Universitat Oberta de Catalunya uoc.edu

IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN DE INTRUSOS IDS MEDIANTE LA INSPECCIÓN DEL TRÁFICO A TRAVÉS DE LA RED

ÓSCAR JURADO SANTIAGO

MÁSTER UNIVERSITARIO EN CIBERSEGURIDAD Y PRIVACIDAD
ANÁLISIS DE DATOS

Joan Caparrós Ramírez
Cristina Pérez Solá

Junio 2022

IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN DE INTRUSOS IDS MEDIANTE LA INSPECCIÓN DEL TRÁFICO A TRAVÉS DE LA RED

Autor: Óscar Jurado Santiago

Tutor: Joan Caparrós Ramírez

PRA: Cristina Pérez Solá

Área del Trabajo Final: Análisis de datos

Titulación: Máster Universitario en Ciberseguridad y Privacidad

Idioma: Español

Palabras clave: IDS, MikroTik, PYME, OpenSource

Resumen

Con este trabajo, se busca una solución para mitigar las intrusiones en las redes corporativas de autónomos, microPYMEs y PYMES, tanto por actores externos como por internos, teniendo en cuenta la capacidad de inversión en estos momentos postpandémicos.

Para ello, se usará un router MikroTik, que cada vez se encuentran más en entornos corporativos, y un servidor de 2ª mano que bien podría ser un servidor virtual en la infraestructura existente.

Las herramientas a utilizar son *Suricata*, un IDS ampliamente utilizado en el mundo de la Ciberseguridad y *Elastic Stack*, un paquete de herramientas de análisis y presentación de datos.

Después de las pruebas realizadas, se puede observar un amplio rango de alertas ante posibles intrusiones, que es la finalidad del proyecto.

IMPLEMENTATION OF AN INTRUSION DETECTION SYSTEM IDS BY INSPECTING TRAFFIC THROUGH THE NETWORK

Author: Óscar Jurado Santiago

Supervisor: Joan Caparrós Ramírez

Co-supervisor: Cristina Pérez Solá

Working area: Data analysis

Degree: MSc in Cybersecurity and Privacy

Language: Spanish

Keywords: IDS, MikroTik, SME, OpenSource

Abstract

With this project, a solution is sought to mitigate intrusions into the corporate networks of self-employed workers, micro-SMEs and SMEs, either by external and internal actors, taking into account the investment capacity in these post-pandemic moments.

To do this, a MikroTik router will be used, which is more and more found in corporate environments, and a second-hand server that could well be a virtual server over the existing infrastructure.

The tools we are going to use are *Suricata*, an IDS widely used in Cybersecurity world, and *Elastic Stack*, a package of data analysis and presentation tools.

After the tests being carried out, a wide range of alerts against possible intrusions can be observed, which is the purpose of the project.

Para mis dos orgullos,
Nacho y Álvaro.

Que observen cómo con diez lustros
se puede seguir aprendiendo...

Papá

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Explicación detallada del problema a resolver

En el ambiente empresarial, cada vez nos encontramos con que las intrusiones en las redes corporativas están a la orden del día, tanto por actores externos como por internos.

Si a ésto le sumamos la pléyade de dispositivos IoT que nos traen los avances tecnológicos (controladores de iluminación y calefacción, robots de limpieza, etc.), los periféricos conectados en red aparentemente “inofensivos” tales como impresoras, escáneres, puntos de acceso inalámbricos, etc., y los dispositivos personales (portátiles, móviles, tabletas...) que se permiten utilizar en/para el trabajo, según la tendencia empresarial de BYOD (tráete tu propio dispositivo, del inglés Bring Your Own Device), hacen que los departamentos de Sistemas y Seguridad de las empresas tengan cada vez más difícil controlar las vulnerabilidades y vectores de intrusión de tanto dispositivo, bien por falta de documentación de los mismos (como en la mayoría de dispositivos IoT), bien por no tener control sobre dispositivos que no pertenecen a la empresa.

Una de las herramientas que se utilizan para la detección de intrusiones son los sistemas IDS, que acompañados de una solución de análisis de los logs generados, ofrecen un dashboard o panel de conexiones y avisos ante detección de comportamientos malintencionados.

El coste de estas soluciones en entornos «propietarios» puede superar la capacidad de inversión de un autónomo, microPYME e incluso de una PYME,

y más aún en estos momentos que la pandemia del COVID-19 está forzando a reducir costes a límites inusitados.

Para intentar mitigar estas intrusiones, se buscará soluciones de código abierto (Open Source) para evitar el gasto en licencias, y la implantación en hardware de coste reducido, e incluso que pudiese estar presente en una estructura empresarial tipo.

La idea original es hacer la captura de los datos en un router MikroTik, que cada vez con mayor frecuencia se encuentra en las PYMES españolas, en lugar de utilizar un switch gestionable y un dispositivo externo para la captura de información.

Para la implantación del IDS y la herramienta de análisis y presentación de datos, se pretende utilizar un equipo de 2ª mano, que bien podría ser una máquina virtual en un servidor existente en un posible usuario empresarial.

1.2. Objetivos del Proyecto

El objetivo principal de este TFM es plantear una solución práctica para la detección de intrusos en una red mediante la inspección del tráfico a través de la misma, y para ello nos marcamos los siguientes subobjetivos:

- Investigación del estado del arte de sistemas IDS y de herramientas de análisis y presentación de datos.
- Selección de herramientas software y hardware, y estudio posibilidades de integración.
- Aprender a instalar y configurar el IDS seleccionado en la herramienta hardware elegida.
- Aprender a instalar y configurar la solución de análisis y presentación de datos seleccionadas.
- Refrescar la utilización de LaTeXpara redacción de documentos complejos.
- Desarrollar el Trabajo Fin de Máster.

1.3. Descripción de la metodología

El desarrollo del proyecto se planteó, en una primera aproximación, acometerlo en 4 fases coincidentes con las entregas parciales del mismo, que se ejecutarían secuencialmente entre sí y en paralelo con la documentación del mismo:

1. Definición del proyecto y primera aproximación temporal.
2. Investigación y toma de decisiones.
3. Implantación y pruebas.
4. Optimización, conclusiones y redacción de la memoria final.

Pero un proyecto es un ente dinámico, por lo que conforme a la evolución en el desarrollo del mismo se ha considerado que varias tareas pueden ejecutarse en paralelo o de forma no secuencial, pero sin perder la estructura base del mismo.

En cada fase, se irá actualizando la documentación generada, se comparará la planificación con respecto a la previsión inicial, se hará una evaluación del trabajo realizado, considerando posibles mejoras, decisiones tomadas y problemas encontrados.

Para la ejecución de cada una de las fases del proyecto, se ha adaptado una metodología tipo Kanban a las características de este proyecto. Para ello, se ha utilizando la herramienta Trello (`https://trello.com`), que se puede utilizar tanto en plataforma Web como en aplicación de escritorio o móvil.

1.4. Listado de las tareas a realizar

El proyecto en sí (investigación e implantación), se compone de 3 fases, más una previa de *Planificación* y una posterior de *Redacción de la memoria final*, terminando con la *Presentación y defensa del TFM*.

Para cada una de las fases del proyecto, definiremos las tareas a desarrollar.

1.4.1. Fase de investigación

Con los conocimientos previos que nos facilita la experiencia y las asignaturas de este Máster, **revisaremos el *estado del arte*** de la situación actual con respecto al proyecto a ejecutar.

Evaluaremos las herramientas que consideremos factibles, tanto hardware (software de base incluido) como herramientas software a utilizar, de forma que la conjunción de todas ellas sea viable **para la consecución del objetivo que hemos definido**.

Redactaremos un primer entregable en el que se exponga de forma razonada la decisión inicial tomada, y revisaremos la planificación de la ejecución del proyecto.

1.4.2. Fase de implantación

En esta fase, **instalaremos y configuraremos las herramientas escogidas** en la fase anterior, analizaremos la correcta integración de las mismas, y **evaluaremos la viabilidad** del proyecto con las decisiones tomadas en el punto anterior. Si consideráramos que el proyecto no fuese viable con tales decisiones, sería el momento de plantear una nueva decisión y ejecutarla. Posteriormente **se pondrá en marcha la solución implementada** para la recolección de datos con miras a la siguiente fase.

Redactaremos un segundo entregable, en el que reflejemos las experiencias observadas, tanto positivas como negativas, y volveremos a revisar la planificación del proyecto.

1.4.3. Fase de optimización y conclusiones

Con los datos obtenidos en la fase anterior, **analizaremos si hubiese que hacer algún ajuste y/o optimización** en la solución implementada, y **valoraremos las conclusiones** a las que el proyecto desarrollado nos conlleva.

Como último paso, procederemos a la **redacción final del TFM** y , la realización de un **vídeo explicativo** del mismo.

1.5. Planificación temporal detallada con diagrama de Gantt

Consideraremos para la planificación del Trabajo Fin de Máster que tiene una asignación de 12 créditos como asignatura y que cada crédito corresponde a 25 horas de dedicación, por lo que el total de horas estimadas de dedicación total al TFM debería ser de 300 horas.

Etapas	Tareas	Tiempo (horas)
Planificación		
	Definir objetivos	7
	Elaborar cronograma del trabajo	10
	Documentar	18
Investigación		
	Estudiar y seleccionar IDS	14
	Estudiar y seleccionar herram. análisis y presentación de datos	22
	Estudiar y seleccionar herramientas hardware	22
	Documentar	40
Implantación		
	Configurar hardware y S.O. Base	8
	Configurar IDS	10
	Configurar herram. análisis y presentación de datos	20
	Documentar	38
Optimización y redacción		
	Optimizar sistema	14
	Obtener conclusiones	7
	Redactar TFM	50
	Preparar vídeo	18
Presentación y defensa del TFM		
	Presentar el TFM en vídeo	1
	Defender el TFM	1
Total		**300**

Figura 1.1: Planificación temporal del TFM

En la planificación temporal (figura 1.1), se han incluido *ex-profeso* fines de semana, y se han excluido festividades, debido a que por las circunstancias laborales, familiares y sociales del autor del TFM, la dedicación al mismo será un

tanto atípica.

Esta primera aproximación temporal se irá reajustando (figura 1.2), pues las actividades de las fases de Investigación e Implantación podrían realizarse de forma cuasiparalela.

