# The mpm you choose also decides the distribution of mpm contexts for
# signature groups, specified by the conf - "detect.sgh-mpm-context".
# Selecting "ac" as the mpm would require "detect.sgh-mpm-context"
# to be set to "single", because of ac's memory requirements, unless the
# ruleset is small enough to fit in one's memory, in which case one can
# use "full" with "ac".  Rest of the mpms can be run in "full" mode.
#
# There is also a CUDA pattern matcher (only available if Suricata was
# compiled with --enable-cuda: b2g_cuda. Make sure to update your
# max-pending-packets setting above as well if you use b2g_cuda.

mpm-algo: auto

# Select the matching algorithm you want to use for single-pattern searches.
#
# Supported algorithms are "bm" (Boyer-Moore) and "hs" (Hyperscan, only
# available if Suricata has been built with Hyperscan support).
#
# The default of "auto" will use "hs" if available, otherwise "bm".

spm-algo: auto

# Suricata is multi-threaded. Here the threading can be influenced.
threading:
  set-cpu-affinity: no
  # Tune cpu affinity of threads. Each family of threads can be bound
  # on specific CPUs.
  #
  # These 2 apply to the all runmodes:
  # management-cpu-set is used for flow timeout handling, counters
  # worker-cpu-set is used for 'worker' threads
  #
  # Additionally, for autofp these apply:
```

```
  # receive-cpu-set is used for capture threads
  # verdict-cpu-set is used for IPS verdict threads
  #
  cpu-affinity:
    - management-cpu-set:
        cpu: [ 0 ]  # include only these cpus in affinity settings
    - receive-cpu-set:
        cpu: [ 0 ]  # include only these cpus in affinity settings
    - worker-cpu-set:
        cpu: [ "all" ]
        mode: "exclusive"
        # Use explicitely 3 threads and don't compute number by using
        # detect-thread-ratio variable:
        # threads: 3
        prio:
          low: [ 0 ]
          medium: [ "1-2" ]
          high: [ 3 ]
          default: "medium"
    #- verdict-cpu-set:
    #    cpu: [ 0 ]
    #    prio:
    #      default: "high"
  #
  # By default Suricata creates one "detect" thread per available CPU/CPU core.
  # This setting allows controlling this behaviour. A ratio setting of 2 will
  # create 2 detect threads for each CPU/CPU core. So for a dual core CPU this
  # will result in 4 detect threads. If values below 1 are used, less threads
  # are created. So on a dual core CPU a setting of 0.5 results in 1 detect
  # thread being created. Regardless of the setting at a minimum 1 detect
  # thread will always be created.
  #
  detect-thread-ratio: 1.0

# Luajit has a strange memory requirement, it's 'states' need to be in the
# first 2G of the process' memory.
#
# 'luajit.states' is used to control how many states are preallocated.
# State use: per detect script: 1 per detect thread. Per output script: 1 per
# script.
luajit:
  states: 128

# Profiling settings. Only effective if Suricata has been built with the
# the --enable-profiling configure flag.
#
profiling:
  # Run profiling for every xth packet. The default is 1, which means we
  # profile every packet. If set to 1000, one packet is profiled for every
  # 1000 received.
  #sample-rate: 1000

  # rule profiling
  rules:

    # Profiling can be disabled here, but it will still have a
```

```
    # performance impact if compiled in.
    enabled: yes
    filename: rule_perf.log
    append: yes

    # Sort options: ticks, avgticks, checks, matches, maxticks
    sort: avgticks

    # Limit the number of items printed at exit (ignored for json).
    limit: 100

    # output to json
    json: yes

  # per keyword profiling
  keywords:
    enabled: yes
    filename: keyword_perf.log
    append: yes

  # per rulegroup profiling
  rulegroups:
    enabled: yes
    filename: rule_group_perf.log
    append: yes

  # packet profiling
  packets:

    # Profiling can be disabled here, but it will still have a
    # performance impact if compiled in.
    enabled: yes
    filename: packet_stats.log
    append: yes

    # per packet csv output
    csv:

      # Output can be disabled here, but it will still have a
      # performance impact if compiled in.
      enabled: no
      filename: packet_stats.csv

