
**Universidad de Valladolid**

# Escuela de Ingeniería Informática

# Trabajo Fin de Grado

Grado en Ingeniería Informática

Mención Tecnologías de la Información

## Diseño e Implementación de una Plataforma de detección de amenazas de red

Autor:

**D. Alberto Alonso Frutos**


**Universidad de Valladolid**

# Escuela de Ingeniería Informática

# Trabajo Fin de Grado

Grado en Ingeniería Informática

Mención Tecnologías de la Información

## Diseño e Implementación de una Plataforma de detección de amenazas de red

Autor:

**D. Alberto Alonso Frutos**

Tutor:

**D. Javier Isaac Ramos López**

# Resumen

Este TFG presenta una guía completa de los aspectos teóricos y prácticos necesarios para realizar el diseño e implementación de una plataforma de detección de amenazas en la red del Departamento de Informática de la Universidad de Valladolid. Como introducción al proyecto se nos describen los problemas de seguridad a los que se expone una red y las distintas medidas existentes para lograr protegerla. Esta introducción nos presenta las medidas de seguridad que tendrán relevancia a lo largo del proyecto, estas medidas son los sistemas de detección de intrusos (IDS) y los sistemas de seguridad de la información y gestión de eventos(SIEM). Con el fin de realizar la elección de las herramientas empleadas en la plataforma de detección, se ha realizado un estudio individualizado de las herramientas IDS y SIEM en el mercado. Este estudio nos ha permitido decidir emplear la distribución Security Onion para la creación de la plataforma de detección.

Mediante el diseño e implementación de una arquitectura distribuida Security Onion se ha logrado monitorizar el tráfico del Departamento de Informática y con ello permitir desarrollar un estudio con las posibles vulnerabilidades de la red. Como comprobaciones complementarias, se ha realizado la inyección sobre la plataforma de detección de archivos PCAP con tráfico malicioso que nos ha permitido corroborar la correcta detección de ataques.

# Índice


1 Introducción . . . . . 9

1.1 Contexto del TFG 10

1.1.1 Situación actual de la ciberseguridad . . . . . 10

1.2 Motivaciones 11

1.3 Objetivos 11

1.4 Organización de la memoria 12

2 Marco tecnológico . . . . . 13

2.1 Seguridad de la red 13

2.1.1 Métodos de protección de la red . . . . . 14

2.1.2 Tipos de ataques . . . . . 19

2.1.3 Ataques más empleados . . . . . 19

2.1.4 Sistemas de detección de intrusos (IDS) . . . . . 21

2.1.5 Sistemas de prevención (IPS) . . . . . 25

2.2 Seguridad de la información y gestión de eventos (SIEM) 26

2.2.1 Herramientas SIEM . . . . . 26

2.2.2 Splunk . . . . . 26

2.2.3 ArcSight . . . . . 27

2.2.4 IBM QRadar . . . . . 28
**2.3 Herramientas de detección 29**
2.3.1 SNORT . . . . . 29
2.3.2 Suricata . . . . . 32
2.3.3 Zeek . . . . . 35
2.3.4 OSSEC . . . . . 36
2.3.5 OSSEC Wazuh . . . . . 39
2.3.6 Tripwire . . . . . 40
2.3.7 AIDE . . . . . 41
2.3.8 SAMHAIN . . . . . 41
2.3.9 Comparativa general herramientas . . . . . 43

**3 Sistema de monitorización . . . . . 47**
**3.1 Security Onion 47**
3.1.1 Herramientas integradas . . . . . 49
3.1.2 Arquitecturas . . . . . 53
3.1.3 Ventajas y desventajas de Security Onion . . . . . 57

**4 Análisis y Diseño . . . . . 59**
**4.1 Tecnologías elegidas 59**
**4.2 Sistema de virtualización 61**
**4.3 Diseño de Security Onion 62**

**5 Despliegue . . . . . 67**
**5.1 Configuraciones iniciales 67**
**5.2 Instalación nodo *manager* 68**
**5.3 Instalación nodo *search* 73**
**5.4 Instalación nodo sensor 74**
**5.5 Instalación máquina analista 76**

**6 Analisis de Resultados . . . . . 77**
**6.1 Intervalo de tiempo escogido 77**

6.2 Descripción de los datos 78
6.3 Datos obtenidos 79
6.4 Análisis de la red 82
6.5 Alertas de red generadas 85
6.6 Monitorización del sistema 93

7 Pruebas ..... 97
7.1 Conjunto de datos 97
7.2 Importación de los datos 98
7.3 Análisis de alertas obtenidas 99

8 Conclusiones y Trabajo Futuro ..... 105
8.1 Conclusiones 105
8.2 Trabajo futuro 106

Bibliografía ..... 107
Libros 107
Web 107

9 Anexos ..... 113
9.1 Ejemplo log de red 113
9.2 Ejemplo log de host 117
9.3 Ejemplo alerta de red 119
9.4 Mensaje alerta por descarga de troyano 124

# 1. Introducción

En la actualidad es una realidad que cada vez está cobrando más y más importancia el uso de las redes y el valor de la información. Como podemos observar en nuestra vida diaria, los comportamientos de la sociedad tanto en lo profesional como personal se han modificado drásticamente por el uso de las tecnologías. Este continuo incremento, unido a que cada vez se almacenan datos de mayor importancia tales como números de cuenta, contabilidad empresarial o datos personales, nos hacen darnos cuenta de la importancia que tiene establecer unas medidas de seguridad que sean capaces de protegernos de la continua evolución de los ataques contra las redes informáticas. Mediante la aplicación de estas medidas de seguridad deberemos ser capaces de garantizar su detección y proporcionarnos una respuesta efectiva.

Hoy en día, la gran mayoría de las organizaciones y empresas del mundo poseen complejas redes y sistemas TIC que guardan infinidad de datos sensibles y de valor. El número de amenazas que se reciben desde el exterior a estos sistemas son cada vez mayores y provienen de una gran cantidad y variedad de dispositivos conectados a Internet. Estas redes no sólo están expuestas a amenazas externas, si no también internas. Es fundamental que las organizaciones detecten de forma precoz la presencia de agujeros de seguridad susceptibles de convertirse en potenciales amenazas. Por todo ello, el análisis de vulnerabilidades ha ganado una gran relevancia durante los últimos años y, las herramientas que existen para desarrollar esta labor cada vez son más numerosas. En este TFG se ha ahondado en el contexto en el que nos encontramos, se ha estudiado en detalle el funcionamiento de estas herramientas, se ha seleccionado una y, por último, se ha realizado un despliegue real para un uso activo de la misma. Este proyecto pretende analizar las vulnerabilidades de la red del Departamento de

Informática de la Universidad de Valladolid.

## 1.1 Contexto del TFG

Este proyecto se enfoca en ayudar a mejorar la seguridad del Departamento de Informática mediante la detección de vulnerabilidades. Teniendo como base esta finalidad, definiríamos nuestro contexto como el conjunto de servicios de red del Departamento de Informática de la Universidad de Valladolid y el estudio de herramientas que vigilen o detecten vulnerabilidades en el mismo.

Con el fin de demostrar la necesidad de aplicar una plataforma de detección sobre el Departamento de Informática de la Universidad de Valladolid, vamos a proceder a detallar la situación global en términos de ciberseguridad. Detallaremos el contexto global mediante la muestra de datos actuales sobre el gran impacto de los ataques a los que nos exponemos por el uso de las redes y a su vez como de efectiva es la protección existente para responder a ellos.

### 1.1.1 Situación actual de la ciberseguridad

A lo largo de los últimos años podemos ver que hay una gran cantidad de ataques sobre las redes que han tenido un gran impacto sobre los afectados. Poniendo algún ejemplo concreto podemos detallar el ataque realizado sobre la ciudad Lake City, de Florida, EEUU [1] que resultó en un pago de 500.000 dólares a los atacantes. Enfocándonos en nuestro país, también disponemos de ataques informáticos que han tenido una gran relevancia. Uno de los más importantes es el ataque [2] realizado sobre el SEPE (Servicio Público de Empleo Estatal), el cual supuso una paralización en la tramitación de prestaciones en las 710 oficinas presenciales y en las 52 telemáticas, todo ello unido a una posible fuga de datos de empresas y trabajadores.

Según un informe de IBM realizado en el 2020 [3], el coste medio de una brecha de datos de seguridad en una compañía es de 3.86 millones de dólares. Únicamente este dato, unido a la pérdida de confianza generada en la población respecto a nuestra organización, nos hace darnos cuenta de la importancia de encontrar las vulnerabilidades de nuestra red antes de que sean empleadas por los ciberdelicuentes.

En el año 2020, el tiempo medio para identificar y contener una brecha de seguridad fue de 280 días por tanto las herramientas empleadas actualmente no están respondiendo de forma satisfactoria. Es para responder a este valor donde surge el proyecto planteado. Con el fin de explicar más en detalle cual es la situación actual de la seguridad a nivel global, se han analizado los documentos publicados [4] por la agencia europea para la ciberseguridad también denominada ENISA.

En primer lugar, hay que destacar los dos principales cambios que han ocurrido durante este último

año. El primero de ellos viene debido a la aparición del COVID-19, puesto que ha forzado a las tecnologías a emplearse en mayor cantidad de aspectos como por ejemplo la aplicación del teletrabajo en muchos hogares.

El segundo cambio más relevante, es la continua evolución de los métodos empleados para atacar una red, lo que conlleva que sea necesario un refinamiento periódico de las medidas de seguridad.

Estos dos hechos han supuesto que las medidas de seguridad sean cada vez menos efectivas y que no se adapten a las circunstancias actuales. Con el fin de ajustar la tecnología actual a las nuevas necesidades, los expertos de ciberseguridad han tenido que implementar una serie de soluciones que conllevan un descenso de la seguridad de las redes, en concreto podemos destacar la posibilidad de acceso remoto a los dispositivos o la integración de aplicaciones para la realización de videollamadas.

Este descenso de la seguridad conlleva que existan un mayor número de vulnerabilidades, las cuales será de vital importancia detectar.

## 1.2 Motivaciones

Tras la realización del grado de Ingeniería Informática, en el cual se nos enseña el funcionamiento de las redes y la importancia de la seguridad, he considerado como una parte importante de mi desarrollo como ingeniero informático la realización de un proyecto práctico que pueda aplicarse a un entorno real y que ponga a prueba todos los conocimientos aprendidos.

Unido a este aspecto, durante el desarrollo de este proyecto pretendo tener una visión más global de las herramientas que se emplean en una red en el ámbito empresarial.

Debido a que actualmente me encuentro en un puesto laboral en el cual mis funcionalidades se centran en el desarrollo de software, he querido realizar un proyecto que se centre en las redes informáticas y que me permita ampliar mi perfil profesional para solucionar posibles trabajos futuros.

## 1.3 Objetivos

El principal objetivo que se quiere lograr a lo largo de este TFG es ser capaz de diseñar y desplegar una plataforma de detección de amenazas en la red del Departamento de Informática de la Universidad de Valladolid, que nos permita encontrar las posibles vulnerabilidades de la red del departamento. Mediante la aplicación de una plataforma de detección, lograremos implementar en nuestro entorno una variedad de herramientas de seguridad que nos permitirán obtener información en tiempo real de la red y tras el análisis de esos datos permitirnos encontrar posibles vulnerabilidades.

Aunque el mencionado anteriormente es el principal objetivo, también deberemos lograr los siguientes subobjetivos:

1. Evaluar las mejores soluciones en el mercado respecto a herramientas de detección y elegir la mejor para nuestro entorno.
2. Comprender la instalación y funcionamiento de las herramientas de detección más empleadas en la actualidad.
3. Diseñar el entorno en el cual se aplicará la plataforma de detección. En concreto habrá que configurar las tecnologías escogidas y los recursos necesarios para su correcto funcionamiento.
4. Simular ataques reales sobre la red del departamento y con ello verificar el funcionamiento de la plataforma.

## 1.4 Organización de la memoria

La memoria del proyecto se divide de forma general en tres partes. La primera parte contendrá los aspectos teóricos necesarios para dar contexto al TFG y argumentar las decisiones tomadas. Esta primera parte se organizará en 3 secciones diferenciadas:

- **Introducción.**
- **Marco tecnológico.** Presentación de los conceptos teóricos necesarios. Descripción y comparación de las tecnologías disponibles en el mercado.
- **Herramienta escogida.** Elección y desarrollo de la herramienta empleada.

La segunda parte se centrará en primer lugar en la descripción de la solución práctica y en segundo lugar en el análisis de los datos obtenidos debido a su funcionamiento. Esta segunda parte se dividirá en las siguientes secciones.

- **Diseño y análisis.** Diseño y análisis de las partes que forman la plataforma de detección.
- **Despliegue.** Enumeración de los pasos necesarios para el funcionamiento de la plataforma de detección.
- **Análisis de resultados.** Explicación de los datos obtenidos debido al funcionamiento sobre el departamento.
- **Pruebas.** Descripción de pruebas realizadas de forma complementaria.

Por último, la tercera parte contendrá las conclusiones obtenidas tras la realización del TFG y el trabajo futuro a realizar.

# 2. Marco tecnológico

## 2.1 Seguridad de la red

En la actualidad, debido al continuo incremento en el uso de las tecnologías, cada vez se tiene un menor conocimiento del tráfico que fluye por las redes. Este hecho conlleva que cada vez se detecte una menor cantidad de los ataques que se realizan sobre la red. Como consecuencia, cada vez es más difícil conseguir una seguridad de red efectiva que pueda responder a las continuas amenazas.

Como primer paso para tratar este problema, vamos a definir que es la seguridad y como se logra. La seguridad [5], es un término muy amplio para referirnos a todas las medidas realizadas para la protección de nuestro sistema con el fin de garantizar la integridad, confidencialidad y disponibilidad de nuestros dispositivos y datos.

Para explicar más detalladamente la anterior definición, vamos a proceder a definir que es la integridad, confidencialidad y accesibilidad de la información.

- La **integridad** [6] se define como la propiedad que garantiza la precisión y validez de la información a lo largo de su existencia.
- La **confidencialidad** [7] es la propiedad por la que se garantiza que la información es accesible únicamente por personal autorizado.
- La **disponibilidad** [5] es la propiedad que nos garantiza que se pueda acceder a la información en cualquier momento.

Debido a que la seguridad de una red no se logra mediante una única herramienta, sino que engloba la aplicación de un conjunto de técnicas, vamos a desarrollar un modelo de seguridad basado en capas que se aplica para la protección de la información. Cada una de las capas representa un aspecto distinto

que debe aplicarse en la organización a proteger.

Las principales capas que vamos a detallar son la seguridad física, seguridad técnica y seguridad administrativa [8].

- La **seguridad física**[9] se basa en evitar el acceso físico a nuestra red con el fin de evitar manipulaciones en dispositivos como *routers* u ordenadores personales
- La **seguridad técnica** se centra en aplicar técnicas software y hardware para garantizar la protección de los datos tanto del interior de nuestra organización como del exterior.
- La **seguridad administrativa** [9] se enfoca en aplicar una serie de políticas de seguridad con el fin de controlar como aspecto principal el comportamiento de los usuarios. Este tipo de seguridad engloba aspectos como la forma de autenticarse, los niveles de acceso existentes, el cumplimiento de los reglamentos presentes respecto a seguridad o la realización de auditorías periódicamente.

Teniendo en cuenta los objetivos que queremos lograr en nuestro proyecto, vamos a enfocarnos en la aplicación de la seguridad técnica.

### 2.1.1 Métodos de protección de la red

Como ya hemos mencionado la seguridad solo se logra mediante la agrupación de distintas técnicas, por este motivo vamos a definir una serie de medidas que se aplican de forma general en el apartado de seguridad técnica de las redes. Gracias a la implementación de las medidas descritas por la compañía Cisco [10], unido a la plataforma de detección que desarrollaremos en este documento, nos permitirá garantizar la seguridad de la red.

- **Utilización Firewall.** Un *firewall* [11] es un dispositivo de seguridad que monitoriza el tráfico de red de entrada y salida basándose en un conjunto de reglas de seguridad anteriormente definidas. Básicamente funcionaría como una barrera entre la red interna de la organización e Internet. El funcionamiento se puede observar de forma más clara en la imagen 2.1.

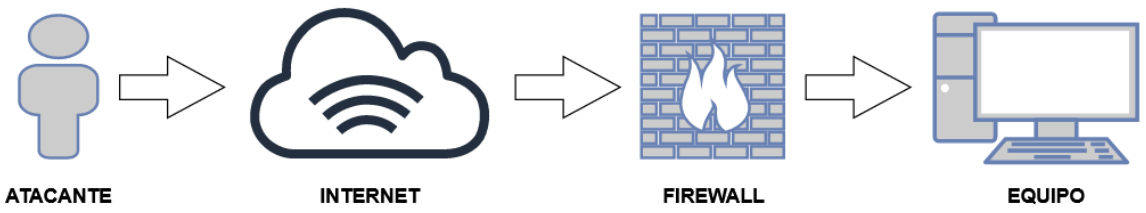


Imagen 2.1: Funcionamiento de un firewall

- **Seguridad correo electrónico.** Mediante la aplicación de un software de control de los correos entrantes en nuestra organización, podemos evitar una gran cantidad de amenazas de seguridad como por ejemplo campañas de *phising*. En concreto, detallaremos la técnica del *phising* [12], la cual se basa en contactar con el personal de nuestra organización simulando ser una institución legitima, por ejemplo un banco o una red social, con el fin de que el usuario introduzca información personal como nombres, contraseñas o datos bancarios.
- **Instalación Antivirus.** Un antivirus [13] es un software que está diseñado para la prevención, búsqueda, detección y eliminación de programas que tengan un comportamiento malicioso en nuestro sistema. Como ejemplos de estos programas maliciosos podríamos destacar los virus, gusanos y troyanos.


Imagen 2.2: Antivirus más famosos actualmente [14]

- **Seguridad de aplicaciones.** Cada una de las aplicaciones que se ejecutan dentro de nuestra organización pueden contener vulnerabilidades que permitan a los atacantes poner en peligro la seguridad de nuestra red. La implementación de medidas que nos permitan detectar y eliminar esas vulnerabilidades supone un gran beneficio para la seguridad de nuestra red.
- **Analíticas de comportamiento.** Esta medida de seguridad se basa en la aplicación de herramientas que generan estadísticas del comportamiento normal de nuestra red [15]. Mediante estas estadísticas podemos detectar si el funcionamiento de la red es distinto al normal y con ello identificar un posible problema de seguridad.
- **Segmentación de red.** La segmentación de red [16] consiste en dividir la totalidad de la red en partes más pequeñas denominadas subredes. Debido al menor tamaño de las subredes se facilita la aplicación de políticas de seguridad y permitir lograr un aumento del rendimiento y del control de la red. En la imagen 2.3 podemos observar una red segmentada en 3 distintas subredes.

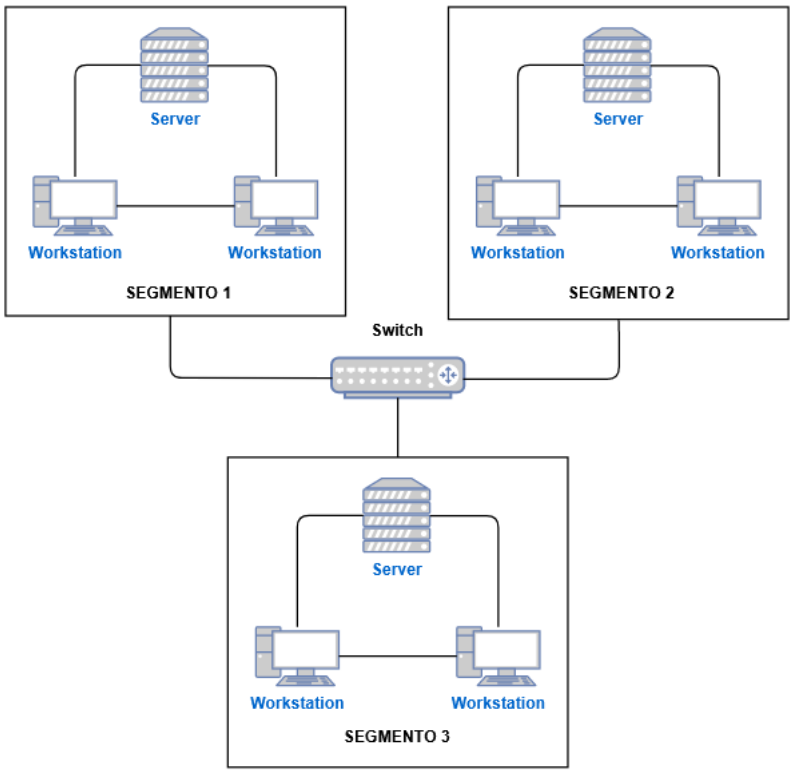


Imagen 2.3: Segmentación de red

- **Seguridad en la nube.** En la actualidad las organizaciones disponen de aplicaciones, servicios y datos almacenados en la nube. Este hecho conlleva que sea necesaria la aplicación de tecnologías y procesos para la protección de la nube. Una de las medidas que se emplea para asegurar la nube, es el cifrado de la información almacenada, como mejor ejemplo de una compañía que emplee la nube y aplique este cifrado podemos destacar *Amazon Web Service(AWS)* [17]
- **Software de prevención de perdida de datos.** La implementación de un software de prevención de perdida de datos [18] nos garantiza que no se produzcan filtraciones de información confidencial mediante la monitorización de los datos clave almacenados en nuestra organización.
- **Control de acceso.** Con el fin de evitar el acceso de posibles atacantes es necesario reconocer a cada usuario y dispositivo. Por este motivo, es importante implementar una tecnología de control de accesos [19] en nuestra red que emplee distintas políticas de seguridad. Podemos detallar un ejemplo concreto de red que implementa control de acceso mediante la imagen 2.4. En el diagrama mostrado, se dispone de un servidor de autenticación mediante el cual se aplican las políticas de seguridad deseadas. Impidiendo, en caso de no cumplir las políticas de acceso establecida, la conexión al servidor web tanto desde Internet como desde el equipo.

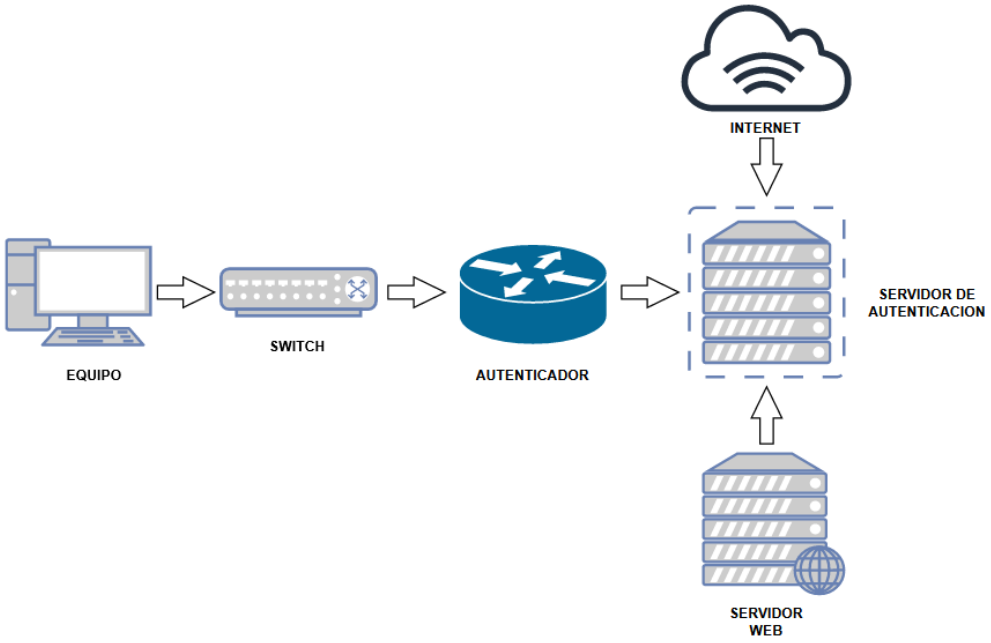


Imagen 2.4: Control de Acceso

- **Sistemas de detección (IDS).** Este tipo de software se centra en la monitorización de redes o sistemas con el fin de detectar actividades anómalas en nuestra organización. Como podemos observar en la imagen 2.5, el IDS se encontraría dentro de nuestra red y se encargaría, dependiendo del IDS empleado, de analizar el tráfico que pasa a través del *router* o de monitorizar los equipos dentro de nuestra organización.

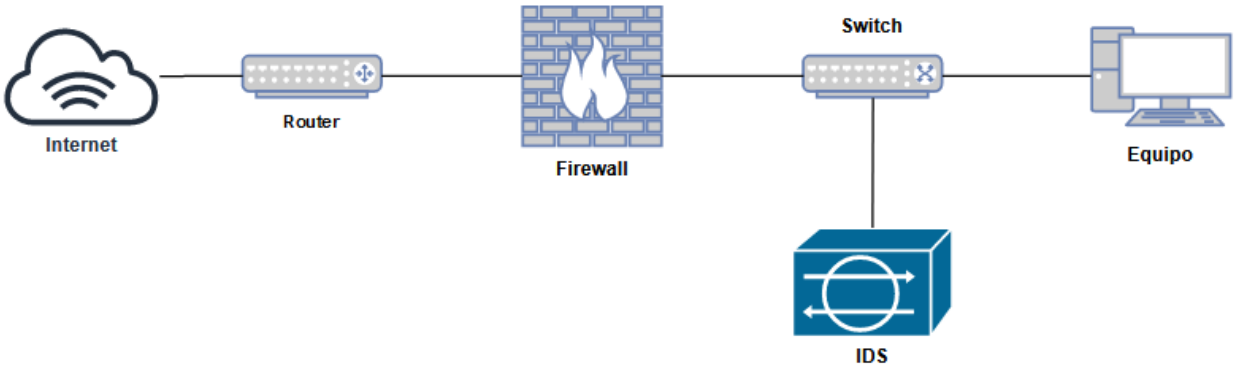


Imagen 2.5: Funcionamiento de un IDS

En posteriores apartados desarrollaremos de forma más detallada este tipo de herramienta.

- **Seguridad de dispositivos móviles.** El número de aplicaciones móviles que se utilizan en las organizaciones se ha incrementado en los últimos años. Con el fin de garantizar la seguridad es necesario controlar el acceso y configuración de los dispositivos móviles permitidos en nuestra red.

- **Seguridad de la información y gestión de eventos.** Existen un tipo de herramientas denominadas SIEM, (*Security Information and event management*) que implementadas en nuestra red nos permiten agrupar toda la información de la que disponemos respecto a la seguridad de nuestra organización. Esto nos permitirá observar de forma general nuestra organización y poder responder e identificar cualquier tipo de problema más rápidamente.
- **VPN.** La aplicación de una VPN o también denominada red virtual privada permite a un dispositivo concreto conectarse a una red de forma encriptada, de esta forma se garantiza una conexión segura entre los dispositivos de nuestra organización e Internet.
- **Seguridad web.** [20] Existen potenciales problemas de seguridad durante la navegación web tales como acceso a webs maliciosas o inyección SQL, entre otros muchos ejemplos. En concreto en el caso de una inyección SQL [21], se emplean los campos de entrada de datos de las webs con el fin de ejecutar consultas maliciosas en la base de datos. Mediante un software que monitorice la seguridad web, seremos capaces de controlar que cualquiera de los miembros de nuestra organización no acceda a webs con alguno de los peligros descritos.
- **Seguridad redes inalámbricas.** Las redes inalámbricas, conocidas coloquialmente como WIFI, suponen un mayor peligro que las redes cableadas. Por este motivo debemos disponer de herramientas que controlen el acceso y uso de los usuarios de la red inalámbrica.

Tras mencionar en este apartado las principales medidas empleadas para lograr una seguridad efectiva, vamos a proceder a explicar a qué problemas de seguridad se enfrentan las redes, es con esta problemática donde surge el concepto de ataque.

### 2.1.2 Tipos de ataques

Como paso previo a clasificar los ataques, vamos a definir el concepto de ataque puesto que será empleado a lo largo de todo el documento. Definimos ataque [22] como un asalto a la seguridad del sistema derivado de un acto inteligente y deliberado para eludir los servicios de seguridad y violar la política de seguridad de un sistema.

Para posteriores apartados en los que detallaremos la detección de ataques, vamos a necesitar realizar una clasificación respecto al impacto que generan sobre el sistema. Siguiendo esta división, diferenciamos entre ataques pasivos y ataques activos [23].

- **Ataques pasivos.** [24] Su objetivo es intentar conocer o hacer uso de información del sistema, pero sin afectar a los recursos del mismo. Debido a que se basan en escuchar u observar de forma no autorizada no realizan ninguna modificación y su detección es más compleja. Por este último motivo, cobra mayor importancia la prevención antes que la detección.
- **Ataques activos.** [23] El objetivo de este tipo de ataques es alterar los recursos del sistema o afectar a su funcionamiento, para lograrlo realizan modificaciones sobre el flujo de datos o crean flujos de datos falsos. En este caso la detección es más sencilla y la mejor estrategia para tratarlos es la detección y recuperación rápida.

### 2.1.3 Ataques más empleados

Una vez hemos definido y clasificado los ataques, uno de los aspectos más importantes es definir cuales son los más empleados [25] en la actualidad. A continuación, vamos a realizar una descripción detallada de cada uno de los ataques que mas obtendremos en nuestra plataforma de detección.

- **Ataque de negación de servicio.** Un ataque de negación de servicio es un intento de inhabilitar un servicio o red mediante la saturación de este con paquetes de red. En el ámbito de la seguridad, se diferencia entre ataque de negación de servicio distribuido (DDOS) y ataque de negación de servicio (DOS). La principal diferencia entre ellos es que, en el caso de un DDOS, el ataque a un sistema se realiza desde varios sistemas mientras que en un DOS se emplea un único sistema. Entre los posibles tipos de ataques de negación de servicio [26] podemos destacar:
  - **Ataques basados en volumen.** Se basan en la saturación del ancho de banda de red mediante el envío masivo de paquetes UDP y ICMP.
  - **Ataques de protocolo.** Se basa en la utilización de determinadas vulnerabilidades de los protocolos como *Ping of Death attack* o *Smurf DDoS* para consumir recursos del servidor. Un ataque *Smurf DDos* [27] se basa en saturar el servidor con paquetes del protocolo ICMP (*Internet Control Message Protocol*) que tienen en sus cabeceras valores de IP falsificados.
  - **Ataques de la capa de aplicación.** Incluye ataques que tienen como objetivo explotar las

vulnerabilidades de las aplicaciones de red contenidas en la capa de aplicación del modelo ISO-OSI. Podemos destacar ataques que tienen como objetivo vulnerabilidades presentes en sistemas operativos.

- **Ataques de escaneo.** El objetivo del escaneo de red es conocer información sobre la red objetivo con el fin de facilitarnos un ataque futuro, gracias a este ataque podemos conocer datos sobre la topología, el tráfico de red aceptado o sistema operativo entre otros muchos datos. Este ataque se realiza mediante el envió de paquetes a un sistema y a la posterior comprobación del contenido obtenido de la respuesta.
- **Ataques de penetración.** Se basa en la utilización de posibles errores en el software para obtener acceso como *root* y disponer así de control total sobre el sistema.

Una vez mencionados los ataques más empleados en la actualidad, nos vamos a centrar en explicar las herramientas que emplearemos para ponerles solución.

### 2.1.4 Sistemas de detección de intrusos (IDS)

Los sistemas de detección de intrusos o IDS (*Intrusion Detection System*)[28] son un tipo de software de seguridad que monitoriza los ordenadores individuales y analiza el tráfico de red, logrando así ser capaz de detectar distintos tipos de ataques provenientes tanto del interior como del exterior de nuestra organización.

Con el fin de realizar una explicación más sencilla, podemos establecer un símil de un IDS con la alarma antirrobo de un vehículo. De forma similar al robo de un vehículo y el consecuente sonido de la alarma, un IDS funcionaría de forma análoga a la alarma, alertándonos en caso de que se intente entrar a través del *firewall* o se realice algún acceso no autorizado u algún otro tipo de brecha en la seguridad. En la imagen 2.5 podemos ver un esquema del despliegue de un IDS en una red.

Según la procedencia de los datos podemos clasificar los IDS en los dos siguientes tipos:

- En caso de que los datos que analiza el IDS provengan de un dispositivo individual lo denominamos ***Host Intrusion Detection System***, o según sus siglas **HIDS**. La información que evalúan los HIDS, proviene del análisis de los *logs* generados en el ordenador y de la información proporcionada por el sistema operativo del sistema en el que se encuentren instalados. A partir de este momento vamos a definir a cada ordenador individual de nuestra infraestructura como host.
- Cuando la fuente de datos de nuestro IDS son los paquetes que circulan por nuestra red, denominamos el software como ***Network Intrusion Detection System*** o según sus siglas **NIDS**. De forma general, el funcionamiento de un NIDS se basaría en monitorizar una interfaz de red y con ello obtener alertas para el tráfico que fluye por ella.