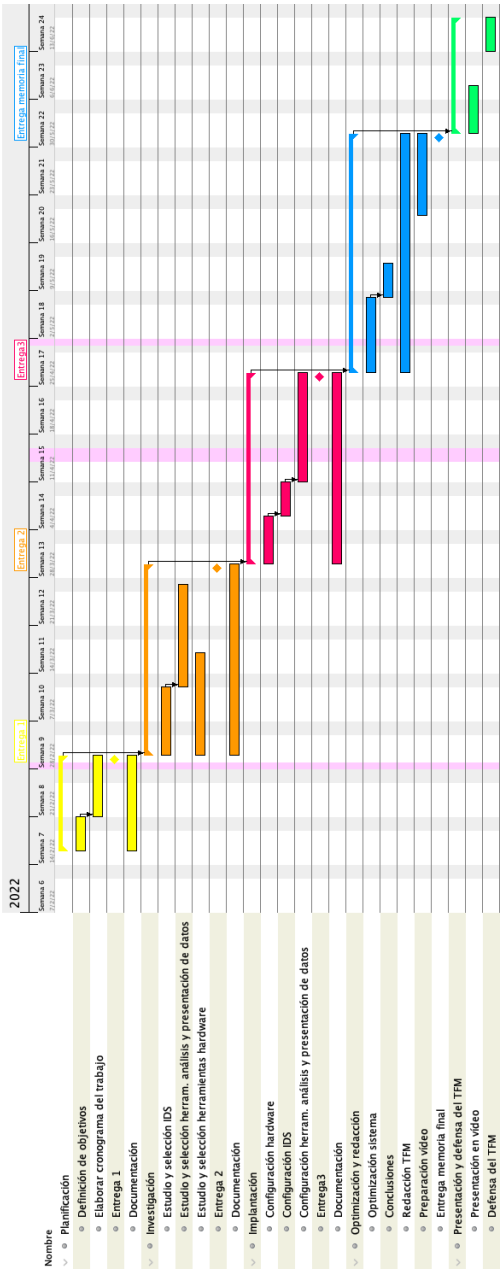

Figura 1.2: Diagrama de Gantt del proyecto

1.6. Revisión del estado del arte

Como una primera aproximación al proyecto, se ha realizado una breve investigación de la situación actual en el ámbito comercial que pudiera cubrir el campo de ejecución de este proyecto, así como de las soluciones Open Source que actualmente se encuentran disponibles.

Sistemas UTM comerciales. Los sistemas UTM son dispositivos físicos de red (algunos se ofrecen también en formato virtual) que combinan diversas funcionalidades de seguridad: cortafuegos, VPN, IDS/IPS, antivirus perimetral, filtrado de contenidos, antispam, antispyware, monitorización, sistema de alertas e informes, etc.

Varios de los servicios que prestan se ofrecen bajo suscripción anual.

Son soluciones que, en su gama de entrada, sólo el dispositivo y las funcionalidades básicas superan los 1.000€, costando los servicios bajo suscripción anualmente sobre un 50 % del valor de adquisición del producto.

Sistemas UTM Open Source. Hay algunas soluciones UTM Open Source, cuyos desarrolladores también las ofrecen en appliances físicos y/o servicios comerciales añadidos.

Sistemas IDS/IPS comerciales. También hay sistemas puros IDS/IPS comerciales, que se ofrecen en forma de dispositivo físico, y cuyo precio supera más aún al de un UTM.

Sistemas IDS Open Source. Los principales sistemas IDS Open Source que hemos encontrado se instalan como herramientas software sobre un Sistema Operativo base, y aunque algunos son multiplataforma, todos se encuentran disponibles bajo GNU/Linux, que es el S.O. recomendado para una solución de este tipo.

Tenemos que tener en consideración que podemos encontrar tanto HIDS como NIDS, siendo éstos últimos los interesantes para este proyecto.

Herramientas de análisis y presentación de datos comerciales. Generalmente, las empresas detrás de los desarrollos de las herramientas de análisis y presentación de datos Open Source ofrecen sus

soluciones y servicios en formato comercial, pero también hay soluciones únicamente comerciales.

Estas soluciones tienen un coste de partida que suele superar los 1.000€/mes.

Herramientas de análisis y presentación de datos Open Source. Para el análisis y presentación de datos, nos encontramos diversas herramientas que generalmente tenemos que integrar, y muy pocas vienen ya paquetizadas.

En el Apéndice A podemos ver la relación de soluciones que se han investigado para el estado del arte y para el desarrollo de este TFM.

1.7. Recursos y presupuesto del proyecto

Aunque para el proyecto se utilizará hardware que estaría disponible en una empresa tipo, vamos a hacer un presupuesto de los materiales a utilizar en el proyecto.

No se considerará el equipo utilizado para el desarrollo del proyecto, pues no es un gasto inherente al mismo.

En los recursos software, como la idea del proyecto es utilizar siempre que se pueda herramientas Open Source, el gasto en licencias será 0€.

MikroTik Router RB2011UiAS-RM	105,81€	(nuevo en Amazon)
HP Proliant MicroServer G8 (Intel G1610T, RAM 8GB, HD 2*2TB, 2*EthGB)	466,70€	(segunda mano en eBay)
2 Cables UTP Cat.6 1m	2,90€	(nuevo en Amazon)
Licencias software	0,00€	
Total:	**575,41€**	

Tabla 1.1: Presupuesto del proyecto

1.8. Análisis de riesgos

A la hora de analizar cuales son los principales riesgos que pudieran poner en peligro el proyecto, se han contemplado dos causísticas muy diferenciadas: por riesgos personales y por riesgos intrínsecos al proyecto.

Los riesgos personales no se suelen tener en cuenta y, por experiencia propia, pueden hacer que un proyecto se retrase de manera significativa, o incluso hacer que fracase.

Riesgos personales

1. **Problemas de salud**
   Una enfermedad grave, un accidente, una situación de pandemia, son riesgos reales que están a la orden del día y pueden forzar a un aplazamiento del TFM.

2. **Problemas laborales**
   Un cambio de trabajo, una situación de desempleo, son situaciones que nos podemos encontrar en el mundo real y pueden retrasar la ejecución del proyecto.

3. **Problemas familiares**
   Podemos encontrarnos con diversos problemas en el ámbito familiar que nos desajusten el calendario planificado.

Afectación: Muy alta.

Mitigación del riesgo: Estos tipos de riesgos, al ser excepcionales y que afectan al autor del proyecto, no podemos hacer nada por mitigarlos, solamente aplazar el TFM a otro semestre.

Intrínsecos al proyecto

5. **Retrasos en redacción del TFM**
   Con afán de avanzar más rápido de la cuenta en el desarrollo del

proyecto, nos podemos encontrar que la redacción del TFM se vea retrasada, o bien que no se recuerde qué y por qué se hizo determinada tarea.

Afectación: Alta

Mitigación del riesgo: Ir documentando cada una de las fases del proyecto conforme se vaya avanzando en la investigación o implantación.

6. **Objetivo demasiado ambicioso/extenso**
   Al realizar el proyecto, podemos perdernos profundizando o extendiéndonos en temas que provocarían un sobreexceso en la cantidad de tiempo estimado.

   Afectación: Media

   Mitigación del riesgo: Ceñirnos a la planificación temporal, acotando el tiempo dedicado a cada tarea y evitando profundizar más de lo estrictamente necesario.
   En caso necesario, reducir el nivel de los objetivos no prioritarios.

7. **Problemas de integración entre herramientas**
   Nos podemos encontrar que las herramientas elegidas no se integren correctamente las unas con las otras.

   Afectación: Baja

   Mitigación del riesgo: Analizar correctamente las herramientas seleccionadas, y en caso necesario, cambiar las selección de las mismas.

8. **Problemas en implantación**
   Al ejecutar el proyecto, nos podemos encontrar diversos errores en la implantación del mismo, bien en recursos hardware como en S.O. base o herramientas seleccionadas.

Afectación: Baja

Mitigación del riesgo: Con respecto al hardware, no se prevé cambios aparte de los estrictamente necesarios debido a su coste, por lo que habría que adaptar los demás elementos.
Con respecto al S.O. y/o herramientas, seleccionar y utilizar alguna alternativa de las evaluadas, que mirándolo por el lado positivo nos serviría como aprendizaje (otro de los objetivos del proyecto).

9. **Pérdida de documentación o de configuración**
   Por un fallo hardware, software o humano, podemos perder todo lo que tenemos avanzado en el TFM, tanto de redacción de memoria como de implantación del proyecto.

   Afectación: Muy alta

   Mitigación del riesgo: Copias de seguridad periódicas.

1.9. Implicaciones éticas y legales

Desde un punto de vista ético, la principal implicación es que hay parte de los datos de las comunicaciones de la red que van a ser almacenados, por lo que los usuarios de la misma deberían ser conscientes de dicha acción. Para ello, la empresa debería comunicárselo y el uso que se le va a dar a los mismos.

Desde un punto de vista legal, la principal implicación viene desde el mismo momento que tenemos que cumplir con el RGPD/LOPDGDD.

Capítulo 2

Investigación

2.1. Diseño de la infraestructura

Para comenzar el desarrollo del proyecto en sí, primero hay que definir el entorno de la infraestructura que vamos a utilizar.

Partiremos de una infraestructura tipo que podríamos encontrarnos en una PYME y después ir avanzando en las alternativas, pues dependiendo de cómo y dónde incorporemos los elementos, el resultado va a ser completamente distinto.

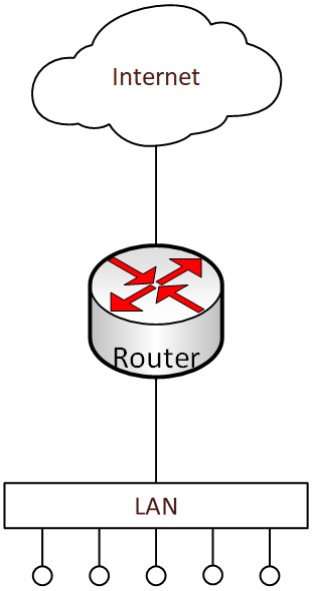

Figura 2.1: Infraestructura típica en una PYME

La decisión clave va a estar en el lugar donde coloquemos el IDS: en la red externa (WAN) o en la red interna (LAN).

Veamos las características y resultados que obtendríamos en cada una de las opciones:

- **IDS en red externa(WAN)**
  En este caso, vamos a capturar el tráfico entrante/saliente de la boca WAN del router.
  Vamos a detectar, por un lado, los ataques contra nuestra infraestructura que provengan de Internet y, por otro, los ataques que partan de nuestra red como podría ser un ordenador/red *zombie* o un equipo comprometido.