  # profiling of locking. Only available when Suricata was built with
  # --enable-profiling-locks.
  locks:
    enabled: no
    filename: lock_stats.log
    append: yes

  pcap-log:
    enabled: no
    filename: pcaplog_stats.log
    append: yes

##
```

```
## Netfilter integration
##

# When running in NFQ inline mode, it is possible to use a simulated
# non-terminal NFQUEUE verdict.
# This permit to do send all needed packet to suricata via this a rule:
#       iptables -I FORWARD -m mark ! --mark $MARK/$MASK -j NFQUEUE
# And below, you can have your standard filtering ruleset. To activate
# this mode, you need to set mode to 'repeat'
# If you want packet to be sent to another queue after an ACCEPT decision
# set mode to 'route' and set next-queue value.
# On linux >= 3.1, you can set batchcount to a value > 1 to improve performance
# by processing several packets before sending a verdict (worker runmode only).
# On linux >= 3.6, you can set the fail-open option to yes to have the kernel
# accept the packet if suricata is not able to keep pace.
# bypass mark and mask can be used to implement NFQ bypass. If bypass mark is
# set then the NFQ bypass is activated. Suricata will set the bypass mark/mask
# on packet of a flow that need to be bypassed. The Nefilter ruleset has to
# directly accept all packets of a flow once a packet has been marked.
nfq:
#  mode: accept
#  repeat-mark: 1
#  repeat-mask: 1
#  bypass-mark: 1
#  bypass-mask: 1
#  route-queue: 2
#  batchcount: 20
#  fail-open: yes

#nflog support
nflog:
   # netlink multicast group
   # (the same as the iptables --nflog-group param)
   # Group 0 is used by the kernel, so you can't use it
  - group: 2
   # netlink buffer size
   buffer-size: 18432
   # put default value here
  - group: default
   # set number of packet to queue inside kernel
   qthreshold: 1
   # set the delay before flushing packet in the queue inside kernel
   qtimeout: 100
   # netlink max buffer size
   max-size: 20000

##
## Advanced Capture Options
##

# general settings affecting packet capture
capture:
  # disable NIC offloading. It's restored when Suricata exists.
  # Enabled by default
  #disable-offloading: false
  #
```

```
  # disable checksum validation. Same as setting '-k none' on the
  # commandline
  #checksum-validation: none

# Netmap support
#
# Netmap operates with NIC directly in driver, so you need FreeBSD wich have
# built-in netmap support or compile and install netmap module and appropriate
# NIC driver on your Linux system.
# To reach maximum throughput disable all receive-, segmentation-,
# checksum- offloadings on NIC.
# Disabling Tx checksum offloading is *required* for connecting OS endpoint
# with NIC endpoint.
# You can find more information at https://github.com/luigirizzo/netmap
#
netmap:
   # To specify OS endpoint add plus sign at the end (e.g. "eth0+")
 - interface: eth2
   # Number of receive threads. "auto" uses number of RSS queues on interface.
   #threads: auto
   # You can use the following variables to activate netmap tap or IPS mode.
   # If copy-mode is set to ips or tap, the traffic coming to the current
   # interface will be copied to the copy-iface interface. If 'tap' is set, the
   # copy is complete. If 'ips' is set, the packet matching a 'drop' action
   # will not be copied.
   # To specify the OS as the copy-iface (so the OS can route packets, or forward
   # to a service running on the same machine) add a plus sign at the end
   # (e.g. "copy-iface: eth0+"). Don't forget to set up a symmetrical eth0+ -> eth0
   # for return packets. Hardware checksumming must be *off* on the interface if
   # using an OS endpoint (e.g. 'ifconfig eth0 -rxcsum -txcsum -rxcsum6 -txcsum6' for
FreeBSD
   # or 'ethtool -K eth0 tx off rx off' for Linux).
   #copy-mode: tap
   #copy-iface: eth3
   # Set to yes to disable promiscuous mode
   # disable-promisc: no
   # Choose checksum verification mode for the interface. At the moment
   # of the capture, some packets may be with an invalid checksum due to
   # offloading to the network card of the checksum computation.
   # Possible values are:
   #  - yes: checksum validation is forced
   #  - no: checksum validation is disabled
   #  - auto: suricata uses a statistical approach to detect when
   #  checksum off-loading is used.
   # Warning: 'checksum-validation' must be set to yes to have any validation
   #checksum-checks: auto
   # BPF filter to apply to this interface. The pcap filter syntax apply here.
   #bpf-filter: port 80 or udp
 #- interface: eth3
   #threads: auto
   #copy-mode: tap
   #copy-iface: eth2
   # Put default values here
 - interface: default

# PF_RING configuration. for use with native PF_RING support
```

```
# for more info see http://www.ntop.org/products/pf_ring/
pfring:
  - interface: eth0
    # Number of receive threads (>1 will enable experimental flow pinned
    # runmode)
    threads: 1

