

Despliegue de Zeek IDS y su posterior explotación para el análisis de actividades sospechosas en la red

Estudiante: Juan Francisco Rodríguez Vílchez

Plan de Estudios: Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC).

Área del trabajo final: Análisis de datos

Director TFM: Borja Guaita Pérez

Profesor/a responsable de la asignatura: Víctor García Font

Fecha Entrega: 2 de junio de 2020



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-sa/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	Despliegue Zeek IDS y su posterior explotación para el análisis de actividades sospechosas en la red
Nombre del autor:	<i>Juan Francisco Rodríguez Vílchez</i>
Nombre del consultor/a:	<i>Borja Guaita Pérez</i>
Nombre del PRA:	<i>Víctor García Font</i>
Fecha de entrega (mm/aaaa):	Junio de 2020
Titulación:	Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC).
Área del Trabajo Final:	<i>Análisis de datos</i>
Idioma del trabajo:	<i>Castellano</i>
Palabras clave	<i>IDS, SIEM</i>
Resumen del Trabajo (máximo 250 palabras):	
<p>Actualmente, la mayoría de las corporaciones ofrecen servicios que dependen de sitios web y aplicaciones basadas en internet. Esto provoca que los riesgos para la privacidad y seguridad a los que están expuestos aumenten considerablemente.</p> <p>Los ciberataques han aumentado estando al alcance de personas con conocimientos escasos. Lo que anteriormente se hacía por diversión, reconocimiento o ganas de aprender, ahora se ha convertido en uno de los negocios que mueve más cantidad de dinero.</p> <p>Para solucionar estos riesgos, las empresas utilizan sistemas de protección que generan un volumen de datos enorme, siendo muy difícil de analizar rápidamente y filtrado en alertas procesables.</p> <p>La finalidad de este proyecto reside en la implementación de un sistema, compuesto por un IDS integrado en un SIEM, que proporciona un análisis exhaustivo de la red de</p>	

la organización, vigilando y monitorizando el tráfico y detectando posibles eventos maliciosos.

Mediante este sistema, el equipo de seguridad contará con un método efectivo para automatizar procesos y centralizar la gestión de la seguridad. Además, se realizará una estandarización de datos, transformando los datos provenientes de diferentes fuentes a un formato común utilizado para ejecutar el análisis y correlación reduciendo la carga de trabajo del equipo.

El proyecto aportará un grado de seguridad de tipo preventivo ante cualquier actividad sospechosa haciendo que el equipo de seguridad reciba alertas anticipadas y puedan actuar con antelación suficiente para mitigar el problema y minimizar la afectación.

Abstract (in English, 250 words or less):

Currently, most corporations offer services that rely on internet-based websites and applications. This situation causes that the privacy and security risks increase considerably.

Cyber attacks have increased being available to people with limited knowledge. What was previously done for fun, recognition or desire to learn, has become in one of the businesses that move more money.

To solve these risks, companies use protection systems that generate huge volumen of data, being very difficult to analyze it quickly and filter it in actionable alerts.

The project aims to alleviate this situation by implementing a system consisting of an Intrusion Detection System (IDS) integrated in a Security Information and Event Management (SIEM) that provides an exhaustive analysis of the organization's network, monitoring traffic and detecting posible malicious events.

By using this system, security teams will have an effective method to automate processes and centralize security management. In addition, data standardization is performed, transforming the data from different sources into a common format used to run análisis and correlation, reducing the team's workload.

This project will provide a degree of preventive security against any suspicious activity, making the security team receive early alerts and be able to act quickly enough to mitigate the problem and minimize the impact.

Índice

1. Introducción.....	1
1.1. Contexto y justificación del Trabajo	1
1.2. Objetivos del Trabajo.....	2
1.3. Enfoque y método seguido	4
1.4. Planificación del Trabajo	5
1.5. Planificación temporal detallada	6
1.6. Estado del Arte.....	8
1.7. Presupuesto y recursos necesarios	11
2. Fase de Investigación.....	13
2.1. Sistemas de Detección de Intrusiones (IDS).....	13
2.1.1. Introducción.....	13
2.1.2. Tipos de IDS.....	14
2.1.2.1. Actividad Analizada	14
2.1.2.1.1. Sistema de Detección de Intrusiones en el host (HIDS).....	14
2.1.2.1.2. Sistema de Detección de Intrusiones de red (NIDS).....	15
2.1.2.2. Método de detección.....	17
2.1.2.2.1. Basados en firmas.....	17
2.1.2.2.2. Basados en anomalías.....	17
2.1.3. Herramientas IDS más destacadas	18
2.1.3.1. Snort.....	18
2.1.3.2. Suricata.....	19
2.1.3.3. OSSEC.....	20
2.1.4. Herramienta a implementar: Zeek IDS.....	21
2.1.4.1. Arquitectura.....	22
2.2. Sistemas de gestión de información y eventos de seguridad (SIEM).....	23
2.2.1. Introducción.....	23
2.2.2. Funcionamiento.....	25
2.2.3. Herramientas SIEM más destacadas	26
2.2.3.1. SolarWinds Security Event Manager	26
2.2.3.2. Splunk Enterprise Security.....	27

2.2.3.3.	McAfee Enterprise Security Manager	28
2.2.3.4.	Micro Focus ArcSight ESM	29
2.2.4.	Herramienta a implementar: ELK Stack	30
2.2.4.1.	Recolección de datos: Beats	32
2.2.4.2.	Procesamiento de datos: Logstash.....	33
2.2.4.3.	Almacenamiento: Elasticsearch.....	34
2.2.4.4.	Visualización: Kibana.....	35
2.3.	Integración ELK y Zeek: Escenario propuesto.....	37
3.	Fase de Implementación	39
3.1.	Agentes.....	39
3.1.1.	Filebeat.....	40
3.1.2.	MetricBeat.....	41
3.1.3.	AuditBeat.....	42
3.2.	Zeek IDS.....	42
3.2.1.	Configuración.....	43
3.2.2.	Logs.....	44
3.2.2.1.	Cargando la Inteligencia (intel.log).....	47
3.2.3.	Scripting.....	49
3.3.	ELK Stack.....	50
3.3.1.	Recolección de datos.....	50
3.3.2.	Almacenamiento e indexación.....	54
3.3.3.	Exploración y visualización de datos.....	56
3.4.	Escenario de Pruebas	59
3.4.1.	Detección de tráfico malicioso.....	61
3.4.1.1.	Fuerza Bruta.....	61
3.4.1.2.	Framework de Inteligencia.....	62
3.4.1.3.	Escaneo de puertos.....	63
3.4.1.4.	Inyección SQL.....	64
3.4.2.	Exploración y visualización de eventos.....	65
3.4.2.1.	Anomalías detectadas por Zeek IDS.....	65
3.4.2.2.	Datos estadísticos.....	68
3.4.2.3.	Datos de auditoría.....	69
4.	Conclusiones Finales	70
4.1.	Seguimiento de objetivos	71

4.2.	Seguimiento de la planificación establecida.....	73
4.2.1.	Dificultades encontradas.....	74
4.3.	Líneas de trabajo futuro	75
5.	<i>Glosario</i>.....	77
6.	<i>Bibliografía</i>.....	79
7.	<i>Anexos</i>.....	84
7.1.	Instalaciones.....	84
7.1.1.	Agentes.....	84
7.1.2.	Zeek IDS.....	85
7.1.3.	ELK Stack.....	86
7.1.3.1.	Elasticsearch.....	86
7.1.3.2.	Kibana.....	88
7.1.3.3.	Logstash.....	90
7.2.	Ficheros de configuración.....	91
7.3.	Exploración y visualización de eventos adicionales	96
7.3.1.	Datos estadísticos.....	96
7.3.2.	Datos de auditoría	98
7.4.	Ficheros de logs Zeek IDS.....	102

Lista de figuras

<i>Ilustración 1: Empresas relevantes que utilizan Elastic como solución SIEM.</i>	10
<i>Ilustración 2: Tratamiento de logs obtenidos con Zeek IDS por parte de ELK.</i>	10
<i>Ilustración 3: Diagrama que indica el salario según el puesto de trabajo.</i>	12
<i>Ilustración 4: Sistema de detección de intrusiones en el host.</i>	15
<i>Ilustración 5: Sistema de detección de intrusiones en la red.</i>	16
<i>Ilustración 6: Logotipo de Snort.</i>	18
<i>Ilustración 7: Logotipo de Suricata.</i>	19
<i>Ilustración 8: Logotipo de OSSEC.</i>	20
<i>Ilustración 9: Logotipo de Zeek.</i>	21
<i>Ilustración 10: Arquitectura de Zeek IDS.</i>	22
<i>Ilustración 11: Arquitectura típica de un sistema SIEM.</i>	25
<i>Ilustración 12: Interfaz de SolarWinds Security Event Manager.</i>	27
<i>Ilustración 13: Intefaz de Splunk Enterprise Security</i>	28
<i>Ilustración 14: Interfaz de McAfee Enterprise Security Manager.</i>	29
<i>Ilustración 15: Interfaz de Micro Focus ArcSight ESM.</i>	30
<i>Ilustración 16: Arquitectura de ELK Stack.</i>	31
<i>Ilustración 17: Arquitectura de Beats donde se muestran los distintos tipos de recolectores de datos.</i>	32
<i>Ilustración 18: Pipeline de Logstash y envío de datos a Elasticsearch.</i>	34
<i>Ilustración 19: Tipos de nodos en Elasticsearch.</i>	35
<i>Ilustración 20: Dashboard de métricas en Kibana.</i>	36
<i>Ilustración 21: Escenario propuesto a implementar.</i>	37
<i>Ilustración 22: Ejemplo registro intel.log.</i>	45
<i>Ilustración 23: Ejemplo registro notice.log.</i>	46
<i>Ilustración 24: Ejemplo registro conn.log.</i>	46
<i>Ilustración 25: Token CSIRTG generado.</i>	47
<i>Ilustración 26: Arquitectura Logstash.</i>	50
<i>Ilustración 27: Resultado Test conexión Filebeat - Logstash.</i>	54
<i>Ilustración 28: Índices creados en Elasticsearch.</i>	56
<i>Ilustración 29: Creación de patrón de índice en Kibana.</i>	56
<i>Ilustración 30: Selección de marca de tiempo en Kibana.</i>	57
<i>Ilustración 31: Opción "Discover" de Kibana.</i>	57
<i>Ilustración 32: Ejemplo de filtro aplicado en Kibana.</i>	58
<i>Ilustración 33: Diagrama de barras en Kibana.</i>	58

<i>Ilustración 34: Diagrama añadido a un Dashboard en Kibana.</i>	59
<i>Ilustración 35: Entorno para el escenario de pruebas.</i>	60
<i>Ilustración 36: Login aplicación web Django.</i>	65
<i>Ilustración 37: SQLi aplicado sobre aplicación Django.</i>	65
<i>Ilustración 38: Resumen de eventos registrados por Zeek IDS.</i>	66
<i>Ilustración 39: Anomalías registradas en notice.log.</i>	66
<i>Ilustración 40: Anomalías detectadas por el Framework de Inteligencia.</i>	67
<i>Ilustración 41: Ej. split de tablas por cada anomalía detectada (Certificados inválidos).</i>	67
<i>Ilustración 42: Ej. split de tablas por cada anomalía detectada (Escaneo de puertos).</i>	67
<i>Ilustración 43: Dashboard de detección de anomalías en Kibana.</i>	68
<i>Ilustración 44: Uso de CPU y memoria del servidor.</i>	68
<i>Ilustración 45: Dashboard Resumen del servidor web.</i>	69
<i>Ilustración 46: Verificación instalación Elasticsearch.</i>	88
<i>Ilustración 47: Interfaz de Kibana.</i>	89
<i>Ilustración 48: Tráfico entrante y saliente obtenido del servidor web.</i>	96
<i>Ilustración 49: Uso de CPU del servidor web.</i>	96
<i>Ilustración 50: Uso de memoria del servidor web.</i>	97
<i>Ilustración 51: Tráfico de red (paquetes).</i>	97
<i>Ilustración 52: Procesos por memoria más utilizados.</i>	97
<i>Ilustración 53: Procesos por CPU más utilizados.</i>	98
<i>Ilustración 54: Interfaces del servidor web con más tráfico entrante.</i>	98
<i>Ilustración 55: Interfaces del servidor web con más tráfico saliente.</i>	98
<i>Ilustración 56: Cantidad de hosts monitorizados por Auditbeat.</i>	99
<i>Ilustración 57: Información de inicios de sesión en el servidor web.</i>	99
<i>Ilustración 58: Usuarios registrados e interactuando en el servidor web.</i>	100
<i>Ilustración 59: Dashboard que muestra información sobre los procesos del servidor web.</i>	100
<i>Ilustración 60: Dashboard de auditorías de eventos Kernel del servidor web.</i>	101
<i>Ilustración 61: Dashboard de integridad de ficheros en servidor web.</i>	101
<i>Ilustración 62: Campos del registro intel.log.</i>	102
<i>Ilustración 63: Campos del registro notice.log.</i>	102
<i>Ilustración 64: Campos del registro conn.log.</i>	102

1. Introducción

1.1. Contexto y justificación del Trabajo

En la actualidad, la mayoría de las corporaciones ofrecen servicios para crear y consumir negocios que dependen de sitios web y aplicaciones basadas en internet, de forma que los riesgos para la privacidad y seguridad a los que están expuestos aumentan considerablemente.

Los ciberataques ya no se limitan a objetivos de alto nivel, pueden afectar a empresas de cualquier tamaño que dependan de sistemas en red. Lo que anteriormente se hacía por diversión, reconocimiento o ganas de aprender, ahora ha pasado a convertirse en uno de los negocios que mueve más cantidades de dinero. Además, se trata de un negocio al alcance de cualquier persona con las habilidades básicas necesarias para realizar un ciberataque, por ejemplo, en la Dark Web se pueden encontrar “kits de exploits” que con un par de clicks pueden realizar numerosos ataques: DDoS, instalación de malware, phishing, Botnets..., lo cual hace que no se requieran amplios conocimientos técnicos para realizar estos ataques. Por motivos como este, la ciberseguridad se convierte en una prioridad para cualquier empresa presente en internet.

Como se ha comentado anteriormente, la posibilidad de realizar un ciberataque está al alcance de personas con conocimientos escasos, creciendo en complejidad en muy poco tiempo y haciendo que las defensas tradicionales como antivirus, firewall, etc, se queden cortas para prevenir, mitigar y resolver los problemas que estos ataques puedan ocasionar. Si a esto le sumamos la enorme cantidad de datos que se recogen desde distintos nodos de la red, la tarea de detección de amenazas se convierte en un trabajo altamente costoso, tanto en tiempo como en esfuerzo.

Otra preocupación existente es la posibilidad de que, accidentalmente, los empleados configuren erróneamente los ajustes de seguridad, dejando los datos vulnerables a un ataque.

Para solucionar estas problemáticas, las empresas utilizan sistemas de protección que generan numerosa información para monitorizar. Esta información es muy difícil de interpretar en su totalidad para poder reconocer los problemas reales. De hecho, el

volumen de datos generados es enorme, siendo totalmente inútil, a menos que pueda ser rápidamente analizado y filtrado en alertas procesables.

Con un IDS (Intrusion Detection System) integrado en un SIEM (Security Information and Event Management), los administradores de sistemas cuentan con un método efectivo para automatizar procesos y centralizar la gestión de la seguridad de forma que ayude a simplificar la difícil tarea de proteger información sensible. Además, permiten estandarizar los datos, ya que, al obtener miles o millones de entradas provenientes de diferentes sistemas y fuentes, todo puede ser transformado a un formato común y utilizado para ejecutar un análisis y correlación reduciendo la carga de trabajo del equipo.

Estas herramientas son utilizadas para aumentar la seguridad de la red, vigilando el tráfico, examinando y analizando paquetes en busca de datos sospechosos con el objetivo de detectar comportamientos anómalos y patrones de compartimiento maliciosos dentro la red de la organización.

En este proyecto se pretende desplegar un IDS (Zeek IDS) e integrarlo en un SIEM (ELK) logrando un análisis exhaustivo de la red de la organización, vigilando y monitorizando el tráfico y detectando posibles eventos maliciosos. De esta forma, se aporta un grado de seguridad de tipo preventivo ante cualquier actividad sospechosa haciendo que los administradores de sistemas reciban alertas anticipadas y puedan actuar con antelación suficiente para mitigar el problema.

1.2. Objetivos del Trabajo

Los principales objetivos de este trabajo de fin de máster se han agrupado en dos grandes bloques:

Objetivos de investigación

Centrados en la investigación y el análisis de los componentes principales.

- Investigación y estudio de sistemas IDS.
- Investigación y análisis del sistema IDS elegido: Zeek IDS.
- Investigación, análisis y estudio de sistemas SIEM.

- Investigación y estudio del sistema SIEM elegido: ELK.
- Viabilidad y alcance de la integración de Zeek IDS con ELK.
- Estudio de los requisitos necesarios para la integración.

Objetivos de implantación

Estos objetivos resumen las principales tareas de implantación, despliegue y configuración de los sistemas utilizados.

- Despliegue e instalación de Zeek IDS.
- Configuración del sistema IDS.
- Análisis sobre configuración de reglas y pruebas de detección.
- Despliegue e instalación de ELK.
- Configuración del sistema SIEM.
- Realización de pruebas de funcionamiento y visualización de logs.
- Análisis e integración de IDS con SIEM.
- Pruebas de monitorización para comprobar el correcto funcionamiento de ambos sistemas.

Además de los dos bloques anteriores, cabe destacar una serie de objetivos adicionales relacionados con las entregas y documentación de la memoria del trabajo:

- Desarrollo y redacción de las entregas parciales de la memoria del trabajo.
- Envío de las entregas parciales en tiempo.
- Recopilación de la distinta información obtenida de cada entrega parcial y desarrollo de la memoria.
- Desarrollo de la memoria final.
- Preparación del vídeo de presentación.

Analizando los objetivos anteriores, se puede observar que el trabajo tiene dos partes claramente diferenciadas: búsqueda de información e investigación sobre las herramientas utilizadas IDS y SIEM, y la parte más práctica y de aplicación como es el despliegue y configuración de las herramientas.

Por último, y tras la consecución del resto de objetivos, el sistema implementado, a su vez, debería de cumplir con los siguientes objetivos finales:

- Detección de actividades anómalas o sospechosas que ocurran en la red.
- Reducción de los riesgos de intrusión minimizando posibles daños graves.
- Ayuda a evitar comportamientos indebidos dentro de la red como el abuso de privilegios.

1.3. Enfoque y método seguido

El enfoque de este proyecto consistirá en el análisis y uso de la tecnología Zeek IDS y su integración con ELK, con la finalidad de detectar actividades sospechosas y anómalas que ocurran en la red de alguna organización. De esta forma, se logrará evitar comportamientos indebidos por usuarios no autorizados en la red y se reducirá el riesgo de intrusión al máximo posible.

El proyecto cuenta con dos bloques característicos:

- **Sección de investigación** que abarca el estudio y análisis desde el punto de vista teórico de los sistemas IDS y SIEM, centrándose en Zeek IDS y ELK. Esta parte se entregará en la segunda práctica (PEC 2).
- **Sección de implementación** que abarca el despliegue, instalación y configuración de todos los componentes del sistema consiguiendo como objetivo final la detección de actividades sospechosas en red y su correcta visualización a través del sistema. Esta parte se entregará en la tercera práctica (PEC 3).

El tiempo límite necesario para la realización del proyecto es un poco escaso, por lo que se intentará lograr el desarrollo de todos los objetivos. En función de los avances y los problemas que aparezcan, la planificación puede ser modificada de forma que se solapen tareas de investigación y de implementación. De esta forma, se intentarán cumplir en plazo con todos y cada uno de los objetivos detallados en la sección anterior.

También cabe destacar que en la planificación del proyecto no se han tenido en cuenta vacaciones ni posibles viajes. Por lo tanto, es probable que se minimice el ritmo de avance del proyecto en ciertas etapas.

Sin embargo, se va a procurar cumplir con todos los objetivos e hitos del proyecto siguiendo la planificación detallada y entregando la documentación solicitada en cada una de las PEC.

1.4. Planificación del Trabajo

La planificación del proyecto se ha agrupado en cuatro bloques: Planificación, Investigación, Implantación y Defensa y Presentación, abarcando desde el 23 de Febrero de 2020 hasta el 2 de Junio de 2020 como fecha de entrega final.

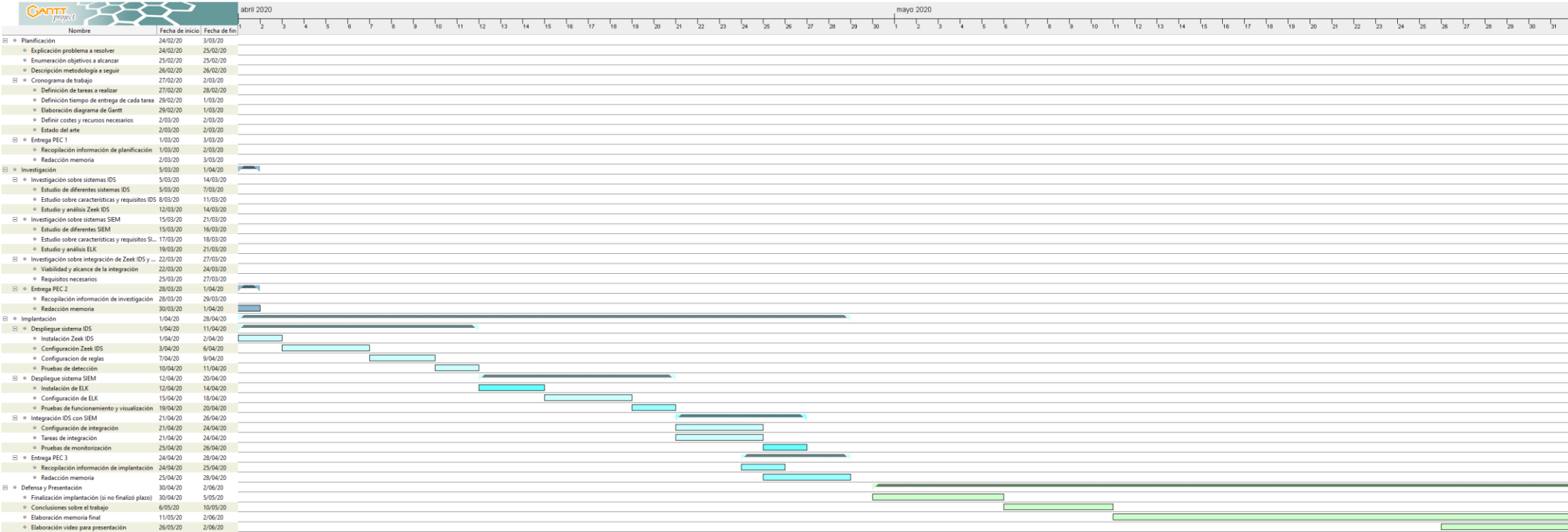
Elemento	Actividad	Inicio	Duración	Finalización
1	Planificación			
1.1	Explicación problema a resolver	23/02/2020	2	24/02/20
1.2	Enumeración de objetivos a alcanzar	25/02/2020	1	25/02/20
1.3	Descripción metodología a seguir	26/02/2020	1	26/02/20
1.4	Cronograma de trabajo			
1.4.1	Definición de tareas a realizar	27/02/2020	2	28/02/20
1.4.2	Definición tiempo de entrega de cada tarea	29/02/2020	2	01/03/20
1.4.3	Elaboración diagrama de Gantt	29/02/2020	2	01/03/20
1.5	Definir costes y recursos necesarios	02/03/2020	1	02/03/20
1.6	Estado del arte	02/03/2020	1	02/03/20
1.7	Entrega PEC 1			
1.7.1	Recopilación información de planificación	01/03/2020	2	02/03/20
1.7.2	Redacción memoria	02/03/2020	2	03/03/20
2	Investigación			
2.1	Investigación sobre sistemas IDS			
2.1.1	Estudio de diferentes sistemas IDS	05/03/2020	3	07/03/20
2.1.2	Estudio sobre características y requisitos de sistemas IDS	08/03/2020	3	11/03/20
2.1.3	Estudio y análisis de Zeek IDS	12/03/2020	3	14/03/20
2.2	Investigación sobre sistemas SIEM			
2.2.1	Estudio de diferentes sistemas SIEM	15/03/2020	2	16/03/20
2.2.2	Estudio sobre características y requisitos de sistemas SIEM	17/03/2020	2	18/03/20
2.2.3	Estudio y análisis de ELK Stack	19/03/2020	3	21/03/20
2.3	Investigación sobre integración de Zeek IDS y ELK			
2.3.1	Viabilidad y alcance de la integración	22/03/2020	3	24/03/20
2.3.2	Requisitos necesarios	25/03/2020	3	27/03/20
2.4	Entrega PEC 2			
2.4.1	Recopilación información de investigación	28/03/2020	2	29/03/20

2.4.2	Redacción memoria	30/03/2020	3	01/04/20
3	Implantación			
3.1	Despliegue sistema IDS			
3.1.2	Instalación de Zeek IDS	01/04/2020	2	02/04/20
3.1.3	Configuración de Zeek IDS	03/04/2020	4	06/04/20
3.1.4	Configuración de reglas	07/04/2020	3	09/04/20
3.1.5	Pruebas de detección	10/04/2020	2	11/04/20
3.2	Despliegue sistema SIEM			
3.2.1	Instalación de ELK	12/04/2020	3	14/04/20
3.2.2	Configuración de ELK	15/04/2020	4	18/04/20
3.2.3	Pruebas de funcionamiento y visualización	19/04/2020	2	20/04/20
3.3	Integración IDS con SIEM			
3.3.1	Configuración de la integración	21/04/2020	4	24/04/20
3.3.2	Tareas de integración	21/04/2020	4	24/04/20
3.3.3	Pruebas de monitorización	25/04/2020	2	26/04/20
3.4	Entrega PEC 3			
3.4.1	Recopilación información de implantación	24/04/2020	2	25/04/20
3.4.2	Redacción memoria	25/04/2020	4	28/04/20
4	Defensa y Presentación			
4.1	Finalización implantación (si no se finalizó en plazo)	30/04/2020	6	05/05/20
4.2	Conclusiones sobre el trabajo	06/05/2020	5	10/05/20
4.3	Elaboración memoria final	11/05/2020	22	02/06/20
4.4	Elaboración video para la presentación	26/05/2020	8	02/06/20

1.5. Planificación temporal detallada



DiagramaGantt.png



En el diagrama de Gantt anterior, se detallan cada una de las tareas que se van a llevar a cabo para el desarrollo satisfactorio del proyecto.

Como se puede observar, el proyecto comenzó a desarrollarse desde finales de Febrero de 2020 hasta Junio de 2020. Durante este proceso, se van a investigar herramientas, librerías y diversas tecnologías para cumplir los objetivos anteriores. Además, se van a realizar diversas pruebas de monitorización y detección de amenazas de red para completar el resultado.

Una vez que los sistemas sean implementados, se continuará con la creación de la documentación, lo cual abarcará desde principios de Mayo hasta mediados de Junio. Posteriormente, se continuará con la preparación de la defensa y de un vídeo explicativo del sistema.

1.6. Estado del Arte

La ciberdelincuencia va creciendo año tras año, siendo fundamental implantar defensas, ya que es muy arriesgado pensar que una organización nunca va a sufrir un incidente de seguridad. Las soluciones de ciberseguridad de alerta de los sistemas IDS y SIEM son clave para desplegar un plan de seguridad corporativa TI.

Por lo tanto, es de especial importancia el uso de soluciones de ciberseguridad que permitan identificar comportamientos sospechosos y reaccionar y abortar ataques tanto internos como externos. Los sistemas IDS y SIEM son una de estas soluciones.

Analizando el panorama actual, existe gran variedad de sistemas IDS Opensource, entre los cuales, podemos destacar los siguientes:

Snort

Snort es el IDS veterano por excelencia. La primera versión apareció en 1998 y desde entonces, fue evolucionando poco a poco hasta convertirse en un estándar para sistemas de detección de intrusiones.

Como ventaja principal cabe destacar el fuerte apoyo de la comunidad con el que cuenta, siendo, por tanto, una herramienta muy probada. Sin embargo, Snort no

cuenta con una interfaz gráfica de usuario por defecto, por lo que no es tan fácil de administrar como otros IDS.

Suricata

Suricata suele ser el sustituto de Snort. Presenta una arquitectura muy distinta, aunque se comporta de forma parecida a Snort utilizando las mismas firmas.