Además de una clasificación según la procedencia de los datos, los IDS también se diferencian según el método que utilicen para la detección de ataques, pudiendo distinguir entre los dos siguientes tipos:

- **Los IDS centrados en la utilización de firmas** se basan en comparar los paquetes de red con patrones denominados firmas o reglas, en los casos en los que se determine que el paquete de red coincide con ese patrón, determinaremos que se trata de tráfico malicioso y se enviará una alerta. Para cada IDS se dispone de una base de datos de firmas que puede ser editable por el administrador, pudiendo así añadir firmas propias para determinados casos. Poniendo un ejemplo concreto, podríamos crear firmas específicas para detectar escaneos de puertos o la llegada de peticiones de determinada dirección IP.

  Generalmente las firmas pueden clasificarse [28] en tres tipos distintos que son los siguientes:

  – **Sistemas expertos.** Este sistema se centra en establecer un número de condiciones y en

caso de cumplirse todas ellas lanzar la firma correspondiente. Como ejemplo de esta firma podríamos detallar la aplicación de condiciones en los paquetes de red para detectar determinados caracteres ASCII. Estos caracteres representarían palabras clave dentro del paquete como son la contraseña o el usuario.

  - **Detección de firmas.** Este enfoque de firmas se basa en comparar los eventos que suceden en el sistema con las firmas almacenadas en la base de datos de firmas.
  - **Análisis de transacción de estados.** Para la implementación de un análisis de transacción de estados es necesario considerar un ataque como una secuencia de pasos. Cada uno de los pasos representará el estado en el cual se encuentra el ataque. Este sistema se centra en generar una alerta en los casos en los que un ataque ha llegado al estado considerado como amenaza para la seguridad.

  El principal problema existente en este tipo de enfoque basado en firmas se centra en que el IDS no es capaz de detectar nuevos ataques, esto se debe a que no disponemos de información sobre ellos al no encontrarse en la base de datos de firmas. Este hecho hace que sea de vital importancia realizar una actualización periódica de estas firmas.

- **Los IDS basados en anomalías** funcionan mediante la comparación del estado actual del sistema con un modelo creado anteriormente, este modelo refleja el comportamiento normal de nuestra red en términos de ancho de banda utilizado, puertos abiertos y otros datos de nuestro entorno. En los casos en los que el tráfico de red obtenido conlleve una variación de nuestros sistemas, el sistema de detección generará una alerta.

  Con el fin de establecer que se considera comportamiento normal y crear con ello el modelo mencionado, podemos definir 3 enfoques [28]:

  - **Sistemas basados en conocimiento.** Este tipo de sistema se emplea al inicio del despliegue del IDS, por este motivo no dispone de información de su funcionamiento y emplea las reglas almacenadas en la base de datos de reglas.
  - **Sistemas basados en métodos estadísticos** Se emplea el comportamiento del usuario para realizar el cálculo de métricas y modelos estadísticos.
  - **Sistemas basados en aprendizaje automático** Este enfoque se centra en emplear algoritmos que vayan aprendiendo de los datos en cada iteración, este aprendizaje les permite reconocer patrones y generar modelos del comportamiento del sistema.

  Una de las principales ventajas de este enfoque basado en anomalías es que nos permite detectar ataques nuevos, ya que al centrarnos en nuestro sistema el método de ataque no influye en su detección.

A continuación, vamos a proceder a realizar una comparación entre HIDS y NIDS con el fin de

conocer que nos aporta cada uno de ellos a nuestra infraestructura y establecer los puntos débiles que vamos a tener en cuenta en nuestra implementación.

#### Detección en hosts

Como hemos definido anteriormente un HIDS en un sistema de detección que obtiene los datos de análisis de *host* individuales, este hecho nos proporciona una de sus principales ventajas que consiste en que se nos permite conocer el proceso especifico por el que se ha causado esa alerta en ese *host*. Con este proceso localizado podemos controlar el comportamiento de ese usuario y poder solucionarlo de forma más efectiva.

Respecto a la implementación de este tipo de IDS en nuestra infraestructura podemos determinar que nos proporciona gran escalabilidad y una relación coste eficacia muy beneficiosa, esto se debe a que la carga se distribuye entre los *hosts*, siendo solo necesaria la instalación de un agente en cada uno de ellos. Definiríamos un agente, como un software que monitoriza el sistema buscando *malware*, *rootkits* y anomalías sospechosas. Teniendo en cuenta la gran cantidad de información que se genera en los *logs* del sistema, también tiene sentido que una de las ventajas de los HIDS sea la gran cantidad de información que nos puede proporcionar sobre el *host*.

Obviamente esta tecnología no dispone solo de ventajas, también tiene algunos puntos débiles que deberemos cubrir y que vamos a detallar a continuación. Como hemos mencionado anteriormente, un HIDS nos proporciona gran cantidad de información, aunque esto en principio es una ventaja, con ello también nos surge un problema puesto que los archivos requieren una gran cantidad de uso de disco.

Respecto a los datos obtenidos, dispondremos de aquellos que se generen por el sistema operativo del *host* y en caso de querer determinada información que no se genere puede que sea necesario una modificación del código del sistema operativo. Teniendo en cuenta la instalación en entornos de gran tamaño también podemos establecer como desventaja que el precio de gestión y despliegue de un software no enfocado a entornos grandes puede ser muy costoso.

#### Detección en red

En este apartado, de forma análoga al anterior, vamos a desarrollar las ventajas e inconvenientes que tendría la implementación de un NIDS, en nuestra infraestructura.

En primer lugar, en términos del coste y rendimiento que nos ofrece, se puede indicar que un NIDS no reduce el rendimiento de otros programas ejecutándose sobre la red, es de fácil mantenimiento e instalación y es independiente al sistema operativo instalado, además un solo NIDS puede cubrir una gran cantidad del tráfico de red.Como resultado de todos los aspectos mencionados, se considera a los NIDS un software muy bueno económicamente hablando.

Por supuesto los NIDS también suponen desventajas en nuestra infraestructura. Estas desventajas

tienen que ver con el uso de firmas, puesto que siempre vamos a estar por detrás respecto a ataques no conocidos y en algunos casos hasta podrían ser un potencial peligro ataques conocidos debido a bases de datos de firmas desactualizadas.

### 2.1.5 Sistemas de prevención (IPS)

Un sistema de prevención o IPS (*Intrusion Prevention Systems*) [29], es un software que se instala entre la red interna de nuestra organización y la red externa y que se encarga de leer el flujo de paquetes de red entrantes y a partir de ellos identificar los posibles ataques a nuestra red.

Aunque este tipo de herramientas son similares a un IDS respecto a la detección de ataques, la principal diferencia reside en las acciones que se realizan una vez detectado el ataque. En el caso de un IPS, cuando se realiza la detección de la anomalía esto llevará al sistema a realizar una respuesta automática a ese ataque, que se centra en aplicar tres distintas soluciones que son las siguientes:

- Bloquear el tráfico de la IP fuente del ataque.
- Descartar los paquetes maliciosos.
- Enviar alertas al usuario.

Podemos resumir que un IPS es un software que funciona de forma similar a un IDS respecto a la detección pero que además proporciona respuesta a esas alertar generadas. Aunque también realicen la detección de alertas, los IPS disponen de menos funcionalidades para esta funcionalidad respecto a un IDS.

Actualmente en la industria de la informática, se está realizando una transición a la utilización de IPS sobre un IDS [30]. Esto se debe a que nos proporcionan la posibilidad de responder en tiempo real a los posibles problemas, de todas formas muchas de las herramientas actuales se declaran tanto como IDS como IPS aprovechándose de las ventajas de ambos.

Para nuestro caso práctico nos vamos a enfocar en la utilización de los IDS, puesto que dentro del Departamento de Informática ya disponemos de otras posibilidades, a mayores de los IPS, que realizarían la respuesta a las alertas detectadas.

## 2.2 Seguridad de la información y gestión de eventos (SIEM)

Una vez hemos descrito las distintas opciones de las que disponemos para la implementación de sistemas de detección en nuestra organización, vamos a estudiar como trataremos la información que nos proporcionan. Con el fin de gestionar toda la información que van a generar esos IDS en nuestro entorno surgen las herramientas SIEM, las cuales nos permitirán agrupar toda la información proporcionándonos así una vista general de la seguridad de nuestro sistema.

En primer lugar, definimos SIEM [31] como un conjunto de herramientas que proporcionan una vista centralizada de la seguridad de una infraestructura. Siendo más específico un SIEM es un software que proporciona los siguientes aspectos los cuales son mencionados en el siguiente documento [32]:

- Visibilidad en tiempo real de la información obtenida por los sistemas de seguridad
- Gestión de *logs* provenientes de distintas fuentes.
- Añaden lógica a los eventos registrados en los *logs*.
- Proporcionan métodos para notificar problemas de seguridad.

El funcionamiento de un SIEM [33] se basa en la combinación de dos tecnologías. La primera de ellas se trata de la gestión de información de seguridad también denominado SIM (*Security Information Management*). Mediante esta tecnología se van recogiendo todos los *logs* de las distintas fuentes de la infraestructura para que posteriormente se analicen y con ello se puedan detectar problemas de seguridad. La segunda se trata de la gestión de eventos de seguridad, que nos garantiza la monitorización de la red en tiempo real y la notificación de problemas a los técnicos de seguridad.

Aunque en nuestro entorno se podrían instalar únicamente los sistemas de detección, la gran cantidad de información de la que dispondríamos podría saturar a los profesionales de seguridad de nuestra organización. Es para ese caso en el que reside la principal ventaja de la implementación de un SIEM, puesto que nos permite responder de forma más precisa y rápida presentándonos todos los datos obtenidos de forma resumida.

### 2.2.1 Herramientas SIEM

A continuación, vamos a presentar algunos de los SIEM más populares en el mercado, entre ellos nos encontramos con Splunk, ArcSight, IBM QRadar.

### 2.2.2 Splunk

El software Splunk [34] se encarga de capturar y agrupar los datos obtenidos de las distintas fuentes para permitir su visualización de forma sencilla.


Imagen 2.6: Splunk

De forma general, cada SIEM primero se centra en definir la fuente de datos que vamos a estudiar. Una vez definida la fuente y obtenidos los datos que nos proporciona, los datos son indexados y convertidos en una serie de eventos individuales.

En el caso de la herramienta Splunk, una vez los datos se han obtenido son tratados para proporcionarnos los siguientes aspectos:

- **Indexación.** Tras la obtención de todos los datos de webs, servidores u otras fuentes, Splunk realiza la indexación, almacenamiento, comprensión y mantenimiento de los datos.
- **Búsqueda.** Una de las principales funcionalidades que nos aporta es la capacidad de obtener información de los datos almacenados. En concreto, podemos realizar las siguientes consultas:
  - Devolver eventos por índice.
  - Cálculo de métricas.
  - Búsqueda por condiciones.
  - Identificación de patrones.
  - Predicción de comportamientos futuros.
- **Alertas.** Dispone de la posibilidad de notificar y realizar acciones en caso de que se cumplan determinadas condiciones que se hayan configurado previamente. Algunos ejemplos en concreto serían el envío de correos o la ejecución de *scripts*.
- ***Dashboards.*** Nos permite visualizar la información de forma gráfica mediante el uso de paneles
- **Objetos del modelo.** Permite realizar la organización de los distintos datos y la relación entre ellos.
- ***Pivot.*** Splunk dispone de un editor denominado *Pivot Editor*, que permite relacionar tablas de bases de datos con los distintos objetos del modelo de datos obtenido , todo ello sin la necesidad de escribir consultas.
- ***Reports.*** Permite la creación de informes con los datos obtenidos.

### 2.2.3 ArcSight

ArcSight [34], es un SIEM que nos proporcionan la posibilidad de almacenar hasta 7500 eventos por segundo provenientes de distintas fuentes. Debido a su diseño basado en tres capas distintas, dispone de tres funcionalidades muy diferenciadas. La primera capa denominada ArcSight ESM, se centra en la detección de alertas, la segunda, de nombre *ArcSight Data Platform*, se encarga de la obtención

y distribución de los datos y por último la tercera llamada *ArcSight Investigate*, está enfocada a la investigación y creación de analíticas.


Imagen 2.7: ArcSight

Las principales ventajas de ArcSight vienen de su gran capacidad de obtención de datos provenientes de fuentes muy variables y de su capacidad de integrarse con un entorno SOC. Un entorno SOC [35] es un centro de seguridad de operaciones encargado de monitorizar, prevenir, detectar, investigar y responder a los ciberataques.

En cuestión de desventajas destacaríamos el precio, ya que la licencia no es abierta, junto con su curva de aprendizaje. Este último aspecto se debe aF que algunos usuarios declaran haber tenido problemas al utilizarlo.

### 2.2.4 IBM QRadar

El SIEM de la empresa IBM de nombre IMB QRadar [36], nos proporciona una visión centralizada para detectar, investigar y responder a los problemas más críticos de la seguridad de nuestra organización. Debido a las dificultades que presenta para integrarse con el software que vamos a implementar [37], unido a su alto precio, nos hacen descartarla como opción viable.


Imagen 2.8: IBMQRadar

## 2.3 Herramientas de detección

Una vez hemos presentado que es y todas las ventajas que nos aporta un IDS, es necesario conocer cuáles son los existentes en el mercado y realizar una comparativa entre ellos. Esto nos permitirá decidir de forma argumentada cuál es la mejor opción para utilizar en nuestro despliegue. Tendremos en cuenta el estudio [38].

Para cada una de las siguientes herramientas vamos a explicar las principales funcionalidades de las que disponen, así como también, de una serie de tablas comparativas para destacar las ventajas y desventajas que nos aportarían a nuestro despliegue.

### 2.3.1 SNORT

Snort [39] es una herramienta multiplataforma, gratuita y de software abierto que funciona como NIDS e IPS.


Imagen 2.9: Logo de Snort

Mediante la aplicación de un conjunto de reglas denominadas Talos sobre el tráfico de red obtenido, es capaz de detectar una gran cantidad de alertas. Supone una opción enfocada a la captura del tráfico de red de entornos de pequeño y mediano tamaño, siendo una alternativa viable en casos de no permitirnos una opción comercial.

Cabe destacar que el conjunto de reglas Talos se denomina de esta forma, debido a que se corresponde con el nombre del grupo de expertos que se encargan de encontrar vulnerabilidades y crear las reglas. Respecto a la actualización de las reglas, podemos indicar que se actualizan de forma inmediata en caso de disponer de una subscripción de pago y que en caso contrario será necesario esperar 1 mes hasta su actualización.

Centrándonos en el funcionamiento, Snort dispone de tres modos de trabajo que son los siguientes:

- **Modo *sniffer.*** Este modo se caracteriza por monitorizar los paquetes que pasan por la red y mostrárnoslos en pantalla. Gracias a su funcionamiento disponemos de información sobre todo el tráfico que fluye por nuestra organización. Realiza una funcionalidad similar a otras herramientas conocidas, como por ejemplo *Wireshark*.
- **Modo *log* de paquetes.** Este modo será necesario en caso de querer almacenar los paquetes en nuestro disco. Durante su ejecución, Snort va recogiendo todos los paquetes de red y

agrupándolos jerárquicamente mediante la dirección IP de cada datagrama.

- **Modo NIDS.** Esta modalidad se caracteriza principalmente por aplicar el conjunto de reglas al tráfico de red para la detección de alertas. Siendo más específicos, en el caso de Snort el fichero de configuración de las reglas se denomina snort.conf y los resultados obtenidos de su aplicación se almacenan en el directorio /var/log/snort.

Para nuestro caso particular nos resulta más importante el modo de funcionamiento NIDS, ya que será el que aplicaremos en nuestro ejemplo práctico. Debido a este último motivo, vamos a explicar más detalladamente su funcionamiento incluyendo un ejemplo de alerta estándar que nos podría generar.

### Funcionamiento en modo NIDS

El funcionamiento de Snort [40] se caracteriza por estar divido en 5 módulos que podemos observar en la imagen 2.3.1 y que explicaremos a continuación:

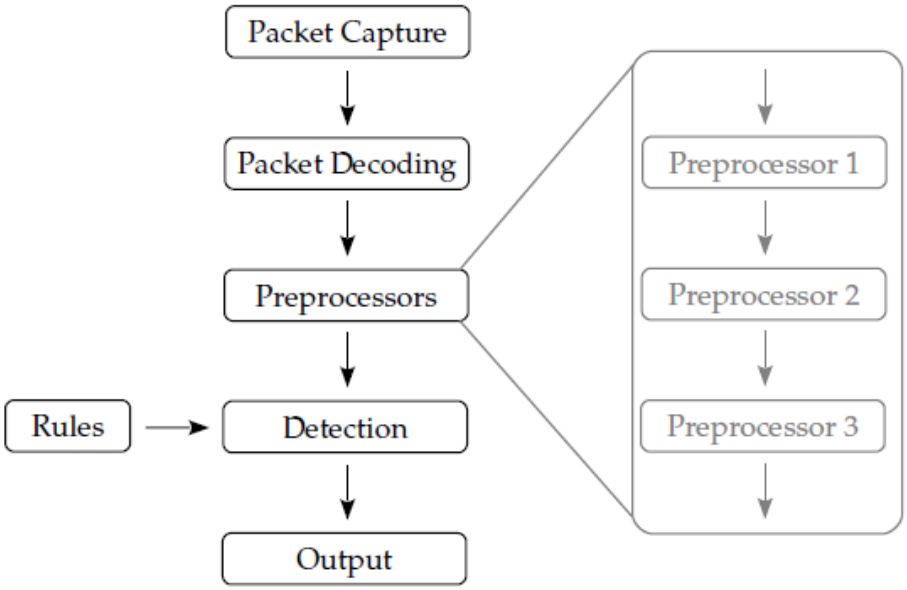


Imagen 2.10: Arquitectura de Snort[40]

- **Captura de paquetes.** Mediante la aplicación de la librería llamada DAQ (*Data acQuisition*), se realiza un registro de todos los datos transmitidos en la red. Una vez obtenidos, son enviados para su decodificación.
- **Decodificación.** Este módulo se encarga de analizar las cabeceras de los paquetes, permitiendo detectar anomalías como por ejemplo problemas en los flags al utilizar el protocolo TCP. Los flags de las cabeceras son campos que se envían previo a los datos que nos indican distintas características de la comunicación entre los dispositivos de red.

- **Preprocesado.** El preprocesado en una ampliación de la decodificación diseñado para realizar el análisis de los datos para la capa de red, transporte y aplicación del modelo ISO-OSI. El aplicado del preprocesado, puede lograr detectar problemas de nuestra red en el uso de protocolos como FTP, SMTP, SSL o SSH entre otros muchos ejemplos.
- **Motor de detección.** En este punto del funcionamiento, se aplican las reglas mencionadas anteriormente para la detección de alertas.
- **Modulo salida.** Una vez obtenidos los resultados estos pueden mostrarse de distintas formas, para ello el módulo de salida nos proporciona la posibilidad de ver los resultados como ficheros *syslog*, ASCII o PCAP.

#### Salida estándar de Snort

Con el fin de comprender de forma sencilla una salida estándar para un IDS, en este caso Snort, vamos a explicar un ejemplo concreto para el que detallaremos toda la información que es capaz de proporcionarnos, el ejemplo se puede observar a continuación.

```
[**] [116:56:1] (snort_decoder): T/TCP Detected [**]
```

- El primer número (116) denominado *GeneratorID* nos indica el componente del IDS que ha generado esta alerta, la lista de *GeneratorID* del IDS se pueden ver en el directorio */etc/generators*.
- El segundo número (56) se trata del Snort ID y nos permite identificar la regla que se ha empleado para su detección, en este caso vemos que es la regla 56 que se trata de la regla *T/TCP event*.
- El tercer número (1) es el *revision ID*, el cual se irá incrementando para cada una de las ejecuciones de la regla.
- Por último, en los dos últimos campos se nos muestra en forma de texto una breve descripción de la alerta detectada.

#### Ventajas y desventajas

En la siguiente tabla 2.1 se nos van a indicar las principales ventajas y desventajas de desplegar Snort como NIDS en un entorno de red.

| VENTAJAS | DESVENTAJAS |
|---|---|
| Multiplataforma | Dificultad de aprendizaje |
| Gratuito | No dispone de GUI |
| Manual y comunidad de soporte | Configuración especial para falsos positivos |
| Reglas actualizadas y customizables | Saturación de información debido a una base de datos de reglas muy amplia |
| Bajos requisitos necesarios | No enfocado a entornos de gran tamaño |
| Disponibilidad de interfaz web | No diseñado para infraestructuras modernas No soporta IPV6, multithread o velocidades altas de red |

Tabla 2.1: Ventajas y desventajas Snort

### 2.3.2 Suricata

Suricata se define [41] como un sofware multiplataforma gratuito y de código libre desarrollado por la *Open Information Security Foundation* que proporciona la funcionalidad de NIDS y NIPS. Gracias a su implementación nos permite monitorizar la red y comparar el tráfico dentro de ella con un conjunto de reglas para detectar y alertar en caso de actividad maliciosa. Una vez ha detectado algún tipo de comportamiento malicioso, también dispone de la posibilidad de aplicar la respuesta necesaria para cada alerta.


Imagen 2.11: Logo de Suricata

Aunque utiliza una colección de reglas similar a Snort, siendo incluso compatible con las mismas, la principal diferencia radica en que nos permite una mayor especificación de reglas para los protocolos de la capa de aplicación.

Una de las principales diferencias respecto a otro NIDS, es que se encuentra preparado para multihilos y para aplicar automáticamente balanceo de carga entre estos hilos. Debido a este motivo, es una herramienta enfocada a ofrecernos la posibilidad de escalar a entornos de mayor tamaño.

Respecto a las salidas que produce, podemos destacar que recoge estadísticas de rendimiento en el archivo stat.log y que se pueden obtener las salidas en formato YAML. Este último formato facilita el uso de herramientas como Logstash que explicaremos en siguientes apartados.

#### Funcionamiento de Suricata

El funcionamiento de Suricata [40] es similar al de Snort puesto que su desarrollo se ha realizado basándose en el que emplea Snort. En el caso de Suricata se divide en 4 módulos los cuales están representada en la imagen 2.12. A continuación, voy a proceder a detallar los aspectos de cada módulo que se diferencien respecto a los mencionados para Snort.

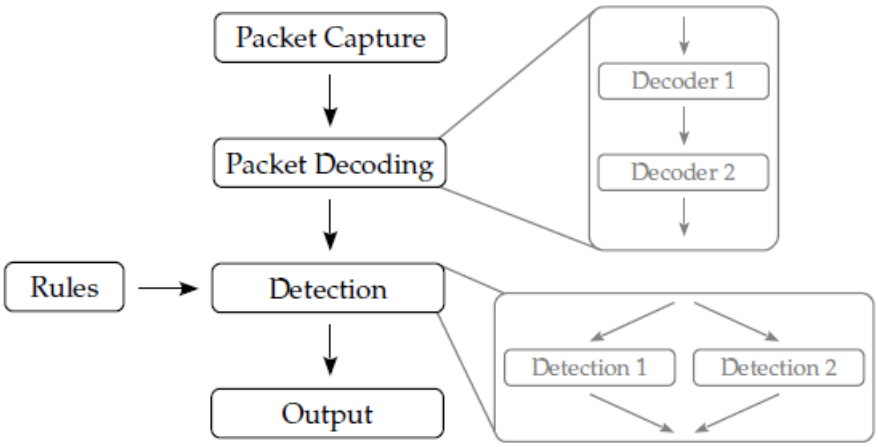


Imagen 2.12: Arquitectura de Suricata [40]

- **Captura de paquetes.** A lo largo de la ejecución de este módulo la información de la red se transforma en una representación interna que simula un paquete de red para Suricata.
- **Decodificación de paquetes.** Este proceso de decodificación consiste en aplicar una a una un conjunto de funciones que añaden información a la representación interna del paquete obtenido previamente. Cabe destacar que este módulo puede ampliar su funcionalidad mediante la creación de nuevas funciones.
- **Detección.** De forma simultánea y una vez ya hemos decodificado el paquete, podemos aplicar el conjunto de reglas definido para la detección de alertas. Este conjunto de reglas se aplica mediante módulos de detección, los cuales agrupan cada uno de ellos una serie de reglas. Entre algunos de esos módulos de detección podemos destacar *ET POLICY* o *ET INFO*, los cuales se detallarán posteriormente mediante casos prácticos. A mayores se aporta la posibilidad de crear nuevos módulos de detección.
- **Salida.** Como último paso, realizamos la salida de los datos obtenidos siguiendo el formato que indiquemos de los disponibles.

### Ventajas y desventajas

En la siguiente tabla 2.2, se nos van a indicar las principales ventajas y desventajas de desplegar Suricata como NIDS en nuestro despliegue.

| VENTAJAS | DESVENTAJAS |
|---|---|
| Multiplataforma | Mayor uso de recursos (CPU, RAM) |
| Gratuito | Obtención de mayor número de falsos positivos |
| Manual de usuario y para desarrollo | Posibilidad de positivos grises |
| Reglas actualizadas y customizables | |
| Mayor precisión que otros IDS | |
| Escalabilidad | |

Tabla 2.2: Ventajas y desventajas Suricata

**Nota 2.3.1** Los IDS pueden obtener 4 resultados dependiendo de si la detección del ataque es correcta o incorrecta.

- Verdadero positivo: Ataques reales predicho como ataque
- Falso positivo: Se identifica como ataque positivo un resultado negativo.
- Verdadero negativo: Se identifica como negativo un resultado que tiene que ser negativo.
- Falso negativo: Se identifica como negativo un resultado que es positivo.

También cabe indicar que un positivo gris es una alerta que puede ser percibida por el IDS tanto como un falso positivo como verdadero positivo dependiendo del contexto.

### 2.3.3 Zeek

La herramienta Zeek [42] es un analizador de tráfico de red de código abierto cuya incorporación a nuestro entorno nos proporciona una gran cantidad de ficheros *logs* con la activad de nuestra red. En estos *logs* se va almacenando información como por ejemplo las sesiones HTTP, certificados SSL o las peticiones DNS, entre otros muchos datos.

Imagen 2.13: Logo de Zeek

También podemos definir Zeek como un NIDS basado en anomalías, que a mayores de las funcionalidades estándar de un sistema de detección, nos proporciona un lenguaje de *scripting* que nos permite un mayor espectro de posibilidades de detectar actividad maliciosa.

#### Arquitectura de Zeek

La arquitectura de Zeek se basa en dos componentes principales como se detalla en la imagen 2.14, el motor de eventos y el intérprete de *scripts*. En primer lugar, actúa el motor de eventos, que se encarga de transformar el flujo de paquetes de red obtenido en una serie de eventos de alto nivel. Los eventos obtenidos solo describen la actividad de la red sin tener en cuenta si pueden provocar posibles problemas en nuestra red, de este aspecto se encargará el intérprete de *scripts*.

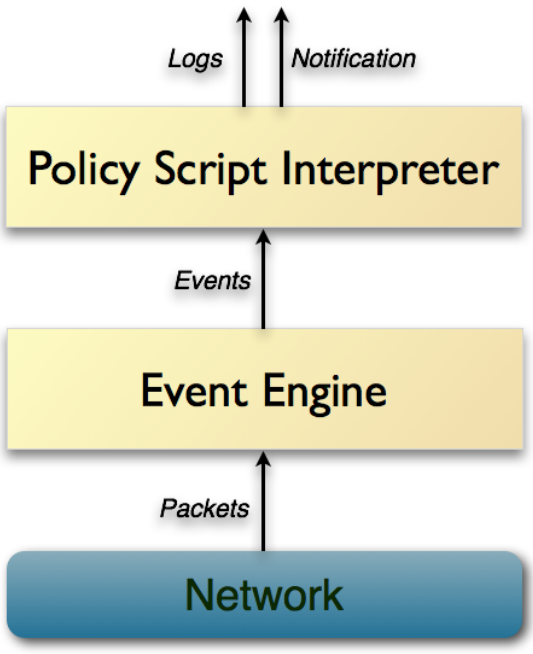


Imagen 2.14: Arquitectura de Zeek [42]

El intérprete de *scripts* recibirá una serie de eventos que se corresponden con la actividad de la red. En este momento, se encargará de ejecutar el conjunto de manejadores de eventos (*event handlers*) definidos, los cuales se encargan de la detección y respuesta de los ataques a nuestra red.

En caso de implementar Zeek en nuestra plataforma, se obtendrán las alertas mediante los *scripts* incluidos en su configuración inicial.

#### Ventajas y desventajas de Zeek

En la siguiente tabla 2.3 se nos van a indicar las principales ventajas y desventajas de desplegar Zeek como NIDS en nuestro despliegue.

| VENTAJAS | DESVENTAJAS |
|---|---|
| Multiplataforma | Necesidad de formación en su propio lenguaje |
| *Open-source* gratuito | No dispone de funcionalidad multithread. |
| Lenguaje de *scripting* propio | |
| Alto rendimiento en redes con altas velocidades(10-100GB) debido al uso de balance de carga | |
| Disponibilidad de manual | |

Tabla 2.3: Ventajas y desventajas de Zeek

### 2.3.4 OSSEC

OSSEC [43] es un HIDS de código abierto y multiplataforma, al tratarse de un HIDS implica que va a obtener los datos que se generan en los dispositivos conectados a la red como ordenadores personales o dispositivos móviles.


Imagen 2.15: Logo de OSSEC

Entre sus principales funcionalidades encontramos la realización de análisis de los *logs* generados dentro del sistema, chequeo de la integridad de los ficheros, permitiéndonos así comprobar que los datos son correctos, monitorización de los registros de Windows, detección de *rootkits*, alertas en tiempo real y respuesta activa a cada alerta generada. Un *rootkit* es un conjunto de herramientas software utilizadas para permitir el acceso privilegiado sobre un dispositivo a un usuario que no

debería disponer de ese acceso. A continuación, vamos a explicar más detalladamente la arquitectura y funcionamiento de OSSEC.

### Funcionamiento de OSSEC

La arquitectura de OSSEC está compuesta por un conjunto de piezas como podemos observar en la imagen 2.16. Como pieza más importante disponemos de un *manager* o *server* central que se encarga de recibir y monitorizar la información de los agentes, *syslogs* y bases de datos.

Un agente Ossec es un pequeño programa o colección de programas instalado en el *host* a monitorizar que recolecta la información en tiempo real y la envía al *manager* para su análisis. Además de los agentes podemos recolectar información procedente de máquinas virtuales, *switches*, *routers* entre otros muchos dispositivos de nuestra red.

Resumiendo, el funcionamiento de OSSEC se basa en el uso de servidores y agentes, siendo el servidor el que gestiona y analiza la información que recibe de los agentes.


Imagen 2.16: Arquitectura OSSEC [43]

### Ventajas y desventajas

En la siguiente tabla 2.4, se nos van a indicar las principales ventajas y desventajas de desplegar OSSEC como HIDS en nuestro despliegue.

| VENTAJAS | DESVENTAJAS |
|---|---|
| Multiplataforma | Dificulta de actualización |
| Gratuito | Envió de claves entre agente y servidor problemático |
| Analiza información de múltiples dispositivos y en formatos distintos | |
| Sistema de respuesta en tiempo real y alertas configurables | |
| Gestión centralizada de políticas de seguridad en sistemas operativos distintos | |
| Manual de uso | |

Tabla 2.4: Ventajas y desventajas de OSSEC

### 2.3.5 OSSEC Wazuh

Wazuh [44] es una plataforma gratuita y de código abierto utilizada para la detección de amenazas, monitorización de seguridad y sistema de respuesta frente a incidencias.

Imagen 2.17: Logo de Wazuh

Centrándonos en su utilización como sistema de detección, definiríamos Wazuh como un HIDS que utiliza un enfoque basado en firmas y un motor de expresiones regulares para el análisis de *logs*. La aplicación de esta tecnología en los agentes nos proporciona la posibilidad de detectar *malware*, *rootkits* y posibles anomalías.

Su arquitectura y funcionamiento general es similar a OSSEC puesto que en su desarrollo comenzó siendo una bifurcación de este, aunque ahora mismo ha logrado proporcionar más funcionalidad, escalabilidad y facilidad de uso.

A continuación, voy a explicar las principales funcionalidades añadidas a Wazuh respecto a OSSEC:

- **Integración con ELK Stack.** Wazuh dispone de su propio *plugin* de Kibana que permite la comunicación entre Kibana y el *manager* con el fin de mostrar estadísticas de los agentes de forma más clara y concisa. Como resultado de esta integración se proporcionan las funcionalidades siguientes.
  - Visualización de los datos obtenidos en Kibana
  - Motor de búsqueda para encontrar información concreta
  - Almacenamiento de datos a largo plazo
- **Conjunto de reglas de Wazuh.** El conjunto de reglas existentes en Wazuh es una versión mejorada y más actualizada de las proporcionadas en OSSEC, este aspecto nos proporciona mayor precisión y reglas añadidas para *Amazon Web Services* ,*docker* o *firewall*.
- **Documentación.** Disponibilidad de mayor cantidad de manuales y tutoriales para su implementación.