- **IDS en red interna (LAN)**
  Aquí el tráfico a capturar es el generado dentro de nuestra red local, así como el tráfico entrante (potencialmente «filtrado» por nuestro router) que provenga de Internet.

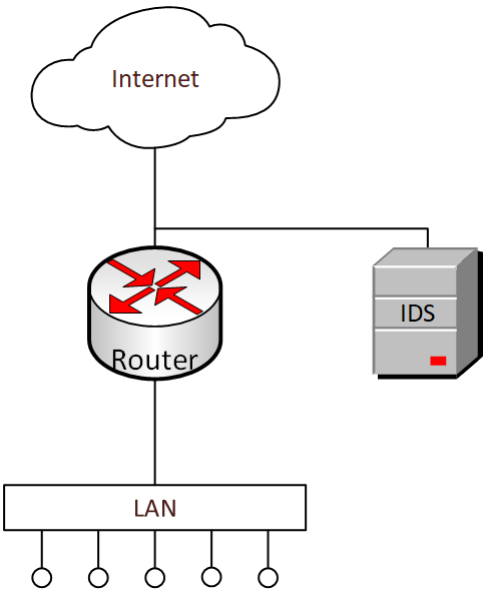

Figura 2.2: Infraestructura con IDS en WAN

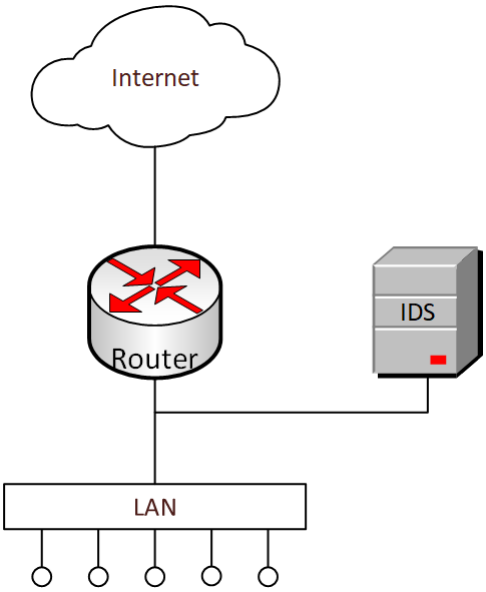

Figura 2.3: Infraestructura con IDS en LAN

Para limitar el alcance de nuestro proyecto, vamos a instalar nuestro IDS en la zona LAN de la red, dejando como una mejora del mismo la instalación en la zona WAN o en ambas.

En la sección 4.1 «Líneas de trabajo futuro» reflejamos esta idea de ampliación.

2.2. Opciones de configuración del MikroTik

Para el envío del tráfico desde el router MikroTik hacia el IDS, tenemos diversas opciones a evaluar:

1. **Port Mirroring**
   El concepto de *port mirroring* es el duplicado de puertos «tradicional» en gestión de redes, en el que se hace espejo de todo el tráfico de una boca de red a otra.
   En el caso del MikroTik 2011, ésto se realiza a nivel del chip de switch Atheros8327, lo que redunda en un mínimo consumo de CPU en el router.

Como inconveniente, que sólo podemos hacer mirroring de un puerto a otro, no de conjuntos de puertos.

2. **Packet Sniffer**
   El sistema operativo de los equipos MikroTik (RouterOS) tiene integrado una utilidad para la captura de paquetes.
   Packet sniffer es una herramienta que puede capturar y/o analizar paquetes que entran, salen o atraviesan el router, aunque el filtrado que permite es básico.
   Permite hacer streaming de paquetes en tiempo real hacia un dispositivo IP para su tratamiento. Para ello encapsula el tráfico con el protocolo TZSP (que va sobre UDP).

3. **Marcado *Mangle***
   RouterOS posee una herramienta en el módulo de Firewall llamada *Mangle* que permite marcar paquetes de manera especial para un procesado posterior. Estas marcas son usadas dentro de otros módulos del router como pueden ser árboles de colas (*queue trees*), NAT, routing... Dichos módulos identifican los paquetes basándose en sus marcas y los procesan acorde con ellas.
   Las marcas *mangle* existen sólo dentro del router, y no son transmitidas a través de la red.
   Adicionalmente la funcionalidad Mangle permite modificar ciertos campos en el encabezado IP como el TOS (DSCP) o el TTL.
   También permite, lo que es lo interesante para este proyecto, streaming de paquetes por protocolo TZSP.

4. **Traffic-Flow**
   Es la implementación de MikroTik del protocolo de red NetFlow de Cisco.
   Traffic-Flow es un sistema que proporciona información estadística sobre los paquetes que pasan por el router. Permite mediciones de tráfico granulares y precisas, así como recopilación de tráfico agregado de alto nivel.
   Analizando esta información, los administradores de red pueden identificar diversos problemas que puedan ocurrir en la red y optimizar el rendimiento general de ésta, como detectar causas de congestión de la red, identificar redes de origen y destino, etc.

La opción 4, aun pareciendo la más interesante, no sería la adecuada para este proyecto, pues aunque ha habido estudios sobre IDS utilizando datos

de NetFlow [8], está más encaminado a detectar problemas en la red que a la detección de intrusiones.

La solución más rápida es la opción 1, pero el principal inconveniente es que restringimos la captura de información a un puerto, y no tenemos opción alguna de filtrado.

Las opciones 2 y 3, simplificando, se diferencian en el nivel de filtrado que, mientras en la primera es básico, en la segunda permite muchísimas más opciones.

El principal inconveniente para ambas es que los IDS OpenSource evaluados no contemplan el protocolo TZSP como fuente de entrada de datos, por lo que tendríamos que implementar un «conversor» para convertir el flujo de TZSP a paquetes en un puerto virtual.

2.3. Evaluación de los sistemas IDS

Como bien indica el título de éste TFM, *Implementación de un sistema de detección de intrusos IDS mediante la inspección del tráfico a través de la red*, no tendría sentido la implantación de un HDIS, por lo que vamos a centrarnos en las soluciones NDIS.

Si nos fijamos en la tabla A.4 «Principales Sistemas IDS Open Source», nos hemos centrado en 4 soluciones NIDS:

- Snort
- Suricata
- Zeek
- Kismet

2.3.1. Snort

Snort es el veterano de los IDS/IPS Open Source. No en vano ha sido durante mucho tiempo el estándar *de facto* de los sistemas IDS.

Creado originalmente en 1998, su característica original era la utilización de reglas para la detección de amenazas emergentes, que otros sistemas copiaron posteriormente.

Sus principales ventajas son su amplia adopción junto con la gran comunidad que lo mantiene, lo que lleva a que haya sido probado exhaustivamente y sea muy fiable.

Y su principal inconveniente es precisamente «su edad»: adolece de características que otras alternativas posteriores poseen desde hace tiempo, como la capacidad multithreading y el manejo de IPv6.

2.3.2. Suricata

Algo más moderno es *Suricata*, cuya primera versión data de 2009, creada con la idea de adaptarse a las nuevas demandas de mayor rendimiento que las infraestructuras mas modernas necesitaban.

Aunque puede utilizar la mayoría de reglas de Snort, incorpora un lenguaje de scripting, *Lua*, que le proporciona mayor flexibilidad y capacidades.

Entre sus ventajas, además del uso de Lua, está el alto rendimiento, con multi-threading y el soporte nativo de aceleración hardware (como p.e. de tarjetas gráficas) y la detección automática de protocolos.

2.3.3. Zeek

Zeek, antes conocido como *Bro*, liberó su primera versión en 1995, pero empezó a recibir atención en 2010 a raiz de una subvención de la *National Science Foundation (NSF)*.

Mientras que Snort y Suricata son sistemas IDS/IPS, Zeek es, según ellos mismos, un «analizador de tráfico de red» que puede ser utilizado como IDS, pero adolece de funcionalidad IPS.

La filosofía detrás de Zeek es distinta a los sistemas anteriores; utiliza scripts en lugar de reglas para analizar el tráfico, lo que facilita un espectro más amplio para detectar actividades maliciosas, incluyendo detección semántica de usos indebidos, deteccion de anomalías y análisis de comportamiento.

Otra ventaja es su diseño para soportar «Zeek Clusters», es decir, que está preparado para ofrecer alto rendimiento soportando balanceo de carga escalable, lo que le permite analizar grandes redes empresariales.

Los principales inconvenientes de Zeek son su complicada configuración y su elevada curva de aprendizaje, además de un gran consumidor de recursos (debido a la inspección profunda de paquetes o *deep-packet inspection*).

2.3.4. Kismet

Kismet es un IDS especializado en redes inalámbricas, que aun siendo muy interesante por las características distintivas que tienen las redes inalámbicas, no sería de aplicación para este proyecto.

2.4. Evaluación de las herramientas de análisis y presentación de datos

Aunque al hacer la «Revisión del estado del arte» se vio que hay una variada diversidad para elegir una herramienta de análisis y presentación de los datos obtenidos por el IDS, hay una que por su proyección pasada y futura es una candidata ideal para proyectos futuros del autor de este TFM, por lo que se ha decidido utilizar **Elastic Stack**.

2.5. Toma de decisiones

Tras el análisis de las distintas soluciones IDS, se acotó a dos sistemas: Suricata y Zeed.

Al final. la decisión recayó en **Suricata** por dos motivos principales:

- el sistema de reglas de Snort ya era conocido (por asignaturas del máster y por la experiencia profesional) mientras que la curva de aprendizaje de Zeek es muy alta
- los menores requerimientos hardware de Suricata frente a Zeek

Tras haber decidido la instalación de **Suricata** como IDS y de **Elastic Stack** como herramienta de análisis y presentación de los datos, la siguiente decisión recae sobre qué sistema operativo montarlo.

La decisión es clara: **GNU/Linux**; por el ahorro de licencias, la disponibilidad de los paquetes elegidos, la gran documentación y foros que hay para consulta, la estabilidad del sistema, etc.

En un primer momento, el montar una solución UTM que lo integrase todo fue la decisión elegida. Para ello se realizó una prueba con **Security Onion**.

Esta solución, aun pareciendo muy prometedora, hubo que descartarla después de las primeras pruebas, pues aunque se puedan deshabilitar los módulos no necesarios, necesita para correr con cierta fluidez unos requisitos hardware más elevados que los contemplados en el presupuesto del presente proyecto.

