



# UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO  
INGENIERÍA INFORMÁTICA

Securización y análisis de vectores de ataque en sistemas Linux mediante modelos de inteligencia artificial avanzados.

---

## Cyberlynx

**Autor**

Mario Daniel Castro Almenzar

**Directores**

Pablo García Sánchez

Rafael Alejandro Rodríguez Gómez



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

—  
Granada, a 24 de junio de 2024









Securización y análisis de vectores de ataque en  
sistemas Linux mediante modelos de inteligencia  
artificial avanzados.

---

# Cyberlynx

**Autor**

Mario Daniel Castro Almenzar

**Directores**

Pablo García Sánchez  
Rafael Alejandro Rodríguez Gómez



# Cyberlynx. Securización y análisis de vectores de ataque en sistemas Linux mediante modelos de inteligencia artificial avanzados

Mario Daniel Castro Almenzar

**Palabras clave:** ciberseguridad, análisis, logs, Linux, MITRE, SIEM, IA, clustering

## Resumen

En un contexto de evolución tecnológica constante y un incremento notable en la cantidad de ataques diarios, se vuelve imperativo contar con sistemas de seguridad robustos y eficientes. Este trabajo presenta un estudio sobre diversos modelos de inteligencia artificial para detectar vectores de ataque en sistemas operativos Linux. Para esta investigación, se adoptó un enfoque integral que involucra la simulación de ataques reales mediante el marco MITRE ATT&CK, y la creación de *datasets* de *logs*, tanto manuales como sintéticos, utilizando técnicas de *prompt-engineering*.

El desarrollo del proyecto se estructuró en varias fases clave. Inicialmente, se creó un *dataset* de *logs* a través de la simulación de ataques en un entorno controlado usando contenedores Docker y herramientas como el framework CALDERA. Posteriormente, se procedió a la recopilación de otros conjuntos de datos y a la generación de un *dataset* sintético. Estos *logs* se preprocesaron aplicando técnicas de vectorización de texto y reducción de dimensionalidad, junto con algoritmos de *clustering* como K-means, DBSCAN y *clustering* jerárquico, para identificar patrones y anomalías en los datos.

Los resultados muestran que los modelos de *clustering* jerárquico, en conjunto con técnicas de vectorización de texto como TF-IDF o *Count Vectorizer*, han mejorado significativamente la detección de incidentes de seguridad en sistemas Linux. Específicamente, se observó que los métodos de *clustering* jerárquico y aglomerativo proporcionan una mayor precisión para identificar vectores de ataque en comparación con otros métodos evaluados.

Este trabajo no solo evidencia la efectividad de los modelos de inteligencia artificial en la detección de ciberataques, sino que también compara estos resultados con otras investigaciones recientes en este ámbito. Se concluye que la incorporación de estas técnicas de IA en herramientas de ciberseguridad preexistentes, como los Sistemas de Gestión de Información y Eventos de Seguridad (SIEM), podría mejorar notablemente la capacidad de respuesta y mitigación de amenazas.



# Cyberlynx. Securization and analysis of attack vectors in Linux systems using advanced artificial intelligence models

Mario Daniel, Castro Almenzar

**Keywords:** cybersecurity, analysis, logs, Linux, MITRE, SIEM, AI, clustering

## Abstract

In a context of constant technological evolution and a significant increase in daily attacks, it becomes imperative to have robust and efficient security systems. This work presents a study on various artificial intelligence models for detecting attack vectors in Linux operating systems. For this research, a comprehensive approach was adopted involving the simulation of real attacks using the MITRE ATT&CK framework, and the creation of *datasets of logs*, both manual and synthetic, using *prompt-engineering* techniques.

The project development was structured in several key phases. Initially, a *dataset of logs* was created through the simulation of attacks in a controlled environment using Docker containers and tools such as the CALDERA framework. Subsequently, other data sets were compiled, and a synthetic *dataset* was generated. These *logs* were preprocessed using text vectorization and dimensionality reduction techniques, along with *clustering* algorithms such as K-means, DBSCAN, and *hierarchical clustering*, to identify patterns and anomalies in the data.

The results show that hierarchical *clustering* models, in conjunction with text vectorization techniques such as TF-IDF and *Count Vectorizer*, have significantly improved the detection of security incidents in Linux systems. Specifically, it was observed that hierarchical and agglomerative clustering methods provide greater accuracy in identifying attack vectors compared to other methods evaluated.

This work not only demonstrates the effectiveness of artificial intelligence models in detecting cyberattacks but also compares these results with other recent research in this field. It concludes that the incorporation of these AI techniques into pre-existing cybersecurity tools, such as Security Information and Event Management Systems (SIEM), could significantly enhance the capability for response and threat mitigation.



---

Yo, **Mario Daniel Castro Almenzar**, alumno del grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 75928205F, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Mario Daniel Castro Almenzar

Granada a 24 de junio de 2024



---

D. **Pablo García Sánchez**, Profesor del Área de Inteligencia Artificial del Departamento de Ingeniería de Computadores, Automática y Robótica de la Universidad de Granada.

D. **Rafael Alejandro Rodríguez Gómez**, Profesor del Área de Telemática del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

**Informan:**

Que el presente trabajo, titulado *Cyberlynx, Securización y análisis de vectores de ataque en sistemas Linux mediante modelos de inteligencia artificial avanzados*, ha sido realizado bajo su supervisión por **Mario Daniel Castro Almenzar**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 24 de junio de 2024.

**Los directores:**

Pablo García Sánchez

Rafael Alejandro Rodríguez Gómez



# Agradecimientos

Me gustaría empezar agradeciendo a la Escuela por todos estos años, en los que además de formarme en aquello que me gusta, he conocido a personas maravillosas, tanto compañeros que han acabado siendo grandes amigos como profesores con los que más allá de las clases he compartido mi pasión por la Informática, el *hacking* y los videojuegos.

Quiero agradecer especialmente a Pablo y a Rafa, que a lo largo de estos meses han procurado guiarme para que fuese capaz de llegar hasta aquí, y han depositado la confianza en que pudiera entregar algo de lo que estuviese suficientemente orgulloso.

A nivel más personal, quiero agradecer a mis padres por haberme brindado la oportunidad de estudiar aquello que quería y su apoyo incondicional. Y por último, a mi abuela y a mi novia, que sin duda a través de sus consejos y mensajes de ánimo, día tras día, han conseguido que no tirara la toalla frente a todas las adversidades que se han ido planteando. Espero corresponderos al menos una pequeña parte de todo lo que hacéis por mí.