### 2.3.6 Tripwire

El software de código abierto Tripwire [45] es un HIDS centrado en monitorizar y alertar acerca de modificaciones no autorizadas sobre ficheros y directorios del *host* en el que se encuentre implementado.


Imagen 2.18: Logo de Tripwire

Esta herramienta se encuentra diseñada para funcionar en sistemas operativos Linux, sobre el cual es capaz de controlar todos los ficheros añadidos y modificados en el sistema. Un aspecto a destacar es que para cada uno de los ficheros almacena marcas de tiempo de sus modificaciones y accesos.

Respecto a las versiones existentes, Tripwire dispone de una versión de pago denominada *Tripwire Entreprise* la cual complementa la versión gratuita con más funcionalidades.

#### Funcionamiento de Tripwire

El funcionamiento de este HIDS se centra en el cálculo de valor *hash* para cada uno de los ficheros y su posterior comparación con el valor *hash* obtenido para el fichero original.

Como etapa inicial Tripwire almacena el estado normal de cada archivo existente en un archivo encriptado, este archivo almacena los permisos, modificaciones y accesos realizados junto con el valor *hash* original. Una vez obtenida la situación inicial, la herramienta se ejecuta de forma periódica para calcular el valor *hash* de los ficheros del sistema en ese determinado instante, siendo el último paso la realización de comparaciones con los valores *hash* originales con el fin de detectar posibles cambios.

#### Ventajas y desventajas

En la siguiente tabla 2.5 se nos van a indicar las principales ventajas y desventajas de desplegar Tripwire como HIDS en nuestro despliegue.

| VENTAJAS | DESVENTAJAS |
|---|---|
| Encripta su base de datos y archivo de configuración | No genera alertas en tiempo real |
| | No detecta errores previos a su instalación |
| | Enfocado al control de un número bajo de sistemas |
| | Linux como único SO |
| | Versión gratuita limitada |

Tabla 2.5: Ventajas y desventajas de Tripwire

### 2.3.7 AIDE

Aide [46] es un HIDS enfocado en el control de la integridad de ficheros. Esta herramienta registra el valor inicial de atributos importantes de los ficheros y directorios como los inodos, permisos o contenido. Durante su ejecución monitoriza el sistema de ficheros para determinar si algunos de los atributos mencionados han sido modificados.

Esta monitorización se realiza de forma similar a Tripwire puesto que para cada uno de los ficheros se realiza el cálculo de un valor *hash* que se utilizará para la comparación y detección de modificaciones. En el caso de esta herramienta el número de algoritmos *hash* que se pueden aplicar a los ficheros es mucho mayor.

#### Ventajas y desventajas

En la siguiente tabla 2.6 se nos van a indicar las principales ventajas y desventajas de desplegar Tripwire como HIDS en nuestro despliegue.

| VENTAJAS | DESVENTAJAS |
|---|---|
| Creación de base de datos de directorios del sistema | Sin alertas en tiempo real |
| Incluye gran cantidad de algoritmos como SHA-256, SHA-512,MD5,SHA-1 | La captura inicial no está encriptada,lo que puede suponer un problema de seguridad |
| Generación automática de un *report* diario | |

Tabla 2.6: Ventajas y desventajas de AIDE

### 2.3.8 SAMHAIN

La herramienta Samhain [47] es un HIDS diseñado para la monitorización de múltiples *hosts* con sistemas operativos distintos y que proporciona para cada uno de ellos gestión de ficheros *logs*, chequeo de la integridad de ficheros y detección de *rootkits*, puertos y procesos ocultos.

Samhain se ejecuta de forma continua en nuestro dispositivo mediante un *daemon*, este demonio irá realizando el cálculo de valores *hash* descrito anteriormente para la detección de modificaciones. Aunque el control de la integridad de los ficheros es un proceso similar a Tripwire y Aide, Samhain nos permite ofrecer una mayor cantidad de funcionalidades puesto que nos permite controlar los puertos abiertos en el sistema o encontrar ejecuciones no autorizadas por el uso de un SUID. Un SUID es un permiso de los ficheros que si se encuentra activo, permite a cualquier usuario heredar los permisos del usuario logeado sobre ese fichero.

Respecto al envío de las alertas generadas, Samhain dispone de un servidor central que recibirá todos los mensajes necesarios mediante conexiones encriptadas TCP.

#### Ventajas y desventajas

En la siguiente tabla 2.7 se nos van a indicar las principales ventajas y desventajas de desplegar Samhain como HIDS en nuestro despliegue.

| **VENTAJAS** | **DESVENTAJAS** |
|---|---|
| Conexión con servidor central encriptada | Mayor dificultad instalación |
| Independiente del SO del *host* | |
| Gestión centralizada de *logs* | |
| Dispone de documentación | |
| Puede realizar comprobaciones incrementales de los *logs* | |

Tabla 2.7: Ventajas y desventajas de Samhain

### 2.3.9 Comparativa general herramientas

Una vez hemos explicado en los previos apartados 2.3 una serie de herramientas de detección tanto NIDS como HIDS. Vamos a realizar una comparativa general entre ellas, en las que tendremos en cuenta los siguientes aspectos:

- Sistemas operativos que soporta.
- Licencia.
- Funcionalidad.
- Escalabilidad.
- Disponibilidad de manuales y soporte.

#### Sistemas operativos soportados

Respecto a los sistemas operativos que soporta cada herramienta, podemos observar los datos en la tabla 2.8. Hay que destacar el sistema operativo CentOS, ya que dispone de soporte para la mayoría de los IDS explicados.

En el caso de las herramientas NIDS Wazuh y Samhain, aunque se ejecute sobre uno de los sistemas operativos mencionados, se dispone de la posibilidad de instalar agentes en sistemas operativos distintos al del instalado en el *manager*.

| **IDS** | **Sistemas Operativos** |
|---|---|
| SNORT | FreeBSD, Microsoft Windows, Linux, CentOS |
| SURICATA | FreeBSD, Microsoft Windows, Linux, CentOS |
| ZEEK | FreeBSD, Linux, macOS,CentOS |
| WAZUH | FreeBSD, Microsoft Windows, Linux, CentOS |
| TRIPWIRE | GNU/Linux |
| AIDE | GNU/Linux |
| SAMHAIN | Unix, Cygwin/ Windows, Linux |

Tabla 2.8: Sistemas Operativos por IDS

#### Licencia

En este apartado, vamos a detallar el precio que nos costaría cada herramienta, junto con información sobre su licencia. Los datos mencionados se muestran en la tabla 2.9. Las licencias GPL(*General Public License*) y BSD(*Berkeley Source Distribution*), garantizan el acceso gratuito al software y permiten modificar, ejecutar y compartir el código.

| IDS | Licencia |
|---|---|
| SNORT | Licencia GPL, gratuito |
| SURICATA | Licencia GPL, gratuito |
| ZEEK | Licencia BSD, gratuito |
| WAZUH | Licencia GPL, gratuito |
| TRIPWIRE | Precio licencia completa desde 599$ a 6,995$ |
| AIDE | Licencia GPL, gratuito |
| SAMHAIN | Licencia GPL, gratuito |

Tabla 2.9: Licencias por IDS

#### Funcionalidad

En este apartado, vamos a mencionar en primer lugar las distintas características que nos proporciona cada herramienta y en segundo lugar la capacidad de las herramientas para detectar los distintos ataques mencionados en el apartado 2.1.3. En la siguiente tabla 2.10 mencionamos las principales características de cada herramienta.

| IDS | Funcionalidad |
|---|---|
| SNORT | Detección de alertas de red en tiempo real |
| SURICATA | Detección de alertas de red en tiempo real |
| ZEEK | Análisis del tráfico de red, detección mediante uso de lenguaje de *scripting* |
| WAZUH | Análisis información de múltiples dispositivos y en formatos distintos, sistema de respuesta en tiempo real |
| TRIPWIRE | Monitorizar y alertar de modificaciones en el sistema de ficheros |
| AIDE | Monitorizar y alertar de modificaciones en el sistema de ficheros |
| SAMHAIN | Monitorización de *hosts* |

Tabla 2.10: Funcionalidades por IDS

Respecto a la capacidad de detección de ataques, vamos a detallar las posibilidades que nos ofrecen Snort, Suricata y OSSEC. Podemos observar los datos en las siguientes tablas 2.11 y 2.12.

| REGLAS | BAD TRAFFIC | ATTACK-RESPONSES | EXPLOIT | ORACLE | SCAN | MYSQL | FINGER | SNMP | FTP | SMTP | TELNET | IMAP | RPC | POP2 | RSERVICES | POP3 | DOS | NNTP | DDOS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **SNORT** | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES |
| **SURICATA** | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES |
| **OSSEC** | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | NO | YES | YES | YES | YES | YES |

Tabla 2.11: Ataques detectables por IDS

| REGLAS | WEB ATTACKS | DNS | BACKDOORS | TFTP | SHELLCODE | WEB-EGI | POLICY | WEB-COLDFUSION | SQL | WEB-IIS | P2P | WEB-FRONTPAGE | ICMP-INFO | WEB-MISE | VIRUS | WEB-CLIENT | CHAT | WEB-PHP | MULTIMEDIA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **SNORT** | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES |
| **SURICATA** | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES |
| **OSSEC** | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | NO | YES | YES | YES | YES | YES |

Tabla 2.12: Ataques detectables por IDS

### Escalabilidad

Una de las características más importantes en una herramienta es la posibilidad que ofrece de escalar y por tanto adaptarse a entornos de mayor tamaño.

| **IDS** | **Escalabilidad** |
|---|---|
| SNORT | Enfocado a entornos de pequeño y mediano tamaño |
| SURICATA | Altamente escalable, enfocado a entornos de gran tamaño |
| ZEEK | Alta escalabilidad |
| WAZUH | Altamente escalable, cada Wazuh *manager* soporta hasta 14000 agentes |
| TRIPWIRE | Alta escalabilidad debido a su bajo impacto en el sistema |
| AIDE | Alta escalabilidad debido a su bajo impacto en el sistema |
| SAMHAIN | Alta escalabilidad debido a su bajo impacto en el sistema |

Tabla 2.13: Escalabilidad por IDS

Una de las conclusiones que podemos obtener, es que la mayoría de las herramientas HIDS analizadas escalan de forma satisfactoria mientras que en las herramientas NIDS tienen que cumplir que se adapten a las nuevas tecnologías tales como ipv6 o altas velocidades de red.

### Disponibilidad de manuales y soporte

Tanto para la instalación como para el funcionamiento, es importante que las herramientas empleadas dispongan de documentación de calidad que nos faciliten su utilización. A mayores supone un factor positivo, disponer de soporte o comunidad de desarrollo para esa herramienta, que nos permita solucionar los problemas de forma más sencilla. Para las herramientas analizadas podemos observar esta información en la tabla 2.14.

| IDS | Manuales y soporte |
|---|---|
| SNORT | Gran disponibilidad de documentación y comunidad de desarrollo |
| SURICATA | Gran disponibilidad de documentación y comunidad de desarrollo |
| ZEEK | Dispone de documentación propia y soporte a organizaciones |
| WAZUH | Dispone de documentación propia y soporte de pago por parte de los desarrolladores |
| TRIPWIRE | Dispone de manual de utilización en el sistema operativo Linux |
| AIDE | No dispone de documentación de calidad |
| SAMHAIN | Dispone de documentación oficial |

Tabla 2.14: Documentación por IDS

# 3. Sistema de monitorización

## 3.1 Security Onion

En el ámbito empresarial se dispone de una gran cantidad de herramientas destinadas a la detección de posibles problemas de seguridad. Las más empleadas en la actualidad las hemos mencionado en el apartado 2.3, en el cual hemos destacado algunas como Zeek, Suricata, Snort o Wazuh. Cada una de las opciones mencionadas funciona individualmente de forma muy satisfactoria, pero no nos permiten observar una situación general de la seguridad.

Es en este punto en el que surge la opción de la utilización de Security Onion, ya que nos permite combinar las funcionalidades tanto de los IDS como de los SIEM. Siendo mas específicos, la herramienta nos permitirá emplear todas las herramientas de detección deseadas además de presentarnos los datos obtenidos de forma clara y sencilla. De esta forma nos facilita el análisis sobre los datos obtenidos y con ello nos permite aumentar las posibilidades de encontrar posibles vulnerabilidades en nuestra red.

Se han considerado otras opciones que nos ofrezcan funcionalidades similares a Security Onion pero han sido descartadas debido a que no integraban las herramientas de detección que queríamos emplear o debido a su alto precio de compra. Poniendo un ejemplo concreto, podemos destacar *SolarWinds Security Event Manager*, que aunque es una herramienta muy potente, ha sido rechazada debido a su precio de licencia (2.127€).

Security Onion [48] es una distribución Linux gratuita y *open-source* enfocada a la caza de amenazas, monitorización de seguridad y gestión de *logs*. Incluye una gran variedad de herramientas de seguridad como TheHive, CyberChef, Elasticsearch, Logstash, Kibana, Suricata, Snort, Zeek, Wazuh. Su

implementación nos proporciona la posibilidad de construir un sistema que sea capaz de proveernos de visibilidad y contexto de todo el tráfico que fluye por nuestra red.

Imagen 3.1: Logo de Security Onion

Una vez conocemos Security Onion, voy a detallar las aportaciones que obtenemos debido a su implementación mediante el despliegue realizado en la red de ejemplo presentada en la imagen 3.2

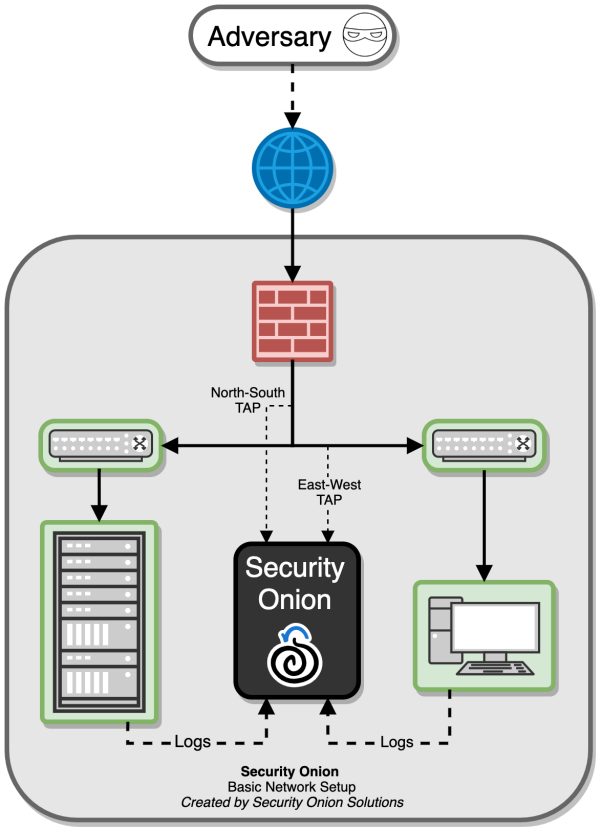


Imagen 3.2: Diagrama explicativo Security Onion [48]

En nuestra red ejemplo podemos observar que disponemos de *firewall*, servidores y *workstations* y dos tipos distintos de tráfico. El primero de ellos denominado *North-South TAP* se trata de tráfico de red proveniente o con dirección a un elemento que reside fuera de nuestro entorno, por otra parte, el segundo denominado *East-West TAP* es tráfico de red que procede y tiene como dirección los elementos de nuestra red.

Mediante la recopilación de *logs* de estos dos tipos de tráfico, Security Onion es capaz de detectar problemas a lo largo de nuestra red y en estaciones individuales al mismo tiempo.

### 3.1.1 Herramientas integradas

Para diferenciar de forma sencilla las distintas herramientas que componen Security Onion, vamos a clasificarlas basándonos en las 4 funciones generales que realiza esta distribución, las cuales vamos a mencionar a continuación:

- **Obtención de información sobre el tráfico de red.** Para proveernos de visibilidad del tráfico de red, Security Onion dispone de Suricata y Zeek ya mencionadas en la sección 2.3.
- **Obtención de información en los *hosts*.** Respecto a la monitorización de *host*, se dispone de varias posibilidades, la más empleada es el HIDS Wazuh explicado anteriormente en la sección 2.3. Su uso extendido como HIDS se debe que es la solución actual en el mercado que engloba más funcionalidades.

  A mayores disponemos de otras opciones para la obtención de datos de *host* que son Osquery, Beats o simplemente obtener los datos via Syslog.
- **Recogida y almacenamiento de *logs* generados.** Todos los *logs* generados debido al funcionamiento de las herramientas NIDS y HIDS son gestionados por el paquete ELK Stack, el cual explicaremos a continuación. Podemos destacar también el uso de Redis para la gestión de las colas de *logs* que va a crear el paquete ELK Stack.
- **Visualización de los datos obtenidos.** Una vez disponemos de todos los *logs* Security Onion, dispone de múltiples herramientas para facilitarnos su comprensión entre ellas nos encontramos Security Onion Console (SOC), TheHive, Kibana, CyberChef y Playbook.

Una vez clasificadas las distintas herramientas vamos a describir cada una de las mencionadas que no hayan sido explicadas anteriormente.

#### ELK Stack

El proyecto ELK Stack [49] es una colección de los 4 productos de código abierto ElasticSearch, Logstash, Kibana y Beats.

Imagen 3.3: Logo de ELK Stack

La aplicación de estos 4 productos nos permite realizar una gestión centralizada de los *logs* logrando así transformar sistemas heterogéneos y distribuidos en sistemas más sencillos. La simplificación del sistema nos permite una mayor facilidad de monitorización y con ello una mejor gestión de la seguridad.

La mejor forma de definir ELK Stack es mencionando las siguientes 4 capacidades claves que nos aporta:

- **Agregación.** Capacidad para recolectar *logs* de fuentes de datos distintas.
- **Procesamiento.** Capacidad de transformar los mensajes presentes en los *logs* en datos de mayor relevancia para su análisis.
- **Almacenamiento.** Capacidad de almacenar los datos durante un periodo de tiempo extendido.
- **Análisis.** Capacidad de visualizar y realizar consultas sobre los datos.

Cada una de estas funcionalidades se realiza mediante la aplicación de Beats, Logstash, Elastic-Search y Kibana de la forma que vemos en la imagen 3.4.

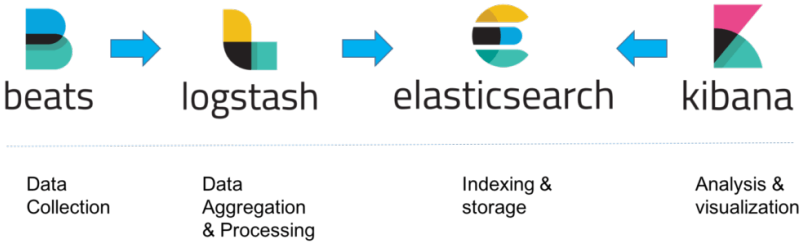


Imagen 3.4: Funcionamiento ELK Stack

Siguiendo la línea temporal de la imagen 3.4, podemos determinar que en primer lugar Beats se encarga de la recolección de datos en *host*, Logstash de la agregación y procesamiento de los *logs* obtenidos, ElasticSearch se encarga de indexar y almacenar estos datos y por último Kibana nos proporciona una interfaz de usuario para visualizar y realizar consultas sobre los resultados obtenidos.

En el caso concreto de Beats, vamos a detallar que dispone de un software incluido denominado FileBeat que se encarga de obtener los *logs* del sistema de ficheros del *host* en el que se encuentra. A mayores de los mencionados, la distribución Security Onion emplea el software Curator. Curator se centra en gestionar los índices que emplea ElasticSearch para el almacenamiento de los datos, de forma que se logre la solución más eficiente posible.

### Osquery

Osquery [50] es un *framework* para Windows, OS X (macOS), Linux, y FreeBSD encargado de la recolección de datos del sistema operativo.


Imagen 3.5: Logo de Osquery

Su funcionamiento se basa en considerar el sistema operativo como una base de datos relacional de alto rendimiento. De esta forma es capaz de realizar consultas para obtener datos del sistema operativo, por ejemplo podríamos obtener los procesos existentes en el sistema o los dispositivos USB conectados a él.

### Redis

Redis [51] es una base de datos en memoria que nos permite acceder de forma rápida a datos necesarios para el funcionamiento de Security Onion en algunas arquitecturas.


Imagen 3.6: Logo de Redis

Su principal función en esta distribución es almacenar en memoria las colas de los *logs* que va a utilizar el paquete ELK Stack. Estas colas se emplean para poder planificar el tratamiento de los *logs*, pudiendo ir obteniendo la cantidad de *logs* deseada en cada momento. Con el fin de conocer el aumento de eficiencia que se logra al implementar Redis en vez de una base de datos relacional, se ha estudiado un *benchmark* [52] que establece una comparación entre el rendimiento de MySQL y de Redis.

El *benchmark* estudiado, el cual podemos observar en la tabla 3.1, consiste en comparar el tiempo total en el que se realiza un número de peticiones a dos bases de datos con MySQL en la que la segunda implementa a mayores Redis.

| N.º de peticiones | Tiempo total MySQL(s) | Tiempo total Redis(s) |
|---|---|---|
| 1000 | 0.0856 | 0.2618 |
| 10000 | 0.7908 | 1.9814 |
| 100000 | 7.4592 | 15.8395 |
| 1000000 | 67.7995 | 59.4885 |
| 10000000 | 679.9420 | 512.9122 |

Tabla 3.1: *Benchmark* Redis vs Mysql

Tras el estudio de los datos, podemos observar que la implementación de Redis disminuye de forma notable el tiempo necesario para un número de peticiones elevado. Si estudiamos Redis como herramienta independiente podemos determinar que es capaz de realizar 110.000 peticiones SETs por segundo, y 81.000 peticiones GETs por segundo.

### Security Onion Console(SOC)

La herramienta de Security Onion dispone de una consola a la que podremos acceder mediante el buscador web. Esta consola, mostrada en la imagen 3.7, nos ofrecerá la posibilidad de acceder a las distintas herramientas presentes en Security Onion.

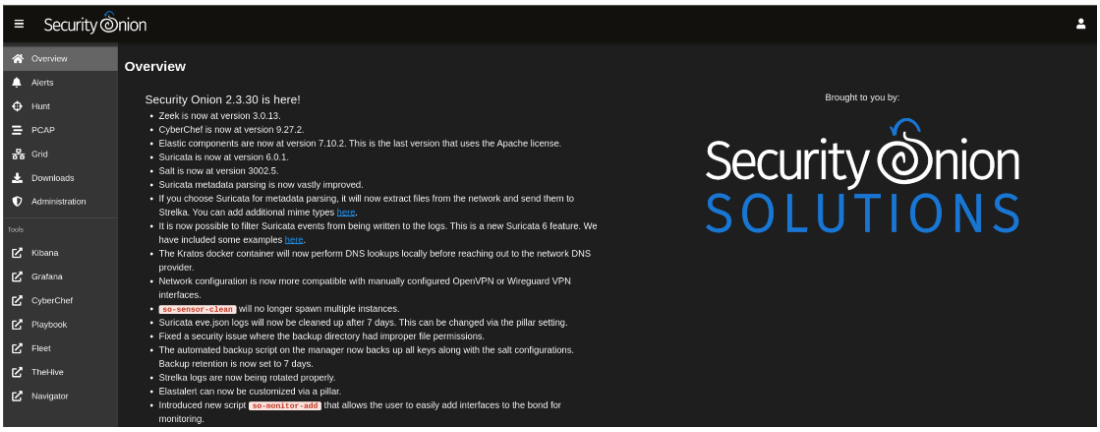


Imagen 3.7: Security Onion Console

De esta consola vamos a detallar las opciones Alert, Hunt, PCAP y Grid.

- **Alert.** Alert nos presenta una interfaz para ver todas las alertas generadas por los NIDS y HIDS de nuestro despliegue Security Onion. Además, nos permite realizar un filtrado de estas alertas por varios parámetros.
- **Hunt.** En el caso de Hunt, esta opción nos presenta una interfaz para ver todos los *logs* obtenidos por los IDS en nuestro despliegue Security Onion.
- **PCAP.** La opción PCAP nos permite observar más en detalle los paquetes que recorren nuestra red.

- **Grid.** En el apartado Grid se nos muestra una interfaz con todos los nodos conectados en el despliegue Security Onion y una serie de datos para cada uno de ello.

#### TheHive

TheHive es una plataforma de respuesta a incidentes de seguridad. Su funcionamiento se basa en permitir la creación de expedientes sobre las alertas que podemos ver en las opciones de Alert y Hunt.


Imagen 3.8: Logo de TheHive

Podríamos definir un expediente como un documento que permite ser compartido y editado por los distintos técnicos de seguridad, en el cual se mostrara toda la información existente para la alerta que queremos estudiar. Este software permitirá a todos los expertos trabajar de forma simultánea, con el fin de solucionar de forma conjunta el problema que causa la alerta.

#### CyberChef

CyberChef [53] es una aplicación web centrada en ofrecer la posibilidad de realizar operaciones avanzadas sobre los datos. Entre las operaciones que incluye podemos destacar la posibilidad de aplicar algoritmos de encriptación y realizar el cálculo de valores *hash* y *checksums*.

En la distribución Security Onion, esta aplicación se enfoca principalmente en permitirnos realizar operaciones sobre los datos de los paquetes obtenidos en la opción PCAP.

#### PlayBook

Como última herramienta vamos a explicar PlayBook, que se trata de una aplicación web que nos permite crear un libro en el que podremos almacenar cada ataque detectado como una entrada. Playbook nos permitirá definir una guía con las medidas empleadas para su detección y solución, de esta forma podremos consultar los pasos a realizar en caso de ataques futuros.

### 3.1.2 Arquitecturas

En la siguiente sección vamos a desarrollar los distintos despliegues existentes de Security Onion. La implementación de uno u otro dependerá de las funcionalidades que busquemos o de los recursos disponibles. Vamos a distinguir entre 4 arquitecturas denominadas *import*, *evaluation*, *standalone* y *Standalone*.

#### *Import*

La arquitectura *import* se caracteriza por ser la más sencilla ya que solo necesita un único nodo que ejecute un capturador de paquetes. Una vez el capturador de paquetes está funcionando, los IDS Suricata y Zeek se encargan del análisis de los paquetes y la posterior generación de *logs*. Como último paso, los *logs* son recogidos por Filebeat y mandados a ElasticSearch para su análisis e indexación, proporcionándonos así la posibilidad de ver los resultados mediante la consola de Security Onion o algún otro visualizador de datos.

#### *Evaluation*

Esta arquitectura indicada en la imagen 3.9 requiere la necesidad de disponer de una interfaz de red dedicada a obtener el tráfico que han ido obteniendo un TAP o *span port*. Un TAP es un dispositivo físico o software que puede ver y almacenar todo el tráfico de la red en tiempo real y un *span port* es un puerto de un *switch* dedicado a realizar y enviar copias de todo el tráfico de red que ha pasado por ese *switch*.

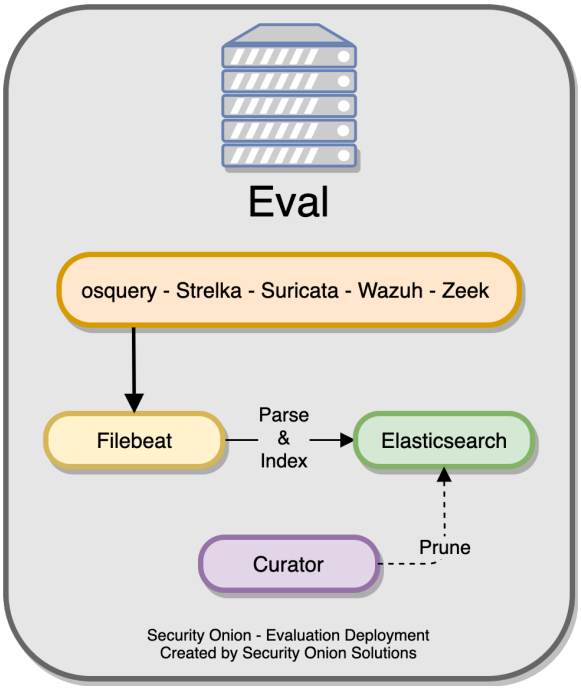


Imagen 3.9: Arquitectura *evaluation* [48]

En esta arquitectura, los *logs* son obtenidos mediante la utilización de las herramientas NIDS y HIDS. Estos *logs* resultantes, son tratados por Filebeat y enviados a ElasticSearch de forma similar a como se realiza en un nodo *import*. Esta arquitectura, incluye a mayores Curator para el tratamiento de los índices que emplea ElasticSearch en el almacenamiento de los logs.

#### *Standalone*

La arquitectura *standalone*, la cual se puede observar en la imagen 3.10, funciona de forma similar a una arquitectura *evaluation* respecto a la obtención de paquetes y de *logs*. Una vez disponemos de los *logs*, estos son tratados de forma distinta ya que se mandan en primer lugar a Logstash para su posterior tratamiento por Elasticsearch. Logstash aparece en la arquitectura en dos ocasiones, esto se debe a que en primer lugar se emplea para mandar datos a Redis para la creación de las colas de *logs*. Estas colas van a ser empleadas por Logstash en su segunda aparición, en la cual su función va a ser ir obteniendo *logs* de la cola y enviárselos a ElasticSearch.

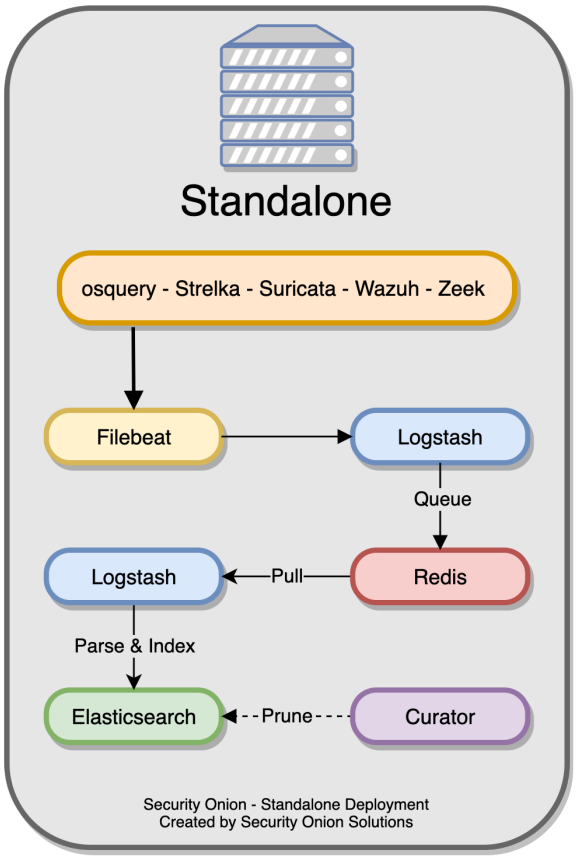


Imagen 3.10: Arquitectura *standalone* [48]

#### Distribuida

El despliegue de una arquitectura distribuida se basa en la partición de las funcionalidades en distintos nodos. Esta división, aunque aumenta el coste, garantiza una mayor escalabilidad y rendimiento. En una arquitectura distribuida simple como la indicada en la imagen 3.11, se definen 3 tipos de nodos:

- **Nodos *forward* o Nodos sensores.** Ejecutan los distintos IDS configurados en nuestro entorno para la obtención de los *logs*. Posteriormente se encarga de su envio al *manager Node* mediante la utilización de Filebeat.

- ***Manager.*** Mediante el uso de Elasticsearch y Redis se ocupa de realizar las colas de *logs* y el balance de carga entre los distintos *Search Nodes*.
- **Nodos *search*.** Ejecutan una instancia local de ElasticSearch las cuales van obteniendo *logs* de la cola para su indexación y almacenamiento.