Por ello, la alternativa más factible y la que se utilizó definitivamente es montar un sistema base limpio, en modo consola, y sobre él instalar y configurar las herramientas necesarias. La distribución GNU/Linux elegida es **Debian**, con la que el autor de este TFM tiene experiencia previa.

Capítulo 3

Implantación

3.1. Primera prueba: Security Onion

La solución **Security Onion** fue la primera opción para este proyecto, y podemos encontrarla en `https://securityonionsolutions.com/software`

Tal y como indican en su página web:

> «Security Onion en una distribución Linux libre y abierta para la búsqueda de amenazas, la supervisión de la seguridad empresarial y la gestión de registros.
>
> Security Onion incluye Elasticsearch, Logstash, Kibana, Suricata, Zeek (antes conocida como Bro), Wazuh, Stenographer, CyberChef, NetworkMiner, y muchas otras herramientas de seguridad.»

En un solo paso, nos encontramos con un GNU/Linux, Suricata y Elastic Stack instalados, pero al tener tantos módulos preinstalados, además de estar Linux en modo gráfico, el servidor funcionaba de forma inusitadamente lenta.

Se deshabilitaron todos los paquetes no imprescindibles, pero aún así, no funcionaba con la suficiente fluidez: los requisitos hardware deberían ser mayores que los del servidor utilizado en el proyecto.

Por esa razón, se tomó la decisión de dar un paso atrás y descartarla, procediendo a montar lo estrictamente necesario.

3.2. Instalación y configuración SO base

Después de descartar la alternativa anterior, y como ya se indicó en la sección 2.5 el siguiente paso sería la instalación de Debian.

Para ello descargamos la última versión estable de 64 bits de Debian GNU/-Linux 11.3 (nombre en clave *bullseye*) en versión *netinstall* desde la página oficial:

`https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/debian-11.3.0-amd64-netinst.iso`

Una vez descargada en nuestra máquina (corriendo Linux), creamos un disco de instalación en una memoria USB con los comandos:

```
root@deathstar:~# cp debian.iso /dev/sdc

root@deathstar:~# sync
```

siendo `sdc` el dispositivo de la unidad USB.[1]

La instalación la realizamos en modo texto, instalando sólo los paquetes básicos, para hacer más ligero el servidor.

En el paso de la configuración de red, al disponer de dos puertos GB-Ethernet, elegimos como interfaz primaria la correspondiente a la boca Ethernet1 (`eno1`) y la configuramos con la IP 192.168.1.10/24 y puerta de enlace 192.168.1.1. Este puerto Ethernet será el que utilicemos para acceder y configurar el servidor. Más adelante configuraremos la segunda interfaz.

A la hora de instalar componentes adicionales, el único que se marcará sera el de «Servidor SSH» para poder conectarnos remotamente de forma segura y tener un servidor que podamos administrar sin pantalla y teclado.

Una vez instalado el S.O., se configuró la boca Ethernet2 (`eno2`) y la configuramos con la IP 192.168.1.11/24, cuya única función será el recibir los datos a analizar.

Finalizado la instalación de los paquetes básicos necesarios y la con-

[1] La imagen debe escribirse en el dispositivo del disco completo y no en una partición, p. ej. `/dev/sdc` y no `/dev/sdc1`

figuración de las tarjetas de red, actualizamos todos los paquetes Debian instalados:

```
root@tfm:~# apt-get update

root@tfm:~# apt-get upgrade
```

3.3. Instalación y configuración de Suricata

En los repositorios de Debian se encuentra la versión 6.0.1 de Suricata, pero la última versión estable de Suricata es la 6.0.4, por lo que vamos a instalar esta última.

Debian es una distribución GNU/Linux conocida por su fiabilidad, y por ello los paquetes de los repositorios de sus versiones *stable* pueden ser algo antiguos. Para corregir ésto, un grupo de trabajo de Debian han creado un repositorio llamado *backports* para que las versiones Debian GNU/Linux de la rama *stable* puedan encontrar las versiones estables más recientes de los paquetes y programas más habituales.

Para utilizar esta característica, y siguiendo las indicaciones de la web de Suricata, instalamos la última versión desde los *backports*:

```
root@tfm:~# echo "deb http://deb.debian.org/debian bullseye-backports main" >
    /etc/apt/sources.list.d/backports.list

root@tfm:~# apt-get update

root@tfm:~# apt-get install suricata -t bullseye-backports
```

Comprobamos la versión de Suricata que se ha instalado con `suricata --build-info` y el estado del servicio con `systemctl status suricata`:

```
root@tfm:~# suricata --build-info
This is Suricata version 6.0.4 RELEASE
...
```

```
root@tfm:~# systemctl status suricata
 suricata.service - Suricata IDS/IDP daemon
     Loaded: loaded (/lib/systemd/system/suricata.service; enabled; vendor pres>
     Active: active (running) since Mon 2022-04-02 19:22:28 CEST; 5s ago
       Docs: man:suricata(8)
             man:suricatasc(8)
             https://suricata-ids.org/docs/
    Process: 2195 ExecStart=/usr/bin/suricata -D --af-packet -c /etc/suricata/s>
   Main PID: 2197 (Suricata-Main)
      Tasks: 8 (limit: 9442)
     Memory: 47.0M
        CPU: 440ms
     CGroup: /system.slice/suricata.service
             2197 /usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.>

apr 02 19:22:28 tfm systemd[1]: Starting Suricata IDS/IDP daemon...
apr 02 19:22:28 tfm suricata[2195]: 2/4/2022 -- 19:22:28 - <Notice> - This is S>
apr 02 19:22:28 tfm systemd[1]: Started Suricata IDS/IDP daemon.
```

Antes de iniciar la configuración de Suricata, recordemos la interfaz y la dirección IP sobre el que vamos a configurarlo:

```
root@tfm:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
    default qlen 1000
    link/ether 01:23:45:67:89:ab brd ff:ff:ff:ff:ff:ff
    altname enp3s0f0
    inet 192.168.1.10/24 brd 192.168.1.255 scope global eno1
       valid_lft forever preferred_lft forever
    inet6 fe80::b25a:daff:fe87:8034/64 scope link
       valid_lft forever preferred_lft forever
3: eno2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
    default qlen 1000
    link/ether 01:23:45:67:89:cd brd ff:ff:ff:ff:ff:ff
    altname enp3s0f1
```

```
    inet 192.168.1.11/24 brd 192.168.1.255 scope global eno2
       valid_lft forever preferred_lft forever
    inet6 fe80::b25a:daff:fe87:8035/64 scope link
       valid_lft forever preferred_lft forever
```

Suricata por defecto viene configurado con la interfaz de red `eth0`.

Debemos remplazar `eth0` de toda configuración en el archivo principal de suricata, y cambiar por nuestra interfaz (en nuestro caso `eno2`).

Para mejorar la precisión y el rendimiento, también ajustaremos la dirección de la red.

Y como vamos a utilizarlo en modo IDS, activamos las 2 últimas líneas (`use-nmap` y `tpacket-v3`).

Editamos estos cambios en el fichero de configuración de Suricata que se encuentra en

`/etc/suricata/suricata.yaml`

```
    vars:
      # more specific is better for alert accuracy and performance
      address-groups:
        HOME_NET: "[192.168.1.0/24]"

    # Linux high speed capture support
    af-packet:
      - interface: eno2
          cluster-id: 99
          cluster-type: cluster_flow
          defrag: yes
          use-mmap: yes
          tpacket-v3: yes
```

Reiniciamos el servicio y comprobamos que está funcionando sin errores.

```
root@tfm:~# systemctl restart suricata

root@tfm:~# systemctl status suricata
```

Actualizamos reglas con `suricata-update` y reiniciamos con las nuevas reglas:

```
root@tfm:~# suricata-update update-sources

2/4/2022 -- 19:24:28 - <Info> -- Using data-directory /var/lib/suricata.
2/4/2022 -- 19:24:28 - <Info> -- Using Suricata configuration
    /etc/suricata/suricata.yaml
2/4/2022 -- 19:24:28 - <Info> -- Using /etc/suricata/rules for Suricata
    provided rules.
2/4/2022 -- 19:24:28 - <Info> -- Found Suricata version 6.0.4 at
    /usr/bin/suricata.
2/4/2022 -- 19:24:28 - <Info> -- Saved
    /var/lib/suricata/update/cache/index.yaml

root@tfm:~# suricata-update list-sources

root@tfm:~# suricata-update enable-source oisf/trafficid
root@tfm:~# suricata-update enable-source sslbl/ssl-fp-blacklist
root@tfm:~# suricata-update enable-source ptresearch/attackdetection


root@tfm:~# suricata-update list-enabled-sources
2/4/2022 -- 19:26:48 - <Info> -- Using data-directory /var/lib/suricata.
2/4/2022 -- 19:26:48 - <Info> -- Using Suricata configuration
    /etc/suricata/suricata.yaml
2/4/2022 -- 19:26:48 - <Info> -- Using /etc/suricata/rules for Suricata
    provided rules.
2/4/2022 -- 19:26:48 - <Info> -- Found Suricata version 6.0.4 at
    /usr/bin/suricata.
Enabled sources:
  - oisf/trafficid
  - ptresearch/attackdetection
  - sslbl/ssl-fp-blacklist
  - et/open

root@tfm:~# suricata-update

root@tfm:~# systemctl restart suricata
```

Comprobamos si se arranca con el sistema:

```
root@tfm:~# systemctl is-enabled suricata
enabled
```

3.4. Elastic Stack

Para instalar los componentes de *Elastic Stack*, antes conocido como *ELK Stack*, tenemos dos alternativas:

- Descargar los paquetes desde la web de Elastic e instalarlos con `dpkg`
- Usar los repositorios de Elastic para instalarlos con las herramientas de Debian

Vamos a utilizar la segunda opción.

Para ello, hay que realizar unos pasos previos:

1. **Importar la clave PGP de Elastic.**
   Elastic firma todos sus paquetes con una clave PGP que descargamos e instalamos:

   ```
   root@tfm:~# wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch
       | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg
   ```

2. **Configurar el repositorio APT de Elastic en Debian.**
   Tenemos que instalar previamente el paquete `apt-transport-https` en Debian, grabar la definición del repositorio en `/etc/apt/sources.list.d/elastic-8.x.list` y finalmente actualizar el listado de los paquetes de los repositorios de nuestro Debian:

   ```
   root@tfm:~# sudo apt-get install apt-transport-https

   root@tfm:~# echo "deb
       [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg]
       https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee
       /etc/apt/sources.list.d/elastic-8.x.list
   ```

```
root@tfm:~# sudo apt-get update
```

Y ya estamos en disposición de instalar los componentes estables de Elastic Stack como de cualquier otro paquete Debian.