    # Default clusterid.  PF_RING will load balance packets based on flow.
    # All threads/processes that will participate need to have the same
    # clusterid.
    cluster-id: 99

    # Default PF_RING cluster type. PF_RING can load balance per flow.
    # Possible values are cluster_flow or cluster_round_robin.
    cluster-type: cluster_flow
    # bpf filter for this interface
    #bpf-filter: tcp
    # Choose checksum verification mode for the interface. At the moment
    # of the capture, some packets may be with an invalid checksum due to
    # offloading to the network card of the checksum computation.
    # Possible values are:
    #  - rxonly: only compute checksum for packets received by network card.
    #  - yes: checksum validation is forced
    #  - no: checksum validation is disabled
    #  - auto: suricata uses a statistical approach to detect when
    #  checksum off-loading is used. (default)
    # Warning: 'checksum-validation' must be set to yes to have any validation
    #checksum-checks: auto
  # Second interface
  #- interface: eth1
  #  threads: 3
  #  cluster-id: 93
  #  cluster-type: cluster_flow
  # Put default values here
  - interface: default
    #threads: 2

# For FreeBSD ipfw(8) divert(4) support.
# Please make sure you have ipfw_load="YES" and ipdivert_load="YES"
# in /etc/loader.conf or kldload'ing the appropriate kernel modules.
# Additionally, you need to have an ipfw rule for the engine to see
# the packets from ipfw.  For Example:
#
#   ipfw add 100 divert 8000 ip from any to any
#
# The 8000 above should be the same number you passed on the command
# line, i.e. -d 8000
#
ipfw:

  # Reinject packets at the specified ipfw rule number.  This config
  # option is the ipfw rule number AT WHICH rule processing continues
  # in the ipfw processing system after the engine has finished
  # inspecting the packet for acceptance.  If no rule number is specified,
  # accepted packets are reinjected at the divert rule which they entered
  # and IPFW rule processing continues.  No check is done to verify
```

```
  # this will rule makes sense so care must be taken to avoid loops in ipfw.
  #
  ## The following example tells the engine to reinject packets
  # back into the ipfw firewall AT rule number 5500:
  #
  # ipfw-reinjection-rule-number: 5500


napatech:
    # The Host Buffer Allowance for all streams
    # (-1 = OFF, 1 - 100 = percentage of the host buffer that can be held back)
    hba: -1

    # use_all_streams set to "yes" will query the Napatech service for all configured
    # streams and listen on all of them. When set to "no" the streams config array
    # will be used.
    use-all-streams: yes

    # The streams to listen on
    streams: [1, 2, 3]

# Tilera mpipe configuration. for use on Tilera TILE-Gx.
mpipe:

  # Load balancing modes: "static", "dynamic", "sticky", or "round-robin".
  load-balance: dynamic

  # Number of Packets in each ingress packet queue. Must be 128, 512, 2028 or 65536
  iqueue-packets: 2048

  # List of interfaces we will listen on.
  inputs:
  - interface: xgbe2
  - interface: xgbe3
  - interface: xgbe4


  # Relative weight of memory for packets of each mPipe buffer size.
  stack:
    size128: 0
    size256: 9
    size512: 0
    size1024: 0
    size1664: 7
    size4096: 0
    size10386: 0
    size16384: 0

##
## Hardware accelaration
##

# Cuda configuration.
cuda:
  # The "mpm" profile.  On not specifying any of these parameters, the engine's
  # internal default values are used, which are same as the ones specified in
```

```
  # in the default conf file.
  mpm:
    # The minimum length required to buffer data to the gpu.
    # Anything below this is MPM'ed on the CPU.
    # Can be specified in kb, mb, gb.  Just a number indicates it's in bytes.
    # A value of 0 indicates there's no limit.
    data-buffer-size-min-limit: 0
    # The maximum length for data that we would buffer to the gpu.
    # Anything over this is MPM'ed on the CPU.
    # Can be specified in kb, mb, gb.  Just a number indicates it's in bytes.
    data-buffer-size-max-limit: 1500
    # The ring buffer size used by the CudaBuffer API to buffer data.
    cudabuffer-buffer-size: 500mb
    # The max chunk size that can be sent to the gpu in a single go.
    gpu-transfer-size: 50mb
    # The timeout limit for batching of packets in microseconds.
    batching-timeout: 2000
    # The device to use for the mpm.  Currently we don't support load balancing
    # on multiple gpus.  In case you have multiple devices on your system, you
    # can specify the device to use, using this conf.  By default we hold 0, to
    # specify the first device cuda sees.  To find out device-id associated with
    # the card(s) on the system run "suricata --list-cuda-cards".
    device-id: 0
    # No of Cuda streams used for asynchronous processing. All values > 0 are valid.
    # For this option you need a device with Compute Capability > 1.0.
    cuda-streams: 2

