Universitat Oberta de Catalunya

uoc.edu

Desplegar la herramienta "Zeek IDS" y su posterior explotación para el análisis de actividades sospechosas en la red

Ignacio Galván Vitas
Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones
Análisis de datos

Borja Guaita Pérez
Helena Rifà Pous

Junio 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Desplegar la herramienta "Zeek IDS" y su posterior explotación para el análisis de actividades sospechosas en la red*
Nombre del autor:	*Ignacio Galván Vitas*
Nombre del consultor/a:	*Borja Guaita Pérez*
Nombre del PRA:	*Helena Rifà Pous*
Fecha de entrega (mm/aaaa):	06/2020
Titulación:	*Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones*
Área del Trabajo Final:	*Análisis de datos*
Idioma del trabajo:	*Español*
Palabras clave	*zeek, malware, elastic*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El objetivo principal planteado era la evaluación de la capacidad de una herramienta IDS como Zeek combinada con un entorno de visualización y explotación de datos como ELK Stack para analizar actividades sospechosas en la red. Adicionalmente se han integrado distintas fuentes de inteligencia de amenazas y se han correlacionado a través de ELK Stack con los datos obtenidos de Zeek IDS.

Este documento describe todo el proceso llevado a cabo para diseñar, planificar, desplegar la solución y, posteriormente, analiza los resultados obtenidos. Adicionalmente incluye el código fuente de los distintos «scripts», «playbooks» de Ansible y ficheros de configuración de las distintas aplicaciones utilizadas.

La metodología de gestión de proyectos en cascada ha permitido mantener un control constante del proyecto al mismo tiempo que ha facilitado su gestión.

Las principales conclusiones que se han obtenido es que es totalmente posible y viable disponer de un sistema de detección de intrusos utilizando como base Zeek IDS y el conjunto ELK Stack. Además también se ha confirmado que este tipo de soluciones se puede desarrollar mediante herramientas de código abierto y manteniendo un bajo presupuesto. Por último también se ha concluido que es sencillo integrar fuentes de información sobre amenazas y que están sean correlacionadas con la información obtenida de Zeek.

Abstract (in English, 250 words or less):

The main objective was to assess the ability of an IDS tool like Zeek combined with a data visualization and exploitation environment like ELK Stack to analyze suspicious activity on the network. Additionally, different sources of threat intelligence have been integrated and correlated through ELK Stack with the data obtained from Zeek IDS.

This document describes the entire process carried out to design, plan, deploy the solution and then analyze the results obtained. It also includes the source code of the different scripts, Ansible playbooks and configuration files of the different applications used.

The waterfall project management methodology has allowed to maintain a constant control of the project while facilitating its management.

The main conclusions obtained are that it is totally possible and viable to have an intrusion detection system using Zeek IDS and the ELK Stack as a base. It has also been confirmed that this type of solution can be developed using open source tools and maintaining a low budget. Finally, it has also been concluded that it is easy to integrate sources of information on threats and that they are correlated with the information obtained from Zeek.

Índice

Lista de figuras

1 Introducción

1.1 Contexto y justificación del Trabajo

Hasta no hace mucho nuestra primera línea de defensa en la seguridad de la red han sido los cortafuegos. El mejor símil que se me ocurre es que son como las murallas de una ciudad, creadas para proteger a los ciudadanos y controlar quién podía entrar en ellas.

Pero, al igual que las murallas, los cortafuegos no son impenetrables. Cada día que pasa, los diseñadores de software malicioso consiguen crear nuevos ataques diseñados para circunvalar nuestras defensas y aprovecharse de sus fallos.

Además, no todos los enemigos están fuera de las murallas, ojala fuera tan fácil. Muchas veces, la principal amenaza se encuentra entre nuestros propios ciudadanos. La era del «trae tu propio dispositivo» (BYOD), las continuas vulnerabilidades en el software o la incapacidad de muchas organizaciones para mantener su infraestructura tecnológica, ponen en riesgo a la organización tanto como lo hace el peor de los delincuentes informáticos.

Esto significa que ya no solo se necesita tener personal cualificado en materias de seguridad cuidando la muralla, sino que también necesitamos a ese personal monitorizando, analizando, comprobando y supervisando el estado de las instalaciones, de la electrónica de red, de los servidores, de las estaciones de trabajo y del demás equipamiento interconectado, sino que también se hace imprescindible examinar las actividades y acciones que se realizan dentro de la organización por parte de los usuarios.

Quizás en entornos pequeños sea posible contratar un pequeño grupo de estos expertos para que realicen estas actividades de forma manual, pero cuando la organización crece, se multiplican todos los elementos anteriores y con ellos, el tráfico de red. ¿Alguien se imagina a una persona evaluando capturas de tráfico de gigabits de tamaño durante todo el día?

Parece evidente que se necesita un sistema de vigilancia centralizado que permita controlar el tráfico de red para evaluar su comportamiento y avisar a los expertos para que puedan actuar donde sea necesario en caso de que se produzca una emergencia.

Es aquí donde tienen cabida los Sistemas de Detección de Intrusos o IDS por sus siglas en inglés. Estos sistemas se encargan de monitorizar el tráfico de red en busca de los patrones de comportamiento malicioso, almacenan en un sistema centralizado la información necesaria para hacerles un seguimiento y, al mismo tiempo, emiten las alertas necesarias.

Normalmente los IDS corporativos suelen ser elementos de software cerrado y con un coste elevado que generan una dependencia difícil de eliminar con un fabricante determinado y muchas veces poco accesible para pequeños negocios.

Este trabajo ha surgido de la necesidad de evaluar y validar una solución abierta y modular que evite la dependencia tecnológica y que permita monitorizar este tipo de actividad de forma automatizada y autónoma, con una

interfaz simple y visual y que tenga un coste contenido que pueda ser asumido incluso por las PYME.

1.2 Objetivos del Trabajo

El objetivo principal del presente trabajo es la evaluación de la capacidad de una herramienta IDS como Zeek combinada con un entorno de visualización y explotación de datos como ELK Stack para analizar actividades sospechosas en la red.

Como objetivos secundarios se han establecido los siguientes:

- Diseñar una solución basada el software comentado anteriormente que pueda ser desplegada en poco tiempo y de forma automatizada en cualquier arquitectura.
- Evaluar las capacidades de aprendizaje de máquina de las que dispone ELK Stack.
- Utilización de componentes de software libre y/o de fuentes abiertas que permitan la libre utilización y disponibilidad de esta solución en el futuro.
- Integrar listas de reputación de software malicioso y/o redes de robots informáticos para detectar posibles ataques y generar alertas.
- Diseñar un cuadro de mandos con la información más relevante que permita disponer de la información más relevante en un único punto.
- Exponer un conjunto de recomendaciones y mejoras que se podrían realizar basado en la experiencia obtenida.

1.3 Enfoque y método seguido

Dada la tipología del trabajo, el mejor enfoque posible es la implementación de una solución que integre las distintas herramientas y posteriormente, a través de una análisis empírico, evaluar los datos obtenidos.

Para estructurar la planificación del trabajo, se ha optado por usar una metodología clásica de gestión de proyectos de tipo cascada que permita gestionar de una forma eficiente, ordenada y sencilla el proyecto dentro de un marco de tiempo muy breve y con fechas estrictas de entrega.

En base a esta premisa, se ha dividido el proyecto en las siguientes seis fases de trabajo:

1. Definición del Plan de Trabajo.
2. Estudio y diseño de la solución.
3. Despliegue y puesta en marcha.
4. Análisis y evaluación de los datos obtenidos.
5. Video presentación.
6. Defensa del Trabajo de Fin de Máster.

En los siguientes apartados se realiza una breve descripción de cada fase.

1.3.1 Definición del Plan de Trabajo

En esta fase se han desarrollado las tareas previas que han permitido definir el Plan de Trabajo que se ha seguido durante el resto del proyecto.

Las tareas que se han realizado durante esta fase son las siguientes:

- Definición del alcance.
- Enumeración de los objetivos.
- Propuesta de la metodología utilizada durante el resto del proyecto.
- Evaluación de los riesgos y descripción de las medidas de mitigación adoptadas.
- Enumeración de las tareas previstas que se han desarrollado durante la ejecución del proyecto.
- Elaboración de la planificación temporal de tareas.
- Evaluación de los recursos necesarios.
- Calculo de los costes incurridos.
- Revisión del estado del arte.

Por ultimo, se ha procedido a la redacción del documento solicitado en la entrega denominada «Entrega 1» cuya puesta a disposición del consultor ha dado por finalizada esta fase.

1.3.2 Estudio y diseño de la solución

En esta fase se ha realizado la revisión de la documentación técnica de los productos que era necesario que formaran parte de la solución y se han realizado diversas pruebas de concepto para evaluar sus capacidades con el objetivo final de completar el diseño de la solución.

Se han evaluado los productos de forma paralela e incremental para ir cubriendo los objetivos según las prioridades establecidas.

Por ultimo, se ha procedido a la redacción del documento solicitado en la entrega denominada «Entrega 2» cuya puesta a disposición del consultor ha dado por finalizada esta fase.

1.3.3 Despliegue y puesta en marcha

En esta fase se ha realizado el despliegue técnico de la solución sobre la plataforma elegida.

A continuación, una vez completados los pasos de despliegue, se ha realizado un conjunto de pruebas controladas orientadas a validar la solución. Una vez se han superado estas pruebas se ha dado por finalizada la puesta en marcha del sistema.

Por último, una vez se ha evaluado la correcta funcionalidad de la solución, se ha iniciado una fase de captura de datos en tiempo real a través del sistema que ha servido como base para la fase posterior de análisis y evaluación.

Por ultimo, se ha procedido a la redacción del documento solicitado en la entrega denominada «Entrega 3» cuya puesta a disposición del consultor ha dado por finalizada esta fase.

1.3.4 Análisis y evaluación de los datos generados

En esta fase se ha realizado una evaluación de los datos generados durante la fase anterior con la que se ha evaluado la viabilidad real de la solución propuesta. Adicionalmente han evaluado las posibles mejoras a la solución planteada que se han podido quedar fuera del alcance y se han considerado recomendables para su explotación fuera del entorno académico dentro del que se ha desarrollado este proyecto.

En esta última fase también se ha completado la redacción de toda la documentación incluida la memoria final y se han preparado los «scripts» de despliegue y configuración según lo que se ha solicitado en la entrega denominada «Entrega 4», con cuya puesta a disposición del consultor se ha dado por finalizada esta fase.

1.3.5 Video presentación

En esta fase se ha preparado el video de presentación junto con la presentación de diapositivas digital para su posterior defensa ante el Tribunal.

La puesta a disposición del consultor de estos entregables según lo que se ha especificado en la entrega denominada «Entrega 5» se ha dado por concluida esta fase.

1.3.6 Defensa del Trabajo de Fin de Máster

En esta fase se ha realizado la defensa virtual de este trabajo frente al Tribunal que ha realizado su evaluación.

Con la finalización del proceso que conforma la entrega denominada «Defensa del TFM», se ha dado también por finalizado el proyecto.

1.4 Análisis de riesgos

1.4.1 Riesgos detectados

Del análisis preliminar del proyecto se han detectado los siguientes posibles riesgos que pueden afectar al desarrollo del proyecto.

1. **Margen de tiempo reducido**. Se ha dispuesto de un margen de tiempo de poco más de tres meses para llevar a cabo la ejecución completa del proyecto.
2. **Entregas parciales con fecha rígida**. Las entregas parciales tenían fechas rígidas para su entrega que no podían ser pospuestas ni negociadas.
3. **Disponibilidad parcial**. No era posible dedicarse en exclusiva al proyecto debido a limitaciones de tiempo impuestas por cuestiones laborales. Esto podría suponer que pudieran existir momentos en los que no fuera posible seguir la planificación prevista para este proyecto.
4. **Novedad tecnológica**. No se disponía de conocimiento previo del software Zeek ni del marco de trabajo ELK Stack, por lo que fue necesario necesario aprender su uso mientras se desarrolla el proyecto.

1.4.2 Matriz de riesgos

La siguiente matriz de riesgos presenta los riesgos relacionándolos con la probabilidad estimada de su ocurrencia en comparación con su impacto relativo en la consecución de los objetivos del proyecto.

Matriz de riesgos					
Probabilidad	Insignificante	Menor	Moderada	Mayor	Catastrófica
Raro	-		-	-	MT
Improbable	-	-	-	-	EP
Posible	-	-	-	NT	-
Probable	-	-	-	-	-
Casi seguro	-	DP	-	-	-

Figura 1. Matriz de riesgos del proyecto

Dentro de la matriz de riesgos se han asignado los códigos indicados en la siguiente tabla.

Riesgo	Código
Margen de tiempo reducido	MT
Entregas parciales con fecha rígida	EP
Disponibilidad parcial	DP
Novedad tecnológica	NT

Tabla 1. Leyenda de la matriz de riesgos

1.4.3 Medidas adoptadas

Del análisis de riesgos anterior se ha decidido adoptar las siguientes medidas con el objetivo de atenuar la materialización de alguno de los riesgos detectados.

1. Establecimiento de una planificación temporal de las distintas tareas con una previsión de carga de trabajo que no supere las dos horas diarias y holgura suficiente para evitar los incumplimientos por falta de tiempo.
2. Limitación del alcance del proyecto a un conjunto de objetivos que pueda asegurarse su cumplimiento.
3. Utilización de herramientas de automatización de infraestructura como Ansible y Docker que permitan reducir los tiempos de despliegue y configuración.

1.5 Planificación del Trabajo

La siguiente tabla presenta el listado de tareas que se ha planificado para la ejecución del proyecto.

La duración representa las jornadas de trabajo. Cada jornada de trabajo se ha definido con una duración de dos horas, por lo que se han estimado **un total de 244 horas de trabajo total** invertidas.

Nombre	Fecha de inicio	Fecha de fin	Duración
Plan de trabajo	**19/2/20**	**2/3/20**	**13**
Definición del alcance	19/2/20	20/2/20	2
Enumeración de objetivos	19/2/20	20/2/20	2
Propuesta metodología	21/2/20	22/2/20	2
Análisis de riesgos	23/2/20	23/2/20	1
Elaboración de la planificación	24/2/20	26/2/20	3
Definición de tareas	24/2/20	26/2/20	3
Calculo del esfuerzo	24/2/20	26/2/20	3
Evaluación de recursos	27/2/20	27/2/20	1
Cálculo de costes	27/2/20	27/2/20	1
Revisión del estado del arte	27/2/20	28/2/20	2
Entrega 1 - Plan de trabajo	20/2/20	2/3/20	12
Adquisición de información	20/2/20	28/2/20	9
Redacción del documento	29/2/20	2/3/20	3
Entrega del documento	3/3/20	3/3/20	0
Estudio y diseño	**4/3/20**	**30/3/20**	**27**
Investigación de productos	4/3/20	10/3/20	7
Estudio Zeek IDS	4/3/20	10/3/20	7
Estudio listas malware	4/3/20	10/3/20	7
Estudio Elastic	4/3/20	10/3/20	7
Diseño de la solución	11/3/20	13/3/20	3
Pruebas de concepto	15/3/20	21/3/20	7
Entrega 2 - Análisis y diseño	14/3/20	30/3/20	17
Adquisición de información	14/3/20	23/3/20	10
Redacción del documento	22/3/20	28/3/20	7
Entrega del documento	31/3/20	31/3/20	0
Despliegue y puesta en marcha	**1/4/20**	**28/4/20**	**28**
Despliegue y configuración software	1/4/20	18/4/20	18
Zeek IDS	1/4/20	5/4/20	5
Instalación software	1/4/20	1/4/20	1
Configuración software	2/4/20	3/4/20	2
Verificación y pruebas	4/4/20	5/4/20	2
Integración listas malware	6/4/20	10/4/20	5
Instalación software	6/4/20	6/4/20	1
Configuración integración	7/4/20	8/4/20	2
Verificación y pruebas	9/4/20	10/4/20	2
ELK Stack	11/4/20	18/4/20	8
Instalación software	11/4/20	12/4/20	2
Configuración software	13/4/20	15/4/20	3
Verificación y pruebas	16/4/20	18/4/20	3
Pruebas de integración	19/4/20	21/4/20	3
Captura y análisis de datos	22/4/20	28/4/20	7
Análisis capacidad Machine Learning	22/4/20	28/4/20	7
Entrega 3 - Despliegue y puesta en marcha	6/4/20	27/4/20	22
Adquisición de información	6/4/20	21/4/20	16
Redacción del documento	22/4/20	27/4/20	6
Entrega del documento	28/4/20	28/4/20	0
Memoria final	**29/4/20**	**1/6/20**	**34**
Análisis final de datos	29/4/20	8/5/20	10
Conclusiones	9/5/20	13/5/20	5
Análisis de posibles mejoras	14/5/20	17/5/20	4
Entrega 4 - Memoria final	8/5/20	1/6/20	25
Adquisición de información	8/5/20	17/5/20	10
Redacción del documento	18/5/20	1/6/20	15
Entrega del documento	2/6/20	2/6/20	0
Video presentación	**3/6/20**	**8/6/20**	**6**
Elaboración del vídeo	3/6/20	8/6/20	6
Elaboración de la presentación	3/6/20	8/6/20	6
Entrega 5 - Presentación en vídeo	9/6/20	9/6/20	0
Defensa del TFM	**15/6/20**	**19/6/20**	**5**

Figura 2: Enumeración del listado de tareas realizado

A continuación se muestra una captura del diagrama de Gantt que contiene toda la planificación del proyecto.

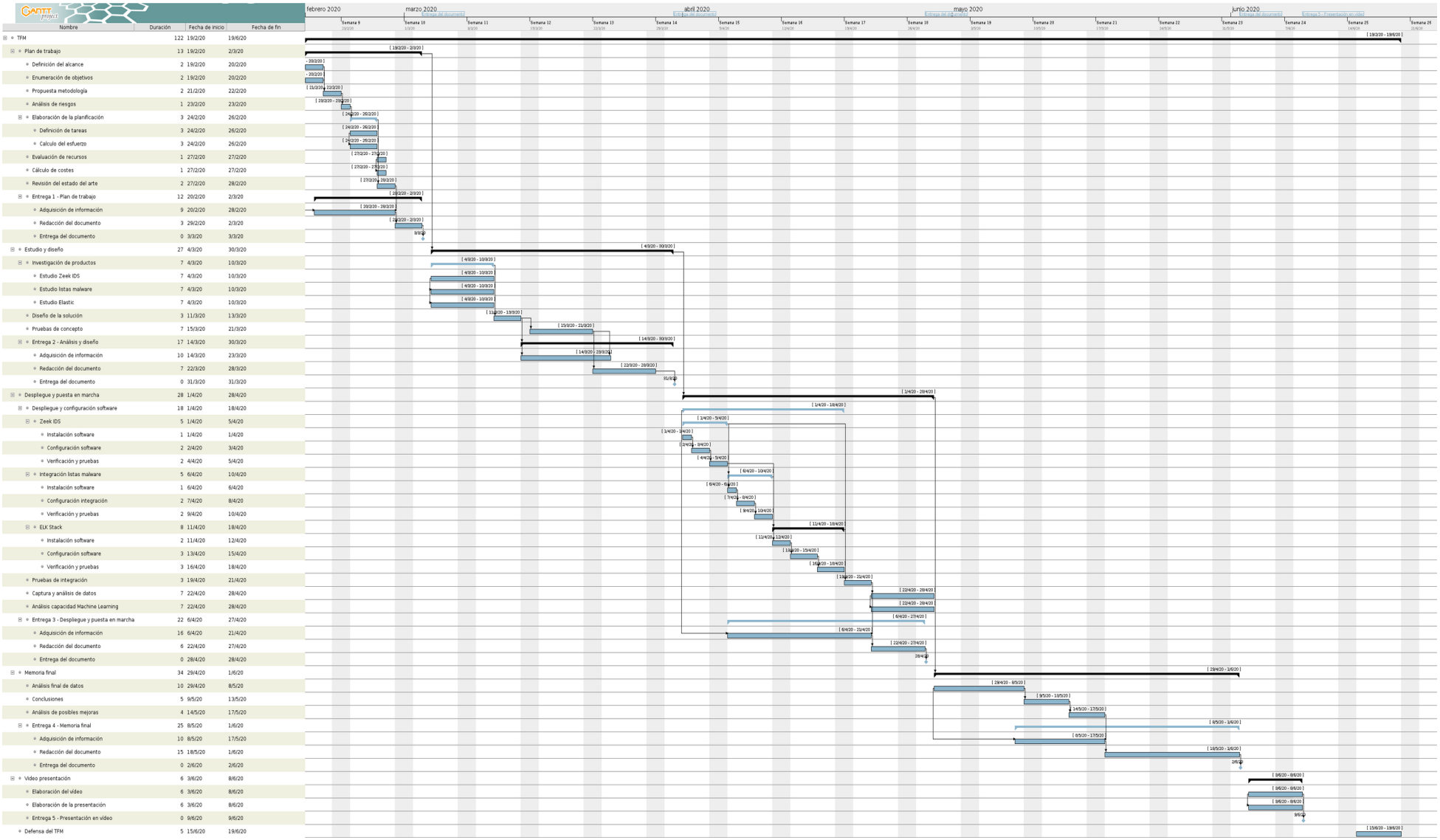


Figura 3. Diagrama de Gantt con la planificación temporal

1.5.1 Recursos materiales

A continuación se especifican los recursos materiales que se han considerado necesarios para el desarrollo del proyecto.

A nivel de equipamiento «hardware» tan solo es necesario un servidor físico con procesador compatible x64 sobre el que se pueda instalar un sistema operativo «Unix-like».

Con el objetivo de minimizar los costes operativos, se ha contratado un servidor dedicado de bajo coste con el proveedor Kimsufi del tipo KS-7.

Las características principales de este modelo de servidor son las siguientes:

Característica	Descripción
Procesador	Intel i3-2130
RAM	8 GB DDR3 1333 MHz
Disco	2 TB

Tabla 2. Características del servidor físico

1.5.2 Costes estimados

El único coste estimado para este proyecto fue el alquiler del servidor por un importe mensual de 18,14 euros impuestos incluidos. Puesto que se ha estimado una duración aproximada de cuatro meses, **el coste total del proyecto fue de 72,56 euros**.

1.6 Breve sumario de productos obtenidos

Como resultado del trabajo realizado se esperaba obtener los siguientes productos:

- Ficheros necesarios para construir la solución mediante Docker y Ansible.
- Memoria de trabajo.
- Presentación de diapositivas en formato digital.
- Video de presentación.

1.7 Estado del arte

Si se mira hacia las soluciones basadas en software libre en relación a los IDS, las principales alternativas la conforman la siguiente terna:

- **Snort**. Snort es un sistema de prevención de intrusiones en redes de código abierto, capaz de realizar análisis de tráfico en tiempo real y registro de paquetes en redes IP. Puede realizar análisis de protocolos, búsqueda/conexión de contenidos y puede ser usado para detectar una variedad de ataques y sondas, como desbordamientos de búfer, escaneos de puertos sigilosos, ataques CGI, sondas SMB, intentos de huellas digitales del SO y mucho más. [1]
- **Zeek IDS**. Zeek proporciona una plataforma integral para el análisis del tráfico de la red, con un enfoque particular en el monitoreo de la

seguridad semántica a escala. Aunque a menudo se compara con los sistemas clásicos de detección/prevención de intrusos, Zeek adopta un enfoque bastante diferente al proporcionar a los usuarios un marco flexible que facilita la supervisión personalizada y en profundidad, mucho más allá de las capacidades de los sistemas tradicionales. Con versiones iniciales ya en despliegue operacional a mediados de los 90, Zeek se encuentra basado en más de 20 años de investigación. [2]

- **Suricata**. Suricata es un motor de alto rendimiento de IDS, IPS y de monitoreo de seguridad de la red. Es de código abierto y propiedad de una fundación sin ánimo de lucro dirigida por la comunidad, la Fundación Abierta de Seguridad de la Información (OISF). Suricata es desarrollado por la OISF. [3]

En referencia al producto objecto de este proyecto, en el estudio «*A Comparative Study of Open Source Network Based Intrusion Detection Systems*» [4] podemos observar la siguiente tabla comparativa entre Snort, Zeek (Bro) y Suricata.

Parameters	Open Source Tools		
	Snort	Bro	Suricata
Developer	Sourcefire, Inc.	National Science Foundation (NSF)	Open Information Security Foundation (OISF)
Multi-thread	No	No	Yes
Operating System Compatibility	Any	Unix like system	Any
Rules Support	VRT Snort rules SO rules Emerging Threats rules	Contextual Signatures	VRT Snort rules Emerging Threats rules
Installation /deployment	Installation also available from packages.	Manual installation	Manual installation.
User community	Large	Small	Small
Documentation	Well documented	Few resources	Few resources
GUI Support	A lot	Few	Few
High Network speed Support	Medium	High	High

Figura 4. Tabla comparativa entre Snort, Zeek y Suricata

En las conclusiones de este documento hace la siguiente referencia explicita a Zeek IDS:

> «*Bro*[1] *podría considerarse un entorno de investigación de alto rendimiento debido a su gran flexibilidad. Snort está orientado a los paquetes mientras que Bro está orientado a la conexión. Su poderosa característica de scripting es definitivamente una mayor*

1 Zeek IDS era conocido anteriormente como «Bro».

ventaja comparada con los conjuntos de reglas en Snort o Suricata». [4]

Respecto los entornos de análisis de datos de tipo web con licenciamiento de fuentes abiertas, cabe mencionar las siguientes soluciones:

- **ELK Stack**. «ELK» es el acrónimo de tres proyectos de código abierto: Elasticsearch, Logstash y Kibana. Elasticsearch es un motor de búsqueda y análisis. Logstash es una tubería de procesamiento de datos del lado del servidor que ingiere datos de múltiples fuentes simultáneamente, los transforma y luego los envía a un «escondite» como Elasticsearch. Kibana permite a los usuarios visualizar los datos con tablas y gráficos en Elasticsearch. [5]
- **Graylog**. Graylog es una solución de gestión centralizada de registros construida con estándares abiertos para capturar, almacenar y permitir el análisis en tiempo real de los terabytes de datos de las máquinas. Construido con el propósito de realizar un moderno análisis de logs, Graylog elimina la complejidad de la exploración de datos, las auditorías de cumplimiento y la caza de amenazas para que puedas encontrar rápida y fácilmente el significado de los datos y tomar medidas con mayor rapidez. [6]

Como contrapunto a las soluciones abiertas, dentro del nicho de los IDS comerciales, y según un artículo de «*9 Top Intrusion Detection and Prevention Systems: Guide to IDPS*» [7], los nueve actores principales en sistemas de detección y prevención de intrusión en el año 2018 eran los siguientes:

- McAfee NSP.
- Trend Micro TippingPoint.
- Hillstone NIPS.
- Darktrace Enterprise Immune System.
- NSFocus NGIPS.
- H3C SecBlade IPS.
- Huawei NIP.
- Entrust IoTrust Identity and Data Security.
- Cisco Firepower NGIPS.

No se han entrado a una realizar una comparativa entre ellos, porque se entiende que todos ellos se corresponden con equipamientos de alta calidad y reconocido prestigio. Su principal hándicap en lo que se refiere a este proyecto es el coste, puesto que se tiene constancia de que es muy alto, rozando en algún caso los cien mil dólares estadounidenses tal y como indica el propio articulo.

Por último, y para cerrar este apartado, se hace referencia al estudio «*Commercial and open-source based Intrusion Detection System and Intrusion Prevention System (IDS/IPS) design for an IP networks*» donde se compara Snort con Cisco ASA y donde en el apartado donde expone los resultados de su comparativa indica lo siguiente:

«*En primer lugar nos gustaría mencionar que no es muy inteligente decir que el sistema comercial o de código abierto es mejor. Depende en gran medida de las redes específicas, su funcionalidad y los requisitos económicos. Eso significa que ambas soluciones tienen ventajas y desventajas comunes y diferentes. La funcionalidad de ambos sistemas es casi la misma. Las diferencias se pueden encontrar en la configuración de cada dispositivo. […] En resumen, los dispositivos comerciales lideran en cuanto a funcionalidad y fiabilidad, por lo que el cliente tiene que pagar. De ello se deduce naturalmente que la configuración es más fácil. Por el contrario, las soluciones de código abierto son más baratas, pero la configuración y la gestión son más complicadas.*» [8]

1.8 Breve descripción de los otros capítulos de la memoria

El **capítulo 2**, «Estudio y diseño de la solución», contiene el análisis de las distintas alternativas que se han evaluado para el diseño de la solución, junto con una descripción de cada uno de los productos seleccionados y su finalidad dentro de la solución planteada. Adicionalmente se aporta un esquema lógico de la solución planteada que permite comprender los distintos elementos y su interacción. Por último, se describe el plan de pruebas que permite verificar la correcta integración entre los distintos elementos y, que en última instancia, valida toda la solución.

El **capítulo 3,** «Despliegue y puesta en marcha», contiene la descripción de todos los componentes desplegados y que fueron planteados durante el diseño de la solución. Adicionalmente se justifican las diferentes opciones técnicas que se han incluido con el objetivo de facilitar la compresión y contextualización de todo el entorno. A continuación, se detallan las pruebas realizadas para validar el despliegue realizado y que confirma que la solución es operativa y está lista para la captura inicial de datos. Por último, contiene información sobre las primeras capturas de datos y se realiza una valoración preliminar de los resultados obtenidos.

El **capítulo 4,** «Análisis y evaluación de los datos generados», contiene la descripción de los distintos cuadros de mandos generados a través del análisis de los resultados obtenidos y las conclusiones a las que se ha llegado a través de su análisis.

El **capítulo 5**, «Conclusiones», contiene las conclusiones finales, una revisión de la consecución de los objetivos planteados junto con un análisis del seguimiento de la planificación, y por último, posibles líneas de trabajo futuro que se podrían desarrollar para completar algunos aspectos de este trabajo.

El **capítulo 6**, «Glosario», contiene un breve glosario de algunos términos que se han considerado relevantes.

El **capítulo 7**, «Bibliografía», contiene las referencias a la diversa bibliografía utilizada durante el desarrollo de este trabajo.

Por último, el **capítulo 8,** «Anexos», contiene el código fuente de los distintos «scripts» desarrollados, «playbooks» de Ansible y ficheros de configuración de las distintas aplicaciones utilizadas en el desarrollo de la solución.

2 Estudio y diseño de la solución

Dentro del alcance del proyecto, se han definido como requisitos básicos imprescindibles el uso de los siguientes componentes:

- Zeek.
- Elastic Stack.

El resto de componentes seleccionados se ha realizado en base al cumplimiento de los objetivos planteados en el apartado 1.2 Objetivos del Trabajo.

Durante esta fase se han realizado un estudio teórico de información sobre los distintos componentes utilizando como base la documentación de los distintos fabricantes así como literatura especializada.

Para confirmar los hallazgos se han realizado pequeñas pruebas de concepto sobre maquinas virtuales que han permitido añadir una confirmación empírica al trabajo de estudio.

2.1 Arquitectura hardware

Se ha dispuesto de un único servidor descrito en el apartado 1.5.1 Recursos materiales. Este servidor disponía de una única interfaz de red conectada a Internet mediante la IP 176.31.103.131.

El segmento de red donde se encuentra este servidor es compartido con otros servidores que pertenecían a otros clientes, por lo que, con el objetivo de preservar la confidencialidad de su información, no se ha considerado como una opción valida la configuración de la interfaz de red en modo promiscuo para capturar todo el tráfico que llegase a ese segmento de red.

2.2 Software base

Dado que la arquitectura actual ha estado limitada exclusivamente a un servidor y que la solución propuesta dispone de varios componentes, se ha optado por la utilización un sistema operativo y una solución de virtualización de aplicaciones mediante contenedores basados en software libre.

En lo relacionado con el sistema operativo, se han barajado las siguientes dos opciones:

- **FreeBSD**. Se trata de un sistema operativo derivado de BSD que es compatible con arquitecturas Intel x86. Destaca especialmente por su seguridad y rendimiento. [9]
- **CentOS**. Se trata de una distribución GNU/Linux creada por la comunidad con la intención de ofrecer un producto derivado de las fuentes de Red Hat Enterprise Linux. Destaca especialmente por su estabilidad. [10]

Tras realizar una serie de pruebas de instalación, se ha optado por CentOS en su versión 7.7 por la disponibilidad de binarios de los distintos componentes, en especial Elastic Stack, para esta distribución GNU/Linux frente a FreeBSD.

Una vez seleccionado el sistema operativo del servidor, se ha decidido optar directamente por Docker como solución de contenedores frente a otras como

OpenVZ o LXC dado que se trata de un referente en el mercado actual, ampliamente soportado y utilizado por millones de desarrolladores a lo largo del mundo. [11]

2.3 Honeypot Heralding

Se trata de un «honeypot» muy simple que sirve para recolectar credenciales. Actualmente soporta los siguientes protocolos: ftp, telnet, ssh, rdp, http, https, pop3, pop3s, imap, imaps, smtp, vnc, postgresql and socks5. [12]

La función que desempeña el «honeypot» es la de simular la existencia de diversos servicios vulnerables con puertos abiertos hacia la red exterior, en este caso, Internet.

No se ha utilizado para registrar la actividad de los atacantes, de hecho esa funcionalidad se ha desactivado, sino que se ha pretendido potenciar la ocurrencia de intentos de rastreo y acceso por parte de los posibles atacantes de cara a probar las capacidades de Zeek.

2.4 Zeek IDS

Zeek fue desarrollado por Vern Paxson que es profesor de informática en la Universidad de California en Berkeley y forma parte del Instituto Internacional de Ciencias de la Computación ubicado en la misma ciudad. [13]

Se puede definir como un analizador de tráfico de red pasivo y de código abierto. Es principalmente un monitor de seguridad que inspecciona todo el tráfico de un enlace en profundidad para buscar señales de actividad sospechosa. Sin embargo, de manera más general, Zeek soporta una amplia gama de tareas de análisis de tráfico incluso fuera del dominio de la seguridad, incluyendo mediciones de rendimiento y ayuda en la resolución de problemas. [14]

Por lo tanto, se puede decir que Zeek es más que un IDS, es un analizador de red. Puede localizar usos anormales de los protocolos de red, como el uso de HTTP sobre un puerto poco común. Esto es posible por que Zeek reconoce el tráfico HTTP por características como las peticiones GET y POST, en vez de basarse en los puertos usados durante la sesión. Cuando Zeek ve estos puertos inesperados, puede registrar su actividad. Esta capacidad de analizar los protocolos de la capa de aplicación lo hace perfecto para obtener el conocimiento real de lo que está pasando en la red diariamente y permite crear una línea base a lo largo del tiempo. [15]

Durante la prueba de concepto se ha detectado que la versión binaria del software no dispone del soporte para geolocalización por lo que se ha procedido a su compilación desde las fuentes.

2.5 Elastic Stack

“ELK” es la sigla para tres proyectos «open source»: Elasticsearch, Logstash y Kibana. Elasticsearch es un motor de búsqueda y analítica. Logstash es un pipeline de procesamiento de datos del lado del servidor que ingesta datos de una multitud de fuentes simultáneamente, los transforma y luego los envía a un "escondite", como Elasticsearch. Kibana permite a los usuarios visualizar los datos en cuadros y gráficos con Elasticsearch. [16]

Durante la prueba de concepto se descartó la versión con licencia Basic que incluye el módulo de aprendizaje de máquina como el de SIEM porque el primero requiere del pago de licenciamiento avanzado y el segundo se queda en funcionalidades básicas sin el primero.