Suricata funciona aprovechando los procesadores multi-núcleo y multi-threading, a diferencia de Snort que se ejecuta en modo uni-hilo utilizando únicamente un núcleo de la CPU al mismo tiempo.

Además, Suricata nos proporciona el poder mediante scripting pudiendo combinar varias reglas y analizando no sólo paquetes, sino también certificados TLS/SSL, solicitudes HTTP, peticiones DNS...

Zeek IDS

También, conocido como Bro, es un sistema un poco distinto a los dos anteriores. El tráfico capturado generará una serie de eventos, como, por ejemplo, un inicio de sesión de usuario a un FTP.

La principal ventaja de Zeek IDS es el intérprete de políticas de script, permitiendo personalizar la detección de eventos con su propio lenguaje de administración.

Con Zeek IDS es posible automatizar las tareas de análisis y recolección de datos, por ejemplo, podríamos descargar ficheros encontrados en nuestro entorno, remitirlos para un análisis de malware, notificar si se encuentra un problema y finalmente introducirlo en la lista negra la fuente del mismo. Es posible incluso apagar el equipo remoto del usuario que lo descargó.

Zeek IDS es más complicado de utilizar que Snort o Suricata, sin embargo, permite detectar más patrones de actividad que la mayoría de IDS. Concretamente, en este proyecto, se hará uso de Zeek IDS como sistema IDS.

La información obtenida por el sistema IDS es recopilada y monitorizada por un sistema SIEM. Como sistemas SIEM más conocidos, cabe destacar Splunk, QRadar

de IBM, Fortisiem, AlienVault o ELK (Elastic). Concretamente, en este trabajo, se hará uso de ELK.

Existen numerosas empresas importantes que utilizan Elastic como solución SIEM:



Ilustración 1: Empresas relevantes que utilizan Elastic como solución SIEM.

Con ELK podremos realizar monitorizaciones masivas de servidores, tratados de logs y presentación de los datos recopilados:



Ilustración 2: Tratamiento de logs obtenidos con Zeek IDS por parte de ELK.

ELK está formado por las siguientes tecnologías:

- **Logstash** – Tratamiento de datos. Intercepta cualquier dato, antes de ser enviado a Elasticsearch, para parsearlo y tratarlo si, si procede, en un formato específico y estandarizado.
- **Elasticsearch** – Persistencia de datos. Se trata de un motor de búsqueda que utiliza una base de datos NoSQL que almacena documentos en formato JSON.
- **Kibana** – Presentación de la información. Kibana utiliza la información almacenada en Elasticsearch proporcionando visualización en tiempo real de los índices de manera directa y sencilla, permitiendo una visión global de toda la información relevante de los sistemas y servicios que estamos monitorizando.
- **X-Pack** (Opcionales y de pago) – Extensión de características.

De esta forma, un sistema como este, implementado en nuestra infraestructura informática, nos permitirá mejorar en operativas de seguridad, métricas, monitorización de servicios, entre otras muchas características.

1.7. Presupuesto y recursos necesarios

Para la correcta realización del proyecto, se requiere de una importante inversión de tiempo (detallada en los puntos anteriores). Además, se necesitarán los siguientes recursos:

- **Ordenador:** 1100€. Utilizado para:
 - Redactar la memoria.
 - Instalar el entorno virtual donde desplegar, configurar y hacer pruebas con Zeek IDS y ELK.
- **Conexión a Internet:** 46 €/mes. Utilizado para:
 - Descargar las herramientas.
 - Monitorizar la salida a internet.

Por tanto, el precio total con respecto a los recursos necesarios será de 184 € de conexión a internet más el coste del ordenador, haciendo un total de 1284 €.

A los precios anteriores, se le sumará el precio por hora del total de horas invertidas en el sistema. A continuación, se va a realizar un pronóstico lo más exacto posible del desglose por horas.

- | | |
|-----------------------------|-----------|
| • Investigación y análisis: | 120 horas |
| • Implementación y pruebas: | 100 horas |
| • Documentación: | 50 horas |

Para relacionar un precio a cada hora, se va a proceder a realizar diversos estudios que detallan cuánto gana un ingeniero informático. El Consejo General de Colegios Oficiales de Ingeniería en Informática elaboró en abril de 2015 un estudio en el que se indican los sueldos según el puesto de trabajo.

ESTUDIO NACIONAL SOBRE LA SITUACIÓN LABORAL DE LOS PROFESIONALES DEL SECTOR DE TECNOLOGÍAS DE LA INFORMACIÓN

Gráfica 35. Salario según el puesto de trabajo

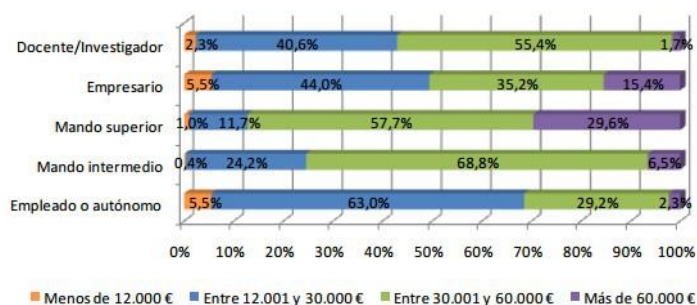


Ilustración 3: Diagrama que indica el salario según el puesto de trabajo.

En este caso nos situaremos en el rango entre 12.000€ y 30.000€, dado que un alto porcentaje recibe ese salario como se muestra en el gráfico anterior. Para realizar el estudio del presupuesto por horas se considerará que se gana alrededor de 20.000€, que está en el punto medio de dicho rango.

Teniendo en cuenta esta cifra, en España la jornada laboral es de un máximo de 40 horas a la semana (5 días trabajando 8 horas diarias) y el año tiene alrededor de 52 semanas, lo que hace un máximo de 2080 horas. Sin embargo, a estas horas tenemos que restarle 21 días de vacaciones, 14 de festivos y 3 de asuntos propios o enfermedad, es decir, 304 horas, por lo que las horas de trabajo se reducen a 1776 horas por año. Por lo tanto:

$$20.000 / 1.776 = 11,26€/hora$$

Es decir, vamos a recibir 11,26€ por cada hora de trabajo. Si el total del proyecto han sido alrededor de:

$$560 \text{ horas} \cdot 11,26 \text{ €/hora} = 6.305,6 \text{ €}$$

Sumando los costes de material y horas invertidas se va a obtener un presupuesto de:

$$6.305,6 \text{ € horas invertidas} + 1284 \text{ € de material} = 7.589,6 \text{ €}$$

Por lo tanto, podemos concluir con que el presupuesto aproximado del proyecto durante los próximos cuatro meses será de 7.589,6 €.

2. Fase de Investigación

2.1. Sistemas de Detección de Intrusiones (IDS)

En esta sección, se introducirán los sistemas de detección de intrusiones haciendo especial énfasis en qué son, los tipos existentes, las categorías, funcionamiento, entre otros aspectos.

Por otro lado, se hará una breve comparación entre los IDS más usados, centrándonos, por último, en Zeek IDS como el sistema IDS que se desplegará.

2.1.1. Introducción

El término IDS (Sistema de detección de intrusiones) hace referencia a un mecanismo que, de forma sigilosa, escucha el tráfico en la red con el objetivo de detectar actividades anormales o sospechosas reduciendo el riesgo de intrusión.

Actualmente, es muy difícil mantener seguros los distintos equipos o dispositivos conectados a la red, ya que a diario se publican numerosas brechas de seguridad. Los IDS nos permiten monitorizar el uso de estos sistemas y dispositivos permitiendo detectar actividades sospechosas lo más rápido posible. Estas actividades pueden abarcar desde intentos para abusar de privilegios por parte de usuarios internos hasta intentos para explotar vulnerabilidades de seguridad por parte de usuarios externos.

El funcionamiento de estos sistemas se basa, como hemos mencionado anteriormente, en el análisis pormenorizado del tráfico de red. Cuando el tráfico entra en el sistema es comparado con firmas de ataques conocidos o comportamientos sospechosos, como puede ser el escaneo de puertos, o análisis de paquetes con cabeceras o parámetros sospechosos. El IDS no sólo analiza qué tráfico es, sino también revisa su comportamiento y contenido.

Generalmente, estas herramientas se integran con un firewall. Un IDS por sí mismo es incapaz de detener los ataques. Debido a esto, se requiere que trabajen conjuntamente con un dispositivo con funcionalidad de firewall, convirtiéndose en una herramienta muy poderosa al disponer de la inteligencia del IDS y el poder de bloqueo de un firewall, al ser el punto donde forzosamente deben pasar los paquetes y pueden ser bloqueados antes de penetrar en la red.

Los IDS suelen disponer de una base de datos que contiene las firmas de numerosos ataques conocidos. De esta forma, si al comparar el tráfico de red detectado con estas firmas, se da positivo, el comportamiento puede ser sospechoso o fraudulento. Por lo tanto, estas firmas permiten al IDS distinguir entre el uso normal del equipo y el uso fraudulento, y entre el tráfico normal de la red y el tráfico sospechoso resultado de un ataque o intento del mismo.

2.1.2. Tipos de IDS

Los sistemas IDS pueden ser clasificados dependiendo de donde se produzca la detección (red o host) o dependiendo del método de detección que se ha empleado (firma o basado en anomalías).

2.1.2.1. Actividad Analizada

Teniendo en cuenta donde se ha producido la detección, se pueden distinguir entre dos tipos: HIDS (HostIDS) o NIDS (NetworkIDS).

2.1.2.1.1. Sistema de Detección de Intrusiones en el host (HIDS)

Con este sistema, se garantiza la seguridad dentro del host. El funcionamiento de un HIDS, depende del éxito de los intrusos, que suelen dejar rastros de sus actividades en el equipo atacado al intentar adueñarse del mismo con el fin de llevar a cabo otras actividades. El HIDS detectará tales modificaciones en el equipo afectado y hará un reporte de sus conclusiones.

El HIDS se encuentra en un host particular, por lo tanto, su software cubre una amplia gama de sistemas operativos como Windows, Linux, Solaris, etc. El sistema actúa como un demonio o servicio estándar en el sistema de un host que analiza la información almacenada en los registros del sistema y logs. También, captura paquetes de la red que se introducen o salen del host para verificar las señales de intrusión: ataques DDoS, puertas traseras, troyanos, accesos no autorizados, ejecución de códigos malignos o ataques de desbordamiento de búfer.

El HIDS actúa monitorizando los cambios de variables en el sistema del host. Estas variables pueden incluir procesos del sistema, entradas de registro, uso de CPU, accesos a ficheros, políticas, cuentas de usuarios o logs. Si se excede el umbral programado, se enviará una alerta a los administradores. Los HIDS pueden ayudar a detectar comportamientos y actividades anómalos que pueden comprometer un equipo, sin embargo, el administrador del sistema deberá emplear más tiempo para analizar la salida de los logs del HIDS y eliminar las alertas de falso positivos.

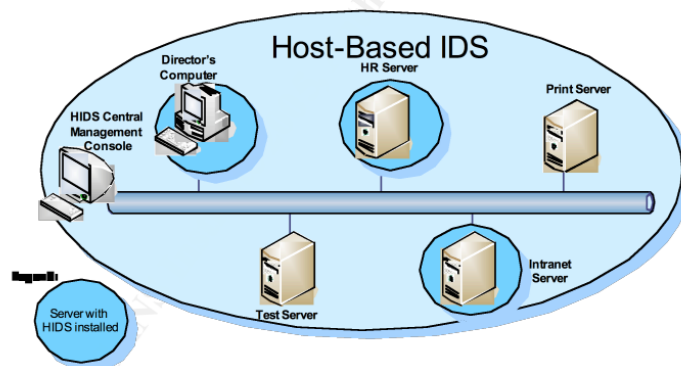


Ilustración 4: Sistema de detección de intrusiones en el host.

2.1.2.1.2. Sistema de Detección de Intrusiones de red (NIDS)

Este sistema garantiza la seguridad en la red. NIDS se encarga de verificar los paquetes de datos, capturándolos, decodificándolos para pasarlo al segmento de red y así descubrir si se ha producido alguna actividad maliciosa o anormal. De esta forma, se analizan los paquetes IP con el objetivo de localizar la firma del atacante. A diferencia de los HIDS, NIDS puede monitorizar un segmento de red entero y puede ser desplegado rápidamente.

El NIDS pone uno o varios de los adaptadores de red del sistema en modo promiscuo. Es muy corriente encontrar diversos IDS en diferentes partes de la red. Generalmente, se suelen colocar sondas fuera de la red para estudiar posibles ataques, a la vez que

también se colocan sondas internas para analizar solicitudes que hayan pasado a través del firewall o que se han realizado desde dentro.

Sin embargo, los NIDS presentan una serie de limitaciones a la hora de procesar el tráfico de red:

- **Redes de Switches (Red Nivel 2).** NIDS presenta varios inconvenientes en este tipo de redes. Por defecto, un switch transmite paquetes directamente al receptor y no a la red entera como ocurre en el caso de un hub tradicional. Para resolver esto, se puede utilizar un puerto de spanning o un TAP de red, de forma que reciba todo el tráfico que se está transmitiendo por el switch.
- **Redes de alta velocidad.** La velocidad es un factor clave a tener en cuenta a la hora de desplegar un IDS debido a que puede no ser capaz de capturar todo el tráfico al excederse su límite. Además, un atacante puede colapsar el tráfico de red para ejecutar un exploit y que ese exploit no sea detectado.
- **Redes encriptadas.** Si un atacante usa SSH para conectarse a una máquina, el NIDS no podría enviar alertas debido a que el tráfico está encriptado. En este caso, el HIDS puede determinar el comportamiento en esta máquina en concreto.

Los sistemas NIDS y HIDS son complementarios, por lo que la mayoría de proveedores suelen desarrollar arquitecturas que integran ambos. Posteriormente, el equipo de seguridad podría acceder a un servidor centralizado para analizar los logs, los reportes, manejar la configuración de los IDS y customizar las políticas de detección de intrusos.

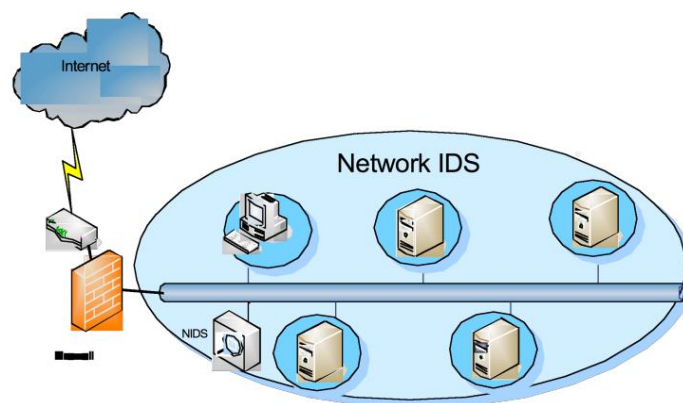


Ilustración 5: Sistema de detección de intrusiones en la red.

2.1.2.2. Método de detección

En función del método de detección empleado, se puede distinguir entre sistemas de detección de intrusiones basados en firma o en anomalías.

2.1.2.2.1. Basados en firmas

Se basa en la búsqueda de evidencias de ataques basándose en el alto conocimiento adquirido de ataques conocidos. Este tipo de IDS puede solo detectar los ataques cuya firma se encuentre registrada en la base de datos. Normalmente, se requiere de numerosas actualizaciones para mantener actualizada la base de datos con los datos de todas las firmas de ataques conocidos.

La eficiencia de estos IDS reside en la precisión de las firmas. Esto permite que los atacantes puedan acceder a los equipos utilizando técnicas de evasión, haciendo que los ataques sean indetectables. Un exploit puede cambiarse fácilmente haciendo que el ataque no sea detectado.

Es posible crear firmas genéricas que detecten más de una variante del mismo ataque, aunque para ello es necesario tener un alto conocimiento del funcionamiento y código de estos ataques junto con los componentes de red involucrados, con el objetivo de bloquear actividades maliciosas sin denegar tráfico válido. Estas firmas definen las características de un ataque (protocolo, servicio, patrones).

2.1.2.2.2. Basados en anomalías.

Este tipo de sistemas IDS consisten en la búsqueda de evidencias de ataques basados en el conocimiento acumulado. Por ejemplo, un alto uso de CPU combinado con otras métricas puede indicar una intrusión en progreso.

Fue introducido para identificar ataques desconocidos debido al rápido desarrollo del malware. El enfoque básico de este tipo de IDS es utilizar el aprendizaje automático para crear un modelo de actividad fiable y, posteriormente, comparar el nuevo comportamiento con este modelo. Debido a que estos modelos pueden ser entrenados en función de las configuraciones de hardware o aplicaciones, el método

basado en el aprendizaje automático tiene mejores propiedades y formas de detectar que los IDS tradiciones basados en firmas.

Sin embargo, es posible que este tipo de detección pueda sufrir falsos positivos, ya que la actividad legítima pueda ser clasificada como maliciosa. Esto hace que sea necesario actualizar el modelo utilizado continuamente para reflejar el comportamiento de un usuario normal y reducir los falsos positivos.

2.1.3. Herramientas IDS más destacadas

Una vez definida la diferencia entre los distintos tipos de IDS que existen, se van a enunciar y describir brevemente las cuatro herramientas IDS más utilizadas e importantes para combatir los ataques, mencionando las ventajas de cada uno y cuando convendría utilizarse en cada caso.

En esta sección, vamos a obviar a Zeek IDS, ya que lo describiremos en el siguiente apartado. En esta parte, únicamente, vamos a centrarnos en el resto de las herramientas IDS más importantes:

2.1.3.1. Snort

Snort surgió en 1998 y desde entonces se ha ido convirtiendo en un estándar para los sistemas de detección de intrusiones.



Ilustración 6: Logotipo de Snort.

Snort es de las herramientas IDS principales y más conocidas. Su principal ventaja radica en la capacidad para realizar el análisis de tráfico en tiempo real y registro de paquetes en redes. Además, es una herramienta muy longeva y que se está desarrollando y mejorando continuamente, algo que nos permite implementarlo sin preocupaciones por el futuro. Cuenta con un gran apoyo de la comunidad, siendo una herramienta muy probada y testeada en todos los sentidos.

Snort es muy utilizado para detectar exploits, gusanos, explorar puertos y otras amenazas maliciosas, gracias a su funcionalidad de análisis de protocolos, búsqueda de contenido y varios preprocesadores.

Como desventaja principal, cabe resaltar que Snort no cuenta con interfaz gráfica por defecto, por lo que es más difícil de administrar que otras.

2.1.3.2. Suricata

Suricata es un sistema de detección de intrusos de red de código abierto, rápido y muy robusto. Suele ser el sustituto cuando no se utiliza Snort. Presenta una arquitectura muy distinta pero el funcionamiento es muy parecido, utilizando las mismas firmas.

El motor de Suricata permite detectar intrusos en tiempo real, prevenir intrusiones en línea y monitorizar la seguridad de la red. Permite capturar el tráfico que pasa en un flujo antes de la decodificación, es decir, a diferencia de Snort, configura los flujos separados después de capturar y especifica cómo se separará el flujo entre los procesadores.



Ilustración 7: Logotipo de Suricata.

Entre sus principales características, caben destacar las siguientes:

- **Multi-hilo.** Snort se ejecuta en modo uni-hilo, por lo que solo puede aprovechar un núcleo de CPU al mismo tiempo.
- **Aceleración mediante hardware.** Se pueden utilizar tarjetas gráficas para inspeccionar tráfico de red.
- **Extracción de ficheros.** Se puede capturar malware y estudiarlo desde Suricata.
- **Scripting.** Es muy personalizable, ya que podemos utilizar el scripting para combinar varias reglas y buscar elementos con mayor eficiencia.
- **No sólo analiza paquetes.** Suricata puede revisar certificados TLS/SSL, peticiones DNS, entre otros.

2.1.3.3. OSSEC

A diferencia de las otras dos herramientas (NIDS), OSSEC es un IDS basado en cliente (HIDS), por lo que analiza la actividad que tiene lugar a nivel interno de la máquina.

En el mundo de los HIDS no existe muchas más opciones más allá de OSSEC. Es el único sistema completo, probado y con buen soporte, es decir, es la referencia absoluta en el ámbito.

OSSEC funciona en cualquier sistema operativo y presenta una arquitectura cliente-servidor. Este tipo de arquitectura es muy importante en estos sistemas, ya que un HIDS podría ser también afectado por un ataque, siendo de vital importancia que toda la información tratada sea evacuada a una ubicación segura lo más rápido posible.

Otra ventaja por destacar es que OSSEC está equipado con una arquitectura centralizada y multiplataforma que permite a los administradores supervisar de forma precisa varios sistemas desde un servidor único. Dado que los despliegues pueden alcanzar cientos o miles de clientes, por el bien de los administradores es necesario hacerlo en bloque.



Ilustración 8: Logotipo de OSSEC.

Como principales características cabe destacar:

- Agentes disponibles para cualquier sistema operativo. Instalación sencilla.
- Funcionalidad muy extensa abarcando tareas como como análisis de registros, pruebas de integridad, supervisión del registro de Windows, detección de rootkits, alertas basadas en el tiempo y respuesta activa.
- Agentes compilados para Windows.

2.1.4. Herramienta a implementar: Zeek IDS

Tras el análisis de las herramientas más importantes mencionadas anteriormente, en este proyecto, se utilizará Zeek IDS (también conocido como Bro IDS) como sistema de detección de intrusiones.

Zeek IDS es una de las herramientas IDS más conocidas y utilizadas junto a Snort y Suricata. A diferencia de estas dos herramientas, Zeek es capaz de detectar más patrones de actividad. Por lo tanto, en esta sección se va a proceder a analizar la herramienta brevemente, analizando todas las ventajas que presenta.

Zeek es un analizador de tráfico de red, tanto basado en firmas como en anomalías, que se utiliza para recopilar datos de la red, realizar investigaciones forenses, etc. El tráfico capturado generará una serie de eventos como por ejemplo un inicio de sesión de un usuario a un FTP o conexión a un servicio web.

La principal ventaja reside en que Zeek proporciona una enorme cantidad de ficheros de logs que recogen toda la actividad que está ocurriendo en la red. Estos logs incluyen no solo información de cada conexión, sino que también datos sobre la capa de aplicación como todas las sesiones HTTP con sus determinadas URIs, cabezas, tipos MIME, respuestas del servidor; peticiones DNS, certificados SSL, sesiones SMTP, entre otros. Por defecto, Zeek escribe toda esta información de forma ordenada en diferentes ficheros logs que pueden ser procesados por software externo como backends que hagan de interfaz con bases de datos externas, por ejemplo.

Con Zeek es posible automatizar las tareas de análisis y recolección de datos. Por ejemplo, se podrían descargar ficheros encontrados en el entorno, pasarlos por un analizador de malware, notificar si se encuentra un problema y posteriormente, introducirlo en la lista negra. Finalmente, incluso se podría apagar el equipo remoto del usuario que lo utilizó.



Ilustración 9: Logotipo de Zeek.

Como características principales podemos destacar las siguientes:

- Cuenta con un intérprete de políticas de script con su propio lenguaje de administración que ofrece posibilidades de automatización y análisis muy interesantes.
- Incluye un conjunto de archivos de registro para registrar actividades de red como sesiones HTTP con URIs, encabezados de claves, respuesta de servidor, certificados SSL, sesiones SMTP, etc.
- Proporciona funcionalidad para el análisis y detección de amenazas, extracción de archivos de sesiones HTTP, detección de vulnerabilidades software, ataques de fuerza bruta SSH y validación de cadenas de certificados SSL.

2.1.4.1. Arquitectura

La arquitectura de Zeek se encuentra dividida en dos componentes principales. El **event engine (motor de eventos)** que reduce el flujo de paquetes entrantes en una serie de eventos. Estos eventos reflejan la actividad de la red, por ejemplo, describen qué se ha detectado, pero no por qué o si esta información detectada es relevante. Por ejemplo, cada petición HTTP realizada se transforma en el correspondiente evento *http_request* que contiene las direcciones IP y puertos involucrados, el URI solicitado y la versión HTTP usada. Sin embargo, este evento no refleja ninguna interpretación, por ejemplo, no dice si la URI corresponde con un sitio malware conocido.

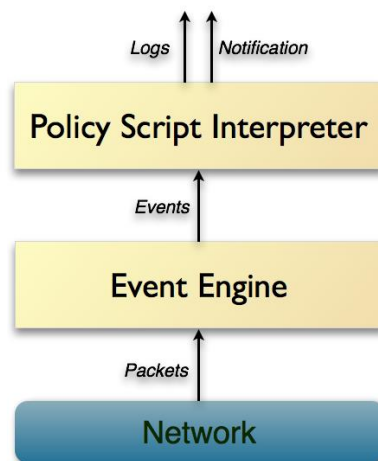


Ilustración 10: Arquitectura de Zeek IDS.

Esta interpretación necesaria reside en el segundo componente de Zeek: el **intérprete de scripts**. Este componente se encarga de ejecutar un conjunto de controladores de eventos escritos en el lenguaje de scripting de Zeek. Estos scripts pueden definir una política de seguridad de un sitio, como por ejemplo, qué acciones se deberían tomar cuando el monitor detecta diferentes tipos de actividad. Más concretamente, estas acciones pueden derivar en propiedades y estadísticas procedentes del tráfico entrante.

El lenguaje de scripting de Zeek IDS soporta numerosas funcionalidades y permite a los scripts mantener el estado durante cierto tiempo permitiendo realizar un seguimiento y correlación de la evolución que se estaría observando durante la conexión. Estos scripts pueden, además, generar alertas en tiempo real y ejecutar programas externos arbitrarios para, por ejemplo, responder ante un ataque.

2.2. Sistemas de gestión de información y eventos de seguridad (SIEM)

En esta sección, se introducirán los sistemas de gestión de Eventos e Información de Seguridad haciendo especial énfasis en qué son, los tipos existentes, las categorías, funcionamiento, entre otros aspectos.

Por otro lado, se hará una breve comparación entre los SIEM más utilizados, centrándonos, por último, en ELK Stack como el sistema SIEM que se desplegará e integrará con Zeek IDS.

2.2.1. Introducción

La mayoría de las organizaciones y empresas están conectadas internamente y con el exterior teniendo enormes ventajas operativas, pero aumentando los riesgos de ciberataques y amenazas potencialmente graves. El número de estos ciberataques crece cada día de manera exponencial, siendo cada vez más potentes y peligrosas.

Actualmente, no solo basta con prepararnos y protegernos de dichos ataques, sino que también debemos anticiparnos a ellos y reducir la amenaza lo mayor posible. Para

realizar esta tarea de prevención y otorgar mayor seguridad informática a las empresas se han desarrollado los sistemas SIEM.

Un sistema SIEM (*Security Information and Event Management*) se trata de un sistema de gestión de información y eventos de seguridad. Es una tecnología capaz de detectar rápidamente, responder y neutralizar las amenazas informáticas. Su objetivo principal es el de proporcionar una visión global de la seguridad de la tecnología de la información.

La tecnología SIEM nace de la combinación de las funciones SEM (gestión de eventos de seguridad) y SIM (gestión de información de seguridad).

- **SEM** centraliza el almacenamiento y permite un análisis casi en tiempo real de lo que está sucediendo, detectando patrones anormales y dando mayor visibilidad a los sistemas de seguridad.
- **SIM** recopila los datos a largo plazo en un repositorio central para luego analizarlo proporcionando informes automatizados al personal de seguridad informática.

Estas funciones nos permiten actuar rápidamente sobre los ataques, ya que nos ofrecen más visibilidad y permiten utilizar los datos para la supervisión y el análisis de seguridad en tiempo real, avisando de los ataques que se están produciendo o incluso los que se van a producir en un futuro cercano.

La tecnología SIEM controla tanto amenazas externas como internas, garantizando un control de las amenazas cibernéticas más difíciles de detectar. A diferencia de un firewall u otra tecnología similar que no nos permiten reaccionar a tiempo frente a numerosos tipos de amenazas.

Con la información obtenida, se realizan informes que se distribuyen al personal de seguridad. Con estos informes podemos analizar una alerta antes de que ocurra el desastre. De esta forma, la recogida de información se convierte en algo fundamental y de numeroso valor para las empresas, ya que también permite que mejoren en capacidades de investigación y ayuda a cumplir los mandatos de seguridad.