3.4.1. Elasticsearch

Al tener configurado el repositorio de *Elastic Stack* en nuestro Debian, instalamos Elasticsearch con «`apt-get install elasticsearch`»:

```
root@tfm:~# apt-get install elasticsearch
...
Desempaquetando elasticsearch (8.1.3) ...
Configurando elasticsearch (8.1.3) ...
-------------------------- Security autoconfiguration information
    ------------------------------

Authentication and authorization are enabled.
TLS for the transport and HTTP layers is enabled and configured.

The generated password for the elastic built-in superuser is :
    estaeslaclavegenerada

If this node should join an existing cluster, you can reconfigure this with
'/usr/share/elasticsearch/bin/elasticsearch-reconfigure-node --enrollment-token
    <token-here>'
after creating an enrollment token on your existing cluster.

You can complete the following actions at any time:

Reset the password of the elastic built-in superuser with
'/usr/share/elasticsearch/bin/elasticsearch-reset-password -u elastic'.

Generate an enrollment token for Kibana instances with
 '/usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s
    kibana'.

Generate an enrollment token for Elasticsearch nodes with
'/usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s node'.
```

```
--------------------------------------------------------------------------------
    -----------------
```

Hacemos unos cambios en el fichero de configuración:

`/etc/elasticsearch/elasticsearch.yml`

```
    network.bind_host: ["127.0.0.1", "192.168.1.11"]
    discovery.seed_hosts: []
    discovery.type: single-node
```

Recargamos los valores de los *daemon*, habilitamos el servicio *elasticsearch* para que arranque con el inicio del sistema, y arrancamos dicho servicio:

```
root@tfm:~# systemctl daemon-reload

root@tfm:~# systemctl enable elasticsearch.service

root@tfm:~# systemctl start elasticsearch.service
```

Comprobamos que el servicio esté correctamente arrancado y hacemos una prueba de su funcionamiento:

```
root@tfm:~# systemctl status elasticsearch
 elasticsearch.service - Elasticsearch
     Loaded: loaded (/lib/systemd/system/elasticsearch.service; enabled;
    vendor>
     Active: active (running) since Mon 2022-04-02 20:41:34 CEST; 23s ago
       Docs: https://www.elastic.co
   Main PID: 1606 (java)
      Tasks: 67 (limit: 9442)
     Memory: 4.3G
        CPU: 1min 7.645s
     CGroup: /system.slice/elasticsearch.service
             1606 /usr/share/elasticsearch/jdk/bin/java -Xshare:auto -Des.net>
             1878 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x>

apr 02 20:41:05 tfm systemd[1]: Starting Elasticsearch...
apr 02 20:41:34 tfm systemd[1]: Started Elasticsearch.
```

```
root@tfm:~# curl --cacert /etc/elasticsearch/certs/http_ca.crt -u elastic
    https://localhost:9200
Enter host password for user 'elastic':
{
  "name" : "tfm",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "IfCk9tJMSsuIHMhbTp20Jw",
  "version" : {
    "number" : "8.1.3",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "39afaa3c0fe7db4869a161985e240bd7182d7a07",
    "build_date" : "2022-04-19T08:13:25.444693396Z",
    "build_snapshot" : false,
    "lucene_version" : "9.0.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Deberemos modificar la variable del sistema `vm.max_map_count` para darle más memoria virtual:

```
root@tfm:~# sysctl -w vm.max_map_count=262144
```

3.4.2. Kibana

Como ya hemos importado las keys y hemos metido en el repositorio `elastic.co`, lanzamos directamente la instalación de Kibana con «`apt-get install kibana`»:

```
root@tfm:~# apt-get install kibana
...
Desempaquetando kibana (8.1.3) ...
Configurando kibana (8.1.3) ...
```

Configuramos Kibana, cuyo fichero de configuración se encuentra en

/etc/kibana/kibana.yml

```
server.host: "192.168.1.10"
server.name: "tfm"
elasticsearch.hosts: ["https://192.168.1.10:9200"]
server.publicBaseUrl: "https://192.168.1.10:5601"
elasticsearch.serviceAccountToken: "<token>""
```

Y lo arrancamos y habilitamos para su arranque en el inicio

```
root@tfm:~# /bin/systemctl daemon-reload

root@tfm:~# /bin/systemctl enable kibana.service

root@tfm:~# systemctl start kibana.service

root@tfm:~# systemctl is-enabled kibana
enabled

root@tfm:~# systemctl status kibana
 kibana.service - Kibana
     Loaded: loaded (/etc/systemd/system/kibana.service; enabled; vendor
    preset>
     Active: active (running) since Mon 2022-04-02 22:40:56 CEST; 1s ago
       Docs: https://www.elastic.co
   Main PID: 2931 (node)
      Tasks: 7 (limit: 9442)
     Memory: 75.9M
        CPU: 1.987s
     CGroup: /system.slice/kibana.service
             2931 /usr/share/kibana/bin/../node/bin/node /usr/share/kibana/bi>

apr 02 22:40:56 tfm systemd[1]: Started Kibana.
```

Comprobamos que esté ok con

```
root@tfm:~# journalctl -u kibana.service
```

Y abrimos en un navegador para comprobar su funcionamiento

http://192.168.1.10:5601

3.4.3. Filebeat

Al igual que hemos hecho con los paquetes anteriores, lanzamos directamente la instalación de Filebeats con «`apt-get install filebeat`»:

```
root@tfm:~# apt-get install filebeat
...
Desempaquetando filebeat (8.1.3) ...
Configurando filebeat (8.1.3) ...
```

Configuramos *Filebeat*:

`/etc/filebeat/filebeat.yml`

```
setup.kibana:
  host: "192.168.1.10:5601"


output.elasticsearch:
  hosts: ["192.168.1.10:9200"]

  protocol: "https"

  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  #username: "elastic"        (descomentamos y rellenamos)
  #password: "changeme"        (descomentamos y rellenamos)
```

Activamos módulo de Suricata:

```
root@tfm:~# filebeat modules enable suricata
Enabled suricata
```

Editamos la configuración del módulo de Suricata:

`/etc/filebeat/modules.d/suricata.yml`

```
- module: suricata
  # All logs
  eve:
    enabled: true

    # Set custom paths for the log files. If left empty,
    # Filebeat will choose the paths depending on your OS.
    var.paths: ["/var/log/suricata/eve.json"]
```

Ahora configuramos Filebeat con el comando

```
filebeat setup -e
```

Nos encontramos que nos da un error de certificado porque no reconoce la autoridad del firmante:

```
root@tfm:~# filebeat setup -e
...
{"log.level":"error","@timestamp":"2022-04-11T23:03:53.682+0200","log.logger":"
    esclientleg","log.origin":{"file.name":"transport/logging.go","file.line":3
    7},"message":"Error dialing x509: certificate signed by unknown
    authority","service.name":"filebeat","network":"tcp","address":"localhost:9
    200","ecs.version":"1.6.0"}
{"log.level":"error","@timestamp":"2022-04-11T23:03:53.682+0200","log.logger":"
    esclientleg","log.origin":{"file.name":"eslegclient/connection.go","file.li
    ne":231},"message":"error connecting to Elasticsearch at
    https://localhost:9200: Get \"https://localhost:9200\": x509: certificate
    signed by unknown
    authority","service.name":"filebeat","ecs.version":"1.6.0"}
{"log.level":"error","@timestamp":"2022-04-11T23:03:53.682+0200","log.origin":{
    "file.name":"instance/beat.go","file.line":1022},"message":"Exiting:
    couldn't connect to any of the configured Elasticsearch hosts. Errors:
    [error connecting to Elasticsearch at https://localhost:9200: Get
    \"https://localhost:9200\": x509: certificate signed by unknown
    authority]","service.name":"filebeat","ecs.version":"1.6.0"}
Exiting: couldn't connect to any of the configured Elasticsearch hosts. Errors:
    [error connecting to Elasticsearch at https://localhost:9200: Get
    "https://localhost:9200": x509: certificate signed by unknown authority]
```

Para solucionarlo, obtenemos la huella del CA (Certificate Authority) para ponerla como autoridad de confianza:

```
root@tfm:~# openssl x509 -fingerprint -sha256 -in
    /etc/elasticsearch/certs/http_ca.crt
SHA256 Fingerprint=FE:DC:BA:98:76:54:32:10:01:23:45:67:89:AB:CD:EF:FE:DC:BA:98:
    76:54:32:10:01:23:45:67:89:AB:CD:EF
-----BEGIN CERTIFICATE-----
<<Eliminado por seguridad>>
-----END CERTIFICATE-----
```

Tomamos el *fingerprint* y sin los dos puntos entre cada par de dígitos hexadecimales, los copiamos en la configuración de Filebeat:

`/etc/filebeat/filebeat.yml`

```
output.elasticsearch:

ssl:
    enabled: true
    ca_trusted_fingerprint:
        "FEDCBA9876543210012345678 9ABCDEFFEDCBA9876543210012345678 9ABCDEF"
```

Volvemos a lanzar el *setup* de Filebeats y comprobamos que ya no nos da error:

```
root@tfm:~# filebeat setup -e
...