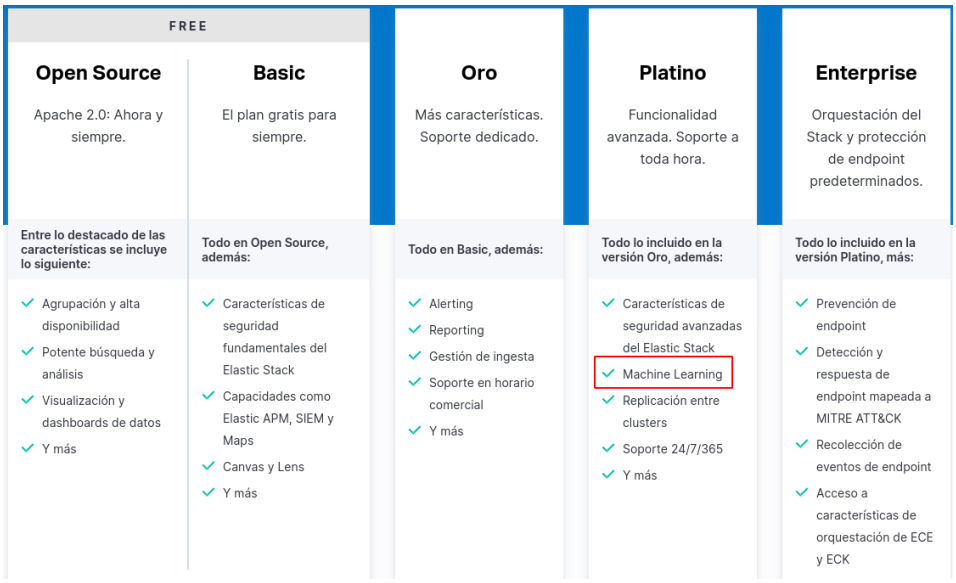


Figura 5. Suscripciones de Elastic Stack

La imagen de la figura anterior tiene la referencia bibliográfica [17].

2.5.1 Filebeat

Aunque la familia Beats está compuesta por varios programas, en lo que se refiere a este proyecto, tan solo tiene relevancia el producto Filebeat, que es el único usado entre todos ellos.

Filebeat forma parte del conjunto de herramientas Beat y su único propósito es reenviar el contenido de los ficheros de registros desde los servidores a un servidor Logstash o Elasticsearch centralizado.[18]

En lo relacionado con este proyecto se ha utilizado para reenviar los registros desde el contenedor de Zeek al contenedor de Logstash para su posterior tratamiento.

Dado que se ha decidido utilizar la versión de fuentes abiertas de Elastic Stack, no se dispone del módulo de procesamiento especifico para los registros de Zeek. Esto supone que se han tenido que tratar estos registros en el componente de Logstash para proporcionales el formato adecuado antes de remitirlos a la base de datos Elasticsearch.

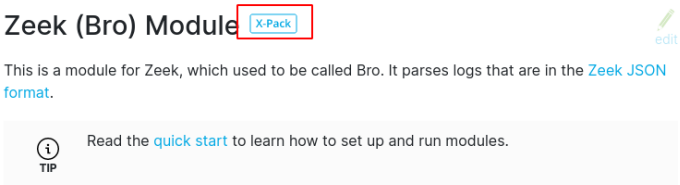


Figura 6: Módulo para Zeek de X-Pack

2.5.2 Logstash

Logstash ayuda a centralizar los datos de eventos como registros, métricas, o cualquier otro dato en cualquier formato. Se trata de un componente clave de Elastic Stack, cuyo rol es centralizar de forma escalable la recolección de datos desde un número amplio de fuentes, transformarlos y enviar estos datos diferentes formatos de salida. [18]

Se pueden configurar diferentes fuentes de entrada y salida para leer o escribir de ellas, al mismo tiempo que se aplican diferentes complementos que permiten transformar los datos de entrada; por ejemplo, se puede modificar un campo, transformar su tipo o añadir datos de geolocalización utilizando como base una dirección IP. [19]

En lo relacionado con este proyecto se ha utilizado para transformar los datos recibidos de los registros del contenedor de Zeek a través de Filebeat y agregarles la información de geolocalización y de las fuentes de información sobre amenazas. Posteriormente se han insertado los datos en la base de datos Elasticsearch.

2.5.3 Elasticsearch

Elasticsearch proporciona búsqueda y análisis en tiempo real para todo tipo de datos. Tanto si tiene texto estructurado como no estructurado, datos numéricos o datos geoespaciales, Elasticsearch puede almacenarlos e indexarlos de forma eficiente de manera que permita realizar búsquedas rápidas. Puede ir mucho más allá de la simple recuperación de datos y de la información agregada para descubrir tendencias y patrones en sus datos. Y a medida que el volumen de datos y consultas crece, la naturaleza distribuida de Elasticsearch permite que su despliegue crezca sin problemas junto con él. [20]

Elasticsearch proporciona una sencilla y coherente API REST para gestionar su clúster e indexar y buscar sus datos.

Aunque se puede utilizar Elasticsearch como un almacén de documentos y recuperar los documentos y sus metadatos, el verdadero poder proviene de poder acceder fácilmente al conjunto completo de capacidades de búsqueda construidas en la biblioteca del motor de búsqueda Apache Lucene.[21]

En lo relacionado con este proyecto, se ha utilizado para almacenar la información procesada recibida desde el contenedor de Logstash y adicionalmente para servir de fuente de datos a la aplicación web Kibana.

2.5.4 Kibana

En Elastic Stack, Kibana se ha usado principalmente para proveer la interfaz gráfica de usuario. Kibana posee muchas características diferentes, aunque las principales son las siguientes:

- Descubrimiento de información a través de su exploración.
- Análisis de datos a través de la aplicación de diferentes métricas.
- Visualización de datos creando diferentes tipos de gráficas para mostrar visualmente la información.
- Aplicación de aprendizaje automático sobre los datos para obtener anomalías en los datos y tendencias futuras.

- Monitorización de aplicación mediante APM
- Una consola para ejecutar expresiones de Elasticsearch.
- Monitorización del estado de Elastic Stack. [18]

En lo relacionado con este proyecto, este producto se ha utilizado como la herramienta grafica de visualización de los datos recopilados. En esta instancia de Kibana se han diseñado los siguientes componentes:

- Cuadros de mandos.
- Gráficas de contadores individuales.

2.5.5 Casos de uso de Elastic Stack

Existen muchas áreas donde es posible utilizar Elastic Stack, como, por ejemplo, las siguientes:

- **Monitorización del rendimiento de sistemas**. Permite integrar los contadores de rendimiento de CPU, memoria, disco, etc. para monitorizarlos de forma integral y emitir alertas ante determinados comportamientos.
- **Gestión de registros**. Se trata de uno de los usos principales de Elastic Stack desde su creación.
- **Monitorización del rendimiento de aplicaciones**. Ayuda a los desarrolladores y administradores de sistemas a monitorizar el rendimiento y disponibilidad de las aplicaciones monitorizando el código e integrándolo con las estadísticas de servidor, de red, etc.
- **Visualización de datos**. Se trata de una de las características principales de Kibana, que permite mediante la mezcla de gráficas, series de tiempo y mapas realizar complejas relaciones entre la información. [18]

2.6 Fuentes de información sobre amenazas

Las fuentes de información sobre amenazas son flujos de datos en tiempo real que proporcionan información sobre posibles amenazas y riesgos cibernéticos.

Por lo general, estas fuentes suelen ofrecer información sobre las siguientes categorías:

- Direcciones IP asociadas a actividades maliciosas.
- Dominios sospechosos.
- Listas de «hashes» de «malware» conocidos.

Dado que la solución se basa en la escucha pasiva del tráfico entrante desde Internet, se han elegido fuentes de información de origen abierto que contengan información sobre listas negras de IP.

Para garantizar un mínimo de calidad en la información, se realizó una búsqueda a través de Internet para localizar las listas de este tipo que mayor cantidad de referencias y mejores valoraciones tenían. Posteriormente, entre estas, se han seleccionado las que garantizaban actualizaciones regulares.

Con estas listas se han etiquetado las conexiones en función de las IP de origen para la generación de estadísticas basadas en los distintos tipos de actividades malintencionadas que conecten contra el «honeypot».

En concreto, se han elegido las listas de los siguientes proveedores:

- **Blocklist.de**. Se trata de «un servicio gratuito y voluntario prestado por un especialista en fraude/abuso, cuyos servidores suelen ser atacados por medio de SSH, Mail-Login, FTP, Webserver y otros servicios. [22]
- **BadIPs.com**. Se trata de un servicio de tipo comunitario donde los propios usuarios se encargan de la recopilación de IP relacionadas con actividades maliciosas.
- **Turris.cz**. Se trata de un proyecto dirigido por Cz.nic que se encarga de recopilar, procesar y etiquetar el tráfico que accede a sus enrutadores Turris.[23]
- **Dan.Me.uk**. Se trata de un proyecto personal que contiene múltiples herramientas. Entre ellas existe una lista de IP de nodos de salida de la red Tor.
- **Malware Bazaar**. Se trata de un proyecto personal ubicado en abuse.ch que contiene múltiples listas de actividades maliciosas. En este caso se ha utilizado la lista que contiene los «hashes» de diferentes ficheros que se ha comprobado que son software malicioso.

En lo relacionado con este proyecto, se han elaborado un conjunto de «scripts» en lenguaje Python para la descarga de estas fuentes de información desde su origen y su procesado para convertirlas en ficheros CSV que se han utilizado posteriormente desde Logstash.

2.7 Esquema gráfico de la solución

En la siguiente figura se representa el esquema lógico que muestra la relación entre los distintos componentes de la solución.

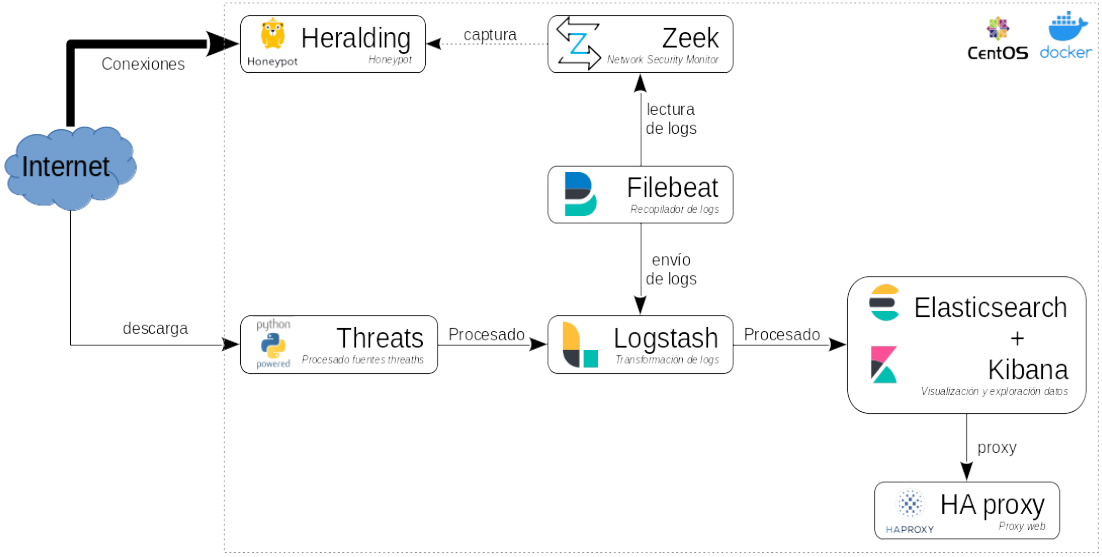


Figura 7: Esquema lógico de la solución propuesta

La imagen de la figura anterior tiene las referencias bibliográficas de la [24] a la [30].

2.8 Plan de pruebas

Con el objetivo de validar de forma gradual toda la plataforma se ha diseñado un plan de pruebas parciales que se ha aplicado durante la fase del despliegue de la solución. Este plan de prueba se ha dividido en los siguientes puntos de control.

- **Funcionamiento del «honeypot»**. Se han lanzado pruebas de conexiones al «honeypot» mediante escaneos de red con Nmap para confirmar el funcionamiento del software.
- **Captura de tráfico desde Zeek**. Se han realizado conexiones manuales desde un equipo externo hacia el «honeypot» para verificar que Zeek captura y procesa correctamente el tráfico de red.
- **Reenvío de logs desde Filebeat**. Se ha reenviado un conjunto controlado de logs de Zeek para confirmar que Filebeat es capaz de reenviarlos sin errores ni perdida de información en situaciones normales y especificas como la parada y arranque de los contenedores.
- **Procesado de fuentes de información sobre amenazas**. Se ha procesado con los scripts un subconjunto controlado de datos para confirmar que se procesan correctamente para construir las base de datos de fuentes de información sobre amenazas.
- **Procesado de los logs en Logstash**. Se ha usado un subconjunto controlado de los logs recibidos desde Filebeat que se han procesado desde Logstash para validar la transformación de datos y la integración de las fuentes de información sobre amenazas.
- **Almacenamiento de datos en Elasticsearch**. Se han remitido desde Logstash un pequeño subconjunto de datos para validar el almacenamiento de la información en la base de datos.
- **Visualización de información desde Kibana**. Se han realizado pruebas de visualización de los datos importados durante la prueba sobre Elasticsearch para confirmar el correcto funcionamiento de Kibana.

3 Despliegue y puesta en marcha

En este capitulo se describe el proceso de despliegue y puesta en marcha de la solución diseñada anteriormente. Se debe tener en cuenta que cuando se habla de «describir el proceso», a lo que hace referencia es a una descripción a alto nivel de los distintos componentes utilizados y los motivos que han llevado a utilizar esos medios. El detalle especifico de cada fichero debe ser interpretado por el propio lector a través de los ficheros que se adjuntan a este documento como parte del material entregado al finalizar el proyecto.

Como ya se ha comentado en capítulos anteriores, el concepto básico sobre el que se ha trabajado a la hora de diseñar la solución es que se pudiera adaptar a diversos entornos. Por este motivo se ha optado por utilizar contenedores

Docker puesto que mediante este mecanismo se independiza la «parte software» de la solución del entorno donde se tiene que ejecutar.

Adicionalmente al uso de contenedores, otro objetivo que se ha planteado es la posibilidad de que se pudiera desplegar de forma rápida y reproducible, por este motivo se ha utilizado Ansible como solución para la automatización y despliegue en el servidor del que se ha dispuesto para este proyecto.

3.1 Contenedores Docker

Como se ha comentado anteriormente, cada componente de la solución se ha desplegado en un contenedor Docker diferente con el propósito de modularizar el despliegue.

Los distintos componentes han dispuesto de una red interna por la que se han comunicado entre ellos o, cuando no es necesaria la comunicación mediante una red, se ha dispuesto que compartan volúmenes Docker para compartir aquellos ficheros que le sean necesarios. De esta forma se asegura que no era necesario tomar medidas especificas de seguridad entre los distintos componentes, facilitando de esta forma su integración y minimizando los recursos necesarios, y, por otro lado, que la seguridad se situase en el servidor que actuaba de anfitrión.

A continuación se describe cada uno de los contenedores desplegados basándose en la figura Error: no se encontró el origen de la referencia que muestra de forma gráfica la solución planteada.

3.1.1 Contenedor Heralding

Este contenedor ejecutaba una instancia del «honeypot» Heralding que simulaba ser una máquina con diversos servicios vulnerables ejecutándose y abiertos hacia la red exterior, en este caso, Internet.

La idea de la que se parte es que la mayoría de escáneres de servicios vulnerables son «bots» que rastrean los servidores en busca de vulnerabilidades para posteriormente utilizar la información recopilada para focalizar los ataques o compartir esa información entre diferentes redes de atacantes. La utilización de un «honeypot» pretendía «motivar» a que se produjeran esos rastreos y/o ataques sin poner en riesgo un «servidor real».

3.1.1.1 Fichero Dockerfile

A grandes rasgos en el fichero Dockerfile se realizan las siguientes tareas:

1. Actualizar la imagen base del sistema operativo.
2. Instalar los paquetes de Python que son requisitos previos para la posterior ejecución de Heralding.
3. Generar el usuario y el grupo «heralding» que ejecutará el software de Heralding.
4. Asignar permisos al ejecutable de Python para poder vincularse a «puertos privilegiados».
5. Copiar los ficheros de instalación Heralding a una ruta temporal y el fichero de configuración personalizado a su ruta definitiva.

6. Ajustar los permisos del fichero del punto de entrada e instalar Heralding desde los ficheros copiados anteriormente.
7. Asignar el directorio de trabajo y el punto de entrada.

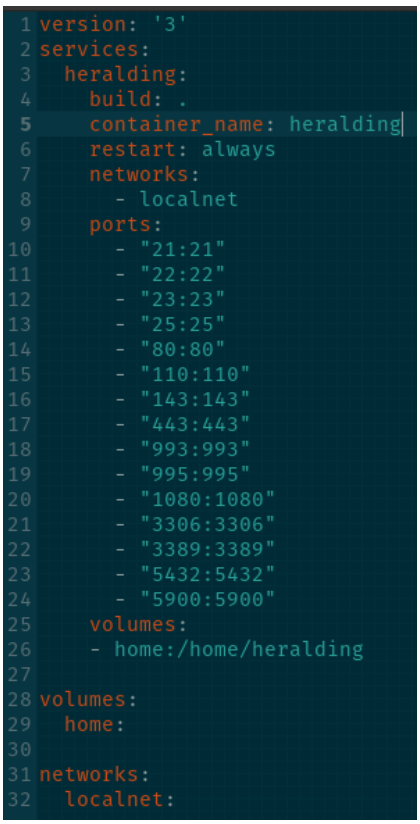
```
FROM centos:7

LABEL maintainer "igalvit@uoc.edu"

RUN yum update -y \
    && yum -y install python3 python3-pip python3-devel \
       gcc gcc-c++ make openssl-devel libffi-devel postgresql-devel \
       git \
    && yum clean all \
    && rm -rf /var/cache/yum \
    && groupadd -g 1500 heralding \
    && useradd -u 1500 -g 1500 heralding \
    && setcap cap_net_bind_service=+ep /usr/bin/python3.6

USER heralding:heralding

COPY --chown=heralding:heralding ./config /home/heralding

COPY --chown=heralding:heralding ./heralding_src /tmp/heralding

RUN chmod +x /home/heralding/entrypoint.sh \
    && mkdir /home/heralding/logs \
    && cd /tmp/heralding \
    && pip3 install --user --no-cache-dir -r requirements.txt \
    && pip3 install --user --no-cache-dir . \
    && rm -rf /tmp/heralding

WORKDIR /home/heralding

ENV PATH $PATH:/home/heralding/.local/bin

ENTRYPOINT ["/home/heralding/entrypoint.sh"]
```

Figura 8: Heralding. Contenido Dockerfile

3.1.1.2 Fichero docker-compose.yml

En el fichero «docker-compose.yml» se realizan las siguientes tareas:

1. Construir el contenedor con la información del fichero Dockerfile.
2. Asignar la red «localnet».
3. Exponer a la red externa el conjunto de puertos detallados en el apartado siguiente.
4. Configura un solo volumen llamado «home» que se utilizará para contener el software Heralding.



```
version: '3'
services:
  heralding:
    build: .
    container_name: heralding
    restart: always
    networks:
      - localnet
    ports:
      - "21:21"
      - "22:22"
      - "23:23"
      - "25:25"
      - "80:80"
      - "110:110"
      - "143:143"
      - "443:443"
      - "993:993"
      - "995:995"
      - "1080:1080"
      - "3306:3306"
      - "3389:3389"
      - "5432:5432"
      - "5900:5900"
    volumes:
    - home:/home/heralding

volumes:
  home:

networks:
  localnet:
```

Figura 9: Heralding. Contenido docker-compose.yml

3.1.1.3 Configuración de Heralding

La instancia de Heralding se ha configurado para simular la existencia de los servicios identificados en la siguiente tabla.

Servicio	Puerto	Banner
FTP	TCP/21	Microsoft FTP Server
SSH	TCP/22	SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.8
Telnet	TCP/23	---
SMTP	TCP/25	Microsoft ESMTP MAIL service ready
HTTP	TCP/80	---
POP3	TCP/110	---
IMAP	TCP/143	* OK IMAP4rev1 Server Ready
HTTPS	TCP/443	---
IMAPS	TCP/993	* OK IMAP4rev1 Server Ready
POP3S	TCP/995	---
Socks5	TCP/1080	---
MySQL	TCP/3306	---
RDP	TCP/3389	---
PostgreSQL	TCP/5432	---
VNC	TCP/5900	---

Tabla 3: Puertos y «banners» falsos servicios Heralding

3.1.2 Contenedor Zeek

Este contenedor ejecuta una instancia de Zeek que se ha encargado de capturar todo el tráfico de la tarjeta de red del servidor anfitrión en busca de los comportamientos anómalos que se produjeran, principalmente, por la interacción de los atacantes con el contenedor Heralding.

Se ha configurado la red del contenedor de esta para mejorar el rendimiento y evitar un nuevo NAT que podría ofuscar las conexiones reales desde el exterior. Se debe tener en cuenta que esta configuración solo es valida si el servidor anfitrión tiene un sistema operativo GNU/Linux pero se ha tomado porque supone una mejora importante en el rendimiento.[31]

3.1.2.1 Fichero Dockerfile

A grandes rasgos en el fichero Dockerfile se realizan las siguientes tareas:

1. Actualizar la imagen base del sistema operativo.
2. Descarga el fichero del repositorio de paquetería de Zeek para posteriormente instalar el paquete.
3. Copiar el fichero de configuración del nodo de Zeek y el fichero del punto de entrada.
4. Ajustar los permisos del fichero de entrada.
5. Asignar el directorio de trabajo y el punto de entrada.

```
FROM centos:7

LABEL maintainer "igalvit@uoc.edu"

RUN yum update -y \
    && yum -y install wget \
    && wget https://download.opensuse.org/repositories/security:zeek/CentOS_7/-
security:zeek.repo -O /etc/yum.repos.d/zeek.repo \
    && yum update -y \
    && rpm --import http://download.opensuse.org/repositories/security:/zeek/-
CentOS_7/repodata/repomd.xml.key \
    && yum update -y \
    && yum -y install zeek-lts \
    && yum clean all && rm -rf /var/cache/yum

ENV PATH=/opt/zeek/bin/:$PATH

COPY ./config/entrypoint.sh /opt/zeek/entrypoint.sh

COPY ./config/local.zeek /opt/zeek/share/zeek/site/local.zeek

RUN chmod +x /opt/zeek/entrypoint.sh

WORKDIR /opt/zeek/logs

ENTRYPOINT ["/opt/zeek/entrypoint.sh"]
```

Figura 10: Zeek. Contenido Dockerfile

3.1.2.2 Fichero docker-compose.yml

En el fichero «docker-compose.yml» se realizan las siguientes tareas:

1. Construir el contenedor con la información del fichero Dockerfile.
2. Asignar la red «host» para que pueda capturar el tráfico de red.
3. Configurar tres volúmenes Docker:
    1. Volumen «etc». Volumen interno que contiene la configuración general de Zeek.

2. Volumen «logs». Volumen interno que contiene los ficheros de registro generados por Zeek.
3. Volumen «site». Volumen interno que contiene la configuración local del nodo de Zeek.

```
version: '3'
services:
  zeek:
    build: .
    container_name: zeek
    restart: always
    network_mode: host
    volumes:
    - etc:/opt/zeek/etc
    - logs:/opt/zeek/spool/zeek
    - site:/opt/zeek/share/zeek/site

volumes:
  etc:
  logs:
  site:
```

Figura 11: Zeek. Contenido docker-compose.yml

3.1.2.3 Configuración de Zeek

Apenas se ha modificado la configuración por defecto con la que viene configurado el software. Las modificaciones se han limitado a que emita sus ficheros de registro en formato JSON. Este cambio facilita su posterior procesado por parte del contenedor Filebeat.

```
@load policy/tuning/json-logs.zeek

redef LogAscii::use_json = T;
```

Figura 12: Zeek. Fichero configuración

3.1.3 Contenedor Filebeat

Este contenedor ejecuta una instancia de Filebeat que se ha encargado de leer los registros desde el volumen Docker que ha compartido con el contenedor Zeek para posteriormente remitirlos al contenedor Logstash al puerto TCP/5044.

Este contenedor no era realmente necesario dado que Logstash es capaz de leer los ficheros que produce Zeek, pero se ha tomado la decisión de desplegarlo porque en un entorno más grande es posible que sea necesario disponer de un clúster de nodos Zeek o de varios servidores Zeek ubicados, por ejemplo, en diferentes localizaciones y en ese caso se podrían desplegar varios contenedores con Filebeat que se encarguen de reenviar los registros de esos servidores.

3.1.3.1 Fichero Dockerfile

A grandes rasgos en el fichero Dockerfile se realizan las siguientes tareas:

1. Actualizar la imagen base del sistema operativo.
2. Descargar desde Internet la versión 7.6.1 OSS de Filebeat y descomprimirla en su ruta.
3. Copiar los ficheros de configuración de Filebeat.

4. Ajustar los permisos del fichero de entrada y borrando el fichero descargado para instalar Filebeat.
5. Asignar el usuario de ejecución, el directorio de trabajo y el punto de entrada.

```
FROM centos:7

LABEL maintainer "igalvit@uoc.edu"

RUN yum update -y \
    && yum -y install wget \
    && yum clean all \
    && cd /tmp \
    && wget https://artifacts.elastic.co/downloads/beats/f
    && tar xzvf filebeat-oss-7.6.1-linux-x86_64.tar.gz \
    && mv filebeat-7.6.1-linux-x86_64 /opt/filebeat \
    && mkdir /opt/filebeat/logs

ENV PATH $PATH:/opt/filebeat

COPY ./config/filebeat.yml /opt/filebeat/filebeat.yml

COPY ./entrypoint.sh /opt/entrypoint.sh

RUN chmod +x /opt/entrypoint.sh \
    && rm -f /tmp/filebeat*

WORKDIR /opt/filebeat

ENTRYPOINT ["/opt/entrypoint.sh"]
```

Figura 13: Filebeat. Contenido Dockerfile

3.1.3.2 Fichero docker-compose.yml

En el fichero «docker-compose.yml» se realizan las siguientes tareas:

1. Construir el contenedor con la información del fichero Dockerfile.
2. Asignar la red externa «elasticsearch_localnet».
3. Configura un solo volumen externo llamado «zeek_logs» en modo solo lectura que contiene los ficheros de registro del contenedor Zeek.

```
version: '3'
services:
  filebeat:
    build: .
    container_name: filebeat
    restart: always
    networks:
      - elasticsearch_localnet
    volumes:
    - zeek_logs:/opt/zeek/spool/zeek:ro

volumes:
  zeek_logs:
    external: true

networks:
  elasticsearch_localnet:
    external: true
```

Figura 14: Filebeat. Contenido docker-compose.yml

3.1.3.3 Configuración de Filebeat

Filebeat se ha configurado con las siguientes opciones:

- Procesa todos los ficheros con extensión «.log» de la ruta «/opt/zeek/spool/zeek/» excepto aquellos que no aportan información sobre el tráfico de red generado.

- La salida se remite al contenedor Logstash por el puerto 5044.
- Se guardan registros propios de Filebeat del procesamiento que realiza para depuración.

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /opt/zeek/spool/zeek/*.log
  json.keys_under_root: true
  json.add_error_key: true
  json.message_key: uid
  exclude_lines: ['^#']
  exclude_files:
    - broker.log
    - capture_loss.log
    - cluster.log
    - config.log
    - loaded_scripts.log
    - notice.log
    - packet_filter.log
    - print.log
    - prof.log
    - reporter.log
    - stats.log
    - stderr.log
    - stdout.log

filebeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false

#---------------------------- Logstash
output.logstash:
  # The Logstash hosts
  hosts: ["logstash:5044"]

#------------------------------- Logg
logging.level: info
logging.to_files: true
logging.files:
  path: ${path.config}/logs/filebeat
  name: filebeat
  keepfiles: 7
  permissions: 0644
```

Figura 15: Filebeat. Fichero de configuración

3.1.4 Contenedor Threats

Este contenedor se ejecuta los «scripts» escritos en Python que se han encargado de descargar y procesar las fuentes de inteligencia de amenazas descritas anteriormente.

Se ha programado con el «cron» del contenedor la ejecución de los distintos «scripts» con una periodicidad de seis horas y separando su inicio entre cinco y diez minutos para evitar solapamientos y sobreutilización de recursos por parte del contenedor.

```
scripter@9a69e2a99acd:~
[root@tfm ~]# docker exec -it threats /bin/bash
[root@9a69e2a99acd scripter]# su - scripter
Last login: Mon Apr 13 16:23:24 UTC 2020 on pts/0
[scripter@9a69e2a99acd ~]$ crontab -l
0 0-23/6 * * * python turris.py
5 0-23/6 * * * python blocklistde.py
10 0-23/6 * * * python badips.py
20 0-23/6 * * * python danmeuk.py
25 0-23/6 * * * python malwarebazaar.py
[scripter@9a69e2a99acd ~]$
```

Figura 16: Threats. Contenido crontab

3.1.4.1 Fichero Dockerfile

A grandes rasgos en el fichero Dockerfile se realizan las siguientes tareas:

1. Actualizar la imagen base del sistema operativo.

2. Instalar las dependencias de Python que son necesarias para la ejecución de los «scripts».
3. Generar el usuario y el grupo «scripter» que ejecutará los «scripts».
4. Copiar los ficheros de los «scripts».
5. Añadir al cron del usuario «scripter» la configuración con la planificación de ejecución de los scripts.
6. Realizar una primera ejecución de los «scripts» que permita generar los primeros resultados.
7. Asignar el usuario de ejecución y el punto de entrada.

```
FROM centos:7

LABEL maintainer "igalvit@uoc.edu"

RUN yum install -y epel-release \
    && yum update -y \
    && yum -y install python2-pip python-pandas crontabs python-requests \
    && yum clean all \
    && rm -rf /var/cache/yum \
    && groupadd -g 1500 scripter \
    && useradd -u 1500 -g 1500 scripter

USER scripter:scripter

COPY --chown=scripter:scripter ./scripts /home/scripter

RUN (crontab -l ; echo "0 0-23/6 * * * python turris.py") | crontab - \
    && (crontab -l ; echo "5 0-23/6 * * * python blocklistde.py") | crontab - \
    && (crontab -l ; echo "10 0-23/6 * * * python badips.py") | crontab - \
    && (crontab -l ; echo "20 0-23/6 * * * python danmeuk.py") | crontab - \
    && (crontab -l ; echo "25 0-23/6 * * * python malwarebazaar.py") | crontab -

WORKDIR /home/scripter

RUN python turris.py \
    && python blocklistde.py \
    && python badips.py \
    && python danmeuk.py \
    && python malwarebazaar.py

USER root:root

CMD ["crond", "-n"]
```

Figura 17: Threats. Contenido Dockerfile

3.1.4.2 Fichero docker-compose.yml

En el fichero «docker-compose.yml» se realizan las siguientes tareas:

1. Construir el contenedor con la información del fichero Dockerfile.
2. Asignar la red «localnet».
3. Configura un solo volumen llamado «files» que contiene los ficheros de los «scripts» y los resultados y registros de sus ejecuciones.

```yaml
version: '3'
services:
  threats:
    build: .
    container_name: threats
    restart: always
    networks:
      - localnet
    volumes:
    - files:/home/scripter

volumes:
  files:

networks:
  localnet:
```

Figura 18: Threats. Contenido docker-compose.yml

3.1.4.3 Scripts de procesado

La lógica que siguen los «scripts» es muy similar en todos ellos y se puede resumir básicamente en los siguientes pasos:

1. Comprimir los resultados de la anterior ejecución. Para preservar el resultado de la anterior ejecución, los ficheros se comprimen en formato zip. Tan solo se mantiene la ejecución anterior.
2. Descargar los ficheros con los datos desde sus repositorios en Internet.
3. Procesarlos para convertirlos en ficheros CSV eliminando duplicados y unificando la información. El formato de estos ficheros se comenta en el apartado 3.1.4.4 Formato de los ficheros CSV.
4. Finalmente, se notifica mediante un «bot» de Telegram el resultado de la ejecución de los «scripts».

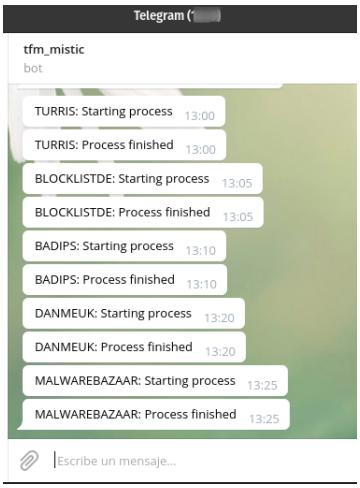


Figura 19: Telegram. Mensajes ejecución correcta

Cuando se produce un fallo, la notificación en Telegram muestra un mensaje solicitando que se revisen los registros del «script».

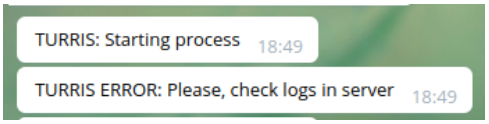


Figura 20: Telegram. Error en la ejecución de un script

Nota: Es necesario darse de alta en el sistemas de «bots» de Telegram para configurar en los «scripts» de recopilación de información las siguientes dos variables.

- bot_token. Identificador del «bot» creado.
- bot_chatID. Identificador de la sala del «bot».

3.1.4.4 Formato de los ficheros CSV

Por cada una de las fuentes se genera un único fichero CSV con los datos procesados para que sean fácilmente utilizables por Logstash.

Existen dos formatos de fichero de resultados diferentes. El primero es utilizado para las fuentes de BadIPs, Blocklist.de, Turris y DanMe.uk, mientras que el segundo se utiliza exclusivamente para la fuente Malware Bazaar.

El primer formato consta de los siguientes dos campos:

- **Address**. Se trata de la dirección IP del atacante.
- **Tags**. Se trata de las etiquetas que se le asocian, separadas por una coma.

```
[scripter@9a69e2a99acd downloads]$ head badips.csv
"Address","Tags"
"0.0.10.40","brute_force_login"
"0.0.7.209","brute_force_login"
"1.1.221.157","brute_force_login"
"1.10.156.102","brute_force_login,bots"
"1.10.159.46","brute_force_login,ssh,bots"
"1.11.201.19","ftp"
"1.11.244.148","brute_force_login"
"1.119.0.247","brute_force_login,ssh"
"1.119.12.115","ssh"
[scripter@9a69e2a99acd downloads]$
```

Figura 21: Threats. Fichero resultados badips

El segundo formato consta de los siguientes dos campos:

- **md5_hash**. Se trata del «hash» en formato MD5 del fichero.
- **signature**. Se trata de las etiquetas de «malware» que se le asocian.