*“Rodearte de gente mejor que  
tú puede ser tu mayor virtud.”*

**Bernardo Quintero**

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación y contexto . . . . .	1
1.1.1. El problema de la seguridad digital . . . . .	1
1.1.2. La inteligencia artificial como solución . . . . .	2
1.2. Objetivos . . . . .	3
1.3. Estructura de la memoria . . . . .	4
<b>2. Planificación</b>	<b>5</b>
2.1. Temporización . . . . .	5
2.2. Presupuesto . . . . .	8
<b>3. Estado del Arte</b>	<b>13</b>
3.1. Seguridad en sistemas operativos Linux . . . . .	13
3.1.1. <i>Hardening</i> del sistema . . . . .	15
3.1.2. Generación y monitorización de logs . . . . .	17
3.2. El marco de MITRE ATT&CK . . . . .	21
3.2.1. Tácticas . . . . .	23
3.2.2. Técnicas . . . . .	23
3.2.3. Procedimientos . . . . .	24
3.3. Modelos de IA para la detección de vectores de ataque . . . . .	26
3.3.1. Principales <i>Datasets</i> de logs . . . . .	27
3.4. SIEM . . . . .	30
3.4.1. Principales SIEM en el escenario actual . . . . .	31
3.4.2. Nuevas tendencias y mejoras en SIEM . . . . .	33
<b>4. Metodología de trabajo</b>	<b>35</b>
4.1. Complejidad del problema a resolver . . . . .	35
4.2. Algoritmos de <i>machine learning</i> escogidos . . . . .	38
4.2.1. K-means . . . . .	40
4.2.2. DBSCAN . . . . .	43
4.2.3. Clustering Jerárquico . . . . .	45
4.3. <i>Set-up</i> experimental: tecnologías y herramientas . . . . .	47
4.3.1. Herramientas para la simulación de ataques . . . . .	47
4.3.2. Virtualización de las máquinas víctima . . . . .	49
4.3.3. Librerías de Python utilizadas . . . . .	50
<b>5. Desarrollo e implementación</b>	<b>55</b>
5.1. Simulación de ataques y generación de logs . . . . .	55
5.1.1. Dataset Manual . . . . .	55
5.1.2. Dataset Sintético . . . . .	61
5.2. Preparación de los <i>datasets</i> . . . . .	63

5.3. Implementación del modelo . . . . .	69
5.3.1. Técnicas utilizadas . . . . .	69
5.3.2. Desarrollo del <i>clustering</i> . . . . .	72
5.4. Selección de métricas de evaluación . . . . .	76
<b>6. Análisis de resultados</b>	<b>79</b>
6.1. Evaluación de la efectividad de los modelos . . . . .	79
6.2. Estudios previos para la detección de amenazas . . . . .	89
6.3. Sinergias e integración con sistemas SIEM . . . . .	93
6.3.1. Beneficios de la integración . . . . .	93
6.3.2. Desafíos y soluciones . . . . .	94
<b>7. Conclusiones y trabajo futuro</b>	<b>95</b>
7.1. Resumen de los logros obtenidos . . . . .	95
7.2. La importancia del Software Libre . . . . .	96
7.3. Líneas de investigación y desarrollo futuras (I+D) . . . . .	97
<b>Bibliografía</b>	<b>105</b>
<b>A. Formación Inicial</b>	<b>107</b>
<b>B. Repositorio del proyecto</b>	<b>111</b>
<b>C. Instalación y configuración de CALDERA</b>	<b>115</b>
<b>Glosario</b>	<b>121</b>

# Índice de figuras

2.1. Diagrama de Gantt . . . . .	7
3.1. Comparativa entre sistemas operativos usados por profesionales [1] . . . . .	13
3.2. Contenido del registro de <i>/var/log</i> . . . . .	18
3.3. Contenido del registro de <i>auth.log</i> . . . . .	19
3.4. Contenido del registro de <i>dpkg.log</i> . . . . .	19
3.5. Contenido de los logs de <i>dmesg</i> . . . . .	20
3.6. Contenido de los logs de <i>journalctl</i> . . . . .	20
3.7. Matriz de técnicas y tácticas de MITRE ATT&CK para enterprise [2] . . . . .	21
3.8. Listado de técnicas de la táctica de <i>Reconnaissance</i> [2] . . . . .	24
3.9. Sub-técnicas de <i>Active Scanning</i> [2] . . . . .	24
3.10. Información referente al procedimiento asignado <i>Scanning IP Blocks</i> [2] . . . . .	24
3.11. Ejemplos de mitigación y detección frente a <i>Scanning IP Blocks</i> [2] . . . . .	25
4.1. Sintaxis de un <i>log</i> de Linux . . . . .	35
4.2. Sintaxis del <i>message</i> de un <i>log</i> de Linux . . . . .	35
4.3. Estructura del <i>timestamp</i> de un <i>log</i> de Linux . . . . .	36
4.4. Sintaxis de un <i>log</i> de BGL . . . . .	37
4.5. Estructura del <i>timestamp</i> de un <i>log</i> de BGL . . . . .	37
4.6. Sintaxis de un <i>log</i> de HDFS . . . . .	37
4.7. Estructura del <i>timestamp</i> de un <i>log</i> de HDFS . . . . .	37
4.8. Ejemplo de secuencia de <i>logs</i> generados por fuerza bruta a SSH . . . . .	38
4.9. Tipos de agrupamiento en técnicas de <i>clustering</i> . . . . .	39
4.10. Método del codo para la elección del parámetro <i>k</i> . . . . .	41
4.11. Ejemplo de clasificación de puntos de algoritmo de <i>clustering</i> DBSCAN . . . . .	44
4.12. Ejemplo de un dendrograma para AHC . . . . .	46
4.13. Infraestructura del framework CALDERA [3] . . . . .	48
4.14. Comparativa estructural de Máquinas Virtuales vs Contenedores . . . . .	50
4.15. Diagrama estructural del Trabajo de Fin de Grado . . . . .	53
5.1. Distribución de Linux utilizada para la Simulación de ataques . . . . .	55
5.2. Arquitectura del entorno de simulación de ataques . . . . .	56
5.3. Ejemplo de ejecución de una operación de Discovery con CALDERA . . . . .	59
5.4. Monitorización de <i>logs</i> generados en máquinas víctima por operaciones . . . . .	60
5.5. Ataque por diccionario a servicio SSH mediante Hydra . . . . .	60
5.6. Ataque con Metasploit para enumeración de usuarios en un servidor SSH . . . . .	61
5.7. Resultado del <i>dataset</i> generado de forma sintética. . . . .	62
5.8. Ejemplo de <i>logs</i> residuales del conjunto de datos . . . . .	63
5.9. Estructura del <i>dataset</i> tras fase de preprocesamiento . . . . .	68
5.10. Aplicación del Algoritmo del Codo sobre <i>dataset</i> generado manualmente . . . . .	73
5.11. Cálculo del valor <i>k</i> óptimo mediante Silhouette Score . . . . .	73