2.2.2. Funcionamiento

Los sistemas SIEM ofrecen visibilidad y capacidad para detectar amenazas haciendo que el personal de seguridad sepa cuál es el mejor modo para actuar. Además, monitorizan las actividades que se realizan dentro de la red, recogiendo información necesaria sobre la actividad de los usuarios y diversos dispositivos.

Esta tecnología mezcla la interpretación de registros y el almacenamiento de datos con la gestión de los mismos para reconocer patrones en la detección de ataques y prevenir o hacer frente a las amenazas.

El SIEM centraliza la información para visualizar los potenciales riesgos. Para ello, utiliza un repositorio central para analizar las tendencias y patrones que ayudarán a crear informes automáticos de posibles riesgos. El repositorio se llena mediante registros de seguridad proporcionados por eventos recopilados de las fuentes monitorizadas, desde antivirus, firewall o sistema de detección de intrusiones (IDS). Concretamente, un SIEM observa y graba absolutamente todo lo que pasa en la red de nuestra empresa.

Otra ventaja principal de la tecnología SIEM consiste en la estandarización de datos, es decir, permite el procesamiento de múltiples sucesos provenientes de diferentes fuentes sin importar el tamaño de la red o el sistema. De esta forma, ofrece la posibilidad de analizar todo tipo de dispositivos y tráfico sin importar el número de equipos o usuarios, pudiendo priorizar, analizar y hacer frente a todo tipo de amenazas, gracias a su eficacia y rapidez.

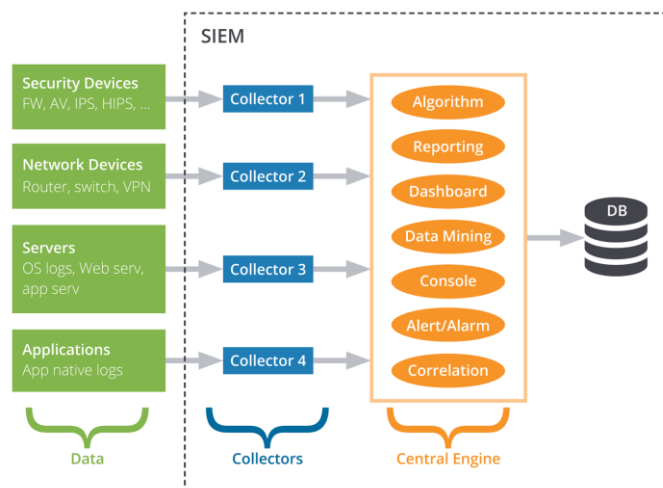


Ilustración 11: Arquitectura típica de un sistema SIEM.

Por lo tanto, el funcionamiento de estos sistemas reside en la recopilación de eventos relacionados con la seguridad de los dispositivos de usuario final, servidores, equipos de red como firewall, antivirus o IDS (en este proyecto, nos centraremos en la integración de un sistema IDS con el sistema SIEM). Todos ellos trabajan enviando eventos al repositorio central del SIEM que es el que gestiona, inspecciona y determina comportamientos anómalos. Finalmente, el SIEM tiene que tener creadas condiciones óptimas para detectar dichas anomalías de los eventos que inspecciona.

2.2.3. Herramientas SIEM más destacadas

En esta sección, se van a enunciar y describir brevemente las cinco herramientas SIEM más utilizadas e importantes para combatir los ataques, mencionando las ventajas de cada una y cuándo convendría utilizarse en cada caso.

ELK se explicará en la siguiente sección, ya que será la solución utilizada en este proyecto para integrar con Zeek. Por tanto, en esta parte, únicamente, vamos a centrarnos en el resto de las herramientas SIEM más importantes.

2.2.3.1. SolarWinds Security Event Manager

SolarWinds Security Event Manager es una de las mejores y más utilizadas herramientas SIEM. La solución proporciona numerosas funcionalidades para gestionar los logs: correlación de eventos de seguridad, informes de cumplimiento y diversas funciones analíticas avanzadas. Está diseñado para empresas que buscan en concreto un sistema de monitorización de logs robusto, así como una buena priorización y respuesta a la gestión de incidentes.

Otra ventaja a destacar de la herramienta SIEM de SolarWinds es que permite administrar el almacenamiento de la tarjeta de memoria. La herramienta de informes viene con formatos que cumplen numerosos estándares de integridad de datos: PCI DSS, HIPAA, ISO, etc.



Ilustración 12: Interfaz de SolarWinds Security Event Manager.

Como ventajas principales cabe destacar:

- Permite realizar búsquedas avanzadas y análisis forense.
- Con la detección de actividad sospechosa, será cada más rápido identificando estas amenazas.
- Soporta numerosos estándares de integridad de datos: HIPAA, PCI, DSS, DISA, STIG, etc.
- Mantiene de forma continua la seguridad.

2.2.3.2. Splunk Enterprise Security

Splunk Enterprise Security es una opción muy popular y utilizada durante la última década. Es fácil de usar y entender. Todas las versiones del software vienen con un analizador de datos que permite a la empresa ordenar clasificar y filtrar de manera eficiente los registros leídos de los archivos.

La interfaz está muy bien estructurada presentando numerosos paneles personalizables: gráficos y diagramas, análisis de estadísticas, clasificaciones, revisiones e investigaciones de incidentes. Además, posee diversas características de gestión de alertas, prioridad de riesgos, etc, proporcionando servicios de seguridad a las diversas organizaciones.

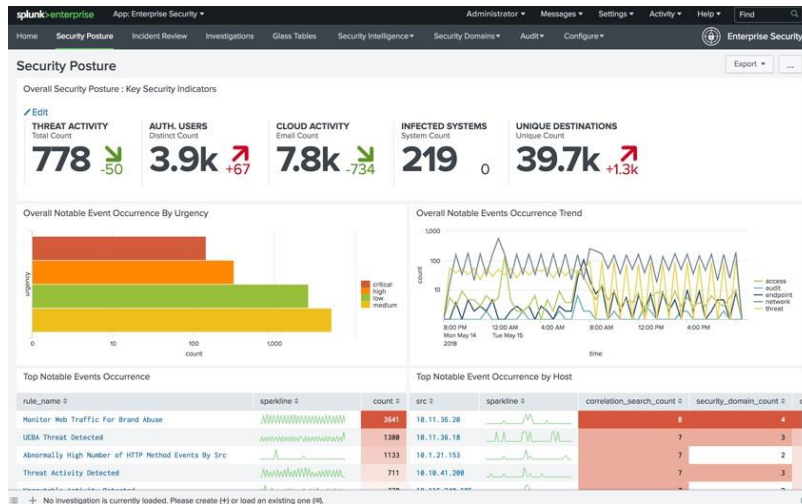


Ilustración 13: Interfaz de Splunk Enterprise Security

Como ventajas principales se pueden destacar:

- Puede funcionar con cualquier dato, aun estando en la nube.
- Permite el uso de workflows y acciones automatizadas para respuestas rápidas y precisas.
- Rápida detección de amenazas.
- Tiene la capacidad de definir secuencias de eventos.
- Permite el uso de IA y aprendizaje automático.
- Los paneles son personalizables.

2.2.3.3. McAfee Enterprise Security Manager

McAfee ofrece algunas de las mejores soluciones SIEM del mercado. El servicio McAfee SIEM ofrece a las empresas la posibilidad de recolectar una gran cantidad de logs procedentes de distintos dispositivos. Si la empresa tiene implementados otros productos de McAfee como el antivirus, tiene más sentido aún la elección de su solución SIEM para monitorizar sus operaciones.

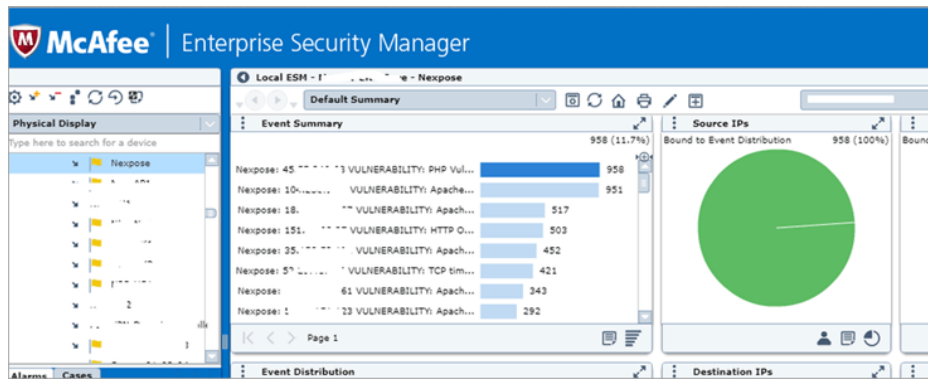


Ilustración 14: Interfaz de McAfee Enterprise Security Manager.

McAfee ESM proporciona visibilidad en tiempo real de las actividades que ocurren en el sistema, redes, bases de datos y aplicaciones de la compañía.

Esta solución ofrece una gran cantidad de funcionalidades: investigadores de incidencias, herramientas de correlación, aplicaciones de monitorización y gestión de logs, y herramientas para gestionar eventos, IA y búsqueda de logs.

Algunas de las características principales de esta solución son las siguientes:

- Priorización de alertas.
- Con el análisis avanzado de estadísticas, será más fácil detectar y priorizar las amenazas y el riesgo que conlleva.
- Presentación dinámica de datos, permitiendo investigar, contener, remediar y adaptar las alertas y patrones.
- Las interfaces de las distintas funcionalidades son modificables y fácil de usar.
- Los datos serán monitorizados y analizados desde una infraestructura heterogénea y segura.

2.2.3.4. Micro Focus ArcSight ESM

La solución SIEM ofrecida por ArcSight proporciona un conjunto de funcionalidades muy completas destinadas a compañías grandes que utilizan entornos regulados y comerciales. Esta solución ofrece diversas funcionalidades de MicroFocus que aseguran la monitorización completa de la plataforma y la gestión de enormes cantidades de datos manejados en tiempo real.

A diferencia de la mayoría de las herramientas SIEM, ArcSight ESM permite el procesamiento de datos de una amplia gama de dispositivos (500 tipos aprox.). Además, ofrece la posibilidad de utilizar sus datos fuera del propio entorno de ArcSight que puede ser de utilidad para los equipos de seguridad más expertos.

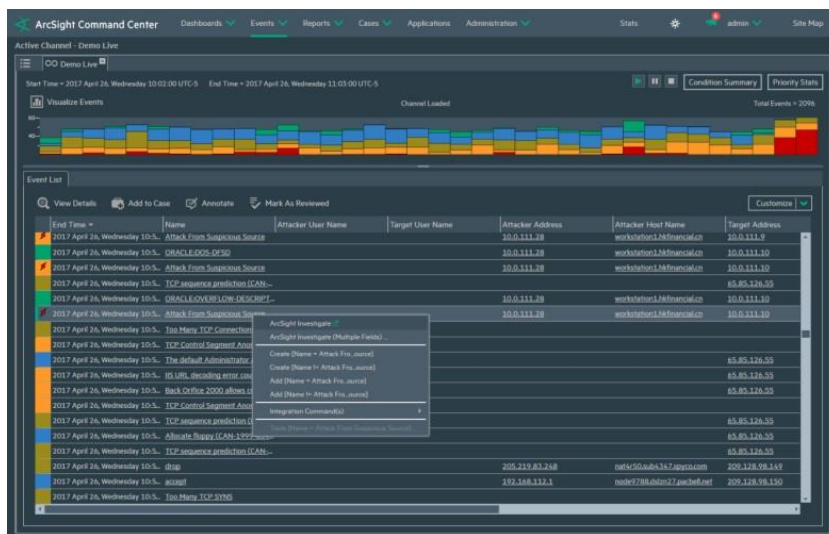


Ilustración 15: Interfaz de Micro Focus ArcSight ESM.

Esta herramienta ofrece funcionalidades de correlación distribuida y vistas a nivel de clústeres. Está disponible mediante software, AWS y Microsoft Azure.

Algunas de las características más relevantes de esta herramienta son:

- Proporciona un sistema de correlación distribuida combinando el motor de correlación del SIEM con tecnología de clústeres distribuidos.
- Utiliza agentes o conectores, soportando más de 300 conectores.
- Puede ser integrado con plataformas de IA y machine learning.

2.2.4. Herramienta a implementar: ELK Stack

Tras el análisis de las herramientas SIEM más relevantes mencionadas anteriormente, quedaría por detallar y explicar ELK Stack como quinto sistema SIEM más utilizado. Concretamente, en este proyecto se va a hacer uso de ELK como solución SIEM.

ELK ofrece un conjunto de herramientas que se combinan para crear una herramienta de administración de registros permitiendo la monitorización, consolidación y análisis de logs generados en múltiples servidores. Estas herramientas son: Elasticsearch,

Logstash y Kibana. A estas herramientas se le añadiría Beats como solución de recolección de datos.

La arquitectura de ELK Stack sería la siguiente:

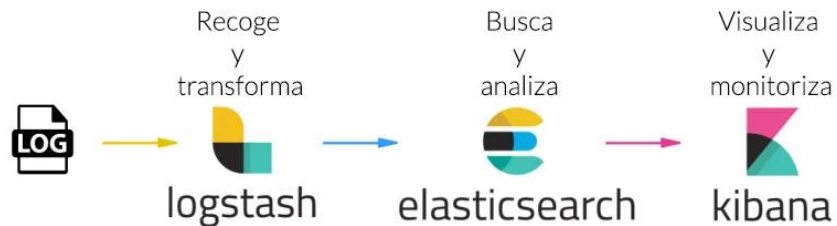


Ilustración 16: Arquitectura de ELK Stack.

- **Logstash.** Motor que permite recolectar datos de distintas fuentes, normalizarlos y distribuirlos.
- **Elasticsearch.** Motor de búsquedas distribuido que permite indexar y recuperar documentos JSON en distintos formatos.
- **Kibana.** Herramienta que permite visualizar y explorar en tiempo real grandes cantidades de datos.
- **Beats.** Cargadores de datos que se instalan en los servidores y funcionan como agentes que envían información operacional a Elasticsearch.

A modo resumen, ELK se encarga de realizar las siguientes tareas principales:

- Recolectar logs de eventos y aplicaciones.
- Procesar esta información recolectada y ponerla a disposición de quien lo necesita.
- Formatear los campos y convertirlos en opciones de búsqueda y filtrado. Numerosos logs tienen formatos irregulares o directamente no presentan ningún formato.
- Presentar esa información en distintos dashboards, en los cuales podremos realizar búsquedas, filtrados, agregaciones y ver la información de forma mucho más clara y precisa que en módulos de texto.

2.2.4.1. Recolección de datos: Beats

Los beats son recolectores de información que se encargan de recopilar la información procedente de ficheros, logs de datos, métricas del sistema (CPU, RAM), eventos, comprobaciones de servicios, análisis de red de los paquetes, tiempo de respuesta entre ellos, etc.

Los beats de tipo Filebeat enviarán esta información de logs para su carga en Elasticsearch. Esta información, también, abarca la recolección de métricas para poder monitorizar y visualizar los datos operativos.

Estos agentes envían datos de las máquinas a Logstash o a ElasticSearch. Hay distintos tipos:

- **Filebeat.** Permite la recolección, parseo y envío de datos de ficheros de logs.
- **Metricbeat.** Permite la recolección y envío de métricas a nivel de sistema: uso de CPU, uso de memoria, sistema de ficheros, etc.
- **Packetbeat.** Permite la monitorización de servicios y aplicaciones en tiempo real, obteniendo métricas como latencia, patrones de acceso, accesos HTTP...
- **Winlogbeat.** Permite la recolección y envío de eventos del log de sistemas Windows.
- **Auditbeat.** Permite la recolección y envío de métricas de auditoría de sistemas.
- **Heartbeat.** Permite la monitorización de la disponibilidad y tiempo de respuestas de servicios.

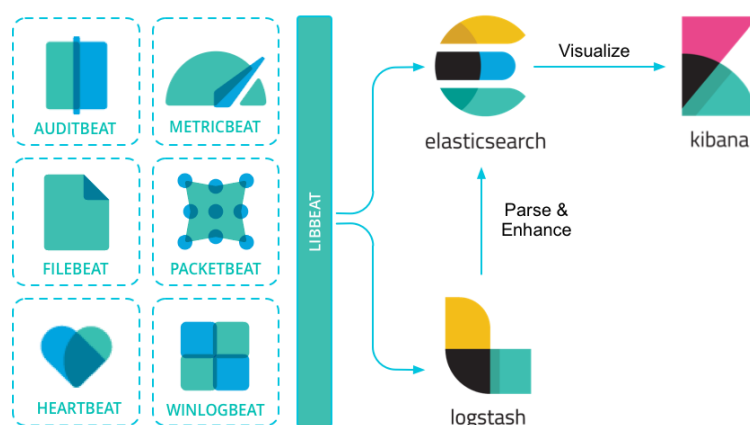


Ilustración 17: Arquitectura de Beats donde se muestran los distintos tipos de recolectores de datos.

Todos los datos recolectados por Logstash y Beats se integran en Elasticsearch y posteriormente, son visualizados gracias a Kibana.

2.2.4.2. Procesamiento de datos: Logstash

Logstash es la herramienta que recolecta la información de logs y la envía a Elasticsearch. Logstash recopila los datos de múltiples fuentes simultáneamente, los transforma y los almacena en un motor de búsqueda.

El funcionamiento de Logstash reside en la definición e integración de entradas, códecs, filtros y salidas:

- Las **entradas** son las distintas fuentes de datos, como por ejemplo, entradas de ficheros de logs generados por los distintos componentes de una plataforma.
- Los **códecs** se utilizan para convertir los distintos formatos de las entradas en otro aceptado por Logstash, permitiendo unificar el formato de los logs y facilitar su lectura y gestión.
- Los **filtros** que permiten procesar distintos eventos.
- Las **salidas** que serían los destinos en los que se almacenarán los datos ya procesados.

Logstash se basa en el uso de plugins que soportan los distintos elementos de la herramienta mencionados: entradas, códecs, filtros y salidas.

Generalmente, la carga de ficheros de logs mediante Logstash se realizará por ssh, aunque también se podrían utilizar Beats de tipo Filebeat como hemos comentado anteriormente. En cualquier caso, Logstash dispone de un gran conjunto de plugins para la entrada de datos:

- **Beats:** datos de Elastic Beats.
- **Elasticsearch:** resultados de consultas a Elasticsearch.
- **Exec:** salida de comandos.
- **File:** ficheros.
- **jdbc:** datos de BBDD.

- **http**: datos recibidos por http o https.
- **pipe**: datos obtenidos del comando pipe.
- **tcp**: datos recibidos por un socket tcp.

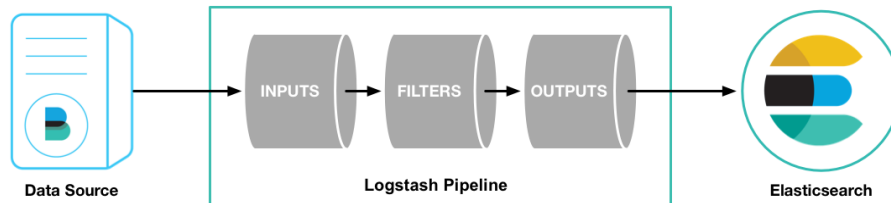


Ilustración 18: Pipeline de Logstash y envío de datos a Elasticsearch.

2.2.4.3. Almacenamiento: Elasticsearch

Elasticsearch es una base de datos distribuida (NoSQL) cuya función consiste en distribuir la información y el procesamiento en todos los nodos, siendo tolerante a fallos y presentando alta disponibilidad. De esta forma, cuando se realiza una búsqueda o consulta, como la información se encuentra distribuida, será cada nodo el que procese dicha información y devuelva los resultados. Por tanto, se pueden obtener mejores rendimientos.

La información recolectada, tanto por Logstash como por Beats, se integra en Elasticsearch. Elasticsearch es un motor de búsqueda basado en JSON que posee una interfaz RESTful, permitiendo consultar cualquier dato en tiempo real a través de consultas a su API.

Como principales características de Elasticsearch destacan las siguientes:

- Al tratarse de un sistema distribuido, se puede desplegar en una infraestructura escalable y de alta disponibilidad.
- Permite el uso de índices en un cluster (conjunto de nodos) y realizar consultas sobre uno o más de estos índices al mismo tiempo.
- Indexa todos los campos de los documentos JSON sin seguir un esquema estricto.
- Permite búsquedas vía API REST mediante ElasticSearch Query DSL: geolocalización, contextual, autocompletar, etc.

En cuanto a la infraestructura, Elasticsearch sigue un sistema distribuido que puede ser desplegado en un clúster sobre uno o varios nodos. Existen distintos tipos de nodos:

- **Nodos maestros.** Controlan el clúster.
- **Nodos de datos.** Mantienen los datos y los operan.
- **Nodos cliente.** Pequeños balanceadores que intervienen en determinados pasos del procesamiento.
- **Nodos 'Ingest'.** Dedicados a operaciones pesadas de transformación de datos.
- **Nodos 'Tribe'.** Realizan operaciones de coordinación entre nodos y clúster.

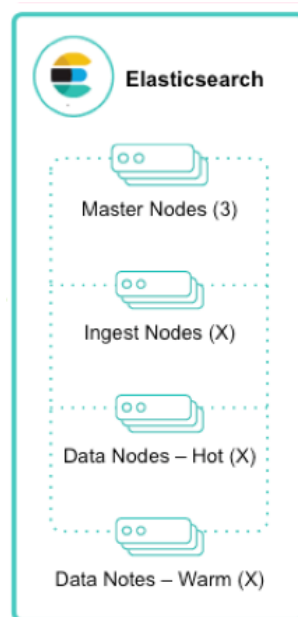


Ilustración 19: Tipos de nodos en Elasticsearch.

2.2.4.4. Visualización: Kibana

Kibana es la parte más visual de ELK Stack. Esta herramienta permite visualizar y navegar de forma interactiva por los datos almacenados en Elasticsearch.

Kibana permite generar visualizaciones sobre la información y dashboards permitiendo monitorizar por logs. Kibana representa los siguientes tipos de información:

- Histogramas, gráficos de barras, gráficos de tarta, y una gran cantidad de tipos de gráficos y diagramas más.

- Gráficos de relaciones de datos.
- Análisis de series de datos en el tiempo.
- Visualización de datos con información geoespacial en mapas.

Los procesos de Kibana realizan consultas a Elasticsearch para obtener en tiempo real el valor de los KPIs mostrados. Por ejemplo, en la siguiente figura se muestra un ejemplo de dashboard que permite darnos una idea del potencial de esta herramienta:

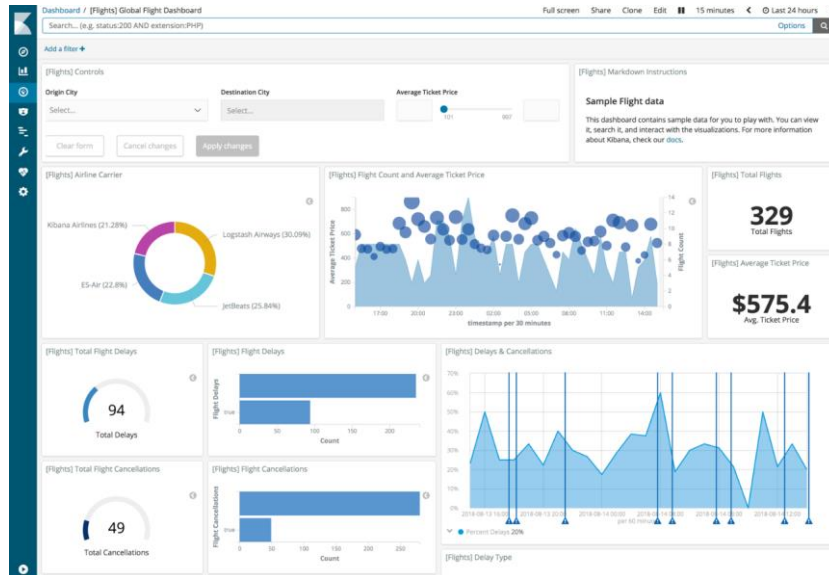


Ilustración 20: Dashboard de métricas en Kibana.

Dentro de todos los beneficios que nos proporciona Kibana, se pueden destacar los siguientes:

- Proporciona dashboards que permiten visualizar información indexada de los clusters de elastic.
- Permite búsquedas de información indexada en tiempo real.
- Permite manipular, buscar o visualizar la información de logs almacenada en Elasticsearch.
- Permite ejecutar queries de datos y visualizar los datos obtenidos en distintos tipos de gráficos, tablas y mapas.
- Permite configurar los dashboards para adecuarlos al gusto del usuario.
- Proporciona históricos de datos en forma de gráficos o diagramas.

2.3. Integración ELK y Zeek: Escenario propuesto

Para el despliegue e integración de Zeek IDS y ELK Stack se propone desplegar una red sobre un entorno virtual (VirtualBox) que quedaría de la siguiente forma:

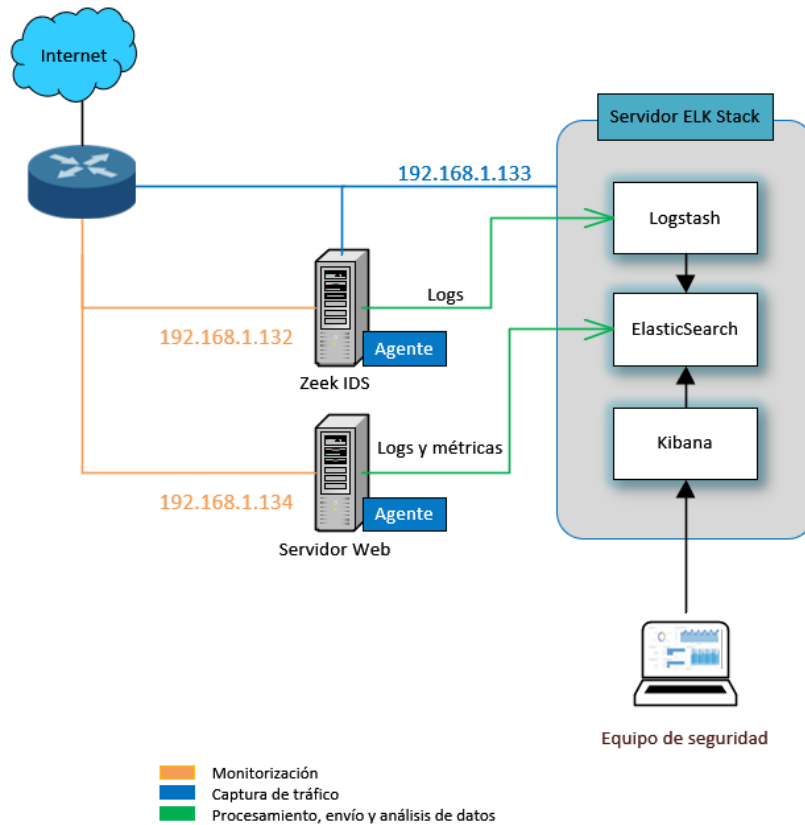


Ilustración 21: Escenario propuesto a implementar.

La arquitectura estará compuesta de los siguientes elementos:

Servidor Zeek IDS

La herramienta Zeek IDS, como se ha comentado anteriormente, actuará como el principal generador de información con el fin de detectar anomalías en la red. Para ello y ya que la solución será desplegada en un entorno virtual, se configurará la tarjeta de red de Zeek en modo promiscuo para permitir capturar los paquetes que circulan en una determinada subred.

Las características de este servidor serán las siguientes:

- Sistema operativo: Ubuntu Server 18.04.4.
- Hardware: 2 x CPU, 4 GB de RAM y disco 200 GB.

- Plataformas instaladas:
 - Sistema IDS: Zeek/Bro IDS.
 - Agente de envío de logs.

Servidor ELK Stack

El sistema SIEM implementado será ELK Stack compuesto por las herramientas Elasticsearch, Logstash y Kibana. Este servidor contendrá este sistema y será la plataforma que realizará las funciones de recopilación, normalización, almacenamiento, visualización y análisis de los datos en función de los registros y logs generados en la red.

Las características de este servidor serán las siguientes:

- Sistema operativo: Ubuntu Server 18.04.4.
- Hardware: 2 x CPU, 4 GB de RAM y disco 200 GB.
- Plataformas instaladas:
 - Sistema SIEM: ELK Stack (Elasticsearch, Logstash y Kibana).

Servidor web

Se trata de un dispositivo a monitorizar mediante la información generada por Zeek y la recolección de varios tipos de datos adicionales obtenidos por distintos agentes instalados localmente. Este servidor web (Tomcat) tendrá desplegada una aplicación web sencilla desarrollada con el framework *Spring* que consta de un inicio de sesión y una fase de autorización a ciertos recursos dependiendo de los permisos que posee el usuario introducido.