```
[scripter@9a69e2a99acd downloads]$ head malwarebazaar.csv
"md5_hash","signature"
"a3c0d3d0c68fe280721f953443f8cf22","njrat"
"2fba4776c084d6602c5203a2b4e0da64","njrat"
"f7f7159cd1a75d6bf193840530ef41de","AgentTesla"
"a58e29e3b47366a7133898c1fd26b9eb","FormBook"
"d2c03864dcf6003537851a411452464d","n/a"
"0acecad57c4015e14d9b3bb02b433d3e","n/a"
"aa538e51b5b26100b2fd194b0ff83bda","WSHRAT"
"4d37387b65241e14a0beca273623fbbc","n/a"
"50f553d2ed9cfadd009151764d850f87","WSHRAT"
[scripter@9a69e2a99acd downloads]$
```

Figura 22: Threats. Fichero resultados Malware Bazaar

3.1.5 Contenedor Logstash

Este contenedor ejecuta una instancia de Logstash cuya función era procesar los registros que transmitía el contenedor Filebeat a través del puerto

TCP/5044. Sobre estos registros se han aplicado el siguiente conjunto de transformaciones:

- Convertir las IPs que se encuentran en los registros en datos geoespaciales para posteriormente poder contabilizarlo en gráficas de tipo mapa.
- Etiquetar los registros en función del fichero de registro de origen.
- Comparar las IPs y las sumas MD5 de los ficheros con los datos de las fuentes de inteligencia de amenazas para etiquetar los comportamientos anómalos.
- Eliminar referencias a tráfico interno local del servidor anfitrión dado que no es relevante para este proyecto.
- Eliminar campos que no aportan información relevante como, por ejemplo, el identificador del agente de Filebeat.

3.1.5.1 Fichero Dockerfile

A grandes rasgos en el fichero Dockerfile se realizan las siguientes tareas:

1. Actualizar la imagen base del sistema operativo
2. Instalar los siguientes paquetes:
    1. OpenJDK 1.8.
    2. La tarea cron para actualizar GeoIP.
3. Descargar la versión 7.6.1 OSS de Logstash y descomprimirla en «/opt/logstash» borrando el fichero original descargado.
4. Copiar al contenedor:
    1. Los ficheros de configuración de Logstash.
    2. El fichero de configuración de GeoIP.
    3. El fichero del punto de entrada.
5. Ajustar los permisos del fichero de entrada y ejecutar «geoipupdate» para que realice la primera descarga de la base de datos de GeoIP.
6. Asignar el directorio de trabajo y el punto de entrada.

```
FROM centos:7

LABEL maintainer "igalvit@uoc.edu"

RUN yum update -y \
    && yum -y install wget java-1.8.0-openjdk geoipupdate-cron \
    && yum clean all \
    && cd /tmp \
    && wget https://artifacts.elastic.co/downloads/logstash/logstash-oss-7.6.1.tar.gz \
    && tar xzvf logstash-oss-7.6.1.tar.gz \
    && mv logstash-7.6.1 /opt/logstash \
    && rm -f /tmp/logstash* \
    && mkdir /opt/geolite2

ENV PATH $PATH:/opt/logstash/bin

COPY config /opt/logstash/config

COPY geoip/GeoIP.conf /etc/GeoIP.conf

COPY entrypoint.sh /opt/entrypoint.sh

RUN chmod +x /opt/entrypoint.sh \
    && geoipupdate

WORKDIR /opt/logstash

ENTRYPOINT ["/opt/entrypoint.sh"]
```

Figura 23: Logstash. Contenido Dockerfile

3.1.5.2 Fichero docker-compose.yml

En el fichero «docker-compose.yml» se realizan las siguientes tareas:

1. Construir el contenedor con la información del fichero Dockerfile.
2. Asignar la red externa «elasticsearch_localnet»
3. Exponer el puerto TCP/9600 al resto de contenedores.
4. Configurar dos volúmenes Docker:
    1. Volumen «config». Volumen interno que contiene la configuración de Logstash.
    2. Volumen «threats_files». Volumen externo en solo lectura que proviene del contenedor Threats que contiene los ficheros CSV con la información de amenazas.

```
version: '3'
services:
  logstash:
    build: .
    container_name: logstash
    restart: always
    networks:
      - elasticsearch_localnet
    expose:
    - "9600"
    volumes:
    - config:/opt/logstash/config
    - threats_files:/opt/threats:ro

volumes:
  config:
  threats_files:
    external: true

networks:
  elasticsearch_localnet:
    external: true
```

Figura 24: Logstash. Contenido docker-compose.yml

3.1.5.2.1 Configuración de GeoIP

Es necesario configurar en el fichero «geoip/GeoIP.conf» los siguientes valores:

- AccountID. Identificador de la cuenta proporcionado por MaxMind.
- LicenseKey. Clave de la licencia proporcionada por MaxMind.
- EditionIDs. El tipo de base de datos que se va a utilizar. En este caso se ha utilizado «GeoLite2-City» que se corresponde a un nivel de detalle de ciudades.
- DatabaseDirectory. Ubicación dentro del contenedor de la base de datos Geolite2, en este caso la ruta es «/opt/geolite2/».

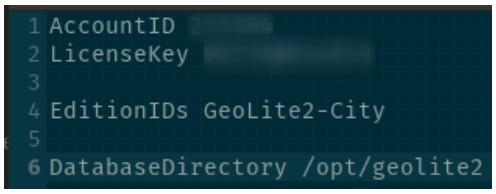


Figura 25: Logstash. Configuración GeoLite2

Nota: Para obtener esta información es necesario registrarse en la URL https://www.maxmind.com/en/geolite2/signup de MaxMind. Los datos correspondientes al registro se deberán añadir en el fichero de configuración de GeoIP que se encuentra en el Anexo dentro del apartado 8.8.8 Fichero GeoIP.conf. En concreto, se deben aportar los siguientes datos:

- **AccountID**. Identificador de la cuenta de Maxmind.
- **LicenseKey**. Número de licencia obtenida.

3.1.5.3 Configuración de Logstash

Se ha mantenido la configuración por defecto excepto para los siguientes valores:

- El nombre del nodo, dado que solo hay un nodo, se ha nomenclaturado como «logstash-1».
- El identificador del «pipeline» principal, que en este caso es «main».
- El número de trabajadores en paralelo que se van a ejecutar, que dado que el servidor dispone de dos procesadores, se ha establecido en «2».
- La ruta en disco del fichero de configuración del «pipeline» principal que se encuentra en la ruta «/opt/logstash/config/main.conf».

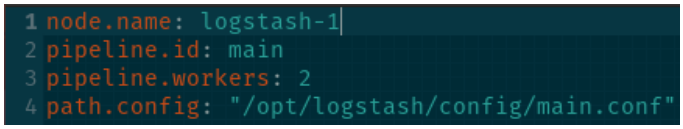


Figura 26: Logstash. Fichero de configuración

El contenido completo de este fichero se encuentra en el apartado 8.8.5 Fichero logstash.yml.

3.1.5.4 Configuración del «pipeline» principal

La configuración del «pipeline» principal se encuentra en el fichero «/opt/logstash/config/main.conf».

Este fichero contiene la información correspondiente a los siguientes elementos:

- **Entrada de registros**. Existe una única entrada de tipo «beats» que proviene del contenedor Filebeat a través del puerto 5044.

```
1 input {
2   beats {
3     port => 5044
4   }
5 }
```

Figura 27: Logstash. Configuración entrada

- **Filtros aplicados**. Contiene el conjunto de filtros que se aplican a los registros. Se ha realizado una breve introducción sobre los filtros en el apartado 3.1.5.5 Filtros aplicados en el «pipeline».
- **Salida de la información**. La salida de la información procesada se remite al contenedor de Elasticsearch a través del puerto TCP/9200. Adicionalmente se indica la expresión regular que permitirá conformar los indices y la plantilla de mapeos de Elasticsearch.

```
177 output {
178   elasticsearch {
179     hosts => ["http://elasticsearch:9200"]
180     index => "tfm-%{+YYYY.MM.dd}"
181     template => "/opt/logstash/config/tfm.json"
182     template_name => "tfm"
183   }
184 }
```

Figura 28: Logstash. Configuración salida

El contenido completo de este fichero se encuentra en el apartado 8.8.6 Fichero main.conf.

3.1.5.5 Filtros aplicados en el «pipeline»

A continuación se detallan los principales filtros que se han generado en el pipeline para transformar los datos que se envían a Elasticsearch:

- **Nombre fichero registro Zeek**. Se extrae del nombre del fichero de registro de Zeek para crear el campo «log_type».

```
13   grok {
14     match => { "[log][file][path]" => ["^%{GREEDYDATA}%
{WORD:file_type}.log"] }
15     named_captures_only => true
16     add_field => { "log_type" => "%{file_type}" }
17     tag_on_failure => []
18   }
```

Figura 29: Logstash. Filtro nombre ficheros registros Zeek

- **Geolocalización IP**. Se analizan todas los campos con IP para extraer datos de geolocalización a través de Geolite2 excluyendo la IP del servidor anfitrión.

```
20   if [id.orig_h] != "176.31.103.131" {
21     geoip {
22       source => "id.orig_h"
23       target => "geoip_orig_h"
24       tag_on_failure => []
25     }
```

Figura 30: Logstash. Filtro geolocalización IP

- **Búsqueda en fuentes de inteligencia de amenazas**. Cada IP que no sea la propia del servidor anfitrión se busca en la distintas fuentes procesadas.

```
26     translate {
27       field => "id.orig_h"
28       destination => "threat_type"
29       dictionary_path => "/opt/threats/downloads/turris.csv"
30       refresh_behaviour => "replace"
31       add_field => { "threat_source" => "turris" }
32       add_tag => "threat"
33     }
```

Figura 31: Logstash. Filtro búsqueda fuentes inteligencia amenazas

- **Separación del campo de tipos de amenazas**. Se separan los valores unidos por comas del campo «threat_type».

```
140
141   mutate {
142     split => { "threat_type" => ',' }
143   }
```

Figura 32: Logstash. Separación datos campo amenazas

- **Eliminación de campos no necesarios**. Se eliminan aquellos campos que no aportan información relevante para el proyecto, incluyendo el tráfico interno de los contenedores.

```
145   mutate {
146     remove_field => [ "ts", "[log][offset]", "[agent][ephemeral_id]",
    "[agent][id]", "[agent][type]", "[agent][version]", "[log][file][path]",
    "[input][type]", "file_type" ]
147     remove_tag => "beats_input_raw_event"
148   }
149
150   if [local_orig] == 'true' and [local_resp] == 'true' {
151     drop { }
152   }
153 }
```

Figura 33: Logstash. Eliminación datos innecesarios

3.1.5.6 Configuración de la plantilla de mapeos

La configuración de la plantilla de mapeos se encuentra en el fichero «/opt/logstash/config/tfm.json». Esta plantilla es necesaria para que Elasticsearch tenga una definición exacta del tipo para todos o algunos de los campos que va a almacenar.

Este fichero es bastante extenso, pero su contenido básicamente contiene los mapeos de propiedades necesarios para la geolocalización de las IP registradas y de para los datos de tipo IP, y que por defecto, no son mapeados correctamente por Elasticsearch.

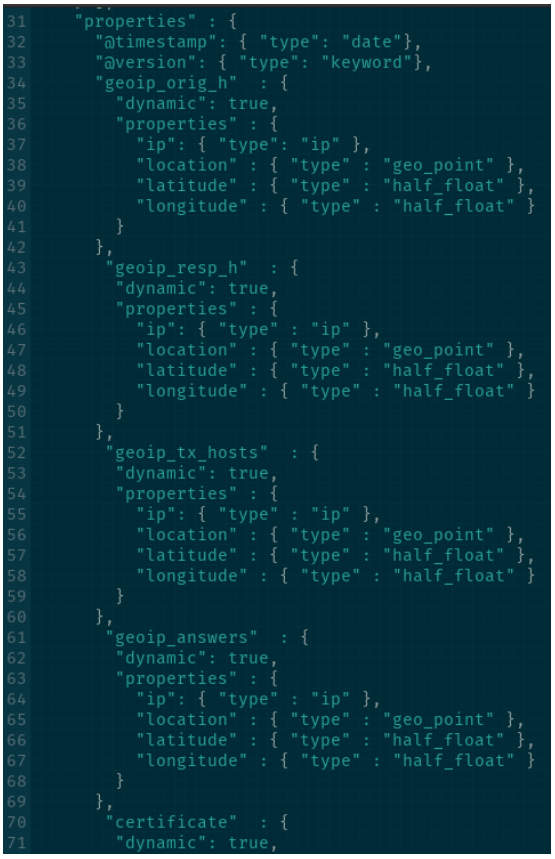


Figura 34: Logstash. Fichero configuración mapeos

El contenido completo de este fichero se encuentra en el apartado 8.8.7 Fichero tfm.json.

3.1.6 Contenedor Elasticsearch

Este contenedor ejecuta una instancia de Elasticsearch que se encarga de almacenar toda la información procesada por el contenedor Logstash que este le remite a través del puerto TCP/9200.

En este proyecto se ha desplegado un clúster de un único nodo basandose en dos motivos:

- La escasa carga esperada derivada del almacenamiento de la información.
- La limitación de recursos que tiene el «hardware» disponible.

3.1.6.1 Fichero Dockerfile

A grandes rasgos en el fichero Dockerfile se realizan las siguientes tareas:

1. Actualizar la imagen base del sistema operativo.
2. Instalar el paquete OpenJDK 1.8.0.
3. Crear el usuario y el grupo «elasticsearch» junto con las rutas donde se instalará Elasticsearch.
4. Descargar desde Internet la versión 7.6.1 OSS de Elasticsearch y descomprimirla en su ruta borrando el fichero descargado.
5. Copiar los ficheros de configuración de Elasticsearch.

6. Ajustar los permisos de los directorios de instalación y del fichero de entrada.
7. Asignar el usuario de ejecución, el directorio de trabajo y el punto de entrada.

```
FROM centos:7

LABEL maintainer "igalvit@uoc.edu"

RUN yum update -y \
    && yum -y install wget java-1.8.0-openjdk \
    && yum clean all

RUN groupadd -g 1200 elasticsearch \
    && useradd -u 1200 -g 1200 -d /opt/elasticsearch elasticsearch \
    && mkdir /opt/elasticsearch/data \
    && mkdir /opt/elasticsearch/logs \
    && cd /tmp \
    && wget https://artifacts.elastic.co/downloads/elasticsearch/-
elasticsearch-oss-7.6.1-linux-x86_64.tar.gz \
    && tar xzvf elasticsearch-oss-7.6.1-linux-x86_64.tar.gz \
    && mv elasticsearch-7.6.1/* /opt/elasticsearch/ \
    && rm -rf /tmp/elasticsearch*

ENV PATH $PATH:/opt/elasticsearch/bin

COPY config /opt/elasticsearch/config

COPY entrypoint.sh /opt/entrypoint.sh

RUN chown -R elasticsearch. /opt/elasticsearch \
    && chown elasticsearch. /opt/entrypoint.sh \
    && chmod +x /opt/entrypoint.sh

WORKDIR /opt/elasticsearch

USER elasticsearch:root

ENTRYPOINT ["/opt/entrypoint.sh"]

CMD ["elasticsearch"]
```

Figura 35: Elasticsearch. Contenido Dockerfile

3.1.6.2 Fichero docker-compose.yml

En el fichero «docker-compose.yml» se realizan las siguientes tareas:

1. Construir el contenedor con la información del fichero Dockerfile.
2. Asignar la red «elasticsearch_localnet».
3. Exponer el puerto TCP/9200 al resto de contenedores.
4. Configurar dos volúmenes Docker:
    1. Volumen «config». Volumen interno que contiene la configuración de Elasticsearch.
    2. Volumen «data». Volumen interno que contiene la base de datos de Elasticsearch.

```yaml
version: '3'
services:
  elasticsearch:
    build: .
    container_name: elasticsearch
    restart: always
    networks:
      - localnet
    expose:
      - "9200"
    volumes:
      - config:/opt/elasticsearch/config
      - data:/opt/elasticsearch/data

volumes:
  config:
  data:

networks:
  localnet:
```

Figura 36: Elasticsearch. Contenido docker-compose.yml

3.1.6.3 Configuración de Elasticsearch

Se ha mantenido la configuración por defecto excepto para los siguientes valores:

- El nombre del clúster, que se ha definido como «tfm-cluster».
- El nombre del nodo, que se ha definido como «node-1».
- Las rutas de los directorios de datos («data») y de registros («logs») dentro de la ruta «/opt/elasticsearch».
- Que escuche en todas las interfaces de red.
- Configurar el nodo maestro, que es el único nodo desplegado.

```
cluster.name: tfm-cluster
node.name: node-1
path.data: /opt/elasticsearch/data
path.logs: /opt/elasticsearch/logs
network.host: 0.0.0.0
cluster.initial_master_nodes: ["node-1"]
```

Figura 37: Elasticsearch. Fichero de configuración

3.1.7 Contenedor Kibana

Este contenedor ejecuta una instancia de Kibana que es el entorno web gráfico sobre el que interactuarán los usuarios.

Se comunica con el contenedor Elasticsearch a través del puerto TCP/9200 y a su vez expone el puerto TCP/5601.

3.1.7.1 Fichero Dockerfile

A grandes rasgos en el fichero Dockerfile se realizan las siguientes tareas:

1. Actualizar la imagen base del sistema operativo.
2. Crear el usuario y el grupo «kibana».
3. Descargar desde Internet la versión 7.6.1 OSS de Kibana y descomprimirla en su ruta borrando el fichero descargado.
4. Copiar los ficheros de configuración de Kibana.

5. Ajustar los permisos del fichero de entrada.
6. Asignar el usuario de ejecución, el directorio de trabajo y el punto de entrada.

```
FROM centos:7

LABEL maintainer "igalvit@uoc.edu"

RUN yum update -y \
    && yum -y install wget \
    && yum clean all \
    && groupadd -g 1300 kibana \
    && useradd -u 1300 -g 1300 -d /opt/kibana kibana

USER kibana:kibana

RUN cd /tmp \
    && wget https://artifacts.elastic.co/downloads/kibana/kibana-oss-7.6.1-linux-x86_64.tar.gz \
    && tar xzvf kibana-oss-7.6.1-linux-x86_64.tar.gz \
    && mv kibana-7.6.1-linux-x86_64/* /opt/kibana/ \
    && rm -rf /tmp/kibana*

ENV PATH $PATH:/opt/kibana/bin

COPY --chown=kibana:kibana config /opt/kibana/config

COPY --chown=kibana:kibana entrypoint.sh /opt/kibana/entrypoint.sh

RUN chmod +x /opt/kibana/entrypoint.sh

WORKDIR /opt/kibana

ENTRYPOINT ["/opt/kibana/entrypoint.sh"]

CMD ["kibana"]
```

Figura 38: Kibana. Contenido Dockerfile

3.1.7.2 Fichero docker-compose.yml

En el fichero «docker-compose.yml» se realizan las siguientes tareas:

1. Construir el contenedor con la información del fichero Dockerfile.
2. Asignar la red externa «elasticsearch_localnet».
3. Exponer el puerto TCP/5601 al resto de contenedores.
4. Configurar un volúmen Docker interno llamado «config» que contiene la configuración de Kibana.

```
version: '3'
services:
  kibana:
    build: .
    container_name: kibana
    restart: always
    networks:
      - elasticsearch_localnet
    expose:
      - "5601"
    volumes:
      - config:/opt/kibana/config
volumes:
  config:

networks:
  elasticsearch_localnet:
    external: true
```

Figura 39: Kibana. Contenido docker-compose.yml

3.1.7.3 Configuración de Kibana

Se ha mantenido la configuración por defecto excepto para:

- Que escuche en todas las interfaces de red.
- Asignarle un nombre de servidor, en este caso «kibana».
- Indicar cuál es el clúster de Elasticsearch al que debe conectarse.

```
1 server.host: 0.0.0.0
2 server.name: "kibana"
3 elasticsearch.hosts: ["http://elasticsearch:9200"]
```

Figura 40: Kibana. Fichero de configuración

Antes de poder visualizar los datos desde la interfaz web de Kibana es necesario importar manualmente el fichero «export.ndjson» que se encuentra en la URL XXXXXXXX. Este paso se realiza exclusivamente la primera vez que se accede a Kibana. Se debe acceder a la sección de Administración y dentro de esta a la opción «Saved Objects».

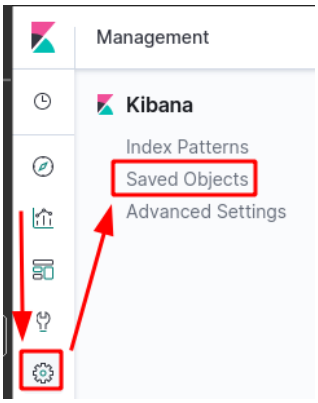


Figura 41: Kibana. Opción «Saved Objects»

A continuación hay que pulsar en el botón 

de la parte superior derecha y se debe seleccionar el fichero en la ruta donde se haya descargado.

Por ultimo es necesario pulsar el botón 

que se encuentra en la parte inferior derecha y si el proceso finaliza correctamente se visualizar el mensaje de la captura siguiente.

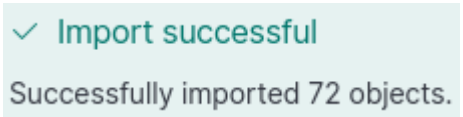


Figura 42: KIbana. Importación correcta

Por último, es necesario indicarle a Kibana que utilice el patrón de indices definido en el fichero recién importando. Para realizar este paso, se debe pulsar sobre «Advanced Settings» y buscar la opción «Default index». En este campo se debe escribir «tfm-*» y pulsar el botón «Save» tal y como se observa en la siguiente figura.

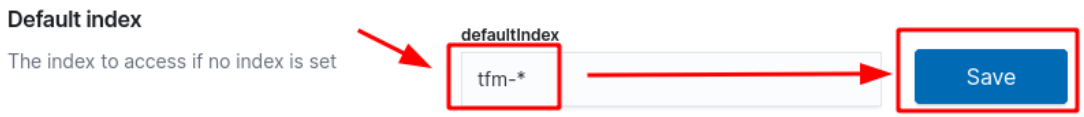


Figura 43: KIbana. Configuración patrón indices por defecto

3.1.8 Contenedor haproxy

Este contenedor ejecuta una instancia del software HA Proxy que se encarga de permitir la comunicación del contenedor Kibana con el exterior actuando como un «proxy web». Las dos funciones principales que realiza son:

- **Autenticación de usuarios**. Dado que la versión OSS de Kibana no incluye ningún módulo de autenticación, se ha utilizado este componente para disponer de autenticación básica.
- **Cifrado de las comunicaciones**. Mediante un certificado digital se encarga de establecer una comunicación cifrada mediante SSL a través del puerto TCP 5602 con los consumidores de la interfaz web de Kibana.

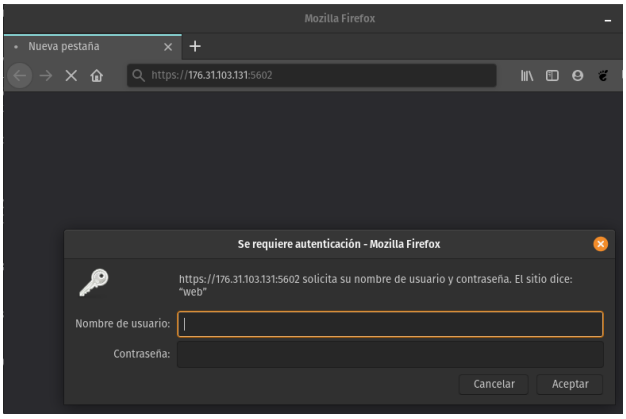


Figura 44: HA Proxy. Autenticación en Kibana

Una vez que se ha autenticado correctamente, se permite el acceso al componente de Kibana.

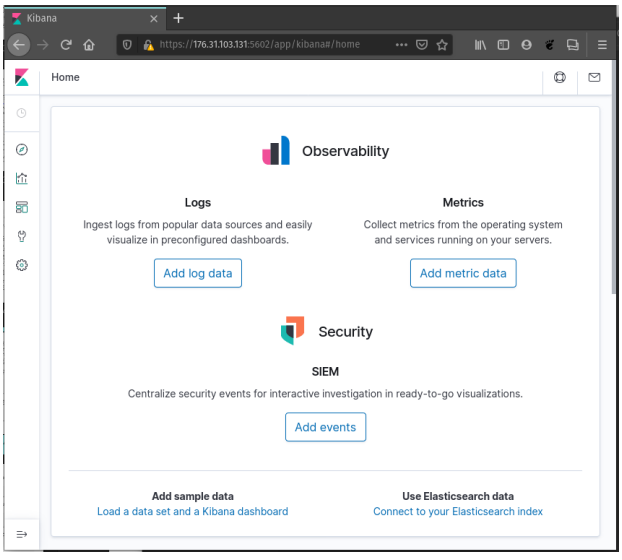


Figura 45: HA Proxy. Acceso a Kibana

3.1.8.1 Configuración del contenedor

3.1.8.2 Fichero Dockerfile

A grandes rasgos en el fichero Dockerfile se realizan las siguientes tareas:

1. Actualizar la imagen base del sistema operativo e instalar el paquete de HA proxy.
2. Copiar los ficheros de configuración de HA proxy.
3. Ajustar los permisos del fichero de entrada.
4. Asignar el punto de entrada.

```
FROM centos:7

LABEL maintainer "igalvit@uoc.edu"

RUN yum update -y \
    && yum -y install haproxy \
    && yum clean all

COPY config /etc/haproxy

RUN chmod +x /etc/haproxy/entrypoint.sh

CMD ["haproxy", "-f", "/etc/haproxy/haproxy.cfg"]
```

Figura 46: HA Proxy. Contenido Dockerfile

3.1.8.3 Fichero docker-compose.yml

En el fichero «docker-compose.yml» se realizan las siguientes tareas:

1. Construir el contenedor con la información del fichero Dockerfile.
2. Asignar la red externa «elasticsearch_localnet».
3. Exponer el puerto TCP/5602 hacia la red exterior.
4. Configurar un volumen Docker interno llamado «config» que contiene la configuración de HA Proxy.
5. Crear un enlace externo entre hacia el contenedor Kibana para que este contenedor pueda resolver el nombre «kibana».

```
version: '3'
services:
  haproxy:
    build: .
    container_name: haproxy
    restart: always
    networks:
      - elasticsearch_localnet
    ports:
      - "5602:5602"
    volumes:
      - config:/opt/haproxy
    external_links:
      - kibana
volumes:
  config:

networks:
  elasticsearch_localnet:
    external: true
```

Figura 47: HA Proxy. Contenido docker-compose.yml

3.2 Playbook Ansible

Con el objetivo de facilitar el despliegue y puesta en marcha de la solución, se ha generado un «playbook» de Ansible que se encarga de las siguientes tareas:

1. Instalar y configurar el servidor anfitrión donde se van a desplegar los contenedores Docker.
2. Desplegar cada uno de los contenedores Docker copiando los ficheros que los conforman, construyéndolos y finalmente, iniciándolos.
3. Por último, modifica la configuración del servidor anfitrión para añadir mayor seguridad evitando la conexión mediante SSH a través de contraseñas o con la cuenta «root».

3.2.1 Fichero principal

El fichero principal del «playbook» se ha denominado «main.yml» y se encuentra en la raíz del directorio. Este se encarga de que se ejecute la secuencia de roles, que excluyendo al primero y al último, se corresponden con el conjunto de contenedores que conforman la solución.

```
1  ---
2  - hosts: tfm
3    vars:
4      main_path: '/opt/tfm_mistic/'
5    gather_facts: true
6    become: yes
7    roles:
8      - prerequisitos
9      - threats
10     - heralding
11     - elasticsearch
12     - zeek
13     - filebeat
14     - kibana
15     - logstash
16     - haproxy
17     - postconfig
```

Figura 48: Ansible. Fichero principal

3.2.2 Fichero hosts

Contiene el inventario de servidores, en este caso uno, sobre el que se va a desplegar la solución. Este fichero requiere de personalización para adaptarlo al entorno dónde se realice el despliegue. En concreto se debe modificar los siguientes atributos:

- **ALIAS_HOST**. Alias del servidor anfitrión para Ansible.
- **ansible_host**. IP o FQDN del servidor anfitrión.
- **ansible_port.** Puerto de acceso SSH del servidor anfitrión.
- **ansible_user**. Identificador del usuario que accede por SSH al servidor anfitrión.
- **ansible_ssh_pass.** Contraseña del usuario que accede por SSH al servidor anfitrión.
- **ansible_sudo_pass**. Contraseña para hacer «sudo».

3.2.3 Roles

A continuación se describe el conjunto de roles creados para el «playbook». No se realiza una descripción exhaustiva porque los que despliegan los contenedores se limitan a realizar el despliegue apoyándose en los ficheros de definición de los contenedores ya detallados en apartados anteriores.

1. **prerequisitos**. Despliegue los requisitos iniciales como la actualización del servidor, usuarios de acceso, claves RSA y la instalación de Docker.
2. **threats**. Despliega el contenedor Threats.
3. **heralding**. Despliega el contenedor Heralding.
4. **elasticsearch**. Despliega el contenedor Elasticsearch.
5. **zeek**. Despliega el contenedor Zeek.
6. **filebeat**. Despliega el contenedor Filebeat.
7. **kibana**. Despliega el contenedor Kibana.
8. **logstash**. Despliega el contenedor Logstash.
9. **haproxy**. Despliega el contenedor HAproxy.
10. **postconfig**. Securiza el servidor anfitrión configuración las actualizaciones automáticas y limitando la forma de conexión mediante SSH para permitirla exclusivamente mediante claves RSA.

3.3 Pruebas de integración

3.3.1 Funcionamiento del «honeypot»

Para verificar el funcionamiento del «honeypot» Heralding se ha realizado un escaneo mediante la herramienta Zenmap. En concreto se ha realizado un escaneo mediante el perfil «Quick scan».

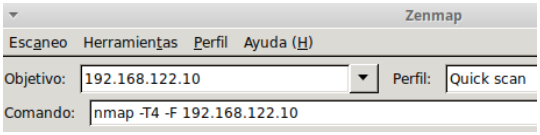


Figura 49: Pruebas integración. Escaneo Zenmap

En la salida de Zenmap se puede comprobar los resultados del escaneo y se puede verificar que los servicios abiertos en Heralding responden.

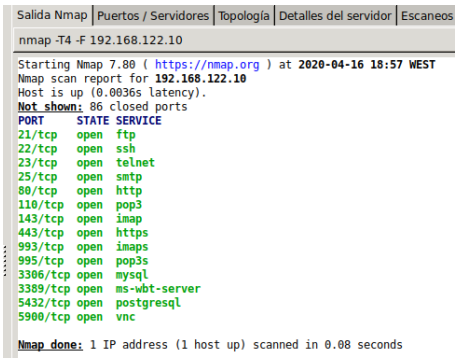


Figura 50: Pruebas integración. Resultado escaneo Zenmap

Se puede comprobar en el fichero «log_session.csv» del contenedor Heralding que han quedado registrados el escaneo realizado.

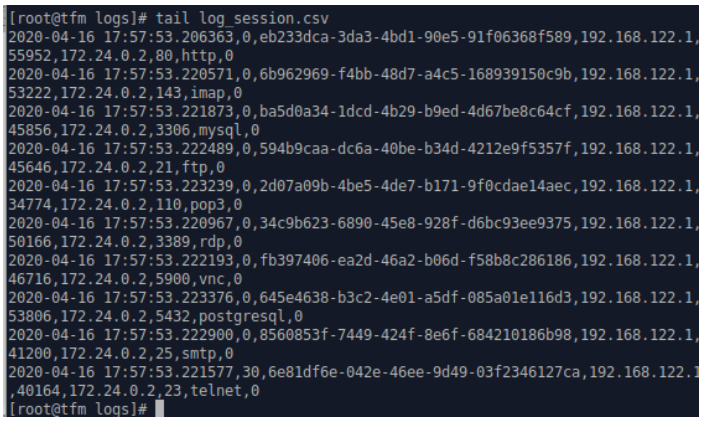


Figura 51: Pruebas integración. Fichero registro Heralding

3.3.2 Captura de tráfico desde Zeek

Para verificar el correcto funcionamiento de Zeek se ha revisado el registro de conexiones de Zeek («conn.log») en busca de los registros correspondientes al escaneo de red realizado para verificar el funcionamiento del «honeypot» Heralding.

Para completar esta tarea se ha utilizado el siguiente comando:

```
zgrep '"id.orig_h":"192.168.122.10"' conn.17\:20\:52-18\:00\:00.log.gz |grep
1587059873
```

En la siguiente captura se puede observar la salida y se puede verificar que se corresponden con los datos del escaneo comentado.

Figura 52: Pruebas integración. Fichero registro Zeek

3.3.3 Reenvío de logs desde Filebeat

A través de la visualización de los datos en Elasticsearch y en Kibana se ha podido verificar de forma indirecta que los registros se han remitido a Logstash puesto que los registros se encuentran insertados en la base de datos.

Se ha realizado una segunda prueba que ha consistido en modificar la configuración de Filebeat para que muestre por pantalla los registros capturados.

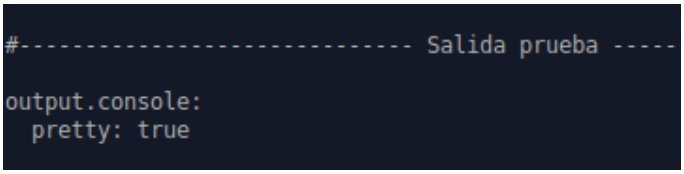


Figura 53: Pruebas integración. Configuración pruebas Filebeat

En la siguiente captura se puede observar que el contenedor comienza a mostrar por consola los datos capturados de los registros de Zeek.

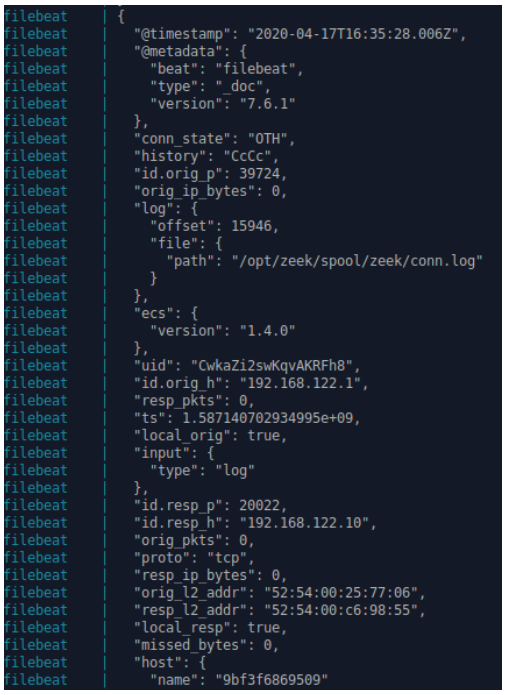


Figura 54: Pruebas integración. Salida pruebas Filebeat

3.3.4 Procesado de fuentes de información sobre amenazas

Se ha realizado una ejecución manual de los «scripts» para verificar que producen la salida correcta. Una vez ejecutados los «scripts» se han extraído las cinco primeras líneas de cada uno de los ficheros de resultados y se ha verificado que contienen los resultados en el formato esperado.

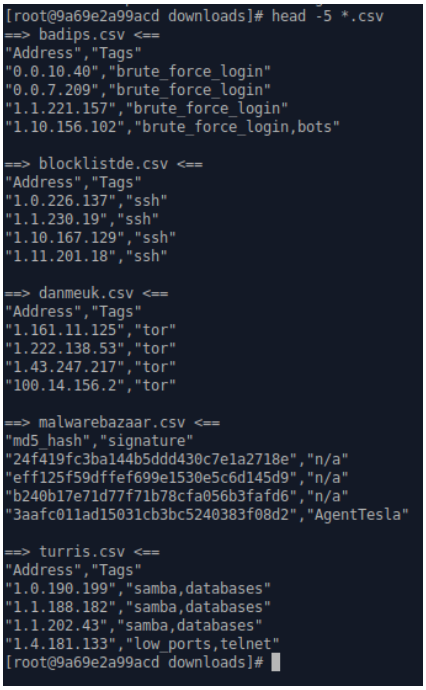


Figura 55: Pruebas integración. Salida ficheros resultados scripts amenazas

3.3.5 Procesado de los registros en Logstash

A través de la visualización de los datos en Elasticsearch y en Kibana se ha podido verificar de forma indirecta que el procesamiento de los registros a través de Logstash ha funcionado puesto que los registros se encontraban insertados en la base de datos.

Se ha realizado una segunda prueba que ha consistido en modificar la configuración de Logstash para que mostrase por pantalla la salida del procesamiento de los registros.