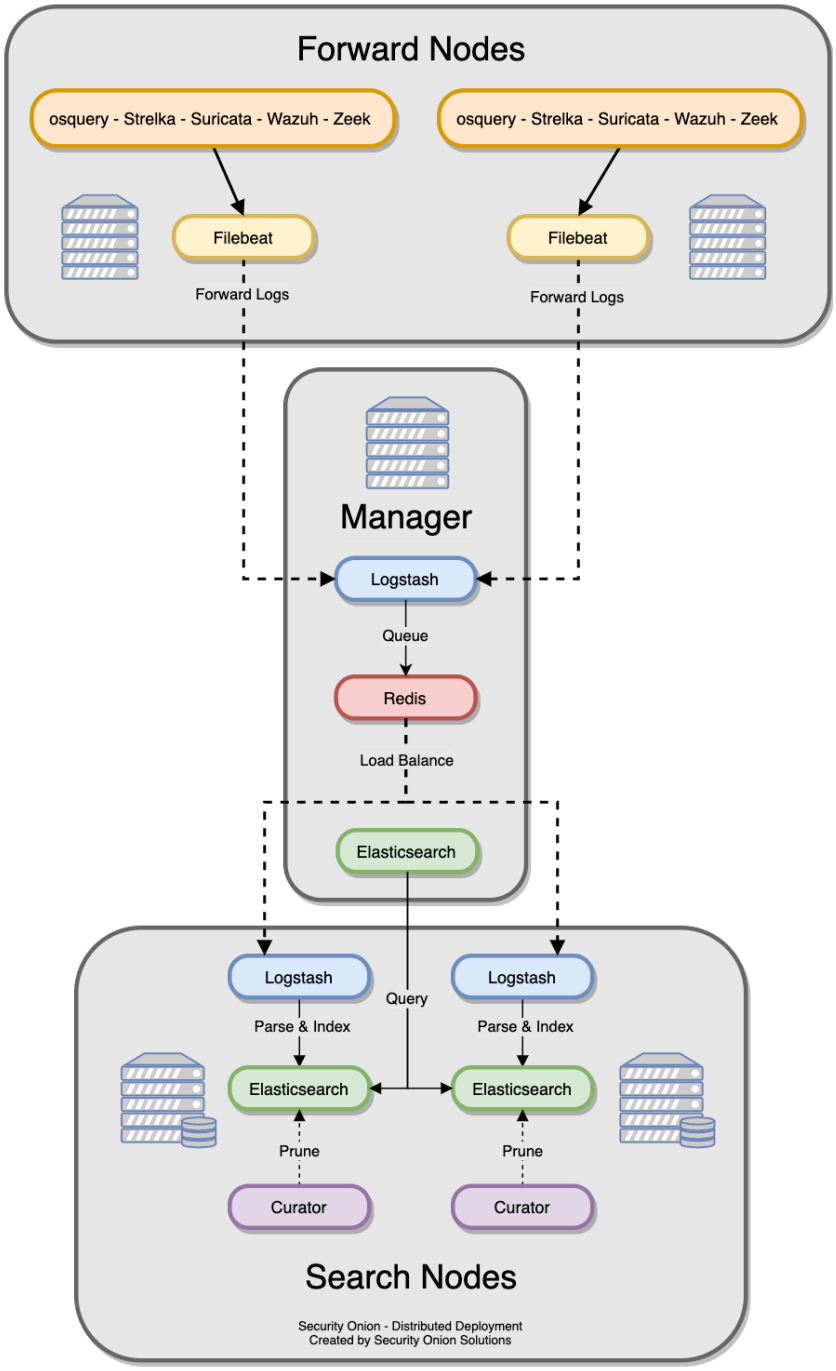


Imagen 3.11: Arquitectura distribuida [48]

Una vez los datos se hayan obtenido por los nodos sensores y se encuentren almacenados en los nodos *search*, podrán ser consultados a través del *manager*.

A mayores de los anteriormente mencionados, existen dos tipos de nodos enfocados a circunstancias

específicas de la red que son los siguientes:

- ***Heavy node.*** Se caracteriza por ser una unión de *Forward* y *Search node* encargándose por tanto de la obtención de *logs* y de su almacenamiento.
- ***Fleet Standalone Node.*** Este nodo está enfocado a emplearse cuando existe una gran cantidad de *hosts* que implementan osquery. En los casos en los que se cumpla esta condición, podremos separar la gestión de los *hosts* que ejecuten osquery del *manager* y encargar esta función a un nodo *fleet* independiente.

### 3.1.3 Ventajas y desventajas de Security Onion

#### Ventajas

- *Open-source.*
- Permite una gran flexibilidad para la gestión de la seguridad de red (Permite personalización a cada entorno y requisitos).
- Dispone de herramientas para el análisis en *host* y del tráfico red que pueden ser gestionadas por IDS ya instalados en el SO.
- Actualizaciones de forma regular.

#### Desventajas

- Solo se encarga de la detección, no dispone de características de un IPS ( *Intrusion Prevention System*).
- No soporta WIFI para la gestión.
- Necesidad de requisitos técnicos en el personal para un uso eficiente.
- No realiza *backups* automáticos de los archivos de configuración, excepto los archivos de reglas.

# 4. Análisis y Diseño

En esta sección, se describirá de forma detallada el diseño de la plataforma de detección que se ejecutará sobre el Departamento de Informática de la Universidad de Valladolid. Como paso previo, se analizará la elección de todas las herramientas empleadas basándonos en las comparaciones teóricas realizadas en la sección 2.3.

Una vez elegidas las herramientas, detallaremos el sistema de virtualización empleado, el diseño de Security Onion implementado y el hardware necesario en cada máquina.

## 4.1 Tecnologías elegidas

Con el fin de desarrollar una plataforma de detección que sea capaz de descubrir ataques sobre nuestra red, además de poder mostrarnos los datos para su posterior análisis, hemos considerado la implementación de herramientas que realicen las funcionalidades de los IDS y los SIEM. En concreto, vamos a desarrollar la solución escogida respecto a los siguientes tipos de herramientas:

- Herramientas NIDS (*Network Intrusion Detection System*).
- Herramientas HIDS (*Host Intrusion Detection System*).
- Herramientas SIEM (*Security information and event management*).

#### Herramientas NIDS.

Según hemos definido en el apartado 2.1.4, actualmente podemos distinguir entre NIDS basados en firmas e NIDS basados en anomalías, cada uno de ellos encargados de la detección de un tipo de ataques. Los NIDS basados en firmas, se encargarán de la detección de ataques ya existentes mediante

la utilización de las reglas registradas en una base de datos. Por otro lado, los NIDS basados en anomalías pueden detectar ataques nuevos que no estén contemplados en las reglas que emplean los IDS basados en reglas. Debido a sus distintas posibilidades de detección, será de vital importancia para la detección de todo tipo de ataques la elección de herramientas NIDS que utilicen ambos enfoques.

- **Herramienta NIDS basada en firmas.** Como herramientas NIDS basadas en firmas hemos descrito Snort y Suricata puesto que se tratan de las más implementadas en la actualidad. Ambas herramientas son gratuitas, disponen de una base de datos de reglas actualizada y customizable y disponen de manual con lo que cualquiera de ellas sería una buena opción.

  Para nuestro despliegue nos vamos a decantar por la utilización de Suricata ya que según los estudios analizados nos ofrece un mayor rendimiento en entornos de mayor tamaño, aunque ello conlleve un mayor uso de recursos. Como aspecto diferencial respecto a Snort, Suricata soporta IPV6 y permite multihilos. Este último aspecto permite a la plataforma una mayor escalabilidad, que sería un aspecto fundamental en caso de querer añadir la plataforma en otros departamentos de la universidad.
- **Herramientas NIDS basadas en anomalías.** Para la detección basada en anomalías hemos escogido la opción de Zeek. Este software nos proporcionará una gran cantidad de ficheros *logs* de la actividad de la red además de un lenguaje de *scripting* que nos permitirá aumentar las posibilidades de detectar ataques.

#### Herramientas HIDS

Respecto a la herramienta HIDS la elegida es OSSEC Wazuh, esta variante de OSSEC nos proporciona en su versión *open source* un software gratuito multiplataforma, con capacidad de responder en tiempo real a las alertas generadas y que actualmente está considerada la mejor solución HIDS en el mercado.

En comparación con otro HIDS como AIDE o Tripwire, Wazuh destaca puesto que dispone de más funcionalidades y no se centra únicamente en garantizar la integridad de los ficheros del sistema.

Como otro de sus mayores puntos fuertes, Wazuh sobresale por el uso de un modelo de agentes y servidor, el cual nos permitiría instalar un agente muy ligero en cada uno de los dispositivos a monitorizar.

#### Herramientas SIEM

En el caso de las herramientas SIEM disponemos de una gran cantidad de ellas como las mencionadas Splunk, ArcSight, IBMQRadar o OSSIM que podrían realizar perfectamente la función de proporcionarnos una visión centralizada de la información de seguridad obtenida por los IDS.

La característica más importante que buscamos para nuestra solución, se centra en su capacidad de

integrarse con las otras herramientas de seguridad escogidas que como hemos visto son Suricata,Zeek y OSEEC Wazuh.

Teniendo en cuenta esta necesidad de integración, hemos escogido para realizar la función de herramienta SIEM a la distribución Security Onion, puesto que nos proporciona todos los IDS mencionados así como las herramientas para la visualización de los datos.

## 4.2 Sistema de virtualización

Vamos a emplear el sistema de virtualización del Departamento de Informática para ejecutar las máquinas necesarias en el despliegue de Security Onion. Hemos necesitado 9 máquinas virtuales distintas que se ejecutaran en nodos del sistema de virtualización diferenciados. Como podemos ver en la imagen 4.1 se han empleado los nodos c1blade2, c1blade4, deckard, taffy y tyrell. Cada uno de estos nodos, ejecutaran las máquinas necesarias para nuestro despliegue de Security Onion.

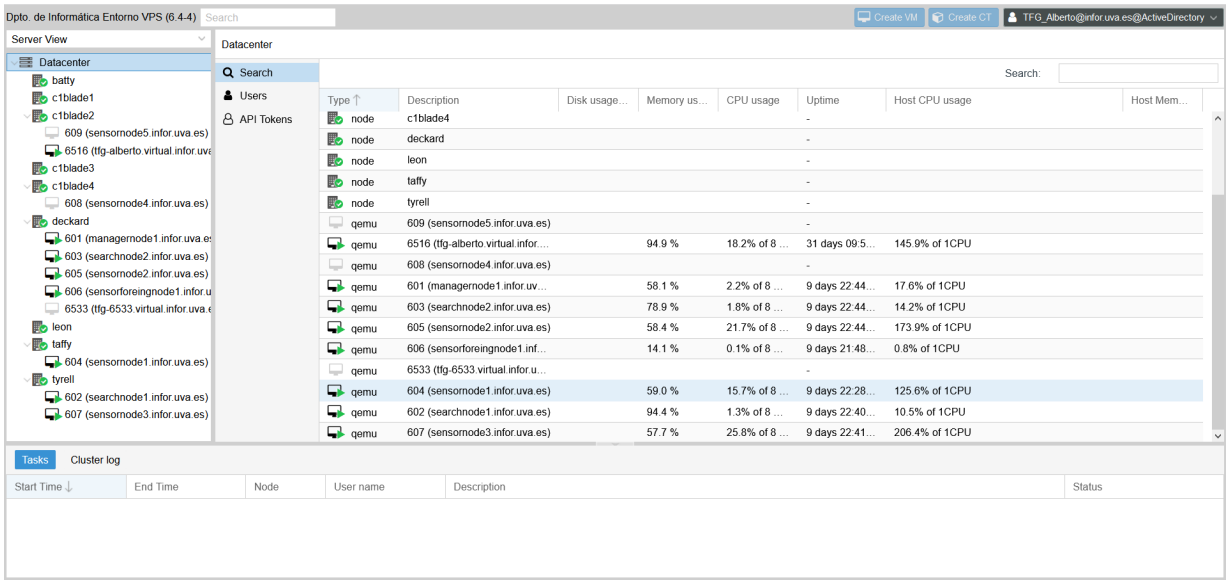


Imagen 4.1: Sistema de virtualización

El hardware necesario en cada máquina ha sido establecido de forma que se emplee la menor cantidad de recursos posible. Para lograr este objetivo, se ha realizado una instalación personalizada estableciendo el valor de disco y memoria en función de las necesidades en tiempo real de Security Onion. Siendo más concretos, hemos fijado discos diferenciados con el mínimo tamaño necesario que apunten a los directorios del sistema operativo que más crecen a lo largo del uso de Security Onion.

De esta forma hemos logrado reducir los requisitos mínimos fijados en la documentación de Security Onion, permitiendo en el caso de que un directorio concreto se esté quedando sin espacio, disponer de la posibilidad de aumentarlo en caliente, es decir, sin necesidad de apagar la máquina. En el momento

que decidamos la arquitectura de Security Onion elegida, mencionaremos los recursos necesarios para cada nodo.

## 4.3 Diseño de Security Onion

Una vez hemos descrito en la sección 3.1.2 las distintas arquitecturas en las cuales se puede desplegar Security Onion, es de vital importancia escoger aquella que mejor encaja con nuestro entorno.

Debido a que queremos demostrar la posibilidad que tiene la solución propuesta de escalar unido a que disponemos de recursos suficientes tanto de memoria como CPU y disco, hemos escogido una arquitectura distribuida que es la que mejor satisface estas necesidades. Gracias a esta arquitectura se nos posibilita la opción de añadir los nodos tanto de almacenamiento como sensores que sean necesarios en cada momento, unido a que nos proporciona un mayor rendimiento que cualquier otra arquitectura.

Como ya hemos mencionado, este tipo de arquitectura dispone de 3 tipos de nodos que resumiendo según su funcionamiento son:

- **Nodo *manager.*** Centrado en la gestión general del sistema.
- **Nodo *search.*** Encargado del almacenamiento de los datos.
- **Nodo sensor.** Se centra en la obtención de los *logs*.

A mayores de los indicados dispondremos de un nodo que llamaremos máquina de analista. Esta máquina nos permitirá disponer de un buscador web para acceder a la consola de Security Onion(SOC) y a todas las herramientas de visualización de las que disponemos.

Aunque la instalación de esta máquina de analista es independiente al rol del nodo, con lo que podría realizarse en un nodo *manager*, *search* o *sensor*, se ha decidido la utilización de una máquina distinta para que exista la mayor independencia posible entre cada una de las funcionalidades.

El diseño distribuido de Security Onion constara de 8 nodos más la máquina de analista como podemos observar en la imagen 4.2.

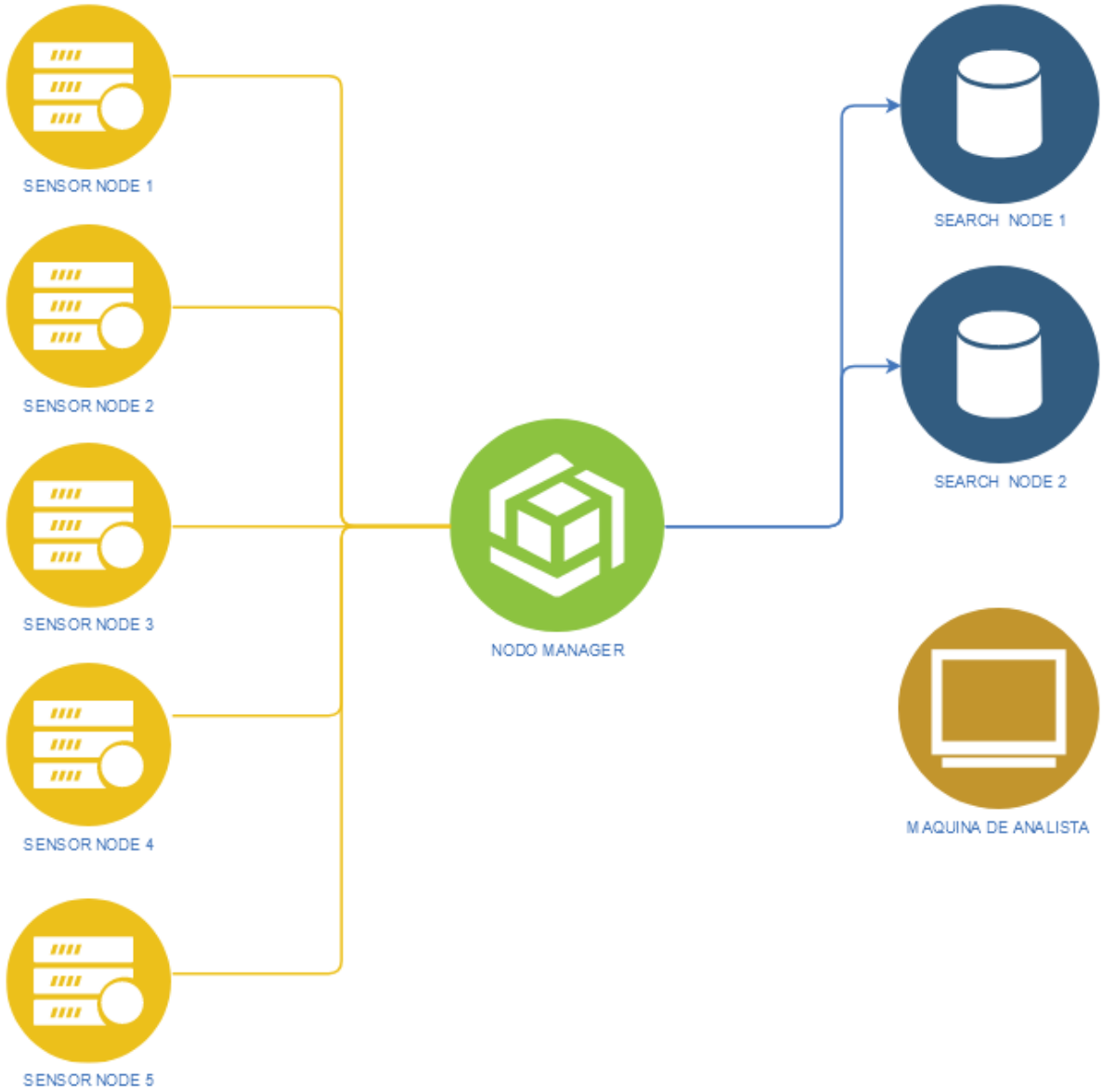


Imagen 4.2: Diseño Security Onion

La elección de este número de nodos se ha realizado para simular un sistema lo más real posible a una implementación en explotación. Además, teniendo en cuenta los recursos de los que disponíamos, se ha decidido por representar 5 sensores distintos que están situados en distintos nodos del sistema de virtualización.

Es en este momento en el que conocemos todos los nodos de los que dispondremos, cuando vamos a especificar el hardware necesario para cada uno de ellos. Aunque los distintos nodos empleen recursos distintos, todos dispondrán de la ISO del sistema operativo CentOS, esto se debe a que es el indicado como preferible en la documentación de Security Onion.

#### Nodo *manager*

Los recursos necesarios para el nodo *manager* son los mostrados en la tabla 4.1. Podemos destacar que solo necesita una interfaz de red de monitorización, esta interfaz es la empleada para comunicarse

entre todos los nodos que forman el sistema distribuido.

| Nombre | Dirección IP | Memoria | CPU | Interfaz de red |
|---|---|---|---|---|
| managernode1.infor.uva.es | 10.124.60.1 | 32 GB | 8 Núcleos | ETH0 |

Tabla 4.1: Nodo *manager*

Como hemos especificado anteriormente, hemos creado distintos discos para cada máquina, en el caso del nodo *manager* seguirán el esquema mostrado en la tabla 4.2

| Disco | Espacio | Directorio |
|---|---|---|
| Disco 1 | 10GB | / |
| Disco 2 | 20GB | /nsm |
| Disco 3 | 2GB | /tmp |
| Disco 4 | 25GB | /var/lib/docker |

Tabla 4.2: Discos nodo manager

Los directorios especificados para el disco 2 y el disco 4 se deben en el caso del disco 2 a que todos los *logs* de las herramientas IDS se almacenan en el directorio /nsm y en el caso del disco 4 se debe a que cada una de las herramientas se ejecutan mediante el despliegue de Dockers, los cuales se almacenan en el directorio /var/lib/docker. Docker [54] es una tecnología de creación de contenedores que permite la creación y uso de contenedores Linux. Un contenedor se define como una pequeña parte aislada del sistema que dispone de lo esencial para ejecutar un programa. En nuestro despliegue estos contenedores van a ejecutar las herramientas que emplea Security Onion, por ejemplo los IDS (Suricata, Zeek, Wazuh), el paquete ELK o las herramientas de visualización (PlayBook, TheHive).

#### Nodos *search*

En nuestro diseño disponemos de 2 nodos *search* que dispondrán de los recursos mostrados en las tablas 4.3, 4.4. La estructura de discos necesaria es igual a la del nodo *manager*.

| Nombre | Dirección IP | Memoria | CPU | Interfaz de red |
|---|---|---|---|---|
| searchnode1.infor.uva.es | 10.124.60.2 | 16 GB | 8 Núcleos | ETH0 |

Tabla 4.3: Nodo *search* 1

| Nombre | Dirección IP | Memoria | CPU | Interfaz de red |
|---|---|---|---|---|
| searchnode2.infor.uva.es | 10.124.60.3 | 16 GB | 8 Núcleos | ETH0 |

Tabla 4.4: Nodo *search* 2

#### Nodos sensores

Todos los nodos sensores van a necesitar los mismos recursos, esto se debe a que se encargarán de monitorizar tráficos de red similares. En diferencia con el nodo *manager*, los sensores dispondrán de una interfaz de *sniffing*. Esta interfaz dispone de la capacidad de capturar los paquetes tanto de entrada como de salida. A continuación, vamos a especificar los 5 nodos sensores mediante las tablas 4.5, 4.6, 4.7, 4.8 y 4.9.

| **Nombre** | **Dirección IP** | **Memoria** | **CPU** | **Interfaz de red** |
|---|---|---|---|---|
| sensornode1.infor.uva.es | 10.124.60.4 | 16 GB | 8 Núcleos | ETH0, ETH1 |

Tabla 4.5: Nodo sensor 1

| **Nombre** | **Dirección IP** | **Memoria** | **CPU** | **Interfaz de red** |
|---|---|---|---|---|
| sensornode2.infor.uva.es | 10.124.60.5 | 16 GB | 8 Núcleos | ETH0, ETH1 |

Tabla 4.6: Nodo sensor 2

| **Nombre** | **Dirección IP** | **Memoria** | **CPU** | **Interfaz de red** |
|---|---|---|---|---|
| sensornode3.infor.uva.es | 10.124.60.7 | 16 GB | 8 Núcleos | ETH0, ETH1 |

Tabla 4.7: Nodo sensor 3

| **Nombre** | **Dirección IP** | **Memoria** | **CPU** | **Interfaz de red** |
|---|---|---|---|---|
| sensornode4.infor.uva.es | 10.124.60.8 | 16 GB | 8 Núcleos | ETH0, ETH1 |

Tabla 4.8: Nodo sensor 4

| **Nombre** | **Dirección IP** | **Memoria** | **CPU** | **Interfaz de red** |
|---|---|---|---|---|
| sensornode5.infor.uva.es | 10.124.60.9 | 16 GB | 8 Núcleos | ETH0, ETH1 |

Tabla 4.9: Nodo sensor 5

Respecto a la necesidad de discos de los sensores, mostrados en la tabla 4.10, cabe destacar que son los encargados de almacenar la mayoría de los datos que se obtienen y por tanto hemos aumentado el disco del directorio /nsm.

| **Disco** | **Espacio** | **Directorio** |
|---|---|---|
| Disco 1 | 10GB | / |
| Disco 2 | 30GB | /nsm |
| Disco 3 | 2GB | /tmp |
| Disco 4 | 25GB | /var/lib/docker |

Tabla 4.10: Discos nodo sensor

### Máquina de analista

La máquina de analista no requiere de una gran cantidad de recursos, ya que su funcionalidad es únicamente permitirnos utilizar las herramientas y visualizar los datos. Como la cantidad de recursos no era relevante se ha realizado una clonación con los mismos recursos de un nodo *manager*, los cuales podemos observar en la tabla 4.11.

| **Nombre** | **Dirección IP** | **Memoria** | **CPU** | **Interfaz de red** |
|---|---|---|---|---|
| sensorforeignnode1.infor.uva.es | 10.124.60.6 | 16 GB | 8 Núcleos | ETH0 |

Tabla 4.11: Máquina analista

# 5. Despliegue

En esta sección vamos a describir los pasos necesarios para el despliegue del diseño descrito en la sección 4. Siendo más específicos, el conjunto de pasos se centrará en la instalación y puesta en funcionamiento de los nodos *manager*, *search*, sensor y la máquina de analista.

## 5.1 Configuraciones iniciales

Como paso previo a la instalación de cada uno de los nodos, las maquinas empleadas deben disponer de una configuración común.

En primer lugar, todos los nodos se deberán configurar sobre una máquina que ejecute el sistema operativo CentOS y en cada uno de ellos será necesario establecer las configuraciones de red que se requieren para su tipo de nodo. Para la instalación de CentOS dispondremos del archivo ISO en la documentación de Security Onion.

Con el fin de simplificar el despliegue, hemos creado un modelo que sirva como base para la creación de las máquinas que forman el diseño. Este modelo dispondrá de 16 GB de memoria, 8 núcleos y 4 discos que serán de 2,10, 20 y 25 GB, siendo la estructura de discos similar a la mostrada anteriormente en la tabla 4.2.

Una vez disponemos del modelo base, se ha empleado el sistema de virtualización para clonar esta máquina y de esta forma obtener todas las máquinas virtuales empleadas en nuestra plataforma de detección. Para cada una de las máquinas clonadas únicamente será necesario establecer el nombre que le corresponda según el diseño y su configuración de red. Este último aspecto relativo a la configuración de red constara de la asignación de su dirección IP, la comprobación de que puede conectarse por SSH

con las otras máquinas y la creación de las interfaces ETH1 en caso de necesitarlas.

La instalación del tipo de nodo (*manager*, *search*, sensor) que ejecutara cada una de las máquinas virtuales, se realizará mediante la utilización del ejecutable *so-setup-network*, situado en el directorio /SecurityOnion. Tras su ejecución nos mostrará una serie de ventanas que seguirán el mismo formato al mostrado en la imagen 5.1. Esta interfaz nos permitirá escoger la arquitectura de Security Onion deseada y el tipo de nodo a instalar junto con su configuración.

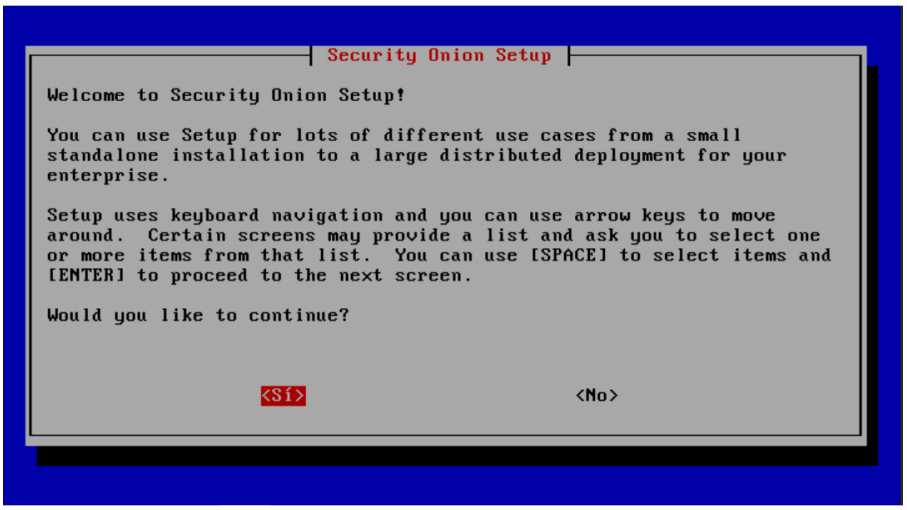


Imagen 5.1: Ventana inicial instalación nodo

Para los nodos descritos en nuestro diseño, escogeremos una arquitectura distribuida como podemos mostrar en la imagen 5.2.

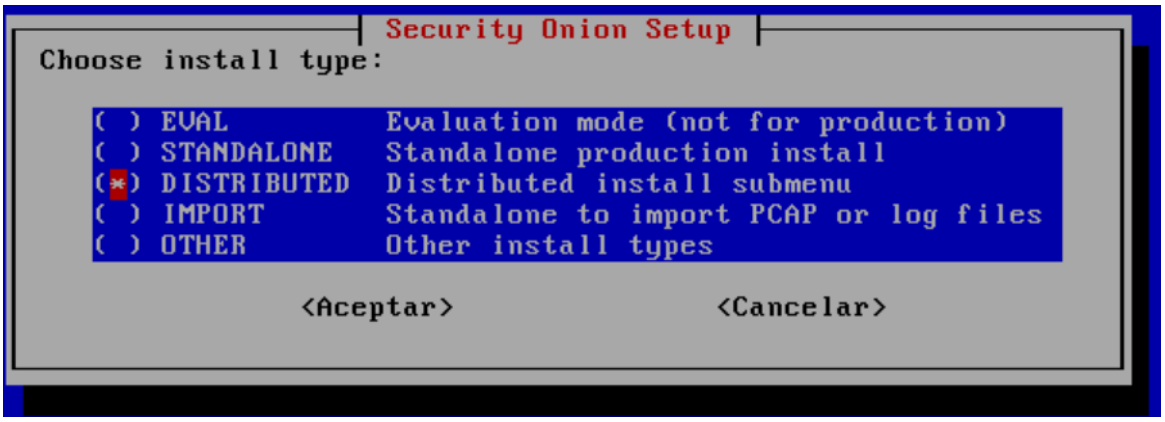


Imagen 5.2: Elección arquitectura nodo

## 5.2 Instalación nodo *manager*

El nodo *manager* debe ser el primer nodo en crearse puesto que funcionará como punto de unión de los demás nodos a la arquitectura distribuida. A lo largo de los siguientes puntos, vamos a argumentar siguiendo el orden de aparición en la instalación, cada una de las configuraciones realizadas.

### Elección tipo de Nodo

Como se nos muestra en la imagen 5.3 seleccionaremos la creación de un nodo *manager*.


Imagen 5.3: Elección nodo *manager*

### Elección nombre del nodo

Los nodos de tipo *manager* seguirán la estructura *managernode* unido al número de nodo *manager* que vayamos a desplegar, en nuestro caso, al disponer de un único nodo, el *hostname* del nodo es el indicado en la imagen 5.4.

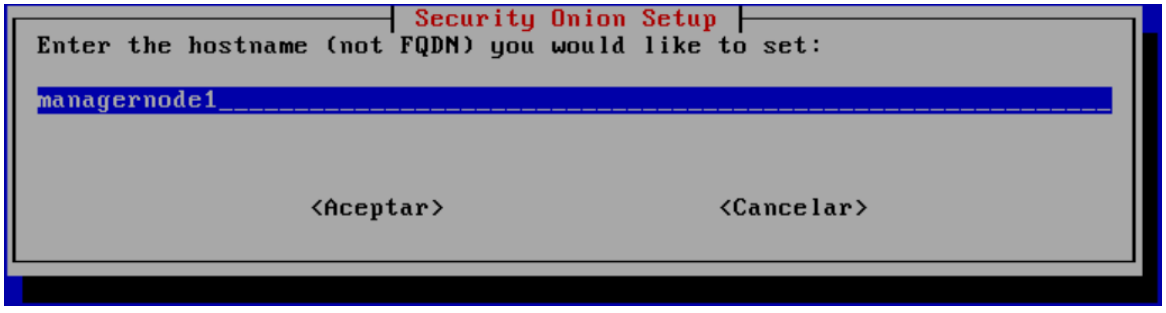


Imagen 5.4: Elección *hostname manager*

### Selección interfaz monitorización

Estableceremos la interfaz ETH0 como interfaz de monitorización como se puede observar en la imagen 5.5. Los nodos *manager* solo necesitan la configuración de una única interfaz para la comunicación con otros nodos.

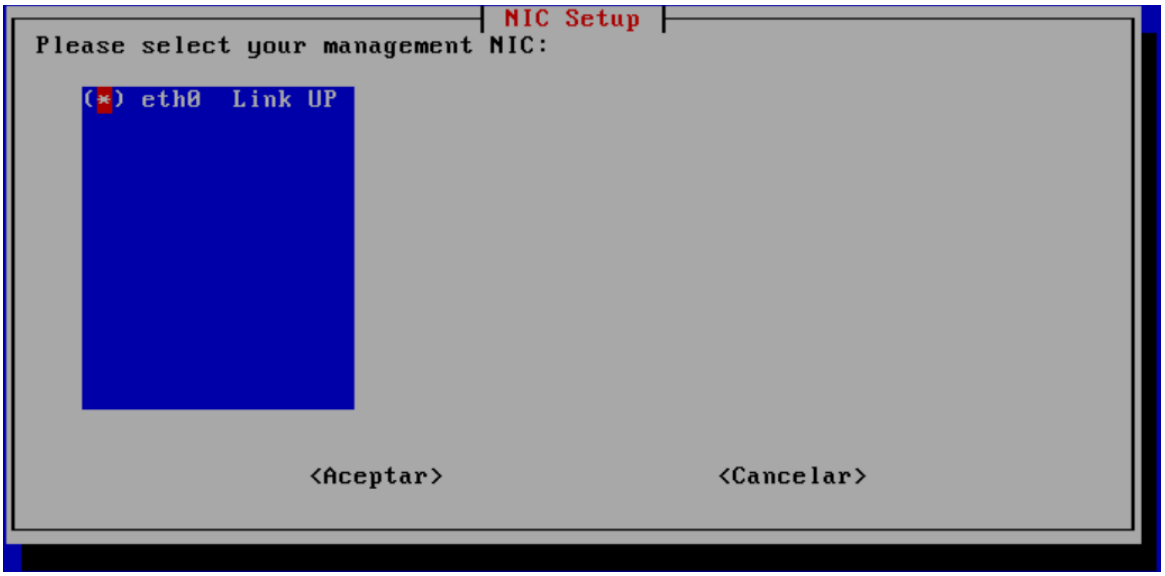


Imagen 5.5: Elección interfaz *Manager*

### Método actualización

Respecto a la forma de actualizarse, observable en la imagen 5.6, indicaremos que las actualizaciones se realicen de forma automática y que todos los nodos se conecten al nodo *manager* para su realización.