6.1. Clustering con K-Means utilizando TF-IDF y t-SNE . . . . .	80
6.2. Clustering con AHC utilizando <i>Count Vectorizer</i> y t-SNE . . . . .	81
6.3. Resultados obtenidos en <i>dataset</i> manual según índice de Calinski-Harabasz	81
6.4. Cálculo del método del codo para el <i>dataset</i> <i>02_linux-logs-2k.csv</i> . . . . .	82
6.5. Clustering de <i>Linux_2k</i> con K-means utilizando TF-IDF y t-SNE . . . . .	82
6.6. Clustering de <i>Linux_2k</i> con K-means utilizando <i>Count-Vectorizer</i> y t-SNE . .	82
6.7. Clustering de HDFS con AHC utilizando TF-IDF y t-SNE . . . . .	83
6.8. Clustering de BGL con K-means utilizando TF-IDF y UMAP . . . . .	84
6.9. Clustering de AHC en <i>dataset</i> sintético con <i>Count Vectorizer</i> y t-SNE . . .	85
6.10. Clustering de HC en <i>06_auth-logs-20k.csv</i> con <i>Count Vectorizer</i> y t-SNE	86
6.11. Resultados del <i>clustering</i> obtenidos en el <i>Dataset</i> manual . . . . .	87
6.12. Diagrama de la arquitectura propuesta por Tharunika et. al [4] . . . . .	91
A.1. Módulos de TryHackMe sobre Fundamentos de Seguridad en Linux [5] . . .	109
A.2. Módulos de TryHackMe sobre Logs y Monitoreo de sistemas Linux I . . . .	109
A.3. Módulos de TryHackMe sobre Logs y Monitoreo de sistemas Linux II . . . .	110
C.1. Clonación del repositorio de CALDERA en la máquina atacante . . . . .	115
C.2. Instalación de dependencias utilizadas por CALDERA . . . . .	116
C.3. Ejecución del comando para levantar el servicio de CALDERA . . . . .	116
C.4. Ejecución del comando para levantar el servicio de CALDERA (II) . . . .	117
C.5. Fichero con credenciales por defecto de CALDERA . . . . .	117
C.6. Pestaña de Inicio de sesión del <i>framework</i> . . . . .	117
C.7. Dashboard de la interfaz web del <i>framework</i> . . . . .	118
C.8. Proceso de configuración de un agente de CALDERA . . . . .	119
C.9. Selección del agente Sandcat para la infección de las máquinas víctima . .	119
C.10. Ejecución del <i>payload</i> para generar un agente conectado a la máquina . .	120
C.11. Verificación del estado de los agentes configurados . . . . .	120
C.12. Ejemplo de filtrado en el panel de navegación de habilidades de CALDERA	121
C.13. Configuración de un perfil de adversario en CALDERA . . . . .	121
C.14. Configuración de una operación de caldera a partir de un perfil adversario .	121

# Índice de tablas

1.1. Relación del TFG con Objetivos de Desarrollo Sostenible (ODS) . . . . .	3
2.1. Presupuesto total del proyecto . . . . .	8
2.2. Presupuesto hardware del Proyecto . . . . .	8
2.3. Recursos software utilizados en el proyecto . . . . .	9
2.4. Coste total de Recursos Humanos asociados a tutorización . . . . .	11
3.1. Estructura del sistema de archivos de SOs basados en Linux . . . . .	14
3.2. Estructura de permisos en Sistemas Linux [6] . . . . .	15
3.3. Tipos y propietarios de archivos en Sistemas Linux [6] . . . . .	15
3.4. Niveles de prioridad de syslog . . . . .	17
3.5. <i>Facilities</i> de Syslog . . . . .	18
3.6. Identificadores del marco de <i>MITRE ATT&amp;CK</i> . . . . .	22
3.7. Matrices de MITRE ATT&CK . . . . .	22
3.8. Conjunto de tácticas ofensivas del marco de <i>MITRE ATT&amp;CK</i> . . . . .	23
3.9. Conjunto de <i>Datasets</i> de logs para el análisis impulsado por IA . . . . .	28
3.10. Herramientas software de seguridad . . . . .	30
3.11. Herramientas integrables o complementarias al SIEM . . . . .	30
3.12. Principales SIEM en escenario actual según Cuadrante Mágico de Gartner [7]	32
4.1. Descripción del formato de <i>timestamp</i> según el estándar ISO 8601 [8] . . . . .	36
4.2. Factores de un modelo de <i>clustering</i> . . . . .	39
4.3. Parámetros y atributos del algoritmo K-Means en sklearn . . . . .	42
4.4. Clasificación de puntos en DBSCAN . . . . .	44
4.5. Pasos del algoritmo de Clustering Jerárquico Aglomerativo . . . . .	45
4.6. Terminología utilizada por el <i>framework</i> de CALDERA . . . . .	48
4.7. Comparación de tecnologías de virtualización para la simulación de ataques . . . . .	49
5.1. Máquinas utilizadas para la simulación de ataques y recolección de <i>logs</i> . . . . .	56
5.2. Técnicas de balanceo de clases para grandes conjuntos de datos . . . . .	64
5.3. Estructura del <i>dataset</i> preprocesado . . . . .	66
5.4. Listado de conjuntos de datos preprocesados . . . . .	68
5.5. Intervalos de oscilación para las métricas de evaluación del <i>clustering</i> . . . . .	78
6.1. Métricas obtenidas en los distintos conjuntos de datos . . . . .	88
6.2. Resultados obtenidos por Datta et al. [9] empleando algoritmos de <i>clustering</i> para la detección de amenazas . . . . .	92
A.1. Módulos del Curso de Deep Learning & Neural Networks, Andrew Ng. [10]	107
A.2. Módulos del Curso de NLP de Hugging Face [11] . . . . .	108
A.3. Máquinas de <i>Hacking</i> Ético resueltas por Plataforma . . . . .	109

---

B.1. Listado de <i>Milestones</i> utilizados para agrupar objetivos a cumplir . . . . .	111
B.2. Estructura jerárquica del repositorio Cyberlynx . . . . .	112
C.1. Descripción de las secciones principales en la interfaz de CALDERA . . . . .	118
C.2. Formato de la tabla de un agente de CALDERA . . . . .	120

# Capítulo 1

## Introducción

### 1.1. Motivación y contexto

#### 1.1.1. El problema de la seguridad digital

En la actualidad, las personas viven continuamente conectadas. A veces, sin darse cuenta, están haciendo uso de dispositivos electrónicos manejados a un alto nivel, confiando ciegamente en la implementación que hay detrás de esa gran capa de abstracción. La tecnología se ha vuelto ubicua, presente en cada aspecto de nuestras vidas, y del mismo modo, la amplia mayoría de las personas confía ciegamente en su seguridad y en la de todas las herramientas que utilizan en su día a día, o quizás no consideran que sea algo suficientemente importante como para estar alerta.

Y en cierto modo tienen razón, nuestra labor como ingenieros informáticos es precisamente poder garantizar a los usuarios que todos sus bienes digitales estén siempre disponibles y cumplan, al menos, con las principales primitivas de seguridad descritas por la triada CIA (*Confidencialidad, Integridad y Disponibilidad*).

Con el paso de los años, empresas y entidades gubernamentales han empezado a aplicar estándares de certificación que les permitan tener ciertas garantías. En el caso de España, esta acreditación se lleva a cabo por parte del CCN (*Centro Criptológico Nacional*) [12] a través del catálogo CPSTIC (*Catálogo de Productos y Servicios de Seguridad de las TIC del CCN*) [13]. Además, cada vez más empresas invierten en personal con formación en el ámbito de la seguridad y, en función del tamaño de la empresa u organismo, incluso en unidades para la gestión y análisis de riesgos a través de SOCs (*Security Operations Center*).

Sin embargo, dentro del campo de la seguridad informática es bien sabido que la robustez de cualquier sistema está definida por la más débil de sus capas. Por consiguiente, es de vital importancia asegurar cada eslabón y adicionalmente llevar a cabo un seguimiento continuo de la eficacia de estas medidas.

No es sencillo encontrar una entidad que no haya sido nunca atacada. El pasado 2023 [14], se registraron más de 28000 CVEs (*Common Vulnerabilities and Exposures*) [15], y aunque esto resulte alarmante, la realidad es que la generalidad de problemas vienen de vulnerabilidades con una antigüedad significativa y que no

han sido solucionadas por la inacción de estas entidades. Es por ello que urge el desarrollo de mecanismos que agilicen la securización de sistemas, minimizando el esfuerzo humano, de modo que este pueda atender a otras cuestiones que sí que requieran de su capacidad de decisión.