Index setup finished.
Loading dashboards (Kibana must be running and reachable)
...
Loaded dashboards
...
{"log.level":"info","@timestamp":"2022-04-15T16:47:01.553+0200","log.origin":{"
    file.name":"fileset/modules.go","file.line":103},"message":"Enabled
    modules/filesets: suricata
    (eve)","service.name":"filebeat","ecs.version":"1.6.0"}
...
Loaded Ingest pipelines
```

Habilitamos y comprobamos el servicio:

```
root@tfm:~# systemctl daemon-reload

root@tfm:~# systemctl enable filebeat.service

root@tfm:~# systemctl is-enabled filebeat.service
enabled

root@tfm:~# systemctl status filebeat.service
 filebeat.service - Filebeat sends log files to Logstash or directly to Elasticsearch.
     Loaded: loaded (/lib/systemd/system/filebeat.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2022-04-15 16:48:07 CEST; 9min ago
       Docs: https://www.elastic.co/beats/filebeat
   Main PID: 30101 (filebeat)
      Tasks: 8 (limit: 9442)
     Memory: 108.4M
        CPU: 14.518s
     CGroup: /system.slice/filebeat.service
             30101 /usr/share/filebeat/bin/filebeat --environment systemd -c /etc/filebeat/filebe

apr 15 16:52:40 tfm filebeat[30101]: {"log.level":"info","@timestamp":"2022-04->
apr 15 16:53:10 tfm filebeat[30101]: {"log.level":"info","@timestamp":"2022-04->
apr 15 16:53:40 tfm filebeat[30101]: {"log.level":"info","@timestamp":"2022-04->
apr 15 16:54:10 tfm filebeat[30101]: {"log.level":"info","@timestamp":"2022-04-
apr 15 16:54:40 tfm filebeat[30101]: {"log.level":"info","@timestamp":"2022-04->
apr 15 16:55:10 tfm filebeat[30101]: {"log.level":"info","@timestamp":"2022-04->
apr 15 16:55:40 tfm filebeat[30101]: {"log.level":"info","@timestamp":"2022-04->
apr 15 16:56:10 tfm filebeat[30101]: {"log.level":"info","@timestamp":"2022-04->
apr 15 16:56:40 tfm filebeat[30101]: {"log.level":"info","@timestamp":"2022-04->
apr 15 16:57:10 tfm filebeat[30101]: {"log.level":"info","@timestamp":"2022-04->
lines 1-21/21 (END)
```

3.4.4. Búsqueda de Dashboards en Kibana

Entramos con un navegador en la web de nuestra instalación de Kibana (`http://192.168.1.10:5601`) con el usuario «`elastic`» y su clave.

Dentro del entorno de Kibana, vamos a cargar los *dashboard* de Suricata. Para ello, en el cuadro de búsqueda ponemos

```
type:dashboard suricata
```

y vemos los dos dashboards que Kibana tiene para Suricata.

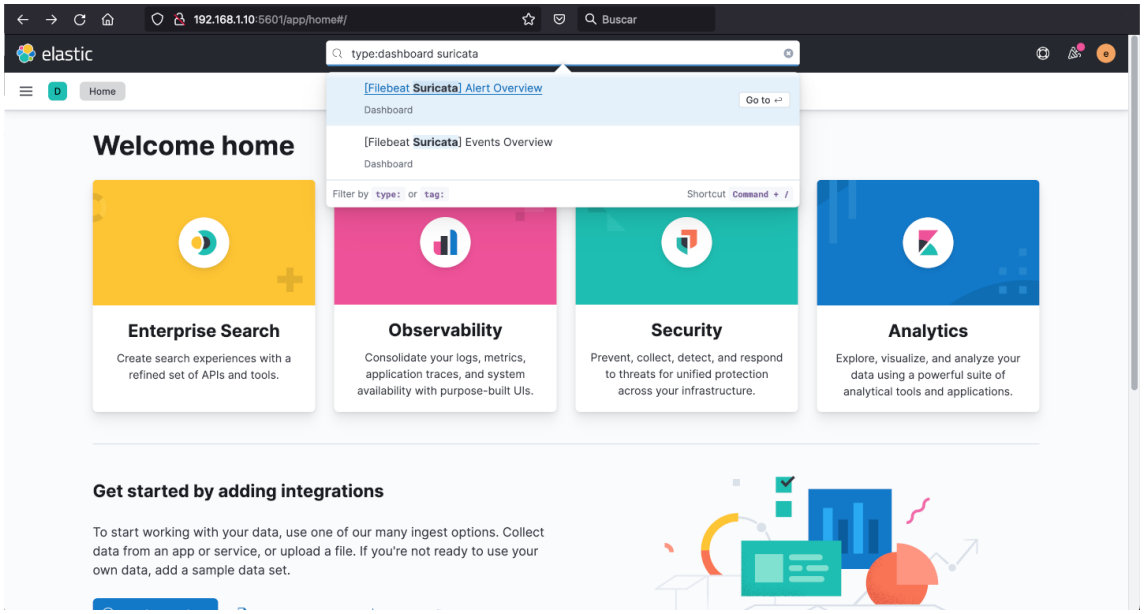

Figura 3.1: Kibana, búsqueda de dashboards de Suricata

3.5. Configuración del router MikroTik

El router MikroTik RB2011UiAS-RM dispone de 5 puertos Gigabit Ethernet, 5 puertos Fast Ethernet y un zócalo SFP.

Figura 3.2: Router MikroTik RB2011UiAS-RM

El router que se va a utilizar ya está configurado, securizado y en uso, y como

la configuración completa de un router MikroTik excede del alcance de este TFM, vamos a obviarlo y darlo por realizado como prerequisito.

En la instalación utilizada de referencia, los puertos utilizados (de interés para este proyecto) son los siguientes:

- Eth1 - enlace WAN
- Eth2 - enlace switch red LAN
- Eth4 - puerto Eth1 del servidor
- Eth5 - puerto Eth2 del servidor

Los router MikroTik permiten varias formas de conectarse a él para su configuración:

- Por línea de comandos vía Telnet, SSH o cable serie
- Por interfaz gráfico via Web
- Usando el programa de configuración *WinBox*[2] (aplicación Windows compatible con Wine[3])

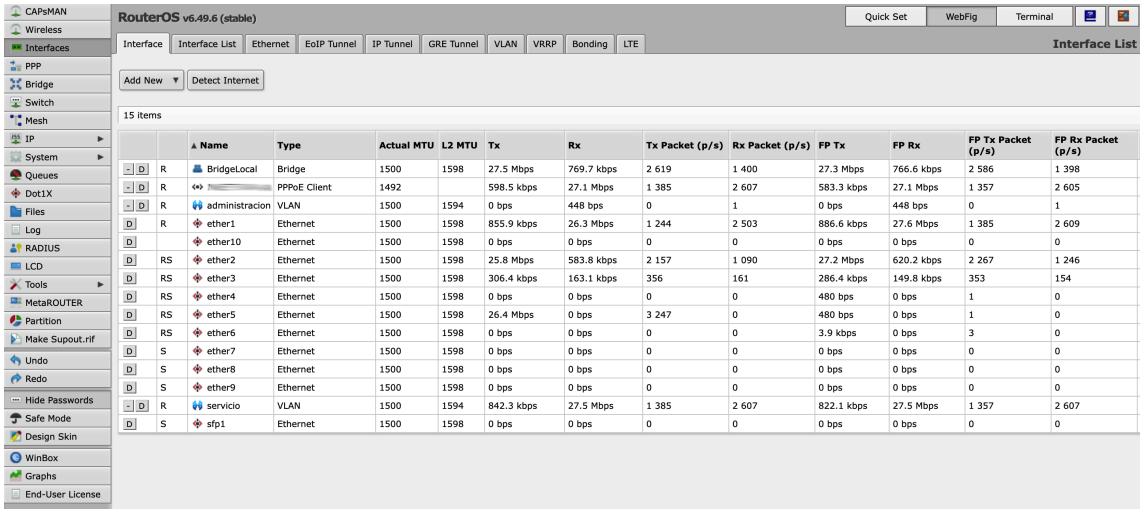

Figura 3.3: Interfaz Web Mikrotik

[2]https://mikrotik.com/download

[3]Wine es una capa de compatibilidad para ejecutar aplicaciones DOS y Windows en sistemas operativos basados en Unix

Aunque se pueda configurar el router en modo gráfico, vamos a configurarlo por consola para una mejor descripción de lo que se realiza.

Vamos a configurar el router MikroTik en modo *port mirroring*. Vamos a duplicar el tráfico del puerto `Eth2` al puerto `Eth5`. Para ello introducimos en la consola de comandos:

```
/interface ethernet switch
set switch1 mirror-source=ether2 mirror-target=ether5
```

3.6. Dashboards de Suricata en Kibana

Tomando de partida los dos *dashboards* que tiene Kibana de visualización de datos de Suricata (figura 3.1, se ha realizado unas modificaciones para mostrar en primera pantalla los datos que nos interesan.

En el panel de eventos, podemos observar el tráfico detectado, los tipos de eventos, protocolos de transporte, protocolos de red y los paises de origen y destino de los paquetes:

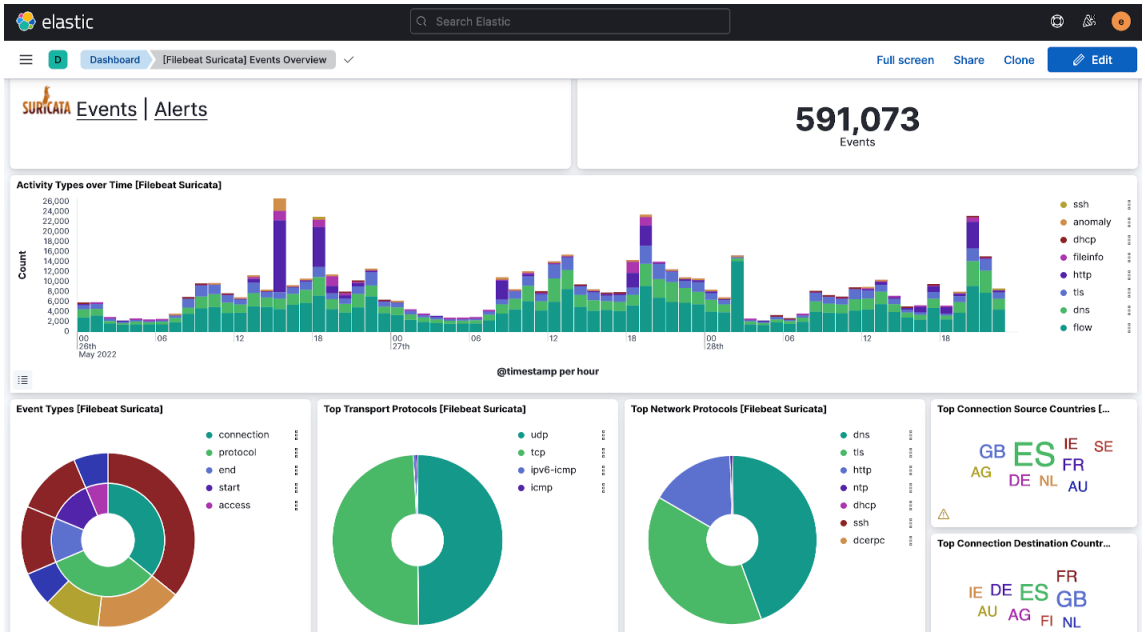

Figura 3.4: Kibana - Eventos de Suricata

En el panel de alertas, vemos las principales alertas con su categoría, y la distribución en mapa del origen y destino de dichas alertas:

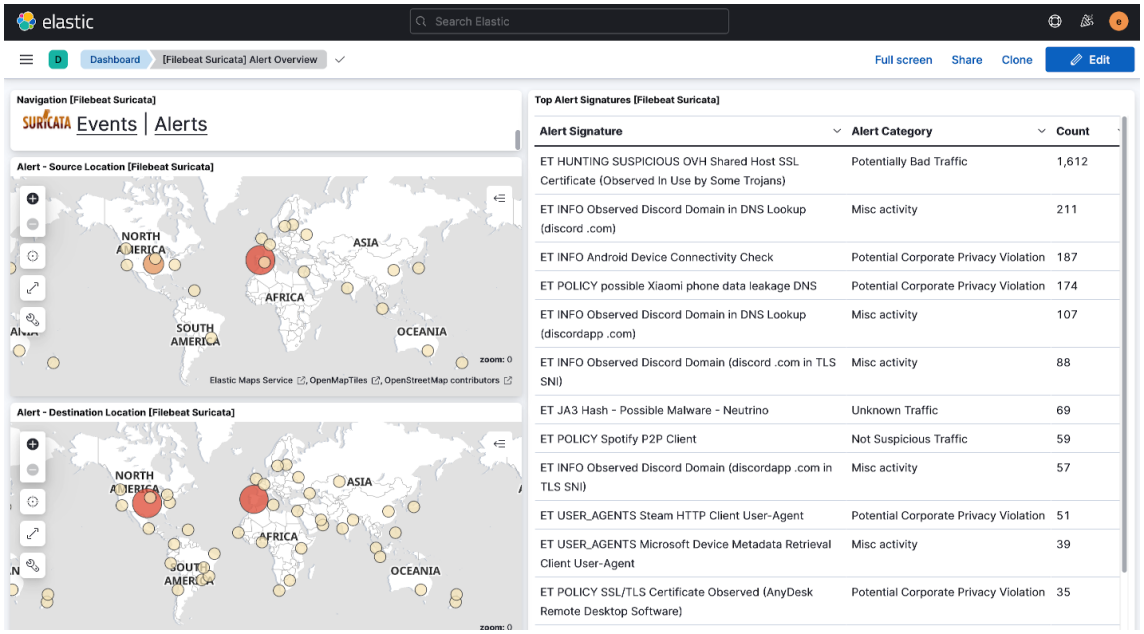

Figura 3.5: Kibana - Alertas de Suricata

3.7. Configuración alternativa para router en modo *packet sniffer*

Como ya indicamos, además del modo *port mirroring*, podemos configurar el router en modo *packet sniffer*, lo que conlleva también hacer una conversión del protocolo TZSP a tráfico de red en una interfaz virtual.

3.7.1. Conversor de protocolo TZSP a tráfico de red en interfaz *dummie*

Para realizar este conversor, vamos a necesitar 2 herramientas:

- tzsp2pcap[4]
- tcpreplay[5]

Empecemos instalando *tzsp2pcap*:

```
root@tfm:~# wget -c
    https://github.com/thefloweringash/tzsp2pcap/archive/refs/heads/master.zip
    -O tzsp2pcap.zip

root@tfm:~# unzip tzsp2pcap.zip
Archive:  tzsp2pcap.zip
4fb8147c15ad5fcf887a8e880d40596c955fbead
   creating: tzsp2pcap-master/
  inflating: tzsp2pcap-master/.travis.yml
  inflating: tzsp2pcap-master/COPYING
  inflating: tzsp2pcap-master/Makefile
  inflating: tzsp2pcap-master/README.org
  inflating: tzsp2pcap-master/default.nix
   creating: tzsp2pcap-master/nix/
  inflating: tzsp2pcap-master/nix/package.nix
  inflating: tzsp2pcap-master/tzsp2pcap.c

root@tfm:~# cd tzsp2pcap-master/
```

[4]https://github.com/thefloweringash/tzsp2pcap
[5]https://github.com/appneta/tcpreplay

```
root@tfm:~/tzsp2pcap-master# make
cc -o tzsp2pcap -std=c99 -D_DEFAULT_SOURCE -Wall -Wextra -pedantic -O2 -g
    tzsp2pcap.c -lpcap

root@tfm:~/tzsp2pcap-master# cp tzsp2pcap /usr/local/bin
```

Ahora hagamos lo respectivo con *tcpreplay*:

```
root@tfm:~# apt-get install build-essential libcap-dev

root@tfm:~# wget -c https://github.com/appneta/tcpreplay/releases/download/v4.4
    .1/tcpreplay-4.4.1.tar.xz

root@tfm:~# tar -xvf tcpreplay-4.4.1.tar.xz

root@tfm:~# cd tcpreplay-4.4.1/

root@tfm:~# ./configure

root@tfm:~# make

root@tfm:~# make install
```

Creamos una interface de red dummy para reconstruir el tráfico TZSP como `tzsp0` e IP `192.168.1.12`:

```
root@tfm:~# nano /etc/systemd/network/tzsp.netdev
[NetDev]
Name=tzsp0
Kind=dummy


root@tfm:~# nano /etc/systemd/network/tzsp.network
[Match]
Name=tzsp*

[Link]
MTUBytes=2000

[Network]
```

```
Address=192.168.1.12/24
DHCP=no
```

Creamos un servicio que combine `tcpreplay` y `tzsp2pcap`:

```
root@tfm:~# nano /etc/systemd/system/TZSPreplay@tzsp0.service
[Unit]
Description=TZSP Replay on dev %i
After=network.target network-online.target
Requires=network-online.target

[Service]
Type=simple
ExecStart=/bin/sh -c "/usr/bin/tzsp2pcap -f | /usr/bin/tcpreplay-edit
    --topspeed --mtu=$(cat /sys/class/net/%I/mtu) --mtu-trunc -i %I -"
Restart=always
RestartSec=3
ProtectSystem=full
ProtectHome=true

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

Reiniciamos el servicio de red para que cree la nueva interfaz de red, comprobamos las interfaces de red que hay disponibles, habilitamos y arrancamos el servicio y comprobamos que esté activo:

```
root@tfm:~# systemctl restart systemd-networkd.service

root@tfm:~# networkctl list
IDX LINK  TYPE     OPERATIONAL SETUP
  1 lo    loopback carrier     unmanaged
  2 eno1  ether    routable    unmanaged
  3 eno2  ether    routable    unmanaged
  4 tzsp0 ether    routable    configured

4 links listed.

root@tfm:~# systemctl enable --now TZSPreplay@tzsp0.service
Created symlink
    /etc/systemd/system/multi-user.target.wants/TZSPreplay@tzsp0.service →
    /etc/systemd/system/TZSPreplay@tzsp0.service.
```

```
root@tfm:~# systemctl status TZSPreplay@tzsp0.service
 TZSPreplay@tzsp0.service - TZSP Replay on dev tzsp0
     Loaded: loaded (/etc/systemd/system/TZSPreplay@tzsp0.service; enabled;
    ven>
     Active: activating (auto-restart) (Result: exit-code) since Mon
    2022-04-23>
    Process: 107223 ExecStart=/bin/sh -c /usr/bin/tzsp2pcap -f |
    /usr/bin/tcpre>
   Main PID: 107223 (code=exited, status=127)
        CPU: 10ms
```

3.7.2. Ajustes de Suricata

```
root@tfm:~# nano /etc/suricata/suricata.yaml

# Linux high speed capture support
af-packet:
  - interface: tzsp0

root@tfm:~# systemctl restart suricata

root@tfm:~# systemctl status suricata
```

3.7.3. Configuración del MikroTik para modo *packet sniffer*

Vamos a configurar el router MikroTik en modo *packet sniffer*. Primero, paramos el modo *port mirroring* y después habilitamos el modo *packet sniffer*, enviando los paquetes como streaming TZSP a la IP 192.168.1.11:

```
/interface ethernet switch set switch1 mirror-source=none mirror-target=none

/tool sniffer
set streaming-enabled=yes streaming-server=192.168.1.11 filter-stream=yes
start
```

Capítulo 4

Conclusiones

4.1. Líneas de trabajo futuro

Entre las principales mejoras y ampliaciones que se podrían desarrollar en un futurible del proyecto desarrollado se encuentran:

- **Securización del proyecto.** En el proyecto desarrollado, por centrar el foco en el desarrollo del mismo, se han dejado en un segundo término la securización de diversos componentes. Como trabajo futuro se cifrarían, por un lado el acceso al dashboard de Kibana, y por otro la interconexión entre los distintos módulos de *Elastic stack*.
- **Módulo de alertas.** Aunque en un principio se tenía previsto realizar un módulo para la comunicación de alertas, por problemas en la ejecución del TFM, hubo que dejarlo aparcado. La opción que se comenzó a evaluar es *Kibana Alerting*.
- **Creación de más *dashboards* personalizados.** Aunque hemos adaptado los dashboards que Kibana tiene para la visualización de los datos de Suricata, la creación de nuevos dashboards personalizados nos daría la posibilidad de ajustar toda la información que consideremos necesaria.
- **Instalación del IDS en zonas LAN y WAN.** Como ya se indicó en 2.1, una mejora de este proyecto sería instalar el IDS en ambas zonas.
- **Conversión del proyecto en un IPS.** Considerando las funcionalidades de Suricata, y que los routers MikroTik disponen de un API, la actuación sobre

las amenazas a nivel del router sería una gran ampliación a este TFM.

4.2. Reflexión sobre el TFM

Mis circunstancias personales han hecho que me haya sido complicado llevar el TFM adelante: todos los riesgos personales descritos en la sección 1.8 «Análisis de riesgos» no son fruto de la inventiva de este alumno, sino fiel reflejo de la realidad acaecida.

Pero a pesar de todo, este TFM ha sido muy gratificante, pues la idea original de buscar una solución funcional y asequible que pudiera orientarse a autónomos, microPYMES y PYMES se ha resuelto de forma satisfactoria: con las pruebas realizadas se puede observar un amplio rango de alertas ante posibles intrusiones, con unos requisitos hardware que bien pudieran estar en la infraestructura de las empresas objetivo.

A nivel personal, el retomar LaTeXdespués de más de 25 años (desde el Proyecto Fin de Carrera allá por los 90) ha sido todo un rejuvenecimiento.