Las características de este servidor serán las siguientes:

- Sistema operativo: Ubuntu Server 18.04.4.
- Hardware: 2 x CPU, 2 GB de RAM y disco 100 GB.
- Plataformas instaladas:
 - Aplicación web de prueba desplegada con el servidor de aplicaciones Tomcat y desarrollada con el framework “Spring”.
 - Agentes de envío de logs y métricas.

3. Fase de Implementación

En esta fase, se detallará el despliegue y configuración de las herramientas Zeek IDS y ELK Stack. Además, se llevará a cabo una batería de pruebas utilizando el sistema Kali Linux para generar tráfico malicioso y, de esta forma, comprobar que Zeek y ELK detectan y muestran los resultados correctamente.

3.1. Agentes

Los agentes son los encargados del envío de datos procedentes de las diferentes fuentes que componen el sistema. Por lo tanto, se requiere implementar un agente en cada una de las fuentes que queremos monitorizar para que recopilen y envíen los datos relevantes que serán utilizados por ELK. Concretamente, se encargarán de enviar los datos producidos por Zeek.

Además de los datos recopilados por Zeek, se obtendrá información de otras fuentes adicionales que permitirán complementar la correcta identificación de anomalías de seguridad:

- **Datos de auditoría:** Métricas que permiten identificar cualquier modificación de ficheros, estados de servicios, accesos o inicios de sesión con éxito o fallidos realizados por un determinado usuario, etc. Estos agentes serán instalados en el servidor web detallado en el escenario propuesto.
- **Datos estadísticos:** Métricas que permiten realizar la supervisión de recursos: CPU, memoria, tráfico de red, disco, etc. Este agente recogerá los datos del servidor web y los enviará a Elasticsearch.

Los agentes que recopilarán los datos relevantes anteriores serán los siguientes:

- **Filebeat** se encargará del envío de los logs generados por Zeek hasta Logstash. Para ello, se ha instalado este agente en el servidor Zeek IDS y se ha configurado para enviar los logs a través del puerto de escucha 5044 (Logstash).

- **AuditBeat** y **MetricBeat** se encargarán del envío de logs de auditoría y las métricas generadas por el servidor web. Para ello, se ha instalado ambos agentes en el servidor web y se han configurado para enviar datos a Elasticsearch, a través del puerto de escucha 9200. Estos datos no presentan la necesidad de ser normalizados, por lo que se envían directamente a Elasticsearch para su indexación, es decir, no necesitan ser transformados y modificados, a diferencia, por ejemplo, de la información generada por Zeek.

3.1.1. Filebeat

Filebeat es un agente que monitoriza la existencia de registros en una ubicación concreta. Se encarga de recoger los eventos generados y enviarlos a Logstash o Elasticsearch para su procesamiento o almacenamiento.

Filebeat ha sido instalado en el servidor Zeek IDS con el objetivo de monitorizar los registros generados por este y su posterior envío a Logstash para su procesamiento. Para que realice esa función, en el fichero de configuración de Filebeat `/etc/filebeat/filebeat.yml`, hay que definir los siguientes elementos:

- **Entradas:** Se define el tipo de entrada y las rutas donde se almacenan los registros que se van a enviar a Logstash. En este caso, se indicarán las rutas definidas para los registros generados por Zeek:
 - **intel.log:** Datos de inteligencia obtenidos del Intelligence Framework de Zeek.
Se encuentran en `/usr/local/zeek/logs/current/intel.log`.
 - **notice.log:** Anomalías detectadas.
Se encuentran en `/usr/local/zeek/logs/current/notice.log`.
 - **conn.log:** Datos de conexión TCP/UDP/ICMP.
Se encuentran en `/usr/local/zeek/logs/current/conn.log`.
- **Salidas:** En la salida de Filebeat se especifica la dirección IP y puerto de escucha de Logstash, 5044.

Los elementos anteriores son reflejados en el fichero de configuración de Filebeat (`filebeat.yml`) que se encuentra en el Anexo 7.2.

3.1.2. MetricBeat

Metricbeat se encargará de recoger distintas métricas por medio de dos componentes: los módulos (*modules*) y los juegos de métricas (*metricsets*).

- **Módulos:** Especifican el servicio sobre el cual se va a recolectar la métrica, cómo conectarse a ese servicio y cada cuanto tiempo realizar la consulta. Entre estos módulos destacan: MongoDB, Apache, MySQL, *System*, etc.

En este caso, utilizaremos el módulo *System* para recopilar las métricas relacionadas con el uso de CPU, memoria, disco o red. Este módulo es el establecido por defecto si no se indica ninguno otro.

- **Métricas:** Como cada servicio puede devolver muchos valores, Metricbeat los agrupa en un juego de métricas en formato normalizado antes de enviarlo.

El fichero de configuración Metricbeat (*metricbeat.yml*) se encuentra detallado en el Anexo 7.2.

Seguidamente, se comprueba si la sintaxis del fichero de configuración es correcta:

```
sudo metricbeat -e -c /etc/metricbeat/metricbeat.yml
```

Este agente se utilizará para recopilar información relevante del servidor web y enviarla a Elasticsearch. Su envío se realiza de forma directa, sin pasar previamente por Logstash, ya que estos datos no requieren ser normalizados.

Existe la posibilidad de configurar Kibana para que utilice los dashboards que vienen por defecto. De esta forma, se podrán visualizar los datos obtenidos por los *beat s* en Kibana sin necesidad de crear manualmente las visualizaciones y dashboards:

```
metricbeat setup --dashboards
```

Finalmente, se ejecuta el servicio y se configura para que se inicie cuando arranque el servidor:

```
sudo systemctl enable metricbeat.service
sudo systemctl start metricbeat.service
```

3.1.3. AuditBeat

Auditbeat se encargará de llevar el registro y control de eventos de Kernel de Linux, así como las modificaciones de ficheros y actividades y procesos de usuarios. Es capaz de retener los mensajes que no ha enviado en caso de no haber conexión con Elasticsearch. Se requiere de la siguiente configuración:

- **Módulos:** Especifican el servicio sobre el cual se va a recolectar la métrica, cómo conectarse a ese servicio y cada cuanto tiempo realizar la consulta. Entre estos módulos destacan: Auditd, integridad de ficheros y sistema.
- **Salidas:** Se establece la dirección IP del servidor ELK Stack donde se encuentra instalado Elasticsearch y el puerto de escucha 9200.

El fichero de configuración Auditbeat (*auditbeat.yml*) se encuentra detallado en el Anexo 7.2.

A continuación, se comprueba que la sintaxis del fichero de configuración es correcta:

```
sudo auditbeat -e -c /etc/auditbeat/auditbeat.yml
```

Este agente se utilizará para recopilar información relevante del servidor web y para su posterior envío a Elasticsearch. Su envío se realiza de forma directa, sin pasar previamente por Logstash, ya que estos datos no requieren ser normalizados.

Finalmente, se ejecuta el servicio y se configura para que inicie cuando arranque el servidor:

```
sudo systemctl enable auditbeat.service  
sudo systemctl start auditbeat.service
```

3.2. Zeek IDS

A continuación, se abordará la instalación (Anexo 7.1.1) y configuración de la herramienta junto con una explicación detallada del registro de logs en el que se tiene que prestar especial énfasis para analizar las anomalías detectadas en la red. Por último, se analizará el uso de scripts con el lenguaje de scripting de Zeek. Estos scripts permitirán detectar el tráfico sospechoso generado a partir de un ataque o intento del mismo.

3.2.1. Configuración

Zeek almacena las conexiones, sesiones e incluso datos de la capa de aplicación en ficheros de logs para su posterior procesamiento en un sistema de recopilación de datos.

Zeek, por defecto, funciona en modo *standalone*, es decir, el análisis del tráfico cursado será realizado por un único nodo. Zeek no soporta la funcionalidad de multihebra, por lo que una vez que la capacidad máxima del núcleo del procesador se alcance, la única opción viable sería distribuir las cargas entre otros muchos cores, o incluso a otros ordenadores físicos. Por este motivo, existe el modo cluster que realiza un balanceo de carga y distribuye el tráfico entre varios nodos. En este proyecto, debido a que no se pretende analizar altas tasas de tráfico y volumen, la configuración utilizada será en modo *standalone*.

Los ficheros básicos que se configurarán se encuentran ubicados en `/usr/local/zeek/etc`:

- **node.cfg**: Configuración de la interfaz a monitorizar.

```
[zeek]
type=standalone
host=localhost
interface=enp0s3
```

- **networks.cfg**: Configuración de las redes que Zeek monitorizará y considerará locales al entorno a monitorizar.

```
10.0.0.0/8 Private IP space
172.16.0.0/12 Private IP space
192.168.0.0/16 Private IP space
```

- **zeekctl.cfg**: Configuración global de Zeek. Por ejemplo, se puede establecer el tiempo de rotación que tendrán los logs almacenados.

```
# Opciones de email
MailTo = jfrodriguezvilchez@uoc.edu
MailConnectionSummary = 1
MinDiskSpace = 5
MailHostUpDown = 1
# Opciones de logging
LogRotationInterval = 3600
```

```

LogExpireInterval = 0
StatsLogEnable = 1
StatsLogExpireInterval = 0
# Otras opciones
StatusCmdShowAll = 0
CrashExpireInterval = 0
SitePolicyScripts = local.zeek
LogDir = /usr/local/zeek/logs
SpoolDir = /usr/local/zeek/spool
CfgDir = /usr/local/zeek/etc

```

3.2.2. Logs

Una vez que Zeek ha sido instalado, configurado en el entorno y está monitorizando tráfico de red, comenzará a producir logs en formato ASCII, organizados en columnas y separados por tabuladores. Los logs generados por Zeek son numerosos. A continuación, se muestran los más relevantes:

Archivo log	Descripción
conn.log	Conexiones TCP/UDP/ICMP
dns.log	Actividad DNS
http.log	Peticiones y respuestas HTTP
ssh.log	Conexiones SSH
files.log	Análisis de archivos
intel.log	Coincidencias de datos de Inteligencia
notice.log	Notificaciones de detección

Zeek almacena los logs en `/usr/local/zeek/logs/current`. De todos los archivos de log anteriores, para la identificación de anomalías que es el tema que se está abordando, cabrían destacar los siguientes:

- **intel.log**: El registro `intel.log` almacena información relevante obtenida con el Framework de Inteligencia (*Intelligence Framework*). El objetivo de este Framework radica en consumir los datos procedentes de diferentes fuentes con información relevante y actualizada sobre amenazas potenciales.

El elemento principal de los datos de inteligencia es el indicador que puede ser una dirección IP, dirección de correo electrónico, hash de un fichero, una subred, dominio, etc.

Normalmente, los datos de inteligencia pueden ser descargados libremente en la mayoría de los casos. Para el proyecto, se obtendrán datos de inteligencia CIF (*Collective Intelligence Framework*), el cual, ofrece información obtenida desde diferentes datasets permitiendo consumirlos para poder identificar y detectar actividad sospechosa.

Además de los campos del registro *intel.log* (reflejados en el Anexo 7.4), también aparece información sobre el *cif*, en cuestión, que ha hecho *match* con el tráfico registrado: tag del *cif*, fuente del *cif*, descripción, cuándo se detectó como malicioso por primera y por última vez, etc.

A continuación, se puede observar un ejemplo del registro *intel.log*. En este registro, se representa la detección de tráfico malicioso en la red que corresponde con los datos de inteligencia registrados en Zeek. En este caso, desde el servidor 192.168.1.128 (host anfitrión), se está accediendo a la dirección IP 208.93.152.4 que está registrada como maliciosa:

```
root@zeek:~/ids/usr/local/zeek/logs/current# cat intel.log
#separator \x00
#set_separator ,
#empty_field (empty)
#unset_field -
#path intel
#open 2020-04-22-15-29-23
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p seen.indicator_type seen.indicator_type seen.where seen.node matched sources fuid file_mime_type
be file_desc cif.tags cif.confidence cif.source cif.description cif.firstseen
n cif.lastseen
#types time string addr port addr port string enum enum string set[enum] s
et[string] string string string string double string string string string
1587569363.026086 CjQ2nv2N2lXKGWfxa9 192.168.1.128 54138 208.93.152.4 80 208.
93.152.4 Intel::ADDR Conn::IN_RESP zeek Intel::ADDR csirtg.io - -
scanner 9.0 - iptable drop logs - -
1587569363.026096 Cxqkat3NjcyfTcnufj 192.168.1.128 54139 208.93.152.4 80 208.
93.152.4 Intel::ADDR Conn::IN_RESP zeek Intel::ADDR csirtg.io - -
scanner 9.0 - iptable drop logs - -
1587569363.026927 CjQ2nv2N2lXKGWfxa9 192.168.1.128 54138 208.93.152.4 80 208.
93.152.4 Intel::ADDR HTTP::IN_HOST_HEADER zeek Intel::ADDR csirtg.io - -
scanner 9.0 - iptable drop logs - -
1587569369.837794 CLnJAS1dkf3EFzr1h 192.168.1.128 54141 208.93.152.4 80 208.
93.152.4 Intel::ADDR Conn::IN_RESP zeek Intel::ADDR csirtg.io - -
scanner 9.0 - iptable drop logs - -
1587569369.837817 CGoJRS37XOFFJV3MUG 192.168.1.128 54140 208.93.152.4 80 208.
93.152.4 Intel::ADDR Conn::IN_RESP zeek Intel::ADDR csirtg.io - -
scanner 9.0 - iptable drop logs - -
1587569369.838322 CGoJRS37XOFFJV3MUG 192.168.1.128 54140 208.93.152.4 80 208.
93.152.4 Intel::ADDR HTTP::IN_HOST_HEADER zeek Intel::ADDR csirtg.io - -
scanner 9.0 - iptable drop logs - -
```

Ilustración 22: Ejemplo registro intel.log.

- **notice.log:** Este registro recoge información sobre distintos eventos de seguridad, tales como: escaneo de puertos, inyecciones SQL, ataques de fuerza bruta, uso de certificados SSL no válidos o caducados, etc.

En la siguiente imagen, se refleja un ejemplo del registro *notice.log* (los campos se encuentran descritos en el Anexo 7.4). En este registro, aparecen una serie de anomalías detectadas:

- **SSL::Invalid_Server_Cert:** Tres anomalías detectadas correspondientes a certificados caducados o no válidos detectados.

- 192.168.1.135 scanned at least 15 unique ports of host 192.168.1.134 in 0m0s: Indica que se ha realizado un escaneo de puertos al servidor web.
- 192.168.1.135 appears to be guessing SSH passwords (seen in 3 connections) Sampled Servers: 192.168.1.132: Se ha detectado un ataque de fuerza bruta contra el servidor Zeek IDS (192.168.1.132).

```
#path notice
#open 2020-04-22-07-33-26
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p fuid file
#mime_type file_desc proto note msg sub src dst p n peer
#descr actions suppress_for remote_location.country_code remote_location.region remote_locat
ion.city remote_location.latitude remote_location.longitude
#types time string addr port addr port string string string enum enum stri
ng string addr addr port count string set[enum] interval string stri
ng string double double
1587540806.347801 CEFFon2FdRzrSF2Vmd 192.168.1.128 50975 51.124.78.146 443 FDPa
Be10kYuuApD7bf - - tcp SSL::Invalid_Server_Cert SSL certificate validation f
ailed with (unable to get local issuer certificate) CN=settings-win.data.microsoft.com,OU=MSF,OU
Microsoft,L=Redmond,ST=WA,C=US 192.168.1.128 51.124.78.146 443 - - Notice::ACTI
ON_LOG 3600.000000
1587541342.351139 CV1v1N2oniyKPDg73i 192.168.1.128 51012 192.168.1.134 8443 FaRW
h18FK11EBKkw - - tcp SSL::Invalid_Server_Cert SSL certificate validation f
ailed with (self signed certificate) CN=localhost,OU=insectores,0=insectores,L=madrid,ST=madrid,C
=ma 192.168.1.128 192.168.1.134 8443 - - Notice::ACTION_LOG 3600.000000-
- -
1587541474.855854 - - - - - - - - Capt
ureLoss::Too_Much_Loss The capture loss script detected an estimated loss rate above 25.936% - -
- - Notice::ACTION_LOG 3600.000000 - - - -
1587541557.373310 - - - - - - - - Scan
::Port_Scan 192.168.1.135 scanned at least 15 unique ports of host 192.168.1.134 in 0m0s loca
l 192.168.1.135 192.168.1.134 - - - - Notice::ACTION_LOG 3600.000000-
- -
1587541828.549101 - - - - - - - - SSH:
Password_Guessing 192.168.1.135 appears to be guessing SSH passwords (seen in 3 connections).S
ampled servers: 192.168.1.132, 192.168.1.132, 192.168.1.132 192.168.1.135 - - - -
Notice::ACTION_LOG 3600.000000 - - - -
1587542154.343108 Ckclnp1282uCvoNCIa 192.168.1.128 51103 192.168.1.134 8444 FRKR
rs782YPruDvhe - - tcp SSL::Invalid_Server_Cert SSL certificate validation f
ailed with (self signed certificate) CN=localhost,OU=insectores,0=insectores,L=madrid,ST=madrid,C
=ma 192.168.1.128 192.168.1.134 8444 - - Notice::ACTION_LOG 3600.000000-
- -
#close 2020-04-22-08-00-00
```

Ilustración 23: Ejemplo registro notice.log.

- **conn.log:** Este registro almacena todas las conexiones establecidas a través de los protocolos de red TCP/UDP e ICMP.

La siguiente figura corresponde con un ejemplo del registro *conn.log* (los campos del fichero se encuentran descritos en el Anexo 7.4). En este registro, se representan una serie de conexiones UDP y TCP realizadas por el dispositivo con IP 192.168.128 (host anfitrión) y 192.168.1.133 (servidor ELK).

```
root@zeekids:/usr/local/zeek/logs/current# cat conn.log | more
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path conn
#open 2020-04-22-14-00-10
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p proto serv
ice duration orig_bytes resp_bytes conn_state local_orig local_resp n
issued_bytes history orig_pkts resp_ip_bytes resp_pkts resp_ip_bytes tunnel_paren
ts
#types time string addr port addr port enum string interval count count
t string bool bool count string count count count set[string]
1587564000.544278 CGqE1E4DJRrhDzPMXe 192.168.1.132 34700 46.6.113.34 53 udpc
ns 0.071001 0 221 SHR T F 0 Cd 0 0 2 2
77
- -
1587563939.486753 C1mEah4P9y5egGCBHj 192.168.1.128 49457 216.58.211.238 443 udp-
15.060009 2438 2445 SF T F 0 Dd 5 2578 4 2557
- -
1587564014.915961 CNdizA2mL9vGvsIL86 192.168.1.128 63019 212.231.6.7 53 udpc
ns 0.006727 47 63 SF T F 0 Dd 1 75 1 9
1
- -
1587564021.394910 CdRFnu3MR4XnxSIB45 192.168.1.133 15908 169.254.169.254 80 tcp-
8.026311 0 0 S0 T F 0 S 3 180 0 0 -
1587564021.395160 Cxf0UhbZ8kuFyWkjk 192.168.1.133 51910 169.254.169.254 80 tcp-
8.026065 0 0 S0 T F 0 S 3 180 0 0 -
1587564021.395228 CIHW1a38PhZCFK1BLb 192.168.1.133 51912 169.254.169.254 80 tcp-
8.026017 0 0 S0 T F 0 S 3 180 0 0 -
1587564028.582905 Cop7h3j8rnCBNHFM3 192.168.1.133 51912 169.254.169.254 80 tcp-
- - - - S0 T F 0 S 1 60 0 0 -
```

Ilustración 24: Ejemplo registro conn.log.

3.2.2.1. Cargando la Inteligencia (intel.log)

Para que Zeek pueda detectar amenazas potenciales, es necesario cargar los datos de inteligencia. Zeek consumirá los datos procedentes de diferentes fuentes con información relevante y actualizada sobre amenazas potenciales.

Para el proyecto, se obtendrán los datos de inteligencia CIF (*Collective Intelligence Framework*). CIF ofrece información obtenida desde diferentes datasets permitiendo consumirlos para poder identificar y detectar actividad sospechosa.

Se utilizará CSIRT Gadgets para obtener los datos de inteligencia en un formato que Zeek pueda entender y consumir, generando logs o alertas siempre que detecte algo que aparece registrado en estos datos. Para generar estos datos, hay que realizar los siguientes pasos:

1. Es necesario registrarse en CSIRTG (<https://csirtg.io/>) y obtener una cuenta. De esta forma, podremos generar un token que utilizaremos, posteriormente, para obtener los datos de inteligencia.

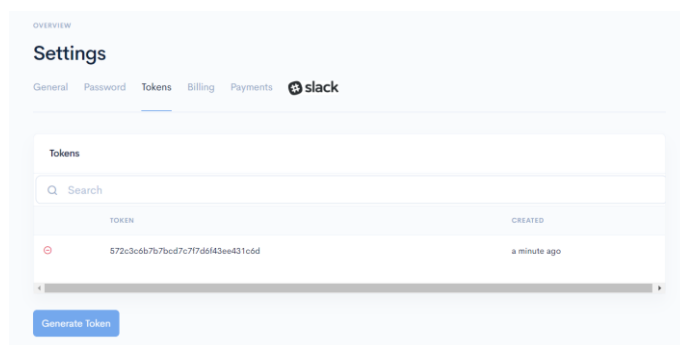


Ilustración 25: Token CSIRTG generado.

2. Se requiere Python 3.5 instalado junto con pip:

```
sudo wget https://www.python.org/ftp/python/3.5.6/Python-3.5.6.tgz
sudo tar xzf Python-3.5.6.tgz
cd Python-3.5.6
sudo ./configure --enable-optimizations
sudo make
sudo make test
sudo make install
```



```
sudo ldconfig
```

```
sudo apt install python-pip
```

3. Se instala el Gadget CSIRTG:

```
sudo pip install csirtgsdk  
sudo pip install csirtg-smrt
```

4. Se procede a configurar el CSIRTGsdk. Se crea el fichero `.csirtg.yml` y se especifica el usuario y el token generados anteriormente:

```
user: juanfranrv  
token: 572c3c6b7b7bcd7c7f7d6f43ee431c6d
```

5. Tras esto, podremos lanzar `csirtg -feeds` para comprobar que todos los datos de inteligencia se encuentran disponibles.

Adicionalmente, con el siguiente comando se obtienen feeds de distintos tipos:

- Feed de direcciones IP bloqueadas por un Firewall: `port-scanners`.
- Feed de URLs comprometidas: `uce-urls`.
- Feed de direcciones IP o cuentas de usuario que han sido comprometidas: `uce-ip`.

[FEED] será de tipo url, ip o puerto y [RUTA FICHERO] el fichero ubicado en `/opt/feeds` que será consultado por Zeek.

```
sudo csirtg-smrt -r csirtg.yml -f [FEED] --format zeek > [RUTA FICHERO]
```

6. Finalmente, tal y como se detalla en el siguiente apartado “Scripting”, se añaden una serie de políticas necesarias al fichero `local.zeek` para el funcionamiento del Framework de Inteligencia. Este fichero define los scripts que se ejecutarán cuando arranque Zeek. Posteriormente, se procede a desplegar la nueva configuración mediante el comando `zeekctl deploy`.

Cuando Zeek detecte algún patrón que coincide con los datos de inteligencia registrados en los ficheros ubicados en `/opt/feeds`, generará el fichero *intel.log* que registrará la información asociada a este tráfico malicioso.

3.2.3. Scripting

Como se mencionó anteriormente, Zeek consta de dos componentes principales: el motor de eventos y el intérprete de scripts. En esta sección, nos vamos a centrar en este último, el intérprete de scripts, que permite definir acciones a llevar a cabo cuando se detecta cierta actividad de acuerdo con las políticas que poseen los scripts.

Los scripts principales que posee Zeek para la detección de incidencias de seguridad serían los siguientes:

- *detect-bruteforcing.zeek*: Proporciona información sobre posibles ataques de fuerza bruta al servicio SSH. Este script analiza si se están realizando numerosos intentos fallidos de inicios de sesión con usuarios y contraseñas procedentes desde una misma dirección. Para ello, utiliza un umbral para el número de intentos y un intervalo de tiempo a monitorizar.

Cuando un usuario inicia sesión satisfactoriamente, el servidor SSH responde con un paquete de tamaño 5k. Por lo tanto, Zeek marcará el parámetro "auth_success" con una "T" en el fichero *SSH.log*. Este script monitorizará los eventos SSH que tengan el atributo anterior con valor "F", lo que significa que se ha realizado un intento de fuerza bruta.

- *scan.zeek*: Detecta escaneos de puertos realizados mediante diversas herramientas como *nmap*.
- *detect-MHR.zeek*: Permite comparar los hashes de ficheros con hashes de malware conocidos con el objetivo de identificar una posible amenaza. Este script es proporcionado por la organización llamada Team Cymru. La base de datos de hashes puede ser consultada mediante peticiones DNS (UDP 53), Whois (TCP 43), HTTP (TCP 80) o HTTPS (TCP 443).
- *detect-sqli.zeek*: Detecta ataques de tipo SQL injection. Para comprobar el funcionamiento de este script y que Zeek detecta estos ataques correctamente, realizaremos ataques de tipo SQLi utilizando la herramienta *nmap*.

Estos son algunos scripts de ejemplo. Sin embargo, Zeek cuenta con multitud de ellos para detectar diversas amenazas o para definir acciones que se llevarán a cabo al detectar cierto tráfico sospechoso.

En el fichero `local.zeek` ubicado en `/usr/local/zeek/share/zeek/site`, se definen los scripts que Zeek ejecuta a la hora de detectar anomalías en la red. A la configuración por defecto, se le ha añadido una serie de políticas necesarias para el funcionamiento del Framework de Inteligencia. El contenido del fichero se encuentra detallado en el Anexo 7.2.

3.3. ELK Stack

A continuación, se abordará la instalación (Anexo 7.1.2) y configuración de ELK englobando todas las partes que lo componen: Elasticsearch, Logstash y Kibana. Seguidamente, se detallará el flujo seguido desde que se reciben los datos de Zeek hasta que se muestran en Kibana. Por último, se realizarán una serie de pruebas de monitorización y visualización de datos para analizar el funcionamiento completo del sistema implementado.

3.3.1. Recolección de datos

Logstash será el motor de recopilación de los logs enviados por el agente Filebeat. Como hemos visto, el objetivo de Logstash consiste en recopilar los datos generados por Zeek y normalizarlos mediante la modificación del formato.

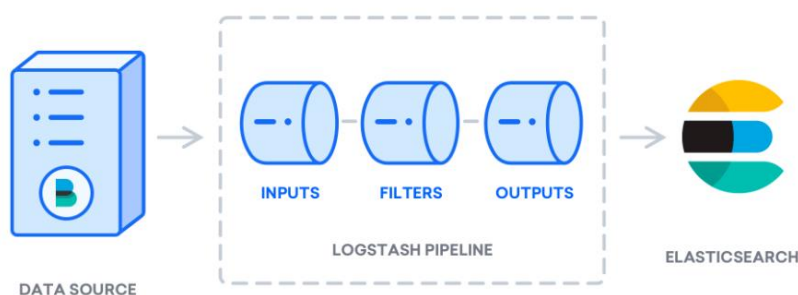


Ilustración 26: Arquitectura Logstash.

Observando la figura anterior, los datos de logs llegarían a Logstash que procesa la información en base a la creación de una pipeline formada por: Inputs (Entradas), Filters (Filtros) y Outputs (Salidas).

Cada uno de estos elementos que forman la pipeline tienen que ser definidos en un archivo de configuración dentro del directorio de Logstash.

Inputs (Entradas)

Los complementos de entrada consumen datos procedentes de una fuente. En este caso, la información es enviada desde el agente Filebeat, a través del puerto 5044, que es el utilizado para establecer la comunicación entre Filebeat y Logstash.