```
# --------- Salida de prueba -----------

output {
  stdout { codec => json }
}
```

Figura 56: Pruebas integración. Configuración pruebas Logstash

En la siguiente captura se puede observar que el contenedor comienza a mostrar por consola los datos procesados que le llegan desde Filebeat.

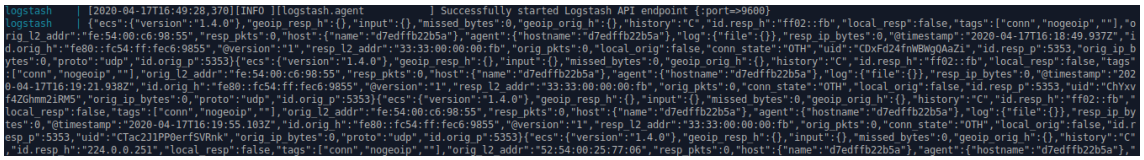


Figura 57: Pruebas integración. Salida pruebas Logstash

3.3.6 Almacenamiento de datos en Elasticsearch

A través de la interfaz web de Kibana se ha realizado una consulta directa a la base de datos Elasticsearch con un filtro sobre el rango de fechas correspondiente a la prueba inicial. De la búsqueda se ha obtenido un resultado que informa que se han obtenido 135 resultados que se corresponde con el número de entradas en el registro de Zeek.

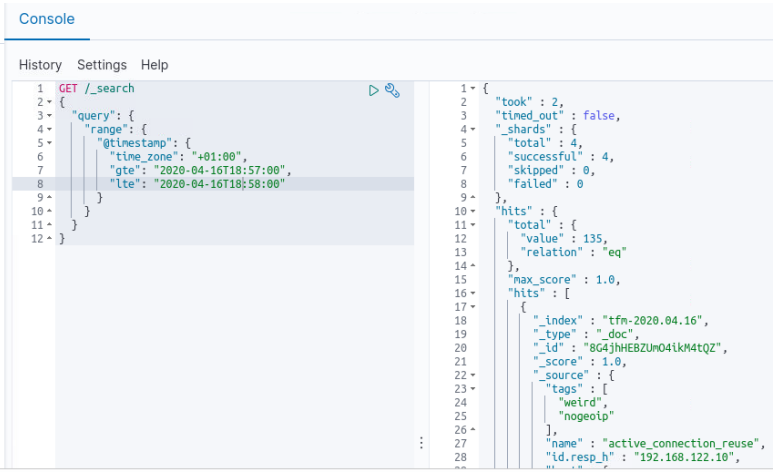


Figura 58: Pruebas integración. Primera prueba consulta Elasticsearch

3.3.7 Visualización de información desde Kibana

Para verificar el funcionamiento de la interfaz web de Kibana se ha accedido a la misma y se ha realizado una búsqueda de los datos registrados de la prueba inicial. Se ha verificado que la información se visualizaba correctamente.

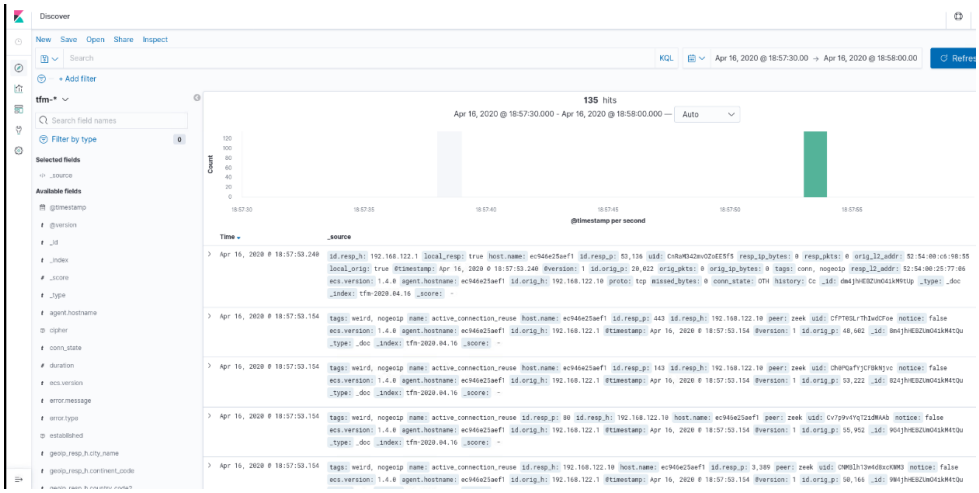


Figura 59: Pruebas integración. Visualización registros mediante Kibana

Se ha realizado una segunda prueba que consiste en generar una visualización de tipo nube de etiquetas.

Figura 60: Pruebas integración.
Nube de etiquetas Kibana

3.4 Captura y análisis de datos preliminar

En este apartado se describe la información obtenida de las primeras capturas de datos y se realiza un análisis preliminar de la información obtenida.

Este análisis previo no se corresponde con las conclusiones finales, sino que pretende aportar una primera visión de lo que la solución ha sido capaz de capturar durante los primeros días de vida del sistema completo expuesto a la red de Internet.

Durante esta fase, se ha generado un cuadro de mandos especifico con información sobre métricas generales. La información capturada comprende exclusivamente siete días, lo que no se considera un periodo lo suficientemente amplio para que la información no se encuentre sesgada.

3.4.1 Cuadro de mandos generado

El cuadro de mandos generado contiene las siguientes diez visualizaciones y tiene el aspecto mostrado en la siguiente captura.

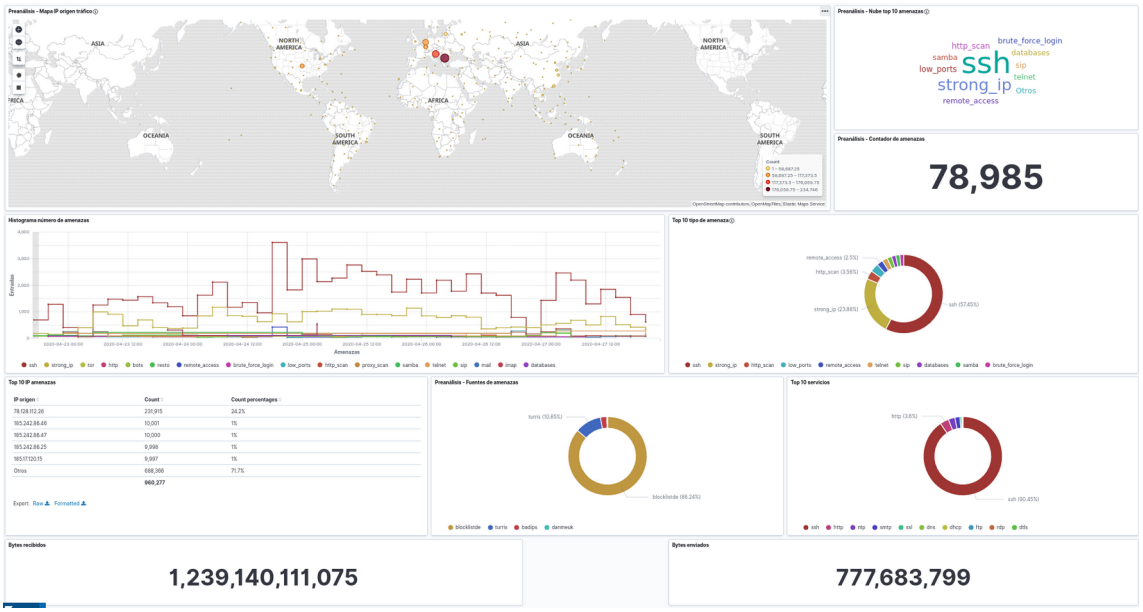


Figura 61: Análisis preliminar. Cuadro de mandos análisis preliminar

El cuadro de mandos contiene las siguientes visualizaciones:

- **Mapa IP origen.** Se trata de un mapa cartográfico que muestra la localización geográfica de las distintas IPs de origen de las conexiones agrupando por orígenes las localizaciones y graduandolas mediante colores.
- **Nube top 10 amenazas**. Se trata de una nube de palabras que representa los diez tipos de amenazas que más se han detectado. Se ha añadido porque permite realizar de forma fácil filtros por tipo de amenaza que a su vez que afectan a todo el cuadro de mandos.
- **Contador de amenazas.** Se trata de una métrica que indica el número total de amenazas detectadas.
- **Histograma número de amenazas**. Se trata de un histograma con el número de amenazas detectadas a lo largo del tiempo donde cada serie representa un tipo de amenaza distinta.
- **Top 10 tipo de amenazas**. Diagrama de tipo tarta que contiene los diez tipos de amenazas más detectados.
- **Top 10 IP amenazas.** Lista de tipo tabla que contiene las diez IP que más conexiones han establecido con el «honeypot».
- **Fuentes de amenazas**. Diagrama de tipo tarta que contiene la proporción del origen de las fuentes de amenazas detectadas.
- **Top 10 servicios.** Diagrama de tipo tarta que contiene los diez servicios más accedidos del «honeypot».
- **Bytes recibidos**. Métrica que contiene el total de los bytes recibidos desde todas las conexiones.
- **Bytes enviados**. Métrica que contiene el total de los bytes enviados desde todas las conexiones.

3.4.2 Análisis preliminar de datos

De los datos recabados sorprende enormemente la cantidad de actividades sospechosas que se han detectado, que suman un total de 78985 entradas, teniendo en cuenta que se trata de un servidor expuesto a la red de Internet por muy poco tiempo y sobre el que se no se ha realizado ninguna clase de publicidad ni de actividad comercial.

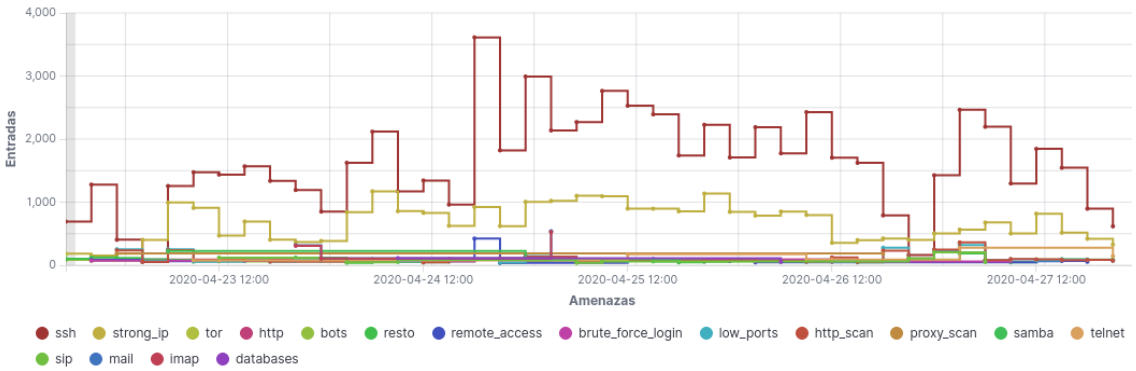


Figura 62: Histograma de ocurrencia de amenazas

Figura 63: Análisis preliminar. Contador de amenazas

En relación a los servicios más atacados, destaca sobremanera el servicio SSH que se lleva prácticamente dos tercios de los intentos, y a continuación seguido de los servicios DNS, NTP, SMTP, HTTP y HTTP seguro (SSL). Se esperaba este posicionamiento en lo relacionado al servicio SSH pero ha resultado todo una sorpresa que los servicios DNS y NTP estuvieran por delante de un servicio susceptible de ser el destino de muchos ataques como suele ser HTTP.

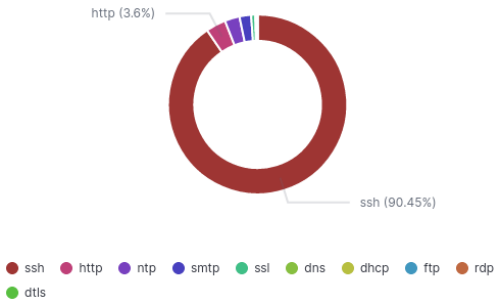


Figura 64: Análisis preliminar. Top 10 servicios

En relación al tipo de amenazas detectadas, el tipo de amenaza que más se ha detectado es el de tipo «ssh» con una ocurrencia del 57.45%, seguido de «strong_ip» con un 23.88% y en tercer lugar «http_scan» con un 3.56%. Estos resultados son lógicos en relación al punto anterior, y lo que nos confirma la preferencia por parte de los atacantes por intentar vulnerar el acceso remoto por SSH.

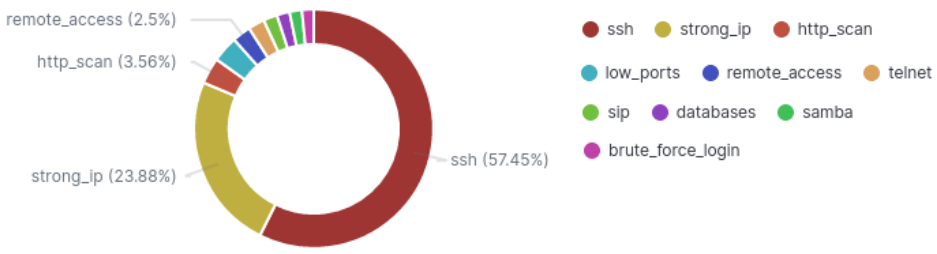


Figura 65: Análisis preliminar. Top 10 tipo de amenazas

En relación a las fuentes de inteligencia de amenazas, se confirma que están aportando bastante información sobre el origen de las conexiones y su categorización, por lo que a pesar de ser fuentes de carácter abierto y gratuito contienen información confiable. En esta primera evaluación las listas de Blocklist.de y Turris están aportando más del 97 por cierto de esta información.

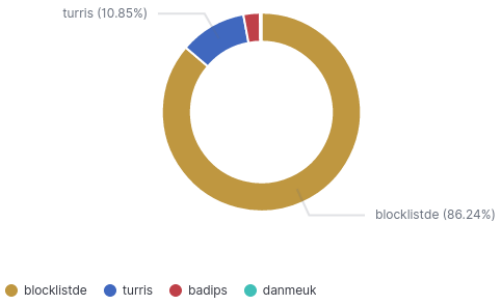


Figura 66: Análisis preliminar. Fuentes amenazas

En relación a la geolocalización de los orígenes de las conexiones, me ha sorprendido que la mayoría del tráfico proviene de Bulgaria y Alemania, seguidos de China, la Isla de Man, otra sorpresa, y Estados Unidos de América. Independientemente de esto, si se revisan los datos de los «números pequeños» se puede ver actividad desde prácticamente cualquier territorio del planeta.

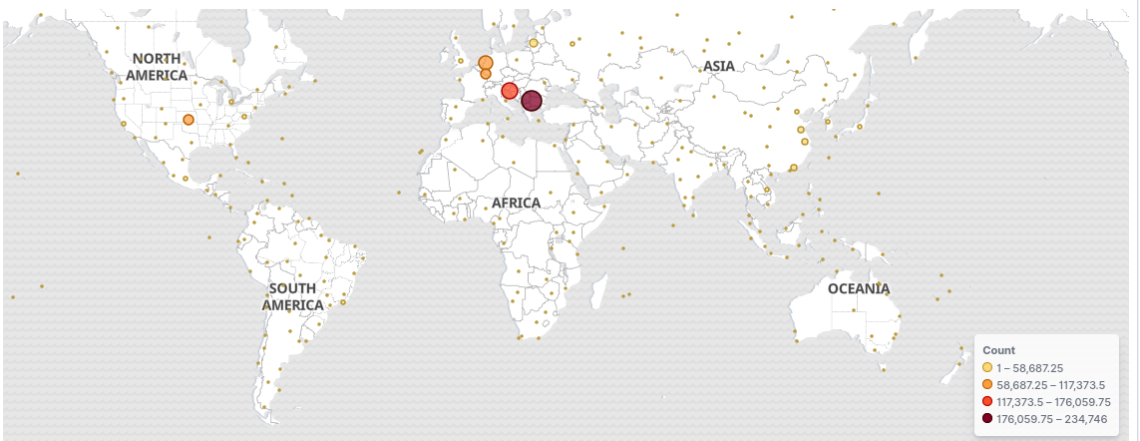


Figura 67: Análisis preliminar. Mapa IP origen conexiones

Por último quiero comentar que la máquina está soportando una cantidad enorme de tráfico para tratarse de poco más que un «honeypot». Se han contabilizado casi 1,24 TB de tráfico entrante y poco más de 0,77 GB de tráfico saliente.

Figura 68: Análisis preliminar. Bytes recibidos

Figura 69: Análisis preliminar. Bytes enviados

4 Análisis y evaluación de los datos generados

Durante un periodo de diez días se estuvo recopilando información de las distintas conexiones que se iban produciendo sin que hubiera una actuación proactiva para generar tráfico adicional más allá del necesario para poder manipular la plataforma.

4.1 Cuadro de mandos generados

Por cada uno de los ficheros de registro generados por la herramienta Zeek, se ha generado un cuadro de mandos especifico.

Todos los cuadros de mandos cuentan con las siguientes gráficas comunes:

- **Histograma de conexiones**. Muestra la evolución de las conexiones que se han ido realizado a lo largo del tiempo.
- **Contador de amenazas**. Indica el número de peticiones de conexión etiquetadas como amenazas.
- **Contador de conexiones**. Indica el número de peticiones de conexión recibidas.
- **Media de amenazas por día**. Indica la media de peticiones de conexión etiquetadas como amenaza por día.
- **Top 10 IP origen amenazas**. Muestra la distribución de las IP de origen que más peticiones etiquetadas como amenazas se han registrado.
- **Porcentaje de amenazas**. Muestra la distribución del número de peticiones por tipo de amenaza etiquetadas.
- **Fuentes de amenazas**. Muestra la distribución de las fuentes de amenazas a la hora de proporcionar el etiquetado como amenaza.

Adicionalmente para cada cuadro de mandos secundario se han generado una serie de gráficas adicionales que proporcionan información relacionada con el tipo de tráfico especifico.

4.1.1 Cuadro de mandos Principal

Este cuadro de mandos contiene la información acerca del tráfico de las conexiones correspondiente al fichero «conn.log». Se ha utilizado para generar una visión general de todo el tráfico sin entrar en las especificidades de cada tipo que es capaz de analizar Zeek IDS.

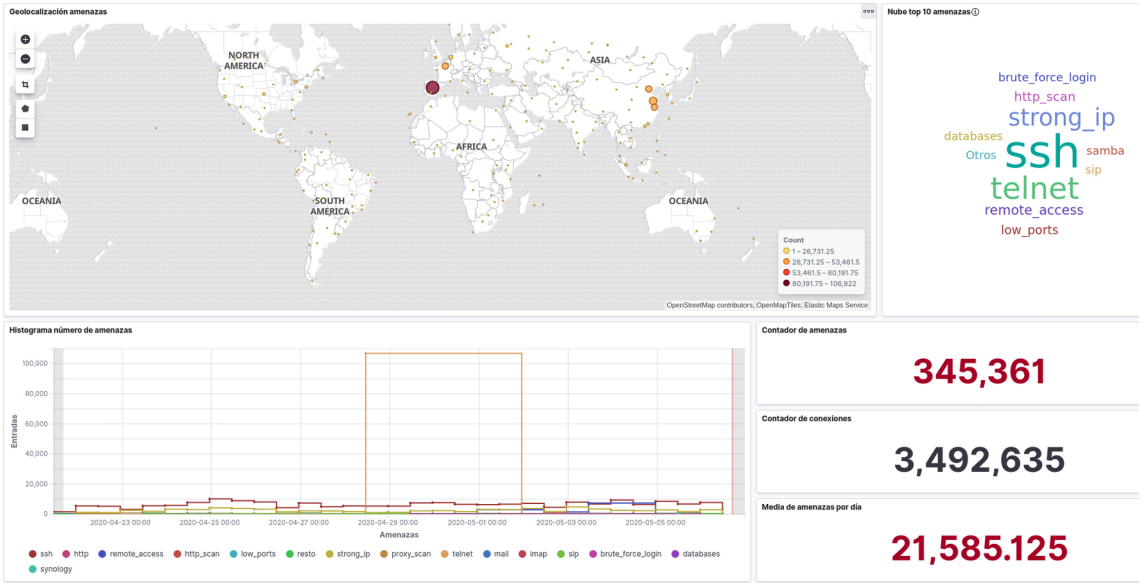


Figura 70: C.M. Principal. Parte superior del cuadro

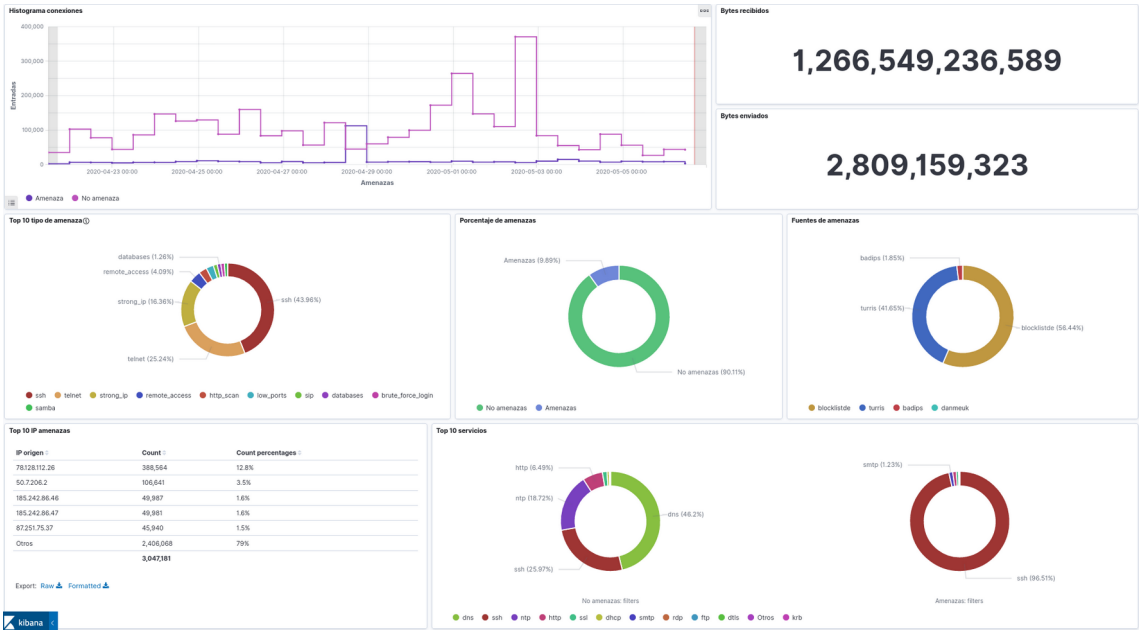


Figura 71: C.M. Principal. Parte inferior del cuadro

4.1.2 Cuadro de mandos DHCP

Este cuadro de mandos es diferente del resto porque el tráfico DHCP detectado se corresponde con el segmento de red donde se encuentra el servidor anfitrión que aloja toda la solución y es tráfico DHCP generado por el proveedor de servicios para el mantenimiento de sus servidores.

Por cuestiones de privacidad en la captura que se muestra se han ofuscado los datos de los servidores.

mac.keyword: Descending	Count	Count percentages
0	18	14.9%
0	15	12.4%
0	15	12.4%
3	15	12.4%
e	9	7.4%
0	8	6.6%
0	7	5.8%
0	6	5%
6	5	4.1%
3	3	2.5%
Otros	20	16.5%
	121	

Figura 72: C.M. DHCP. Top 10 Direcciones MAC peticiones

server_addr.keyword: Descendin	Count	Count percentages
176	68	56.2%
176	53	43.8%
	121	

Figura 73: C.M. DHCP. Listado de servidores DHCP

4.1.3 Cuadro de mandos DNS

Este cuadro de mandos aporta información acerca de las conexiones de tipo DNS que se han producido al «honeypot». Se ha generado cuatro visualizaciones adicionales, la primera enumera las consultas DNS más solicitadas, la segunda la proporción de peticiones por protocolo TCP o UDP, la tercera muestra la proporción de tipos de clases de consulta DNS, y la última la proporción por tipo de consulta.

query.keyword: Descending	Count	Count percentages
*\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00	145	37.7%
_services._dns-sd._udp.local	56	14.5%
version.bind	41	10.6%
www.example.com	32	8.3%
	18	4.7%
dnsscan.shadowserver.org	14	3.6%
www.wikipedia.org	13	3.4%
com	8	2.1%
1x1.cz	6	1.6%
www.google.es	6	1.6%
Otros	46	11.9%
	385	

Figura 74: C.M. DNS. Top 10 consultas

Figura 75: C.M. DNS. Protocolos

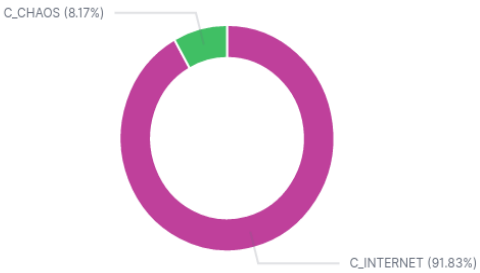


Figura 76: C.M. DNS. Clases consulta

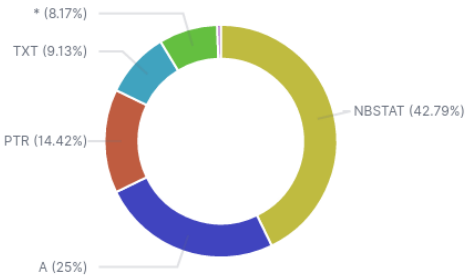


Figura 77: C.M. DNS. Tipos de consulta

4.1.4 Cuadro de mandos DPD

Este cuadro de mandos aporta información acerca del tráfico que no cumple con el protocolo estandarizado por el que se establece la comunicación. Con este cuadro de mandos no se pretende mostrar información de ataques específicos, sino comportamientos extraños que provienen de orígenes sospechosos con el objetivo de analizar comportamientos, que si bien, pueden parecer «errores de software» pueden subvertir otro tipo de intenciones como intentar explotar algún fallo o vulnerabilidad.

id.orig_h: Descending	Count	Count percentages
103.10.208.230	2	10%
164.68.112.178	2	10%
2.50.51.90	1	5%
12.111.128.210	1	5%
24.41.166.50	1	5%
24.93.200.253	1	5%
92.118.161.25	1	5%
96.56.160.50	1	5%
96.83.14.61	1	5%
122.52.129.69	1	5%
Otros	8	40%
	20	

Figura 78: C.M. DPD. Top 10 IP origen amenazas

failure_reason.keyword: Descending	Count	Count percentages
not a http request line	27	81.8%
Binpac exception: binpac exception: string mismatch at /home/abuild/rpmbuild/BUILD/zeek-3.0.5/src/analyzer/protocol /imap/imap-protocol.pac:18: expected pattern: "[[:alnum:][:punct:]]+" actual data: "\x16\x03\x02\x01o\x01"	6	18.2%
	33	

Figura 79: C.M. DPD. Motivos del error

4.1.5 Cuadro de mandos FILES

Este cuadro de mandos aporta información acerca de los ficheros que se han descargado. Se ha generado dos visualizaciones adicionales, la primera muestra la proporción del tipo de tráfico por el que se han descargado los ficheros y la segunda enumera en una tabla los tipos MIME de los ficheros.

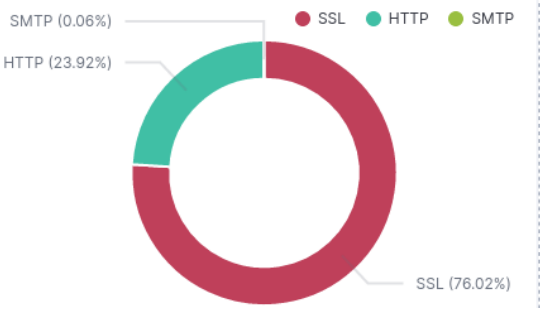


Figura 80: C.M. FILES. Fuentes del tráfico

Tipo MIME	Count	Count percentages
application/x-x509-ca-cert	1,192	38.5%
application/x-x509-user-cert	1,160	37.5%
application/xml	645	20.8%
text/plain	66	2.1%
application/x-bzip2	14	0.5%
text/html	14	0.5%
application/soap+xml	3	0.1%
	3,094	

Figura 81: C.M. FILES. Tipos MIME

4.1.6 Cuadro de mandos HTTP

Este cuadro de mandos aporta información acerca de las conexiones de tipo HTTP que se han producido al «honeypot». Se ha generado tres visualizaciones adicionales, la primera enumera el campo HELO, la segunda el campo MAIL FROM y la tercera si la conexión se ha intentado establecer mediante TLS.

uri.keyword: Descending	Count	Count percentages
/	873	64%
www.so.com:443	44	3.2%
www.baidu.com:443	41	3%
www.sogou.com:443	40	2.9%
/robots.txt	21	1.5%
/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php	16	1.2%
/cgi-bin/mainfunction.cgi	11	0.8%
/cgi-bin/mainfunction.cgi?action=loginkeyPath=' /bin/sh${IFS}-c${IFS}'cd${IFS}/tmp;${IFS}rm${IFS}-rf${IFS}arm7;${IFS}busybox${IFS}wget${IFS}http://19ce033f.ngrok.io/arm7;${IFS}chmod${IFS}777${IFS}arm7;${IFS}./arm7' '&loginUser=a&loginPwd=a	11	0.8%
/?XDEBUG_SESSION_START=phpstorm	8	0.6%
/?a=fetch&content=<php>die(@md5(HelloThinkCMF))</php>	8	0.6%

Figura 82: C.M. HTTP. Top 10 URI consultadas amenazas

Top 10 agentes de usuario amenazas

user_agent.keyword: Descending	Count	Count percentages
Mozilla/5.0 (Windows NT 5.1; rv:9.0.1) Gecko/20100101 Firefox/9.0.1	711	55.3%
ZmEu	137	10.7%
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0; Trident/5.0)	125	9.7%
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36	56	4.4%
Mozilla/5.0 (Windows; U; Windows NT 6.0;en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6)	32	2.5%
python-requests/2.9.1	30	2.3%
Mozilla/5.0 (compatible; Nimbostratus-Bot/v1.3.2; http://cloudsystemnetworks.com)	23	1.8%
Mozilla/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm)	20	1.6%
Mozilla/5.0 zgrab/0.x	18	1.4%
Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0	16	1.2%
Otros	117	9.1%
	1,285	

Figura 83: C.M. HTTP. Top 10 agentes de usuario amenazas

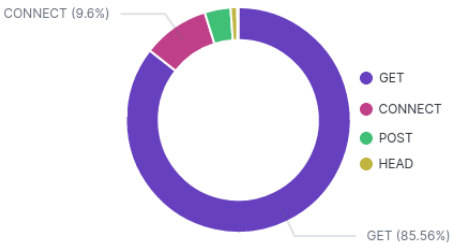


Figura 84: C.M. HTTP. Métodos peticiones amenazas

4.1.7 Cuadro de mandos KERBEROS

Este cuadro de mandos aporta información acerca de las conexiones de tipo Kerberos que se han producido al «honeypot». Este cuadro de mandos no tiene ninguna visualización propia adicional.

4.1.8 Cuadro de mandos NTP

Este cuadro de mandos aporta información acerca de las conexiones de tipo NTP que se han producido al «honeypot». Este cuadro de mandos no tiene ninguna visualización propia adicional.

4.1.9 Cuadro de mandos RDP

Este cuadro de mandos aporta información acerca de las conexiones de tipo RDP que se han producido al «honeypot». Este cuadro de mandos no tiene ninguna visualización propia adicional.

4.1.10 Cuadro de mandos SIP

Este cuadro de mandos aporta información acerca de las conexiones de tipo SIP que se han producido al «honeypot». Se ha generado una visualización adicional de tipo tabla que enumera el origen («From») y el destino («To») de las peticiones de sesión que han intentado establecerse.

De	Para	Count	Count percentages
"sipvicious"<sip:100@1.1.1.1>	"sipvicious"<sip:100@1.1.1.1>	1,040	80.2%
"sipvicious"<sip:90501@1.1.1.1>	"sipvicious"<sip:90501@1.1.1.1>	60	4.6%
"ppobx"<sip:100@1.1.1.1>	"ppobx"<sip:100@1.1.1.1>	51	3.9%
"17631103131" <sip:17631103131@176.31.103.131>	"01146441408560" <sip:01146441408560@176.31.103.131>	16	1.2%
<sip:-@176.31.103.131>	<sip:-@176.31.103.131>	16	1.2%
Otros	<sip:b@b>	14	1.1%
Otros	"Bob"<sip:100@46.166.160.136>	11	0.8%
Otros	"AmooT"<sip:100@1.1.1.1>	9	0.7%
Otros	"FPBX"<sip:100@1.1.1.1>	9	0.7%
Otros	"eyeBeam"<sip:100@1.1.1.1>	9	0.7%
Otros	Otros	62	4.8%
		1,297	

Figura 85: C.M. SIP. Top 10 Peticiones From/To amenazas

4.1.11 Cuadro de mandos SMTP

Este cuadro de mandos aporta información acerca de las conexiones de tipo SMTP que se han producido al «honeypot». Se ha generado tres visualizaciones adicionales, la primera enumera el campo HELO, la segunda el campo MAIL FROM y la tercera si la conexión se ha intentado establecer mediante TLS.

helo.keyword: Descending	Count	Count percentages
WIN-CLJ1B0GQ6JP	2	40%
WIN-344VU98D3RU	1	20%
WIN-G7CPHCGK247	1	20%
WIN-JG1E0O7FSBS	1	20%
	5	

Figura 86: C.M. SMTP. HELO amenazas

Figura 87: C.M. SMTP. Conexión TLS

MAIL FROM

mailfrom.keyword: Descending	Count	Count percentages
spameri@tiscali.it	5	100%
	5	

Figura 88: C.M. SMTP. MAIL FROM

4.1.12 Cuadro de mandos SNMP

Este cuadro de mandos aporta información acerca de las conexiones de tipo SNMP que se han producido al «honeypot». Este cuadro de mandos no tiene ninguna visualización propia adicional.

4.1.13 Cuadro de mandos SSH

Este cuadro de mandos aporta información acerca de las conexiones de tipo SSH que se han producido al «honeypot». Se ha generado una visualización adicional de tipo tabla que permite identificar el tipo de cliente SSH que el peticionario ha utilizado para la petición de conexión.

client.keyword: Descending	Count	Count percentages
SSH-2.0-libssh-0.6.3	93,712	72.7%
SSH-2.0-PUTTY	14,295	11.1%
SSH-2.0-libssh2_1.4.3	11,931	9.2%
SSH-2.0-PuTTY	3,977	3.1%
SSH-2.0-libssh-0.1	1,776	1.4%
SSH-2.0-libssh-0.2	648	0.5%
SSH-2.0-Go	605	0.5%
SSH-2.0-PuTTY_Release_0.63	432	0.3%
SSH-2.0-libssh2_1.7.0	399	0.3%
SSH-2.0-libssh2_1.8.0	291	0.2%
Otros	924	0.7%
	128,990	

Figura 89: C.M. SSH . Tipos de cliente amenazas

4.1.14 Cuadro de mandos SSL

Este cuadro de mandos aporta información acerca de las conexiones de tipo HTTPS que se han producido al «honeypot». Se ha generado una visualización adicional de tipo tabla que permite identificar los nombres de servidor (SNI) que el peticionario ha solicitando.

server_name.keyword: Descending	Count	Count percentages
176.31.103.131	49	48.5%
jsant.fr	22	21.8%
gateway.jsant.fr	5	5%
mail9.jsant.fr	5	5%
couille.jsant.fr	4	4%
dashboard.dev.jsant.fr	4	4%
ns391307.ip-176-31-103.eu	4	4%
www.google-analytics.com	3	3%
176.31.103.131:443	2	2%
gestoertabergeil.dyn-vpn.de	2	2%
Otros	1	1%
	101	

Figura 90: C.M. SSL. Nombre de servidor

4.1.15 Cuadro de mandos SYSLOG

Este cuadro de mandos aporta información acerca de las conexiones de tipo syslog que se han producido al «honeypot». Este cuadro de mandos no tiene ninguna visualización propia adicional.

4.1.16 Cuadro de mandos TUNNEL

Este cuadro de mandos aporta información acerca de las conexiones de tipo túnel que se han producido al «honeypot». Se ha generado dos visualizaciones adicionales, la primera identifica el tipo de túnel que se ha intentado establecer y la segunda el tipo de actividad que ha intentado realizar el peticionario.

Figura 91: C.M. TUNNEL. Tipo de túnel amenazas

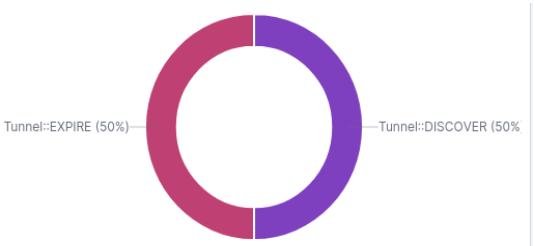


Figura 92: C.M. TUNNEL. Actividades túnel amenazas

4.1.17 Cuadro de mandos WEIRD

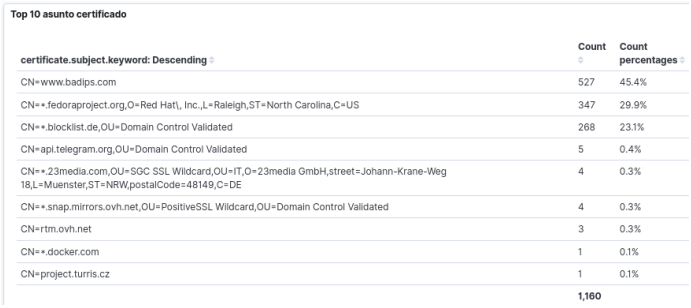
Este cuadro de mandos aporta información acerca de las conexiones que han realizado «acciones extrañas» que el motor de Zeek no ha podido clasificar. Se ha generado una visualización adicional de tipo tabla que permite enumerar los tipos de anomalías que ha capturado Zeek.

Anomalía	Count	Count percentages
data_before_established	156,377	52.2%
inappropriate_FIN	127,421	42.5%
bad_TCP_checksum	6,822	2.3%
active_connection_reuse	3,779	1.3%
TCP_seq_underflow_or_misorder	2,294	0.8%
premature_connection_reuse	1,699	0.6%
data_after_reset	668	0.2%
SYN_seq_jump	242	0.1%
bad_HTTP_request	219	0.1%
FIN_advanced_last_seq	81	0%
Otros	118	0%
	299,720	

Figura 93: C.M. WEIRD. Anomalías

4.1.18 Cuadro de mandos X509

Es último cuadro es también atípico en relación al resto. Muestra información sobre los certificados digitales de tipo X.509 que se han analizado durante las comunicaciones cifradas que se han establecido. Dado que no se dispone de una fuente de amenazas especifica para certificados, no es posible detectar directamente una posible amenaza, pero sí que sirve como referencia para intentar evaluar posible tráfico anómalo mediante la revisión de la información de los certificados capturados.



Top 10 asunto certificado

certificate.subject.keyword: Descending	Count	Count percentages
CN=www.badips.com	527	45.4%
CN=*.fedoraproject.org,O=Red Hat\, Inc.,L=Raleigh,ST=North Carolina,C=US	347	29.9%
CN=*.blocklist.de,OU=Domain Control Validated	268	23.1%
CN=api.telegram.org,OU=Domain Control Validated	5	0.4%
CN=*.23media.com,OU=SGC SSL Wildcard,OU=IT,O=23media GmbH,street=Johann-Krane-Weg 18,L=Muenster,ST=NRW,postalCode=48149,C=DE	4	0.3%
CN=*.snap.mirrors.ovh.net,OU=PositiveSSL Wildcard,OU=Domain Control Validated	4	0.3%
CN=rtm.ovh.net	3	0.3%
CN=*.docker.com	1	0.1%
CN=project.turris.cz	1	0.1%
	1,160	

Figura 94: C.M. X509. Top 10 asunto certificado

san.dns.keyword: Descending	Count	Count percentages
badips.com	527	22.8%
www.badips.com	527	22.8%
*.fedoraproject.org	347	15%
fedoraproject.org	347	15%
*.blocklist.de	268	11.6%
blocklist.de	268	11.6%
api.telegram.org	5	0.2%
www.api.telegram.org	5	0.2%
*.23media.com	4	0.2%
*.snap.mirrors.ovh.net	4	0.2%
Otros	5	0.2%
	2,307	

Figura 95: C.M. X509. Top 10 SAN.DNS

certificate.issuer.keyword: Descending	Count	Count percentages
CN=RapidSSL RSA CA 2018,OU=www.digicert.com,O=DigiCert Inc,C=US	527	45.4%
CN=DigiCert SHA2 High Assurance Server CA,OU=www.digicert.com,O=DigiCert Inc,C=US	347	29.9%
CN=AlphaSSL CA - SHA256 - G2,O=GlobalSign nv-sa,C=BE	268	23.1%
CN=Sectigo RSA Domain Validation Secure Server CA,O=Sectigo Limited,L=Salford,ST=Greater Manchester,C=GB	7	0.6%
CN=Go Daddy Secure Certificate Authority - G2,OU=http://certs.godaddy.com/repository/,O=GoDaddy.com\, Inc.,L=Scottsdale,ST=Arizona,C=US	5	0.4%
CN=COMODO RSA Organization Validation Secure Server CA,O=COMODO CA Limited,L=Salford,ST=Greater Manchester,C=GB	4	0.3%
CN=Amazon,OU=Server CA 1B,O=Amazon,C=US	1	0.1%
CN=Let's Encrypt Authority X3,O=Let's Encrypt,C=US	1	0.1%
	1,160	

Figura 96: C.M. X509. Top 10 emisor certificados

4.2 Evaluación de los datos

4.2.1 Evaluación de datos de las conexiones

Las conexiones se han producido desde cualquier parte del mundo aunque en casi todos los tipos de tráfico destacan casi siempre los Estados Unidos de América, China, Rusia y Francia como focos principales aunque a nivel global figure España debido a un evento que se comenta más adelante en el texto.

Figura 97: Eval. datos. Geolocalización amenazas

También se ha verificado que las conexiones sospechosas se produce de forma constante a lo largo del tiempo y sobre una variedad de servicios muy amplia.

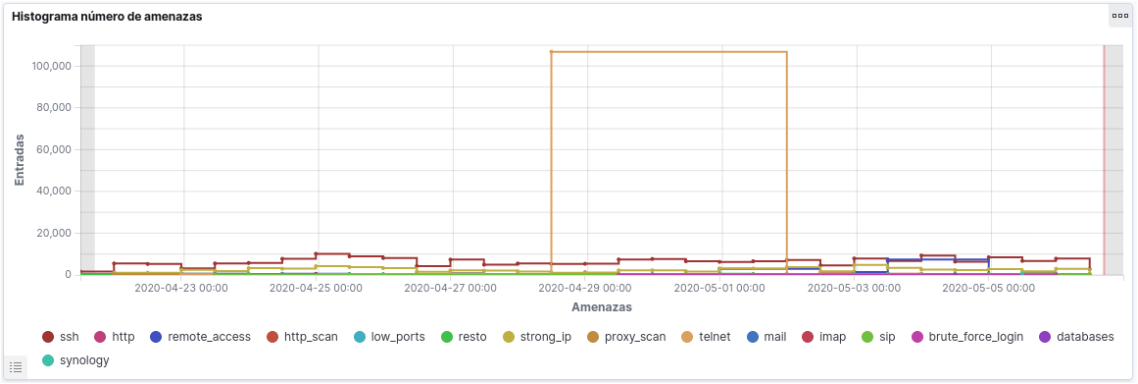


Figura 98: Eval. Datos. Histograma número amenazas

Existe una anomalía en los datos que se corresponde con un ataque dirigido de tipo telnet desde las instalaciones del un proveedor de servicios de San Sebastián de los Reyes (España), que se produjo entre el 28 de abril y el 2 de mayo de 2020 que generó 106693 peticiones de conexión que han deformado los datos tal y como se puede observar en esta gráfica.

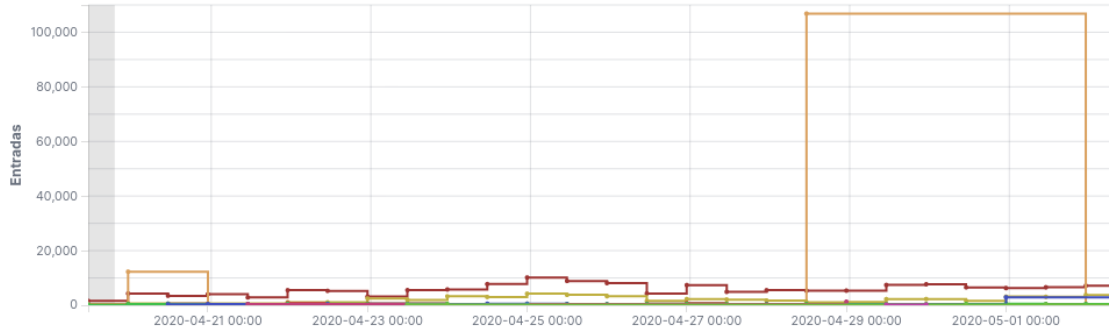


Figura 99: Eval. Datos. Anomalía ataque telnet desde España

Por otra parte, sobre el total de tráfico analizado tan solo se han detectado un 9.3% de amenazas, pero esta cifra está originada por la calidad de las fuentes de inteligencia de amenazas utilizadas puesto que prácticamente todo el tráfico debería ser considerado como amenaza puesto que la plataforma no ofrece servicios reales sino un «honeypot».

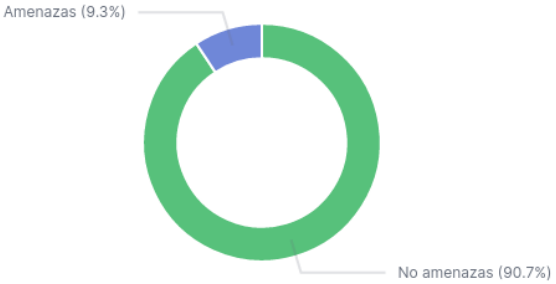


Figura 100: Eval. datos. Ratio de amenazas

Sobre el tipo de peticiones de conexión realizadas, destaca sobre todos los demás el tráfico de tipo SSH que supone el 96.61% del etiquetado como de origen sospechoso y un 30.66% del no marcado como sospechoso.

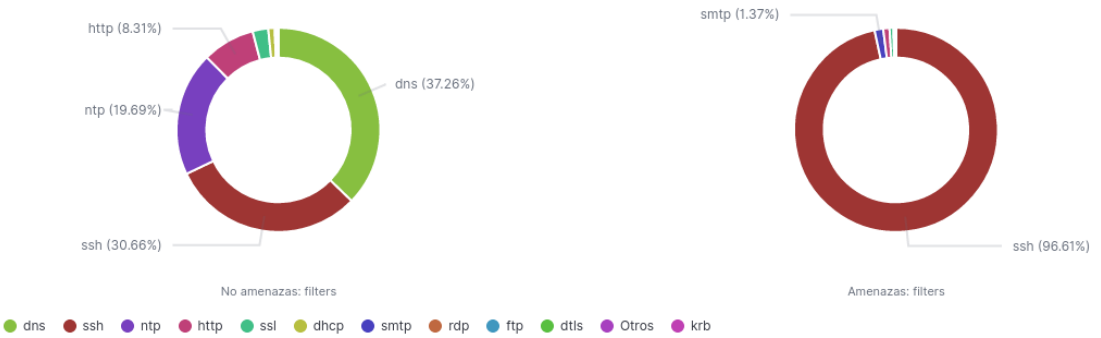


Figura 101: Eval. datos. Tipos de peticiones conexión por servicio

En relación a las fuentes de información de inteligencia, las fuentes Blocklist.de y Turris destacan sobre el resto de forma apabullante dado que entre las dos suman el 98.18% del etiquetado.

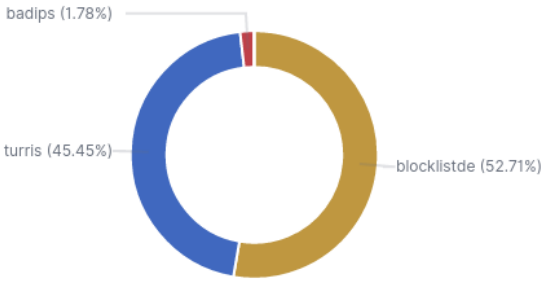


Figura 102: Eval. datos. Ratio de fuentes de amenazas

Sobre el tráfico etiquetado es necesario reseñar que el 41,75% del total se corresponde con una amenaza de tipo SSH, seguida de telnet con un 28,57%. Le siguen los orígenes etiquetados como «strong_ip» con un 14,99%, «acceso remoto» con un 4.3% y «http» con un 2.79%.

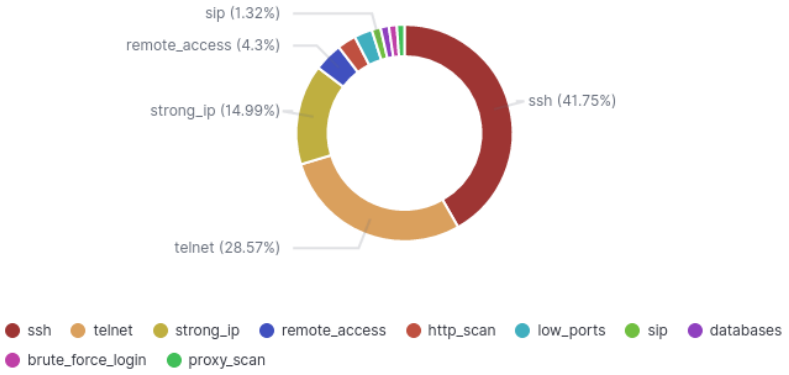


Figura 103: Eval. datos. Ratios tipos de amenazas

Por último cabe destacar la enorme cantidad de información recibida durante la fase de recogida de datos de esta plataforma. Se han recibido 1,27 terabytes de información y se han enviado 2,76 gigabytes de datos.

4.2.2 Evaluación de datos del tráfico DHCP

En lo referente al tráfico DHCP, no se han obtenidos datos específicos sobre inteligencia de amenazas dado el carácter interno del tráfico, por lo que no se han podido obtener conclusiones relevantes.

4.2.3 Evaluación de datos del tráfico DNS

En lo referente al tráfico DNS, el nivel de tráfico obtenido, una vez excluido el propio del servidor anfitrión, no resulta apenas relevante ya que supone un total de 425 peticiones resultado tan solo 228 peticiones detectadas como maliciosas, lo que supone solo un 53.65% de detección.

En relación al origen geográfico de las peticiones maliciosas detectadas, el tráfico procedente de Estados Unidos de América supone más del 68%, seguido de Rusia (7.89%) y Países Bajos (4.39%).

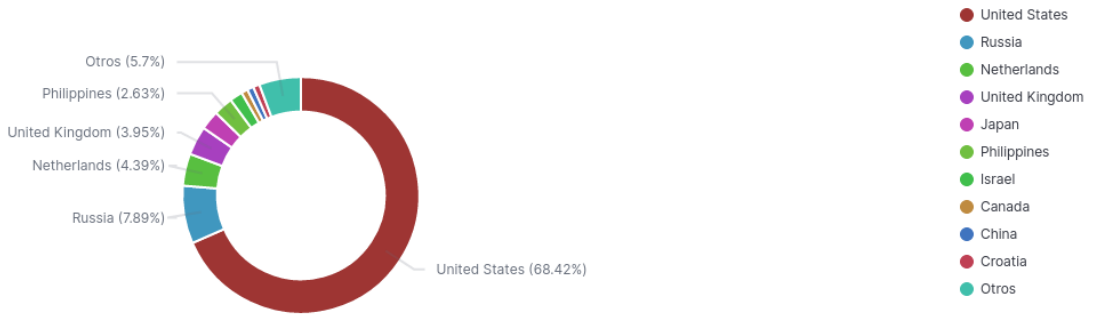


Figura 104: Eval. datos. Top 10 países origen amenazas DNS

En este tipo de trafico, la detección de las amenazas fue realizada casi en su totalidad por la fuente Turris con un 88.6% de las detecciones, seguida de BadIPs con solo un 8.77% de las detecciones.

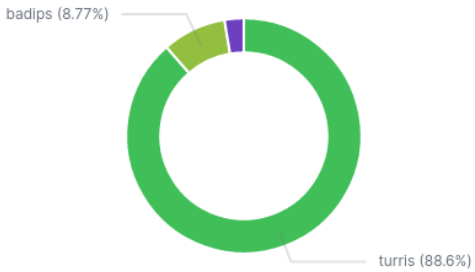


Figura 105: Eval. datos. Fuentes de amenazas DNS

La principal conclusión que se puede obtener es que dado que este servidor no es un servidor DNS real, no deberíamos tener ninguna clase de peticiones de este tipo, por lo que todas deben ser peticiones malintencionadas. Esto nos indica que las fuentes de detección de amenazas no están teniendo mucho éxito a la hora de detectarlo.

4.2.4 Evaluación de datos del tráfico DPD

En lo referente al tráfico que viola el protocolo de comunicación (DPD), si bien es cierto que no se ha producido mucho tráfico, si que es relevante que cerca del 67% de este tráfico fue etiquetado como una amenaza.

Figura 106: Eval. datos. Porcentaje amenazas DPD

En relación al origen de este tipo de trafico se encuentra repartido por bastantes localizaciones, destacando Estados Unidos de América y Alemania con más del 55% conjunto.

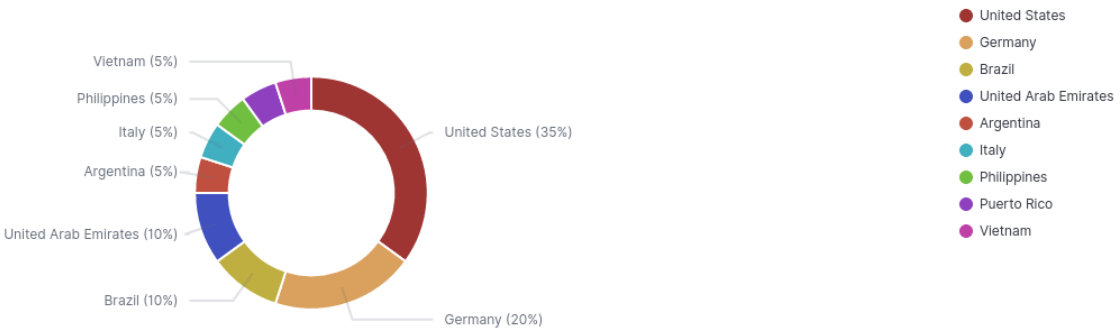


Figura 107: Eval. datos. Top 10 países amenazas DPD

En relación al etiquetado del tráfico como amenaza, la fuente Turris ha sido la que ha aportado toda la información.

4.2.5 Evaluación de datos del tráfico de ficheros

En lo referente al análisis de ficheros (FILES), no se ha obtenido de forma natural ninguna detección. Este resultado es el esperado puesto que el servidor anfitrión tan solo ha realizado la descarga de actualizaciones y de los componentes software que son parte de la solución y son de orígenes conocidos y confiables. Por su parte el «honeypot» utilizado no tiene una implementación completa por lo que realmente ningún atacante sería capaz de subir ficheros.

4.2.6 Evaluación de datos del tráfico HTTP

En lo referente al análisis del tráfico web sin cifrado (HTTP), se han realizado un total de 3390 conexiones de las que 1107 se han etiquetado como maliciosas, esto supone un 32.65% del total.

Figura 108: Eval. datos. Porcentaje amenazas HTTP

En relación a la ubicación geográfica, para este tipo de tráfico los países de origen más recurrentes son Rusia con el 45.39%, en segundo lugar Países

Bajos con el 17.26% y en tercer lugar China con el 16.89% de las peticiones de conexión.

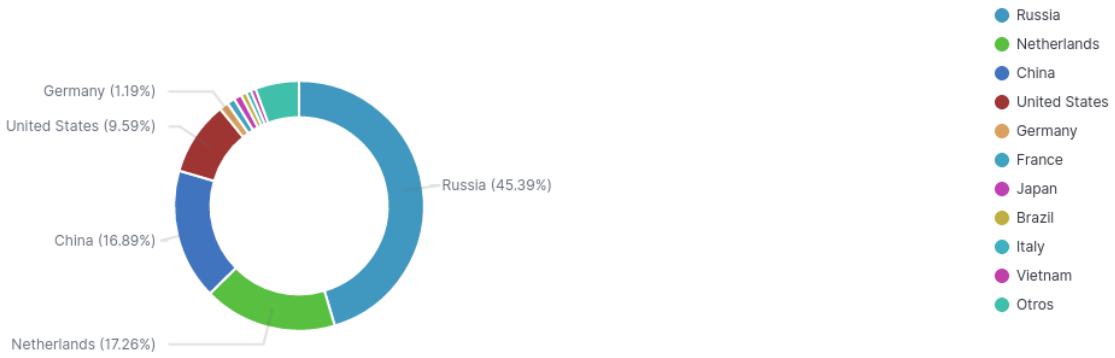


Figura 109: Eval. datos. Top 10 países origen amenazas HTTP