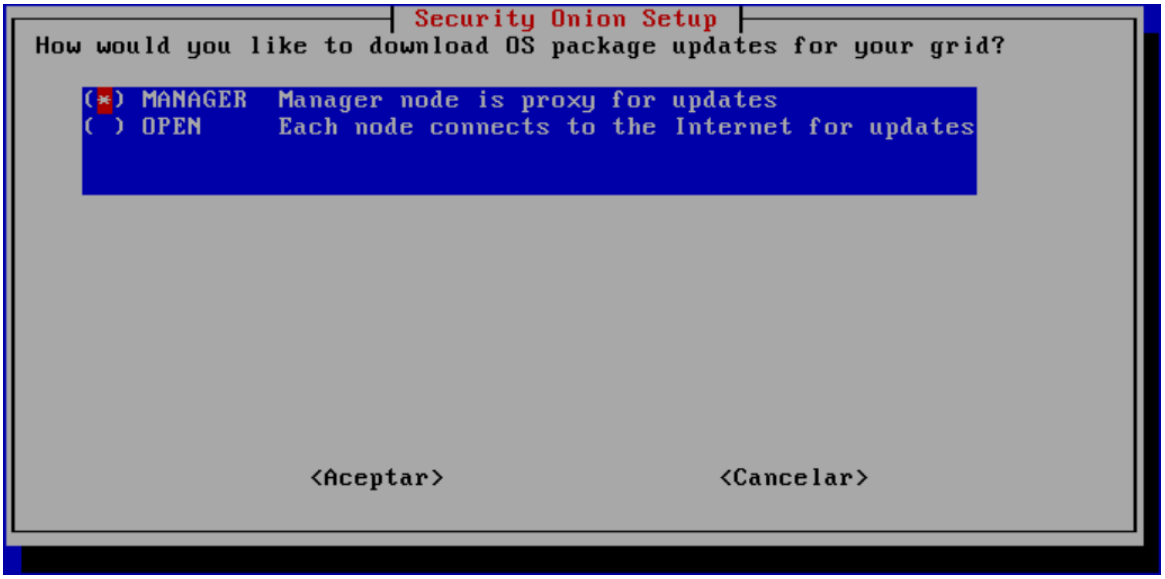


Imagen 5.6: Actualizaciones nodo *Manager*

### Rango de direcciones IP

En este punto, deberemos fijar las subredes que contendrán los nodos que vamos a añadir a la red. En nuestro caso hemos fijado la dirección 192.186.60.0 con mascara de 16 bits que corresponderán con los nodos añadidos en el diseño. A mayores, hemos añadido el rango de direcciones IP privadas 10.0.0.0/8 y 172.16.0.0/12. Esta configuración se puede ver en la imagen 5.7

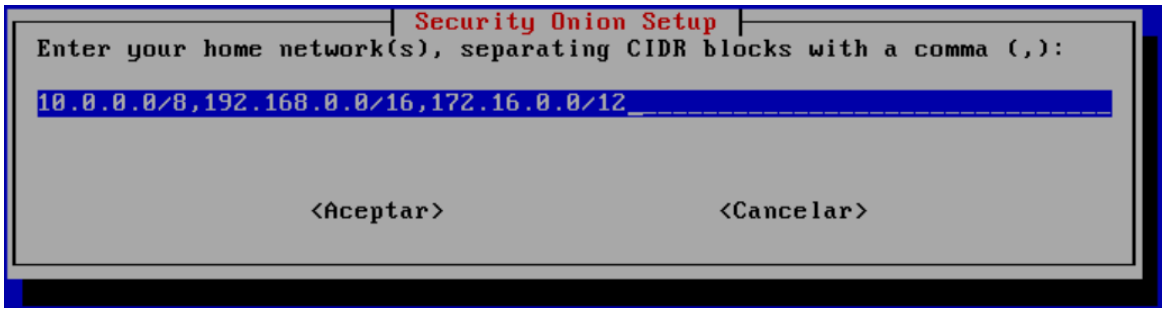


Imagen 5.7: Direcciones IP añadidas *Manager*

### Instalación avanzada

Realizaremos la instalación avanzada, mostrada en la imagen 5.8, para definir las herramientas NIDS que vamos a emplear y las herramientas de Security Onion que incluiremos.

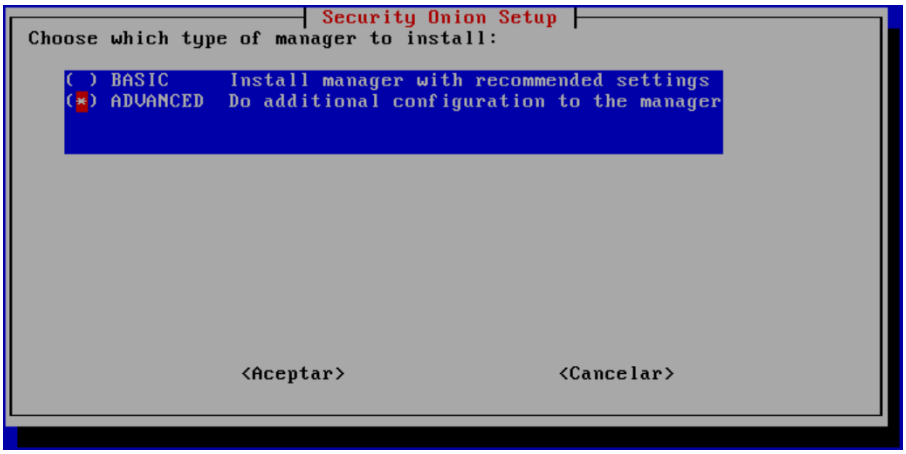


Imagen 5.8: Instalación avanzada *manager*

### Elección herramientas NIDS

Como hemos mencionado en el diseño, seleccionaremos Zeek para la recolección de *logs* de la actividad de red y Suricata para las alertas. Esta elección, junto con una breve descripción, se puede observar en la imagen 5.9.

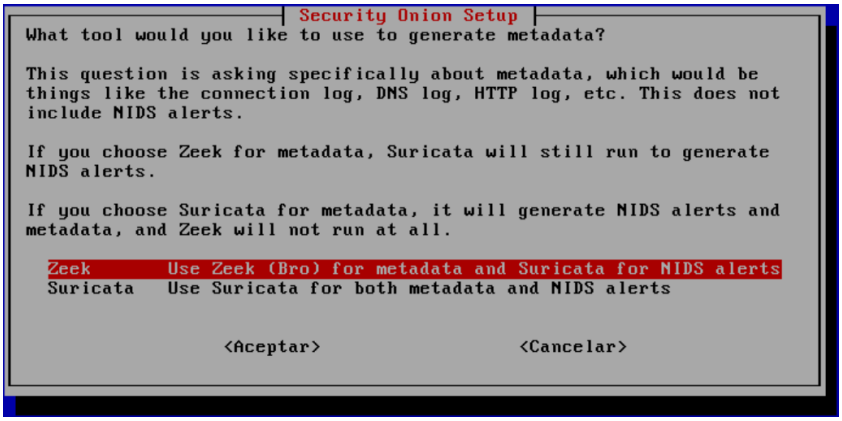


Imagen 5.9: Herramientas NIDS escogidas

En esta configuración avanzada, también se nos permite escoger que *logs* irá almacenando Zeek y el conjunto de reglas que empleará Suricata. En nuestro caso, Zeek va a recoger todos los *logs* posibles como podemos ver en la imagen 5.10 y Suricata empleará las reglas TALOS, las cuales han sido anteriormente mencionadas en la sección 2.3.1. La asignación de las reglas Talos se realizará de la forma que indicamos en la imagen 5.11.

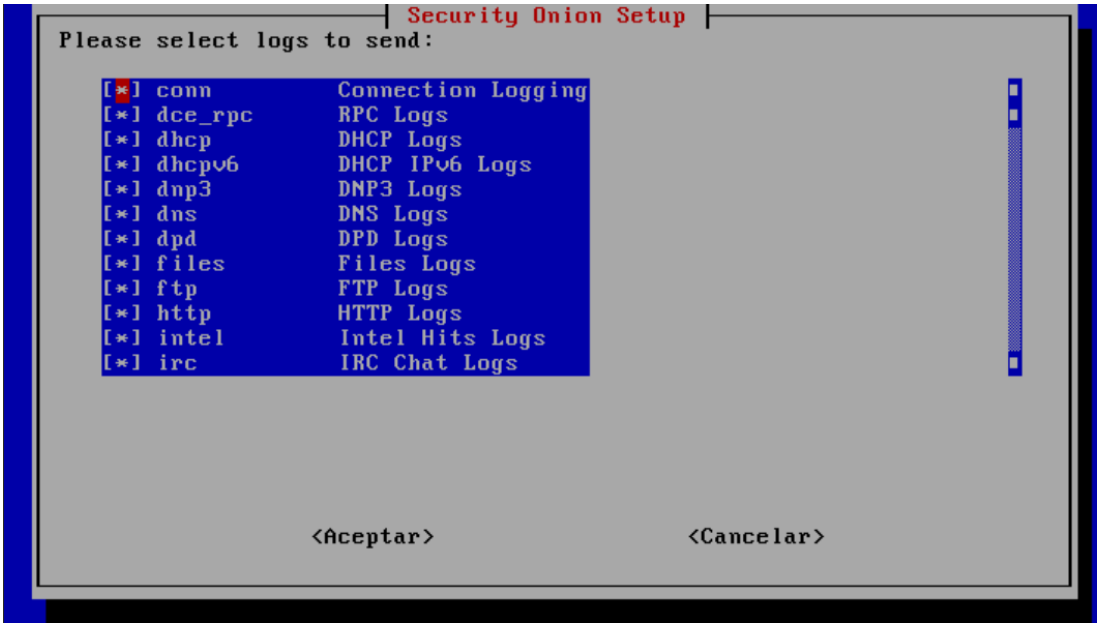


Imagen 5.10: *Logs* que recoge Zeek

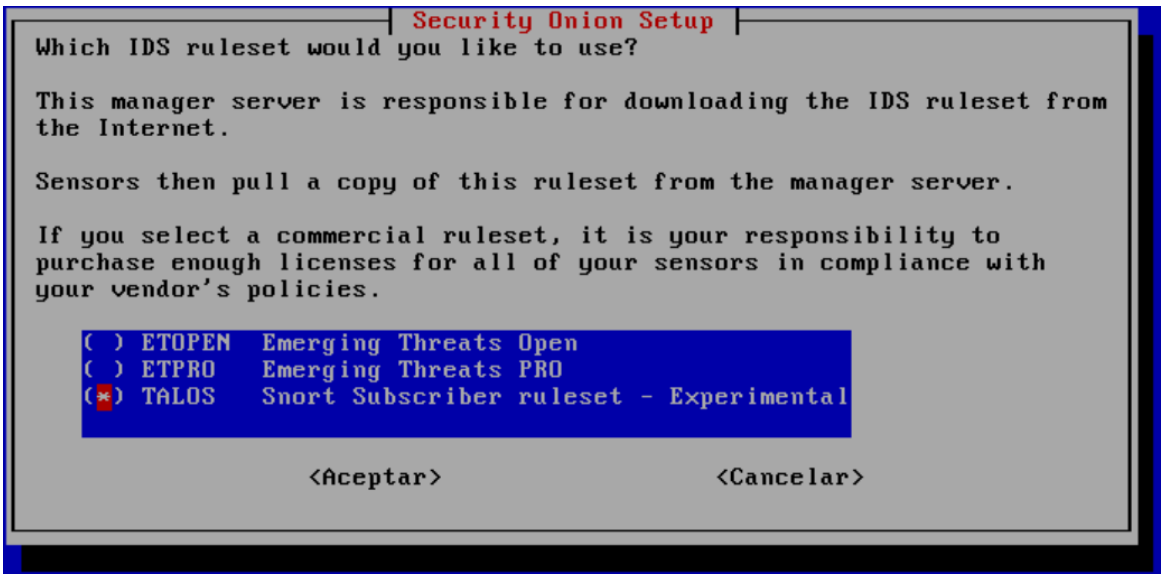


Imagen 5.11: Reglas de Suricata

### Herramientas incluidas

Se han incluido todas las herramientas de las que dispone Security Onion como podemos ver en la imagen 5.12.

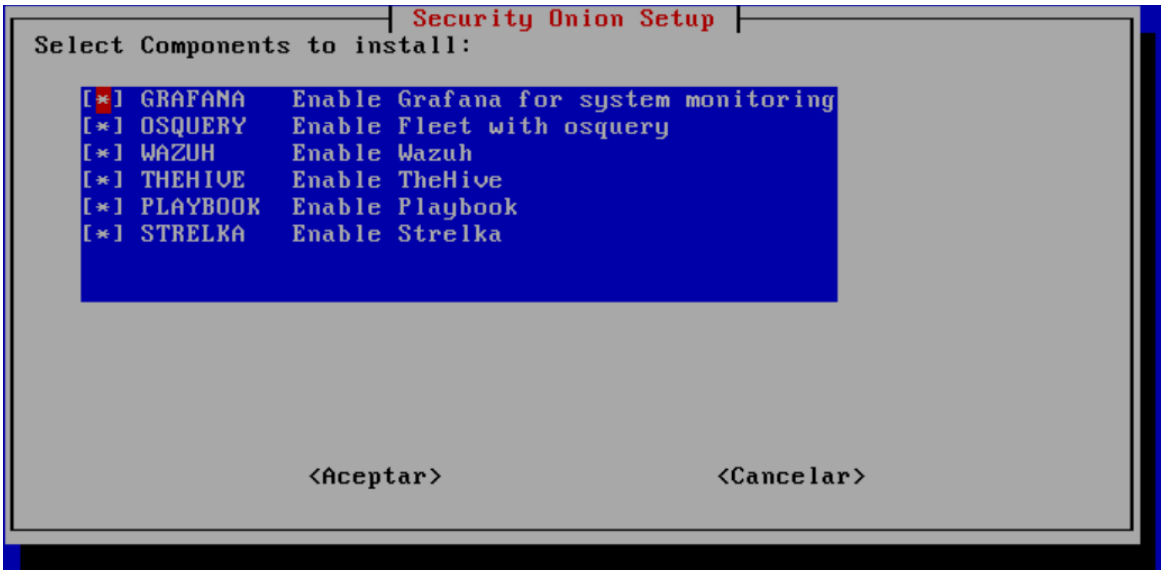


Imagen 5.12: Herramientas incluidas

#### Creación cuenta administrador

Para el acceso a la consola de Security Onion dispondremos de una cuenta de administración que crearemos como vemos en la imagen 5.13.

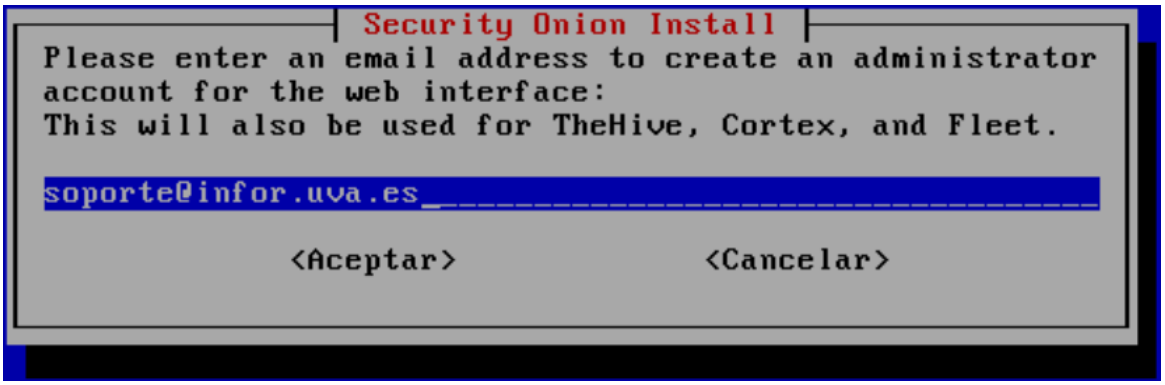


Imagen 5.13: Cuenta de administración

Una vez hemos establecido todas las configuraciones anteriores, podemos realizar la confirmación de instalación del nodo *manager*. Durante esta instalación se desplegarán todos los dockers que ejecutan las herramientas de Security Onion.

## 5.3 Instalación nodo *search*

Con el nodo *manager* ya creado, este tipo de nodos solo necesitaran unirse al *manager* y establecer las mismas configuraciones que hemos especificado anteriormente. Por este motivo, solo vamos a indicar las opciones específicas para los nodos *search*.

#### Incorporación al nodo *manager*

Para añadir los nodos al sistema distribuido, debemos indicar el *hostname* fijado para el nodo *manager*. Una vez disponga del *hostname*, se establecerá una conexión ssh entre el nodo a añadir y el *manager*, este último comprobará las credenciales del equipo a incluir y en caso de ser correctas lo añadirá como nuevo nodo.

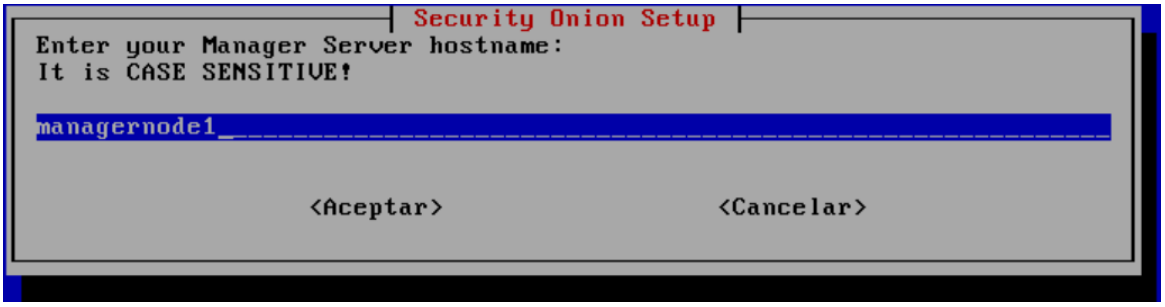


Imagen 5.14: Incorporación al nodo *manager*

#### Instalación avanzada

En este caso, hemos realizado una instalación avanzada pero no hemos realizado ninguna modificación a ninguno de los parámetros. Esto se debe a que los valores indicados son los recomendados en la documentación del paquete ELK Stack.

## 5.4 Instalación nodo sensor

De forma similar a un nodo *search*, los nodos sensores se deben añadir al sistema distribuido mediante la conexión con el nodo *manager*. Las principales diferencias en su instalación respecto a otro tipo de nodo radican en que los sensores necesitan configurar una interfaz de *sniffing* para la obtención de los paquetes de la red y a que son los encargados de ejecutar los IDS, los cuales necesitarán una configuración específica.

#### Interfaz de *sniffing*

Como hemos mencionado en el diseño, la interfaz ETH1 será la encargada de funcionar como interfaz para monitorizar el tráfico de la red. La selección de esta interfaz de *sniffing* se puede observar en la imagen 5.15.

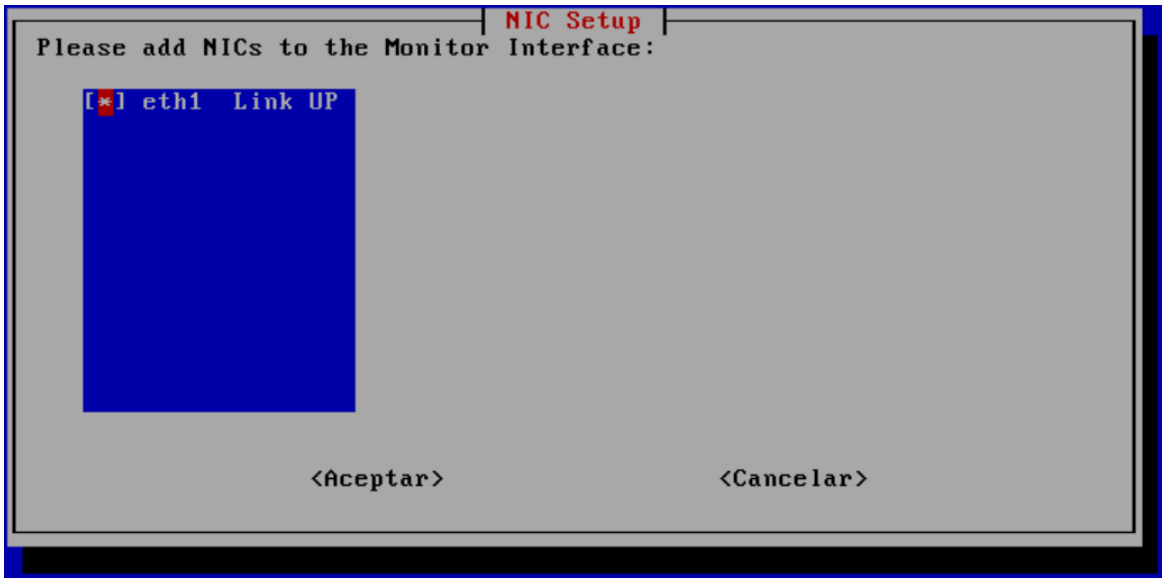


Imagen 5.15: Interfaz de monitorización

### Configuración NIDS

En la documentación de Security Onion, se nos indica que los NIDS funcionan de forma óptima si se encarga su ejecución a núcleos concretos del sistema. Es por este motivo por el cual hemos especificado una instalación avanzada, en la que estableceremos que Zeek se ejecute en los núcleos 0,1 y 2 y que Suricata emplee los núcleos 3,4 y 5. Podemos observar estas configuraciones en las imágenes 5.16 y 5.17.

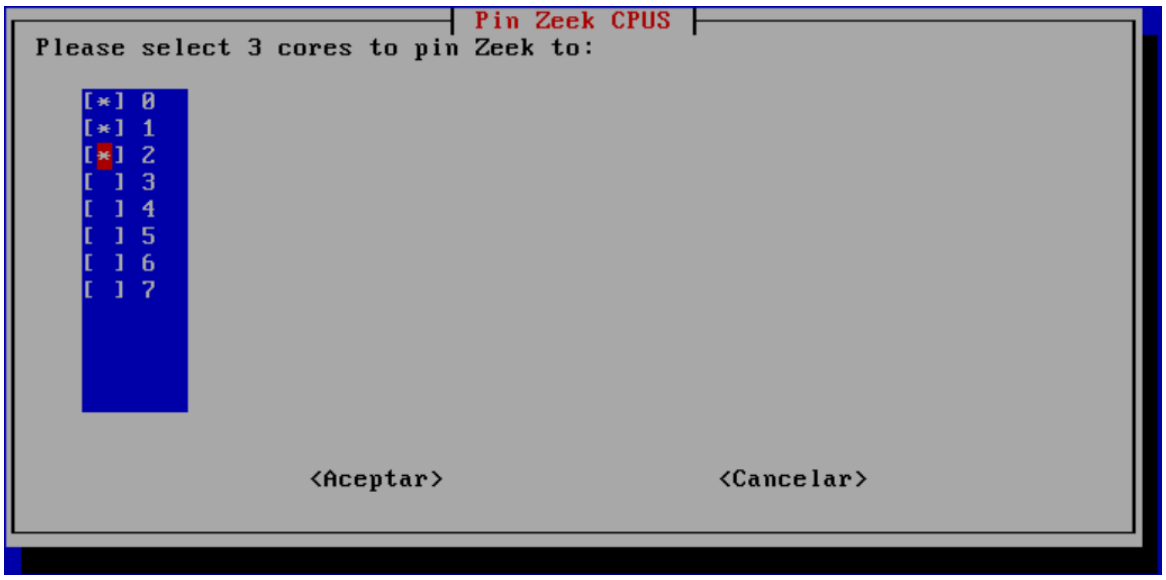


Imagen 5.16: Núcleos asignados a Zeek

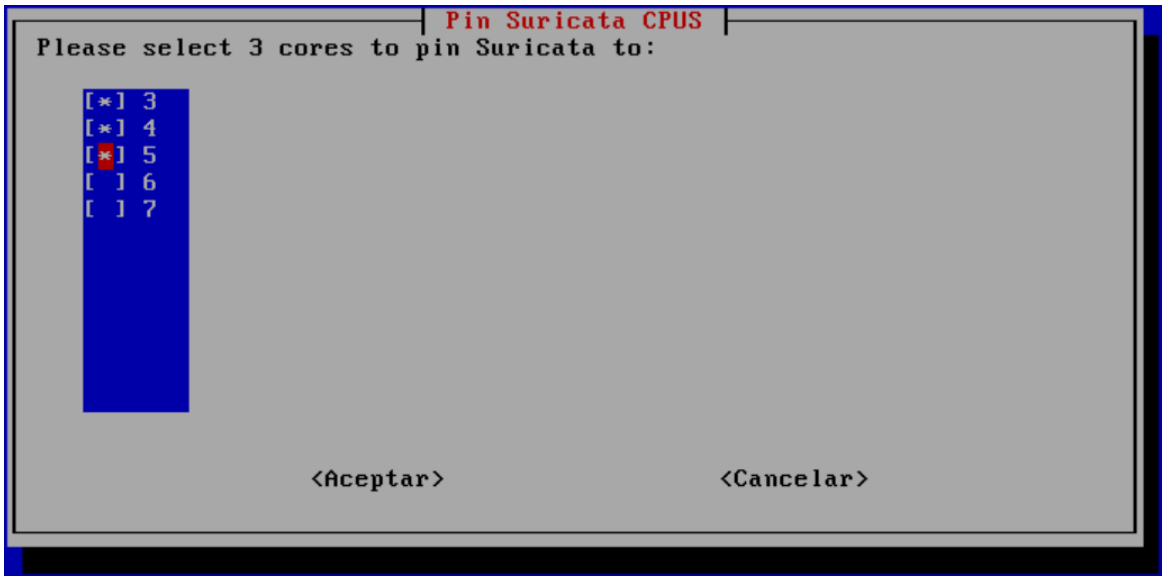


Imagen 5.17: Núcleos asignados a Suricata

## 5.5 Instalación máquina analista

La instalación de una maquina analista es un proceso sencillo, únicamente deberemos utilizar el ejecutable *so-analyst-install*. Una vez este proceso ha realizado la instalación, deberemos indicar al *firewall* que la dirección IP del nuevo nodo se trata de una máquina de analista.

Mediante el uso de *so-allow* en el nodo *manager* se nos muestra la opción de añadir la IP de ese nodo analista, podemos observar esto en la imagen 5.18. En este caso, emplearemos la opción *Analyst* y añadiremos la IP 10.124.60.6 .

```
[root@managernode1 SecurityOnion]# so-allow
This program allows you to add a firewall rule to allow connections from a new I
P address.

Choose the role for the IP or Range you would like to add

[a] - Analyst - ports 80/tcp and 443/tcp
[b] - Logstash Beat - port 5044/tcp
[e] - Elasticsearch REST API - port 9200/tcp
[f] - Strelka frontend - port 57314/tcp
[o] - Osquery endpoint - port 8090/tcp
[s] - Syslog device - 514/tcp/udp
[w] - Wazuh agent - port 1514/tcp/udp
[p] - Wazuh API - port 55000/tcp
[r] - Wazuh registration service - 1515/tcp

Please enter your selection:
```

Imagen 5.18: *Firewall* nodo *manager*

# 6. Analisis de Resultados

En este instante, en el cual ya tenemos funcionando la plataforma de detección mediante Security Onion, vamos a proceder a la descripción y análisis de los datos obtenidos. Junto a este aspecto, mostraremos la herramienta de la que dispone Security Onion para la visualización del funcionamiento del diseño planteado.

Como primer aspecto, definiremos el espacio temporal que vamos a considerar y durante el cual observaremos un comportamiento normal del Departamento de Informática de la Universidad de Valladolid. Una vez delimitado el intervalo de tiempo, realizaremos un estudio general de los datos obtenidos, enfocándonos de forma más detallada en las alertas generadas por cada uno de los IDS Zeek, OSSEC y Suricata.

## 6.1 Intervalo de tiempo escogido

El funcionamiento del departamento de informática de la Universidad de Valladolid está influenciado por determinados factores. El de mayor relevancia, sin lugar a duda, es el calendario académico, puesto que durante determinados días no laborables podríamos obtener datos de un menor uso de nuestra red.

Respecto al número de días que será necesario obtener datos de los nodos sensores, se ha considerado que debido al alto número de *logs* generados por día, disponemos de suficientes datos con realizar una recolección de *logs* durante 7 días. Esta decisión se puede considerar como correcta una vez vemos que el número de *logs* obtenido durante 7 días es cercano a 8 millones, suficientes para realizar un análisis complejo.

En resumen, vamos a fijar una semana lectiva en el calendario académico que represente un

funcionamiento normal. La semana escogida es la semana del 5 al 12 de mayo, la cual dispone de 5 días lectivos y se sitúa en un periodo durante el cual hay bastante uso por parte de los usuarios.

## 6.2 Descripción de los datos

Como ya sabemos, Zeek, Wazuh y Suricata obtienen información de la red y de los *hosts* y la almacenan en archivos de *logs*. Antes de realizar un análisis del contenido que nos ofrecen estos *logs*, vamos a explicar de forma concreta como es un *log* y que datos nos ofrece.

### Descripción de un *log*

En nuestra plataforma, disponemos de *logs* obtenidos por HIDS y NIDS, en ambos casos podemos describir un *log* como un objeto de formato JSON que dispone de parámetros informativos. Independientemente de la herramienta por la que han sido generados, los *logs* disponen de campos comunes para su gestión dentro de Security Onion, entre ellos podemos destacar:

- Índice.
- Versión.
- Marcas temporales.
- Nodo por el cual se ha generado.
- Nodo de almacenamiento.
- Directorio en el que está situado.

Una vez hemos indicado los campos comunes, vamos a explicar aquellos que son característicos en función del tipo de *log*:

- En el caso de los ***logs* generados por herramientas NIDS** o ***logs* de red**, el JSON contendrá información sobre la petición que se envía entre los equipos. Entre sus datos más relevantes, los cuales se pueden observar en el *log* anexo 9.1, podemos destacar:
  - **Datos del servidor.** Se nos indicará la dirección IP y puerto de destino.
  - **Cliente.** Se nos indicará dirección IP y puerto de origen.
  - **Información de los protocolos empleados.** Entre algunos protocolos utilizados podemos mencionar UDP, TCP, DHCP o DNS.
  - **Paquetes enviados.** El JSON contendrá el número de paquetes enviados entre cliente y servidor junto con el tamaño en bytes de los paquetes.
  - **Descripción estado conexión.** Incluye información sobre el establecimiento de la conexión mediante los *flags* (Ej:ACK, SYN).
- Para los ***logs* generados por herramientas HIDS** o ***logs* de *host***, debido a su instalación particular en cada *host*, los parámetros de los que dispone están enfocados en datos pertenecientes

al sistema que monitoriza. Entre sus datos más relevantes, que se pueden observar en el *log* anexo 9.2, podemos destacar:

  - **Mensaje explicativo alerta.**
  - **Información del proceso por el cual es provocado.**

- Como caso específico de *logs*, podemos mencionar las **alertas** que variaran su información dependiendo de si se han generado por NIDS o por HIDS. En el caso de las **alertas de red**, dispondrán de las direcciones IP contenidas en el *log* de red que ha generado esa alerta, mientras que para las **alertas de *Host*** dispondremos de información concreta del sistema obtenida del *log* de *host*. Aunque ambos tipos de *logs* varían en función del *log* que han empleado para su detección, en los dos casos contienen información de la regla que genera esa alerta. Entre estos campos, que se pueden observar en el anexo 9.3, podemos destacar:
  - **Nombre de la regla.**
  - **Categoría de la regla.**
  - **Severidad de la regla.** La severidad nos indica la medida en la que una alerta representa un problema de mayor o menor gravedad. En Security Onion disponemos de 3 valores para la severidad, los cuales son severidad baja, media y alta.

Para la visualización de todo este tipo de *logs* vamos a emplear Kibana, que nos permitirá mediante su interfaz gráfica filtrar por el intervalo de tiempo deseado.