### 1.1.2. La inteligencia artificial como solución

En los últimos años, la IA (Inteligencia Artificial) ha emergido como un poderoso aliado en la lucha contra las amenazas de seguridad. Tal y como indica la ENISA (*European Network and Information Security Agency*) [16], uno de los principales beneficios es su habilidad para automatizar la detección de amenazas.

Estos sistemas pueden examinar continuamente los flujos de datos en busca de actividades sospechosas o anómalas, algo que sería imposible o impráctico para los analistas debido al volumen y velocidad de los datos. Además, los modelos tienen la capacidad de aprender de las interacciones pasadas y mejorar continuamente su precisión y efectividad en la detección de amenazas.

Todo apunta a que la IA jugará también un papel crucial en el desarrollo de sistemas de respuesta a incidentes, analizando rápidamente la causa raíz de una brecha de seguridad y sugiriendo o incluso implementando medidas correctivas para mitigar el daño. Esto es especialmente valioso en situaciones donde el tiempo es crítico, como en el caso de brechas de datos masivas o uno de los más comunes: ataques de *ransomware* [17].

Es importante recalcar que, al igual que otras propuestas, esta también tiene sus limitaciones, y tras la reciente aparición de los LLMs (*Large Language Models*), vivimos en una etapa en la que se está generando una burbuja de expectativas y desinformación acerca de la misma, por lo que es necesario tener una conciencia realista del alcance actual y a corto plazo de su funcionalidad. Un claro ejemplo de ello es la aparición de LLMs optimizados mediante técnicas de *fine-tuning* que aseguran ser capaces de realizar ejercicios de *penetration testing* [18] de forma autónoma sobre escenarios realistas.

La colaboración entre sistemas con modelos de IA integrados y expertos humanos puede proporcionar una defensa más robusta frente a las amenazas. Mientras que la IA maneja tareas de análisis de datos, los profesionales de la seguridad pueden concentrarse en la estrategia de seguridad y la toma de decisiones críticas, aprovechando los conocimientos proporcionados por la IA para actuar de manera más informada y efectiva.

Tras este breve análisis de las ventajas que aporta, resulta acertado afirmar que la integración de la inteligencia artificial en la seguridad informática no solo potencia las capacidades de detección y respuesta de las organizaciones, sino que también transforma la naturaleza del monitoreo de seguridad, haciéndolo más proactivo, predictivo y eficaz en un mundo donde los cibercriminales están constantemente evolucionando debido al crecimiento exponencial de sus recursos económicos y, en consecuencia, logísticos.

## 1.2. Objetivos

El propósito principal del trabajo es la securización y análisis de vectores de ataque en sistemas Linux mediante el uso de modelos avanzados de inteligencia artificial. A partir de este objetivo general, se establecen los siguientes objetivos específicos:

- **OB1.** Investigar la usabilidad y morfología de los *logfiles* y cómo pueden procesarse para ser interpretados.
- **OB2.** Crear un *dataset* a partir de los *logs* generados con la simulación de ataques en sistemas Linux utilizando el marco de técnicas adversarias definidas por MITRE ATT&CK.
- **OB3.** Implementar un conjunto de algoritmos de *clustering* para identificar y clasificar vectores de ataque en sistemas Linux basándose en patrones de comportamiento detectados en los *datasets* generados y otros equivalentes a estos.
- **OB4.** Validar los resultados obtenidos a través de una serie de métricas que permitan medir con precisión la efectividad del modelo con respecto al agrupamiento y detección de vectores de ataque, y comparar estos resultados con las herramientas de seguridad existentes.
- **OB5.** Demostrar la escalabilidad y la capacidad de integración del modelo con sistemas de gestión de información y eventos de seguridad (SIEM), explorando sinergias potenciales y la mejora en la respuesta a incidentes a través de la automatización y el análisis avanzado.

## Objetivos de Desarrollo Sostenible

En el desarrollo de este Trabajo de Fin de Grado (TFG) se busca contribuir a varios Objetivos de Desarrollo Sostenible (ODS) establecidos por la ONU, en línea con el compromiso global con la Agenda 2030.

ODS	Relación	Justificación
ODS 4. Educación de calidad	Medio	El proyecto puede contribuir a la educación de calidad al proporcionar conocimientos avanzados y técnicas en el ámbito de la ciberseguridad y la inteligencia artificial, formando a futuros profesionales en estas áreas cruciales.
ODS 8. Trabajo decente y crecimiento económico	Medio	La mejora de la seguridad digital puede favorecer un entorno económico más seguro y estable, promoviendo así el crecimiento económico y la creación de empleos en el sector tecnológico.
ODS 9. Industria, innovación e infraestructuras	Alto	La investigación y desarrollo de nuevas tecnologías de seguridad digital y su implementación en sistemas Linux fomenta la innovación y mejora las infraestructuras tecnológicas, lo cual es crucial para el desarrollo industrial.
ODS 16. Paz, justicia e instituciones sólidas	Alto	La seguridad tecnológica es fundamental para mantener la paz y la justicia en el ciberspacio. En esta línea, el proyecto puede contribuir a fortalecer las instituciones encargadas de la seguridad y protección de datos.

Tabla 1.1: Relación del TFG con Objetivos de Desarrollo Sostenible (ODS)

En la Tabla 1.1 se presenta la relación de este proyecto con algunos de estos objetivos, así como una breve justificación de cómo se alinea cada uno con los mismos.

### 1.3. Estructura de la memoria

La estructura de la memoria de este Trabajo de Fin de Grado clasificado como de Investigación se organiza en siete capítulos principales y tres apéndices anexos que complementan la documentación. A continuación, se describe brevemente el contenido de cada uno de ellos:

En primer lugar, el capítulo de **Introducción** establece el contexto sobre el cual se desarrolla este Trabajo, desde el gran problema que presenta actualmente la seguridad digital, hasta las soluciones que plantean los avances emergentes en Inteligencia Artificial. Por último, se definen los objetivos a cumplir y se indica qué estructura tendrá la memoria. A continuación, en el capítulo de **Planificación** se realiza un análisis detallado de la temporización del proyecto y del presupuesto asignado.

En tercer lugar, el capítulo de **Estado del arte** discute sobre la seguridad en sistemas operativos Linux, técnicas adversarias de MITRE ATT&CK, modelos de inteligencia artificial para la detección de vectores de ataque y herramientas de gestión de información y eventos de seguridad (SIEM). Seguidamente, en **Metodología de trabajo** se explora el ámbito del problema, se indica cómo se va a diseñar la implementación y se describen las tecnologías utilizadas para ello. En el capítulo de **Desarrollo e implementación** se cubre desde la simulación de ataques y generación de logs, hasta la implementación de modelos capaces de interpretar estos y hacer predicciones para identificar potenciales patrones de ataque.

El capítulo sexto, titulado **Análisis de resultados**, evalúa el modelo mediante distintas métricas seleccionadas, se estudian casos específicos, se compara con herramientas existentes y se explora la integración con sistemas SIEM. Finalmente, en el capítulo de **Conclusiones** se resume los logros obtenidos, el impacto del proyecto en la mejora de la seguridad de sistemas Linux y se discuten futuras investigaciones y posibles mejoras y ampliaciones para la mejorar la precisión de los resultados.