##
## Include other configs
##

# Includes.  Files included here will be handled as if they were
# inlined in this configuration file.
#include: include1.yaml
#include: include2.yaml
```

*7.3. Configuración Oinkmaster.*

```
# This is a Debian specific config file for oinkmaster crafted for suricata,
# you should read oinkmaster documentation to modify this file.
# This config is loaded by default from the suricata-oinkmaster-updater binary
# which is called daily from a cronjob by default

skipfile local.rules
skipfile deleted.rules
skipfile snort.conf
use_external_bins = 0

url = https://rules.emergingthreats.net/open/suricata-3.0/emerging.rules.tar.gz
```

*7.4. Configuración Clasificación de Alertas [Por defecto].*

```
# $Id$
# classification.config taken from Snort 2.8.5.3. Snort is governed by the GPLv2
#
# The following includes information for prioritizing rules
#
# Each classification includes a shortname, a description, and a default
# priority for that classification.
#
# This allows alerts to be classified and prioritized.  You can specify
# what priority each classification has.  Any rule can override the default
# priority for that rule.
#
# Here are a few example rules:
#
#   alert TCP any any -> any 80 (msg: "EXPLOIT ntpdx overflow";
#         dsize: > 128; classtype:attempted-admin; priority:10;
#
#   alert TCP any any -> any 25 (msg:"SMTP expn root"; flags:A+; \
#               content:"expn root"; nocase; classtype:attempted-recon;)
#
# The first rule will set its type to "attempted-admin" and override
# the default priority for that type to 10.
#
# The second rule set its type to "attempted-recon" and set its
# priority to the default for that type.
#

#
# config classification:shortname,short description,priority
#

config classification: not-suspicious,Not Suspicious Traffic,3
config classification: unknown,Unknown Traffic,3
config classification: bad-unknown,Potentially Bad Traffic, 2
config classification: attempted-recon,Attempted Information Leak,2
config classification: successful-recon-limited,Information Leak,2
config classification: successful-recon-largescale,Large Scale Information Leak,2
config classification: attempted-dos,Attempted Denial of Service,2
config classification: successful-dos,Denial of Service,2
config classification: attempted-user,Attempted User Privilege Gain,1
config classification: unsuccessful-user,Unsuccessful User Privilege Gain,1
config classification: successful-user,Successful User Privilege Gain,1
config classification: attempted-admin,Attempted Administrator Privilege Gain,1
config classification: successful-admin,Successful Administrator Privilege Gain,1


# NEW CLASSIFICATIONS
config classification: rpc-portmap-decode,Decode of an RPC Query,2
config classification: shellcode-detect,Executable code was detected,1
config classification: string-detect,A suspicious string was detected,3
config classification: suspicious-filename-detect,A suspicious filename was detected,2
config classification: suspicious-login,An attempted login using a suspicious username
was detected,2
config classification: system-call-detect,A system call was detected,2
```

```
config classification: tcp-connection,A TCP connection was detected,4
config classification: trojan-activity,A Network Trojan was detected, 1
config classification: unusual-client-port-connection,A client was using an unusual port,2
config classification: network-scan,Detection of a Network Scan,3
config classification: denial-of-service,Detection of a Denial of Service Attack,2
config classification: non-standard-protocol,Detection of a non-standard protocol or event,2
config classification: protocol-command-decode,Generic Protocol Command Decode,3
config classification: web-application-activity,access to a potentially vulnerable web application,2
config classification: web-application-attack,Web Application Attack,1
config classification: misc-activity,Misc activity,3
config classification: misc-attack,Misc Attack,2
config classification: icmp-event,Generic ICMP event,3
config classification: kickass-porn,SCORE! Get the lotion!,1
config classification: policy-violation,Potential Corporate Privacy Violation,1
config classification: default-login-attempt,Attempt to login by a default username and password,2
```

*7.5. Script BASH para sincronizar fichero eve.json al SIEM.*

```
#!/bin/bash

#"Sincronizar NIDS con SIEM"

rsync -avzhe ssh /var/log/suricata/eve.json  uoc@192.168.122.48:/home/uoc/SuriLog
```

*7.6. Reglas personalizadas.*