En relación a la detección de los orígenes sospechosos, nuevamente la fuente Turris es capaz de detectar el 91.96% seguido de BadIPs con un 6,5%.

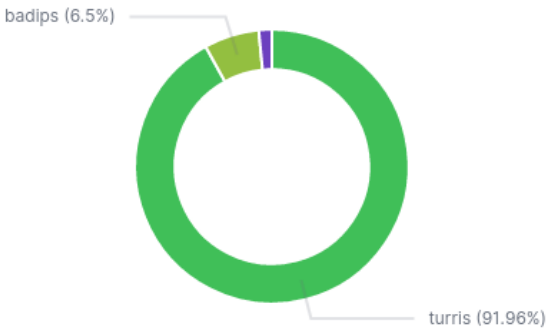


Figura 110: Eval. datos. Fuentes amenazas HTTP

La principal observación que se ha realizado es que nuevamente las fuentes de inteligencia de amenazas no ha sido capaz de etiquetar de forma mayoritaria el tráfico como malicioso.

4.2.7 Evaluación de datos del tráfico Kerberos

Apenas se han recibido cuatro conexiones de este tipo y tres de ellas fueron catalogadas como maliciosas.

En relación a la ubicación geográfica, las tres peticiones se reparten a partes iguales entre Estados Unidos de América, Países Bajos y Vietnam con un 33.33% cada una.

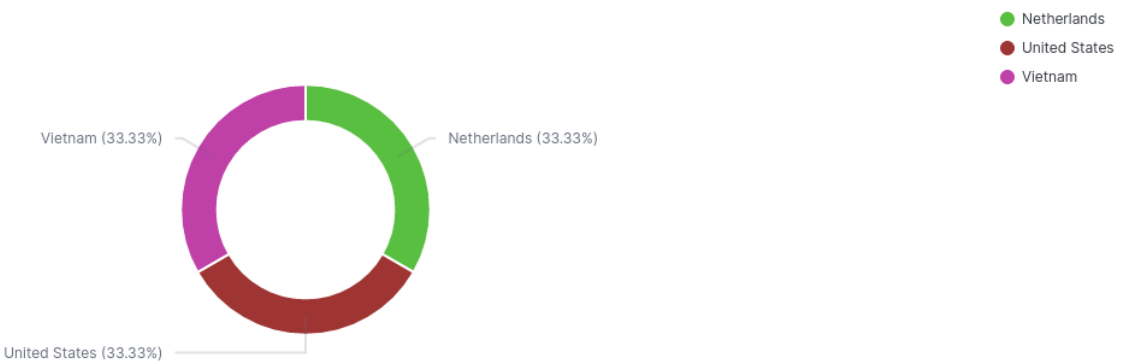


Figura 111: Eval. datos. Top 10 países origen amenazas Kerberos

En relación a la detección de los orígenes sospechosos, dos de las tres detecciones fueron realizadas por la fuente BadIPs y la tercera por Blocklist.de.

Figura 112: Eval. datos. Fuentes amenazas Kerberos

Con tan pocos datos no es posible realizar una correcta evaluación de la plataforma en relación a este tipo de tráfico.

4.2.8 Evaluación de datos del tráfico NTP

En relación a este tipo de tráfico tan solo se han recibido 52 conexiones, etiquetándose 34 como posible amenaza, lo que supone un porcentaje del 65.38%.

Figura 113: Eval. datos. Porcentaje amenazas NTP

En relación a la ubicación geográfica, los Estados Unidos de América suponen el 85.29% del trafico etiquetado como malicioso, seguido de China y Países Bajos, ambos con un 2.94%.

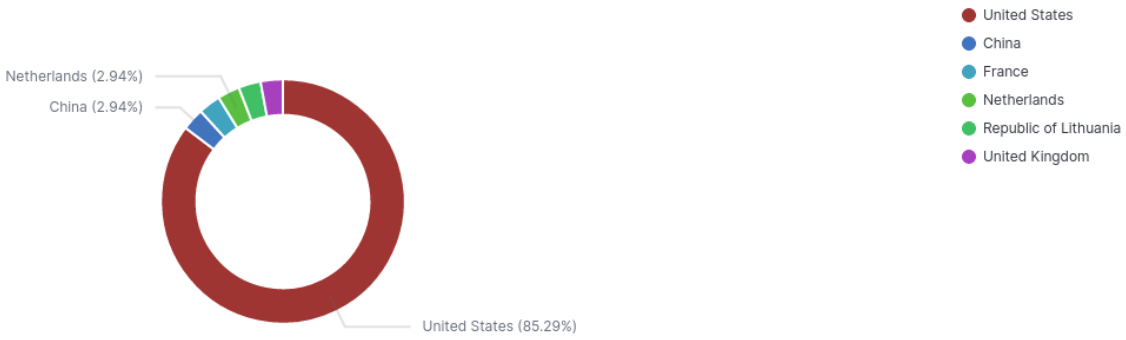


Figura 114: Eval. datos. Top 10 países origen amenazas

En relación a la detección de los orígenes sospechosos, nuevamente la fuente Turris es capaz de detectar el 91.18% seguido de BadIPs con un 8,82%.

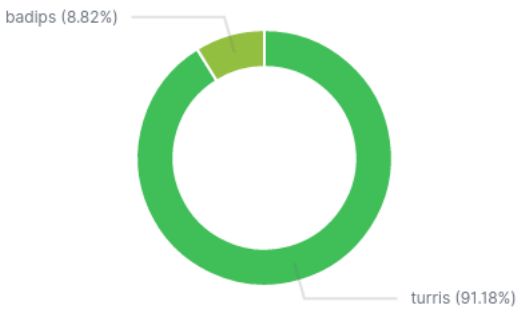


Figura 115: Eval. datos. Fuentes amenazas NTP

No se ha obtenido ninguna conclusión especifica respecto a este tipo de tráfico más allá de lo ya conocido.

4.2.9 Evaluación de datos del tráfico RDP

En relación a este tipo de tráfico tan solo se han recibido un total de 34935 peticiones, etiquetándose como sospechosas tan solo 139 sobre el total, lo que hace apenas 0.4% de éxito en el marcado.

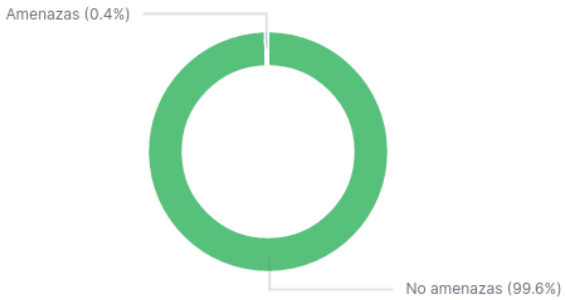


Figura 116: Eval. datos. Porcentaje amenazas RDP

En relación a la ubicación geográfica, los Estados Unidos de América suponen el 82.73% del trafico etiquetado como malicioso, seguido de Países Bajos y Francia, con un 5.04% y un 2,88% respectivamente.

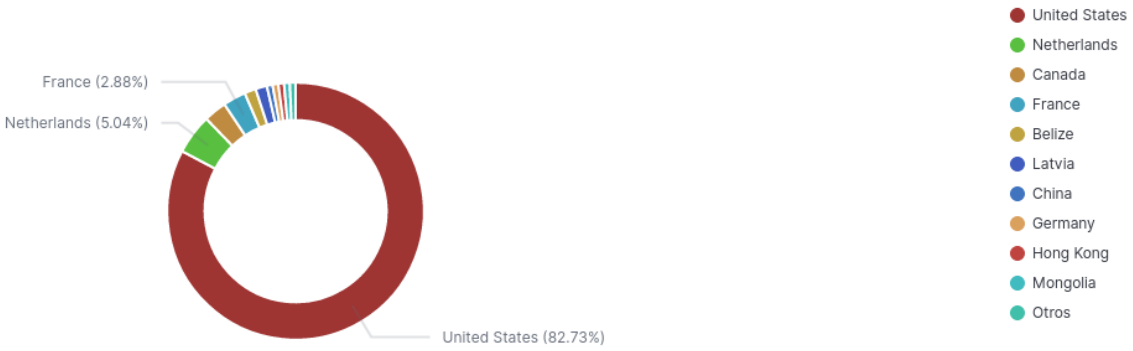


Figura 117: Eval. datos. Top 10 países origen amenazas RDP

En relación a la detección de los orígenes sospechosos, nuevamente la fuente Turris es capaz de detectar el 87.05% seguido de BadIPs con un 8,63%.

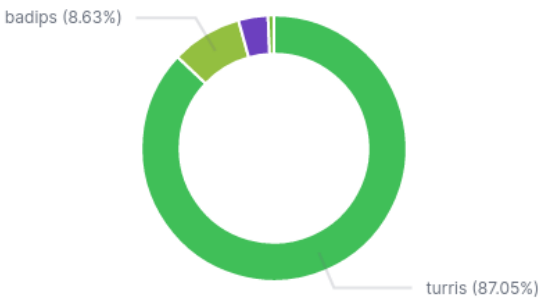


Figura 118: Eval. datos. Fuentes amenazas RDP

Como principal conclusión cabe destacar el mínimo éxito que se ha tenido a la hora de etiquetar el tráfico como sospechoso. En este caso resulta especialmente preocupante porque los ataques a RDP son una de las principales vías utilizadas para acceder de forma ilícita a los servidores con sistemas operativos Microsoft Windows.

4.2.10 Evaluación de datos del tráfico SIP

En relación a este tipo de tráfico tan solo se han recibido un total de 1870 conexiones, siendo etiquetadas como maliciosas 1270 del total, lo que supone una media de 79,94 ataques por día y una una tasa de éxito en la detección del 68,4%.

Figura 119: Eval. datos. Porcentaje amenazas SIP

En relación a la ubicación geográfica, Países Bajos acumulan el 47,87% de las conexiones sospechosas, seguidos de Francia y Estados Unidos de América, ambos con un 17,72%.

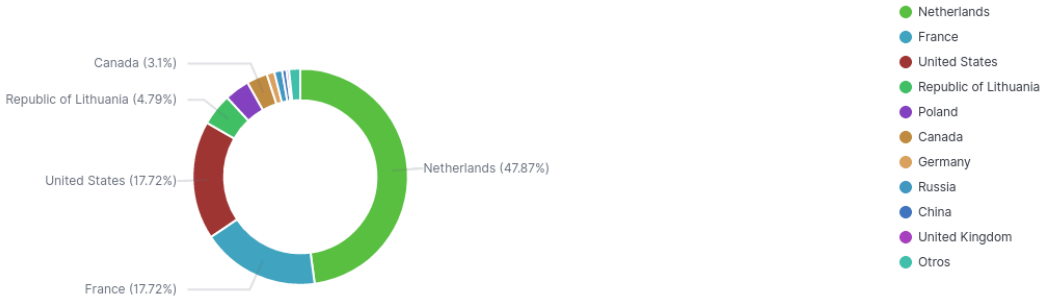


Figura 120: Eval. datos. Top 10 países origen amenazas SIP

En relación a la detección de los orígenes sospechosos, nuevamente la fuente Turris es capaz de detectar el 93,98% seguido de Blocklist.de con un 5,79%.

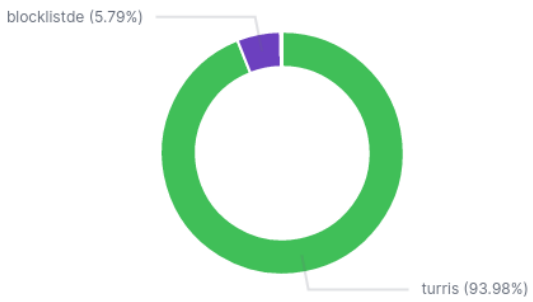


Figura 121: Eval. datos. Fuentes amenazas SIP

4.2.11 Evaluación de datos del tráfico SMTP

En relación a este tipo de tráfico tan solo se han recibido doce peticiones de conexión, resultando cinco de ellas detectadas como amenaza, lo que nos da una media de menos de un ataque por día y una tasa de éxito en la detección del 41,67%.

Figura 122: Eval. datos. Porcentaje amenazas SMTP

En relación a la ubicación geográfica, los datos están divididos justo a la mitad entre los Estados Unidos de América y China.

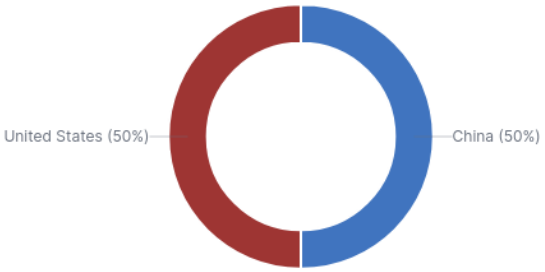


Figura 123: Eval. datos. Top 10 países origen amenazas SMTP

En relación a la detección de los orígenes sospechosos, la fuente Turris acumula el 80% de las detecciones seguido de Blocklist.de con el 20% restante.

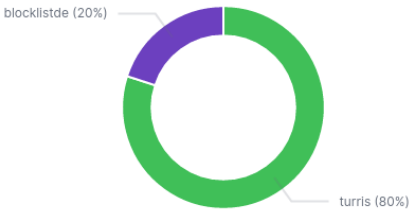


Figura 124: Eval. datos. Fuentes amenazas SMTP

No se ha considerado que se pueda obtener ninguna lección concluyente de este tipo de tráfico dado el escaso volumen de datos disponible.

4.2.12 Evaluación de datos del tráfico SNMP

En relación a este tipo de tráfico se han recibido tan solo se han obtenido 86 peticiones de conexión de las que 47 fueron etiquetadas como sospechosas, lo que hace una media de 3,13 diarias con una tasa de éxito en el etiquetado del 54,65%.

Figura 125: Eval. datos. Fuentes amenazas SNMP

En relación a la ubicación geográfica, los Estados Unidos de América supone el 85,11% del trafico malicioso, seguido de Rusia con un 6,38% y como tercero Canadá con un 2,13% del total.

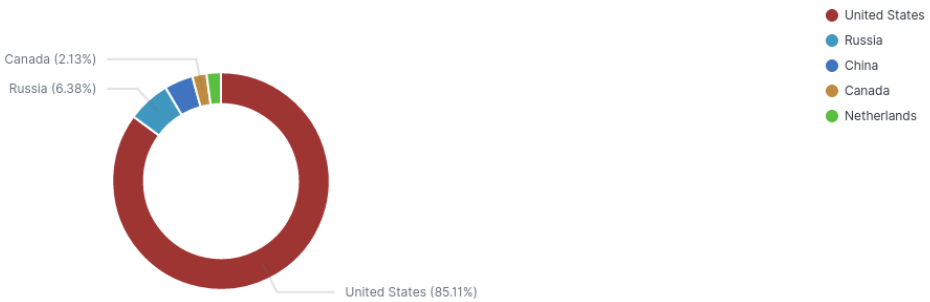


Figura 126: Eval. datos. Top 10 países origen amenazas SNMP

En relación a la detección de los orígenes sospechosos, la fuente Turris ha etiquetado el 91,49% del tráfico sospechoso, seguido de BadIPs con un 6,38% y por último Blocklist.de con un 2,13%.

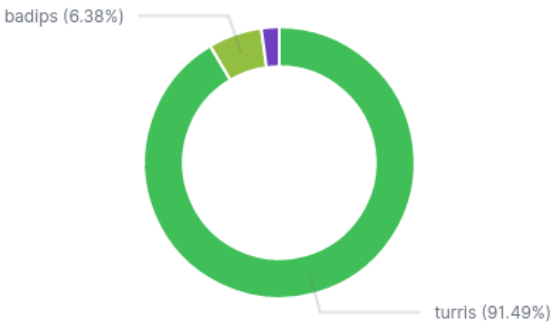


Figura 127: Eval. datos. Fuentes amenazas SNMP

No se ha considerado que se pueda obtener ninguna lección concluyente de este tipo de tráfico dado el escaso volumen de datos disponible.

4.2.13 Evaluación de datos del tráfico SSH

En relación a este tipo de tráfico se han recibido 121.285 peticiones de conexión de las que se han etiquetado como sospechosas 113083 peticiones, lo que hace una media de 7067,69 al día y una tasa de éxito en el etiquetado del 93,24% del total.

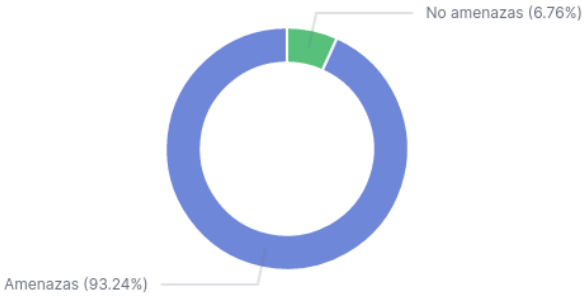


Figura 128: Eval. datos. Fuentes amenazas SSH

En relación a la ubicación geográfica, China supone el 39,25% del trafico malicioso, seguido de Francia con el 10,88% y como tercero Estados Unidos de América con el 9,77% del total.

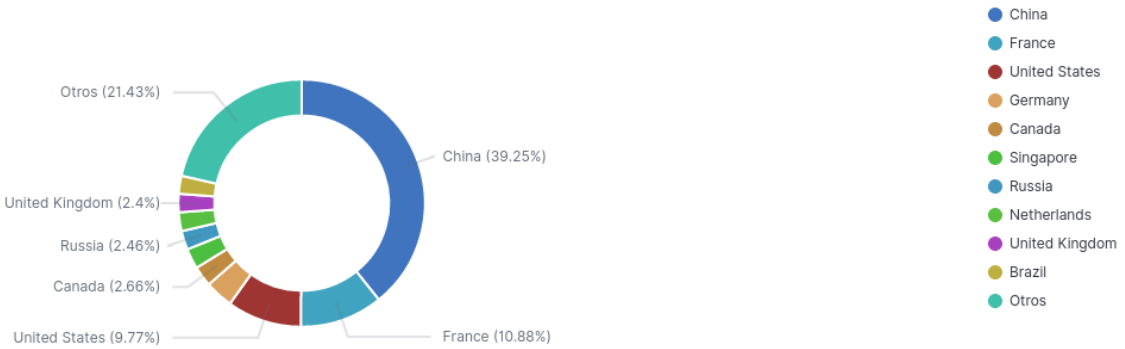


Figura 129: Eval. datos. Top 10 países origen amenazas SSH

En relación a la detección de los orígenes sospechosos, la fuente Blocklist.de acumula el 97,84% de los etiquetados, seguido por BadIPs con el 1.24%, Turris con el 0,76% y por último DanMe.uk con el 0,16% restante.

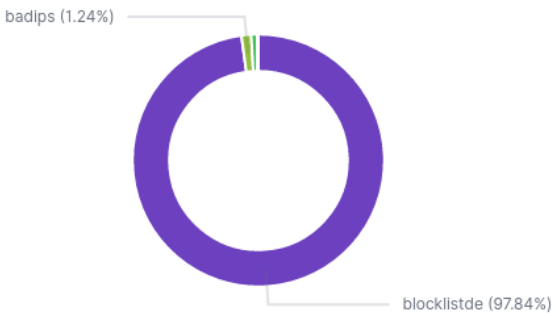


Figura 130: Eval. datos. Fuentes amenazas SSH

Como ya se sospechaba el tráfico SSH es el principal punto de acceso que han intentado utilizar los atacantes. En este caso destaca notablemente la capacidad de detección de amenazas obtenida por las fuentes de inteligencia.

4.2.14 Evaluación de datos del tráfico SSL

En relación a este tipo de tráfico tan solo se han recibido 2026 peticiones de conexión de las que 684 se han etiquetado como sospechosas, lo que hace una media de 45.6 por día y una tasa de éxito en el etiquetado del 33.76%.

Figura 131: Eval. datos. Fuentes amenazas SSL

En relación a la ubicación geográfica, Rusia supone el 54,97% del trafico malicioso, seguido de Estados Unidos de América con el 32,31% y como tercero Países Bajos con el 4,39% del total.

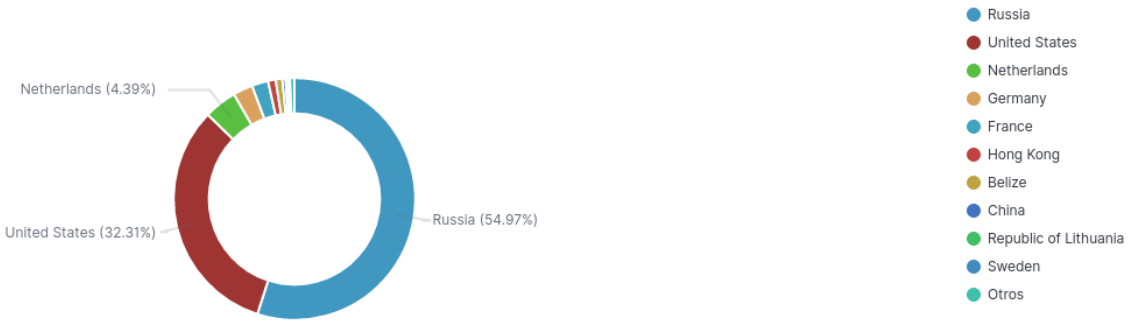


Figura 132: Eval. datos. Top 10 países origen amenazas SSL

En relación a la detección de los orígenes sospechosos, la fuente Turris acumula el 81,14% de los etiquetados, seguido de BadIPs con el 12,13%, Blocklist.de con el 6.58% y por último DanMe.uk con el 0.15%.

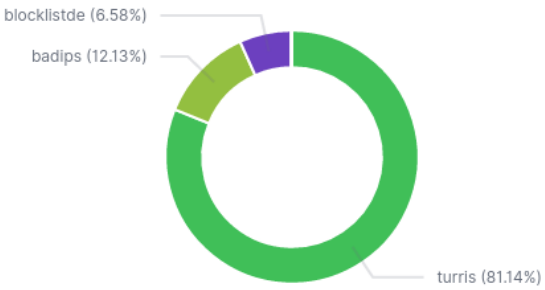


Figura 133: Eval. datos. Fuentes amenazas SSL

Como principal observación cabe destacar la poca cantidad de peticiones de este tipo que se han producido. Nuevamente las fuentes de inteligencia de amenazas no han sido capaces de etiquetar gran parte de las conexiones.

4.2.15 Evaluación de datos del tráfico de syslog

En relación a este tipo de tráfico tan solo se han recibido una conexión desde Países Bajos que no fue detectada como maliciosa. No se puede obtener ninguna conclusión acerca de este suceso, de hecho no se está seguro si es

un intento de escaneo o un error de la detección. Revisando el mensaje completo se ha comprobado que parece una petición HTTP pero fue remitida al puerto 514 mediante protocolo UDP.

#	id.resp_p	514
t	log_type	syslog
t	message	GET / HTTP/1.1 Host: www
t	proto	udp
t	severity	UNSPECIFIED

Figura 134: Eval. datos. Mensaje syslog

4.2.16 Evaluación de datos del tráfico de tipo túnel

En relación a este tipo de tráfico tan solo se han recibido 2 peticiones de conexión de las que 18 fueron etiquetadas como sospechosas, lo que hace una media de 2,57 conexiones por día y una tasa de éxito en el etiquetado del 21,95% del total.

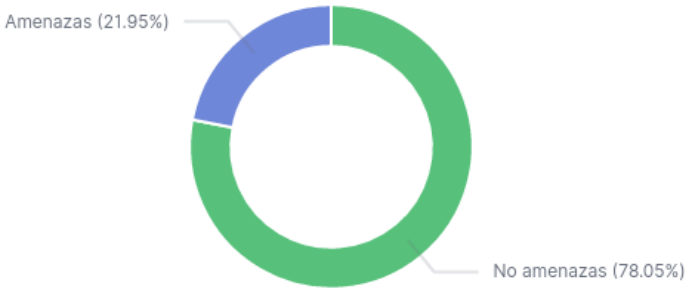


Figura 135: Eval. datos. Fuentes amenazas túnel

En relación a la ubicación geográfica, China ha generado el 88.89% de las conexiones sospechosas y Francia el 11,11% restante.

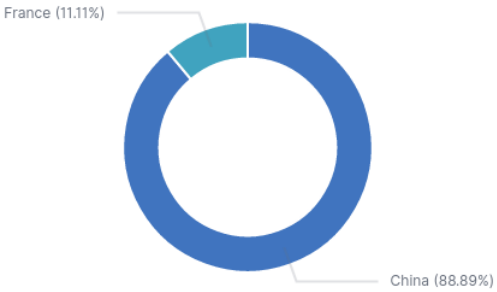


Figura 136: Eval. datos. Top 10 países origen amenazas túnel

En relación a la detección de los orígenes sospechosos, todos los etiquetados se corresponden con la fuente Turris.

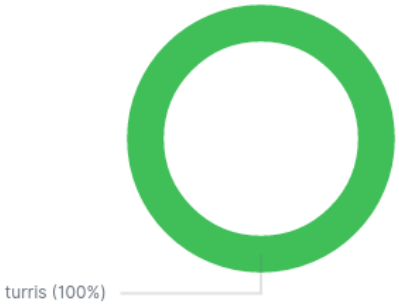


Figura 137: Eval. datos. Fuentes amenazas túnel

No se ha considerado que se pueda obtener ninguna lección concluyente de este tipo de tráfico dado el escaso volumen de datos disponible.

4.2.17 Evaluación de datos del tráfico extraño

En relación a este tipo de tráfico tan solo se recibido 2.017.740 conexiones de las que 289.752 se han etiquetado como sospechosas, lo que hace una media de 18109,5 diarias y una tasa de éxito en el etiquetado de, 14.36%.

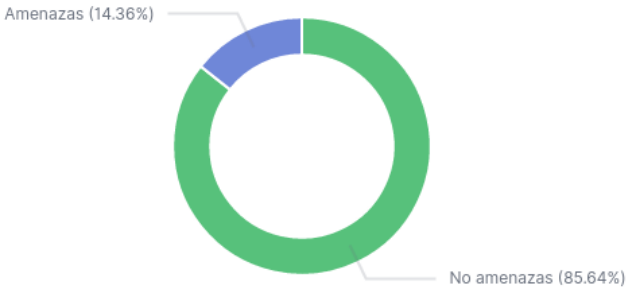


Figura 138: Eval. datos. Fuentes amenazas tráfico extraño

En relación a la ubicación geográfica, China supone el 27,89% del trafico malicioso, seguido de Francia con el 15,39% y como tercero España con el 12,12% del total por el ya comentado anteriormente ataque de telnet.

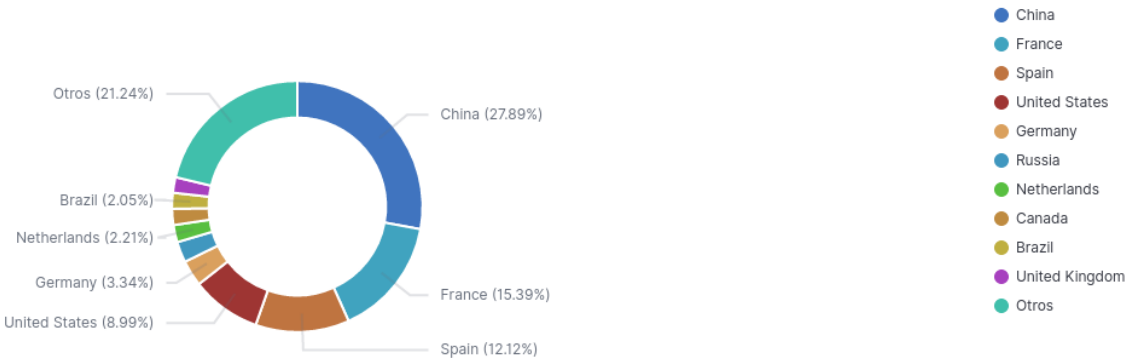


Figura 139: Eval. datos. Top 10 países origen amenazas tráfico extraño

En relación a la detección de los orígenes sospechosos, la fuente Blocklist.de ha etiquetado el 74,15% del total, seguido de Turris con un 24.23%, BadIps con un 1,46% y por último Danme.uk con un 0.15%.

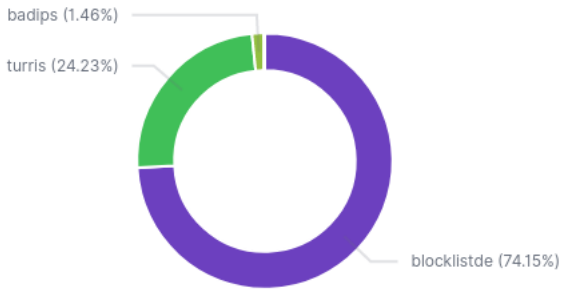


Figura 140: Eval. datos. Fuentes amenazas tráfico extraño

En este tipo de tráfico, más que los orígenes detectados, resulta interesante analizar las anomalías detectadas por Zeek.

Anomalía	Count	Count percentages
data_before_established	156,377	52.2%
inappropriate_FIN	127,421	42.5%
bad_TCP_checksum	6,822	2.3%
active_connection_reuse	3,779	1.3%
TCP_seq_underflow_or_misorder	2,294	0.8%
premature_connection_reuse	1,699	0.6%
data_after_reset	668	0.2%
SYN_seq_jump	242	0.1%
bad_HTTP_request	219	0.1%
FIN_advanced_last_seq	81	0%
Otros	118	0%
	299,720	

Figura 141: Eval. datos. Anomalías detectadas

4.2.18 Evaluación de datos del tráfico X509

En relación a este tipo de tráfico tan solo se han recibido 1159 peticiones de las que ninguna se ha etiquetado como maliciosa, lo que es lo esperado teniendo en cuenta que no se disponen de fuentes de información de amenazas especificas para certificados digitales.

La información más relevante en este caso proviene de los asuntos y emisores de los certificados que se han utilizado para verificar que los certificados se corresponden con emisores cualificados y que los asuntos se corresponden con los sitios esperados.

certificate.subject.keyword: Descending	Count	Count percentages
CN=www.badips.com	525	45.4%
CN=*.fedoraproject.org,O=Red Hat\, Inc.,L=Raleigh,ST=North Carolina,C=US	347	30%
CN=*.blocklist.de,OU=Domain Control Validated	267	23.1%
CN=api.telegram.org,OU=Domain Control Validated	6	0.5%
CN=*.23media.com,OU=SGC SSL Wildcard,OU=IT,O=23media GmbH,street=Johann-Krane-Weg 18,L=Muenster,ST=NRW,postalCode=48149,C=DE	4	0.3%
CN=*.snap.mirrors.ovh.net,OU=PositiveSSL Wildcard,OU=Domain Control Validated	3	0.3%
CN=rtm.ovh.net	3	0.3%
CN=*.docker.com	1	0.1%
CN=project.turris.cz	1	0.1%
	1,157	

Figura 142: Eval. datos. Top 10 asunto certificados

certificate.issuer.keyword: Descending	Count	Count percentages
CN=RapidSSL RSA CA 2018,OU=www.digicert.com,O=DigiCert Inc,C=US	525	45.4%
CN=DigiCert SHA2 High Assurance Server CA,OU=www.digicert.com,O=DigiCert Inc,C=US	347	30%
CN=AlphaSSL CA - SHA256 - G2,O=GlobalSign nv-sa,C=BE	267	23.1%
CN=Go Daddy Secure Certificate Authority - G2,OU=http://certs.godaddy.com/repository/,O=GoDaddy.com\, Inc.,L=Scottsdale,ST=Arizona,C=US	6	0.5%
CN=Sectigo RSA Domain Validation Secure Server CA,O=Sectigo Limited,L=Salford,ST=Greater Manchester,C=GB	6	0.5%
CN=COMODO RSA Organization Validation Secure Server CA,O=COMODO CA Limited,L=Salford,ST=Greater Manchester,C=GB	4	0.3%
CN=Amazon,OU=Server CA 1B,O=Amazon,C=US	1	0.1%
CN=Let's Encrypt Authority X3,O=Let's Encrypt,C=US	1	0.1%
	1,157	

Figura 143: Eval. datos. Top 10 emisor certificados

5 Conclusiones

5.1 Conclusiones finales

A través del trabajo desarrollado se han obtenido las siguientes conclusiones finales:

- Es totalmente posible y viable disponer de un sistema de detección de intrusos utilizando como base Zeek IDS y el conjunto ELK Stack. Ambos productos son soluciones sencillas. Zeek IDS ha resultado especialmente sencillo de utilizar para desgranar la información contenida en cada conexión y la plataforma ELK ha resultado ser muy útil a la hora de procesar y representar de forma visual toda esta información.
- Este tipo de soluciones se puede desarrollar mediante herramientas de código abierto y con un bajo presupuesto. Teniendo en cuenta esto, resulta muy asequible para las organizaciones disponer de este tipo de soluciones. Además, al tratarse de soluciones personalizadas, permite adquirir un gran conocimiento del estado de la seguridad a nivel de redes de la organización.
- Es sencillo integrar fuentes de información sobre amenazas y que están sean correlacionadas con la información obtenida de Zeek. A pesar de no se una funcionalidad propia de ELK Stack en su versión de código abierto, con unos simples «scripts» se ha podido suplir esta carencia.
- La principal dificultad que se ha detectado en relación a las fuentes de información de amenazas es la calidad de las propias fuentes. A pesar de haber utilizado varias de las que disponen de mejor reputación a tenor de la información localizada por Internet, los ratios de detección de amenazas no han sido altos con la excepción del tráfico SSH.
- El alto número de peticiones y la constancia a lo largo del tiempo que ha recibido el «honeypot» ha sido sorprendente. Esto no hace más que confirmar que la posibilidad de ser la victima de actividades maliciosas es alto y que es necesario establecer mecanismos de protección tanto pasivos como activos para reducir este riesgo.
- El principal hándicap que se ha encontrado es que requieren de un gran esfuerzo para mantenerlas operativas y actualizadas. En el caso de pequeñas organizaciones puede no que no sea asumible dedicar recursos específicos y por eso pueda resultar más viable el uso de soluciones comerciales que sean gestionadas externamente.

5.2 Revisión de los objetivos planteados

Tras completar el trabajo, se realiza un breve análisis de los objetivos planteados al comienzo de este proyecto.

Se confirma que el objetivo principal planteado que era «la evaluación de la capacidad de una herramienta IDS como Zeek combinada con un entorno de visualización y explotación de datos como ELK Stack para analizar actividades sospechosas en la red» se ha conseguido plenamente tal y como se ha

demostrado con las entregas realizadas y que se confirma a lo largo del desarrollo del presente documento.

Adicionalmente se planteó una serie de objetivos secundarios que se repasan a continuación:

- «Diseñar una solución basada el software comentado anteriormente que pueda ser desplegada en poco tiempo y de forma automatizada en cualquier arquitectura». Este objetivo se ha completado mediante la creación de «playbook» de Ansible que permite el despliegue automatizado en minutos de toda la solución.
- «Utilización de componentes de software libre y/o de fuentes abiertas que permitan la libre utilización y disponibilidad de esta solución en el futuro». Se ha verificado que todos los componentes utilizados disponen de una licencia de código abierto y los «scripts» desarrollados ad hoc se liberan también con una licencia abierta.
- «Evaluar las capacidades de aprendizaje de máquina de las que dispone ELK Stack». Este objetivo no se pudo realizar porque entraba en conflicto directo con el objetivo anterior de mantener solo componentes de fuentes abiertas.
- «Integrar listas de reputación de software malicioso y/o redes de robots informáticos para detectar posibles ataques y generar alertas». Se han integrado con éxito cuatro fuentes de inteligencia de amenazas diferentes, por lo que se ha cubierto el objetivo.
- «Diseñar un cuadro de mandos con la información más relevante que permita disponer de la información más relevante en un único punto». Se han diseñado varios cuadros de mandos, uno general y luego varios específicos para cada tipo de tráfico detectado, por lo que se ha cubierto el objetivo.
- «Exponer un conjunto de recomendaciones y mejoras que se podrían realizar basado en la experiencia obtenida». En los apartado de Análisis y evaluación de los datos generados y en el apartado de Conclusiones se han detallado las recomendaciones y mejoras propuestas, por lo que se da por cubierto este objetivo.

5.3 Análisis del seguimiento de la planificación y metodología

Dado que la planificación propuesta era bastante conservadora, se han podido desarrollar todos los hitos sin mayor problema y dentro de los plazos previstos.

Durante la evaluación de los componentes se detectó que la versión normal que incluye los módulos de aprendizaje maquina e inteligencia de amenazas no dispone de una licencia de código abierto, por lo que fue necesario realizar un cambios en los objetivos para eliminar esta parte y a su vez sustituir los productos por las versiones libres. Estos cambios no han supuesto ningún retraso porque fue posible asumirlos dentro de la holgura del proyecto.

La metodología de gestión de proyectos en cascada ha permitido mantener el control constante de las tareas desarrolladas sin que fuera necesario dedicar

muchos recursos a esta tareas al mismo tiempo que ha facilitado conocer en cada momento el punto exacto del proyecto se encontraba el desarrollo.

5.4 Líneas de trabajo futuro

En relación a este proyecto se han establecido las siguientes posibles lineas de trabajo que podrían realizarse a futuro:

1. Evaluación de otras fuentes de inteligencia de amenazas para integrarlas en la solución con el objetivo de mejorar la capacidad de detección.
2. Implementación de una herramienta de aprendizaje máquina y especifica de análisis de amenazas dentro de la solución que respete las licencias de código abierto.
3. Sustitución del «honeypot» por uno de alta interacción que responda como si se tratase de servicios reales para conseguir una mayor interacción por parte de los posibles atacantes y aumentar las posibilidades de obtener mayor información.
4. Distribución de los «honeypots» de la solución por diferentes localizaciones alrededor del mundo y aumentar las interacciones. Esto también permitiría poder contrastar los datos de los distintos «honeypots» y estudiar si las interacciones son similares.
5. Evaluación de la plataforma Graylog como sustituto del conjunto ELK como base de correlación y análisis de información.

6 Glosario

- **Ansible**. Herramienta de software que proporciona una automatización sencilla pero potente para el soporte informático multiplataforma. Está destinado principalmente a los profesionales de la informática, que lo utilizan para el despliegue de aplicaciones, las actualizaciones en estaciones de trabajo y servidores, el aprovisionamiento de nubes, la gestión de configuraciones, la orquestación de servicios internos.
- **API**. Es el acrónimo de «Application Programming Interface» que se traduce como Interfaz de Programación de Aplicaciones. Se trata de unsoftware que actúa como intermediario entre dos aplicaciones que necesitan comunicarse entre ellas.
- **APM**. Es el acrónimo de «Application Performande Monitoring» que se traduce como «monitorización del rendimiento de aplicaciones». Se centra en asegurarse de que las aplicaciones se desempeñan tal y como se espera. El objetivo de la supervisión del rendimiento es proporcionar a los usuarios finales una experiencia de calidad.
- **Docker**. Herramienta diseñada para facilitar la creación, despliegue y ejecución de aplicaciones mediante el uso de contenedores. Los contenedores permiten a un desarrollador empaquetar una aplicación con todas las partes que necesita, como las bibliotecas y otras dependencias, e implementarla como un solo paquete
- **Geolocalización**. Consiste en obtener la ubicación geográfica real de un equipo o dispositivo conectado a una red de área ancha como Internet a través de la dirección IP.
- **«Hash»**. Es una función que convierte un valor en otro. Se trata de una práctica común en las ciencias de la computación que se utiliza para la

criptografía, la compresión, la generación de sumas de verificación y la indexación de datos.

- **IDS**. Es el acrónimo de «Intrusion Detection System» que se traduce como «sistema de detección de intrusos». Monitoriza el tráfico de la red en busca de actividades sospechosas permitiendo que se detecten los ataques que se originan dentro de una red.
- **Lista negra**. Es una lista direcciones IP que se considera que están desarrollando actividades maliciosas que deben ser evitadas y en las que no se debe confiar.
- **NAT**. Es el acrónimo de «Network Address Translation» que se traduce como «traducción de direcciones de red». Traduce las direcciones IP de las equipos internos en una red local a una sola dirección IP que permite el acceso de los equipos internos a otras redes como Internet sin tener que conectarse directamente a esta.
- **Pipeline**. Se traduce como «hilo de procesamiento». En el caso de Logstash, los hilos de procesamiento reciben información que se coloca en una cola interna. Estos hilos leen esta información de la cola y los procesan a través de una secuencia de filtros para posteriormente formatearlos para su salida hacía otro medio como, por ejemplo, Elasticsearch.

7 Bibliografía

- [1] *What is Snort?*. Cisco and/or its affiliates, © 2020 [consulta: 1 de marzo de 2020]. Disponible en: https://www.snort.org/faq/what-is-snort.
- [2] *Frequently Asked Questions*. The Bro Project, © 2014 [consulta: 1 de marzo de 2020]. Disponible en: https://www.zeek.org/documentation/faq.html#what-is-zeek.
- [3] *What is Suricata*. OISF, © 2016-2019 [consulta: 1 de marzo de 2020]. Disponible en: https://suricata.readthedocs.io/en/suricata-5.0.2/what-is-suricata.html.
- [4] TAYYEBI, Yakuta; BHILARE D.S. © 2018. *A Comparative Study of Open Source Network Based Intrusion Detection Systems*. Indore, India: International Journal of Computer Science and Information Technologies (Vol. 9 (2) , 2018, 23-26).
- [5] *What is the ELK Stack?*. Elasticsearch B.V.,© 2020. [consulta: 1 de marzo de 2020]. Disponible en: https://www.elastic.co/what-is/elk-stack.
- [6] *About Graylog*. Graylog, © 2020 [consulta: 1 de marzo de 2020]. Disponible en: https://www.graylog.org/about.
- [7] *9 Top Intrusion Detection and Prevention Systems: Guide to IDPS*. Drew Robb, © 2018 [consulta: 1 de marzo de 2020]. Disponible en: https://www.esecurityplanet.com/products/top-intrusion-detection-prevention-systems.html.
- [8] HOCK, Filip; KORTIš, Peter. © 2015. *Commercial and open-source based Intrusion Detection System and Intrusion Prevention System (IDS/ IPS) design for an IP networks*. Zilina, Eslovaquia: 13th International Conference on Emerging eLearning Technologies and Applications (ICETA).

- [9] *Acerca de FreeBSD*. El proyecto FreeBSD, © 1995-2010 [consulta: 6 de marzo de 2020]. Disponible en: https://www.freebsd.org/es/about.html.
- [10] *CentOS Linux*. The CentOS Project, © 2020 [consulta: 6 de marzo de 2020]. Disponible en : https://www.centos.org/about/.
- [11] *Customers*. Docker Inc., © 2020 [consulta: 7 de marzo de 2020]. Disponible en: https://www.docker.com/customers.
- [12] *Heralding*. VESTERGAARD, Johnny. © 2020 [consulta: 7 de marzo de 2020]. Disponible en: https://github.com/johnnykv/heralding.
- [13] GHORBANI, Ali A.; LU, Wei; Tavallaee, Mahbod. © 2010. *Network Intrusion Detection and Prevention: Concepts and Techniques*. Canadá: Springer.
- [14] *Zeek Manual*. The Zeek Project, © 2019 [consulta. 9 de marzo de 2020]. Disponible en: https://docs.zeek.org/en/current/intro/index.html.
- [15] THOMPSON, Eric C. © 2020. *Designing a HIPAA-Compliant Security Operations Center: A Guide to Detecting and Responding to Healthcare Breaches and Events*. Dekalb, Illinois, EE.UU.: Apress.
- [16] *¿Qué es el ELK Stack?* Elasticsearch B.V., © 2020 [consulta: 12 de marzo de 2020]. Disponible en:https://www.elastic.co/es/what-is/elk-stack
- [17] *Suscripciones del Elastic Stack*. Elasticsearch B.V., © 2020 [consulta: 12 de marzo de 2020]. Disponible en: https://www.elastic.co/es/subscriptions
- [18] SHUKLA, Pranav; KUMAR M N, Sharath. © 2019. *Learning Elastic Stack 7.0: Distributed search, analytics, and visualization using Elasticsearch, Logstash, Beats, and Kibana, 2nd Edition*. Birmingham, Reino Unido: Packt.
- [19] SRIVASTAVA, Anurag. © 2019. *Kibana 7 Quick Start Guide: Visualize your Elasticsearch data with ease*. Birmingham, Reino Unido: Packt.
- [20] *Elasticsearch introduction*. Elasticsearch B.V., © 2020 [consulta: 12 de marzo de 2020]. Disponible en: https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html.
- [21] *Information out: search and analyze*. Elasticsearch B.V., © 2020 [consulta: 12 de marzo de 2020]. Disponible en: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-analyze.html.
- [22] *www.blocklist.de*. Blocklist.de. © 2007-2020. [consulta: 13 de marzo de 2020]. Disponible en: https://www.blocklist.de/en/index.html.
- [23] *Greylist*. CZ.NIC, z. s. p. o., © n/d [consulta: 13 de marzo de 2020]. Disponible en: https://project.turris.cz/en/greylist.

- [24] *Honeypot's Logo*. Wikimedia Commons. © 2015. [consulta: 19 de marzo de 2020]. Disponible en: https://commons.wikimedia.org/wiki/File:Logo1_PNG.png
- [25] *Marks & Logos Usage*. The Zeek Project. © 2020. [consulta: 19 de marzo de 2020]. Disponible en: https://zeek.org/marks-and-logos/.
- [26] *The Python Logo*. Python Software Foundation. © 2001-2020. [consulta: 19 de marzo de 2020]. Disponible en: https://www.python.org/community/logos/.
- [27] *Trademark guidelines*. Docker Inc. (c) 2020 [consulta: 19 de marzo de 2020]. Disponible en: https://www.docker.com/legal/trademark-guidelines.
- [28] *The CentOS Logo*. The CentOS Project © 2019 [consulta: 19 de marzo de 2020]. Disponible en: https://wiki.centos.org/ArtWork/Brand/Logo?action=AttachFile&do=view&target=centos-logo-light.svg
- [29] *Elastic branding*. Elasticsearch B.V., © 2020 [consulta: 20 de marzo de 2020]. Disponible en: https://www.elastic.co/es/brand.
- [30] *HAProxy Trademark Policy*. HAProxy Technologies, LLC, © 2020 [consulta: 15 de abril de 2020]. Disponible en: https://www.haproxy.com/trademark-policy/.
- [31] *Use host networking*. Docker Inc., © 2013-2020 [consulta: 20 de abril de 2020].Disponible en: https://docs.docker.com/network/host/.

8 Anexos

8.1 Ansible – Fichero main.yml