En cuanto a los apéndices que complementan este Trabajo, se incluyen:

**A. Formación Inicial:** Detalles sobre la capacitación y las habilidades técnicas iniciales requeridas antes de comenzar el desarrollo, desde cursos de ML, DL y NLP, hasta la realización de máquinas de tipo CTF para ampliar los conocimientos en ciberseguridad.

**B. Repositorio del Proyecto:** Guía sobre la estructura jerárquica de ficheros y carpetas del repositorio de GitHub donde se encuentra alojada la implementación del proyecto.

**C. Instalación y configuración de CALDERA:** Guía práctica que detalla los pasos para realizar la instalación y configuración del *framework* CALDERA para la simulación de ataques sobre sistemas Linux.

# Capítulo 2

## Planificación

A lo largo de este capítulo se describirán tanto el presupuesto requerido para el desarrollo del proyecto en base a los recursos hardware, software y humanos utilizados y se mostrará la temporización de las tareas en las que se ha fraccionado para su consecución.

### 2.1. Temporización

El desarrollo del proyecto se ha estructurado en distintas fases de modo que se lleve a cabo un proceso iterativo. A continuación se ilustra el listado de tareas, así como su conexión con los objetivos establecidos anteriormente (1.2).

- **T1** - Elección del tema a desarrollar y línea de aprendizaje (10h)
- **T2** - Formación inicial (A):
  - **T2.1** - Curso Redes Neuronales y Aprendizaje profundo Coursera (30h)
  - **T2.2** - Curso NLP Hugging Face (30h)
  - **T2.3** - Realización de *walkthroughs* y máquinas CTF (120h)
- **T3** - Investigación sobre *logs*, MITRE ATT&CK y SIEMs (30h) - **OBJ1**.
- **T4** - Implementación (I): Simulación de ataques y creación de *dataset*- **OBJ2**.
  - **T4.1** - Configuración del entorno (10h)
  - **T4.2** - Elección de herramientas de simulación (5h)
  - **T4.3** - Simulación de Ataques y Generación del Dataset (10h)
- **T5** - Implementación (II): Preprocesado de *datasets* (15h) - **OBJ2**.
  - **T5.1** - *Dataset* propio
  - **T5.2** - *Dataset* Linux\_2k
  - **T5.3** - *Dataset* HDFS
  - **T5.4** - *Dataset* BGL
  - **T5.5** - *Dataset* sintético

- **T6** - Implementación (III): Desarrollo del *clustering* (25h) - **OBJ3.**
  - **T6.1** - K-means
  - **T6.2** - DBSCAN
  - **T6.3** - Hierarchical Clustering
- **T7** - Análisis de Resultados obtenidos (5h) - **OBJ4.**
  - **T7.1** - Silhouette score
  - **T7.2** - Índice de Davies-Bouldin
  - **T7.3** - Índice de Cainski-Harabasz
- **T8** - Redacción de la memoria (95h) - **OBJ5.**

En función de las tareas indicadas, el tiempo estimado de esfuerzo para completar el proyecto es de **385h** aproximadamente.

### Diagrama de Gantt

Para cumplimentar todas y cada una de las tareas, se ha desarrollado el Diagrama de Gantt ilustrado en la Figura 2.1 entre los meses de septiembre de 2023 y junio de 2024, de modo que se alineen los esfuerzos en base al tiempo disponible y las tareas se fueran realizando periódicamente. Para ello, se han seguido las pautas indicadas por Raj Jain en el capítulo 5 de su libro [19]. El diagrama ha sido realizado a través de Google Sheets por cuestiones de personalización, aunque existen herramientas automatizadas como GanttPRO [20] que facilitan su realización.

En dicho diagrama se pueden observar las distintas reuniones llevadas a cabo con los tutores para llevar un seguimiento de los pasos realizados y planificar las siguientes tareas a realizar, así como proponer mejoras y cambios en la implementación.

Se ha hecho una división por bloques semanales, de modo que cada columna representa aproximadamente una semana del curso académico. En base al resultado obtenido, puede concluirse que el proyecto se ha realizado paulatinamente, aunque se puede observar un incremento considerable de la carga de trabajo durante los últimos dos meses.



Figura 2.1: Diagrama de Gantt

## 2.2. Presupuesto

Para poder llevar a cabo el proyecto ha sido necesario realizar la planificación de los recursos a utilizar. En primer lugar establecer los recursos hardware, es decir, los dispositivos utilizados para el desarrollo de la memoria y de la implementación, así como los servicios de luz e internet. En segundo lugar, definir los recursos software: servicios en la nube, herramientas software, acceso a libros y artículos, etc. Finalmente, detallar el coste orientativo en recursos humanos en base al salario medio actual para un investigador en las áreas de Ciberseguridad e Inteligencia Artificial en función del tiempo invertido.

Tipo de Recurso	Coste Total
Recursos Hardware	4.454,50 €
Amortización Hardware	411,75 €
Recursos Software	0 €
Recursos Humanos	19.545,80 €
<b>Coste total</b>	<b>24.412,05 €</b>

Tabla 2.1: Presupuesto total del proyecto

A continuación se desglosan cada uno de los elementos de la tabla indicando cómo se han realizado los cálculos.

### Recursos Hardware

En la siguiente tabla se detallan los recursos hardware utilizados, entre los que se incluyen un ordenador portátil y un ordenador de sobremesa, además del coste mensual para los servicios de luz e internet.

Descripción	Uds.	Importe por unidad	Total
Ordenador Portátil MSI Prestige 15 i7-1185G7 32GB RAM 1TB SSD GTX 1650	1	1.561,95 €	1.561,95 €
Ordenador Sobremesa Intel i7-6700K 4.0Ghz 16GB RAM 1TB SSD 1TB NVMe GTX 1080Ti 11GB GDDR5X	1	2.390,60 €	2.390,60 €
Servicio de Internet (mensual)	5	63,55 €	317,75 €
Servicio de Luz (mensual)	5	36,84 €	184,20 €
<b>Coste total</b>			<b>4.454,50 €</b>

Tabla 2.2: Presupuesto hardware del Proyecto

En base a un escenario realista, es necesario tener en cuenta la existencia de los costes de amortización. Según la normativa vigente, el porcentaje límite para la amortización de equipos de "Tratamiento de la Información y Sistemas y Programas Informáticos" se distribuye en un período de cuatro años [21]. Considerando que el proyecto se ha desarrollado en un plazo aproximado de 5 meses, los costes de amortización serían de aproximadamente 411,75 €.