```
# Reglas personalizadas UOC-TFM Seguridad Empresarial
#
#
#
#
#Regla que detecta Ping
alert icmp any any -> any any (msg: "UOC ICMP detected"; classtype:uoc-tfm; sid:3300001; rev:1;)
#Regla que detecta conexiones SSH utilizando el peurto 22
alert tcp any any -> any 22 (msg:"UOC Conexion via SSH";classtype:uoc-tfm; sid:3300002; rev:1;)
#Regla que detecta conexiones SSH utilizando un puerto diferente al 22
#alert tcp any any -> any !22 (msg:"UOC Conexion via SSH Utilizando otro puerto"; classtype:uoc-tfm; app-layer-protocol:ssh; sid:3300003; rev:1;)
#Regla que detecta conexiones RDP
alert tcp any any -> any 3389 (msg:"UOC RDP -Solicitud Conexion "; flow: to_server,established; content:"|03|"; offset: 0; depth: 1; content:"|E0|"; offset: 5; depth: 1; classtype:uoc-tfm; sid:3300003; rev:1;)
alert tcp any 3389 -> any any (msg:"UOC RDP -Conexion Confirmada"; flow: from_server,established; content:"|03|"; offset: 0; depth: 1; content:"|D0|"; offset: 5; depth: 1; classtype:uoc-tfm; sid:3300004; rev:1;)
#Regla que detecta Escaneo de la red en base a busqueda de puertos
```

```
alert tcp $HOME_NET any -> $HOME_NET any (msg:"UOC Scan NMAP Interno"; flow:to_server,established; content:"|20|Nmap"; http_user_agent; fast_pattern; classtype:uoc-tfm; sid:3300005; rev:1;)
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"UOC Scan NMAP hacia Internet"; flow:to_server,established; content:"|20|Nmap"; http_user_agent; fast_pattern; classtype:uoc-tfm; sid:3300006; rev:1;)
```

*7.7. Configuración Red SIEM [Ubuntu 18.04].*

```
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp1s0:
      dhcp4: true
  version: 2
~
```

*7.8. Configuración Elasticsearch.*

```
#                  ========================                  Elasticsearch                  Configuration
=========================
#
# NOTE: Elasticsearch comes with reasonable defaults for most settings.
#       Before you set out to tweak and tune the configuration, make sure you
#       understand what are you trying to accomplish and the consequences.
#
# The primary way of configuring a node is via this file. This template lists
# the most important settings you may want to configure for a production cluster.
#
# Please consult the documentation for further information on configuration options:
# https://www.elastic.co/guide/en/elasticsearch/reference/index.html
#
# ---------------------------------- Cluster -----------------------------------
#
# Use a descriptive name for your cluster:
#
cluster.name: UOC-TFM
#
# ------------------------------------ Node ------------------------------------
#
# Use a descriptive name for the node:
#
node.name: uoc-node
#
# Add custom attributes to the node:
#
#node.attr.rack: r1
#
# ----------------------------------- Paths ------------------------------------
#
# Path to directory where to store the data (separate multiple locations by comma):
#
path.data: /var/lib/elasticsearch
#
```

```
# Path to log files:
#
path.logs: /var/log/elasticsearch
#
# ---------------------------------- Memory -----------------------------------
#
# Lock the memory on startup:
#
#bootstrap.memory_lock: true
#
# Make sure that the heap size is set to about half the memory available
# on the system and that the owner of the process is allowed to use this
# limit.
#
# Elasticsearch performs poorly when the system is swapping the memory.
#
# ---------------------------------- Network -----------------------------------
#
# By default Elasticsearch is only accessible on localhost. Set a different
# address here to expose this node on the network:
#
#network.host: 192.168.0.1
network.host: 0.0.0.0
#
# By default Elasticsearch listens for HTTP traffic on the first free port it
# finds starting at 9200. Set a specific HTTP port here:
#
http.port: 9200
#
# For more information, consult the network module documentation.
#
# --------------------------------- Discovery ----------------------------------
#
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
#discovery.seed_hosts: ["host1", "host2"]
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
#cluster.initial_master_nodes: ["node-1", "node-2"]
#
# For more information, consult the discovery and cluster formation module documentation.
#
# ---------------------------------- Various -----------------------------------
#
# Require explicit names when deleting indices:
#
#action.destructive_requires_name: true
#
#
discovery.type: single-node