Para crear esta entrada, nos dirigimos al directorio de Logstash y creamos el certificado SSL que nos permitirá securizar la conexión entre Filebeat y Logstash:

```
sudo cd /etc/logstash
sudo mkdir ssl
sudo openssl req -subj '/CN=192.168.1.133' -x509 -days 3650 -batch -nodes -
newkey rsa:2048 -keyout ssl/logstash-forwarder.key -out ssl/logstash-
forwarder.crt
```

Posteriormente, en el directorio `/etc/logstash/conf.d/`, se crea el fichero que establecerá la entrada de Filebeat `sudo vim filebeat-input.conf` y se añade el certificado y la clave SSL:

```
input {
  beats {
    port => 5044
    type => syslog
    ssl => true
    ssl_certificate => "/etc/logstash/ssl/logstash-forwarder.crt"
    ssl_key => "/etc/logstash/ssl/logstash-forwarder.key"
  }
}
```

El certificado SSL generado, se copiará en el servidor Zeek, en la ubicación donde se encuentra instalado Filebeat. De esta forma, se especifica una entrada securizada de beats que escuchará en el puerto 5044.

Filters (Filtros)

Los filtros procesan los datos con el objetivo de transformarlos a un formato común y normalizado para que pueda ser manejado fácilmente por el resto de componentes de ELK.

Los siguientes pasos se han llevado a cabo para la normalización de los datos de logs del fichero de Zeek, *conn.log*. Para el resto de ficheros de log, el procedimiento sería similar:

1. Se eliminan los comentarios ubicados en la parte superior de los ficheros de logs. Se hace uso de expresiones regulares para eliminar todas las líneas que empiezan con el carácter almohadilla '#’.
2. Se eliminan los tabuladores mediante el filtro CSV. Este filtro permite el manejo de tabuladores dentro de un fichero.
3. Se modifica la marca de tiempo (*timestamp*) que utiliza Zeek de tipo UNIX a tipo fecha. Para ello, se aplica un filtro que posee Logstash.
4. Se modifican los nombres de los campos y el tipo de los datos presentes.
 - Se utiliza el filtro *Mutate* para modificar los campos que presentan puntos, ya que no es conveniente que Logstash procese estos caracteres al fallar en ciertas ocasiones. Por lo tanto, el cambio de los nombres de los campos para los logs quedaría de la siguiente forma (para el fichero de log *conn.log*):
 - `id.orig_h => id_orig_host`
 - `id.orig_p => id_orig_port`
 - `id.resp_h => id_resp_host`
 - `id.resp_p => id_resp_port`
 - Se modifica el tipo de dato de cada uno de los campos a *float* e *integer* que son soportados por Logstash. Por ejemplo:
 - `id.orig_h => integer`
 - `id.orig_p => integer`
 - `duration => float`
 - `resp_bytes => integer`

Los filtros creados para cada uno de los ficheros de logs que se envían desde Zeek a Logstash, a través de Filebeat, para la normalización del formato de los datos, se encuentran detallados en el Anexo 7.2.

Este fichero `syslog-filter.conf` se encontrará ubicado en `/etc/logstash/conf.d`.

Outputs (Salidas)

En esta etapa, una vez que Logstash realiza la transformación de los datos mediante el uso de los filtros anteriores, estos son enviados a Elasticsearch para su almacenamiento e indexación.

Para llevar a cabo lo indicado anteriormente, se crea el fichero de salida de Elasticsearch, `output-elasticsearch.conf`. De esta forma, se configura Logstash para almacenar los datos de Beats en Elasticsearch creando un índice con el nombre del Beat en cuestión. Además, se indica la dirección y puerto por el que escucha Elasticsearch (*localhost:9200*):

```
output {
  elasticsearch {
    hosts => ["localhost:9200"]
    manage_template => false

    index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
  }
}
```

En este bloque, hay que definir un nombre de índice para visualizar los datos en Kibana. El nombre del índice tendrá el formato `filebeat-yyyy.MM.dd`, ya que los datos recibidos por Logstash son enviados por Filebeat.

Finalmente, se puede hacer uso del siguiente comando para comprobar que no existen errores de sintaxis en los ficheros de configuración creados para el funcionamiento de Logstash:

```
sudo -u logstash /usr/share/logstash/bin/logstash --path.settings
/etc/logstash -t
```

Si no hay errores de sintaxis, el resultado se verá de la siguiente forma:

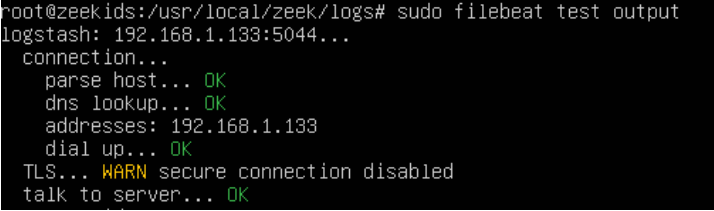
```
Validation Result: OK. Exiting Logstash.
```

Verificación comunicación Filebeat – Logstash

Existe la posibilidad de verificar la configuración de la salida de Filebeat y la comprobación de la conexión hacia Logstash a través del uso de este comando:

```
sudo filebeat test output
```

Si la conexión es correcta, el resultado sería el siguiente:



```
root@zeekids:/usr/local/zeek/logs# sudo filebeat test output
logstash: 192.168.1.133:5044...
connection...
  parse host... OK
  dns lookup... OK
  addresses: 192.168.1.133
  dial up... OK
  TLS... WARN secure connection disabled
  talk to server... OK
```

Ilustración 27: Resultado Test conexión Filebeat - Logstash.

Como se puede observar en la imagen anterior, la comunicación entre Filebeat y Logstash se ha realizado con éxito. Sin embargo, no posee una autenticación basada en TLS/SSL que permita una conexión segura entre ambas entidades.

Para solucionar este problema, se ha optado por proteger la comunicación a través del protocolo TLS/SSL. Para ello, se ha creado un certificado autofirmado, en el cual, ELK (Logstash) actúa como servidor y Filebeat como cliente dentro del entorno. Los ficheros generados tendrán extensión .key y .crt, y estarán almacenados en el servidor ELK. Únicamente, el fichero con extensión .crt será distribuido y almacenado en el cliente Filebeat (servidor Zeek IDS). El procedimiento para realizarlo se encuentra detallado en el Anexo 7.1 donde se detallan los inputs de Logstash.

3.3.2. Almacenamiento e indexación

Elasticsearch proporciona el almacenamiento e indexación de la información enviada por Logstash. La información recolectada por Filebeats y normalizada por Logstash, posteriormente, se integra en Elasticsearch.

Para configurar Elasticsearch, solamente, ha sido necesario definir la dirección IP y el puerto que van a utilizar el resto de componentes para comunicarse con él. La dirección IP definida tuvo que ser modificada de *localhost* a 192.168.1.133 para ser alcanzada desde el cliente Filebeat instalado en el servidor Zeek. Esta comunicación ha sido necesario para crear el índice que utilizará Elasticsearch para almacenar la información, ya que Filebeat solo se comunica con Logstash.

Como se ha indicado, Filebeat no envía ninguna información a Elasticsearch directamente. Sin embargo, es necesario que Filebeat cargue la plantilla de índices para procesar los campos que contienen los ficheros de logs generados por Zeek. Por lo tanto, hay que cargar la plantilla de índices manualmente, ya que Filebeat no se comunica con Elasticsearch directamente. El comando utilizado ha sido el siguiente:

```
filebeat setup --template -E output.logstash.enabled=false -E 'output.  
elasticsearch.hosts=["192.168.1.133:9200"]'
```

Siendo la dirección IP 192.168.1.133 la correspondiente al servidor ELK Stack donde está desplegado Elasticsearch y el 9200, el puerto de escucha.

El resto de beats (metricbeat y auditbeat) enviarán los datos a Elasticsearch directamente. Esto facilita la creación automática de la plantilla de índices:

- **Filebeat:** filebeat-6.8.8-yyyy.mm.dd
- **Auditbeat:** auditbeat-7.6.2-yyyy.mm.dd
- **Metricbeat:** metricbeat-7.6.2-yyyy.mm.dd

Teniendo Elasticsearch configurado y el índice creado, Logstash almacenará la información normalizada. Para validar que Elasticsearch está almacenando esta información, se puede ejecutar la siguiente petición GET:

```
curl -X GET http://localhost:9200/filebeat*/_search?pretty
```

Obteniendo como salida un objeto JSON que indica el número total de datos que han sido almacenados en Elasticsearch.

Otra opción sería listar los índices creados a partir de los eventos enviados por Filebeat:


```
root@elkstack:/home/juanfranrv# curl -X GET http://localhost:9200/_cat/indices
yellow open filebeat-6.8.8-2020.04.11 92xJrLkBSf63qfLh3LJj3g 3 1 111 0 116.3kb 116.3kb
green open .kibana_task_manager -Ce4hccoRCWEyJdatFZ_5g 1 0 2 0 12.6kb 12.6kb
yellow open filebeat-6.8.8-2020.04.12 0vhZP2oTSDi6r8r3Wx-jMQ 3 1 2090 0 1mb 1mb
green open .kibana_1 aS2Myzxt8JebWXIyvTVzLQ 1 0 7 1 94.3kb 94.3kb
```

Ilustración 28: Índices creados en Elasticsearch.

Como se puede observar, se encuentran creados los índices de Kibana (se crearon automáticamente) necesarios para almacenar información propia del componente. Además, existen otros dos índices llamados *filebeat-6.8.8-yyyy.mm.dd* que corresponden a los datos de Zeek enviados por Filebeat, y normalizados por Logstash: uno creado el día 11/04/2020 y otro el día 12/04/2020.

3.3.3. Exploración y visualización de datos

Una vez que los datos son enviados desde Zeek IDS, a través de Filebeat, a Logstash y son almacenados en Elasticsearch, Kibana puede extraerlos con el fin de visualizarlos, analizar y obtener información relevante sobre la seguridad de la red.

El primer paso consiste en configurar el patrón de índice y crear los paneles de Kibana para acceder a la información de Elasticsearch. Para ello, accedemos a la dirección de Kibana: *http://192.168.1.133:5601*.

Para extraer la información desde Elasticsearch, es necesario:

- Crear un patrón de índice para visualizar los índices generados por los agentes. En la siguiente imagen, se observa la creación de un patrón de índice llamado "filebeat*". El "*" corresponde con el símbolo utilizado en expresiones regulares para abarcar todos los caracteres. Como los índices tienen la forma *filebeat-yyy.MM.dd*, con el patrón de índice definido, se obtendrían todos los índices generados por el agente Filebeat.

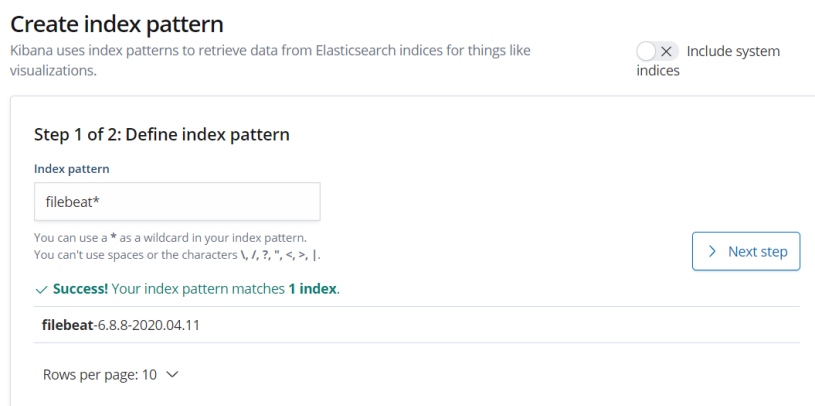


Ilustración 29: Creación de patrón de índice en Kibana.

- Se selecciona la marca de tiempo (timestamp), tal y como se indica en la siguiente figura:

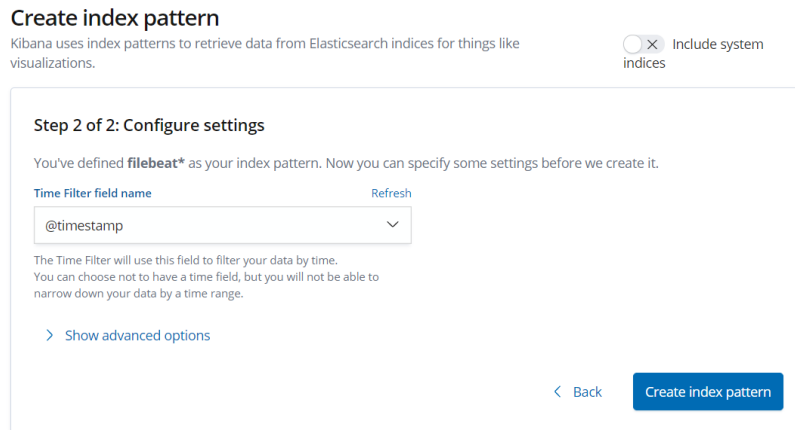


Ilustración 30: Selección de marca de tiempo en Kibana.

Se realizan los mismos pasos anteriores para el resto de beats.

Tras realizar los dos pasos anteriores, ya disponemos de un índice creado. La opción “Discover” de Kibana permite realizar la exploración de los datos agregados. En la siguiente imagen, se puede observar un histograma junto con los campos y valores de los datos recibidos.

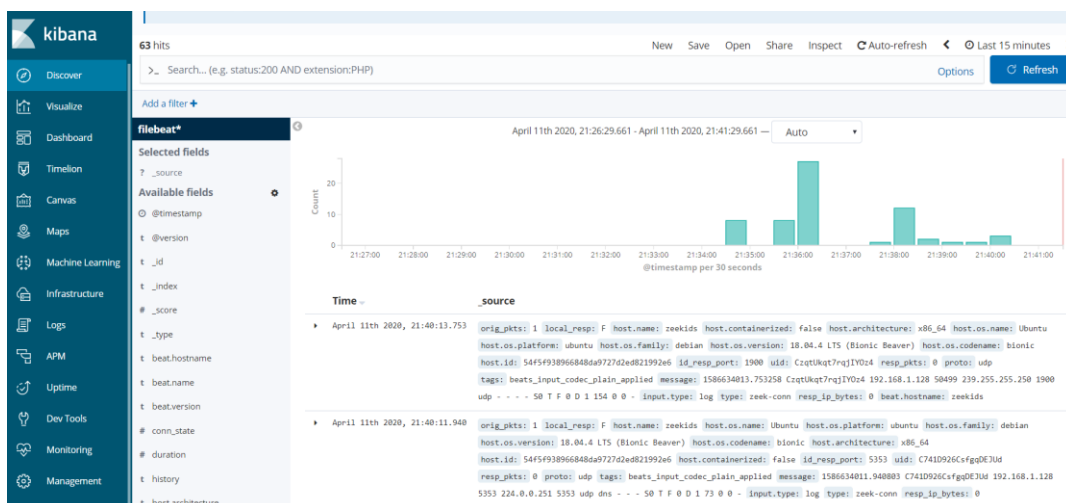


Ilustración 31: Opción "Discover" de Kibana.

Si seleccionamos uno de los documentos recibidos de Zeek, se observan los detalles completos del conjunto de datos en formato JSON o en formato tabla. Se pueden aplicar distintos filtros sobre estos para buscar más fácilmente la información deseada.

A modo de ejemplo, en la siguiente figura, se aplica un filtro para realizar la búsqueda de todas las conexiones realizadas utilizando el protocolo UDP. Estos datos se han obtenido del registro *conn.log* de Zeek.

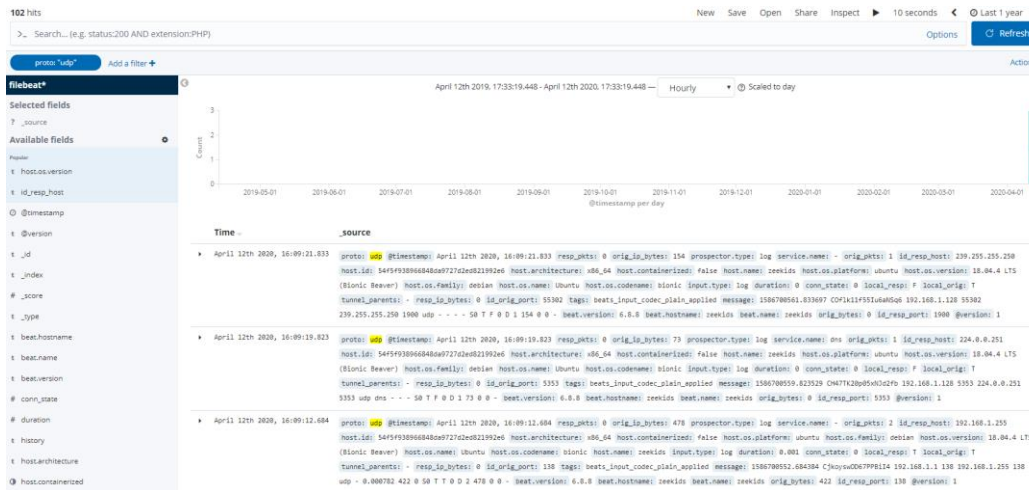


Ilustración 32: Ejemplo de filtro aplicado en Kibana.

Kibana ofrece la posibilidad de visualizar los datos en numerosos tipos de diagramas diferentes. Estos diagramas son completamente personalizables. Por ejemplo, en la ilustración 41, se ha diseñado un diagrama de barras que indica el número de conexiones establecidas utilizando los protocolos TCP, UDP e ICMP dentro de la red.

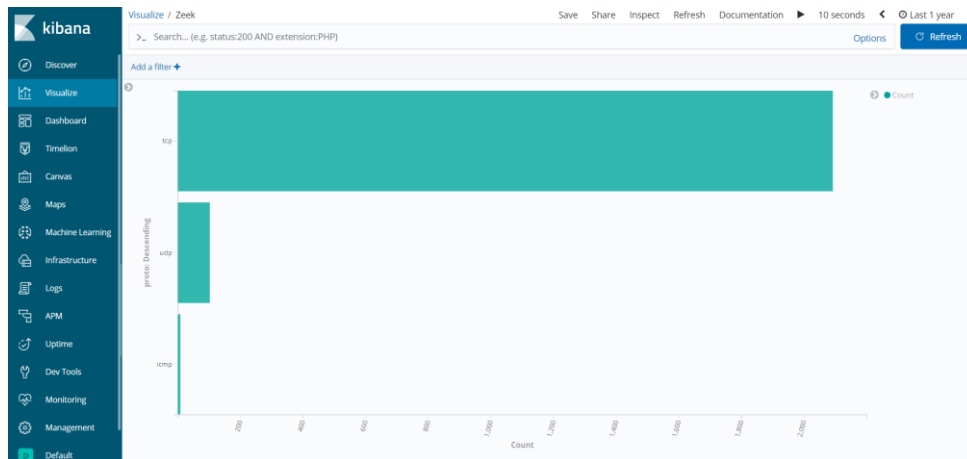


Ilustración 33: Diagrama de barras en Kibana.

Estos diagramas (visualizaciones) se pueden añadir a uno o varios paneles (dashboards). De esta forma, podemos visualizar los eventos que están ocurriendo en nuestra red de forma más rápida y ágil al irse actualizando en tiempo real los diferentes diagramas que forman los dashboards.

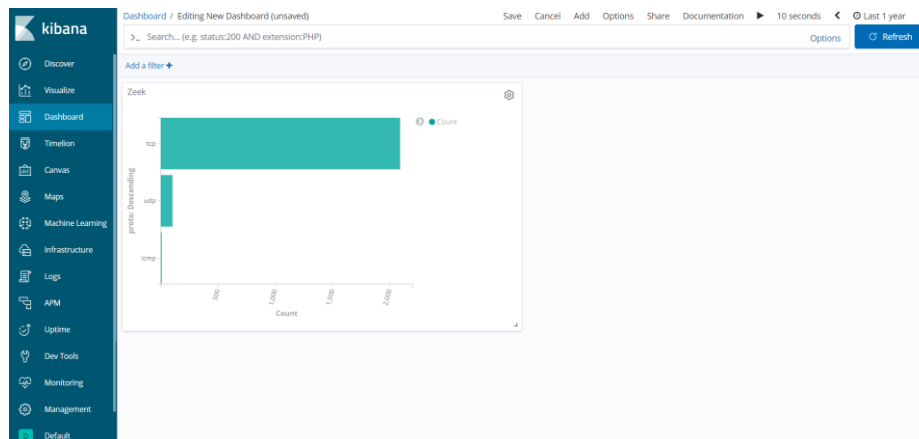


Ilustración 34: Diagrama añadido a un Dashboard en Kibana.

En Kibana se pueden cargar dashboards predefinidos para visualizar los datos obtenidos de diferentes fuentes. Se han cargado los dashboards de Metricbeat y Auditbeat que mostrarán distintas métricas obtenidas del servidor web:

- Los dashboards de Metricbeat ofrecen tablas y diagramas que permiten analizar datos a nivel de sistema o host: CPU, memoria, tráfico entrante y saliente, disco, etc:
- Los dashboards de Auditbeat permiten visualizar diversos datos, tales como inicios de sesión en el sistema, usuarios, sockets, información del sistema y kernel, auditorías o integridad de ficheros.

3.4. Escenario de Pruebas

Para comprobar el correcto funcionamiento del sistema implementado, se ha desplegado un escenario de pruebas con una máquina virtual con el sistema operativo Kali Linux. Esta máquina nos permitirá realizar diversos ataques contra los equipos que se encuentran en la red. De esta forma, se comprobará que Zeek y ELK detectan este tráfico anómalo y se podrá analizar en detalle a través de la interfaz de Kibana.

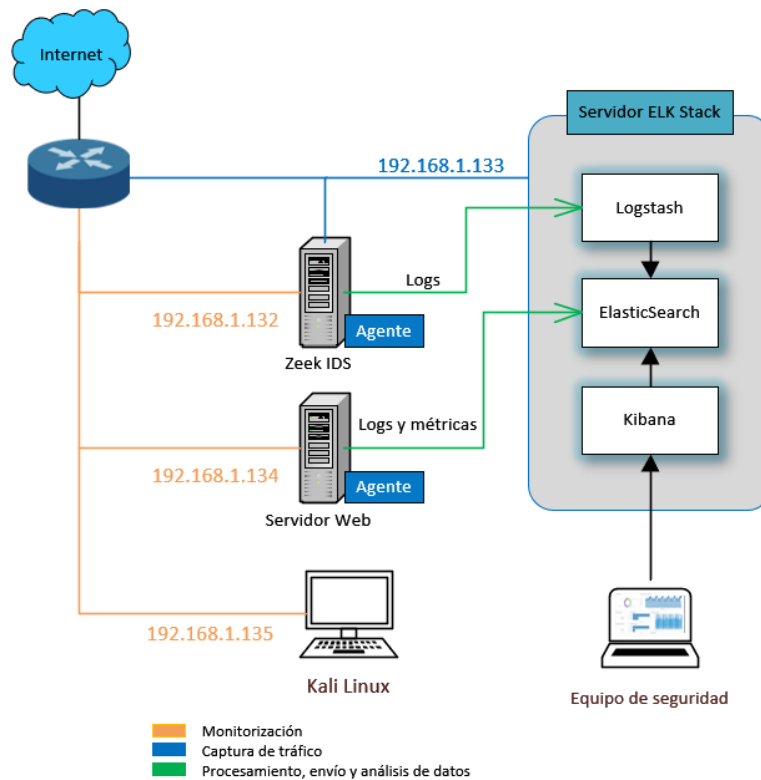


Ilustración 35: Entorno para el escenario de pruebas.

Mediante el uso de Kali Linux, se llevarán a cabo las siguientes pruebas:

- **Ataques de fuerza bruta** para intentar adivinar la contraseña del servicio SSH del servidor de Zeek IDS.
- **Framework de Inteligencia.** Se generará tráfico malicioso desde un PC conectado a la red para comprobar si Zeek asocia dicho tráfico con el CIF utilizado, es decir, si los patrones (URLs, IPs fraudulentas, puertos, etc) reflejados en el CIF coinciden con el tráfico generado desde un navegador web para verificar que se detecta correctamente.
- **Ataques de inyección SQL (SQL Injection)** contra el servidor web con el objetivo de obtener datos relevantes de la base de datos.
- **Enumeración de servicios y escaneo de puertos** contra el servidor web para obtener un listado de puertos abiertos junto con los servicios que están escuchando en estos.

Zeek deberá detectar este tráfico malicioso producido en la red y generar los logs necesarios para el análisis y visualización en Kibana. Adicionalmente, el sistema deberá de recopilar los datos de métricas estadísticas y de auditoría generados por el servidor web.

3.4.1. Detección de tráfico malicioso

3.4.1.1. Fuerza Bruta

El script *detect-bruteforcing.zEEK* permite detectar posibles ataques de fuerza bruta al servicio SSH o FTP. Este script analiza si se están realizando numerosos intentos fallidos de inicios de sesión con usuarios y contraseñas procedentes desde una misma dirección. Para ello, utiliza un umbral para el número de intentos y un intervalo de tiempo a monitorizar.

En este script, cuando un usuario se logea satisfactoriamente, el servidor SSH responde con un paquete de tamaño 5k. Por lo tanto, Zeek marcará el parámetro "auth_success" con una "T" en el fichero *SSH.log*. Este script monitorizará los eventos SSH que tengan el atributo anterior con valor "F" significando que se ha realizado un intento de fuerza bruta.

Para la ejecución satisfactoria de esta prueba, se ha implementado un script adicional que basándose en el script inicial de *detect-bruteforcing.zEEK*, reduzca el límite y tiempo de los intentos para adivinar la contraseña. Además, al producirse este ataque, se escribe información sobre la anomalía producida en el *notice.log*.

El motivo por el que se ha tenido que implementar este script adicional, ha sido debido a que Zeek no detectaba los ataques fuerza bruta realizados con la herramienta *Hydra* de Kali Linux. Tras añadir las modificaciones necesarias en este script, los ataques fuerza bruta, tanto con *Hydra* como con *TCPReplay*, son correctamente detectados.

```
@load protocols/ssh/detect-bruteforcing
redef SSH::password_guesses_limite=3;
redef SSH:guessing_timeout=60 mins;
evento NetControl::init(){
  local debug_plugin = NetControl::create_debug(T);
  NetControl::activate(debug_plugin, 0);
}
hook Notice::policy(n: Notice::Info){
  If (n$note == SSH:Password_Guessing)
```

```
NetControl::drop_address(n$src, 60min);  
Add n$actions[Notice::ACTION_LOG];}
```

A continuación, se muestra la información generada por Zeek cuando detecta un ataque de fuerza bruta contra algún elemento de la red que está monitorizando. Esta información se registra en el fichero *notice.log*:

```
SSH:Password_Guessing 192.168.1.135 appears to be guessing SSH passwords  
(seen in 3 connections). Sampled servers: 192.168.1.132, 192.168.1.132,  
192.168.1.132 192.168.1.135 - - - - Notice::ACTION_LOG 3600.000000 F -  
- - - -
```

La dirección IP 192.168.1.132 corresponde al servidor Zeek y la 192.168.1.135 corresponde a Kali Linux. El comando utilizado ha sido:

```
hydra -t 4 -V -f -l administrator -P passlist.txt ssh://192.168.1.132
```

- t 4: 4 logins al mismo tiempo en paralelo.
- V: Muestra en pantalla los usuarios y contraseña que se están probando.
- f: Detiene la herramienta cuando encuentre un usuario y contraseña correcto.
- l administrator: Usa el usuario “administrador” para intentar logarse.
- P passlist.txt: Diccionario que se utiliza para adivinar la contraseña.
- ssh://192.168.1.132: Servidor Zeek que queremos atacar.

3.4.1.2. Framework de Inteligencia

El Framework de Inteligencia de Zeek consume datos procedentes de diversas fuentes con información relevante y actualizada sobre amenazas potenciales. En este caso, se han obtenido datos de inteligencia CIF (*Collective Intelligence Framework*) que ofrece información obtenida de diferentes bases de datos permitiendo consumirla para identificar actividad sospechosa.

En esta prueba, se hace uso de los feeds disponibles en el proyecto CSIRT Gadgets. Cuando se produce una coincidencia entre algún elemento de estos feeds con el tráfico analizado por Zeek, se genera información en el registro *intel.log*, tal y como se observa a continuación:

```
1#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p seen.indicator seen.indicator_type
seen.where seen.node matched sources fuid file_mime_type file_desc cif.tags cif.confidence
cif.source cif.description cif.firstseen cif.lastseen
```

```
1587563156.217734 CgdU8Z1iV1aRnftYJ2 192.168.1.134 51701 221.156.106.108
221.156.106.108 Intel::ADDR Conn::IN_RESP zeek Intel::ADDR csirtg.io -
- - scanner 9.0 - iptable drop logs - -
```

Esta dirección IP se encuentra registrada en el fichero *port.intel* que corresponde al feed de direcciones IP bloqueadas por un Firewall. Para realizar la prueba, se ha lanzado un ping (ICMP) a esta dirección IP.