```
---
 - hosts: tfm
   vars:
     main_path: '/opt/tfm_mistic/'
   gather_facts: true
   become: yes
   roles:
     - prerequisitos
     - threats
     - heralding
     - elasticsearch
     - zeek
     - filebeat
     - kibana
     - logstash
     - haproxy
```

```
    - postconfig
```

8.2 Ansible – Fichero hosts

```
---
ungrouped:
  hosts:
    <ALIAS_HOST>:
      ansible_host: <IP_SERVIDOR_HOST>
      ansible_port: <PUERTO_SERVIDOR_HOST>
      ansible_user: <USUARIO_SSH>
      ansible_ssh_pass: <CONTRASEÑA_USUARIO_SSH>
      ansible_sudo_pass: <CONTRASEÑA_USUARIO_SSH>
```

8.3 Ansible – Rol Elasticsearch

8.3.1 Estructura de ficheros

```
.
├── files
│   └── docker
│       ├── config
│       │   ├── elasticsearch.yml
│       ├── docker-compose.yml
│       ├── Dockerfile
│       └── entrypoint.sh
├── tasks
│   └── main.yml
└── vars
    └── main.yml
```

8.3.2 Fichero docker-compose.yml

```
version: '3'
services:
  elasticsearch:
    build: .
    container_name: elasticsearch
    restart: always
    networks:
      - localnet
```

```
    expose:
      - "9200"
    volumes:
      - config:/opt/elasticsearch/config
      - data:/opt/elasticsearch/data
volumes:
  config:
  data:
networks:
  localnet:
```

8.3.3 Fichero Dockerfile

```
FROM centos:7
LABEL maintainer "igalvit@uoc.edu"
RUN yum update -y \
    && yum -y install wget java-1.8.0-openjdk \
    && yum clean all
RUN groupadd -g 1200 elasticsearch \
    && useradd -u 1200 -g 1200 -d /opt/elasticsearch elasticsearch \
    && mkdir /opt/elasticsearch/data \
    && mkdir /opt/elasticsearch/logs \
    && cd /tmp \
    && wget
https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-
oss-7.6.1-linux-x86_64.tar.gz \
    && tar xzvf elasticsearch-oss-7.6.1-linux-x86_64.tar.gz \
    && mv elasticsearch-7.6.1/* /opt/elasticsearch/ \
    && rm -rf /tmp/elasticsearch*
ENV PATH $PATH:/opt/elasticsearch/bin
COPY config /opt/elasticsearch/config
COPY entrypoint.sh /opt/entrypoint.sh
RUN chown -R elasticsearch. /opt/elasticsearch \
    && chown elasticsearch. /opt/entrypoint.sh \
    && chmod +x /opt/entrypoint.sh
WORKDIR /opt/elasticsearch
USER elasticsearch:root
ENTRYPOINT ["/opt/entrypoint.sh"]
CMD ["elasticsearch"]
```

8.3.4 Fichero entrypoint.sh