Además, me ha servido para refrescar y redescubrir este lenguaje que estaba deseando acometer, pues tenía en mente utilizarlo a nivel profesional combinándolo con una Base de Datos para el maquetado automático de Informes Periciales y me va a servir de revulsivo para acometerlo en breve.

Apéndice A

Soluciones evaluadas

SonicWall	https://www.sonicwall.com
Cisco	https://www.cisco.com
FortiNet	https://www.fortinet.com
Sophos	https://www.sophos.com
WatchGuard	https://www.watchguard.com
BluVector	https://www.bluvector.io

Tabla A.1: Principales fabricantes de sistemas UTM comerciales

Endian	https://www.endian.com/community/
OPNSense	https://opnsense.org
pfSense	https://www.pfsense.org
Security Onion	https://securityonionsolutions.com/

Tabla A.2: Principales sistemas UTM Open Source

McAfee Network Security Platform	https://www.mcafee.com/enterprise/en-us/products/network-security-platform.html
Hillstone S-Series Network Intrusion Prevention System	https://www.hillstonenet.com/products/network-intrusion-prevention-system-s-series/
Cisco Secure IPS	https://www.cisco.com/c/en/us/products/security/ngips/index.html
Trend Micro TippingPoint Threat Protection System	https://www.trendmicro.com/en_us/business/products/network/intrusion-prevention.html
Zscaler Cloud IPS	https://www.zscaler.com/products/cloud-ips

Tabla A.3: Principales Sistemas IDS/IPS comerciales

HIDS	OSSEC	https://www.ossec.net
	Samhain	https://la-samhna.de/samhain/index.html
	Wazuh	https://wazuh.com
NIDS	Snort	https://www.snort.org
	Suricata	https://suricata-ids.org
	Zeek	https://zeek.org
	Kismet	https://www.kismetwireless.net

Tabla A.4: Principales Sistemas IDS Open Source

Splunk	https://www.splunk.com
Chronicle	https://chronicle.security
MetricFire	https://www.metricfire.com
InsightIDR	https://www.rapid7.com/products/insightidr
Sumo Logic	https://www.sumologic.com/solutions/security-intelligence
SolarWinds Security Event Manager	https://www.solarwinds.com/security-event-manager
ManageEngine EventLog Analyzer	https://www.manageengine.com/products/eventlog
LogRhythm NextGen SIEM Platform	https://logrhythm.com/products/nextgen-siem-platform
IBM QRadar SIEM	https://www.ibm.com/products/qradar-siem
ArcSight Enterprise Security Manager	https://www.microfocus.com/es-es/products/siem-security-information-event-management/overview

Tabla A.5: Herramientas de análisis y presentación de datos comerciales

Prometheus	https://prometheus.io
Graphite	https://graphiteapp.org
Elastic Stack	https://www.elastic.co/es/elastic-stack
Grafana	https://grafana.com/grafana
AlienVault OSSIM	https://cybersecurity.att.com/products/ossim
Apache Metron	https://metron.apache.org
SIEMonster	https://siemonster.com/community-edition
Prelude	https://www.prelude-siem.org/
Sagan	https://quadrantsec.com/sagan_log_analysis_engine
ACARM-ng	http://www.acarm.wcss.wroc.pl

Tabla A.6: Herramientas de análisis y presentación de datos Open Source

Glosario

BYOD	Tráete tu propio dispositivo (del inglés *Bring Your Own Device*)
HIDS	Sistema de detección de intrusiones de host (del inglés *Host Intrusion Detection System*)
IDS	Sistema de detección de intrusiones (del inglés *Intrusion Detection System*)
IoT	Internet de las cosas (del inglés *Internet of Things*)
IPS	Sistema de prevención de intrusiones (del inglés *Intrusion Prevention System*)
LOPDGDD	Ley Orgánica de Protección de Datos personales y Garantía de los Derechos Digitales
NIDS	Sistema de prevención de intrusiones de red (del inglés *Network Intrusion Detection System*)
PGP	Sistema de cifrado de datos (del inglés *Pretty Good Privacy*)
RGPD	Reglamento General de Protección de Datos
SFP	Forma abreviada de Transceptor SPF (del inglés *Small Form-factor Pluggable transceiver*)
SSH	Shell seguro (del inglés *Secure SHell*)
UDP	Protocolo de datagramas de usuario (del inglés *User Datagram Protocol*)
TFM	Trabajo Fin de Máster

UOC Universitat Oberta de Catalunya

UTM Gestión Unificada de Amenazas (del inglés *Unified Threat Management*)

VPN Red Privada Virtual (del inglés *Virtual Private Network*)

TZSP Protocolo de encapsulación (del inglés *TaZmen Sniffer Protocol*)

Bibliografía

[1] Roser Beneito Montagut. *Presentación de documentos y elaboración de presentaciones*. UOC. http://materials.cv.uoc.edu/cdocent/NR64BN4TZTOGMDE0V2D7.pdf.

[2] Biblioteca de la UOC. Monográfico de la biblioteca sobre citación bibliográfica. http://biblioteca.uoc.edu/es/recursos/citacion-bibliografica.

[3] Eduardo Casilari Pérez, José Antonio Cortés Arrabal, and Luís Molina Tanco. *Breves notas de estilo para la redacción de Proyectos Fin de Carrera y Trabajos Fin de Grado*. ETSIT, Universidad de Málaga, 2014. https://www.uma.es/media/tinyimages/file/Manual_de_estilo_ETSIT_TFG_TFM_PFC.pdf.

[4] Iria da Cunha. *El trabajo de fin de grado y de máster: Redacción, defensa y publicación*. Editorial UOC, 1ª edition, Mayo 2016. ISBN: 978-84-9064-391-4.

[5] Yuri Diogenes and Erdal Ozkaya. *Cybersecurity - Attack and Defense Strategies*. Packt, Enero 2018. ISBN: 978-1788475297.

[6] Ibrahim Ghafir, Vaclav Prenosil, Jakub Svoboda, and Mohammad Hammoudeh. A survey on network security monitoring systems. Agosto 2016. https://www.researchgate.net/profile/Vaclav-Prenosil/publication/309229246_A_Survey_on_Network_Security_Monitoring_Systems/links/58a5ad4b4585150402d2918d/A-Survey-on-Network-Security-Monitoring-Systems.pdf.

[7] INCIBE-CERT. Diseño y Configuración de IPS, IDS y SIEM en Sistemas de Control Industrial. https://www.incibe-cert.es/sites/default/files/c

ontenidos/guias/doc/certsi_diseno_configuracion_ips_ids_siem_en_sci.pdf, Noviembre 2017.

[8] Daniel Medina. An IDS Using NetFlow Data. 2002. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.2495&rep=rep1&type=pdf.

[9] Tobias Oetiker and Contributors. La introducción no-tan-corta a LaTeX 2_ε. Agosto 2014. http://mirrors.ctan.org/info/lshort/spanish/lshort-a4.pdf.

[10] Txus Pampliega Carcedo. Implantación y despliegue SIEM (Security Information and Event Management) & SOC (Security Operation Center) Deployment. https://www.linkedin.com/pulse/implantación-y-despliegue-siem-security-information-txus, Octubre 2018.

[11] Alberto Paro. *Elasticsearch 7.0 Cookbook*. Packt, 4ª edition, Abril 2019. ISBN: 978-1789956504.

[12] Oren Patashnik. BibTeXing. documentation for general BibTeX users. Febrero 1988. http://mirrors.ctan.org/biblio/bibtex/contrib/doc/btxdoc.pdf.

[13] Geoffrey M. Poore. The minted package: Highlighted source code in . Diciembre 2021. http://mirrors.ctan.org/macros/latex/contrib/minted/minted.pdf.

[14] David Santo Orcero. *La Biblia de LaTeX 2_ε*. Ed. Coronado, 2ª edition, Septiembre 2019. ISBN: 978-1795409711.

[15] Pranav Shukla and Sharath Kumar. *Learning Elastic Stack 7.0*. Packt, 2ª edition, Mayo 2019. ISBN: 978-1789954395.

[16] Nita Sáenz Higueras and Rut Vidal Oltra. *Redacción de textos científico-técnicos*. UOC. http://materials.cv.uoc.edu/cdocent/_D_CBBU62JZTHQ9CQQGJ.pdf.

[17] Peter Wilson and Herries Press. The appendix package. Febrero 2020. http://mirrors.ctan.org/macros/latex/contrib/appendix/appendix.pdf.

Webs consultadas

https://es.overleaf.com/

https://www.fundeu.es/recomendacion/comillas-uso-de-este-signo-ortografico/

https://help.mikrotik.com/docs/

https://help.mikrotik.com/docs/display/ROS/Switch+Chip+Features

https://help.mikrotik.com/docs/display/ROS/Packet+Sniffer

https://help.mikrotik.com/docs/display/ROS/Mangle

https://help.mikrotik.com/docs/display/ROS/Traffic+flow

https://en.wikipedia.org/wiki/TZSP

https://etutorials.org/Networking/network+management/Part+II+Implementations+on+the+Cisco+Devices/Chapter+7.+NetFlow/Fundamentals+of+NetFlow/

https://bl\T1\ogg.no/2020/11/traffic-capturing-and-streaming-with-mikrotik-revisited/

https://backports.debian.org/Instructions/

https://packages.debian.org/bullseye-backports/net/

https://bricata.com/blog/snort-suricata-bro-ids/

https://cybersecurity.att.com/blogs/security-essentials/open-source-intrusion-detection-tools-a-quick-overview

https://www.comparitech.com/net-admin/open-source-siem-tools

https://www.dnsstuff.com/free-siem-tools

https://dzone.com/articles/6-open-source-siem-tools-logzio

https://zeek.org/

https://docs.zeek.org/en/master/about.html

https://www.informit.com/articles/article.aspx?p=21778&seqNum=9

https://suricata.readthedocs.io/en/latest/quickstart.html

https://blog.elhacker.net/2021/03/suricata-ids-ips-instalacion-configuracion-reglas-.html

https://www.digitalocean.com/community/tutorials/how-to-install-suricata-on-debian-11

https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html

https://www.elastic.co/guide/en/kibana/current/deb.html

https://www.elastic.co/es/what-is/kibana-alerting

https://www.digitalocean.com/community/tutorials/how-to-build-a-siem-with-suricata-and-elastic-stack-on-debian-11

https://robert.penz.name/849/howto-setup-a-mikrotik-routeros-with-suricata-as-ids/

https://github.com/zzbe/mikrocata

https://github.com/thefloweringash/tzsp2pcap

https://github.com/appneta/tcpreplay