Cabe resaltar que esta misma información se encuentra registrada en el fichero *notice.log*.

3.4.1.3. Escaneo de puertos

La herramienta *Nmap* de Kali Linux es utilizada para realizar escaneos de puertos de diversa índole: identificación de servicios, especificando el protocolo y diversas variantes más. En este caso, Zeek detectará esta actividad y generará información sobre ella en el registro *notice.log*.

En el siguiente fragmento obtenido del fichero *notice.log*, se puede comprobar la información generada por Zeek cuando se produce un escaneo de puertos en la red:

```
Scan::Port_Scan 192.168.1.135 scanned at least 15 unique ports of hosts
192.168.1.134 in 0m0s local 192.168.1.135 192.168.1.134 - - -
Notice::ACTION_LOG 3600.000000 - -
```

Observando la salida anterior, se indica que se realizó un escaneo de puertos procedente de Kali Linux 192.168.1.135 hacia el servidor web con dirección IP 192.168.1.134. El comando empleado ha sido:

```
nmap -O -sV 192.168.1.134
```

Con las opciones anteriores, nmap nos permite identificar qué tecnología hay detrás de un puerto abierto o incluso el sistema operativo instalado en el servidor.

3.4.1.4. Inyección SQL

El script *detect-sqli.zeek* permite detectar posibles ataques de inyección SQL. Para realizar esta prueba, desde Kali Linux se ha lanzado un ataque contra el servidor web, a través de la herramienta *sqlmap*, utilizada para la detección y explotación de vulnerabilidades de tipo SQL injection.

En el siguiente fragmento obtenido del fichero de log *notice.log*, se puede comprobar la información generada por Zeek cuando se produce un ataque de tipo SQLi:

```
1588328350.178809 - - - - - HTTP::SQL_Injection_Attacker An SQL
injection attacker was discovered! - 192.168.1.135 - - - - Notice::ACTION_LOG
3600.000000 - - - - -
1588328350.178809 - - - - - HTTP::SQL_Injection_Attacker An SQL
injection attacker was discovered! - 192.168.1.134 - - - - Notice::ACTION_LOG
3600.000000 - - - - -
```

Se han llevado a cabo dos pruebas SQLi contra dos servidores web desplegados en el servidor con dirección IP 192.168.1.134:

Nota: La dirección IP 192.168.1.134 corresponde al servidor web y la 192.168.1.135 corresponde a Kali Linux.

- Con el siguiente comando, se indica a la herramienta que se trata de una petición de tipo POST. Para comprobar la existencia de la vulnerabilidad, se comprueba si el parámetro *usuario* del login del SSO (*Single Sign On*) de CAS (*Central Authentication Service*) es vulnerable mediante las palabras clave 'or'1'='1.

```
sqlmap -u http://192.168.1.134:8090/cas-server-webapp-4.0.0/login --
data="username='or'1'='1"
```

- El siguiente comando de *sqlmap* se ha aplicado sobre una aplicación simple desarrollada con Django y cuyo índice de noticias es vulnerable a SQLi, es decir, desde el *login* se puede acceder a toda la Base de Datos sin necesidad de autenticación.

```
sqlmap -u http://192.168.1.134:8000/news/?id=1
```

En las siguientes imágenes se puede observar de forma más clara:

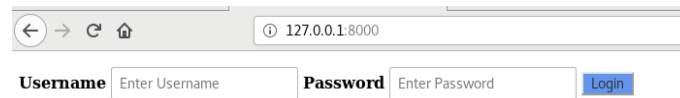


Ilustración 36: Login aplicación web Django.

Si sobre el *login* anterior, se realiza un ataque SQLi, se puede comprobar cómo se logra acceder a la base de datos y obtener los credenciales de usuario (pepe) y contraseña (1234) junto con el email (pepe@test.com) de todos los usuarios que aparecen registrados en el sistema.



Ilustración 37: SQLi aplicado sobre aplicación Django.

3.4.2. Exploración y visualización de eventos

Los eventos, anteriormente, detectados por Zeek IDS junto con los datos estadísticos y de auditoría, se visualizarán en Kibana. Para poder detectar y analizar el tráfico malicioso de la forma más rápida posible, se han configurado una serie de gráficos y dashboards que muestran información relevante actualizada en tiempo real logrando un análisis exhaustivo de la red de la organización, vigilando y monitorizando el tráfico y detectando posibles eventos maliciosos.

De esta forma, se logra aportar un grado de seguridad de tipo preventivo ante cualquier actividad sospechosa haciendo que el equipo de seguridad de la empresa reciba alertas anticipadas y pueda actuar con la mayor antelación posible para mitigar e intentar solucionar el problema.

3.4.2.1. Anomalías detectadas por Zeek IDS

Para analizar las anomalías detectadas por Zeek IDS en los registros *notice.log* e *intel.log*, se ha creado un dashboard con las siguientes visualizaciones (tablas, gráficos y diagramas):

➤ Resumen de posibles eventos maliciosos registrados por Zeek IDS:

src: Descending	dst: Descending	msg: Descending	@timestamp: Descending	Count
192.168.1.128	51.124.78.146	SSL certificate validation failed with (unable to get local issuer certificate)	April 22nd 2020, 09:29:05.565	1
192.168.1.128	177.71.36.213	Intel hit on 177.71.36.213 at Conn::IN_RESP	April 22nd 2020, 17:08:39.310	1
192.168.1.128	52.114.132.22	SSL certificate validation failed with (unable to get local issuer certificate)	April 22nd 2020, 10:01:42.057	1
192.168.1.128	52.114.77.164	SSL certificate validation failed with (unable to get local issuer certificate)	April 22nd 2020, 09:29:07.693	1
192.168.1.135	-	192.168.1.135 appears to be guessing SSH passwords (seen in 3 connections).	April 22nd 2020, 09:50:28.549	2
192.168.1.135	-	192.168.1.135 appears to be guessing SSH passwords (seen in 3 connections).	April 21st 2020, 12:20:48.177	1
192.168.1.135	-	An SQL injection attacker was discovered!	May 1st 2020, 12:19:10.178	1
192.168.1.135	192.168.1.132	192.168.1.135 scanned at least 15 unique ports of host 192.168.1.132 in 0m0s	April 21st 2020, 12:34:14.959	1
192.168.1.135	192.168.1.132	192.168.1.135 scanned at least 15 unique ports of host 192.168.1.132 in 0m0s	April 21st 2020, 12:58:58.843	1
192.168.1.135	192.168.1.132	192.168.1.135 scanned at least 15 unique ports of host 192.168.1.132 in 0m0s	April 21st 2020, 13:20:06.414	1
192.168.1.135	192.168.1.134	192.168.1.135 scanned at least 15 unique ports of host 192.168.1.134 in 0m0s	April 22nd 2020, 09:45:57.373	2
192.168.1.133	192.168.1.132	192.168.1.133 scanned at least 15 unique ports of host 192.168.1.132 in 0m0s	April 20th 2020, 17:05:47.403	1
192.168.1.133	192.168.1.132	192.168.1.133 scanned at least 15 unique ports of host 192.168.1.132 in 0m0s	April 20th 2020, 17:14:59.769	1

Ilustración 38: Resumen de eventos registrados por Zeek IDS.

➤ Anomalías registradas en el registro *notice.log*:

- Inyecciones SQL (2.27%).
- Ataques de fuerza bruta (9.09%).
- Escaneo de puertos (20.45%).
- Framework de Inteligencia (9.09%).
- Certificados no válidos o caducados (20.45%).
- Pérdida de paquetes por ICMP (31.82%).

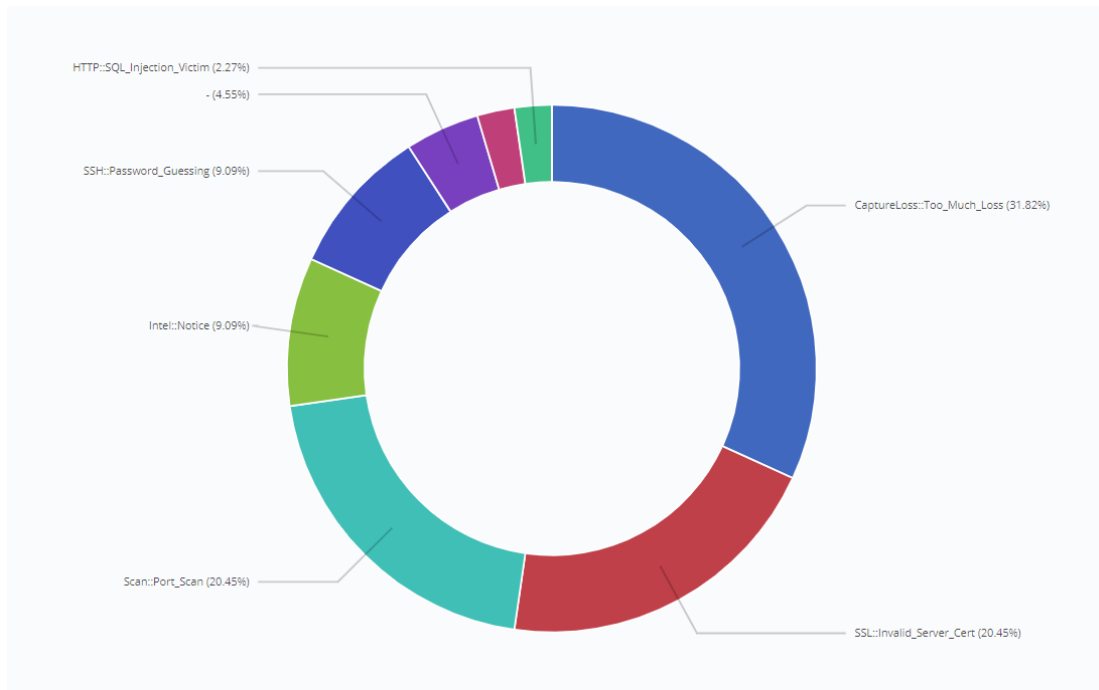


Ilustración 39: Anomalías registradas en notice.log.

➤ Anomalías detectadas por el Framework de Inteligencia. Direcciones IP maliciosas o URLs con las que se ha interactuado:

Nota: La dirección IP 172.217.17.3 corresponde a Google y se utilizó para realizar diversas pruebas con el Framework de Inteligencia.

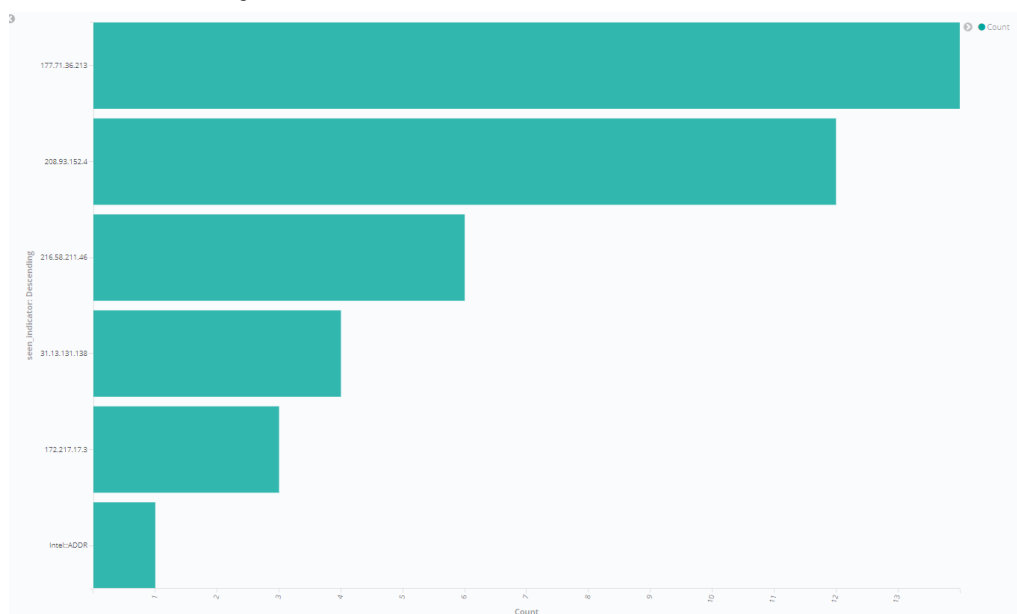


Ilustración 40: Anomalías detectadas por el Framework de Inteligencia.

- Cantidad de direcciones IP origen y destino involucradas en cada una de las anomalías detectadas en el registro *notice.log*. A continuación, se adjuntan dos ejemplos:

SSL::Invalid_Server_Cert: note: Descending		
src: Descending	dst: Descending	Count
192.168.1.128	192.168.1.134	4
192.168.1.128	51.124.78.146	3
192.168.1.128	52.114.132.22	1

Ilustración 41: Ej. split de tablas por cada anomalía detectada (Certificados inválidos).

Scan::Port_Scan: note: Descending		
src: Descending	dst: Descending	Count
192.168.1.135	192.168.1.132	3
192.168.1.135	192.168.1.134	2
192.168.1.133	192.168.1.132	2

Ilustración 42: Ej. split de tablas por cada anomalía detectada (Escaneo de puertos).

Una vez definidas todas las visualizaciones que nos permitirán analizar rápidamente las anomalías detectadas por Zeek IDS, se creará un dashboard que reunirá en una única ventana toda esta información.

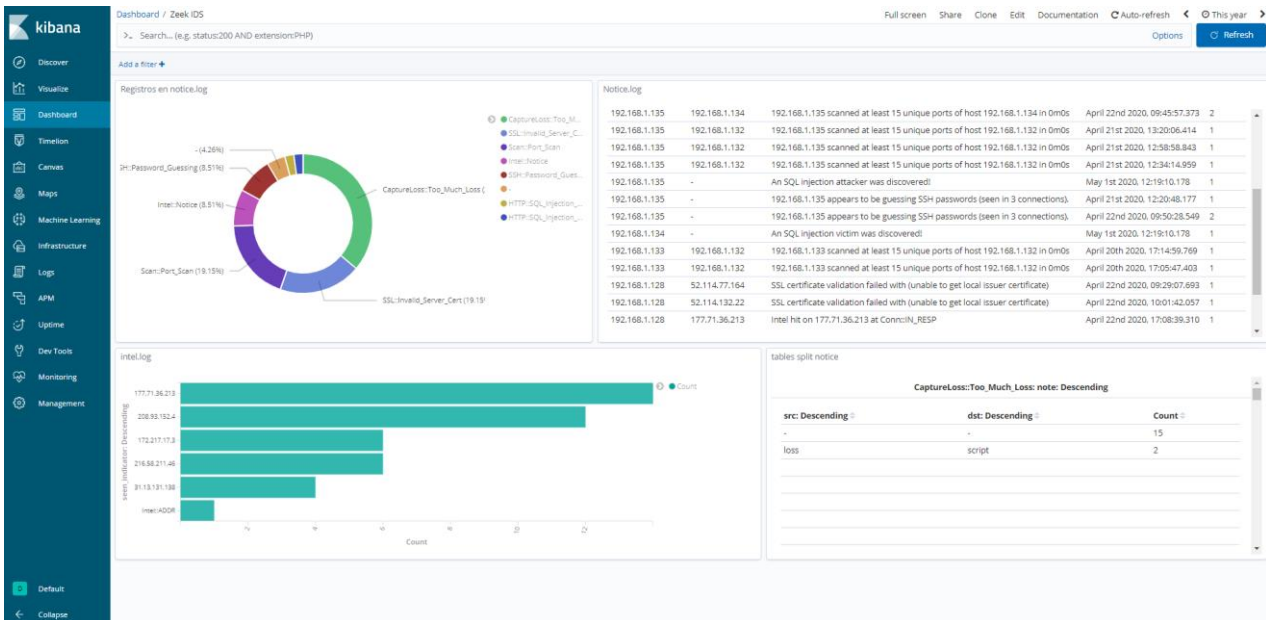


Ilustración 43: Dashboard de detección de anomalías en Kibana.

3.4.2.2. Datos estadísticos

Estos datos complementan los datos anteriores de cara al servidor web desplegado, ya que permiten identificar anomalías o situaciones extrañas producidos por problemas en el rendimiento de ciertos dispositivos en la red.

Revisando estas métricas, por ejemplo, se puede determinar si un dispositivo ha sido o está siendo infectado al presentar una tasa de tráfico anómala, o un uso de CPU o memoria por encima de la normal.

Por ejemplo: Cantidad de CPU y memoria en uso por el servidor web.

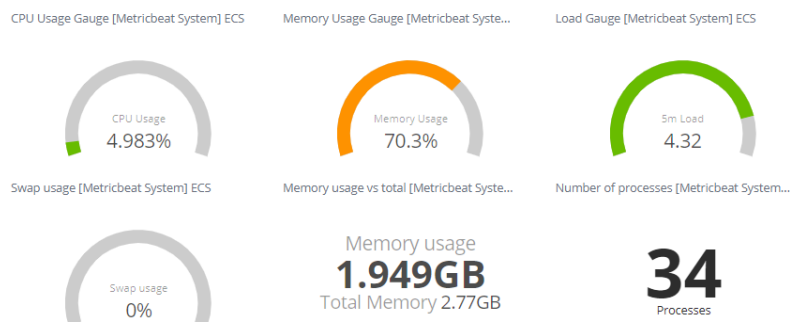


Ilustración 44: Uso de CPU y memoria del servidor.

El resto de visualizaciones de métricas se indican en el Anexo 7.3.

3.4.2.3. Datos de auditoría

De la misma forma que ocurre con los datos estadísticos, los datos de auditorías también complementan los datos anteriores de cara al servidor web desplegado, ya que permiten identificar anomalías o situaciones extrañas que se producen a través de métricas que registran la modificación de ficheros, inicios de sesión con éxito o fallidos realizados por un determinado usuario, integridad de ficheros o auditoría de eventos de Kernel:

Por ejemplo: Aparece cierta información del servidor web como la cantidad de inicios de sesión realizados, usuarios totales, procesos ejecutándose, paquetes transmitidos, entre otras métricas.

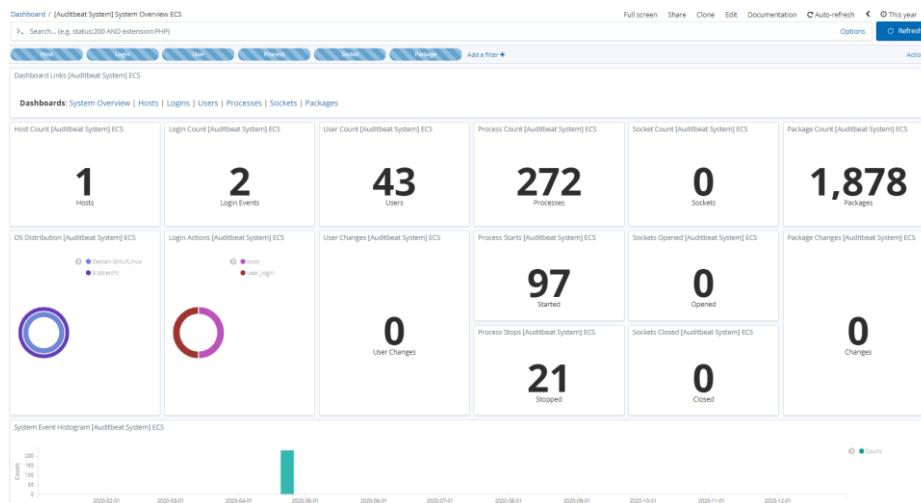


Ilustración 45: Dashboard Resumen del servidor web.

El resto de visualizaciones de métricas se indican en el Anexo 7.3.

4. Conclusiones Finales

Tras la finalización del proyecto y la construcción del sistema, a continuación, se describen las conclusiones finales del trabajo:

- Se ha comprobado que, gracias a la detección de Zeek IDS y las posibilidades que ofrece, ha sido una elección clave como motor de detección de intrusiones. La personalización gracias a los scripts que trae implementados y la posibilidad de utilizar su propio lenguaje de scripting lo convierten en una de las mejores opciones como IDS.
- El uso de ELK Stack como SIEM, ha proporcionado una recolección, procesamiento, búsqueda, almacenamiento y visualización de la información completa e intuitiva. Se ha comprobado que se trata de una herramienta muy completa y que ha cumplido con las expectativas que se buscaban en el proyecto. Adicionalmente, a nivel visual ofrece la posibilidad de crear diagramas y gráficos adaptados a las necesidades y generando dashboards que facilitan la monitorización y la efectiva detección de anomalías y tráfico en la red.
- Existen ciertas herramientas que permiten realizar diversos ataques que habría que comprobar si son detectados por Zeek correctamente o habría que desarrollar/modificar alguno de los scripts. Por ejemplo, como se ha visto, al realizar pruebas con la herramienta de Kali Linux, *hydra*, Zeek no lo detectaba correctamente con la suite de scripts que trae. Por lo tanto, pueden existir otras herramientas que ataquen a los elementos de la red y no sean detectados por Zeek.
- Se ha desarrollado un sistema, compuesto por Zeek y ELK Stack, que logra un análisis exhaustivo de la red, vigilando y monitorizando el tráfico y detectando posibles eventos maliciosos.
- Tras las pruebas realizadas, se puede concluir con que se ha desarrollado un sistema que aporta un grado de seguridad preventiva ante cualquier actividad sospechosa que ocurra en la red que se monitoriza, haciendo que el personal de seguridad detecte el problema y pueda actuar con antelación suficiente para mitigarlo.

4.1. Seguimiento de objetivos

El grado de cumplimiento de cada uno de los objetivos especificados al inicio del proyecto, han sido los siguientes:

Objetivos de investigación

- **Investigación y estudio de sistemas IDS.**
Se han investigado los tipos y posibilidades que ofrecen los sistemas IDS valorándose cuáles son, actualmente, los más utilizados.
- **Investigación y análisis del sistema IDS elegido: Zeek IDS.**
Dentro de los IDS más utilizados, se ha comprobado que Zeek IDS se encuentra entre los más relevantes y personalizables gracias a su lenguaje de scripting. Se ha estudiado la arquitectura de Zeek y las posibilidades que ofrece.
- **Investigación, análisis y estudio de sistemas SIEM.**
Se ha estudiado el funcionamiento de esos sistemas junto con las ventajas de seguridad que proporciona a las organizaciones. Se han valorado los sistemas SIEM más utilizados y las posibilidades que ofrecen.
- **Investigación y estudio del sistema SIEM elegido: ELK.**
Dentro de los sistemas SIEM más destacados, ELK forma parte de una de las soluciones más utilizadas. Se ha estudiado el almacenamiento con Elasticsearch, el tratamiento de logs con Logstash y el diseño de gráficos con Kibana. Adicionalmente, se han investigado distintos tipos de beats para obtener información relevante que complemente a los datos proporcionados por Zeek IDS.
- **Viabilidad y alcance de la integración de Zeek IDS con ELK.**
Se ha investigado diversos aspectos sobre la integración de Zeek IDS con ELK, a través de Filebeat, junto con la distinción de cada registro de logs en Logstash para ser procesados correctamente.
- **Estudio de los requisitos necesarios para la integración.**
El objetivo se considera cumplido, aunque la integración fue más simple de lo que inicialmente parecía, por lo que se requirió un estudio más leve.

Objetivos de implantación

➤ **Despliegue e instalación de Zeek IDS.**

Se ha desplegado e instalado Zeek IDS correctamente realizando la configuración de red de Virtual Box necesaria para la captura del tráfico.

➤ **Configuración del sistema IDS.**

El objetivo se ha cumplido satisfactoriamente, logrando configurar Zeek para monitorizar la red interna completa y capturar los logs necesarios.

➤ **Análisis sobre configuración de reglas y pruebas de detección.**

Se han realizado diversas pruebas de detección para comprobar que Zeek genera y detecta las anomalías necesarias para el proyecto. Se han implementado algunos scripts para añadir funcionalidades extra.

➤ **Despliegue e instalación de ELK.**

El objetivo se ha cumplido completamente. Se ha desplegado el servidor ELK junto con los tres componentes necesarios: Elasticsearch, Logstash y Kibana junto con una serie de beats para enviar datos y recolectar métricas.

➤ **Configuración del sistema SIEM.**

Se ha configurado ELK para que Filebeat recoja los datos de Zeek y los envíe a Logstash para ser procesados y convertidos a un formato estandarizado. Posteriormente, se almacenan en Elasticsearch y se visualizan en Kibana.

➤ **Realización de pruebas de funcionamiento y visualización de logs.**

El grado de cumplimiento de este objetivo ha sido notable. Se ha generado tráfico malicioso con Kali Linux (escaneo de puertos, SQLi, ataques de fuerza bruta) para comprobar que es detectado y enviado a Kibana.

➤ **Análisis e integración de IDS con SIEM.**

Se han integrado satisfactoriamente ambos sistemas mediante el envío de datos por Filebeat a Logstash y la carga de índices para Elasticsearch.

➤ **Pruebas de monitorización para comprobar el correcto funcionamiento de ambos sistemas.**

El objetivo se considera cumplido de forma completa, ya que se han realizado diversos ataques desde Kali Linux a diferentes servidores del sistema, logrando que se detecten en tiempo real y se visualicen en Kibana.

Objetivos finales

➤ **Detección de actividades anómalas o sospechosas que ocurran en la red.**

El objetivo se ha cumplido de forma satisfactoria. Zeek IDS detecta las actividades anómalas o tráfico sospechoso y lo registra en su sistema de logs. Estos datos se visualizan rápidamente desde los dashboards configurados en Kibana.

➤ **Reducción de los riesgos de intrusión minimizando posibles daños graves.**

El objetivo se ha cumplido satisfactoriamente. La detección de tráfico malicioso por Zeek es enviada a ELK. Tras ser estandarizada y almacenada, Kibana mostrará las alertas que se visualizarán en los diagramas presentes en los dashboards. Adicionalmente, en la opción *Discover* de Kibana aparecen toda la información procedente de Zeek y los distintos beats. Filtrando por los índices indicados, se pueden detectar las amenazas reduciendo el posible daño de forma significativa.

➤ **Ayuda a evitar comportamientos indebidos dentro de la red como el abuso de privilegios.**

Este objetivo complementa los dos anteriores. El abuso de privilegios se puede identificar analizando la información capturada por Zeek y complementarse con las métricas enviadas por los beats que reflejan los procesos ejecutados, inicios de sesión e información relevante sobre usuarios de los servidores relevantes del sistema.

4.2. Seguimiento de la planificación establecida

A pesar de la situación actual con el COVID-19, se ha logrado seguir la planificación indicada al inicio del proyecto y conseguir un grado de consecución de objetivos notable.

Aunque, en general, se ha seguido la planificación, ha habido ciertas tareas que han requerido de un tiempo menor y otras que, inicialmente, parecían más sencillas, han resultado necesitar más tiempo del esperado. Por ejemplo, la configuración de reglas y pruebas de detección de Zeek, estaban planificadas para ser finalizadas en un par de días cada una. Sin embargo, numerosas pruebas que generaban tráfico malicioso no eran detectadas por Zeek, ya que no creaba los registros *notice.log* e *intel.log*. Por lo

tanto, la planificación de estas dos tareas pasó de necesitar cuatro días a un total de un par de semanas hasta conseguir identificar la causa.

A pesar de estos problemas, el plan de trabajo se ha seguido de forma exacta de acuerdo al diagrama de Gantt diseñado.

4.2.1. Dificultades encontradas

Durante el desarrollo del proyecto, han existido una serie de problemáticas que han dificultado el seguimiento de la planificación detallada. Las más relevantes han sido las siguientes:

➤ **Framework de Inteligencia.**

Tras descargar los datasets en formato de Zeek que se utilizarán para comparar con el tráfico detectado, añadir los scripts necesarios y configuración de Zeek, se realizaron diversas pruebas accediendo a IPs vulnerables que aparecían en el dataset. Sin embargo, el registro *intel.log* no se creaba al no detectar ninguna anomalía.

Tras realizar numerosas pruebas sin éxito, el problema residió en que había que llevar a cabo un *redploy* de Zeek (*zeekctl install* y *zeekctl deploy*) para que cogiera los cambios y detectara las librerías y scripts añadidos. No era suficiente con un reinicio de Zeek.

➤ **Framework Notice (Ataques de fuerza bruta y SQLi).**

Desde Kali Linux se lanzaron diversos ataques para comprobar si eran detectados por Zeek. Los ataques de fuerza bruta con la herramienta *hydra* no era detectada por Zeek. Tampoco se detectaban los ataques SQLi con la herramienta *sqlmap* contra el servidor web *Tomcat*.