A mayores de un filtrado temporal, se nos ofrece la posibilidad de realizar filtrados más específicos dependiendo de distintos factores. Entre los más empleados podemos destacar:

- Filtrado de los *logs* generados por nodos.
- Filtrado según sean *logs* de red o de *host*.
- Filtrado según la herramienta que ha generado el *log*.
- Filtrado por protocolos de red.
- Filtrado de alertas.

## 6.3 Datos obtenidos

Una vez conocemos cómo y de qué tipo son los datos que obtenemos, vamos a proceder a describir los que se han obtenido durante el funcionamiento de nuestra plataforma.

Durante el periodo establecido, la plataforma ha obtenido 8,523,612 *logs* como se puede observar en la imagen 6.1.

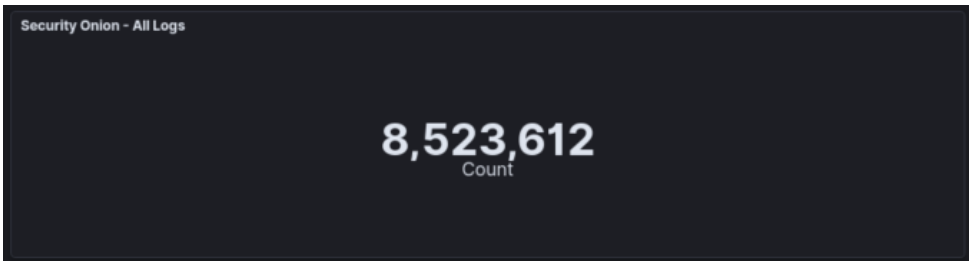


Imagen 6.1: *logs* totales obtenidos

El número de *logs* que se generan por día varía en función del uso que realizan los usuarios, pudiendo observar en la imagen 6.2 como existen 5 picos diferenciados que se corresponden con los días lectivos.

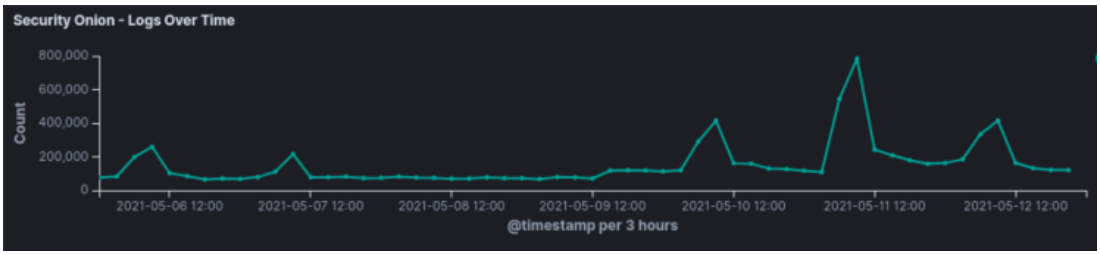


Imagen 6.2: *logs* obtenidos por día

También disponemos de información respecto a la cantidad de *logs* que ha generado cada nodo como se puede observar en la imagen 6.3. Este dato nos permitirá conocer los puntos de nuestra red por los que más tráfico fluye. Respecto al número de *logs* que genera cada herramienta podemos mostrar la imagen 6.4. En esta última imagen destacamos la gran cantidad de *logs* que obtiene Zeek ya que se encarga de monitorizar todo el tráfico que fluye por la red y Ossec que también tiene bastante impacto puesto que obtiene una gran cantidad de *logs* para cada sistema. En el caso de Suricata, debido a que se encuentra centrado en la obtención de alertas, el número de *logs* que obtiene es mucho menor.


Imagen 6.3: *logs* por nodo

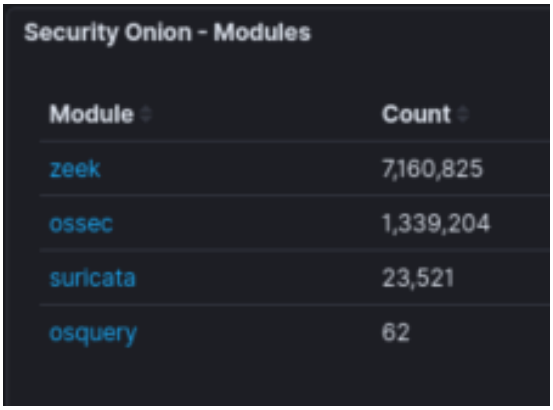


Imagen 6.4: *logs* por herramienta

En este momento en el que ya conocemos todos los *logs* que se han generado en la plataforma vamos a realizar un análisis independiente centrándonos en cada uno de los *logs* explicados. El reparto de los 8,523,612 *logs* según sean de red o *host* se puede observar en la imagen 6.5.

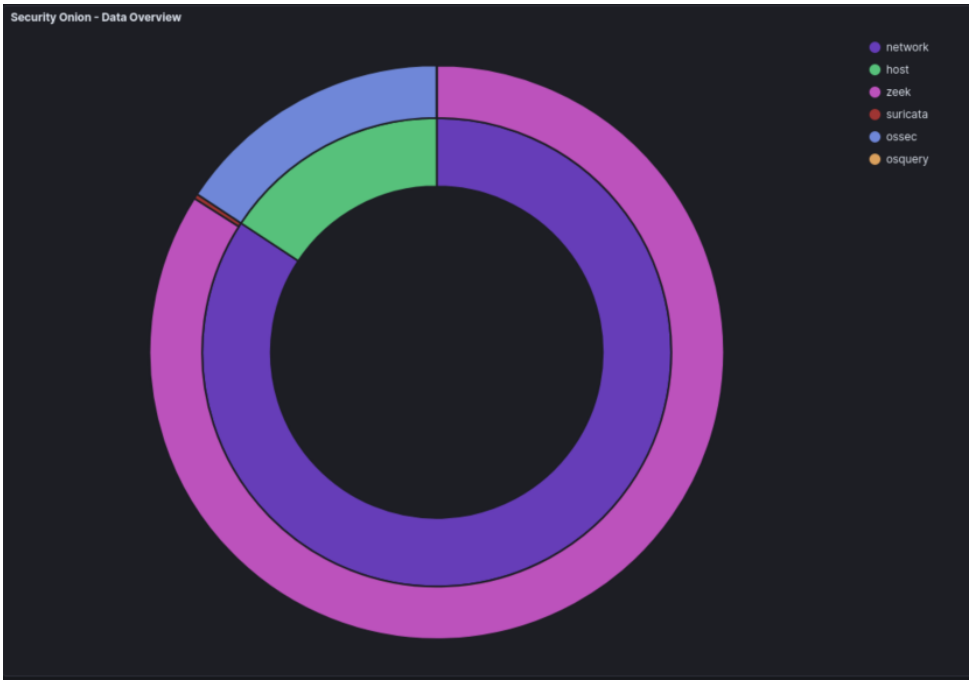


Imagen 6.5: Distribución de los *logs*

A mayores de la división basada en *host* y network, la imagen anterior clasifica los *logs* en función de la herramienta por la cual ha sido generado. Basándonos en este criterio, podemos destacar los siguientes aspectos:

- ***logs* de Zeek.** Generados para la monitorización del tráfico de la red, suponen una gran parte de la totalidad de los *logs* obtenidos. Siendo más concretos, disponemos de 7,160,825 *logs* de Zeek, que representa el 84 % del número total obtenido. Este número tan elevado, se debe a que la plataforma ha analizado durante la semana alrededor de 7 TB de paquetes de red.
- ***logs* de Suricata.** El número de alertas obtenidas por Suricata en nuestra plataforma es de 23,521.
- ***logs* de Ossec.** Se dispone de 1,339,204 *logs* de *host* obtenidos por Ossec.
- ***logs* de Osquery.** Este tipo de *logs* suponen un caso especial en nuestro despliegue, ya que que no hemos implementado el uso de osquery. Los 62 *logs* generados no representan información del departamento, sino que consisten en usuarios de ejemplo realizados por el nodo *manager* para comprobar el funcionamiento de osquery.

Debido a que nuestra plataforma obtiene los *logs* de *hosts* de los propios nodos que forman el diseño de Security Onion, se ha considerado más relevante el análisis de la red. Esto se debe a que nos permite analizar el tráfico real del Departamento de Informática de la Universidad de Valladolid.

## 6.4 Análisis de la red

La plataforma de detección se ha implementado mediante el sistema de virtualización que se encuentra dentro de la Universidad de Valladolid. Este aspecto supone que los datos que obtenemos se vean influenciados debido a las medidas de seguridad desplegadas en la Universidad. Actualmente, como medida de seguridad que nos influye destacan dos *firewalls*, el primero de ellos situado tras el sistema de virtualización que se aplica sobre las máquinas virtualizadas y el segundo situado en la entrada de la red de la UVA. Este último *firewall* se encarga de controlar todo el tráfico entrante a la universidad.

Debido al funcionamiento de los *firewalls*, podemos indicar que el tráfico que obtenemos se encuentra limitado a aquel que se considera aceptable según las comprobaciones que realizan los técnicos de seguridad. Este aspecto será tenido en cuenta en posteriores apartados.

Para la obtención de los *logs* que nos interesan en este apartado, vamos a realizar un filtrado para quedarnos únicamente con los *logs* de red. El número de *logs* resultante es de 7,184,346 como se puede observar la imagen 6.6.

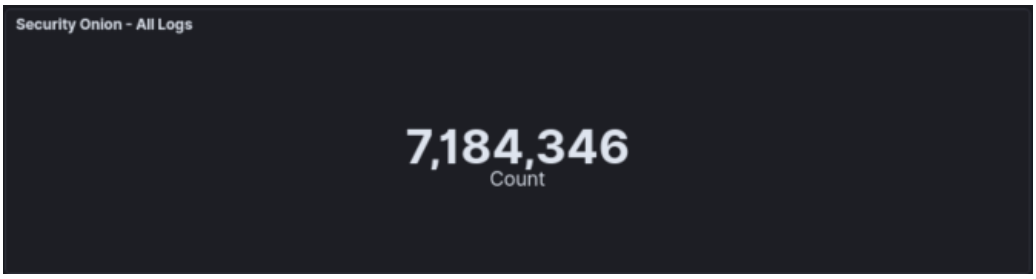


Imagen 6.6: *logs* de red obtenidos

Vamos a realizar un análisis general del contenido de todos estos *logs* de red con el fin de conocer el tráfico que fluye por el Departamento de Informática de la Universidad de Valladolid.

El primer paso que se ha realizado para observar de forma sencilla el tráfico que más influencia tiene en el departamento, es realizar una clasificación de todos los *logs* de red basándonos el conjunto de datos que representa en Kibana. Este conjunto de datos nos separará el tráfico en función de aspectos como el protocolo que emplea. El gráfico se puede observar en la imagen 6.7.

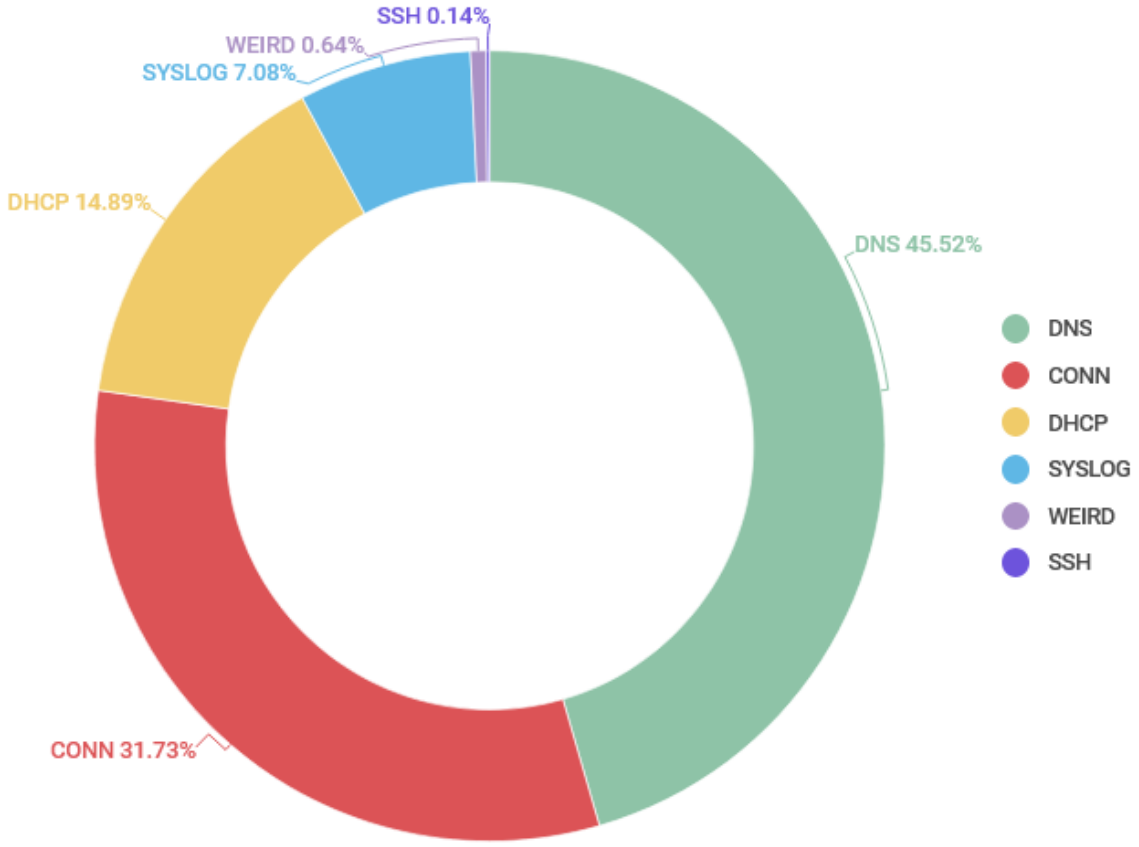


Imagen 6.7: Tráfico de la red

Con el fin de comprender de forma más clara este tráfico,vamos a detallar cada una de las opciones mostradas en la leyenda:

- **DNS.** Las peticiones DNS (*Domain Name System*) nos permiten emplear nombres para obtener las direcciones IP del servidor al que vamos a acceder. Un ejemplo concreto del uso de DNS sería el acceso a *www.google.com*, puesto que mediante una petición DNS obtendríamos la IP del servidor que aloja esa dirección web.
- **CONN.** Las peticiones CONN se centran en el establecimiento de la conexión entre ambas partes de la comunicación. Estas peticiones determinaran entre otros aspectos si el *host* se ha conectado al servidor, si el servidor ha respondido y se encuentra activo y todos aquellos aspectos relacionados con el uso de *flags*, entre los que podemos destacar el *flag* ACK o SYN.
- **DHCP.** El protocolo DHCP (*Dynamic Host Configuration Protocol*) permitirá la asignación de direcciones IP y otros parámetros a todos los dispositivos de nuestra red.
- **SYSLOG.** Mediante el protocolo SYSLOG se permite realizar el envío de mensajes de registro o de eventos a un servidor específico.
- **WEIRD.** Este tipo de tráfico es clasificado por Zeek como extraño puesto que no concuerda con ningún tipo de tráfico conocido. La clasificación como *WEIRD* de un *log*, se puede deber

a errores o modificaciones de los paquetes de red. Debido a este último aspecto, este tráfico se obtiene debido a intentos de ataque que se basen en la modificación de los campos de los paquetes de red.

- **SSH.** El protocolo SSH nos permite el acceso remoto entre dos sistemas mediante el uso de un canal seguro y cifrado.

Aunque este es el tipo de tráfico más frecuente en nuestra red, podemos destacar otros tipos de tráfico que tienen un número mucho menor de *logs*. Entre los más relevantes podemos destacar:

- Tráfico del protocolo SNMP
- Tráfico del protocolo SSL
- Tráfico HTTP
- Descargas de archivos
- Tráfico generado por Kerberos o x509
- Tráfico del protocolo SMTP

Una vez hemos indicado el tráfico existente en nuestra red, es necesario conocer si los resultados obtenidos son aquellos deseados y esperables en una red que funcione correctamente.

Para realizar una comparativa de nuestros datos con el tráfico esperado en una red vamos a emplear un estudio [55] realizado en la Universidad de República Checa. Este estudio, se centra en cuantificar mediante estadísticas los datos del tráfico de determinadas redes de la Universidad de Republica Checa. Se ha considerado como ejemplo representativo debido a que dispone de Eduroam, igual que la Universidad de Valladolid, unido a que en la investigación se monitorizan una serie de entornos similares al departamento escogido.

Como primer aspecto a destacar de nuestro tráfico de red, podemos resaltar el gran uso que se realiza del protocolo UDP que supone el 77% del tráfico total. El mayor uso del protocolo UDP garantiza una mayor velocidad en nuestra red, aunque puede significar mayor pérdida de paquetes de red y, en principio, conexiones menos seguras.

Este porcentaje de uso de UDP es elevado en comparación con los obtenidos en la Universidad de República Checa, el cual se sitúa alrededor del 65%. De todas formas, se sitúa dentro de valores aceptables dentro de una red.

Respecto al uso de los distintos tipos de protocolos como DNS, DHCP, SSH o HTTP, el tráfico que generen variará en función del comportamiento de los usuarios que utilicen la red. Como aspecto común a todas las redes podemos concluir que el tráfico de DNS y de CONN supone un alto porcentaje del tráfico de red. Esta afirmación concuerda con nuestros datos obtenidos y los mostrados en el estudio analizado. Si ponemos un ejemplo concreto podemos detallar el valor del uso de DNS, ya que

para una situación similar en la universidad de República Checa el valor se encuentra sobre el 40%, un valor similar al obtenido en nuestro despliegue. Teniendo en cuenta los demás protocolos, se observa un número de peticiones HTTP y SNMP considerablemente más bajo que en cualquiera de las redes estudiadas.

Una vez observados y explicados los datos, podemos detallar que la mayoría del tráfico de nuestra red se debe al uso de DNS, DHCP y CONN. Este tipo de tráfico es una fuente de un posible ataque a nuestra red. Mediante la generación de alertas vamos a ser capaces de controlarlo y determinar si existe algún tipo de peligro en nuestra red.

En este momento, una vez ya conocemos que existe tráfico en nuestra red que puede suponer una fuente de ataques, vamos a centrarnos en cómo se detectan los posibles ataques asociados a ese tráfico. Para responder a esta necesidad de detección, surgen las alertas de red generadas por Suricata y Zeek. Mediante su estudio se nos proporcionará información relevante para la obtención de posibles vulnerabilidades de la red.

## 6.5 Alertas de red generadas

En este apartado, vamos a mostrar un estudio de las alertas de red obtenidas para cada una de las herramientas NIDS en nuestro despliegue. En primer momento, realizaremos un análisis general que nos ofrecerá una vista de los potenciales problemas de seguridad en nuestro departamento. Una vez conozcamos la situación general, detallaremos algún ejemplo de alerta concreta que nos permita observar de forma clara los beneficios de los NIDS en nuestra seguridad de red.

### Alertas de Suricata

Con el fin de obtener las alertas que nos interesan en este apartado, vamos a realizar un filtrado para quedarnos únicamente con las alertas de red de Suricata.

**Nota 6.5.1 Filtrado de alertas de Suricata**

*event.dataset:alert AND event.module:suricata*

Durante la semana analizada, Suricata ha conseguido obtener 23,521 alertas en nuestra red como podemos observar en la imagen 6.8.

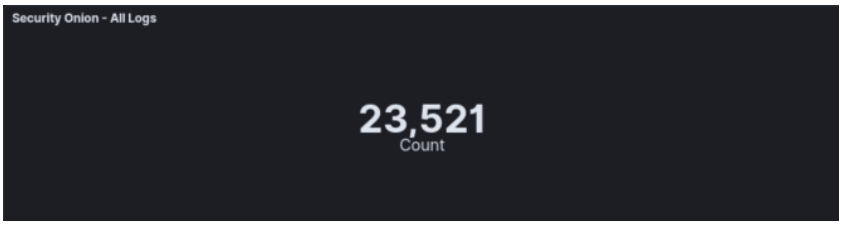


Imagen 6.8: Alertas generadas Suricata

En cuestión del número de alertas generadas diariamente, se puede observar en la imagen 6.9 que existen picos diferenciados que nos permiten detectar intervalos en los que nuestro sistema ha detectado una gran cantidad de alertas. Siendo más concretos, se obtienen pequeños incrementos correspondientes a los días laborables de la semana y un gran pico presente el día 11 de mayo, el cual será analizado posteriormente como un posible ataque grupal a nuestra red.

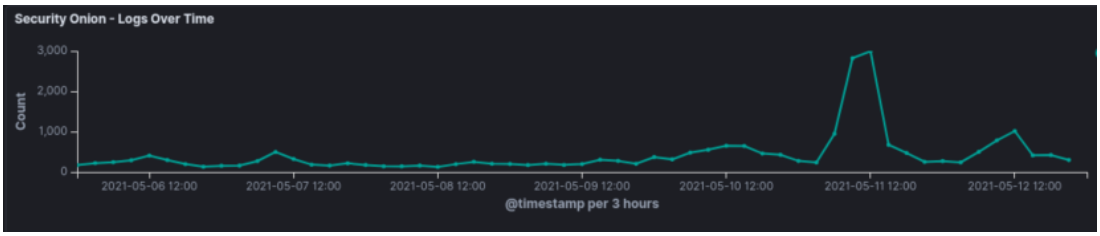


Imagen 6.9: Alertas Suricata obtenidas por día

Cada una de las alertas generadas, se corresponde con la ejecución de una regla en la herramienta Suricata. El estudio de las reglas que más se han ejecutado en nuestra red nos permitirá conocer los posibles ataques sufridos y centrarnos en los casos que nos van a resultar más relevantes. Como primer aspecto, es necesario realizar una clasificación como la mostrada en la tabla 6.1, donde podemos observar las 10 reglas más obtenidas en nuestra plataforma.

| **Regla** | **N.º de alertas** |
|---|---|
| *ET POLICY Spotify P2P Client* | 7108 |
| *ET SCAN Potential SSH Scan* | 6396 |
| *ET DROP Dshield Block Listed Source group 1* | 712 |
| *ET 3CORESec Poor Reputation IP group 13* | 521 |
| *ET COMPROMISED Hostile Host Traffic group 49* | 503 |
| *ET COMPROMISED Hostile Host Traffic group 61* | 452 |
| *ET COMPROMISED Hostile Host Traffic group 84* | 393 |
| *ET COMPROMISED Hostile Host Traffic group 80* | 326 |
| *ET COMPROMISED Hostile Host Traffic group 65* | 297 |
| *GPL SNMP public access udp* | 265 |

Tabla 6.1: Alertas por regla

Como se puede observar un gran porcentaje de las 23521 alertas se han generado debido a estas 10

reglas. Debido a este hecho, vamos a explicar más detalladamente en que consiste cada una de las 10 reglas mostradas:

- ***ET POLICY Spotify P2P Client.*** Esta regla se lanza debido a la política de seguridad establecida en las reglas TALOS, la cual intenta avisar del tráfico P2P que genera Spotify. La severidad de esta alerta es baja y simplemente es informativa y nos permite conocer esta situación en caso de querer evitar este tipo de tráfico.
- ***ET SCAN Potential SSH Scan.*** Como hemos mencionado en la sección 2.1.2, uno de los ataques más realizados consiste en intentar escanear nuestra red en busca de información para posibles ataques futuros. La regla *ET SCAN Potential SSH Scan*. se enfoca en detectar los ataques de escaneo mediante SSH sobre nuestra red.
- ***ET DROP Dshield Block Listed Source group.*** Esta regla se ejecuta para peticiones de red con una dirección IP de origen catalogada como maliciosa. La lista de direcciones IP no confiables está realizada por grupos internacionales de seguridad como el *Internet Storm Center*.
- ***ET 3CORESec Poor Reputation IP Group.*** Suricata dispone de un proceso para el cálculo de la reputación de una dirección IP. La regla se obtendrá en aquellos casos en la que la dirección IP no cumpla determinados criterios y se califique como de baja reputación.
- ***ET COMPROMISED Known Compromised or Hostile Host Traffic Group.*** Esta regla indica que nuestro IDS ha detectado tráfico proveniente de una dirección IP que se encuentra dentro de la lista de amenazas emergentes. Esta lista engloba todos los orígenes o grupos, como los denomina Suricata, para los cuales se conoce que han realizado ataques. En nuestro despliegue nos hemos encontrado gran cantidad de apariciones de este tipo de regla, aunque han sido generadas por grupos distintos.
- ***GPL SNMP public access udp.*** Esta regla se obtiene en los casos en los que Suricata ha detectado tráfico que podría estar utilizando una vulnerabilidad del protocolo SNMP para la realización de un ataque.

Otro de los aspectos más relevantes de las alertas se basa en la severidad que se nos indica para cada una de ellas. Para las alertas Suricata generadas en nuestro despliegue, hemos obtenido la severidad indicada en la imagen 6.10. Respecto a la severidad obtenida, podemos destacar que no disponemos de ninguna alerta que este marcada como severidad alta, lo que nos indica que no será necesario realizar medidas inmediatas.

Como único aspecto a tener en cuenta podemos mencionar el número de escaneos SSH que se realizan sobre nuestra red, llegando a obtener 6396 alertas de este tipo durante la semana estudiada.

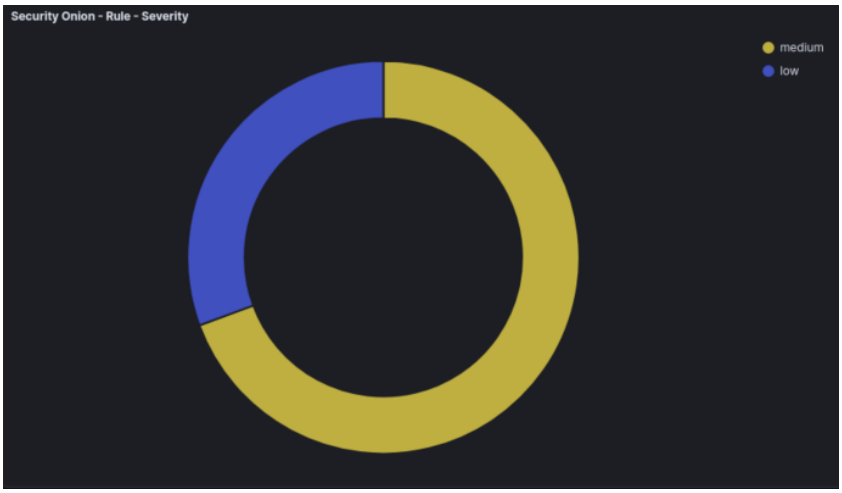


Imagen 6.10: Severidad alertas Suricata

Mediante esta plataforma, también somos capaces de determinar cuáles son los puertos más empleados para la realización de ataques a nuestra red. Estos datos, se pueden entender de forma sencilla mediante la imagen 6.11.

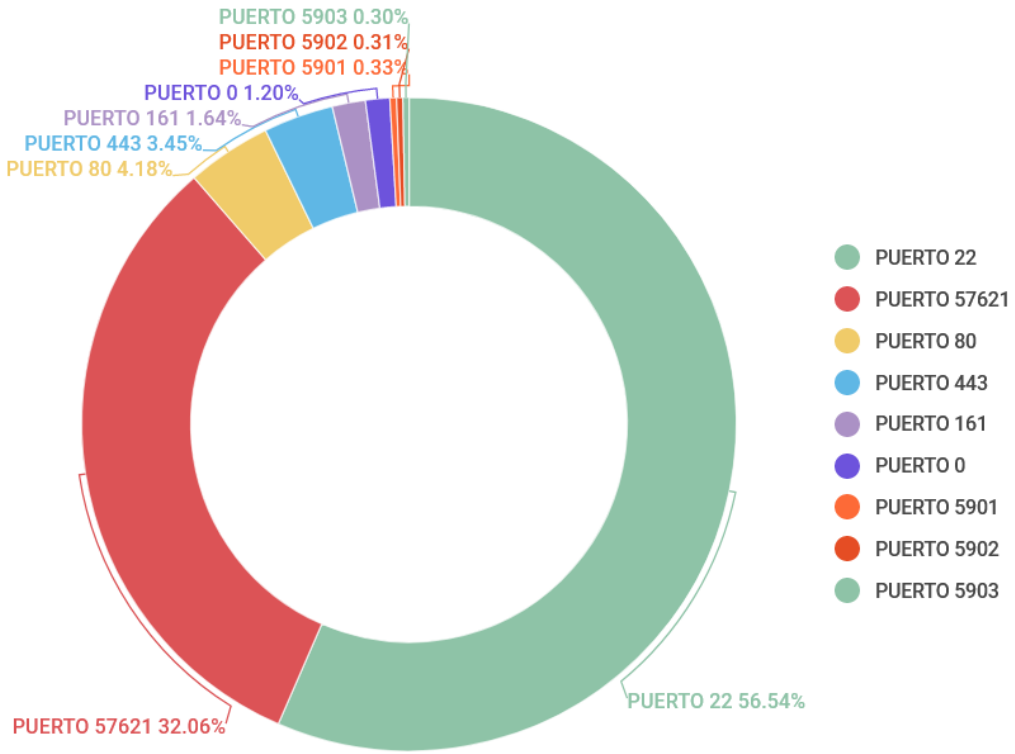


Imagen 6.11: Puertos más atacados

Una vez conocemos los puertos más atacados, vamos a detallar brevemente la función que realizan cada uno de ellos:

- **Puerto 22.** Empleado para las conexiones SSH y SFTP.
- **Puerto 57621.** Empleado por Spotify para la comunicación P2P.

- **Puerto 80.** Empleado para la navegación web mediante http.
- **Puerto 443.** Empleado para la navegación web mediante https.
- **Puerto 8080.** Se trata del puerto alternativo al 80, debido a este aspecto también es empleado para el protocolo http.
- **Puerto 161.** Empleado por el protocolo SNMP.
- **Puerto 0.** Este puerto es un caso especial puesto que no dispone de una funcionalidad fija, sino que funciona como puerto auxiliar para redirigir al puerto correcto. El tráfico que se observa de este puerto suele ser considerado una fuente de ataques y debido a ello se encuentra bloqueado en muchas redes.
- **Puerto 5901 / 5902 / 5903.** Estos puertos se emplean para ejecutar el servicio VNC (*Virtual Network Computing*). El software VNC nos permite compartir sobre varios ordenadores las acciones que se realizan en un ordenador concreto. En el caso de la Universidad de Valladolid el principal uso de VNC es la compartición de pantalla entre profesor y estudiantes, de forma que los estudiantes observen en sus pantallas las acciones del profesor.

Como último aspecto en cuestión de las alertas de Suricata, cabe reflejar que Kibana nos permite obtener las direcciones IP para las cuales se han generado las alertas. Siendo más concretos, nos permitirá conocer la IP de origen del tráfico que ha generado la alerta y la IP a la que se dirigía ese tráfico. Mediante las direcciones IPS de destino, vamos a ser capaces de determinar que sistemas de nuestra red están siendo más atacados y tomar las medidas necesarias para su protección. Los datos de las direcciones IP mas atacadas de nuestro entorno se pueden observar en la tabla 6.2.

| Dirección IP destino | N.º de alertas |
|---|---|
| 157.88.124.253 | 7113 |
| 157.88.124.16 | 3723 |
| 157.88.123.10 | 3342 |
| 157.88.123.125 | 1263 |
| 157.88.124.104 | 1057 |
| 157.88.123.138 | 903 |
| 157.88.124.224 | 617 |
| 157.88.123.227 | 338 |
| 157.88.124.212 | 293 |
| 157.88.124.76 | 257 |

Tabla 6.2: Direcciones IP de destino de las alertas

Mientras que en la imagen 6.3, vamos a disponer de aquellas direcciones IP que constituyen una fuente de ataques a nuestro sistema.

| Dirección IP origen | N.º de alertas |
|---|---|
| 157.88.124.231 | 5168 |
| 157.88.124.234 | 516 |
| 157.88.124.225 | 505 |
| 157.88.124.222 | 401 |
| 62.122.156.74 | 391 |
| 157.88.124.223 | 335 |
| 49.233.204.192 | 324 |
| 202.61.241.17 | 290 |
| 111.67.204.137 | 238 |
| 185.36.81.52 | 232 |

Tabla 6.3: Direcciones IP de origen de las alertas

Llegados a este punto ya conocemos las alertas que más se han generado en nuestra plataforma y las que se consideran de mayor severidad. Debido al alto número obtenido es muy costoso realizar un análisis individual de cada una de ellas. Este hecho, unido a que gran cantidad de alertas tienen una severidad baja, ha supuesto que solo analicemos una alerta suficientemente representativa que se tratará de la alerta por escaneo SSH.