Para ello se han llevado a cabo los siguientes cálculos:

1. Determinar el costo total de los equipos a amortizar:

$$\text{Costo total de los equipos} = \text{Costo PC portátil} + \text{Costo PC sobremesa}$$

$$\text{Costo total de los equipos} = 1.561,95 + 2.390,60 = 3.952,55 \text{ €}$$

2. Calcular la amortización anual:

$$\text{Amortización anual} = \frac{\text{Costo total de los equipos}}{\text{Período de amortización en años}}$$

$$\text{Amortización anual} = \frac{3.952,55}{4} = 988,14 \text{ €}$$

3. Calcular la amortización mensual:

$$\text{Amortización mensual} = \frac{\text{Amortización anual}}{12}$$

$$\text{Amortización mensual} = \frac{988,14}{12} = 82,35 \text{ €}$$

4. Calcular la amortización para 5 meses:

$$\text{Amortización para 5 meses} = \text{Amortización mensual} \times 5$$

$$\text{Amortización para 5 meses} = 82,35 \times 5 = 411,75 \text{ €}$$

## Recursos Software

En el caso de los recursos software, se ha utilizado únicamente servicios, herramientas y documentación de libre acceso. Por lo que el costo ha sido de 0€. Se ha hecho uso de los siguientes elementos:

Recurso	Detalles
Recursos Bibliográficos	Accedidos a través de la cuenta institucional de la Universidad de Granada. Algunos ejemplos son Google Scholar, Science Direct y Scopus.
Recursos de Internet	Accedidos a través de motores de búsqueda. Algunos ejemplos de los utilizados son Google, Firefox y DuckDuckGo.
Conjuntos de datos	Accedidos a través de sitios web como GitHub, SecRepo y Kaggle.
Herramientas de desarrollo	Plataformas de documentación como Overleaf e implementación como Google Colab

Tabla 2.3: Recursos software utilizados en el proyecto

Los sistemas operativos utilizados para el desarrollo del proyecto son software libre, y están detallados en la primera sección del capítulo cinco (5.1).

### Recursos Humanos

Para calcular el coste en recursos humanos, se ha considerado el salario medio de un investigador en Ciberseguridad e Inteligencia Artificial, que es de aproximadamente 30.000€ brutos anuales [22]. El proyecto ha requerido un total de 385 horas de trabajo distribuidas en 5 meses.

1. Determinar el salario mensual:

$$\text{Salario mensual} = \frac{\text{Salario anual}}{12}$$

$$\text{Salario mensual} = \frac{30.000}{12} = 2.500 \text{ €}$$

2. Calcular el salario por hora (considerando una jornada laboral de 40 horas semanales):

$$\text{Salario por hora} = \frac{\text{Salario mensual}}{160} \quad (160 \text{ horas/mes})$$

$$\text{Salario por hora} = \frac{2.500}{160} = 15,63 \text{ €}$$

3. Calcular el coste total en recursos humanos, ajustando a un sobrecoste estimado entre un 200 % y 300 % para estudiantes:

$$\text{Coste por hora (ajustado)} = \text{Salario por hora} \times 2.5$$

$$\text{Coste por hora (ajustado)} = 15,63 \times 2.5 = 39,08 \text{ €} \quad (\text{Sobrecoste del } 250\%)$$

$$\text{Coste total en recursos humanos (estudiante)} = 39,08 \times 385 = \mathbf{15.045,80 \text{ €}}$$

Es importante mencionar que el presupuesto asignado no es competitivo para la investigación que se quiere llevar a cabo. Generalmente, estas tareas son realizadas por expertos en la área, sin embargo este proyecto se formuló para ser ejecutado sin experiencia previa, con el principal objetivo de aprender durante el proceso. Como resultado, las tareas que un investigador experimentado podría completar en menos tiempo se extienden, lo que conlleva un aumento en los costos.

Además del gasto anterior, es necesario tener en cuenta el coste relativo a las horas de tutorización llevadas a cabo por los profesores asociadas a las reuniones llevadas a cabo y detalladas en el Diagrama de Gantt (2.1), como aquellas destinadas a la corrección de esta memoria. Para realizar este cálculo deben contabilizarse las horas de esfuerzo y multiplicarse por el precio estimado por hora para un Profesor Titular de la Universidad de Granada [23]. Por tanto, este coste sería el siguiente:

1. Calcular el precio por hora para profesores, ajustando al sobrecoste que incluye los salarios en presupuestos:

$$\text{Precio por hora} = 90 \text{ €} \quad (\text{Coste por hora típico en proyectos europeos})$$

2. Calcular el coste total por cada profesor que tutoriza este trabajo:

$$\text{Pablo García Sánchez} = 25 \text{ horas} \times 90 \text{ €/ hora} = 2.250 \text{ €}$$

$$\text{Rafael Alejandro Rodríguez Gómez} = 25 \text{ horas} \times 90 \text{ €/ hora} = 2.250 \text{ €}$$

$$\text{Total} = 4.500,00 \text{ €}$$

Fuente: *Datos sobre retribuciones obtenidos de la Universidad de Granada [23]*.

Descripción	Horas	Coste por hora	Coste Total
Pablo García Sánchez (tutor)	25	90 €	2.250 €
Rafael Alejandro Rodríguez Gómez (cotutor)	25	90 €	2.250 €
<b>Coste total en tutorización</b>			<b>4.500,00 €</b>

Tabla 2.4: Coste total de Recursos Humanos asociados a tutorización

---

<sup>0</sup>Un trienio es un complemento de sueldo que se otorga por cada tres años de servicios prestados en la misma entidad, en reconocimiento a la experiencia y antigüedad acumulada.



# Capítulo 3

## Estado del Arte

En este capítulo se llevará a cabo un primer acercamiento a los problemas de seguridad actuales en los sistemas Linux y se introducirán los fundamentos del marco táctico de MITRE ATT&CK. También se realizará un análisis de las principales ventajas que está trayendo consigo el uso de la IA para la detección de vectores de ataque, y por último se explicará cómo funcionan los SIEM más vanguardistas.

### 3.1. Seguridad en sistemas operativos Linux

Las distribuciones Linux son con un 47% [1], tal y como se indica en la gráfica de la Figura 3.1, una de las principales elecciones para las personas que trabajan en el sector tecnológico. Son muchas las ventajas frente a sus principales competidores, como la posibilidad de adaptación y modificación tan amplia, permitiendo personalizar por completo el entorno de trabajo. Otro factor diferencial es que es de código abierto, y su descarga y uso son totalmente gratuitos, abaratando costes y permitiendo un uso escalable para las empresas e instituciones.

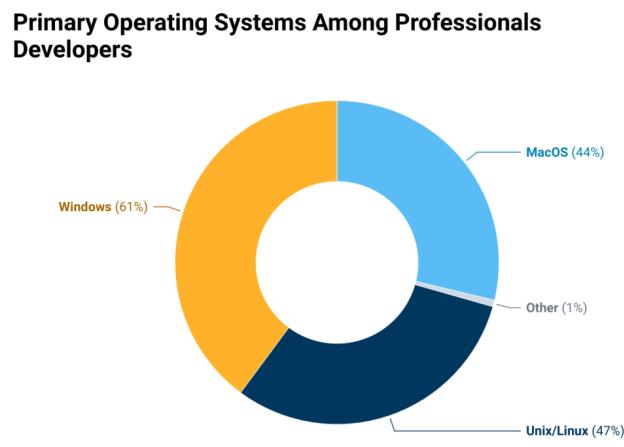


Figura 3.1: Comparativa entre sistemas operativos usados por profesionales [1]

Dentro del sector de la ciberseguridad, destaca por sus distribuciones orientadas al *pentesting* como *Kali Linux* [24], *ParrotOS* [25] o *BlackArch* [26].

### Sistema de Archivos de Linux

El sistema de archivos de Linux (3.1) se distingue por tener una estructura altamente organizada en base a la funcionalidad de los directorios. Entre estos, los ficheros de registro, que son críticos para la seguridad y el diagnóstico del sistema, se almacenan sistemáticamente en `/var/log`, de modo que las todas las actividades y eventos del sistema sean rastreados detalladamente y mapeados en función de una serie de características.