#xpack.security.enabled: true
```

*7.9. Configuración Kibana.*

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

# Specifies the address to which the Kibana server will bind. IP addresses and host names are both valid values.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: "0.0.0.0"

# Enables you to specify a path to mount Kibana at if you are running behind a proxy.
# Use the `server.rewriteBasePath` setting to tell Kibana if it should remove the basePath
# from requests it receives, and to prevent a deprecation warning at startup.
# This setting cannot end in a slash.
#server.basePath: ""

# Specifies whether Kibana should rewrite requests that are prefixed with
# `server.basePath` or require that they are rewritten by your reverse proxy.
# This setting was effectively always `false` before Kibana 6.3 and will
# default to `true` starting in Kibana 7.0.
#server.rewriteBasePath: false

# Specifies the public URL at which Kibana is available for end users. If
# `server.basePath` is configured this URL should end with the same basePath.
#server.publicBaseUrl: ""

# The maximum payload size in bytes for incoming server requests.
#server.maxPayloadBytes: 1048576

# The Kibana server's name.  This is used for display purposes.
server.name: "UOC-TFM"

# The URLs of the Elasticsearch instances to use for all your queries.
elasticsearch.hosts: ["http://localhost:9200"]

# Kibana uses an index in Elasticsearch to store saved searches, visualizations and
# dashboards. Kibana creates a new index if the index doesn't already exist.
#kibana.index: ".kibana"

# The default application to load.
#kibana.defaultAppId: "home"

# If your Elasticsearch is protected with basic authentication, these settings provide
# the username and password that the Kibana server uses to perform maintenance on the Kibana
# index at startup. Your Kibana users still need to authenticate with Elasticsearch, which
# is proxied through the Kibana server.
#elasticsearch.username: "kibana_system"
#elasticsearch.password: "pass"
```

*7.10. Configuración Filebeat.*

```
######################## Filebeat Configuration Example #########################

# This file is an example configuration file highlighting only the most common
# options. The filebeat.reference.yml file from the same directory contains all the
# supported options with more comments. You can use it as a reference.
#
# You can find the full configuration reference here:
# https://www.elastic.co/guide/en/beats/filebeat/index.html

# For more available modules and options, please see the filebeat.reference.yml sample
# configuration file.

# ============================== Filebeat inputs ===============================

filebeat.inputs:

# Each - is an input. Most options can be set at the input level, so
# you can use different inputs for various configurations.
# Below are the input specific configurations.

- type: log

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

  # Paths that should be crawled and fetched. Glob based paths.
  paths:
    - /var/log/*.log
    #- c:\programdata\elasticsearch\logs\*

  # Exclude lines. A list of regular expressions to match. It drops the lines that are
  # matching any regular expression from the list.
  #exclude_lines: ['^DBG']

  # Include lines. A list of regular expressions to match. It exports the lines that are
  # matching any regular expression from the list.
  #include_lines: ['^ERR', '^WARN']

  # Exclude files. A list of regular expressions to match. Filebeat drops the files that
  # are matching any regular expression from the list. By default, no files are dropped.
  #exclude_files: ['.gz$']

  # Optional additional fields. These fields can be freely picked
  # to add additional information to the crawled log files for filtering
  #fields:
  #  level: debug
  #  review: 1
```

```
  ### Multiline options

  # Multiline can be used for log messages spanning multiple lines. This is common
  # for Java Stack Traces or C-Line Continuation

  # The regexp Pattern that has to be matched. The example pattern matches all lines starting with [
  #multiline.pattern: ^\[

  # Defines if the pattern set under pattern should be negated or not. Default is false.
  #multiline.negate: false

  # Match can be set to "after" or "before". It is used to define if lines should be append to a pattern
  # that was (not) matched before or after or as long as a pattern is not matched based on negate.
  # Note: After is the equivalent to previous and before is the equivalent to to next in Logstash
  #multiline.match: after

# filestream is an experimental input. It is going to replace log input in the future.
- type: filestream

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

  # Paths that should be crawled and fetched. Glob based paths.
  paths:
    - /var/log/*.log
    #- c:\programdata\elasticsearch\logs\*

  # Exclude lines. A list of regular expressions to match. It drops the lines that are
  # matching any regular expression from the list.
  #exclude_lines: ['^DBG']

  # Include lines. A list of regular expressions to match. It exports the lines that are
  # matching any regular expression from the list.
  #include_lines: ['^ERR', '^WARN']

  # Exclude files. A list of regular expressions to match. Filebeat drops the files that
  # are matching any regular expression from the list. By default, no files are dropped.
  #prospector.scanner.exclude_files: ['.gz$']

  # Optional additional fields. These fields can be freely picked
  # to add additional information to the crawled log files for filtering
  #fields:
  #  level: debug
  #  review: 1

# ============================== Filebeat modules ==============================

filebeat.config.modules:
  # Glob pattern for configuration loading
  path: ${path.config}/modules.d/*.yml
```

```
  # Set to true to enable config reloading
  reload.enabled: false

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

# ========================== Elasticsearch template setting ==========================

setup.template.settings:
  index.number_of_shards: 1
  #index.codec: best_compression
  #_source.enabled: false


# ================================== General ===================================

# The name of the shipper that publishes the network data. It can be used to group
# all the transactions sent by a single shipper in the web interface.
#name:

# The tags of the shipper are included in their own field with each
# transaction published.
#tags: ["service-X", "web-tier"]

# Optional fields that you can specify to add additional information to the
# output.
#fields:
#  env: staging
# ================================== Dashboards ================================
# These settings control loading the sample dashboards to the Kibana index. Loading
# the dashboards is disabled by default and can be enabled either by setting the
# options here or by using the `setup` command.
#setup.dashboards.enabled: false

# The URL from where to download the dashboards archive. By default this URL
# has a value which is computed based on the Beat name and version. For released
# versions, this URL points to the dashboard archive on the artifacts.elastic.co
# website.
#setup.dashboards.url:

# ================================== Kibana ====================================

# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.
# This requires a Kibana endpoint configuration.
setup.kibana:
        host: "http://localhost:5601"

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