### Alerta de escaneo

La alerta, la cual denominaremos alerta de escaneo, debido a que es causada por la regla *ET SCAN Potential SSH Scan*, puede considerarse relevante para su análisis puesto que es aquella con la severidad más elevada que más veces se ha detectado en nuestra red. Además, si observamos la imagen 6.12, la fecha en las que más alertas de este tipo se han generado se trata del 11 de mayo, suponiendo el tráfico de esta regla la causa del posible ataque grupal indicado anteriormente.

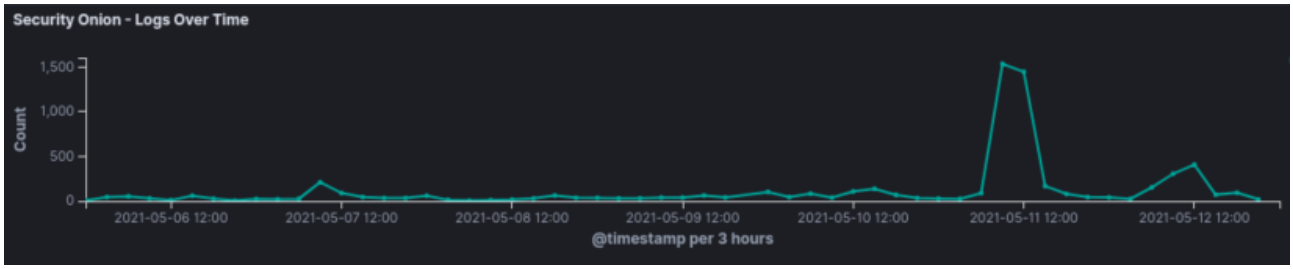


Imagen 6.12: Alertas de escaneo por día

En este momento, vamos a detallar la información de un *log* de alerta concreto que se haya generado por la regla *ET SCAN Potential SSH Scan*. De esta forma vamos a ser capaces de explicar todos los datos de los que disponemos para solucionar el potencial problema de los escaneos SSH sobre nuestra red. Como primer paso, vamos a filtrar en Kibana para quedarnos únicamente con los *logs* de alerta que se han generado esta regla.

**Nota 6.5.2** Filtrado por regla SSH SCAN

*rule.name.keyword: ET SCAN Potential SSH Scan*

Una vez nos situamos sobre un *log* de alerta concreto, vamos a disponer de una gran cantidad de información. En primer lugar, conoceremos el origen y destino del ataque de escaneo. Esta información se puede observar en las imágenes 6.13 y 6.14.


```
"destination": {
  "geo": {
    "continent_name": "Europe",
    "region_iso_code": "ES-VA",
    "city_name": "Valladolid",
    "country_iso_code": "ES",
    "timezone": "Europe/Madrid",
    "ip": "157.88.123.10",
    "country_name": "Spain",
    "region_name": "Valladolid",
    "location": {
      "lon": -4.7237,
      "lat": 41.6552
    }
  },
  "port": 22,
  "ip": "157.88.123.10"
},
```


Imagen 6.13: Destino del ataque de escaneo


```
"source": {
  "geo": {
    "continent_name": "Europe",
    "region_iso_code": "LT-16",
    "city_name": "Kaunas",
    "country_iso_code": "LT",
    "timezone": "Europe/Vilnius",
    "ip": "185.36.81.58",
    "country_name": "Republic of Lithuania",
    "region_name": "Kaunas",
    "location": {
      "lon": 23.9002,
      "lat": 54.9002
    }
  },
  "port": 42552,
  "ip": "185.36.81.58"
},
```


Imagen 6.14: Fuente del ataque de escaneo

El ataque de escaneo estudiado tiene como objetivo el puerto 22 de la IP 157.88.123.125 y que proviene de la dirección IP 185.36.81.58 que se localiza en la ciudad de Kaunas en Lituania.

Respecto a la información que se nos proporciona en el *log* a cerca de la regla empleada, podemos destacar los campos mostrados en la tabla 6.4.

| **Campo** | **Valor** |
|---|---|
| Nombre | *ET SCAN Potential SSH Scan* |
| Categoría | Intento de filtrado de información |
| Conjunto de reglas | *Emerging Threats* |
| Severidad | 2 (Media) |
| Fecha creación | 2010-07-30 |

Tabla 6.4: Campos de alerta de escaneo

En este caso concreto, el experto en seguridad ya dispone de numerosos datos sobre los ataques de escaneo que sufre el Departamento de Valladolid y puede establecer una forma de responder a este potencial problema de seguridad.

### Alertas de Zeek

En el caso de la herramienta Zeek se han generado un número cercano a casi 4 millones de archivos de *logs*, cuyo análisis ha obtenido únicamente 4 alertas. El bajo número de alertas obtenido se debe a que el tráfico ya está muy filtrado debido a los *firewalls* mencionados. Estas cuatro alertas están generadas por la regla *SSL:Invalid_Server_Cert*, la cual consiste en un fallo en la validación del certificado SSL. El certificado SSL permite autenticar una página web y con ello establecer una conexión segura. La información que nos muestra Kibana de estas alertas se puede observar en la imagen 6.15

| Time | source.ip | source.port | destination.ip | destination.port |
|---|---|---|---|---|
| May 11, 2021 @ 03:13:07.485 | 157.88.123.125 | 34364 | 209.132.183.108 | 443 |
| May 11, 2021 @ 03:13:07.484 | 157.88.123.125 | 34364 | 209.132.183.108 | 443 |
| May 11, 2021 @ 03:13:07.484 | 157.88.123.125 | 34364 | 209.132.183.108 | 443 |
| May 11, 2021 @ 03:13:07.481 | 157.88.123.125 | 34364 | 209.132.183.108 | 443 |

Imagen 6.15: Alertas Certificado SSL no valido

Centrándonos más en estas cuatro alertas, podemos ver que el origen de las peticiones a la página web proviene de la máquina de la Universidad de Valladolid con IP 157.88.123.138 y desde el puerto 34364. Respecto al servidor del cual queremos obtener la página web, conocemos que es una maquina con IP 209.132.183.108 que se encuentra situada en el estado de Arizona en Estados Unidos.

Si accedemos a esta dirección IP para comprobar si esta alerta es correcta, obtenemos el resultado observable en la imagen 6.16 que nos confirma que acceder a esa pagina web puede suponer un riesgo potencial de seguridad.

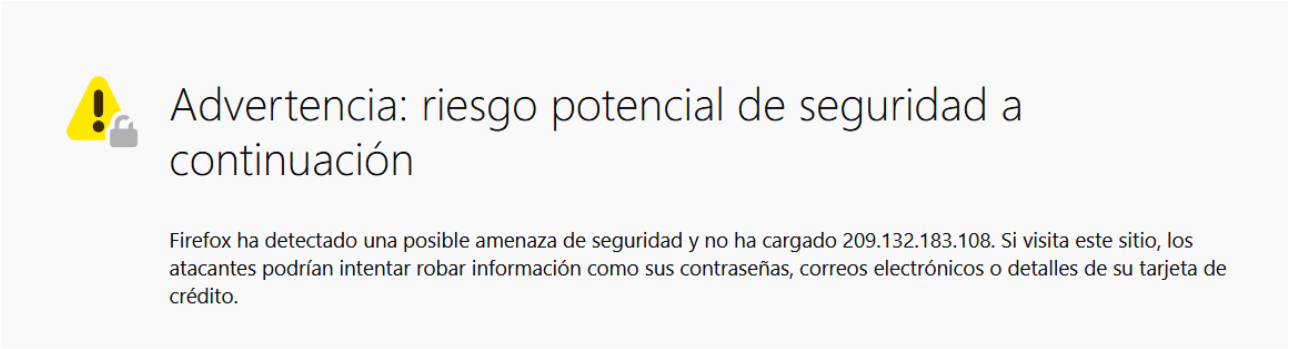


Imagen 6.16: Advertencia por certificado no valido

Como último paso para confirmar que la alerta es correcta, hemos observado que el navegador no confía en el emisor del certificado.

Con la detección de este tipo de alerta, los técnicos de seguridad tomarían la decisión de si es necesario intentar bloquearla la dirección IP del servidor al no considerarla fiable.

## 6.6 Monitorización del sistema

Una vez la plataforma ya se encuentra desplegada, es de vital importancia controlar su funcionamiento e intervenir el sistema en caso de necesidad. Para responder a esta necesidad Security Onion incluye el software Grafana. En nuestro despliegue Grafana nos proporciona una aplicación web para la visualización y análisis de métricas relacionadas con cada nodo de Security Onion instalado.

Entre las métricas que nos proporciona Grafana, podemos destacar el porcentaje de CPU empleado por el sistema y por cada herramienta, la entrada/salida sobre el disco y su porcentaje de uso, o el porcentaje de memoria empleado. Con el fin de mostrar de forma más visual su funcionamiento, vamos a exponer un ejemplo concreto que consistirá en monitorizar el nodo *manager* durante una semana de uso. Para ello, vamos a mostrar el conjunto de paneles obtenidos para ese nodo en Grafana con una breve descripción de su contenido:

- En la imagen 6.17 vemos que se nos permite monitorizar el porcentaje de uso de CPU, así como el de disco en el directorio raíz y en el directorio /nsm. También cabe mencionar que se nos muestra el tiempo que lleva activo el nodo, en este caso 2.05 semanas.

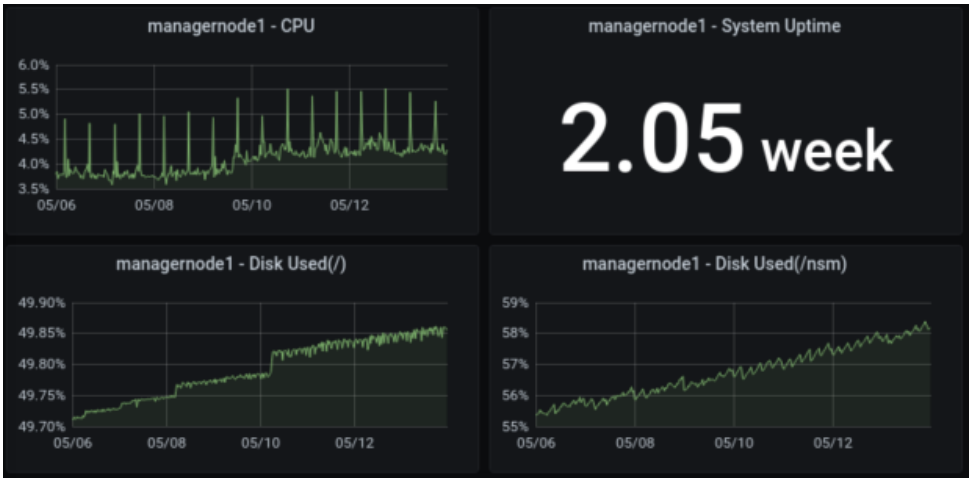


Imagen 6.17: CPU total y porcentaje del disco empleado

> **Nota 6.6.1** La monitorización del directorio /nsm es de vital importancia, esto se debe a que crece a gran velocidad puesto que es el lugar de almacenamiento de los *logs* generados en el nodo.

- En el caso de la imagen 6.18, se nos muestra el porcentaje de uso de CPU empleado en entrada/salida y en las herramientas del sistema Kibana, Logstash e InfluxDB.

**Nota 6.6.2** InfluxDB es la base de datos que emplea Grafana.

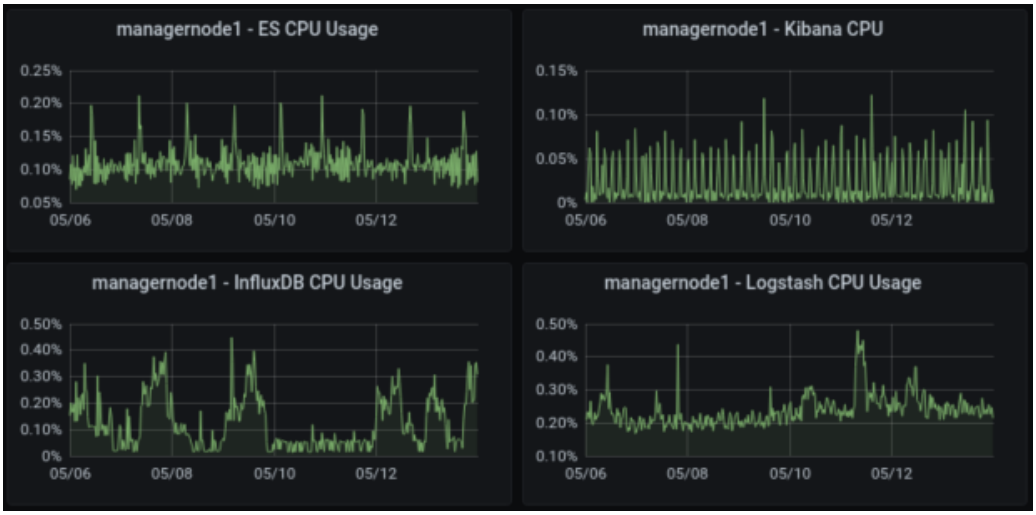


Imagen 6.18: Porcentajes de CPU por tarea

- La imagen 6.19 nos muestra información del funcionamiento de Redis. Para este nodo disponemos del número de *logs* presente en la cola de *logs* generada, así como de la memoria y CPU que emplea. Como aspecto a mayores, este panel muestra el tamaño de la base de datos de InfluxDB.

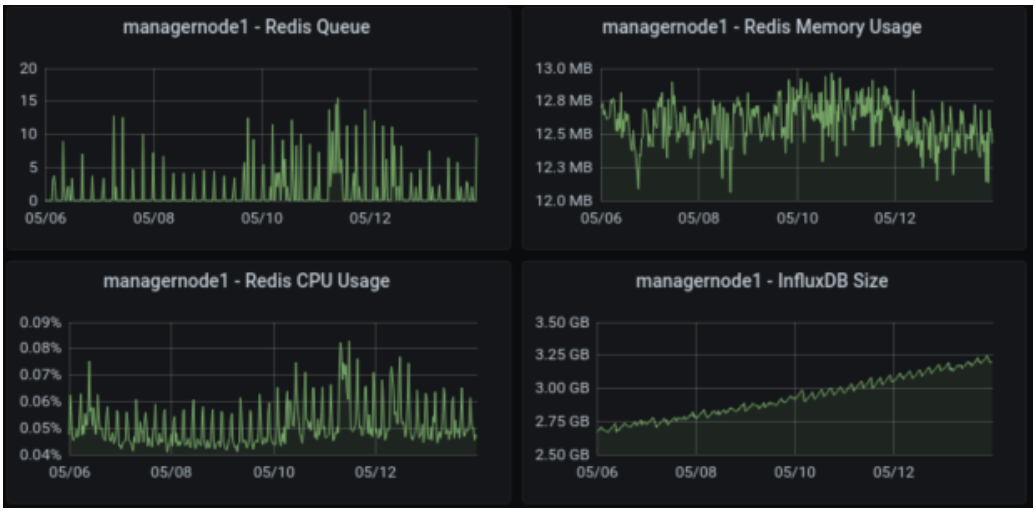


Imagen 6.19: Métricas de Redis

- En términos de uso de memoria, podemos observar en la imagen 6.20, el valor disponible y utilizado de memoria, así como la función para la que se está empleando.

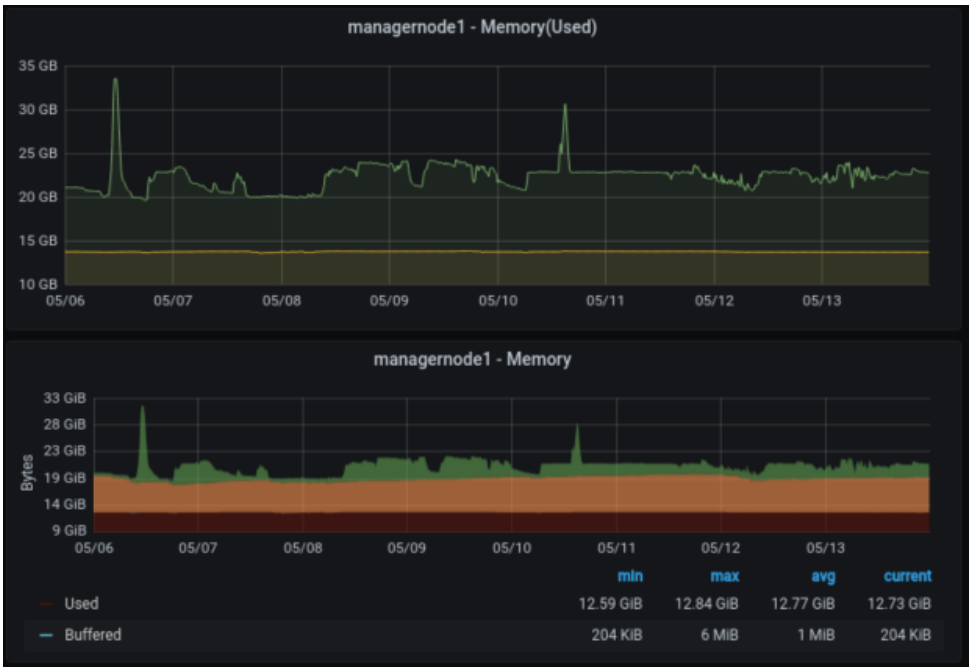


Imagen 6.20: Uso de memoria

- Respecto al uso de disco de nuestro nodo disponemos del panel de la imagen 6.21, en el cual se muestra los *mebibytes (MiB)* escritos y leídos del disco a lo largo de la semana. También se nos proporciona el número de eventos estimados por segundo que están ocurriendo en nuestro nodo. En la imagen de ejemplo observamos los picos de eventos ocurridos en los días laborables de la semana.

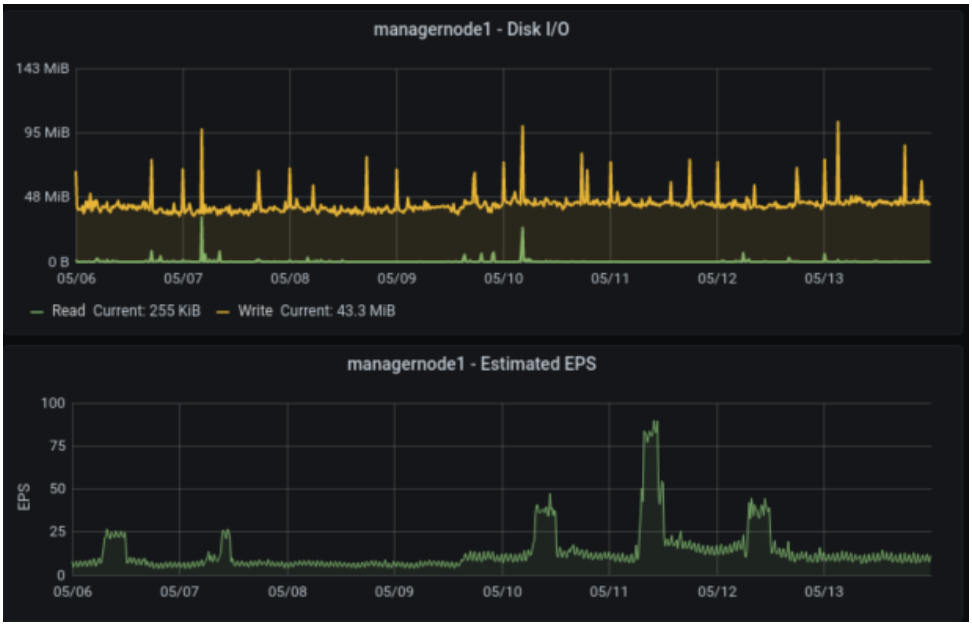


Imagen 6.21: Entrada/salida de disco y eventos del sistema

A mayores de los indicados, existen otros paneles informativos que hemos considerado de menor relevancia. Estos paneles nos presentan datos como el número de procesos o hilos del nodo o la carga

media del nodo.

Mediante el control de los valores de cada métrica seremos capaces de detectar si es necesario aumentar determinados recursos como la memoria o el disco, así como también lograremos identificar la causa en casos de problemas de rendimiento.

# 7. Pruebas

En la sección anterior hemos mostrado las distintas alertas que se han generado por nuestro sistema durante una única semana de estudio y en un entorno en el que ya existen medidas de seguridad. Estos condicionantes conllevan que solo se está poniendo a prueba la plataforma de detección para un pequeño porcentaje de los potenciales ataques que podría sufrir el Departamento de Informática de la Universidad de Valladolid.

Con el fin de resolver esta problemática, vamos a probar que el sistema genera alertas para distintos ataques simulados. La simulación de ataques se realizará mediante el uso de archivos PCAP que contienen paquetes de red. El contenido de estos paquetes simula la realización de ataques concretos sobre nuestra red.

## 7.1 Conjunto de datos

Como hemos descrito, necesitaremos una serie de archivos PCAP que serán obtenidos de una biblioteca [56] centrada en este tipo de archivos con tráfico malicioso, los archivos elegidos son los siguientes:

- PCAP de análisis de tráfico de red del 2021-02-08, que denominaremos PCAP_2021-02-08

  **Nota 7.1.1** Obtención del PCAP:

  wget https://www.malware-traffic-analysis.net/2021/02/08/2021-02-08-traffic-analysis-exercise.pcap.zip

- PCAP de análisis de tráfico de red del 2021-01-21, que denominaremos PCAP_2021-01-

21

**Nota 7.1.2** Obtención del PCAP:

wget https://www.malware-traffic-analysis.net/2021/01/21/2021-01-21-traffic-analysis-exercise.pcap.zip

## 7.2 Importación de los datos

En la sección 3.1.2 hemos detallado los distintos tipos de arquitectura Security Onion. Según la documentación de Security Onion, para realizar la importación de PCAPS es preferible emplear una arquitectura import. Este tipo de despliegue nos permite generar automáticamente alertas sin la necesidad de realizar modificaciones en Suricata, algo que si sería necesario realizar en nuestro despliegue distribuido. A continuación, vamos a mostrar los pasos llevados a cabo para la instalación de la arquitectura import:

- **Clonación de la máquina modelo.**
- **Elección y configuración de la arquitectura.** Se ha empleado el ejecutable so-setup-network de forma similar a cualquier otro tipo de nodo desplegado anteriormente. Como únicas configuraciones que no hayan sido mostradas destaca la elección de la arquitectura y el hostname. Estas configuraciones se pueden observar en las imágenes 7.1 y 7.2.

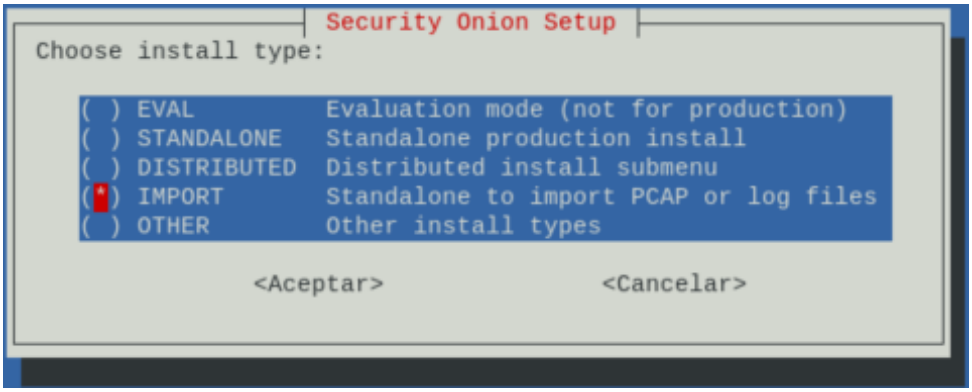


Imagen 7.1: Arquitectura import

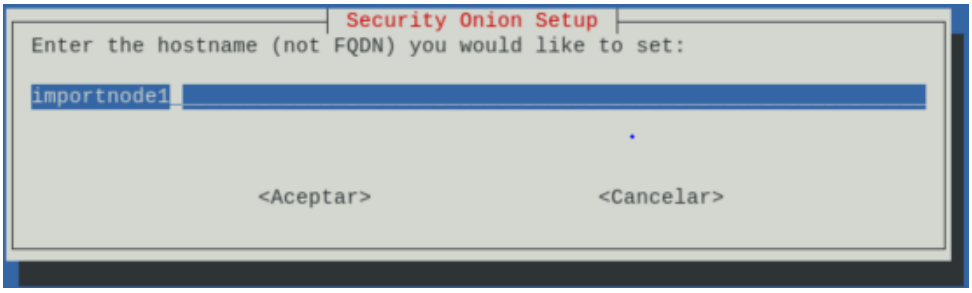


Imagen 7.2: Hostname import

- **Creación máquina analista.** Se ha creado una máquina de analista dentro del nodo Import.

Aunque esta máquina se ha desplegado de forma similar a la explicada anteriormente, como aspecto diferenciador destaca que se encuentra dentro del propio nodo y que por este motivo no es necesario permitir la máquina de analista en el firewall.

Para la generación de *logs* el tráfico contenido dentro del PCAP será enviado a la interfaz de la que dispone el nodo import, simulando de esta forma que se están realizando los ataques durante un funcionamiento normal de la red. Para realizar esta función, disponemos de dos opciones que son el comando so-import-pcap y tcpreplay. Se ha seleccionado so-import-pcap puesto que tiene en cuenta la fecha del tráfico enviado a la interfaz, lo que nos permitirá posteriormente un filtrado por fechas en Kibana. Para los dos conjuntos de datos los comandos empleados se pueden observar en la tabla 7.1.

| **PCAP** | **Comando** |
|---|---|
| PCAP_2021-02-08 | so-import-pcap 2021-02-08-traffic-analysis-exercise.pcap |
| PCAP_2021-01-21 | so-import-pcap 2021-01-21-traffic-analysis-exercise.pcap |

Tabla 7.1: Comandos empleados por PCAP

Una vez hemos realizado la importación de los datos, el PCAP introducido es analizado por Suricata y Zeek en busca de alertas, las cuales nos serán accesibles a través de Kibana y de las opciones Hunt, PCAP y Alert de Security Onion. A mayores de las alertas también dispondremos de *logs* de Zeek especificándonos información sobre el tráfico de red contenido en el PCAP.

## 7.3 Análisis de alertas obtenidas

En esta sección vamos a explicar las alertas obtenidas en cada uno de los PCAP y de esta forma demostrar la variedad de ataques que podemos detectar. Para cada uno de ellos destacaremos aquellas alertas que se consideran relevantes debido a su frecuencia de aparición o severidad.

### Análisis del PCAP_2021-02-08

En el caso de este PCAP, la plataforma de detección ha generado 158 alertas de red como se puede observar en la imagen 7.3.

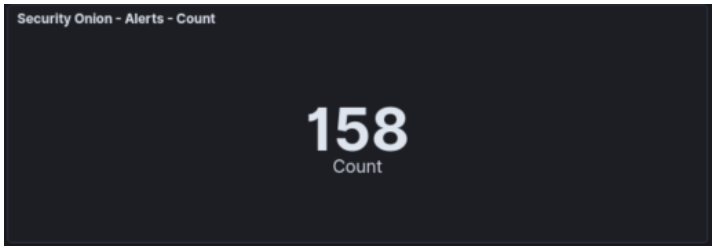


Imagen 7.3: Alertas generadas PCAP_2021-02-08

Respecto a la severidad de las alertas, la cual se muestra en la imagen 7.4, podemos destacar la presencia de varias alertas de severidad alta, las cuales si suponen un problema inmediato en nuestra red.

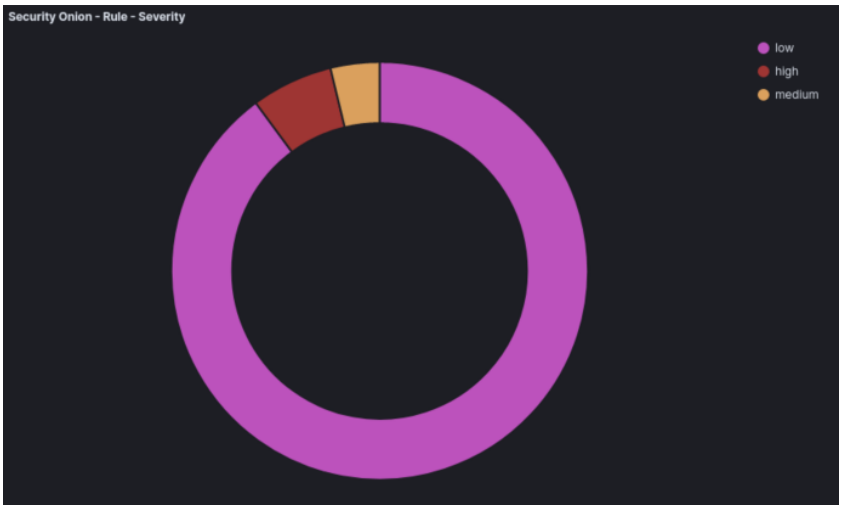


Imagen 7.4: Severidad alertas PCAP_2021-02-08

Como análisis previo de los resultados, vemos que este PCAP se caracteriza por presentarnos alertas variadas que se corresponden a distintos tipos de ataque sobre nuestra red. Las alertas obtenidas se pueden observar en la tabla 7.2.

| **Regla** | **Nº de alertas** |
|---|---|
| ET JA3 Hash - [Abuse.ch] Possible Dridex | 141 |
| ET HUNTING SUSPICIOUS POST with Fake Browser | 5 |
| ET MALWARE Win32/Ficker Stealer Activity | 4 |
| ET PHISHING Lets Encrypt Free SSL- Possible Phishing | 2 |
| ET POLICY DNS Update From External net | 2 |
| ET INFO Packed Executable Download | 1 |
| ET POLICY External IP Lookup (ipify .org) | 1 |
| ET POLICY External IP Lookup api.ipify.org | 1 |
| ET POLICY PE EXE or DLL Windows file download HTTP | 1 |

Tabla 7.2: Reglas empleadas PCAP 2021-02-08

Vamos a proceder a explicar las reglas de mayor relevancia en este PCAP, de esta forma estableceremos que tipos de ataque estamos detectando:

- **ET JA3 Hash - [Abuse.ch] Possible Dridex.** Esta regla nos indica que existe la posibilidad de que se haya ejecutado el malware Dridex en nuestra red. Dridex se especializa en obtener credenciales bancarias, para ello realiza envíos masivos de correos con archivos Word y Excel corruptos. Una vez el usuario abra estos archivos, el malware ya se encontrará funcionando en nuestro dispositivo.

- **ET HUNTING SUSPICIOUS POST to Dotted Quad with Fake Browser.** Este caso se obtiene debido a la recepción de peticiones POST realizadas por navegadores falsos. Estas peticiones pueden formar parte de un posible ataque de negación de servicio realizado por bots sobre nuestra red.
- **ET MALWARE Win32/Ficker Stealer Activity.** La regla Ficker Stealer Activiy nos muestra que existe actividad en nuestra red del malware Ficker. Este malware, simula aplicaciones legitimas mediante páginas web falsas, entre alguna de estas aplicaciones podemos destacar Microsoft Store o Spotify. En el momento en el que ejecutamos los archivos que se nos ofrecen en esas páginas web falsas, los atacantes ya tienen la posibilidad de robar los datos de nuestro sistema.
- **ET PHISHING Lets Encrypt Free SSL- Possible Phishing.** La entidad Lets Encrypt se encarga de proporcionar certificados con los que garantizar la seguridad de las páginas web y evitar ataques de phising. Existen ciertos certificados de Lets Encrypt que han sido alterados y que hacen que consideremos fiables determinadas webs que en realidad se dedican al robo de información. Esta regla se ejecuta en los casos en los que se ha detectado uno de estos certificados modificados.

Respecto a las otras reglas no mencionadas, cabe destacar que se enmarcan dentro de las categorías ET POLICY y ET INFO. La categoría ET POLICY no se obtiene debido a un ataque en particular, sino que simplemente remarca algo que puede violar la política de seguridad establecida por el administrador de la red. Como casos concretos de ET POLICY en este PCAP, nos encontramos los casos siguientes:

- Actualización de servidor DNS desde una red Externa.
- Descarga de un fichero ejecutable de Windows mediante http.
- Intentos de obtención de IPs de nuestra red.

La comprobación de las alertas de este PCAP nos garantiza que nuestra plataforma está detectando correctamente diversos tipos de Malware, como Dridex y Ficker, además de estar analizando en casos de posibles peticiones sospechosas y ataques de phising mediante el uso de certificados no válidos.