Como Zeek no generaba el fichero *notice.log*, se estuvo investigado y realizando numerosas pruebas sin éxito. Finalmente, se modificó el script de detección de ataques de fuerza bruta que utiliza Zeek para bajar el número de intentos de ataque y el tiempo. De esta forma, Zeek detectaba las pruebas que se realizaron.

Respecto a los ataques SQLi que tampoco eran detectados, se comprobó que, si el parámetro de tipo GET o POST no es vulnerable, por mucho que se intente realizar un ataque SQLi, Zeek no lo detectará. Posteriormente, se desplegó una aplicación web simple con Django que era vulnerable a SQLi pudiendo realizar un bypass de login y acceder a los datos de la base de datos. Se atacó con sqlmap a esta aplicación y Zeek detectó el intento de ataque correctamente.

➤ **Filtros y salida de Logstash.**

La información de log enviada por Filebeat no era correctamente procesada por Logstash. Este no distinguía cuál era el registro específico para aplicar los filtros especificados. Tras analizar los logs de Logstash sin éxito, se procedió a investigar la causa en Internet. Se descubrió que existe un comando que permite comprobar si la sintaxis del fichero de filtros es correcta. Extrañamente, había errores de sintaxis que no se reflejaban en los logs aun estando en modo *debug*. Después de arreglar estos errores, la información era estandarizada y enviada a Kibana correctamente.

➤ **Envío de datos desde el servidor de Zeek IDS al servidor de ELK Stack.**

El envío de datos por parte de Filebeat a Logstash no se realizaba correctamente. Esto conllevó el análisis de los logs de Logstash y Filebeat para comprobar por qué la comunicación no se estaba realizando correctamente.

Finalmente, se descubrió que el problema residía en la configuración de la infraestructura virtual de Virtual Box. El tráfico enviado desde el servidor de Zeek IDS no era recibido por el servidor ELK, aun existiendo una configuración de conexión de red de tipo “adaptador puente” que permite situar tanto el anfitrión como las máquinas virtuales en la misma red. El problema se producía porque el modo promiscuo no estaba configurado para “Permitir todo”, por lo que restringía el tráfico entre ambos servidores.

4.3. Líneas de trabajo futuro

Tras completar los objetivos detallados al inicio del proyecto, han quedado pendientes una serie de líneas de trabajo futuro con el objetivo de mejorar el sistema añadiendo y completando las funcionalidades actuales:

- Desarrollar nuevos scripts y modificar los actuales de Zeek IDS para adaptarlos a la situación deseada. Ha quedado pendiente la realización de más pruebas utilizando el lenguaje de scripting de Zeek que proporciona una personalización de la detección destacable.
- Utilizar el modo *cluster* de Zeek en vez del modo *standalone*. Con el modo *cluster* se optimiza y agiliza notablemente la detección de Zeek en una red bastante más compleja que la utilizada en el proyecto.
- Completar el envío de logs actuales con el resto de registros de logs generados por Zeek IDS. Actualmente, se envían los registros más relevantes (*notice.log*, *intel.log* y *conn.log*) que proporcionan una detección de anomalías y tráfico malicioso muy completo. Sin embargo, Zeek IDS genera otros registros como *http.log*, *ssl.log*, *file.log*, etc, que complementan la información anterior y pueden ser decisivos en la detección de algunos casos de anomalías.
- Añadir al Framework de Inteligencia de Zeek nuevas fuentes de obtención de datasets que recopilan información que es catalogada como altamente maliciosa. Actualmente, se utiliza CIF para obtener esta información. Sin embargo, existen otras fuentes que permiten añadir otros datos relevantes y completar la información alertando cuando se produzca interacciones con ellos desde la red interna.
- Mejorar los filtros aplicados en Logstash. Los filtros actuales están reducidos a los tres registros enviados por Zeek IDS. Sería conveniente ampliar y estudiar nuevas posibilidades de filtros que mejoren la información representada en Kibana.
- Desarrollar nuevos diagramas y gráficos en Kibana que se adapten en función de la necesidad presente. Los dashboards actuales se pueden completar para añadir más información relevante que sea necesaria para mejorar la rápida detección de tráfico malicioso.

5. Glosario

- **IDS:** Sistema de detección de intrusos (Intrusion Detection System).
- **SIEM:** Gestión de Eventos e Información de Seguridad (Security Information and Event Management).
- **Zeek IDS.** Herramienta Open-Source para el análisis de tráfico de red cuyo objetivo consiste en reconocer actividades sospechosas.
- **ELK Stack.** Conjunto de herramientas de gran potencial de código abierto que se combinan para crear una herramienta de administración de registros que permite la monitorización, consolidación y análisis de logs generados en múltiples servidores. ELK es la sigla para tres proyectos: Elasticsearch, Logstash y Kibana.
- **Elasticsearch.** Base de datos distribuida. Motor de búsqueda, RESTful basado en JSON. Es fácil de utilizar, escalable y flexible. Distribuye toda la información en todos los nodos, por tanto es tolerante a fallos y presenta alta disponibilidad.
- **Logstash.** Herramienta que proporciona el preprocesamiento antes de guardar la información en Elasticsearch, donde se recoge la entrada, se trabajan los eventos y se sacan por una salida, antes de ser almacenados en la base de datos.
- **Kibana.** Parte más visual de ELK Stack. Genera las visualizaciones sobre la información y dónde vamos a generar los dashboards.
- **Beats.** Familia de agentes de datos que recogen la información, ya sea de un fichero, log de datos, eventos, métricas, hacen comprobación de qué servicios se encuentran activos, analizan a nivel de red los paquetes y el tiempo de respuesta entre ellos.
- **IPS:** Sistema de prevención de intrusos (Intrusion prevention system).
- **SSH:** Protocolo y programa que implementa el acceso remoto a un servidor por medio de un canal seguro en el que toda la información está cifrada.
- **FTP:** Protocolo de transferencia de archivos entre sistemas conectados a una red TCP, basado en la arquitectura cliente-servidor.

- **DNS:** Sistema que asocia información variada con un nombre de dominio asociado.
- **HTTP:** Protocolo de comunicación que permite transferencias de información en la Word Wide Web (WWW).
- **HTTPS:** Protocolo de aplicación basado en el protocolo HTTP anterior, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.
- **HIDS:** Sistema de detección de intrusos en un Host. Su función es la de detectar anomalías revisando las actividades en la máquina (host).
- **NIDS:** Sistema de detección de intrusos en una Red. Su función es la de detectar anomalías analizando el tráfico de la red en tiempo real.
- **Exploit:** Software utilizado con el objetivo de aprovechar una vulnerabilidad de seguridad de un sistema de información para conseguir un comportamiento no deseado del mismo.
- **Kali Linux:** Distribución basada en Debian GNU/Linux diseñada para la auditoría y seguridad informática en general. Trae preinstalados más de 600 programas incluyendo Nmap, Wireshark, John the Ripper, Aircrack-ng, etc.
- **Hydra:** Aplicación de Kali Linux utilizada para crackear contraseñas y conseguir acceder de forma no autorizada a redes y sistemas.
- **Nmap:** Aplicación que sirve para efectuar rastreo de puertos.
- **SQLi:** Inyección SQL. Método de infiltración de código intruso que utiliza una vulnerabilidad presente en una aplicación para realizar operaciones sobre la base de datos.

6. Bibliografía

Sistemas de Detección de Intrusiones (IDS). Introducción.

- Ecured. “IDS”.
URL: <https://www.ecured.cu/IDS>
- CCM. “Sistema de detección de intrusiones (IDS)”.
URL: <https://es.ccm.net/contents/162-sistema-de-deteccion-de-intrusiones-ids>
- Documento GIAC CERTIFICATIONS. “An introduction to Intrusion Detection Systems”.
URL: <https://www.giac.org/paper/gsec/4227/introduction-intrusion-detection-systems/106775>

Sistemas de Detección de Intrusiones (IDS). Tipos de IDS.

- CCM. “Sistema de detección de intrusiones (IDS)”.
URL: <https://es.ccm.net/contents/162-sistema-de-deteccion-de-intrusiones-ids>
- Documento GIAC CERTIFICATIONS. “An introduction to Intrusion Detection Systems”.
URL: <https://www.giac.org/paper/gsec/4227/introduction-intrusion-detection-systems/106775>
- Wikipedia. “Intrusion detection system”.
URL: https://en.wikipedia.org/wiki/Intrusion_detection_system

Sistemas de Detección de Intrusiones (IDS). Herramientas IDS más destacadas.

- Openwebinars. “Las 8 mejores herramientas open source de detección de intrusión”.
URL: <https://openwebinars.net/blog/las-8-mejores-herramientas-open-source-de-deteccion-de-intrusion/>
- Protegermipc. “Mejores IDS Opensource para Detección de Intrusiones”.
URL: <https://protegermipc.net/2017/02/22/mejores-ids-opensource-deteccion-de-intrusiones/>

Sistemas de Detección de Intrusiones (IDS). Herramienta a implementar: Zeek IDS.

- Zeek. "Introduction".

URL: <https://docs.zeek.org/en/current/intro/index.html#overview>

Sistemas de gestión de información y eventos de seguridad (SIEM). Introducción. Funcionamiento.

- Blog imf-formacion. "Qué significa SIEM y cómo funciona".
URL: <https://blogs.imf-formacion.com/blog/tecnologia/que-significa-siem-y-como-funciona-201808/>
- Sofecom. "SIEM, la tecnología capaz de detectar y neutralizar las amenazas informáticas antes de que ocurran".
URL: <https://sofecom.com/que-es-un-siem/>
- Viewnext. "¿Qué es un SIEM?".
URL: <https://www.viewnext.com/que-es-un-siem/>
- Hydrasky. "SIEM Overview".
URL: <https://hydrasky.com/network-security/security-information-and-event-managementsiem-overview/>

Sistemas de gestión de información y eventos de seguridad (SIEM). Herramientas SIEM más destacadas.

- DNSstuff. "10 Best SIEM Tools in 2020".
URL: <https://www.dnsstuff.com/siem-tools>
- Software Testing Help. "Top 11 best SIEM Tools In 2020 For Real-Time Incident Response and Security".
URL: <https://www.softwaretestinghelp.com/siem-tools/>
- Enterprise Management 360. "The Top 10 SIEM Tools to Try for 2019".
URL: <https://www.em360tech.com/continuity/tech-features-featuredtech-news/siem-tools-top-10/>

Sistemas de gestión de información y eventos de seguridad (SIEM). Herramienta a implementar: ELK Stack.

- OpenWebinars. "¿Qué es ELK? Elasticsearch, Logstash y Kibana".
URL: <https://openwebinars.net/blog/que-es-elk-elasticsearch-logstash-y-kibana/>

- Guru99. “ELK Stack Tutorial: Learn Elasticsearch, Logstash and Kibana”.
URL: <https://www.guru99.com/elk-stack-tutorial.html>
- Enimbos. “Monitorización de aplicaciones usando ELK Stack”.
URL: <https://enimbos.com/monitorizacion-de-aplicaciones-usando-elk-stack/>

Agentes. Instalación. Filebeat. Metricbeat. Auditbeat.

- Documentación oficial Elastic. Step 1: Install Filebeat.
URL: <https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-installation.html>
- Documentación oficial Elastic. Step 2: Configure Filebeat.
URL: <https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-configuration.html>
- Logz.io. Integrating Bro IDS with the ELK Stack
URL: <https://logz.io/blog/bro-elk-part-1/>
- ObjectRocket. What are Elasticsearch Beats?
URL: <https://www.objectrocket.com/resource/what-are-elasticsearch-beats/>
- PANDORAFMS. Monitorización Ligera con Elastic Metricbeat ¿Una nueva tendencia?.
URL: <https://pandorafms.com/blog/es/monitorizacion-ligera/>
- Documentación oficial Elastic. Step 1: Install Metricbeat.
URL: <https://www.elastic.co/guide/en/beats/metricbeat/current/metricbeat-installation.html>
- Documentación oficial Elastic. Step 2: Configure Metricbeat.
URL: <https://www.elastic.co/guide/en/beats/metricbeat/current/metricbeat-configuration.html>
- Documentación oficial Elastic. Step 4: Set up the Kibana dashboards (Metricbeat).
URL: <https://www.elastic.co/guide/en/beats/metricbeat/current/load-kibana-dashboards.html>
- Documentación oficial Elastic. Step 1: Install Auditbeat.
URL: <https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-installation.html>
- Documentación oficial Elastic. Step 2: Configure Auditbeat.
URL: <https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-configuration.html>

- Documentación oficial Elastic. Step 3: Load the index template in Elasticsearch.
URL: <https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-template.html>
- Documentación oficial Elastic. Step 4: Set up the Kibana dashboards (Auditbeat)
URL: <https://www.elastic.co/guide/en/beats/auditbeat/current/load-kibana-dashboards.html>

Zeek IDS. Instalación.

- Manual Oficial Zeek. Installation.
URL: <https://docs.zeek.org/en/current/install/install.html>
- Manual Oficial Zeek. Quick Start Guide.
URL: <https://docs.zeek.org/en/current/quickstart/index.html>
- Repositorio Oficial de Zeek en Github.
URL: <https://github.com/zeek/zeek>

Zeek IDS. Configuración, Logs y Scripting.

- Manual Oficial Zeek. Logging.
URL: <https://docs.zeek.org/en/current/examples/logs/index.html>
- Manual Oficial Zeek. Intelligence Framework.
URL: <https://docs.zeek.org/en/current/frameworks/intel.html>
- Manual Oficial Zeek. Logging Framework.
URL: <https://docs.zeek.org/en/current/frameworks/logging.html>
- Manual Oficial Zeek. Log Files.
URL: <https://docs.zeek.org/en/current/script-reference/log-files.html>
- Critical Stack. Resumen Bro/Zeek Logs.
URL: http://gauss.ececs.uc.edu/Courses/c6055/pdf/bro_log_vars.pdf
- Manual Oficial Zeek. IDS Scripts.
URL: <https://docs.zeek.org/en/current/examples/ids/index.html#>
- Manual Oficial Zeek. Zeek Script Index.
URL: <https://docs.zeek.org/en/current/script-reference/scripts.html>
- Exploding Woodchucks. How to integrate CIF feeds into Bro Intel files.
URL: <https://www.explodingwoodchucks.com/feed-bro-intel-files/>

- HOLDMYBEER. DETECTING SSH BRUTE FORCING WITH ZEEK.
URL:<https://holdmybeersecurity.com/2019/04/17/detecting-ssh-brute-forcing-with-zeek/>

ELK Stack. Instalación y configuración. Recopilación de información. Almacenamiento e Indexación. Exploración y visualización de datos.

- Fosslinux. How to install ELK Stack on Ubuntu 18.04.
URL:<https://www.fosslinux.com/6084/how-to-install-elk-stack-on-ubuntu-18-04.htm>
- Digital Ocean. Cómo instalar Elasticsearch, Logstash y Kibana (Elastic Stack) en Ubuntu 18.04.
URL:<https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-18-04-es>
URL:<https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-installation.html>
- Documentación oficial Elastic. Step 2: Configure AuditBeat.
URL:<https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-configuration.html>

7. Anexos

7.1. Instalaciones

7.1.1. Agentes

La instalación de todos los Beats es muy parecida. A continuación, se detalla la instalación de Filebeat que ilustrará, a modo de ejemplo, el procedimiento a seguir para instalar el resto de Beats:

1. Se importa el repositorio utilizando la llave pública GPG:

```
sudo wget -Oo - https://artifacts.elastic.co/GPG-KEY-elasticsearch |  
sudo apt-key add -
```

2. Se instala el paquete “apt-transport-https” (necesario para securizar la conexión con Logstash) y se añade el repositorio anterior a la lista *sources.list*:

```
sudo apt-get install apt-transport-https  
  
sudo echo “deb https://artifacts.elastic.co/packages/6.x/apt stable  
main” | sudo tee -a /etc/apt/sources.list.d/elastic-6.x.list
```

3. Se actualiza la lista de paquetes y se instala Filebeat:

```
sudo apt-get update  
  
sudo apt-get install filebeat
```

4. Seguidamente, tenemos que crear el certificado SSL para Filebeat de forma que la conexión con Logstash sea segura. Para ello, copiamos el certificado *logstash-forwarder.crt* generado anteriormente en el servidor ELK Stack para Logstash, y lo creamos en la siguiente ruta */etc/filebeat/logstash-forwarder.crt*.
5. Finalmente, se lanza el servicio y se configura para que se inicie cuando arranque el servidor:

```
sudo systemctl enable filebeat.service
sudo systemctl start filebeat.service
```

7.1.2. Zeek IDS

Zeek requiere de una serie de librerías y dependencias para su correcto funcionamiento:

- Libpcap
- OpenSSL
- BIND8
- Libz
- Python 2.6 o superior
- Bash (para ZeekControl)

Los pasos necesarios para instalar Zeek serían los siguientes:

1. Actualizar el listado de paquetes disponibles.

```
sudo apt-get update
```

2. Instalar las librerías y dependencias detalladas anteriormente.

```
sudo apt-get install cmake make gcc g++ flex gdb bison libmagic-dev
libgeoip-dev libpcap-dev libssl-dev python-dev swig zlib1g-dev
```

3. Instalar Zeek desde código fuente utilizando el repositorio de Github.

```
git clone --recursive https://github.com/zeek/zeek
./configure
make
sudo make install
```

Zeek se instalará por defecto en el directorio: `/usr/local/zeek`.

4. Configurar la variable de entorno.

```
export PATH=/usr/local/zeek/bin:$PATH
```

Los ficheros de configuración se encuentran ubicados en `/usr/local/zeek/etc`:

- `node.cfg`
- `networks.cfg`
- `broctl.cfg`

La herramienta se puede iniciar utilizando el Shell interactivo llamado ZeekControl mediante el comando:

```
zeekctl
```

Si es la primera vez que se utiliza la Shell, se requiere realizar una instalación de la configuración de ZeekControl y posteriormente, lanzar una instancia:

```
[ZeekControl] > install  
[ZeekControl] > start
```

7.1.3. ELK Stack

Elasticsearch y Logstash requieren de Java 8 para funcionar correctamente. Por lo tanto, vamos a comenzar instalando Java 8 como paso inicial antes de adentrarnos en la instalación de los distintos componentes que forman ELK Stack.

7.1.3.1. Elasticsearch

Los componentes de Elasticsearch no están disponibles en los repositorios de paquetes predeterminados de Ubuntu. Adicionalmente, todos los paquetes de ELK están firmados con la clave de firma de Elasticsearch para proteger su sistema de suplantaciones, por lo que será necesario importar la clave GPG pública de Elasticsearch y agregar la lista de fuentes de paquetes de Elastic para instalarlo.

Los pasos necesarios para instalar y configurar Elasticsearch son los siguientes:

1. Se importa la clave de GPG pública de Elasticsearch en APT:

```
sudo wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch |
sudo apt-key add -
```

2. Se instala el paquete *apt-transport-https* y añadimos el repositorio donde APT buscará nuevas fuentes:

```
sudo apt-get install apt-transport-https "deb
https://artifacts.elastic.co/packages/6.x/apt stable main" | sudo tee -a
/etc/apt/sources.list.d/elastic-6.x.list
```

3. Se actualiza la lista de repositorios y se instala el paquete:

```
sudo apt-get update
sudo apt-get install elasticsearch
```

4. Se modifica el fichero *elasticsearch.yml* añadiendo la siguiente configuración. Elasticsearch escucha el tráfico por el puerto 9200.

```
sudo vim /etc/elasticsearch/elasticsearch.yml
# network.host: 192.168.1.133
```

Con la configuración añadida, se restringe el acceso externo a Elasticsearch para que, únicamente, a través de la dirección IP del servidor, se puedan leer los datos evitando que terceros accedan a esta información. No se deja definido *localhost*, ya que es necesario acceder desde otros servidores como el de Zeek IDS.

5. Se comprueba que Elasticsearch funciona correctamente. Para ello, habilitamos el servicio para que arranque cuando el servidor inicie y chequeemos la instalación:

```
sudo systemctl enable elasticsearch.service
sudo systemctl start elasticsearch.service
sudo curl -XGET 'localhost:9200'
```

Finalmente, ya se encontraría Elasticsearch correctamente instalado en el servidor.


```
root@elkstack:/home# sudo curl -XGET 'localhost:9200/?pretty'
{
  "name" : "FCRw1f1",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "TttjqjIvSjuPyoGD4wa7Wg",
  "version" : {
    "number" : "6.8.8",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "2f4c224",
    "build_date" : "2020-03-18T23:22:18.622755Z",
    "build_snapshot" : false,
    "lucene_version" : "7.7.2",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Ilustración 46: Verificación instalación Elasticsearch.

7.1.3.2. Kibana

Según la documentación oficial, se deberá instalar Kibana tras instalar Elasticsearch. Es importante seguir este orden para garantizar que los componentes de los que depende cada producto estén correctamente implementados.

Los pasos necesarios para instalar y configurar Kibana son los siguientes:

1. Se instala Kibana y se añaden las siguientes líneas a la configuración:

```
sudo apt-get install kibana
sudo vim /etc/kibana/kibana.yml

server.port: 5601
server.host: "192.168.1.133"
elasticsearch.hosts "http://localhost:9200"
```

En el fichero de configuración, se define la dirección IP del servidor para poder acceder desde el servidor de Zeek IDS. El puerto de escucha sería el 5601. El servidor donde reside Elasticsearch se mantiene en *localhost*, ya que ambos componentes se encuentran desplegados en el servidor ELK Stack.

2. Finalmente, habilitamos el servicio para que arranque al iniciar el servidor y lo lanzamos:

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

Seguidamente, se podrá visualizar la interfaz de Kibana escuchando en el puerto 5601. La dirección IP 192.168.1.133 corresponde al servidor ELK (máquina virtual):

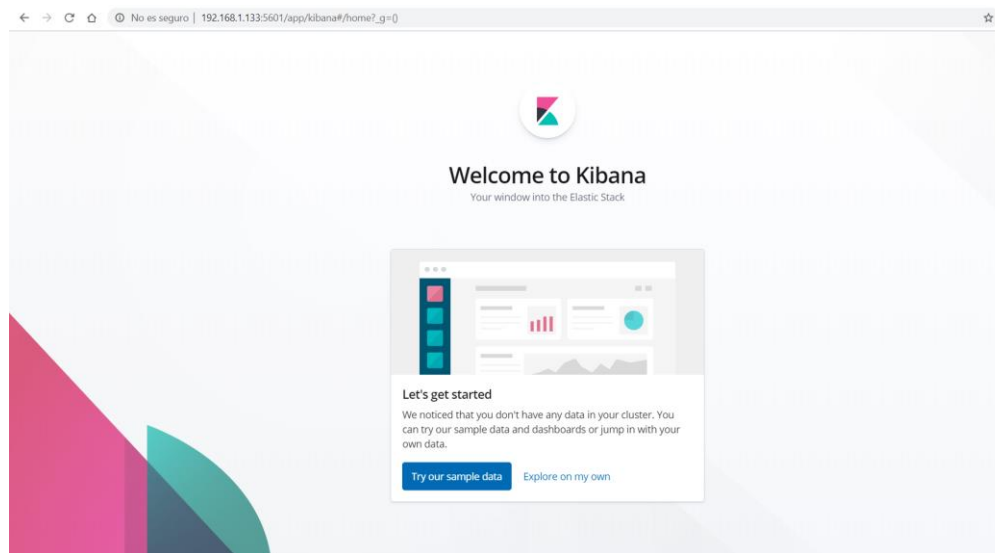


Ilustración 47: Interfaz de Kibana.

Configuración de Nginx como Proxy inverso para Kibana

Debido a que Kibana está configurado para escuchar solo en localhost, debemos configurar un proxy inverso para permitir el acceso externo a este. En este proyecto, no sería necesario realizar estos pasos, ya que solo accederemos desde localhost y desde la propia red interna.

1. Se instala Nginx:

```
sudo apt-get install nginx apache2-utils
```

2. Se crea un archivo de bloque del servidor de Nginx "elk". y se añade el siguiente bloque de código al fichero de configuración:

```
sudo vim /etc/nginx/sites-available/elk
```

```
server {  
    listen 80;  
    server_name elk.com;  
    auth_basic "Restricted Access";  
}
```

```
auth_basic_user_file /etc/nginx/.elkusersecret;
location / {
    proxy_pass http://192.168.1.133:5601;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
}
}
```

Con el código anterior, se configura Nginx para que redirija el tráfico HTTP de su servidor a la aplicación de Kibana que escucha por el puerto 5601. También, se configura Nginx para leer el fichero donde vamos a almacenar las credenciales para acceder a la interfaz web de Kibana.

3. Se crea un usuario y contraseña para la autenticación del navegador web:

```
sudo htpasswd -c /etc/nginx/.elkusersecret elkusr
```

4. Se comprueba que no existan errores en la sintaxis de configuración:

```
sudo nginx -t
```

5. Se habilita Nginx al iniciar el servidor y reiniciamos el servicio:

```
sudo systemctl enable nginx.service
sudo systemctl restart nginx.service
```

7.1.3.3. Logstash

Aunque sería posible que Filebeat enviase los datos directamente a la base de datos de Elasticsearch, se recomienda utilizar Logstash para procesarlos. De esta forma, se recopilarán los datos procedentes de distintas fuentes, se transformarán a un formato común y se exportarán a una base de datos.

Los pasos necesarios para instalar y configurar Logstash son los siguientes:

1. Se instala Logstash:

```
sudo apt-get install logstash
```

2. Los siguientes pasos consisten en crear un certificado SSL para securizar la conexión entre Logstash y Filebeat. Además, se creará un fichero de entrada para recoger la información enviada por Filebeat y un fichero de configuración de filtro de syslog. Este filtro se utilizará para analizar los registros de sistema entrantes con el objetivo de que tengan una estructura y poder utilizarse en los paneles de Kibana. Adicionalmente, es necesario crear un fichero de salida que indique a Logstash que estos datos de Beats tiene que almacenarlos en Elasticsearch.

Estos pasos se han explicado más detalladamente en la sección “Recolección de datos”.

3. Finalmente, le decimos a Logstash que arranque al encender el servidor y lanzamos el servicio:

```
sudo systemctl enable logstash.service  
sudo systemctl start logstash.service
```

7.2. Ficheros de configuración

Fichero de Scripts de Zeek IDS `local.zeek`

```
@load misc/loaded-scripts  
@load tuning/defaults  
# Estimar pérdida de paquetes  
@load misc/capture-loss  
# Permitir estadísticas de memoria, paquetes y lag  
@load misc/stats  
# Carga scripts de escaneo de red  
@load misc/scan  
# Detección de versiones de software vulnerables  
@load frameworks/software/vulnerable  
# Detección de cambios de software  
@load frameworks/software/version-changes  
# Detección de software en varios protocolos  
@load protocols/ftp/software  
@load protocols/smtp/software  
@load protocols/ssh/software
```

```

@load protocols/http/software
# Detectar resultados sospechosos de DNS
@load protocols/dns/detect-external-names
# Detectar sesiones FTP
@load protocols/ftp/detect
# Monitorización de servicios, certificados y dispositivos
@load protocols/conn/known-hosts
@load protocols/conn/known-services
@load protocols/ssl/known-certs
# Validación de certificado
@load protocols/ssl/validate-certs
# Utiliza GeoIP para hacer detecciones geográficas y registros para
tráfico SSH
@load protocols/ssh/geo-data
# Detección de ataques por fuerza bruta SSH
@load protocols/ssh/detect-bruteforcing
# Detección de inyecciones SQL
@load protocols/http/detect-sqli
# Verificación hash de ficheros MD5/SHA
@load frameworks/files/hash-all-files
# Detección de hash SHA1 en Registro Team Cymru
@load frameworks/files/detect-MHR
# Detección de alertas de email incluyendo hostnames
@load policy/frameworks/notice/extend-email/hostnames
# DATOS DE INTELIGENCIA AÑADIDOS
#Soporta metadatos usados por CIF
@load frameworks/intel/seen
@load frameworks/intel/do_notice
@load policy/integration/collective-intel
redef Intel::read_files += {
"/opt/feeds/port.intel", #Rutas donde se encuentran almacenados los
feeds obtenidos del gadget CSIRTG y que utilizará Zeek para comparar
"/opt/feeds/url.intel",
"/opt/feeds/ip.intel",
};

```

Fichero de Configuración de Filebeat filebeat.yml