Directorio	Descripción
<code>/bin</code>	Binarios de comandos esenciales
<code>/boot</code>	Archivos del cargador de arranque del sistema
<code>/dev</code>	Archivos de dispositivo
<code>/etc</code>	Archivos de configuración específicos del host
<code>/home</code>	Directorio de inicio del usuario
<code>/lib</code>	Módulos de biblioteca compartida
<code>/media</code>	Archivos de medios como <i>CD-ROM</i>
<code>/mnt</code>	Sistemas de archivos montados temporalmente
<code>/opt</code>	Paquetes de software de aplicaciones adicionales
<code>/proc</code>	Sistema de archivos generado automáticamente
<code>/root</code>	Directorio de inicio del usuario root
<code>/run</code>	Datos del programa en tiempo de ejecución
<code>/sbin</code>	Binarios del sistema
<code>/srv</code>	Datos específicos del sitio servidos por este sistema
<code>/sys</code>	Directorio virtual que proporciona información del sistema
<code>/tmp</code>	Archivos temporales
<code>/usr</code>	Archivos de usuario solo lectura
<code>/var</code>	Archivos que se esperan cambien continuamente

Tabla 3.1: Estructura del sistema de archivos de SOs basados en Linux

### Gestión de permisos en Linux

Una de las principales ventajas de los sistemas Linux es la gestión de permisos, que se desglosa en tres categorías de usuario. Este modelo divide los derechos de acceso entre el usuario (propietario del archivo), el grupo (otros usuarios que pertenecen al mismo grupo que el archivo) y otros (todos los demás usuarios del sistema). Cada archivo y directorio tiene un conjunto de permisos asociados para estas tres categorías, lo que permite un control granular sobre quién puede leer, escribir o ejecutar un archivo en el sistema.

Los permisos en Linux se identifican comúnmente como lectura (**r**), escritura (**w**) y ejecución (**x**) tal y como se indica en la siguiente tabla (3.2). Además, Linux permite el uso de mecanismos avanzados como los permisos **setuid** y **setgid**, que otorgan a los programas la capacidad de ejecutarse con los privilegios de otro usuario o grupo, respectivamente. Estos permisos se pueden asignar por medio del comando **chmod** [6]. Además, los permisos también pueden estar asociados a un fichero específico o a su nivel de pertenencia (3.3).

```
-rwxr-xr-- 22 root 4096 2024-03-03 18:09 file.txt
```

En este ejemplo, el archivo tiene:

- Permisos de lectura, escritura y ejecución para el usuario propietario (rwx)
- Permisos de lectura y ejecución para el grupo (r-x)
- Solo permisos de lectura para otros usuarios (r-)

Abreviatura	Valor	Acción
r	4	Lectura
w	2	Escritura
x	1	Ejecución
-	0	Sin permiso

Tabla 3.2: Estructura de permisos en Sistemas Linux [6]

Quién	Significado	Tipo de Archivo Abreviatura	Tipo de Archivo
u	Usuario	d	Directorio
g	Grupo	-	Archivo regular
o	Otros	l	Enlace simbólico
a	Todos (ugo)	-	-

Tabla 3.3: Tipos y propietarios de archivos en Sistemas Linux [6]

Es crucial implementar la política de mínima permisibilidad en la gestión de permisos para minimizar los riesgos de ataques que explotan las estructuras de permisos laxos. Limitar el acceso a la información y funcionalidades del sistema solo a quienes realmente lo necesitan ayuda a proteger contra amenazas internas y externas, y es especialmente importante para evitar que los ataques se propaguen a través de la cadena de mando en un organigrama empresarial.

### 3.1.1. *Hardening* del sistema

Existen una serie de medidas básicas de seguridad [27] para añadir una capa de protección a los Sistemas Linux, y que desgraciadamente no suelen llevarse a cabo por desconocimiento, incluso por aquellos profesionales que despliegan sus servidores y acaban siendo atacados a causa de este motivo. Entre estas medidas, destacan:

- A. **Configuración de conexión remota SSH:** cambiar el puerto predeterminado, sustituir el acceso por contraseña por el uso de pares de claves pública-privada y deshabilitar el acceso root directo.

- B. **Listar puertos abiertos:** revisar y filtrar los puertos abiertos para mantener solo los necesarios para la operación segura del sistema.
- C. **Habilitar IPTABLES:** configuración de un *firewall* para controlar el acceso a los servicios según las necesidades de seguridad.
- D. **Prevención de Ataques DDoS (*Distributed Denial-of-Service*):** implementación de medidas para mitigar ataques de este tipo, como limitar conexiones y usar SYN cookies.
- E. **Software Instalado:** gestión de los paquetes instalados para asegurar que solo se mantengan los necesarios y se evite software no confiable.
- F. **Actualización Kernel y Parches de Seguridad:** mantener el sistema actualizado con los últimos parches de seguridad para proteger contra vulnerabilidades conocidas.
- G. **Password aging:** implementación de políticas de caducidad y renovación de contraseñas para mejorar la seguridad de las cuentas.
- H. **Encriptación de disco duro:** aplicación de cifrado al nivel de disco para proteger los datos en caso de acceso físico no autorizado.
- I. **Antivirus:** uso de software antivirus para detectar y prevenir malware y otros tipos de software malicioso.
- J. **Uso de VPN (*Virtual Private Network*):** conexión a redes privadas virtuales para asegurar las conexiones remotas y proteger los datos en tránsito.
- K. **Servidor de archivos NFS:** configuración segura de servidores de archivos en red para compartir recursos dentro de una red de confianza.
- L. **Habilitar SUDO:** control de los privilegios de ejecución de comandos para limitar las operaciones que los usuarios pueden realizar.
- M. **Backups:** implementación de copias de seguridad cronológicas para proteger los datos frente a pérdidas accidentales o maliciosas.
- N. **Monitorización y manejo de logs:** revisión y análisis de *logs* para detectar actividades inusuales y potenciales brechas de seguridad.

De todas las medidas anteriores, este proyecto estará enfocado en la última: monitorización y manejo de *logs*, con el fin de detectar posible patrones de ataque en base al análisis de los eventos registrados en un sistema o conjunto de sistemas Linux. Igualmente, cada una de ellas representan una capa de robustez en el nivel de seguridad de equipo, por lo que su importancia debe ser tenida en cuenta.

### 3.1.2. Generación y monitorización de logs

Los logs son ficheros que registran actividades realizadas en un sistema o servicio. Por consiguiente, estos pueden llegar a ser nuestros principales aliados para detectar intentos de acceso no autorizados, manipulación de datos o incluso intentos de intrusión. Además, proporcionan evidencia crucial en caso de incidentes de seguridad, ayudando en las investigaciones forenses digitales.

Existen diversas herramientas y soluciones para la generación, monitorización y gestión de *logs* que automatizan y facilitan estas tareas. Entre las más destacadas se encuentran:

**ELK Stack** (*Elasticsearch*, *Logstash*, y *Kibana*) [28]: esta suite de herramientas permite la recopilación, indexación, y visualización de *logs* de manera eficiente y escalable. *Elasticsearch* indexa los *logs*, *Logstash* los procesa y *Kibana* proporciona una interfaz gráfica para explorar y visualizar los datos.