#### Análisis del PCAP_2021-01-21

Tras la inyección de este PCAP al sistema y el posterior filtrado temporal en Kibana, hemos obtenido 349 alertas de red, como se puede ver en la imagen 7.5

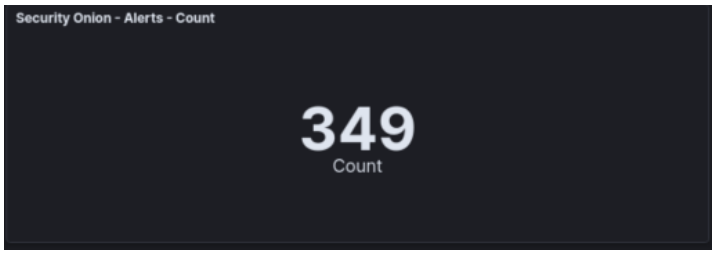


Imagen 7.5: Alertas generadas PCAP_2021-01-21

Tras un breve análisis de las reglas por las cuales se han generado esas alertas, hemos llegado a la conclusión de que este PCAP muestra todos los pasos realizados para lograr que nuestra infraestructura se encuentre infectada por malware. Mediante este ejemplo, se pretende demostrar que la plataforma es capaz de detectar las fases previas del ataque, pudiendo así evitarlo antes de que se realice. Las reglas empleadas se pueden observar en la tabla 7.3

| **Regla** | **Nº de alertas** |
|---|---|
| ET JA3 Hash - [Abuse.ch] Possible Gozi | 178 |
| ET JA3 Hash - [Abuse.ch] Possible Quakbot | 156 |
| ET MALWARE Likely Evil EXE download from MSXMLHTTP | 3 |
| ET INFO Dotted Quad Host RAR Request | 2 |
| ET MALWARE Ursnif Payload Request (grab32.rar) | 2 |
| ET POLICY DNS Update From External net | 2 |
| ET POLICY External IP Lookup Domain | 2 |
| ET HUNTING SUSPICIOUS Dotted Quad Host MZ Response | 1 |
| ET MALWARE Generic .bin download from Dotted Quad | 1 |
| ET MALWARE Zbot Generic URI/Header Struct .bin | 1 |
| ET POLICY PE EXE or DLL Windows file download HTTP | 1 |

Tabla 7.3: Reglas empleadas PCAP 2021-02-08

Las alertas obtenidas, nos indican que actualmente se están ejecutando dos troyanos, los cuales están centrados en la obtención de credenciales bancarias. Los nombres de estos troyanos son Gozi, también denominado Ursnif, y Quakbot. Las siguientes reglas, muestran las alertas obtenidas debido a la ejecución de los dos troyanos mencionados:

- ET JA3 Hash - [Abuse.ch] Possible Gozi
- ET JA3 Hash - [Abuse.ch] Possible Quakbot
- ET MALWARE Ursnif Payload Request (grab32.rar)

Como hemos mencionado, este PCAP nos informa de manera completa de la inclusión de estos troyanos. Debido a este hecho, disponemos de alertas que nos indican la descarga de los archivos maliciosos que forman los troyanos. Las reglas en las que podemos ver que se ha realizado la descarga del ejecutable que contiene el troyano son las siguientes:

- ET MALWARE Likely Evil EXE download from MSXMLHTTP

- ET MALWARE Generic .bin download from Dotted Quad

Si analizamos las alertas de forma individual, se observa que los archivos maliciosos se han descargado en nuestro sistema mediante peticiones MSXMLHTTP realizadas de forma no deseada. La descarga del troyano se ve de forma clara en el mensaje asociado a la alerta, puesto que en él vemos un archivo files en la petición de nombre 1.bin que se corresponde con el mostrado en las alertas de ejecución de los troyanos. Este mensaje asociado se puede observar en la parte final del anexo 9.4.

# 8. Conclusiones y Trabajo Futuro

## 8.1 Conclusiones

Durante el desarrollo de este trabajo se ha logrado cumplir el objetivo de diseñar y desplegar una plataforma de detección de amenazas de red que se ejecute de forma práctica sobre el Departamento de Informática la Universidad de Valladolid. La consecución del objetivo principal se ha logrado de forma incremental, permitiéndome obtener a lo largo de su realización un mejor conocimiento del funcionamiento de las redes y de las medidas de seguridad empleadas, así como también experiencia muy valiosa sobre una gran cantidad de herramientas como Kibana, Zeek, Suricata o Security Onion.

El análisis de los resultado obtenidos debido al funcionamiento de la plataforma de detección en el Departamento de Informática y de los archivos PCAP inyectados, nos permite detallar la situación en términos de seguridad de la red del Departamento de Informática.

Debido al bajo número de alertas generadas unido a la baja severidad media de ellas, podemos determinar que la red del Departamento de la Universidad de Valladolid dispone de las suficientes medidas de seguridad como para ser capaz de evitar la mayoría de los ataques que se realizan sobre ella.

Esta confirmación está condicionada debido al filtrado realizado por los *firewalls* incorporados en la UVA y al insuficiente número de recursos disponible para la creación de las máquinas virtuales. Estos dos aspectos nos han permitido únicamente obtener una parte del tráfico potencial que recorre el Departamento de Informática de la Universidad de Valladolid.

En cuanto a potenciales problemas de seguridad que se han detectado en la red, se puede destacar la presencia de una numerosa cantidad de ataques de escaneo mediante SSH proveniente de diversas

fuentes. El estudio de las alertas asociadas a este ataque de escaneo nos permitirán conocer la fuentes de la que proviene el ataque, así como también nos mostrará los equipos del departamento que están sufriendo el ataque. Esta información nos permitirá establecer cual es la mejor solución a realizar, que para este ataque de escaneo podría ser la comprobación de los puertos abiertos del sistema atacado o el bloqueo en el firewall de las direcciones IP de los atacantes.

Respecto a la funcionalidad que nos ofrece la plataforma, queda demostrado mediante las pruebas realizadas que es capaz de detectar problemas de seguridad como ataques de phising o intentos de incluir malware en nuestros dispositivos. Enfocándonos en la cuestión relativa al malware, se puede confirmar que la implementación de la plataforma de detección de forma permanente nos va a permitir descubrir la existencia de cualquier tipo de programa malicioso ejecutándose sobre nuestra red, así como también va a alertarnos de intentos de infección futuros incluso de forma previa a que tengan impacto sobre nuestra red.

## 8.2 Trabajo futuro

En relación al trabajo futuro, sería una opción interesante el desplegar la plataforma para la totalidad de la Universidad de Valladolid, para ello sería necesario la creación de un diseño de Security Onion con un mayor número de nodos sensores, *search* y *manager*, así como la instalación de agentes en una gran cantidad de *hosts* de la universidad. En cuestión de despliegue de Security Onion también sería interesante realizar un despliegue en *Cloud* empleando la tecnología AWS.

Como otros aspectos a implementar, destacaría la incorporación de sistemas que nos permitan responder de forma automática a las alertas generadas y la realización de mejoras en la detección en el sistema actual. Entre estas mejoras en la detección, destacaría el desarrollo de un nuevo conjunto de reglas propio más avanzado y la implementación de *machine-learning* en los sistemas de detección empleados.

# Bibliografía

## Libros


[22] Carmen Hernández. *Garantia y Seguridad de la Información. Departamento de Informática. Universidad de Valladolid*. 2018 (citado en página 19).

[40] Ghafir Ibrahim et al. *A Survey on Network Security Monitoring Systems [en línea]*. 2016, pages 77–82. DOI: 10.1109/W-FiCloud.2016.30 (citado en páginas 30, 33).

[55] Velan Petr et al. *Network traffic characterisation using flow-based statistics [en línea]*. 2016, pages 907–912. DOI: 10.1109/NOMS.2016.7502924 (citado en página 84).


## Web


[1] Check Point Software Technologies Ltd 2020. *Informe ciberseguridad*. URL: https://resources.checkpoint.com/cyber-security-resources/cyber-security-report-2020. (último acceso: Febrero 2021) (citado en página 10).

[2] Alberto Sierra 2021. *Un ataque informático obliga al SEPE a suspender su actividad en toda España*. URL: https://www.vozpopuli.com/economia_y_finanzas/sepe-ataque-informatico.html. (último acceso: Junio 2021) (citado en página 10).

[3] Pierluigi Paganini 2020. *Reading the 2020 Cost of a Data Breach Report*. URL: https://www.cyberdefensemagazine.com/reading-the-2020-cost-of-a-data-breach-report/. (último acceso: Febrero 2021) (citado en página 10).

[4] European Union Agency for Cybersecurity 2021. *Threat Landscape*. URL: https://www.enisa.europa.eu/topics/threat-risk-management/threats-and-trends. (último acceso: Mayo 2021) (citado en página 10).

[5] Wikipedia 2021. *Seguridad de la información*. URL: https://es.wikipedia.org/wiki/Seguridad_de_la_informaci%C3%B3n. (último acceso: Mayo 2021) (citado en página 13).

[6] Wikipedia 2021. *Data integrity*. URL: https://en.wikipedia.org/wiki/Data_integrity. (último acceso: Mayo 2021) (citado en página 13).

[7] Wikipedia 2020. *Confidencialidad*. URL: https://es.wikipedia.org/wiki/Confidencialidad. (último acceso: Mayo 2021) (citado en página 13).

[8] *Seguridad Física Administrativa y Lógica - Desarrollo y Gestión de Seguridad de Redes*. 2008. URL: https://sites.google.com/site/desygestiondeseguridadderedes/home/seguridad-fisica-administrativa-y-logica. (último acceso: Abril 2021) (citado en página 14).

[9] Cyber Edu 2021. *What is Network Security*. URL: https://www.forcepoint.com/cyber-edu/network-security. (último acceso: Abril 2021) (citado en página 14).

[10] Cisco 2021. *What Is Network Security?* URL: https://www.cisco.com/c/en/us/products/security/what-is-network-security.html#~types. (último acceso: Abril 2021) (citado en página 14).

[11] Cisco 2021. *What is a Firewall*. URL: https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html. (último acceso: Abril 2021) (citado en página 14).

[12] KnowBe4 n.d. *What Is Phishing*. URL: https://www.phishing.org/what-is-phishing. (último acceso: Abril 2021) (citado en página 15).

[13] Webroot Inc. n.d. *What is Antivirus Software*. URL: https://www.webroot.com/us/en/resources/tips-articles/what-is-anti-virus-software. (último acceso: Abril 2021) (citado en página 15).

[14] Juan Pablo Cerón 2017. *La evolución de los Antivirus*. URL: https://www.arame.com.mx/blog/la-evolucion-de-los-antivirus/. (último acceso: Abril 2021) (citado en página 15).

[15] Chris Brook 2020. *What is User and Entity Behavior Analytics A Definition of UEBA, Benefits, How It Works, and More*. URL: https://digitalguardian.com/blog/what-user-and-entity-behavior-analytics-definition-ueba-benefits-how-it-works-and-more. (último acceso: Abril 2021) (citado en página 15).

[16] Illumio n.d. *What is Network Segmentation*. URL: https://www.illumio.com/cybersecurity-101/network-segmentation. (último acceso: Abril 2021) (citado en página 15).

[17] Amazon 2021. *Amazon Web Service*. URL: https://aws.amazon.com/es/security/. (último acceso: Abril 2021) (citado en página 16).

[18] Wikipedia 2021. *Data loss prevention software*. URL: https://en.wikipedia.org/wiki/Data_loss_prevention_software. (último acceso: Abril 2021) (citado en página 16).

[19] Wikipedia 2020. *Control de acceso a red*. URL: https://es.wikipedia.org/wiki/Control_de_acceso_a_red. (último acceso: Abril 2021) (citado en página 16).

[20] GoodFirms 2019. *What is Web Security*. URL: https://www.goodfirms.co/glossary/web-security/. (último acceso: Abril 2021) (citado en página 18).

[21] Ivan Belcic 2021. *¿Qué es la inyección de SQL y cómo funciona?* 2021. URL: https://www.avast.com/es-es/c-sql-injection. (último acceso: Abril 2021) (citado en página 18).

[23] *Difference Between Active and Passive Attacks (Comparison Chart)*. 2021. URL: https://alldifferences.net/difference-between-active-and-passive-attacks/. (último acceso: Marzo 2021) (citado en página 19).

[24] Todd Lammle 2010. *CCNA Wireless Study Guide*. URL: https://books.google.com.pk/books?id=3GPEdQeWB4sC&lpg=PA162&dq=can20we%20detect%20passive%20attacks%20using%20IDS&pg=PA162#v=onepage&q&f=true. (último acceso: Marzo 2021) (citado en página 19).

[25] Rebecca Bace 2001. *NIST Special Publication on Intrusion Detection Systems*. URL: https://apps.dtic.mil/sti/citations/ADA393326. (último acceso: Marzo 2021) (citado en página 19).

[26] Imperval 2021. *DDoS Attack Types  Mitigation Methods: Imperva." Learning Center*. URL: www.imperva.com/learn/ddos/ddos-attacks/. (último acceso: Marzo 2021) (citado en página 19).

[27] Imperval 2020. *What Is a Smurf Attack: DDoS Attack Glossary: Imperva." Learning Center*. URL: www.imperva.com/learn/ddos/smurf-attack-ddos/. (último acceso: Marzo 2021) (citado en página 19).

[28] Incibe 2021. *Diseño y Configuración De IPS, IDS y SIEM En Sistemas De Control Industrial*. URL: www.incibe-cert.es/blog/diseno-y-configuracion-ips-ids-y-siem-sistemas-control-industrial. (último acceso: Abril 2021) (citado en páginas 21, 22).

[29] Jeff Petters 2020. *IDS vs. IPS: What is the Difference?* URL: https://www.varonis.com/blog/ids-vs-ips/. (último acceso: Febrero 2021) (citado en página 25).

[30] Tim Keary 2021. *IDS vs IPS - What's the Difference  Which do You Need*. URL: https://www.comparitech.com/net-admin/ids-vs-ips/#What_is_an_IDS_and_What_Does_it_Do. (último acceso: Febrero 2021) (citado en página 25).

[31] LogPoint 2021. *What is SIEM? A complete guide to SIEM: Benefits of a SIEM solution*. URL: https://www.logpoint.com/en/understand/what-is-siem/. (último acceso: Abril 2021) (citado en página 26).

[32] Imperva 2020. *What is SIEM: Security Information and Event Management Tools: Imperva*. URL: https://www.imperva.com/learn/application-security/siem/. (último acceso: Abril 2021) (citado en página 26).

[33] Techslang 2020. *What is SIEM - Definition by Techslang*. URL: https://www.techslang.com/definition/what-is-siem/. (último acceso: Abril 2021) (citado en página 26).

[34] Jeff Goldman 2018. *ArcSight vs Splunk: SIEM Product Comparison*. URL: https://www.esecurityplanet.com/products/arcsight-vs-splunk/. (último acceso: Abril 2021) (citado en páginas 26, 27).

[35] Check Point Software 2021. *What is SOC (Security Operation Center)*. URL: https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-soc/. (último acceso: Abril 2021) (citado en página 28).

[36] IBM 2021. *Security Information and Event Management (SIEM) Solutions*. URL: https://www.ibm.com/security/security-intelligence. (último acceso: Abril 2021) (citado en página 28).

[37] TrustRadius 2019. *Pros and Cons of IBM QRadar 2021*. URL: https://www.trustradius.com/products/ibm-qradar/reviews?qs=pros-and-cons. (último acceso: Abril 2021) (citado en página 28).

[38] Deepti Vidyarthi 2013 Surya Bhagavan Ambati. *A brief study and comparison of, open source intrusion detection system tools*. URL: http://www.iraj.in/journal/journal_file/journal_pdf/3-27-139087836726-32.pdf. (último acceso: Mayo 2021) (citado en página 29).

[39] Snort 2021. *Pagina web y documentación de Snort*. URL: https://www.snort.org/. (último acceso: Mayo 2021) (citado en página 29).

[41] Suricata 2021. *Documentación Suricata*. URL: https://suricata.readthedocs.io/en/latest/index.html. (último acceso: Mayo 2021) (citado en página 32).

[42] Zeek 2021. *Zeek Documentation*. URL: https://docs.zeek.org/en/master/. (último acceso: Mayo 2021) (citado en página 35).

[43] OSSEC Project Team 2021. *Documentación de Ossec*. URL: https://www.ossec.net/docs/. (último acceso: Mayo 2021) (citado en páginas 36, 37).

[44] Wazuh 2021. *Wazuh · The Open Source Security Platform*. URL: https://wazuh.com/. (último acceso: Mayo 2021) (citado en página 39).

[45] Tripwire 2021. *Security Configuration Management (SCM)*. URL: https://www.tripwire.com/products/tripwire-enterprise/. (último acceso: Mayo 2021) (citado en página 40).

[46] Hannes von Haugwitz 2010. *Documentación de AIDE*. URL: https://aide.github.io/. (último acceso: Mayo 2021) (citado en página 41).

[47] *24 Alternatives To Samhain, Pros, Cons  Questions*. 2020. URL: https://hackerspad.net/software/samhain/#pros. (último acceso: Mayo 2021) (citado en página 41).

[48] Security Onion 2021. *Security Onion Documentation*. URL: https://docs.securityonion.net/en/2.3/. (último acceso: Junio 2021) (citado en páginas 47, 48, 54–56).

[49] Elastic NV 2021. *ELK Stack: Elasticsearch, Logstash, Kibana*. URL: https://www.elastic.co/es/what-is/elk-stack. (último acceso: Mayo 2021) (citado en página 49).

[50] Osquery 2021. *Documentación de Osquery*. URL: https://osquery.readthedocs.io/en/stable/. (último acceso: Mayo 2021) (citado en página 51).

[51] Redis Labs 2021. *Redis*. URL: https://docs.securityonion.net/en/2.3/redis.html#redis. (último acceso: Mayo 2021) (citado en página 51).

[52] Ayush Jain 2020. *Redis vs. MySQL Benchmarks - DZone Database*. URL: https://dzone.com/articles/redis-vs-mysql-benchmarks. (último acceso: Mayo 2021) (citado en página 51).

[53] Gchq 2021. *CyberChef*. URL: https://github.com/gchq/CyberChef. (último acceso: Mayo 2021) (citado en página 53).

[54] Red Hat Inc. n.d. *Información sobre Dockers*. URL: https://www.redhat.com/es/topics/containers/what-is-docker. (último acceso: Mayo 2021) (citado en página 64).

[56] Malware-Traffic-Analysis.net 2021. *Fuente de Pcaps*. URL: `https://www.malware-traffic-analysis.net/index.html`. (último acceso: Mayo 2021) (citado en página 97).

# 9. Anexos

## 9.1 Ejemplo log de red


```
{
  "_index": "managernode1:so-zeek-2021.05.12",
  "_type": "_doc",
  "_id": "peUHY3kBJlSZx3lugbMy",
  "_version": 1,
  "_score": null,
  "_source": {
    "server": {
      "port": "67",
      "bytes": 600,
      "ip": "10.0.1.231",
      "ip_bytes": 656,
      "packets": 2
    },
    "log": {
      "file": {
        "path": "/nsm/zeek/logs/current/conn.log"
      },
      "offset": 27758,
```

```
      "id": {
        "uid": "C68d003wcbzxSUCeg1"
      }
    },
    "destination": {
      "port": 67,
      "ip": "10.0.1.231"
    },
    "source": {
      "port": 68,
      "ip": "255.255.255.255"
    },
    "network": {
      "protocol": "dhcp",
      "community_id": "1:0svgzrxr7mMGGAUzKNnwFzxf5tU=",
      "bytes": 600,
      "transport": "udp"
    },
    "ingest": {
      "timestamp": "2021-05-13T00:01:43.450Z"
    },
    "observer": {
      "name": "sensornode3"
    },
    "ecs": {
      "version": "1.6.0"
    },
    "@version": "1",
    "client": {
      "port": "68",
      "bytes": 0,
      "ip": "255.255.255.255",
      "ip_bytes": 0,
```

```
53      "packets": 0
54    },
55    "connection": {
56      "state_description": "Responder sent a SYN ACK followed by a FIN, we
            never saw a SYN from the originator",
57      "bytes": {
58        "missed": 0
59      },
60      "state": "SHR",
61      "history": "^d",
62      "local": {
63        "responder": true,
64        "originator": false
65      }
66    },
67    "event": {
68      "duration": 42.40098690986633,
69      "module": "zeek",
70      "category": "network",
71      "dataset": "conn"
72    },
73    "message": "{\"ts\":\"2021-05-12T23:59:59.863548Z\",\"uid\":\"C68d003
       wcbzxSUCeg1\",\"id.orig_h\":\"255.255.255.255\",\"id.orig_p\":68,\"
       id.resp_h\":\"10.0.1.231\",\"id.resp_p\":67,\"proto\":\"udp\",\"
       service\":\"dhcp\",\"duration\":42.40098690986633,\"orig_bytes\":0,
       \"resp_bytes\":600,\"conn_state\":\"SHR\",\"local_orig\":false,\"
       local_resp\":true,\"missed_bytes\":0,\"history\":\"^d\",\"orig_pkts
       \":0,\"orig_ip_bytes\":0,\"resp_pkts\":2,\"resp_ip_bytes\":656,\"
       community_id\":\"1:0svgzrxr7mMGGAUzKNnwFzxf5tU=\"}",
74    "tags": [
75      "beats_input_codec_plain_applied"
76    ],
77    "@timestamp": "2021-05-12T23:59:59.863Z"
```

```
  },
  "fields": {
    "@timestamp": [
      "2021-05-12T23:59:59.863Z"
    ],
    "Push to TheHive": [
      "https://managernode1/soctopus/thehive/case/peUHY3kBJlSZx3lugbMy"
    ]
  },
  "sort": [
    1620863999863
  ]
}
```

## 9.2 Ejemplo log de host


```
{
  "_index": "managernode1:so-ossec-2021.05.12",
  "_type": "_doc",
  "_id": "iuUEY3kBJlSZx3lu1a6i",
  "_version": 1,
  "_score": null,
  "_source": {
    "agent": {
      "name": "sensornode5-wazuh-manager",
      "id": "000"
    },
    "manager": {
      "name": "sensornode5-wazuh-manager"
    },
    "log": {
      "file": {
        "path": "/wazuh/archives/archives.json"
      },
      "offset": 24749237,
      "location": "df -P",
      "id": {
        "id": "1620863925.4698"
      },
      "full": "ossec: output: 'df -P': /dev/sdd1         20960256 18647744
           2312512      89% /var/ossec/data"
    },
    "decoder": {},
    "message": "{\"timestamp\":\"2021-05-12T23:58:45.367+0000\",\"agent\":
       {\"id\":\"000\",\"name\":\"sensornode5-wazuh-manager\"},\"manager\"
       :{\"name\":\"sensornode5-wazuh-manager\"},\"id\":\"1620863925.4698
       \",\"full_log\":\"ossec: output: 'df -P': /dev/sdd1         2096025
       6 18647744   2312512      89% /var/ossec/data\",\"decoder\":{\"name
```

```
      \":\"ossec\"},\"location\":\"df -P\"}",
  "tags": [
    "beats_input_codec_plain_applied"
  ],
  "@timestamp": "2021-05-12T23:58:48.640Z",
  "ecs": {
    "version": "1.6.0"
  },
  "@version": "1",
  "host": {
    "name": "sensornode5"
  },
  "event": {
    "code": "",
    "module": "ossec",
    "category": "host",
    "dataset": "ossec",
    "timestamp": "2021-05-12T23:58:45.367+0000"
  }
},
"fields": {
  "@timestamp": [
    "2021-05-12T23:58:48.640Z"
  ],
  "Push to TheHive": [
    "https://managernode1/soctopus/thehive/case/iuUEY3kBJlSZx3lu1a6i"
  ]
},
"highlight": {
  "event.category.keyword": [
    "@kibana-highlighted-field@host@/kibana-highlighted-field@"
  ]
},
```

```
  "sort": [
    1620863928640
  ]
}
```

## 9.3 Ejemplo alerta de red


```
{
  "_index": "managernode1:so-ids-2021.05.12",
  "_type": "_doc",
  "_id": "IeX4YnkBJlSZx3lu7plG",
  "_version": 1,
  "_score": null,
  "_source": {
    "log": {
      "file": {
        "path": "/nsm/suricata/eve-2021-05-12-23:23.json"
      },
      "offset": 13887,
      "id": {
        "uid": "1045948698930669"
      }
    },
    "destination": {
      "geo": {
        "continent_name": "Europe",
        "region_iso_code": "ES-VA",
        "city_name": "Valladolid",
        "country_iso_code": "ES",
        "timezone": "Europe/Madrid",
        "ip": "157.88.123.10",
        "country_name": "Spain",
        "region_name": "Valladolid",
```

```
      "location": {
        "lon": -4.7237,
        "lat": 41.6552
      }
    },
    "port": 8080,
    "ip": "157.88.123.10"
  },
  "rule": {
    "severity": 2,
    "reference": "https://doc.emergingthreats.net/2403378",
    "rev": 65801,
    "metadata": {
      "affected_product": [
        "Any"
      ],
      "attack_target": [
        "Any"
      ],
      "updated_at": [
        "2021_05_11"
      ],
      "created_at": [
        "2013_10_08"
      ],
      "tag": [
        "CINS"
      ],
      "signature_severity": [
        "Major"
      ],
      "deployment": [
        "Perimeter"
```

```
      ]
    },
    "gid": 1,
    "name": "ET CINS Active Threat Intelligence Poor Reputation IP group
        79",
    "ruleset": "Emerging Threats",
    "action": "allowed",
    "rule": "alert ip [80.253.20.94,80.38.139.178,80.50.129.130,80.65.30
       .15,80.65.31.68,80.73.91.130,80.76.185.208,80.82.70.118,80.82.77.
       139,80.82.77.144,80.82.77.227,80.82.77.240,80.82.77.245,80.82.77.
       33,80.85.86.31,80.87.192.169,80.89.151.114,80.93.210.82,80.94.93.
       14,80.94.93.24,80.94.93.33,81.105.223.91,81.12.92.178,81.12.94.12
       2,81.144.241.221,81.16.250.105,81.161.63.100,81.161.63.103,81.161
       .63.253,81.167.205.200,81.167.244.222,81.169.203.68,81.169.247.18
       5,81.17.57.50,81.180.84.246,81.211.16.42,81.211.8.42,81.213.154.2
       0,81.213.199.205,81.214.184.247,81.22.103.129,81.22.98.8,81.225.1
       60.128,81.23.151.223,81.245.100.81,81.27.219.254,81.28.45.130,81.
       4.122.254,81.68.104.162,81.68.106.25] any -> $HOME_NET any (msg:
       \"ET CINS Active Threat Intelligence Poor Reputation IP group 79
       \"; reference:url,www.cinsscore.com; threshold: type limit, track
        by_src, seconds 3600, count 1; classtype:misc-attack; sid:240337
       8; rev:65801; metadata:affected_product Any, attack_target Any,
       deployment Perimeter, tag CINS, signature_severity Major,
       created_at 2013_10_08, updated_at 2021_05_11;)",
    "category": "Misc Attack",
    "uuid": "2403378"
  },
  "source": {
    "geo": {
      "continent_name": "Europe",
      "country_iso_code": "GB",
      "timezone": "Europe/London",
      "ip": "80.94.93.14",
```

```
        "country_name": "United Kingdom",
        "location": {
          "lon": -0.1224,
          "lat": 51.4964
        }
      },
      "port": 46971,
      "ip": "80.94.93.14"
    },
    "network": {
      "community_id": "1:BVtgllKLcQr/5h8qVy/OeMSfUnA=",
      "data": {
        "decoded": ""
      },
      "transport": "TCP"
    },
    "ingest": {
      "timestamp": "2021-05-12T23:45:46.642Z"
    },
    "observer": {
      "name": "sensornode2"
    },
    "ecs": {
      "version": "1.6.0"
    },
    "@version": "1",
    "host": {
      "name": "sensornode2"
    },
    "event": {
      "severity": 2,
      "module": "suricata",
      "category": "network",
```

```
    "dataset": "alert",
    "severity_label": "medium"
  },
  "message": "{\"timestamp\":\"2021-05-12T23:45:43.994797+0000\",\"
    flow_id\":1045948698930669,\"in_iface\":\"bond0\",\"event_type\":\"
    alert\",\"vlan\":[710],\"src_ip\":\"80.94.93.14\",\"src_port\":4697
    1,\"dest_ip\":\"157.88.123.10\",\"dest_port\":8080,\"proto\":\"TCP
    \",\"community_id\":\"1:BVtgllKLcQr/5h8qVy/OeMSfUnA=\",\"alert\":{
    \"action\":\"allowed\",\"gid\":1,\"signature_id\":2403378,\"rev\":6
    5801,\"signature\":\"ET CINS Active Threat Intelligence Poor
    Reputation IP group 79\",\"category\":\"Misc Attack\",\"severity\":
    2,\"metadata\":{\"affected_product\":[\"Any\"],\"attack_target\":[
    \"Any\"],\"created_at\":[\"2013_10_08\"],\"deployment\":[\"
    Perimeter\"],\"signature_severity\":[\"Major\"],\"tag\":[\"CINS\"],
    \"updated_at\":[\"2021_05_11\"]},\"rule\":\"alert ip [80.253.20.94,
    80.38.139.178,80.50.129.130,80.65.30.15,80.65.31.68,80.73.91.130,80
    .76.185.208,80.82.70.118,80.82.77.139,80.82.77.144,80.82.77.227,80.
    82.77.240,80.82.77.245,80.82.77.33,80.85.86.31,80.87.192.169,80.89.
    151.114,80.93.210.82,80.94.93.14,80.94.93.24,80.94.93.33,81.105.223
    .91,81.12.92.178,81.12.94.122,81.144.241.221,81.16.250.105,81.161.6
    3.100,81.161.63.103,81.161.63.253,81.167.205.200,81.167.244.222,81.
    169.203.68,81.169.247.185,81.17.57.50,81.180.84.246,81.211.16.42,81
    .211.8.42,81.213.154.20,81.213.199.205,81.214.184.247,81.22.103.129
    ,81.22.98.8,81.225.160.128,81.23.151.223,81.245.100.81,81.27.219.25
    4,81.28.45.130,81.4.122.254,81.68.104.162,81.68.106.25] any ->
    $HOME_NET any (msg:\\\"ET CINS Active Threat Intelligence Poor
    Reputation IP group 79\\\"; reference:url,www.cinsscore.com;
    threshold: type limit, track by_src, seconds 3600, count 1;
    classtype:misc-attack; sid:2403378; rev:65801; metadata:
    affected_product Any, attack_target Any, deployment Perimeter, tag
    CINS, signature_severity Major, created_at 2013_10_08, updated_at 2
    021_05_11;)\"},\"payload_printable\":\"\",\"stream\":0,\"packet\":
    \"AIAncQAQAB5KN44ACABFpAAoorkAAOwGZaNQXl0OnVh7Crd7H5Cl/8
```

```
      YjAAAAAFACBACi5AAAAAAAAAAA\",\"packet_info\":{\"linktype\":1}}",
    "tags": [
      "beats_input_codec_plain_applied"
    ],
    "@timestamp": "2021-05-12T23:45:43.994Z"
  },
  "fields": {
    "@timestamp": [
      "2021-05-12T23:45:43.994Z"
    ],
    "Push to TheHive": [
      "https://managernode1/soctopus/thehive/case/IeX4YnkBJlSZx3lu7plG"
    ]
  },
  "highlight": {
    "event.module": [
      "@kibana-highlighted-field@suricata@/kibana-highlighted-field@"
    ],
    "event.dataset": [
      "@kibana-highlighted-field@alert@/kibana-highlighted-field@"
    ]
  },
  "sort": [
    1620863143994
  ]
}
```


## 9.4 Mensaje alerta por descarga de troyano


```
{"timestamp":"2021-01-20T23:40:38.527484+0000","flow_id":1078349241093749,
   "pcap_cnt":1365,"event_type":"alert","src_ip":"209.141.51.196","
   src_port":80,"dest_ip":"10.1.21.101","dest_port":49723,"proto":"TCP","
   metadata":{"flowbits":["http.dottedquadhost","et.IE7.NoRef.NoCookie","
```

```
et.MS.XMLHTTP.no.exe.request","et.MS.XMLHTTP.ip.request","ET.http.
binary"]},"community_id":"1:LtLr2zVC3qduTsP2Q7Csk2HEhFo=","tx_id":1,"
alert":{"action":"allowed","gid":1,"signature_id":2022053,"rev":2,"
signature":"ET MALWARE Likely Evil EXE download from MSXMLHTTP non-exe
extension M2","category":"A Network Trojan was detected","severity":1,"
metadata":{"created_at":["2015_11_09"],"former_category":["
CURRENT_EVENTS"],"updated_at":["2015_11_09"]},"rule":"alert http
$EXTERNAL_NET any -> $HOME_NET any (msg:\"ET MALWARE Likely Evil EXE
download from MSXMLHTTP non-exe extension M2\"; flow:established,
to_client; file_data; content:\"MZ\"; within:2; byte_jump:4,58,relative
,little; content:\"PE|00 00|\"; distance:-64; within:4; flowbits:isset,
et.MS.XMLHTTP.no.exe.request; classtype:trojan-activity; sid:2022053;
rev:2; metadata:created_at 2015_11_09, former_category CURRENT_EVENTS,
updated_at 2015_11_09;)"},"files":[{"filename":"/files/1.bin","sid":[],
"gaps":false,"state":"TRUNCATED","stored":false,"size":39974,"tx_id":1}
]}
```