```
  #host: "localhost:5601"

  # Kibana Space ID
  # ID of the Kibana Space into which the dashboards should be loaded. By default,
  # the Default Space will be used.
  #space.id:

# ============================== Elastic Cloud ==============================

# These settings simplify using Filebeat with the Elastic Cloud (https://cloud.elastic.co/).

# The cloud.id setting overwrites the `output.elasticsearch.hosts` and
# `setup.kibana.host` options.
# You can find the `cloud.id` in the Elastic Cloud web UI.
#cloud.id:

# The cloud.auth setting overwrites the `output.elasticsearch.username` and
# `output.elasticsearch.password` settings. The format is `<user>:<pass>`.
#cloud.auth:

# ================================== Outputs ===================================

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

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

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

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

# ------------------------------ Logstash Output -------------------------------
#output.logstash:
  # The Logstash hosts
  #hosts: ["localhost:5044"]

  # Optional SSL. By default is off.
  # List of root certificates for HTTPS server verifications
  #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]

  # Certificate for SSL client authentication
  #ssl.certificate: "/etc/pki/client/cert.pem"
# ================================== Processors ===================================
processors:
  - add_host_metadata:
      when.not.contains.tags: forwarded
  - add_cloud_metadata: ~
```

```
  - add_docker_metadata: ~
  - add_kubernetes_metadata: ~

#                   ================================== Logging
==================================

# Sets log level. The default log level is info.
# Available log levels are: error, warning, info, debug
#logging.level: debug

# At debug level, you can selectively enable logging only for some components.
# To enable all selectors use ["*"]. Examples of other selectors are "beat",
# "publisher", "service".
#logging.selectors: ["*"]

#           ============================== X-Pack Monitoring
==============================
# Filebeat can export internal metrics to a central Elasticsearch monitoring
# cluster.  This requires xpack monitoring to be enabled in Elasticsearch.  The
# reporting is disabled by default.

# Set to true to enable the monitoring reporter.
#monitoring.enabled: false

# Sets the UUID of the Elasticsearch cluster under which monitoring data for this
# Filebeat instance will appear in the Stack Monitoring UI. If output.elasticsearch
# is enabled, the UUID is derived from the Elasticsearch cluster referenced by
output.elasticsearch.
#monitoring.cluster_uuid:

# Uncomment to send the metrics to Elasticsearch. Most settings from the
# Elasticsearch output are accepted here as well.
# Note that the settings should point to your Elasticsearch *monitoring* cluster.
# Any setting that is not set is automatically inherited from the Elasticsearch
# output configuration, so if you have the Elasticsearch output configured such
# that it is pointing to your Elasticsearch monitoring cluster, you can simply
# uncomment the following line.
#monitoring.elasticsearch:

#                   ============================== Instrumentation
==============================

# Instrumentation support for the filebeat.
#instrumentation:
    # Set to true to enable instrumentation of filebeat.
    #enabled: false

    # Environment in which filebeat is running on (eg: staging, production, etc.)
    #environment: ""

    # APM Server hosts to report instrumentation results to.
    #hosts:
    #  - http://localhost:8200

    # API Key for the APM Server(s).
    # If api_key is set then secret_token will be ignored.
```

```
   #api_key:

   # Secret token for the APM Server(s).
   #secret_token:


#                    ================================                    Migration
==================================

# This allows to enable 6.7 migration aliases
#migration.6_to_7.enabled: true.
```

*7.11. Configuración Modulo Filebeat-Suricata.*

```
# Module: suricata
# Docs: https://www.elastic.co/guide/en/beats/filebeat/7.x/filebeat-module-suricata.html

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

    # Set custom paths for the log files. If left empty,
    # Filebeat will choose the paths depending on your OS.
    #var.paths:
~
```

*7.12. Fichero configuración NGINX*

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
        worker_connections 768;
        # multi_accept on;
}

http {

        ##
        # Basic Settings
        ##

        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;
```

```
        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        ##
        # SSL Settings
        ##

        ssl_protocols TLSv1 TLSv1.1 TLSv1.2; # Dropping SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        ##
        # Logging Settings
        ##

        access_log /var/log/nginx/access.log;
        error_log /var/log/nginx/error.log;

        ##
        # Gzip Settings
        ##

        gzip on;

        # gzip_vary on;
        # gzip_proxied any;
        # gzip_comp_level 6;
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types text/plain text/css application/json application/javascript text/xml
application/xml application/xml+rss text/javascript;

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}


#mail {
#       # See sample authentication script at:
#       # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript
#
#       # auth_http localhost/auth.php;
#       # pop3_capabilities "TOP" "USER";
#       # imap_capabilities "IMAP4rev1" "UIDPLUS";
#
#       server {
#               listen     localhost:110;
#               protocol   pop3;
#               proxy      on;
#       }
#
#       server {
```

```
#                   listen     localhost:143;
#                   protocol   imap;
#                   proxy      on;
#         }
#}
```

*7.13. Fichero configuracion Site-Available.*

```
server {
    listen 80;

    server_name uoc.local;

    auth_basic "Restricted Access";
    auth_basic_user_file /etc/nginx/htpasswd.users;

    location / {
        proxy_pass http://localhost:5601;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

*7.14. Fichero configuración htpasswd.users.*

```
kibanaadmin:$apr1$4ScWMpl4$ekp6QKHn2z.OjqX8RlZ2L1
```

*7.15. Script para sincronizar ficheros eve.json.*

```
#!/bin/bash
#Sincronizar ficheros

cat /home/uoc/SuriLog/*.json > /var/log/suricata/eve.json
~
```

*7.16. Configuración Switch CISCO Catalyst.*

```
UOC-TFM>
UOC-TFM>C2950 Boot Loader (C2950-HBO
        ^
% Invalid input detected at '^' marker.

UOC-TFM>en
Password:
Password:
Password:
UOC-TFM#sh
UOC-TFM#show runn
UOC-TFM#show running-config
Building configuration...
```

```
Current configuration : 1322 bytes
!
version 12.1
no service pad
service timestamps debug uptime
service timestamps log uptime
service password-encryption
!
hostname UOC-TFM
!
enable secret 5 $1$KBrM$DyZnYTfeUZL.Z9zTiNsF81
!
ip subnet-zero
!
ip ssh time-out 120
ip ssh authentication-retries 3
!
spanning-tree mode pvst
no spanning-tree optimize bpdu transmission
spanning-tree extend system-id
!
!
!
!
interface FastEthernet0/1
 switchport access vlan 20
 switchport mode access
!
interface FastEthernet0/2
 switchport access vlan 20
 switchport mode access
!
interface FastEthernet0/3
 switchport access vlan 20
 switchport mode access
!
interface FastEthernet0/4
 switchport access vlan 20
 switchport mode access
!
interface FastEthernet0/5
!
interface FastEthernet0/6
!
interface FastEthernet0/7
!
interface FastEthernet0/8
 switchport access vlan 20
 switchport mode access
!
interface FastEthernet0/9
!
interface FastEthernet0/10
!
interface FastEthernet0/11
```

```
 switchport access vlan 20
 switchport mode access
!
interface FastEthernet0/12
!
interface Vlan1
 no ip address
 no ip route-cache
 shutdown
!
interface Vlan20
 ip address 10.10.1.240 255.255.255.0
 no ip route-cache
!
ip http server
!
line con 0
line vty 0 4
 login
line vty 5 15
 login
!
!
!
monitor session 1 source interface Fa0/1 - 5
monitor session 1 destination interface Fa0/8
end

UOC-TFM#
```

Hasta el momento de finalizar este TFM, estos fueron los ficheros de configuración que debimos modificar para poner en ejecución nuestra propuesta.