```
#!/bin/bash
set -e
exec "$@"
```

8.3.5 Fichero elasticsearch.yml

```
cluster.name: tfm-cluster
node.name: node-1
path.data: /opt/elasticsearch/data
path.logs: /opt/elasticsearch/logs
network.host: 0.0.0.0
cluster.initial_master_nodes: ["node-1"]
```

8.3.6 Fichero tasks/main.yml

```
---
- name: copia de los ficheros del servicio
  copy:
    src: "{{ role_path }}/files/docker/"
    dest: "{{ service_path }}"
- name : construcción del servicio
  docker_compose:
    project_name: elasticsearch
    project_src: "{{ service_path }}"
    build: yes
- name : pausa antes de verificar el estado del servicio
  pause:
    seconds: 5
- name: verifica si se está ejecutando
  docker_container_info:
    name: elasticsearch
  register: is_running
- name: comprobar la salida de la verificación
  assert:
    that:
      - is_running.exists
```

8.3.7 Fichero vars/main.yml

```
---
```

```
service_path: "{{ main_path }}/elasticsearch"
```

8.4 Ansible – Rol Filebeat

8.4.1 Estructura de ficheros

```
.
├── files
│   └── docker
│       ├── config
│       │   └── filebeat.yml
│       ├── docker-compose.yml
│       ├── Dockerfile
│       └── entrypoint.sh
├── tasks
│   └── main.yml
└── vars
    └── main.yml
```

8.4.2 Fichero docker-compose.yml

```
version: '3'
services:
  filebeat:
    build: .
    container_name: filebeat
    restart: always
    networks:
      - elasticsearch_localnet
    volumes:
    - zeek_logs:/opt/zeek/spool/zeek:ro
volumes:
  zeek_logs:
    external: true
networks:
  elasticsearch_localnet:
    external: true
```

8.4.3 Fichero Dockerfile

```
FROM centos:7
```

```
LABEL maintainer "igalvit@uoc.edu"
RUN yum update -y \
    && yum -y install wget \
    && yum clean all \
    && cd /tmp \
    && wget
https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-oss-
7.6.1-linux-x86_64.tar.gz \
    && tar xzvf filebeat-oss-7.6.1-linux-x86_64.tar.gz \
    && mv filebeat-7.6.1-linux-x86_64 /opt/filebeat \
    && mkdir /opt/filebeat/logs
ENV PATH $PATH:/opt/filebeat
COPY ./config/filebeat.yml /opt/filebeat/filebeat.yml
COPY ./entrypoint.sh /opt/entrypoint.sh
RUN chmod +x /opt/entrypoint.sh \
    && rm -f /tmp/filebeat*
WORKDIR /opt/filebeat
ENTRYPOINT ["/opt/entrypoint.sh"]
```

8.4.4 Fichero entrypoint.sh

```
#!/bin/bash
set -eox pipefail
filebeat -e
```

8.4.5 Fichero filebeat.yml

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /opt/zeek/spool/zeek/*.log
  json.keys_under_root: true
  json.add_error_key: true
  json.message_key: uid
  exclude_lines: ['^#']
  exclude_files:
    - broker.log
    - capture_loss.log
    - cluster.log
```

```
    - config.log
    - loaded_scripts.log
    - packet_filter.log
    - print.log
    - prof.log
    - reporter.log
    - stats.log
    - stderr.log
    - stdout.log
filebeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false
#------------------------ Logstash output --------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["logstash:5044"]
#---------------------------- Logging ------------------------------
logging.level: info
logging.to_files: true
logging.files:
  path: ${path.config}/logs/filebeat
  name: filebeat
  keepfiles: 7
  permissions: 0644
```

8.4.6 Fichero tasks/main.yml

```
---
- name: copia de los ficheros del servicio
  copy:
    src: "{{ role_path }}/files/docker/"
    dest: "{{ service_path }}"
- name : construcción del servicio
  docker_compose:
    project_name: filebeat
    project_src: "{{ service_path }}"
    build: yes
- name : pausa antes de verificar el estado del servicio
  pause:
```

```
    seconds: 5
- name: verifica si se está ejecutando
  docker_container_info:
    name: filebeat
  register: is_running
- name: comprobar la salida de la verificación
  assert:
    that:
      - is_running.exists
```

8.4.7 Fichero vars/main.yml

```
---
service_path: "{{ main_path }}/filebeat"
```

8.5 Ansible – Rol HAProxy

8.5.1 Estructura de ficheros

```
.
├── files
│   └── docker
│       ├── config
│       │   ├── cert.pem[2]
│       │   ├── entrypoint.sh
│       │   ├── haproxy.cfg
│       │   ├── haproxy.pem[3]
│       │   └── key.pem[4]
│       ├── docker-compose.yml
│       └── Dockerfile
├── tasks
│   └── main.yml
└── vars
    └── main.yml
```

8.5.2 Fichero docker-compose.yml

```
version: '3'
services:
```

2 Este fichero se tiene que generar y ubicar en esta ruta con el nombre cert.pem
3 Este fichero se tiene que generar y ubicar en esta ruta con el nombre haproxy.pem
4 Este fichero se tiene que generar y ubicar en esta ruta con el nombre key.pem

```
  haproxy:
    build: .
    container_name: haproxy
    restart: always
    networks:
      - elasticsearch_localnet
    ports:
      - "5602:5602"
    volumes:
      - config:/opt/haproxy
    external_links:
      - kibana
volumes:
  config:
networks:
  elasticsearch_localnet:
    external: true
```

8.5.3 Fichero Dockerfile

```
FROM centos:7
LABEL maintainer "igalvit@uoc.edu"
RUN yum update -y \
    && yum -y install haproxy \
    && yum clean all
COPY config /etc/haproxy
RUN chmod +x /etc/haproxy/entrypoint.sh
CMD ["haproxy", "-f", "/etc/haproxy/haproxy.cfg"]
```

8.5.4 Fichero entrypoint.sh

```
#!/bin/bash
set -eox pipefail
haproxy -f /etc/haproxy&haproxy.cfg
exec /bin/bash -c "trap : TERM INT; sleep infinity & wait"
```

8.5.5 Fichero haproxy.cfg

```
global
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
```

```
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    tune.ssl.default-dh-param 2048
    ssl-default-bind-options no-sslv3 no-tls-tickets
    ssl-default-bind-ciphers
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA+AE
SGCM:RSA+AES:!aNULL:!MD5:!DSS
    # turn on stats unix socket
    stats socket /var/lib/haproxy/stats
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
userlist kibana_users
  user tfm password CUfced0z7LHps
frontend web
acl auth_ok http_auth(kibana_users)
bind *:5602 ssl crt /etc/haproxy/haproxy.pem
http-request auth unless auth_ok
default_backend kibana
backend kibana
  mode http
  server kibana kibana:5601 check
```

8.5.6 Fichero tasks/main.yml

```
---
- name: copia de los ficheros del servicio
  copy:
    src: "{{ role_path }}/files/docker/"
    dest: "{{ service_path }}"
- name : construcción del servicio
  docker_compose:
    project_name: haproxy
    project_src: "{{ service_path }}"
    build: yes
- name : pausa antes de verificar el estado del servicio
  pause:
    seconds: 5
- name: verifica si se está ejecutando
  docker_container_info:
    name: haproxy
  register: is_running
- name: comprobar la salida de la verificación
  assert:
    that:
      - is_running.exists
```

8.5.7 Fichero vars/main.yml

```
---
service_path: "{{ main_path }}/haproxy"
```

8.6 Ansible – Rol Heralding

8.6.1 Estructura de ficheros

```
.
├── files
│   └── docker
│       ├── config
│       │   ├── entrypoint.sh
│       │   └── heralding.yml
│       ├── docker-compose.yml
│       ├── Dockerfile
```

```
│       └── heralding_src[5]
├── tasks
│   └── main.yml
└── vars
    └── main.yml
```

8.6.2 Fichero docker-compose.yml

```
version: '3'
services:
  heralding:
    build: .
    container_name: heralding
    restart: always
    networks:
      - localnet
    ports:
      - "21:21"
      - "22:22"
      - "23:23"
      - "25:25"
      - "80:80"
      - "110:110"
      - "143:143"
      - "443:443"
      - "993:993"
      - "995:995"
      - "1080:1080"
      - "3306:3306"
      - "3389:3389"
      - "5432:5432"
      - "5900:5900"
    volumes:
    - home:/home/heralding
volumes:
  home:
networks:
  localnet:
```

5 El código fuente de Heralding no está incluido en este documento.

8.6.3 Fichero Dockerfile

```
FROM centos:7
LABEL maintainer "igalvit@uoc.edu"
RUN yum update -y \
    && yum -y install python3 python3-pip python3-devel \
       gcc gcc-c++ make openssl-devel libffi-devel postgresql-devel \
       git \
    && yum clean all \
    && rm -rf /var/cache/yum \
    && groupadd -g 1500 heralding \
    && useradd -u 1500 -g 1500 heralding \
    && setcap cap_net_bind_service=+ep /usr/bin/python3.6
USER heralding:heralding
COPY --chown=heralding:heralding ./config /home/heralding
COPY --chown=heralding:heralding ./heralding_src /tmp/heralding
RUN chmod +x /home/heralding/entrypoint.sh \
    && mkdir /home/heralding/logs \
    && cd /tmp/heralding \
    && pip3 install --user --no-cache-dir -r requirements.txt \
    && pip3 install --user --no-cache-dir . \
    && rm -rf /tmp/heralding
WORKDIR /home/heralding
ENV PATH $PATH:/home/heralding/.local/bin
ENTRYPOINT ["/home/heralding/entrypoint.sh"]
```

8.6.4 Fichero entrypoint.sh

```
#!/bin/bash
set -eox pipefail
cd /home/heralding/logs
heralding -c /home/heralding/heralding.yml
```

8.6.5 Fichero heralding.yml

```
# will request and log the public ip every hours from ipify
public_ip_as_destination_ip: false
# ip address to listen on
bind_host: 0.0.0.0
# logging of sessions and authentication attempts
activity_logging:
```

```
  file:
    enabled: true
     # Session details common for all protocols (capabilities) in CSV
format,
    # written to file when the session ends. Set to "" to disable.
    session_csv_log_file: "log_session.csv"
     # Complete session details (including protocol specific data) in
JSONL format,
    # written to file when the session ends. Set to "" to disable
    session_json_log_file: "log_session.json"
        # Writes each authentication attempt to file, including
credentials,
    # set to "" to disable
    authentication_log_file: "log_auth.csv"
  syslog:
    enabled: false
  hpfeeds:
    enabled: false
    session_channel: "heralding.session"
    auth_channel: "heralding.auth"
    host:
    port: 20000
    ident:
    secret:
  curiosum:
    enabled: false
    port: 23400
hash_cracker:
  enabled: true
  wordlist_file: 'wordlist.txt'
# protocols to enable
capabilities:
  ftp:
    enabled: true
    port: 21
    timeout: 30
    protocol_specific_data:
      max_attempts: 3
      banner: "Microsoft FTP Server"
      syst_type: "Windows-NT"
```

```
  telnet:
    enabled: true
    port: 23
    timeout: 30
    protocol_specific_data:
      max_attempts: 3
  pop3:
    enabled: true
    port: 110
    timeout: 30
    protocol_specific_data:
      max_attempts: 3
  pop3s:
    enabled: true
    port: 995
    timeout: 30
    protocol_specific_data:
      max_attempts: 3
       # if a .pem file is not found in work dir, a new pem file will
be created
      # using these values
      cert:
        common_name: "*"
        country: "US"
        state: None
        locality: None
        organization: None
        organizational_unit: None
        # how many days should the certificate be valid for
        valid_days: 365
        serial_number: 0
  postgresql:
    enabled: true
    port: 5432
    timeout: 30
  imap:
    enabled: true
    port: 143
    timeout: 30
```

```
    protocol_specific_data:
      max_attempts: 3
      banner: "* OK IMAP4rev1 Server Ready"
  imaps:
    enabled: true
    port: 993
    timeout: 30
    protocol_specific_data:
      max_attempts: 3
      banner: "* OK IMAP4rev1 Server Ready"
       # if a .pem file is not found in work dir, a new pem file will
be created
      # using these values
      cert:
        common_name: "*"
        country: "US"
        state: None
        locality: None
        organization: None
        organizational_unit: None
        # how many days should the certificate be valid for
        valid_days: 365
        serial_number: 0
  ssh:
    enabled: true
    port: 22
    timeout: 30
    protocol_specific_data:
      banner: "SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.8"
  http:
    enabled: true
    port: 80
    timeout: 30
    protocol_specific_data:
      banner: ""
  https:
    enabled: true
    port: 443
    timeout: 30
```

```
    protocol_specific_data:
      banner: ""
       # if a .pem file is not found in work dir, a new pem file will
be created
      # using these values
      cert:
        common_name: "*"
        country: "US"
        state: None
        locality: None
        organization: None
        organizational_unit: None
        # how many days should the certificate be valid for
        valid_days: 365
        serial_number: 0
  smtp:
    enabled: true
    port: 25
    timeout: 30
    protocol_specific_data:
      banner: "Microsoft ESMTP MAIL service ready"
       # If the fqdn option is commented out or empty, then fqdn of
the host will be used
      fqdn: ""
  vnc:
    enabled: true
    port: 5900
    timeout: 30
  socks5:
    enabled: true
    port: 1080
    timeout: 30
  mysql:
    enabled: true
    port: 3306
    timeout: 30
  rdp:
    enabled: true
    port: 3389
```

```
    timeout: 30
    protocol_specific_data:
      banner: ""
       # if a .pem file is not found in work dir, a new pem file will
be created
      # using these values
      cert:
        common_name: "*"
        country: "US"
        state: None
        locality: None
        organization: None
        organizational_unit: None
        # how many days should the certificate be valid for
        valid_days: 365
        serial_number: 0
```

8.6.6 Fichero tasks/main.yml

```
---
- name: copia de los ficheros del servicio
  copy:
    src: "{{ role_path }}/files/docker/"
    dest: "{{ service_path }}"
- name : construcción del servicio
  docker_compose:
    project_name: heralding
    project_src: "{{ service_path }}"
    build: yes
- name : pausa antes de verificar el estado del servicio
  pause:
    seconds: 5
- name: verifica si se está ejecutando
  docker_container_info:
    name: heralding
  register: is_running
- name: comprobar la salida de la verificación
  assert:
    that:
      - is_running.exists
```

8.6.7 Fichero vars/main.yml

```
---
service_path: "{{ main_path }}/heralding"
```

8.7 Ansible – Rol Kibana

8.7.1 Estructura de ficheros

```
.
├── files
│   └── docker
│       ├── config
│       │   └── kibana.yml
│       ├── docker-compose.yml
│       ├── Dockerfile
│       └── entrypoint.sh
├── tasks
│   └── main.yml
└── vars
    └── main.yml
```

8.7.2 Fichero docker-compose.yml

```
version: '3'
services:
  kibana:
    build: .
    container_name: kibana
    restart: always
    networks:
      - elasticsearch_localnet
    expose:
      - "5601"
    volumes:
      - config:/opt/kibana/config
volumes:
  config:
networks:
  elasticsearch_localnet:
    external: true
```

8.7.3 Fichero Dockerfile

```
FROM centos:7
LABEL maintainer "igalvit@uoc.edu"
RUN yum update -y \
    && yum -y install wget \
    && yum clean all \
    && groupadd -g 1300 kibana \
    && useradd -u 1300 -g 1300 -d /opt/kibana kibana
USER kibana:kibana
RUN cd /tmp \
     && wget https://artifacts.elastic.co/downloads/kibana/kibana-oss-7.6.1-linux-x86_64.tar.gz \
    && tar xzvf kibana-oss-7.6.1-linux-x86_64.tar.gz \
    && mv kibana-7.6.1-linux-x86_64/* /opt/kibana/ \
    && rm -rf /tmp/kibana*
ENV PATH $PATH:/opt/kibana/bin
COPY --chown=kibana:kibana config /opt/kibana/config
COPY --chown=kibana:kibana entrypoint.sh /opt/kibana/entrypoint.sh
RUN chmod +x /opt/kibana/entrypoint.sh
WORKDIR /opt/kibana
ENTRYPOINT ["/opt/kibana/entrypoint.sh"]
CMD ["kibana"]
```

8.7.4 Fichero kibana.yml

```
server.host: 0.0.0.0
server.name: "kibana"
elasticsearch.hosts: ["http://elasticsearch:9200"]
```

8.7.5 Fichero entrypoint.sh

```
#!/bin/bash
set -e
exec "$@"
```

8.7.6 Fichero tasks/main.yml

```
---
- name: copia de los ficheros del servicio
  copy:
    src: "{{ role_path }}/files/docker/"
    dest: "{{ service_path }}"
```

```
- name : construcción del servicio
  docker_compose:
    project_name: kibana
    project_src: "{{ service_path }}"
    build: yes
- name : pausa antes de verificar el estado del servicio
  pause:
    seconds: 5
- name: verifica si se está ejecutando
  docker_container_info:
    name: kibana
  register: is_running
- name: comprobar la salida de la verificación
  assert:
    that:
      - is_running.exists
```

8.7.7 Fichero vars/main.yml

```
---
service_path: "{{ main_path }}/kibana"
```

8.8 Ansible – Rol Logstash

8.8.1 Estructura de ficheros

```
.
├── files
│   └── docker
│       ├── config
│       │   ├── logstash.yml
│       │   ├── main.conf
│       │   └── tfm.json
│       ├── docker-compose.yml
│       ├── Dockerfile
│       ├── entrypoint.sh
│       └── geoip
│           └── GeoIP.conf
├── tasks
│   └── main.yml
```

```
└── vars
    └── main.yml
```

8.8.2 Fichero docker-compose.yml

```
version: '3'
services:
  logstash:
    build: .
    container_name: logstash
    restart: always
    networks:
      - elasticsearch_localnet
    expose:
    - "9600"
    volumes:
    - config:/opt/logstash/config
    - threats_files:/opt/threats:ro
volumes:
  config:
  threats_files:
    external: true
networks:
  elasticsearch_localnet:
    external: true
```

8.8.3 Fichero Dockerfile

```
FROM centos:7
LABEL maintainer "igalvit@uoc.edu"
RUN yum update -y \
    && yum -y install wget java-1.8.0-openjdk geoipupdate-cron \
    && yum clean all \
    && cd /tmp \
    && wget https://artifacts.elastic.co/downloads/logstash/logstash-
oss-7.6.1.tar.gz \
    && tar xzvf logstash-oss-7.6.1.tar.gz \
    && mv logstash-7.6.1 /opt/logstash \
    && rm -f /tmp/logstash* \
    && mkdir /opt/geolite2
ENV PATH $PATH:/opt/logstash/bin
COPY config /opt/logstash/config
COPY geoip/GeoIP.conf /etc/GeoIP.conf
COPY entrypoint.sh /opt/entrypoint.sh
RUN chmod +x /opt/entrypoint.sh \
    && geoipupdate
WORKDIR /opt/logstash
ENTRYPOINT ["/opt/entrypoint.sh"]
```

8.8.4 Fichero entrypoint.sh

```
#!/bin/bash
set -eox pipefail
logstash
```

8.8.5 Fichero logstash.yml

```
node.name: logstash-1
```

```
pipeline.id: main
pipeline.workers: 2
path.config: "/opt/logstash/config/main.conf"
```

8.8.6 Fichero main.conf

```
input {
  beats {
    port => 5044
  }
}
filter {
  date {
    match => ["ts", "UNIX"]
    target => "@timestamp"
  }
  grok {
    match => { "[log][file][path]" => ["^%{GREEDYDATA}%
{WORD:file_type}.log"] }
    named_captures_only => true
    add_field => { "log_type" => "%{file_type}" }
    tag_on_failure => []
  }
  if [log_type] == "x509" {
    mutate {
      remove_field => ["id"]
    }
    date {
      match => ["certificate.not_valid_after", "UNIX"]
      target => "certificate.not_valid_after"
    }
    date {
      match => ["certificate.not_valid_before", "UNIX"]
      target => "certificate.not_valid_before"
    }
  }
  if [local_orig] == 'true' and [local_resp] == 'true' {
    drop { }
  }
  if [log_type] == "dns" and [id.orig_h] == "176.31.103.131"  {
    drop { }
  }
  if [log_type] == "ntp" and ([id.orig_h] == "176.31.103.131" or
[id.orig_h] == "2001:41d0:8:1883::1") {
    drop { }
  }
  if [id.orig_h] != "176.31.103.131" {
    geoip {
      source => "id.orig_h"
      target => "geoip_orig_h"
      tag_on_failure => []
    }
    translate {
      field => "id.orig_h"
      destination => "threat_type"
      dictionary_path => "/opt/threats/downloads/turris.csv"
      refresh_behaviour => "replace"
      add_field => { "threat_source" => "turris" }
      add_tag => "threat"
```

```
    }
    translate {
      field => "id.orig_h"
      destination => "threat_type"
      dictionary_path => "/opt/threats/downloads/badips.csv"
      refresh_behaviour => "replace"
      add_field => { "threat_source" => "badips" }
      add_tag => "threat"
    }
    translate {
      field => "id.orig_h"
      destination => "threat_type"
      dictionary_path => "/opt/threats/downloads/blocklistde.csv"
      refresh_behaviour => "replace"
      add_field => { "threat_source" => "blocklistde" }
      add_tag => "threat"
    }
    translate {
      field => "id.orig_h"
      destination => "threat_type"
      dictionary_path => "/opt/threats/downloads/danmeuk.csv"
      refresh_behaviour => "replace"
      add_field => { "threat_source" => "danmeuk" }
      add_tag => "threat"
    }
  }
  if [id.resp_h] != "176.31.103.131" {
    geoip {
      source => "id.resp_h"
      target => "geoip_resp_h"
      tag_on_failure => []
    }
    translate {
      field => "id.resp_h"
      destination => "threat_type"
      dictionary_path => "/opt/threats/downloads/turris.csv"
      refresh_behaviour => "replace"
      add_field => { "threat_source" => "turris" }
      add_tag => "threat"
    }

    translate {
      field => "id.resp_h"
      destination => "threat_type"
      dictionary_path => "/opt/threats/downloads/badips.csv"
      refresh_behaviour => "replace"
      add_field => { "threat_source" => "badips" }
      add_tag => "threat"
    }
    translate {
      field => "id.resp_h"
      destination => "threat_type"
      dictionary_path => "/opt/threats/downloads/blocklistde.csv"
      refresh_behaviour => "replace"
      add_field => { "threat_source" => "blocklistde" }
      add_tag => "threat"
    }
    translate {
      field => "id.resp_h"
```

```
      destination => "threat_type"
      dictionary_path => "/opt/threats/downloads/danmeuk.csv"
      refresh_behaviour => "replace"
      add_field => { "threat_source" => "danmeuk" }
      add_tag => "threat"
    }
  }
  if [tx_hosts] != "176.31.103.131" {
    geoip {
     source => "tx_hosts"
     target => "geoip_tx_hosts"
     tag_on_failure => []
    }
    translate {
      field => "md5"
      destination => "threat_type"
      dictionary_path => "/opt/threats/downloads/malwarebazaar.csv"
      refresh_behaviour => "replace"
      add_field => { "threat_source" => "malwarebazaar" }
      add_tag => "threat"
    }
  }
  if [rx_hosts] != "176.31.103.131" {
    geoip {
     source => "rx_hosts"
     target => "geoip_rx_hosts"
     tag_on_failure => []
    }
    translate {
      field => "md5"
      destination => "threat_type"
      dictionary_path => "/opt/threats/downloads/malwarebazaar.csv"
      refresh_behaviour => "replace"
      add_field => { "threat_source" => "malwarebazaar" }
      add_tag => "threat"
    }
  }
  mutate {
    split => { "threat_type" => ',' }
  }
  mutate {
    remove_field => [ "ts", "[log][offset]", "[agent][ephemeral_id]",
"[agent][id]", "[agent][type]", "[agent][version]", "[log][file]
[path]", "[input][type]", "file_type" ]
    remove_tag => "beats_input_raw_event"
  }
}
output {
  elasticsearch {
    hosts => ["http://elasticsearch:9200"]
    index => "tfm-%{+YYYY.MM.dd}"
    template => "/opt/logstash/config/tfm.json"
    template_name => "tfm"
  }
}
```

8.8.7 Fichero tfm.json

```
{
```

```
"template" : "tfm",
"index_patterns" : "tfm-*",
"version" : 60001,
"settings" : {
  "index.refresh_interval" : "5s",
  "number_of_shards": 1
},
"mappings" : {
  "dynamic_templates" : [ {
    "message_field" : {
      "path_match" : "message",
      "match_mapping_type" : "string",
      "mapping" : {
        "type" : "text",
        "norms" : false
      }
    }
  }, {
    "string_fields" : {
      "match" : "*",
      "match_mapping_type" : "string",
      "mapping" : {
        "type" : "text", "norms" : false,
        "fields" : {
          "keyword" : { "type": "keyword", "ignore_above": 256 }
        }
      }
    }
  } ],
  "properties" : {
    "@timestamp": { "type": "date"},
    "@version": { "type": "keyword"},
    "geoip_orig_h"  : {
      "dynamic": true,
      "properties" : {
        "ip": { "type": "ip" },
        "location" : { "type" : "geo_point" },
        "latitude" : { "type" : "half_float" },
        "longitude" : { "type" : "half_float" }
      }
    },
     "geoip_resp_h"  : {
      "dynamic": true,
      "properties" : {
        "ip": { "type" : "ip" },
        "location" : { "type" : "geo_point" },
        "latitude" : { "type" : "half_float" },
        "longitude" : { "type" : "half_float" }
      }
    },
     "geoip_tx_hosts"  : {
      "dynamic": true,
      "properties" : {
        "ip": { "type" : "ip" },
        "location" : { "type" : "geo_point" },
        "latitude" : { "type" : "half_float" },
        "longitude" : { "type" : "half_float" }
      }
    },
```

```
       "geoip_answers"  : {
        "dynamic": true,
        "properties" : {
          "ip": { "type" : "ip" },
          "location" : { "type" : "geo_point" },
          "latitude" : { "type" : "half_float" },
          "longitude" : { "type" : "half_float" }
        }
      },
       "certificate"  : {
        "dynamic": true,
        "properties" : {
          "not_valid_after" : { "type" : "date" },
          "not_valid_before" : { "type" : "date" }
        }
      },
       "id.resp_h" : { "type" : "ip" },
       "id.orig_h" : { "type" : "ip" }
    }
  }
}
```

8.8.8 Fichero GeoIP.conf

```
AccountID <ID_CUENTA>
LicenseKey <CLAVE_LICENCIA>
EditionIDs GeoLite2-City
DatabaseDirectory /opt/geolite2
```

8.8.9 Fichero tasks/main.yml

```
---
- name: copia de los ficheros del servicio
  copy:
    src: "{{ role_path }}/files/docker/"
    dest: "{{ service_path }}"
- name : construcción del servicio
  docker_compose:
    project_name: logstash
    project_src: "{{ service_path }}"
    build: yes
- name : pausa antes de verificar el estado del servicio
  pause:
    seconds: 5
- name: verifica si se está ejecutando
  docker_container_info:
    name: logstash
  register: is_running
- name: comprobar la salida de la verificación
  assert:
    that:
      - is_running.exists
```

8.8.10 Fichero vars/main.yml

```
---
service_path: "{{ main_path }}/logstash"
```

8.9 Ansible – Rol postconfig

8.9.1 Estructura de ficheros

```
.
└── tasks
    └── main.yml
```

8.9.2 Fichero main.yml

```
---
- name: instalar yum-cron
  package:
    name: yum-cron
    state: present
- name: habilitar aplicar actualizaciones automáticamente
  lineinfile:
    path: /etc/yum/yum-cron.conf
    regexp: "^apply_updates = no"
    line: "apply_updates = yes"
    state: present
- name: reiniciar yum-cron
  service:
    name: yum-cron
    enabled: yes
    state: restarted
- name: deshabilitar la autenticación con contraseña
  lineinfile:
    path: /etc/ssh/sshd_config
    regexp: "^PasswordAuthentication"
    line: "PasswordAuthentication no"
    state: present
- name: deshabilitar el acceso como root
  lineinfile:
    path: /etc/ssh/sshd_config
    regexp: "^PermitRootLogin"
    line: "PermitRootLogin no"
    state: present
- name: reiniciar sshd
  service:
    name: sshd
    state: restarted
```

8.10 Ansible – Rol prerequisitos

8.10.1 Estructura de ficheros

```
.
├── files
│   └── auth_key.pub[6]
├── tasks
│   └── main.yml
└── vars
    └── main.yml
```

6 Este fichero se tiene que generar y ubicar en esta ruta con el nombre auth_key.pub

8.10.2 Fichero tasks/main.yml

```
---
- name : verifica modo promiscuo
  debug:
    msg:
      - '************************************************'
      - 'Modo promiscuo no habilitado en {{ nic_defecto }}'
      - '************************************************'
  when: ansible_facts[nic_defecto]['promisc'] == false
- name: añadir usuario tfm_mistic
  user:
    name: tfm_mistic
    groups: wheel
    password: '{{ tfm_mistic_password }}'
    state: present
- name: copiar clave pública SSH
  authorized_key:
    user: tfm_mistic
    state: present
    key: "{{lookup('file', '{{ role_path }}/files/auth_key.pub') }}"
- name: modificar vm.max_map_count a 262144 en sysctl
  sysctl:
    name: vm.max_map_count
    value: "262144"
    state: present
    reload: yes
- name: instalar repo EPEL
  package:
    name: epel-release
    state: latest
    update_cache: yes
- name: paquetes necesarios para Docker
  package:
    name: '{{ item }}'
    state: latest
  with_items:
    - yum-utils
    - device-mapper-persistent-data
    - lvm2
    - python2-pip
    - python-devel
    - gcc
    - gcc-c++
    - make
- name: añadir repositorio Docker
  get_url:
    url: 'https://download.docker.com/linux/centos/docker-ce.repo'
    dest: /etc/yum.repos.d/docker-ce.repo
  become: yes
- name: instalar Docker
  package:
    name: '{{ item }}'
    state: latest
  with_items:
    - docker-ce
    - docker-ce-cli
    - containerd.io
  become: yes
```

```
- name: actualizar pip
  pip:
    name: pip
    state: latest
  become: yes
- name: instalar docker-compose
  pip:
    name: docker-compose
    state: latest
  become: yes
- name: iniciar servicio Docker
  service:
    name: docker
    state: started
    enabled: yes
  become: yes
- name: obtiene la lista de contenedores
  docker_host_info:
    containers: yes
  register: containers_running
- name: detiene todos los contenedores en ejecución
  docker_container:
    name: "{{ item.Names[0] }}"
    state: stopped
  loop: "{{ containers_running.containers }}"
  when: containers_running.containers | length != 0
- name: limpieza completa de recursos Docker
  docker_prune:
    containers: yes
    images: yes
    images_filters:
      dangling: false
    networks: yes
    volumes: yes
    builder_cache: yes
  become: yes
```

8.10.3 Fichero vars/main.yml

```
---
# vars file for prerequisitos
nic_defecto: '{{ansible_default_ipv4.alias}}'
tfm_mistic_password: <CLAVE_USUARIO_TFM>
```

8.11 Ansible – Rol Threats

8.11.1 Estructura de ficheros

```
.
├── files
│   └── docker
│       ├── docker-compose.yml
│       ├── Dockerfile
│       └── scripts
│           ├── badips.py
│           ├── blocklistde.py
```

```
│            ├── danmeuk.py
│            ├── malwarebazaar.py
│            └── turris.py
├── tasks
│   └── main.yml
└── vars
    └── main.yml
```

8.11.2 Fichero docker-compose.yml

```
version: '3'
services:
  threats:
    build: .
    container_name: threats
    restart: always
    networks:
      - localnet
    volumes:
    - files:/home/scripter
volumes:
  files:
networks:
  localnet:
```

8.11.3 Fichero Dockerfile

```
FROM centos:7
LABEL maintainer "igalvit@uoc.edu"
RUN yum install -y epel-release \
    && yum update -y \
    && yum -y install python2-pip python-pandas crontabs python-
requests \
    && yum clean all \
    && rm -rf /var/cache/yum \
    && groupadd -g 1500 scripter \
    && useradd -u 1500 -g 1500 scripter
USER scripter:scripter
COPY --chown=scripter:scripter ./scripts /home/scripter
RUN (crontab -l ; echo "0 0-23/6 * * * python turris.py") | crontab -
\
    && (crontab -l ; echo "5 0-23/6 * * * python blocklistde.py") |
crontab - \
    && (crontab -l ; echo "10 0-23/6 * * * python badips.py") |
crontab - \
    && (crontab -l ; echo "20 0-23/6 * * * python danmeuk.py") |
crontab - \
    && (crontab -l ; echo "25 0-23/6 * * * python malwarebazaar.py")
| crontab -
WORKDIR /home/scripter
RUN python turris.py \
    && python blocklistde.py \
    && python badips.py \
    && python danmeuk.py \
    && python malwarebazaar.py
USER root:root
CMD ["crond", "-n"]
```