```

#Inputs (Entradas)
filebeat.inputs:
#Fichero log CONN.LOG de ZEEK
- type: log
  enabled: true
  paths:
    - /usr/local/zeek/logs/current/conn.log
  fields:
    type: "zeek-conn"
  fields_under_root: true
#Fichero log INTEL.LOG de ZEEK
- type: log
  enabled: true
  paths:
    - /usr/local/zeek/logs/current/intel.log
  fields:
    type: "zeek-intel"
  fields_under_root: true
#Fichero log NOTICE.LOG de ZEEK
- type: log
  enabled: true
  paths:
    - /usr/local/zeek/logs/current/notice.log
  fields:
    type: "zeek-notice"

```

```
fields_under_root: true
#Outputs (salidas) Logstash y definición de certificado SSL/TLS
hosts: ["192.168.1.133:5044"]
ssl.certificate_authorities: ["/etc/filebeat/logstash-forwarder.crt"]
```

Fichero de Configuración de Metricbeat metricbeat.yml

```
metricbeat.config.modules:
# Definición de patrones globales para definir la configuración
path: ${path.config}/modules.d/*.yml
# Habilitar reloading de la configuración
reload.enabled: false
# Configuración template Elasticsearch
setup.template.settings:
  index.number_of_shards: 1
  index.codec: best_compression
# Kibana
setup.kibana:
  host: "192.168.1.132:5601"
# Salida hacia Elasticsearch
output.elasticsearch:
  hosts: ["192.168.1.133:9200"]
# Procesadores
processors:
- add_host_metadata: ~
- add_cloud_metadata: ~
- add_docker_metadata: ~
- add_kubernetes_metadata: ~
```

Fichero de Configuración de Auditbeat auditbeat.yml

```
auditbeat.modules:
- module: auditd
  # Load audit rules from separate files. Same format as audit.rules(7).
  audit_rule_files: [ '${path.config}/audit.rules.d/*.conf' ]
- module: file_integrity
  paths:
  - /bin
  - /usr/bin
  - /sbin
  - /usr/sbin
  - /etc
- module: system
  datasets:
  - host # General host information, e.g. uptime, IPs
  - login # User logins, logouts, and system boots.
  - package # Installed, updated, and removed packages
  - process # Started and stopped processes
  - socket # Opened and closed sockets
  - user # User information
  state.period: 12h
  user.detect_password_changes: true
  # Logs
  login.wtmp_file_pattern: /var/log/wtmp*
  login.btmp_file_pattern: /var/log/btmp*
# Kibana
setup.kibana:
  host: "192.168.1.133:5601"
```

```
# Salida hacia Elasticsearch
output.elasticsearch:
  hosts: ["192.168.1.133:9200"]
```

Fichero de configuración de filtros de Logstash syslog-filter.conf

```
#Paso 1: Se eliminan los comentarios
filter {
  if [message] =~ /^#/ {
    drop { }
  }
}

#Modificaciones en fichero conn.log

  if [type] == "zeek-conn" {

# Paso 2: Se establecen los nombres de campos
  csv {
    columns => ["ts","uid","id.orig_h","id.orig_p","id.resp_h","id.resp_p","proto","service",
"duration","orig_bytes","resp_bytes","conn_state","local_orig","local_resp","missed_bytes","history",
"orig_pkts","orig_ip_bytes","resp_pkts","resp_ip_bytes","tunnel_parents", "orig_cc", "resp_cc"]

#Se filtra por tabulador
    separator => " "
  }

# Paso3 :Cambio de timestamp a tipo fecha
  date {
    match => [ "ts", "UNIX" ]
  }

# Paso 4: Modificación de nombres de campos y tipos de datos
  mutate {
    convert => { "id.orig_p" => "integer" }
    convert => { "id.resp_p" => "integer" }
    convert => { "orig_bytes" => "integer" }
    convert => { "duration" => "float" }
    convert => { "resp_bytes" => "integer" }
    convert => { "missed_bytes" => "integer" }
    convert => { "orig_pkts" => "integer" }
    convert => { "orig_ip_bytes" => "integer" }
    convert => { "resp_pkts" => "integer" }
    convert => { "resp_ip_bytes" => "integer" }
    convert => { "orig_cc" => "integer" }
    convert => { "resp_cc" => "integer" }
    rename => { "id.orig_h" => "id_origen_host" }
    rename => { "id.orig_p" => "id_origen_port" }
    rename => { "id.resp_h" => "id_destino_host" }
    rename => { "id.resp_p" => "id_destino_port" }
  }
}

#Modificaciones para notice.log

# Paso 2: Se establecen los nombres de campos
  if [type] == "zeek-notice" {
    csv {
      columns =>
["ts","uid","id.orig_h","id.orig_p","id.resp_h","id.resp_p","fuid","file_mime_type","file_desc",
"proto","note", "msg","sub","src","dst","p","n","peer_descr","actions","suppress_for","dropped",
```

```

"remote_location.country_code","remote_location.region","remote_location.city",
"remote_location.latitude","remote_location.longitude"]

#Se filtra por tabulador
    separator => " "
}

# Paso3 :Cambio de timestamp a tipo fecha
    date {
        match => [ "ts", "UNIX" ]

    }

# Paso 4: Modificación de nombres de campos y tipos de datos
    mutate {
        convert => ["id.orig_p", "integer"]
        convert => ["id.resp_p", "integer"]
        convert => [ "p", "integer" ]
        convert => [ "n", "integer" ]
        convert => [ "suppress_for", "float"]
        convert => [ "remote_location.latitude", "float"]
        convert => [ "remote_location.longitude", "float" ]
        rename => { "id.orig_h" => "id_origen_host" }
        rename => { "id.orig_p" => "id_origen_port" }
        rename => { "id.resp_h" => "id_destino_host" }
        rename => { "id.resp_p" => "id_destino_port" }
        rename => [ "remote_location.country_code", "remote_location_country_code" ]
        rename => [ "remote_location.region", "remote_location_region" ]
        rename => [ "remote_location.city", "remote_location_city" ]
        rename => [ "remote_location.latitude", "remote_location_latitude" ]
        rename => [ "remote_location.longitude", "remote_location_longitude" ]

    }
}

```

#Modificaciones en fichero intel.log

```

    if [type] == "zeek-intel" {

# Paso 2: Se establecen los nombres de campos
        csv {
            columns =>
["ts","uid","id.orig_h","id.orig_p","id.resp_h","id.resp_p","seen.indicator","seen.indicator_type",
"seen.where","seen.node","matched","sources","fuid","file_mime_type","file_desc","cif.tags",
"cif.confidence","cif.source","cif.description","cif.firstseen","cif.lastseen"]

#Se filtra por tabulador
            separator => " "
        }
# Paso3 :Cambio de timestamp a tipo fecha
        date {
            match => [ "ts", "UNIX" ]

        }

# Paso 4: Modificación de nombres de campos y tipos de datos
        mutate {
            convert => [ "id.orig_p", "integer" ]
            convert => [ "id.resp_p", "integer" ]
            convert => [ "matched", "float" ]
            rename => { "id.orig_h" => "id_origen_host" }
            rename => { "id.orig_p" => "id_origen_port" }
            rename => { "id.resp_h" => "id_destino_host" }
            rename => { "id.resp_p" => "id_destino_port" }
            rename => [ "seen.indicator", "seen_indicator" ]
            rename => [ "seen.where", "seen_where" ]
            rename => [ "seen.node", "seen_node" ]

```



```

rename => [ "cif.tags", "cif_tags" ]
rename => [ "cif.confidence", "cif_confidence" ]
rename => [ "cif.source", "cif_source" ]
rename => [ "cif.description", "cif_description" ]
rename => [ "cif.firstseen", "cif_firstseen" ]
rename => [ "cif.lastseen", " cif_lastseen " ]

}

}

```

7.3. Exploración y visualización de eventos adicionales

Métricas de Auditbeat y Metricbeat analizadas en los dashboards de Kibana.

7.3.1. Datos estadísticos

- Tráfico entrante y saliente y disco utilizado por el servidor web.

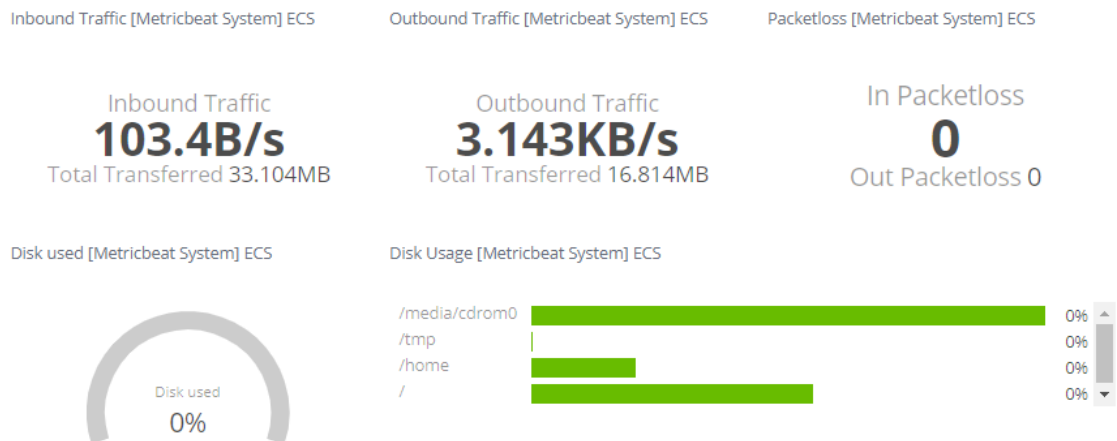


Ilustración 48: Tráfico entrante y saliente obtenido del servidor web.

- Uso de CPU del sistema.

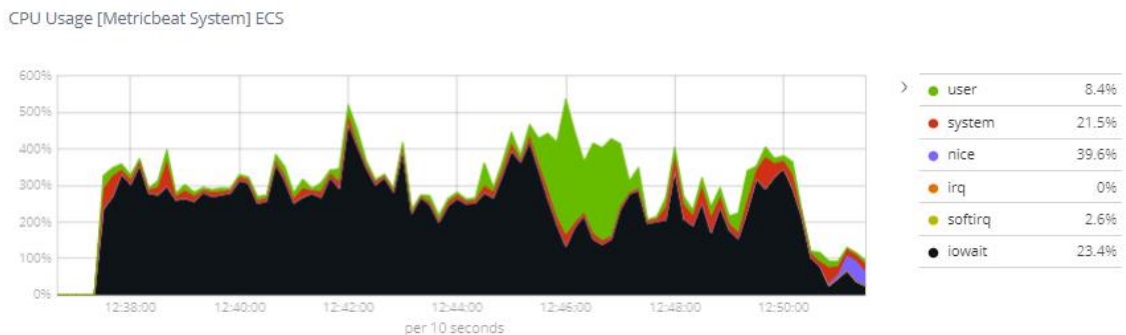


Ilustración 49: Uso de CPU del servidor web.

- Uso de memoria del servidor web (libre, utilizada y en caché).

Memory Usage [Metricbeat System] ECS

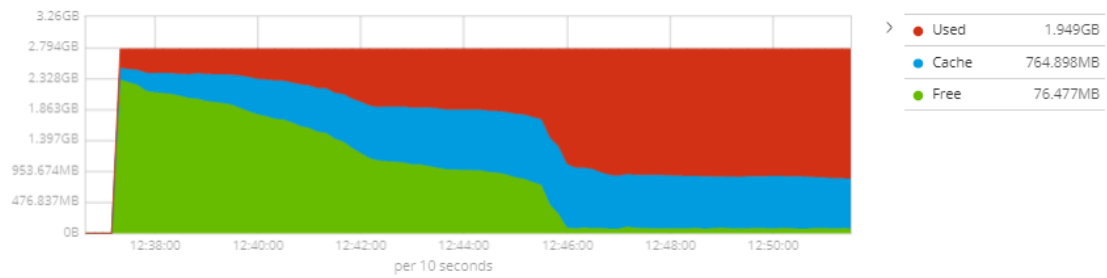


Ilustración 50: Uso de memoria del servidor web.

- Gráfica por horas que refleja el tráfico entrante y saliente en la red.

Network Traffic (Bytes) [Metricbeat System] ECS

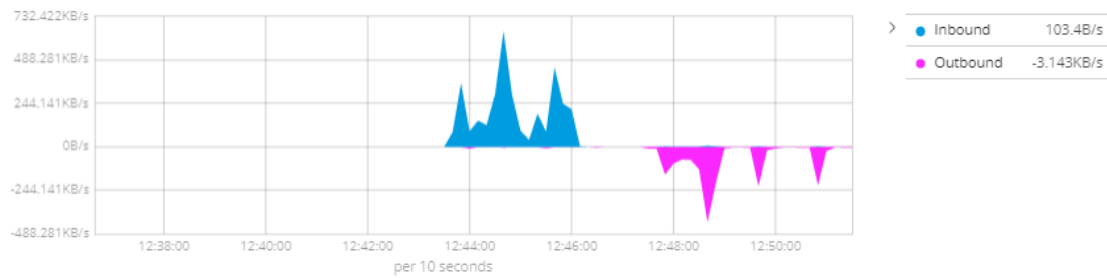


Ilustración 51: Tráfico de red (paquetes).

- Procesos del sistema que están consumiendo mayor cantidad de memoria.

Processes By Memory [Metricbeat System] ECS



Ilustración 52: Procesos por memoria más utilizados.

- Procesos del sistema que están consumiendo mayor cantidad de CPU.



Ilustración 53: Procesos por CPU más utilizados.

- Interfaces del servidor web con más tráfico entrante y saliente.



Ilustración 54: Interfaces del servidor web con más tráfico entrante.



Ilustración 55: Interfaces del servidor web con más tráfico saliente.

7.3.2. Datos de auditoría

- Cantidad de hosts monitorizados por Auditbeat. En este caso, solo el servidor web está siendo monitorizado:

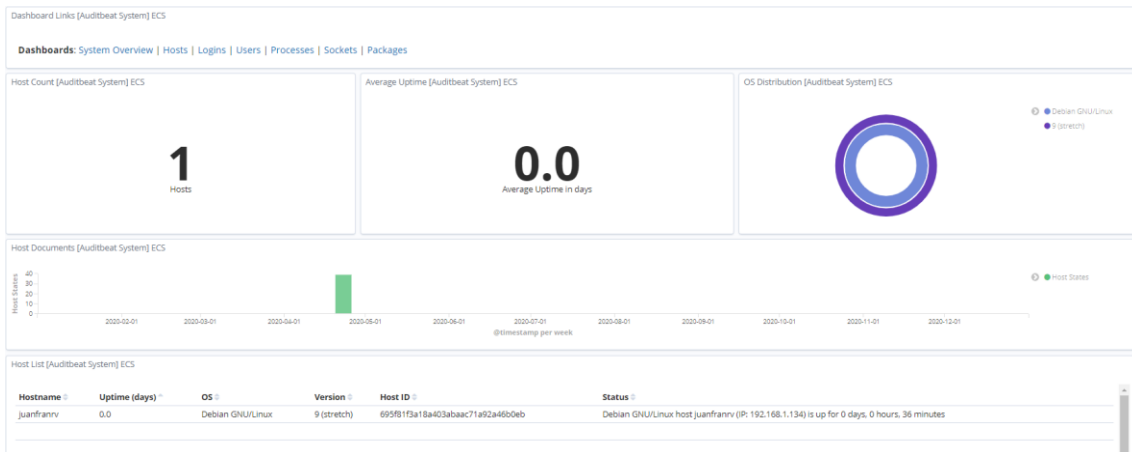


Ilustración 56: Cantidad de hosts monitorizados por Auditbeat.

➤ Información sobre inicios de sesión de usuarios en el servidor web:

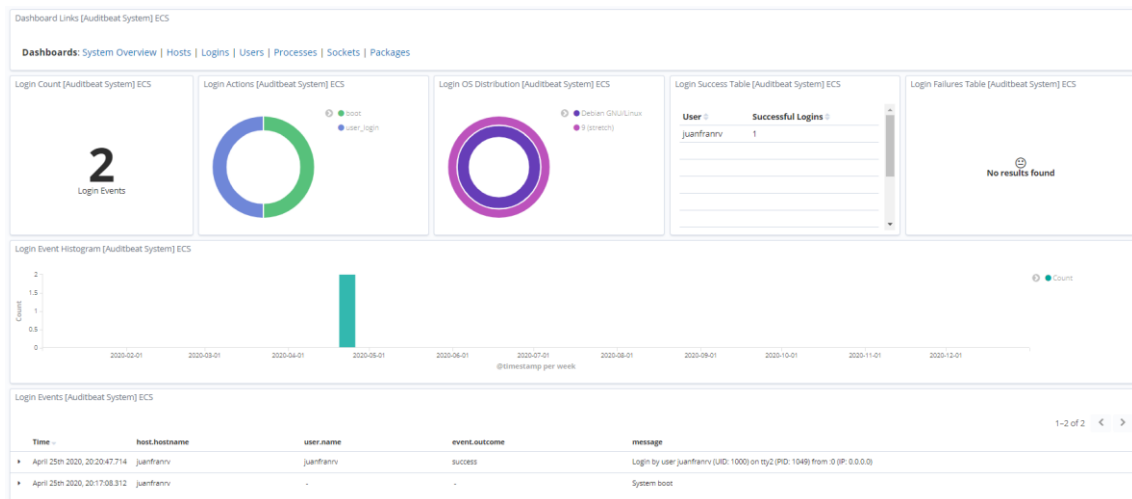


Ilustración 57: Información de inicios de sesión en el servidor web.

➤ Cantidad de usuarios existentes en el servidor web:

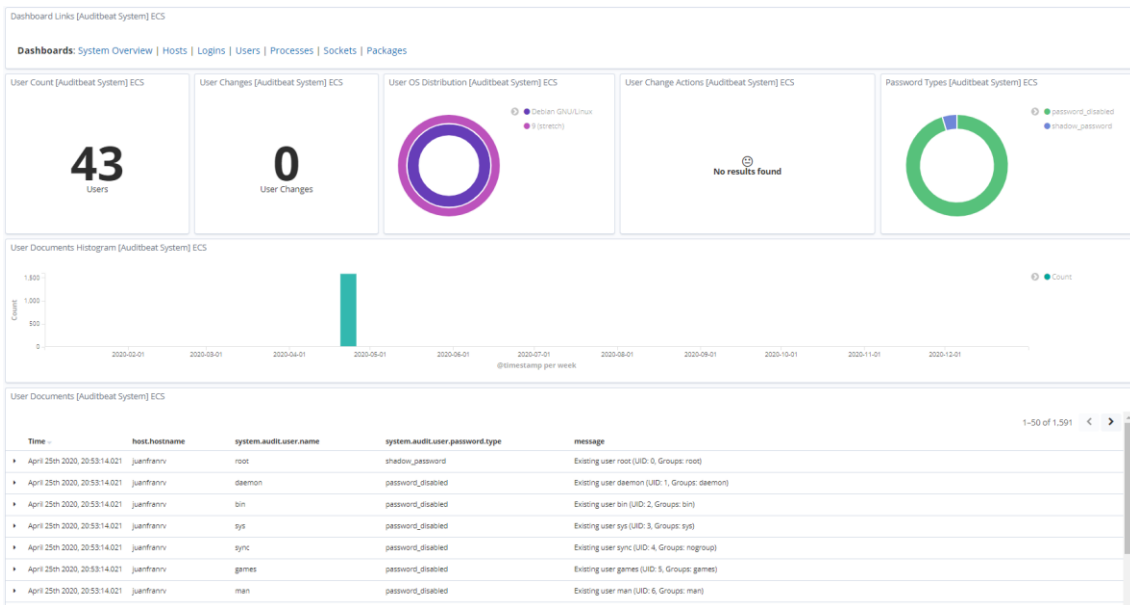


Ilustración 58: Usuarios registrados e interactuando en el servidor web.

- Información sobre el estado de los procesos que se están ejecutando en el servidor web:

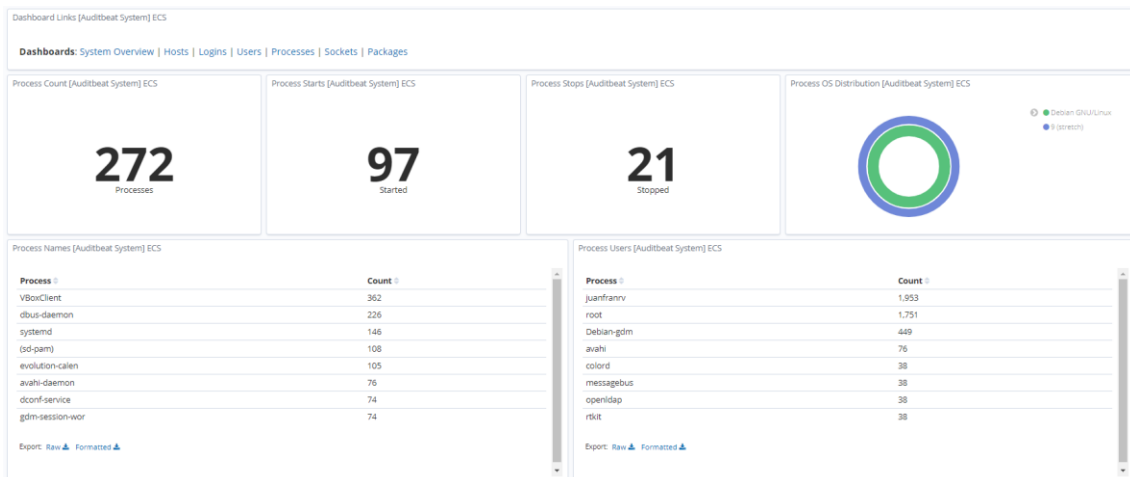


Ilustración 59: Dashboard que muestra información sobre los procesos del servidor web.

- Información acerca de diversos parámetros que proporcionan datos sobre la auditoría de los eventos Kernel que tienen lugar en el servidor web.

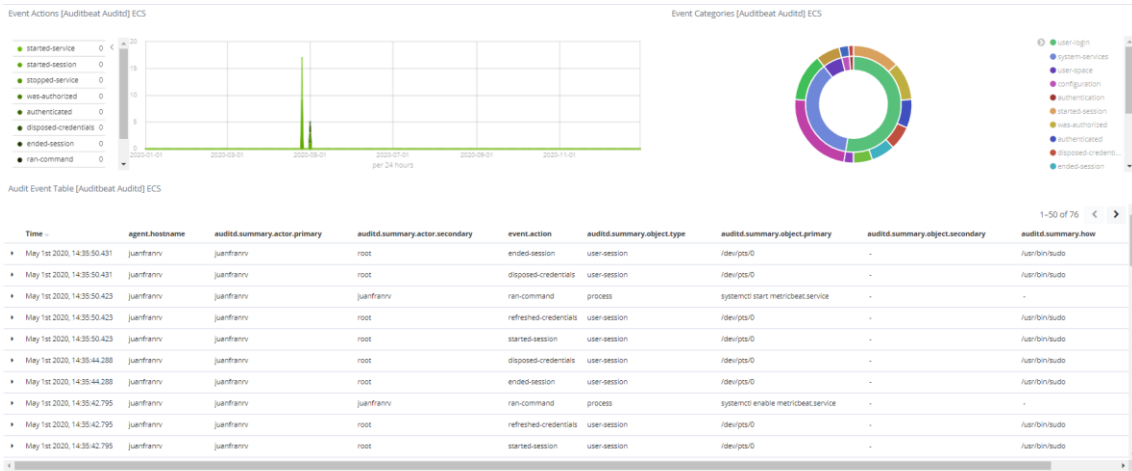


Ilustración 60: Dashboard de auditorías de eventos Kernel del servidor web.

➤ Información de integridad de ficheros que abarca todos los eventos de integridad que se han llevado a cabo:

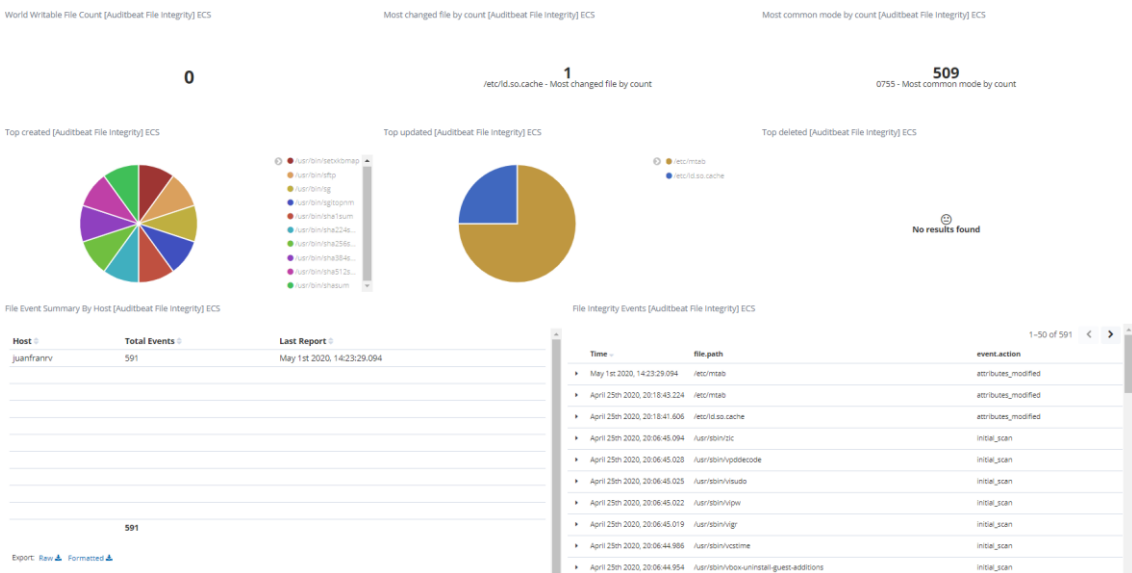


Ilustración 61: Dashboard de integridad de ficheros en servidor web.

7.4. Ficheros de logs Zeek IDS

Field	Type	Description
ts	time	Timestamp of hit
uid	string	Connection unique id
id	record	ID record with orig/resp host/port. See conn.log
fuid	string	The UID for a file associated with this hit, if any
file_mime_type	string	A mime type if the hit is related to a file
file_desc	string	Additional context for file, if available
seen.indicator	string	The intelligence indicator
seen.indicator_type	string	The type of data the indicator represents
seen.where	string	Where the data was discovered
sources	set	Sources which supplied data for this match

Ilustración 62: Campos del registro intel.log.

Field	Type	Description
ts	time	Timestamp
uid	string	Connection unique id
id	record	ID record with orig/resp host/port. See conn.log
fuid	string	File unique identifier
file_mime_type	string	Libmagic sniffed file type
file_desc	string	Additional context for file, if available
proto	transport_proto	Transport protocol
note	string	The type of the notice
msg	string	Human readable message for the notice
sub	string	Sub-message for the notice
src	addr	Source address
dst	addr	Destination address
p	port	Associated port, if any
n	count	Associated count or status code
peer_descr	string	Description for peer that raised this notice
actions	set	Actions applied to this notice
suppress_for	interval	Length of time dupes should be suppressed
dropped	bool	If the src IP was blocked

Ilustración 63: Campos del registro notice.log.

Field	Type	Description
ts	time	Timestamp
uid	string	Unique ID of Connection
id.orig_h	addr	Originating endpoint's IP address (AKA ORIG)
id.orig_p	port	Originating endpoint's TCP/UDP port (or ICMP code)
id.resp_h	addr	Responding endpoint's IP address (AKA RESP)
id.resp_p	port	Responding endpoint's TCP/UDP port (or ICMP code)
proto	transport_proto	Transport layer protocol of connection
service	string	Dynamically detected application protocol, if any
duration	interval	Time of last packet seen – time of first packet seen
orig_bytes	count	Originator payload bytes; from sequence numbers if TCP
resp_bytes	count	Responder payload bytes; from sequence numbers if TCP
conn_state	string	Connection state (see conn.log:conn_state table)
local_orig	bool	If conn originated locally T; if remotely F. If Site::local_nets empty, always unset.
missed_bytes	count	Number of missing bytes in content gaps
history	string	Connection state history (see conn.log:history table)
orig_pkts	count	Number of ORIG packets
orig_ip_bytes	count	Number of ORIG IP bytes (via IP total_length header field)
resp_pkts	count	Number of RESP packets
resp_ip_bytes	count	Number of RESP IP bytes (via IP total_length header field)
tunnel_parents	set	If tunneled, connection UID of encapsulating parent (s)
orig_cc	string	ORIG GeoIP Country Code
resp_cc	string	RESP GeoIP Country Code

Ilustración 64: Campos del registro conn.log.