8.11.4 Fichero badips.py

```
import logging
import os.path
import urllib2
import csv
import sys
import zipfile
import pandas
import requests
from csv import writer
from csv import reader
# VARIABLES
startMessage = '****** BADIPS: Starting process ******'
phase1Message = 'BADIPS PHASE 1: Checking if folders exist.'
phase2Message = 'BADIPS PHASE 2: Zipping old cleaned file and
removing temp files.'
phase3Message = 'BADIPS PHASE 3: Downloading and processing data.'
errorMessage = ' BADIPS ERROR: Please, check logs in server'
finishMessage = '****** BADIPS: Process finished ******'
##LOG FILE
logFolder = 'logs'
logFile = 'badips.log'
logPath = os.path.join(logFolder, logFile)
## TEMP FILE
tmpFolder = 'tmp'
tmpFile = 'tmp_badips.csv'
tmpFilePath = os.path.join(tmpFolder, tmpFile)
## JOIN FILE
joinFile = 'tmp_join_badips.csv'
joinFilePath = os.path.join(tmpFolder, joinFile)
## CLEANED FILE
destFolder = 'downloads'
cleanedFile = 'badips.csv'
cleanedPath = os.path.join(destFolder, cleanedFile)
## ZIPPED FILE
zippedFolder = 'downloads'
zippedFile = 'badips.csv.old.zip'
zippedFilePath = os.path.join(zippedFolder, zippedFile)
## URLs
urls = {
  'ssh' : 'https://www.badips.com/get/list/ssh/5',
  'sql' : 'https://www.badips.com/get/list/sql/2',
  'rfi_attack' : 'https://www.badips.com/get/list/rfi-attack/2',
  'ddos' : 'https://www.badips.com/get/list/ddos/2',
  'ftp' : 'https://www.badips.com/get/list/ftp/2',
  'http' : 'https://www.badips.com/get/list/http/2',
  'dns' : 'https://www.badips.com/get/list/dns/2',
  'xmlrpc' : 'https://www.badips.com/get/list/xmlrpc/2',
  'proxy' : 'https://www.badips.com/get/list/proxy/2',
  'brute_force_login' :
'https://www.badips.com/get/list/bruteforce/2',
  'sip' : 'https://www.badips.com/get/list/voip/2',
  'bots' : 'https://www.badips.com/get/list/badbots/2'
}
# AUX FUNCTIONS
def checkLogFolder():
  """Checks if log folder exists. If not, creates it."""
  logFolderExists = os.path.isdir(logFolder)
```

```
  if not logFolderExists:
    try:
      os.mkdir(logFolder)
    except Exception as e:
      # If I cannot create log folder, I cannot log the error.
      pass
def checkFolder(folderName):
  """Checks if folder exists. If not, creates it."""
  logging.debug('PHASE 1: Checking %s folder exists.', folderName)
  folderExists = os.path.isdir(folderName)
  if not folderExists:
    logging.info('Creating %s folder', folderName)
    try:
      os.mkdir(folderName)
      logging.debug('%s folder created.', folderName)
    except Exception as e:
      logging.debug('Cannot create %s folder.', folderName)
      logging.error(str(e))
      sendMessageTelegram(errorMessage)
      sys.exit(0)
  else:
    logging.debug('Folder %s exists.', folderName)
def zipFile(zippedPath, filePath):
  """Compress as zip the old final file for use as a backup."""
  try:
    zipfile.ZipFile(zippedPath, mode='w').write(filePath,
compress_type=zipfile.ZIP_DEFLATED)
    logging.debug('Zipped file %s.', filePath)
  except Exception as e:
    logging.debug('Cannot zip file %s.', filePath)
    logging.error(str(e))
def setupLog():
  """Setups the config of logging."""
  logging.basicConfig(filename=logPath,
    format='%(asctime)s %(levelname)-2s %(message)s',
    level=logging.DEBUG,
    datefmt='%Y-%m-%d %H:%M:%S')
def downloadFile(url):
  """Downloads data file from an URL to a temp file."""
  logging.info('Downloading %s', url)
  try:
    df = pandas.read_csv(url, names=['Address'])
    df.to_csv(tmpFilePath, index=False)
  except Exception as e:
    logging.debug('Cannot download %s.', url)
    logging.error(str(e))
def addTag (tag):
  """Adds a new column named 'Tags' with the corresponding tag."""
  logging.info('Adding tag column "%s".', tag)
  try:
    df = pandas.read_csv(tmpFilePath)
    df["Tags"] = tag
    df.to_csv(joinFilePath, mode='a', index=False)
    logging.info('Tag column "%s" added.', tag)
  except Exception as e:
     logging.debug('Cannot add tag column %s.', tag)
     logging.error(str(e))
def deleteFile(filePath):
  """Deletes a file from disk."""
```

```
  logging.info('Deleting file %s.', filePath)
  try:
    os.remove(filePath)
    logging.debug('File deleted: %s.', filePath)
  except Exception as e:
    logging.debug('Cannot remove file %s.', filePath)
    logging.error(str(e))
def createJoinFile():
  """Creates a new CSV file to join proccessed data."""
  try:
    df = pandas.DataFrame(columns=["Address","Tags"])
    df.to_csv(joinFilePath, sep=",", index=False)
  except Exception as e:
    logging.debug('Cannot create join file.')
    logging.error(str(e))
    sendMessageTelegram(errorMessage)
    sys.exit(0)
def mergeData():
  logging.info("Merging all duplicate rows.")
  try:
    df = pandas.read_csv(joinFilePath, sep=",")
    df_combined = df.groupby(["Address"],
as_index=False).agg({"Tags":lambda x: ','.join(str(i) for i in
list(x))})
    df_combined.to_csv(cleanedPath, sep=",",
index=False,quoting=csv.QUOTE_NONNUMERIC)
  except Exception as e:
    logging.debug('Cannot merge all duplicate rows.')
    logging.error(str(e))
    sendMessageTelegram(errorMessage)
    sys.exit(0)
def sendMessageTelegram(message):
    """Sends a message to a Telegram Bot"""
    bot_token = '<BOT_TOKEN>'
    bot_chatID = '<BOT_CHATID>'
    send_text = 'https://api.telegram.org/bot' + bot_token +
'/sendMessage?chat_id=' + bot_chatID + '&parse_mode=Markdown&text=' +
message
    response = requests.get(send_text)
# MAIN FUNCTION
def main():
  checkLogFolder()
  setupLog()
  logging.info(startMessage)
  sendMessageTelegram(startMessage)
  logging.info(phase1Message)
  checkFolder(tmpFolder)
  checkFolder(destFolder)
  logging.info(phase2Message)
  zipFile(zippedFilePath, cleanedPath)
  deleteFile(tmpFilePath)
  deleteFile(joinFilePath)
  createJoinFile()
  logging.info(phase3Message)
  for k, v in urls.items():
   downloadFile(v)
   addTag(k)
   deleteFile(tmpFilePath)
  mergeData()
```

```
  deleteFile(joinFilePath)
  logging.info(finishMessage)
  sendMessageTelegram(finishMessage)
if __name__== "__main__":
  main()
```

8.11.5 Fichero blocklistde.py

```
import logging
import os.path
import urllib2
import csv
import sys
import zipfile
import pandas
import requests
from csv import writer
from csv import reader
# VARIABLES
startMessage = '****** BLOCKLISTDE: Starting process ******'
phase1Message = 'BLOCKLISTDE PHASE 1: Checking if folders exist.'
phase2Message = 'BLOCKLISTDE PHASE 2: Zipping old cleaned file and
removing temp files.'
phase3Message = 'BLOCKLISTDE PHASE 3: Downloading and processing
data.'
errorMessage = ' BLOCKLISTDE ERROR: Please, check logs in server'
finishMessage = '****** BLOCKLISTDE: Process finished ******'
##LOG FILE
logFolder = 'logs'
logFile = 'blocklistde.log'
logPath = os.path.join(logFolder, logFile)
## TEMP FILE
tmpFolder = 'tmp'
tmpFile = 'tmp_blocklistde.csv'
tmpFilePath = os.path.join(tmpFolder, tmpFile)
## JOIN FILE
joinFile = 'tmp_join_blocklistde.csv'
joinFilePath = os.path.join(tmpFolder, joinFile)
## CLEANED FILE
destFolder = 'downloads'
cleanedFile = 'blocklistde.csv'
cleanedPath = os.path.join(destFolder, cleanedFile)
## ZIPPED FILE
zippedFolder = 'downloads'
zippedFile = 'blocklistde.csv.old.zip'
zippedFilePath = os.path.join(zippedFolder, zippedFile)
## URLs
urls = {
  'ssh' : 'https://lists.blocklist.de/lists/ssh.txt',
  'mail' : 'https://lists.blocklist.de/lists/mail.txt',
  'http' : 'https://lists.blocklist.de/lists/apache.txt',
  'imap' : 'https://lists.blocklist.de/lists/imap.txt',
  'ftp' : 'https://lists.blocklist.de/lists/ftp.txt',
  'sip' : 'https://lists.blocklist.de/lists/sip.txt',
  'bots' : 'https://lists.blocklist.de/lists/bots.txt',
  'strong_ip' : 'https://lists.blocklist.de/lists/strongips.txt',
  'irc' : 'https://lists.blocklist.de/lists/ircbot.txt',
  'brute_force_login' :
'https://lists.blocklist.de/lists/bruteforcelogin.txt'
```

```
}
# AUX FUNCTIONS
def checkLogFolder():
  """Checks if log folder exists. If not, creates it."""
  logFolderExists = os.path.isdir(logFolder)
  if not logFolderExists:
    try:
      os.mkdir(logFolder)
    except Exception as e:
      # If I cannot create log folder, I cannot log the error.
      pass
def checkFolder(folderName):
  """Checks if folder exists. If not, creates it."""
  logging.debug('PHASE 1: Checking %s folder exists.', folderName)
  folderExists = os.path.isdir(folderName)
  if not folderExists:
    logging.info('Creating %s folder', folderName)
    try:
      os.mkdir(folderName)
      logging.debug('%s folder created.', folderName)
    except Exception as e:
      logging.debug('Cannot create %s folder.', folderName)
      logging.error(str(e))
      sendMessageTelegram(errorMessage)
      sys.exit(0)
  else:
    logging.debug('Folder %s exists.', folderName)
def zipFile(zippedPath, filePath):
  """Compress as zip the old final file for use as a backup."""
  try:
    zipfile.ZipFile(zippedPath, mode='w').write(filePath,
compress_type=zipfile.ZIP_DEFLATED)
    logging.debug('Zipped file %s.', filePath)
  except Exception as e:
    logging.debug('Cannot zip file %s.', filePath)
    logging.error(str(e))
def setupLog():
  """Setups the config of logging."""
  logging.basicConfig(filename=logPath,
    format='%(asctime)s %(levelname)-2s %(message)s',
    level=logging.DEBUG,
    datefmt='%Y-%m-%d %H:%M:%S')
def downloadFile(url):
  """Downloads data file from an URL to a temp file."""
  logging.info('Downloading %s', url)
  try:
    df = pandas.read_csv(url, names=['Address'])
    df.to_csv(tmpFilePath, index=False)
  except Exception as e:
    logging.debug('Cannot download %s.', url)
    logging.error(str(e))
def addTag (tag):
  """Adds a new column named 'Tags' with the corresponding tag."""
  logging.info('Adding tag column "%s".', tag)
  try:
    df = pandas.read_csv(tmpFilePath)
    df["Tags"] = tag
    df.to_csv(joinFilePath, mode='a', index=False)
    logging.info('Tag column "%s" added.', tag)
```

```
  except Exception as e:
     logging.debug('Cannot add tag column %s.', tag)
     logging.error(str(e))
def deleteFile(filePath):
  """Deletes a file from disk."""
  logging.info('Deleting file %s.', filePath)
  try:
    os.remove(filePath)
    logging.debug('File deleted: %s.', filePath)
  except Exception as e:
    logging.debug('Cannot remove file %s.', filePath)
    logging.error(str(e))
def createJoinFile():
  """Creates a new CSV file to join proccessed data."""
  try:
    df = pandas.DataFrame(columns=["Address","Tags"])
    df.to_csv(joinFilePath, sep=",", index=False)
  except Exception as e:
    logging.debug('Cannot create join file.')
    logging.error(str(e))
    sendMessageTelegram(errorMessage)
    sys.exit(0)
def mergeData():
  logging.info("Merging all duplicate rows.")
  try:
    df = pandas.read_csv(joinFilePath, sep=",")
    df_combined = df.groupby(["Address"],
as_index=False).agg({"Tags":lambda x: ','.join(str(i) for i in
list(x))})
    df_combined.to_csv(cleanedPath, sep=",",
index=False,quoting=csv.QUOTE_NONNUMERIC)
  except Exception as e:
    logging.debug('Cannot merge all duplicate rows.')
    logging.error(str(e))
    sendMessageTelegram(errorMessage)
    sys.exit(0)
def sendMessageTelegram(message):
    """Sends a message to a Telegram Bot"""
    bot_token = '811806487:AAFARMtEjMmI2ODeFXBbiXYlOmS6ry4vyG4'
    bot_chatID = '13904777'
    send_text = 'https://api.telegram.org/bot' + bot_token +
'/sendMessage?chat_id=' + bot_chatID + '&parse_mode=Markdown&text=' +
message
    response = requests.get(send_text)
# MAIN FUNCTION
def main():
  checkLogFolder()
  setupLog()
  logging.info(startMessage)
  sendMessageTelegram(startMessage)
  logging.info(phase1Message)
  checkFolder(tmpFolder)
  checkFolder(destFolder)
  logging.info(phase2Message)
  zipFile(zippedFilePath, cleanedPath)
  deleteFile(tmpFilePath)
  deleteFile(joinFilePath)
  createJoinFile()
  logging.info(phase3Message)
```

```
  for k, v in urls.items():
   downloadFile(v)
   addTag(k)
  mergeData()
  deleteFile(tmpFilePath)
  deleteFile(joinFilePath)
  sendMessageTelegram(finishMessage)
  logging.info(finishMessage)
if __name__== "__main__":
  main()
```

8.11.6 Fichero danmeuk.py

```
import logging
import os.path
import urllib2
import csv
import sys
import zipfile
import pandas
import requests
from csv import writer
from csv import reader
# VARIABLES
startMessage = '****** DANMEUK: Starting process ******'
phase1Message = 'DANMEUK PHASE 1: Checking if folders exist.'
phase2Message = 'DANMEUK PHASE 2: Zipping old cleaned file and
removing temp files.'
phase3Message = 'DANMEUK PHASE 3: Downloading and processing data.'
errorMessage = ' DANMEUK ERROR: Please, check logs in server'
finishMessage = '****** DANMEUK: Process finished ******'
##LOG FILE
logFolder = 'logs'
logFile = 'danmeuk.log'
logPath = os.path.join(logFolder, logFile)
## TEMP FILE
tmpFolder = 'tmp'
tmpFile = 'tmp_danmeuk.csv'
tmpFilePath = os.path.join(tmpFolder, tmpFile)
## JOIN FILE
joinFile = 'tmp_join_danmeuk.csv'
joinFilePath = os.path.join(tmpFolder, joinFile)
## CLEANED FILE
destFolder = 'downloads'
cleanedFile = 'danmeuk.csv'
cleanedPath = os.path.join(destFolder, cleanedFile)
## ZIPPED FILE
zippedFolder = 'downloads'
zippedFile = 'danmeuk.csv.old.zip'
zippedFilePath = os.path.join(zippedFolder, zippedFile)
## URLs
urls = {
  'tor' : 'https://www.dan.me.uk/torlist/'
}
# AUX FUNCTIONS
def checkLogFolder():
  """Checks if log folder exists. If not, creates it."""
  logFolderExists = os.path.isdir(logFolder)
  if not logFolderExists:
```

```
    try:
      os.mkdir(logFolder)
    except Exception as e:
      # If I cannot create log folder, I cannot log the error.
      pass
def checkFolder(folderName):
  """Checks if folder exists. If not, creates it."""
  logging.debug('PHASE 1: Checking %s folder exists.', folderName)
  folderExists = os.path.isdir(folderName)
  if not folderExists:
    logging.info('Creating %s folder', folderName)
    try:
      os.mkdir(folderName)
      logging.debug('%s folder created.', folderName)
    except Exception as e:
      logging.debug('Cannot create %s folder.', folderName)
      logging.error(str(e))
      sendMessageTelegram(errorMessage)
      sys.exit(0)
  else:
    logging.debug('Folder %s exists.', folderName)
def zipFile(zippedPath, filePath):
  """Compress as zip the old final file for use as a backup."""
  try:
    zipfile.ZipFile(zippedPath, mode='w').write(filePath,
compress_type=zipfile.ZIP_DEFLATED)
    logging.debug('Zipped file %s.', filePath)
  except Exception as e:
    logging.debug('Cannot zip file %s.', filePath)
    logging.error(str(e))
def setupLog():
  """Setups the config of logging."""
  logging.basicConfig(filename=logPath,
    format='%(asctime)s %(levelname)-2s %(message)s',
    level=logging.DEBUG,
    datefmt='%Y-%m-%d %H:%M:%S')
def downloadFile(url):
  """Downloads data file from an URL to a temp file."""
  logging.info('Downloading %s', url)
  try:
    df = pandas.read_csv(url, names=['Address'])
    df.to_csv(tmpFilePath, index=False)
  except Exception as e:
    logging.debug('Cannot download %s.', url)
    logging.error(str(e))
def addTag (tag):
  """Adds a new column named 'Tags' with the corresponding tag."""
  logging.info('Adding tag column "%s".', tag)
  try:
    df = pandas.read_csv(tmpFilePath)
    df["Tags"] = tag
    df.to_csv(joinFilePath, mode='a', index=False)
    logging.info('Tag column "%s" added.', tag)
  except Exception as e:
     logging.debug('Cannot add tag column %s.', tag)
     logging.error(str(e))
def deleteFile(filePath):
  """Deletes a file from disk."""
  logging.info('Deleting file %s.', filePath)
```

```
  try:
    os.remove(filePath)
    logging.debug('File deleted: %s.', filePath)
  except Exception as e:
    logging.debug('Cannot remove file %s.', filePath)
    logging.error(str(e))
def createJoinFile():
  """Creates a new CSV file to join proccessed data."""
  try:
    df = pandas.DataFrame(columns=["Address","Tags"])
    df.to_csv(joinFilePath, sep=",", index=False)
  except Exception as e:
    logging.debug('Cannot create join file.')
    logging.error(str(e))
    sendMessageTelegram(errorMessage)
    sys.exit(0)
def mergeData():
  logging.info("Merging all duplicate rows.")
  try:
    df = pandas.read_csv(joinFilePath, sep=",")
    df_combined = df.groupby(["Address"],
as_index=False).agg({"Tags":lambda x: ','.join(str(i) for i in
list(x))})
    df_combined.to_csv(cleanedPath, sep=",",
index=False,quoting=csv.QUOTE_NONNUMERIC)
  except Exception as e:
    logging.debug('Cannot merge all duplicate rows.')
    logging.error(str(e))
    sendMessageTelegram(errorMessage)
    sys.exit(0)
def sendMessageTelegram(message):
    """Sends a message to a Telegram Bot"""
    bot_token = '811806487:AAFARMtEjMmI2ODeFXBbiXYlOmS6ry4vyG4'
    bot_chatID = '13904777'
    send_text = 'https://api.telegram.org/bot' + bot_token +
'/sendMessage?chat_id=' + bot_chatID + '&parse_mode=Markdown&text=' +
message
    response = requests.get(send_text)
# MAIN FUNCTION
def main():
  checkLogFolder()
  setupLog()
  logging.info(startMessage)
  sendMessageTelegram(startMessage)
  logging.info(phase1Message)
  checkFolder(tmpFolder)
  checkFolder(destFolder)
  logging.info(phase2Message)
  zipFile(zippedFilePath, cleanedPath)
  deleteFile(tmpFilePath)
  deleteFile(joinFilePath)
  createJoinFile()
  logging.info(phase3Message)
  for k, v in urls.items():
   downloadFile(v)
   addTag(k)
   deleteFile(tmpFilePath)
  mergeData()
  deleteFile(joinFilePath)
```

```
  logging.info(finishMessage)
  sendMessageTelegram(finishMessage)
if __name__== "__main__":
  main()
```

8.11.7 Fichero malwarebazaar.py

```
import logging
import os.path
import urllib2
import csv
import sys
import zipfile
import pandas
import requests
import numpy
from csv import writer
from csv import reader
from io import BytesIO
from zipfile import ZipFile
# VARIABLES
startMessage = '****** MALWAREBAZAAR: Starting process ******'
phase1Message = 'MALWAREBAZAAR PHASE 1: Checking if folders exist.'
phase2Message = 'MALWAREBAZAAR PHASE 2: Zipping old cleaned file.'
phase3Message = 'MALWAREBAZAAR PHASE 3: Downloading and processing
data.'
errorMessage = ' MALWAREBAZAAR ERROR: Please, check logs in server'
finishMessage = '****** MALWAREBAZAAR: Process finished ******'
##LOG FILE
logFolder = 'logs'
logFile = 'malwarebazaar.log'
logFilePath = os.path.join(logFolder, logFile)
## TEMP FILE
tmpFolder = 'tmp'
tmpFile = 'tmp_malwarebazaar.csv'
tmpFilePath = os.path.join(tmpFolder, tmpFile)
## CLEANED FILE
destFolder = 'downloads'
cleanedFile = 'malwarebazaar.csv'
cleanedPath = os.path.join(destFolder, cleanedFile)
## ZIPPED FILE
zippedFolder = 'downloads'
zippedFile = 'malwarebazaar.csv.old.zip'
zippedFilePath = os.path.join(zippedFolder, zippedFile)
## URL
url = 'https://bazaar.abuse.ch/export/csv/full/'
# AUX FUNCTIONS
def checkLogFolder():
  """Checks if log folder exists. If not, creates it."""
  logFolderExists = os.path.isdir(logFolder)
  if not logFolderExists:
    try:
      os.mkdir(logFolder)
    except Exception as e:
      # If I cannot create log folder, I cannot log the error.
      pass
def checkFolder(folderName):
  """Checks if folder exists. If not, creates it."""
  logging.debug('Checking %s folder exists.', folderName)
```

```
  folderExists = os.path.isdir(folderName)
  if not folderExists:
    logging.info('Creating %s folder', folderName)
    try:
      os.mkdir(folderName)
      logging.debug('%s folder created.', folderName)
    except Exception as e:
      logging.debug('Cannot create %s folder.', folderName)
      logging.error(str(e))
      sendMessageTelegram(errorMessage)
      sys.exit()
  else:
    logging.debug('Folder %s exists.', folderName)
def zipFile(zippedPath, filePath):
  """Compress as zip the old final file for use as a backup."""
  try:
    zipfile.ZipFile(zippedPath, mode='w').write(filePath,
compress_type=zipfile.ZIP_DEFLATED)
    logging.debug('Zipped file %s.', filePath)
  except Exception as e:
    logging.debug('Cannot zip file %s.', filePath)
    logging.error(str(e))
def setupLog():
  """Setups the config of logging."""
  logging.basicConfig(filename=logFilePath,
    format='%(asctime)s %(levelname)-2s %(message)s',
    level=logging.DEBUG,
    datefmt='%Y-%m-%d %H:%M:%S')
def downloadFile(url):
  """Downloads data file from an URL to local file."""
  logging.info('Downloading %s', url)
  try:
    zipresp = urllib2.urlopen(url)
    with ZipFile(BytesIO(zipresp.read())) as zfile:
      for i, f in enumerate(zfile.filelist):
        f.filename = tmpFile
      zfile.extract(f, tmpFolder)
    #df = pandas.read_csv(url)
    #df.to_csv(tmpFilePath, index=False)
  except Exception as e:
    logging.debug('Cannot download %s.', url)
    logging.error(str(e))
    sendMessageTelegram(errorMessage)
    sys.exit(0)
def removeHashedLines(filePath):
  """Remove 8 first lines of the file that contains #"""
  logging.info("Cleaning first 8 lines with '#'")
  try:
    with open(filePath, 'r') as tmpFile:
      lines = tmpFile.readlines()
      realData = lines[8:-1]
      realData[0] = realData[0][2:]
      finalCleanedData = realData
    with open(filePath, 'w') as tmpFile:
      tmpFile.writelines(finalCleanedData)
  except Exception as e:
    logging.debug('Cannot clean hashes from %s.', filePath)
    logging.error(str(e))
    sendMessageTelegram(errorMessage)
```

```
    sys.exit(0)
def removeColumns():
  """Removes not necessary columns."""
  logging.info("Cleaning columns.")
  try:
    df = pandas.read_csv(tmpFilePath, error_bad_lines=False,
skipinitialspace=True, quotechar='"')
    keepCols = ['md5_hash', 'signature']
    cleaned = df[keepCols].replace(numpy.nan, 'Unknown', regex=True)
    cleaned.to_csv(cleanedPath, index=False,
quoting=csv.QUOTE_NONNUMERIC)
  except Exception as e:
    logging.debug('Cannot clean columns.')
    logging.error(str(e))
    sendMessageTelegram(errorMessage)
    sys.exit(0)
def deleteFile(filePath):
  """Deletes a file from disk."""
  logging.info('Deleting file %s.', filePath)
  try:
    os.remove(filePath)
    logging.debug('File deleted: %s.', filePath)
  except Exception as e:
    logging.debug('Cannot remove file %s.', filePath)
    logging.error(str(e))
def sendMessageTelegram(message):
    """Sends a message to a Telegram Bot"""
    bot_token = '<BOT_TOKEN>'
    bot_chatID = '<BOT_CHATID>'
    send_text = 'https://api.telegram.org/bot' + bot_token +
'/sendMessage?chat_id=' + bot_chatID + '&parse_mode=Markdown&text=' +
message
    response = requests.get(send_text)
# MAIN FUNCTION
def main():
  checkLogFolder()
  setupLog()
  logging.info(startMessage)
  sendMessageTelegram(startMessage)
  logging.info(phase1Message)
  checkFolder(destFolder)
  checkFolder(tmpFolder)
  deleteFile(tmpFilePath)
  logging.info(phase2Message)
  zipFile(zippedFilePath, cleanedPath)
  logging.info(phase3Message)
  downloadFile(url)
  removeHashedLines(tmpFilePath)
  removeColumns()
  deleteFile(tmpFilePath)
  logging.info(finishMessage)
  sendMessageTelegram(finishMessage)
if __name__== "__main__":
  main()
```

8.11.8 Fichero turris.py

```
import logging
import os.path
```

```
import urllib2
import csv
import sys
import zipfile
import pandas
import requests
from csv import writer
from csv import reader
# VARIABLES
startMessage = '****** TURRIS: Starting process ******'
phase1Message = 'TURRIS PHASE 1: Checking if folders exist.'
phase2Message = 'TURRIS PHASE 2: Zipping old cleaned file.'
phase3Message = 'TURRIS PHASE 3: Downloading and processing data.'
errorMessage = ' TURRIS ERROR: Please, check logs in server'
finishMessage = '****** TURRIS: Process finished ******'
##LOG FILE
logFolder = 'logs'
logFile = 'turris.log'
logFilePath = os.path.join(logFolder, logFile)
## TEMP FILE
tmpFolder = 'tmp'
tmpFile = 'tmp_turris.csv'
tmpFilePath = os.path.join(tmpFolder, tmpFile)
## CLEANED FILE
destFolder = 'downloads'
cleanedFile = 'turris.csv'
cleanedPath = os.path.join(destFolder, cleanedFile)
## ZIPPED FILE
zippedFolder = 'downloads'
zippedFile = 'turris.csv.old.zip'
zippedFilePath = os.path.join(zippedFolder, zippedFile)
## URL
url = 'https://project.turris.cz/greylist-data/greylist-latest.csv'
# AUX FUNCTIONS
def checkLogFolder():
  """Checks if log folder exists. If not, creates it."""
  logFolderExists = os.path.isdir(logFolder)
  if not logFolderExists:
    try:
      os.mkdir(logFolder)
    except Exception as e:
      # If I cannot create log folder, I cannot log the error.
      pass
def checkFolder(folderName):
  """Checks if folder exists. If not, creates it."""
  logging.debug('Checking %s folder exists.', folderName)
  folderExists = os.path.isdir(folderName)
  if not folderExists:
    logging.info('Creating %s folder', folderName)
    try:
      os.mkdir(folderName)
      logging.debug('%s folder created.', folderName)
    except Exception as e:
      logging.debug('Cannot create %s folder.', folderName)
      logging.error(str(e))
      sendMessageTelegram(errorMessage)
      sys.exit()
  else:
    logging.debug('Folder %s exists.', folderName)
```

```
def zipFile(zippedPath, filePath):
  """Compress as zip the old final file for use as a backup."""
  try:
    zipfile.ZipFile(zippedPath, mode='w').write(filePath,
compress_type=zipfile.ZIP_DEFLATED)
    logging.debug('Zipped file %s.', filePath)
  except Exception as e:
    logging.debug('Cannot zip file %s.', filePath)
    logging.error(str(e))
def setupLog():
  """Setups the config of logging."""
  logging.basicConfig(filename=logFilePath,
    format='%(asctime)s %(levelname)-2s %(message)s',
    level=logging.DEBUG,
    datefmt='%Y-%m-%d %H:%M:%S')
def downloadFile(url):
  """Downloads data file from an URL to local file."""
  logging.info('Downloading %s', url)
  try:
    df = pandas.read_csv(url)
    df.to_csv(tmpFilePath, index=False)
  except Exception as e:
    logging.debug('Cannot download %s.', url)
    logging.error(str(e))
    sendMessageTelegram(errorMessage)
    sys.exit(0)
def removeColumns():
  """Removes not necessary columns."""
  logging.info("Cleaning columns.")
  try:
    df = pandas.read_csv(tmpFilePath)
    keepCols = ['Address', 'Tags']
    cleaned = df[keepCols]
    cleaned.to_csv(cleanedPath, index=False,
quoting=csv.QUOTE_NONNUMERIC)
  except Exception as e:
    logging.debug('Cannot clean columns.')
    logging.error(str(e))
    sendMessageTelegram(errorMessage)
    sys.exit(0)
def deleteFile(filePath):
  """Deletes a file from disk."""
  logging.info('Deleting file %s.', filePath)
  try:
    os.remove(filePath)
    logging.debug('File deleted: %s.', filePath)
  except Exception as e:
    logging.debug('Cannot remove file %s.', filePath)
    logging.error(str(e))
def sendMessageTelegram(message):
    """Sends a message to a Telegram Bot"""
    bot_token = '<BOT_TOKEN>'
    bot_chatID = '<BOT_CHAT_ID>'
    send_text = 'https://api.telegram.org/bot' + bot_token +
'/sendMessage?chat_id=' + bot_chatID + '&parse_mode=Markdown&text=' +
message
    response = requests.get(send_text)
# MAIN FUNCTION
def main():
```

```
  checkLogFolder()
  setupLog()
  logging.info(startMessage)
  sendMessageTelegram(startMessage)
  logging.info(phase1Message)
  checkFolder(destFolder)
  checkFolder(tmpFolder)
  deleteFile(tmpFilePath)
  logging.info(phase2Message)
  zipFile(zippedFilePath, cleanedPath)
  logging.info(phase3Message)
  downloadFile(url)
  removeColumns()
  deleteFile(tmpFilePath)
  logging.info(finishMessage)
  sendMessageTelegram(finishMessage)
if __name__== "__main__":
  main()
```

8.11.9 Fichero tasks/main.yml

```
---
- name: copia de los ficheros del servicio
  copy:
    src: "{{ role_path }}/files/docker/"
    dest: "{{ service_path }}"
- name : construcción del servicio
  docker_compose:
    project_name: threats
    project_src: "{{ service_path }}"
    build: yes
- name : pausa antes de verificar el estado del servicio
  pause:
    seconds: 5
- name: verifica si se está ejecutando
  docker_container_info:
    name: threats
  register: is_running
- name: comprobar la salida de la verificación
  assert:
    that:
      - is_running.exists
```

8.11.10 Fichero vars/main.yml

```
---

service_path: "{{ main_path }}/threats"
```

8.12 Ansible – Rol Zeek

8.12.1 Estructura de ficheros

```
.
├── files
│   └── docker
│       ├── config
│       │   ├── entrypoint.sh
```

```
│       │   └── local.zeek
│       ├── docker-compose.yml
│       └── Dockerfile
├── tasks
│   └── main.yml
└── vars
    └── main.yml
```

8.12.2 Fichero docker-compose.yml

```
version: '3'
services:
  zeek:
    build: .
    container_name: zeek
    restart: always
    network_mode: host
    volumes:
    - etc:/opt/zeek/etc
    - logs:/opt/zeek/spool/zeek
    - site:/opt/zeek/share/zeek/site
volumes:
  etc:
  logs:
  site:
```

8.12.3 Fichero Dockerfile

```
FROM centos:7
LABEL maintainer "igalvit@uoc.edu"
RUN yum update -y \
    && yum -y install wget \
    && wget https://download.opensuse.org/repositories/security:zeek/
CentOS_7/security:zeek.repo -O /etc/yum.repos.d/zeek.repo \
    && yum update -y \
    && rpm --import
http://download.opensuse.org/repositories/security:/zeek/CentOS_7/
repodata/repomd.xml.key \
    && yum update -y \
    && yum -y install zeek-lts \
    && yum clean all && rm -rf /var/cache/yum
ENV PATH=/opt/zeek/bin/:$PATH
COPY ./config/entrypoint.sh /opt/zeek/entrypoint.sh
COPY ./config/local.zeek /opt/zeek/share/zeek/site/local.zeek
RUN chmod +x /opt/zeek/entrypoint.sh
WORKDIR /opt/zeek/logs
ENTRYPOINT ["/opt/zeek/entrypoint.sh"]
```

8.12.4 Fichero entrypoint.sh

```
#!/bin/bash
set -eox pipefail
zeekctl deploy
exec /bin/bash -c "trap : TERM INT; sleep infinity & wait"
```

8.12.5 Fichero local.zeek

```
##! Local site policy. Customize as appropriate.
##!
```

```
##! This file will not be overwritten when upgrading or reinstalling!
# This script logs which scripts were loaded during each run.
@load misc/loaded-scripts
# Apply the default tuning scripts for common tuning settings.
@load tuning/defaults
# Estimate and log capture loss.
@load misc/capture-loss
# Enable logging of memory, packet and lag statistics.
@load misc/stats
# Load the scan detection script.  It's disabled by default because
# it often causes performance issues.
#@load misc/scan
# Detect traceroute being run on the network. This could possibly
cause
# performance trouble when there are a lot of traceroutes on your
network.
# Enable cautiously.
#@load misc/detect-traceroute
# Generate notices when vulnerable versions of software are
discovered.
# The default is to only monitor software found in the address space
defined
# as "local".  Refer to the software framework's documentation for
more
# information.
@load frameworks/software/vulnerable
# Detect software changing (e.g. attacker installing hacked SSHD).oss
@load frameworks/software/version-changes
# This adds signatures to detect cleartext forward and reverse
windows shells.
@load-sigs frameworks/signatures/detect-windows-shells
# Load all of the scripts that detect software in various protocols.
@load protocols/ftp/software
@load protocols/smtp/software
@load protocols/ssh/software
@load protocols/http/software
# The detect-webapps script could possibly cause performance trouble
when
# running on live traffic.  Enable it cautiously.
#@load protocols/http/detect-webapps
# This script detects DNS results pointing toward your
Site::local_nets
# where the name is not part of your local DNS zone and is being
hosted
# externally.  Requires that the Site::local_zones variable is
defined.
@load protocols/dns/detect-external-names
# Script to detect various activity in FTP sessions.
@load protocols/ftp/detect
# Scripts that do asset tracking.
@load protocols/conn/known-hosts
@load protocols/conn/known-services
@load protocols/ssl/known-certs
# This script enables SSL/TLS certificate validation.
@load protocols/ssl/validate-certs
# This script prevents the logging of SSL CA certificates in x509.log
@load protocols/ssl/log-hostcerts-only
# Uncomment the following line to check each SSL certificate hash
against the ICSI
```

```
# certificate notary service; see http://notary.icsi.berkeley.edu .
@load protocols/ssl/notary
# If you have GeoIP support built in, do some geographic detections
and
# logging for SSH traffic.
@load protocols/ssh/geo-data
# Detect hosts doing SSH bruteforce attacks.
@load protocols/ssh/detect-bruteforcing
# Detect logins using "interesting" hostnames.
@load protocols/ssh/interesting-hostnames
# Detect SQL injection attacks.
@load protocols/http/detect-sqli
#### Network File Handling ####
# Enable MD5 and SHA1 hashing for all files.
@load frameworks/files/hash-all-files
# Detect SHA1 sums in Team Cymru's Malware Hash Registry.
@load frameworks/files/detect-MHR
# Extend email alerting to include hostnames
@load policy/frameworks/notice/extend-email/hostnames
# Uncomment the following line to enable detection of the heartbleed
attack. Enabling
# this might impact performance a bit.
@load policy/protocols/ssl/heartbleed
# Uncomment the following line to enable logging of connection VLANs.
Enabling
# this adds two VLAN fields to the conn.log file.
@load policy/protocols/conn/vlan-logging
# Uncomment the following line to enable logging of link-layer
addresses. Enabling
# this adds the link-layer address for each connection endpoint to
the conn.log file.
@load policy/protocols/conn/mac-logging
@load policy/tuning/json-logs.zeek
redef LogAscii::use_json = T;
```

8.12.6 Fichero tasks/main.yml

```
---
- name: copia de los ficheros del servicio
  copy:
    src: "{{ role_path }}/files/docker/"
    dest: "{{ service_path }}"
- name : construcción del servicio
  docker_compose:
    project_name: zeek
    project_src: "{{ service_path }}"
    build: yes
- name : pausa antes de verificar el estado del servicio
  pause:
    seconds: 5
- name: verifica si se está ejecutando
  docker_container_info:
    name: zeek
  register: is_running
- name: comprobar la salida de la verificación
  assert:
    that:
      - is_running.exists
```

8.12.7 Fichero vars/main.yml

```
---
service_path: "{{ main_path }}/zeek"
```