**Splunk** [29]: otra solución popular que CISCO [30] ofrece con capacidades avanzadas de recogida, indexación y análisis de grandes volúmenes de datos en tiempo real. *Splunk* es particularmente apreciado por su potente interfaz de búsqueda y visualización.

**Zabbix** [31]: además de monitorizar el rendimiento y la disponibilidad, Zabbix es capaz de recoger y analizar *logs* de diversos sistemas y dispositivos, ofreciendo una solución integral para el seguimiento de infraestructuras IT.

**Syslog** es, según el RFC 5424 [32], una de las herramientas más antiguas y ampliamente utilizadas para la gestión de *logs* en sistemas GNU/Linux. Permite la centralización de *logs* de múltiples sistemas en un servidor **syslog**. Utiliza niveles de prioridad asignados a los mensajes:

Valor	Prioridad	Descripción
0	Emergency	El sistema no se puede usar
1	Alert	Se debe actuar inmediatamente
2	Critical	Condiciones críticas
3	Error	Condiciones de error
4	Warning	Condiciones de advertencia
5	Notice	Condición normal pero significativa
6	Info	Mensajes informativos
7	Debug	Mensajes de nivel de depuración

Tabla 3.4: Niveles de prioridad de syslog

Además, hace uso de las llamadas *facilities*, una serie de categorías predefinidas que clasifican los logs en función de la parte del sistema operativo que las ha generado, ayudando a organizar los mensajes para facilitar su filtrado y procesamiento.

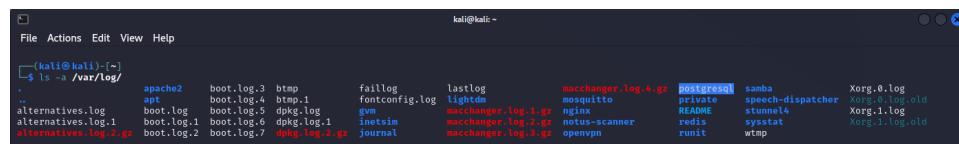
Código	Keyword	Descripción
0	kern	Mensajes del kernel
1	user	Mensajes a nivel de usuario
2	mail	Sistema de correo
3	daemon	Demonios del sistema
4	auth	Mensajes de seguridad/autenticación
5	syslog	Mensajes generados internamente por syslogd
6	lpr	Subsistema de impresora de línea
7	news	Subsistema de noticias de red
8	uucp	Subsistema UUCP ( <i>Unix to Unix Copy Protocol</i> )
9	cron	Subsistema cron
10	authpriv	Mensajes de seguridad/autenticación privados
11	ftp	Demonio FTP ( <i>File Transfer Protocol</i> )
12	ntp	Subsistema NTP ( <i>Network Time Protocol</i> )
13	security	Auditoría de logs
14	console	Alerta de logs
15	solaris-cron	Demonio de planificación
16–23	local0 – local7	Facilities usados localmente

Tabla 3.5: *Facilities* de Syslog

Las implementaciones de **syslog** varían y están adaptadas para diferentes distribuciones de Linux. Por ejemplo, **rsyslog** es más común en distribuciones basadas en Debian, mientras que **syslog-ng**<sup>1</sup> es ampliamente utilizado en distribuciones basadas en RedHat.

Entre las ventajas de **rsyslog** se encuentra su capacidad para manejar grandes volúmenes de datos. Por otro lado, **syslog-ng** destaca por su capacidad de *parsing* y filtrado de *logs* antes de su almacenamiento, lo que facilita una organización más eficiente y una mejor detección de problemas.

Según se observa en la Figura 3.2, estos ficheros se encuentran comúnmente en la ruta **/var/log**, algunos de ellos en subcarpetas asociadas a servicios o programas específicos que cuentan con su propio sistema de monitorización. En cuanto a la nomenclatura utilizada para los ficheros, hay varios que tienen el mismo nombre seguido de la expresión regular **<name>.log.\*** donde \* es un número que aumenta secuencialmente. Esto se debe a que cuando se completa el espacio máximo utilizable de un fichero se genera otro nuevo con ese mismo tamaño utilizable. Por ejemplo, en el caso de **boot.log** o de **dpkg.log**.

Figura 3.2: Contenido del registro de **/var/log**

<sup>1</sup>Donde ng hace referencia a *new-generation*

Los logs pueden presentar diferencias en base a la información que almacenan, al formato y al orden en el que la representan, de modo que inicialmente no resulta sencillo trabajar con dos tipos de ficheros *log* de diferentes formatos. En las siguientes Figuras, se muestra el contenido de *auth.log* y de *dpkg.log* (3.3 y 3.4 respectivamente):

```

File: /var/log/auth.log
1 2024-04-23T06:17:37.104836+04:00 kali sudo: pam_unix(sudo:session): session closed for user root
2 2024-04-23T06:25:01.261384+04:00 kali CRON[1730]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
3 2024-04-23T06:25:01.261785+04:00 kali CRON[1730]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
4 2024-04-23T06:25:01.262330+04:00 kali CRON[1730]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
5 2024-04-23T06:25:01.262404+04:00 kali CRON[1730]: pam_unix(cron:session): session closed for user root
6 2024-04-23T06:25:01.262454+04:00 kali CRON[1730]: pam_unix(cron:session): session closed for user root
7 2024-04-23T06:25:02.875617+04:00 kali CRON[1730]: pam_unix(cron:session): session closed for user root
8 2024-04-23T06:35:01.928467+04:00 kali CRON[21305]: pam_unix(cron:session): session closed for user root
9 2024-04-23T06:35:01.942224+04:00 kali CRON[21305]: pam_unix(cron:session): session opened for user root
10 2024-04-23T06:35:01.942230+04:00 kali CRON[21305]: pam_unix(cron:session): session opened for user root
11 2024-04-23T06:35:01.942230+04:00 kali CRON[21305]: pam_unix(cron:session): session opened for user root
12 2024-04-23T06:35:54.336733+04:00 kali lightdm: pam_unix(lightdm-greeter:session): session opened for user lightdm(uid=110) by (uid=0)
13 2024-04-23T06:35:54.336804+04:00 kali (systemd): pam_unix(systemd-user:session): session opened for user lightdm(uid=110) by (uid=0)
14 2024-04-23T06:35:54.336804+04:00 kali lightdm: pam_unix(lightdm-greeter:session): session opened for user lightdm(uid=110) by (uid=0)
15 2024-04-23T06:35:54.336804+04:00 kali lightdm: pam_unix(lightdm-greeter:session): session opened for user root(uid=0) by (uid=0)
16 2024-04-23T06:35:54.336804+04:00 kali lightdm: pam_unix(lightdm-greeter:session): session opened for user root(uid=0) by (uid=0)
17 2024-04-23T06:45:01.042633+04:00 kali CRON[26108]: pam_unix(cron:session): session closed for user root
18 2024-04-23T06:45:01.042633+04:00 kali CRON[31125]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
19 2024-04-23T06:45:01.042633+04:00 kali CRON[31125]: pam_unix(cron:session): session closed for user root
20 2024-04-23T07:05:01.251627+04:00 kali CRON[35584]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
21 2024-04-23T07:05:01.609532+04:00 kali CRON[35584]: pam_unix(cron:session): session closed for user root
22 2024-04-23T07:05:01.609532+04:00 kali CRON[37407]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
23 2024-04-23T07:05:01.609532+04:00 kali CRON[37407]: pam_unix(cron:session): session closed for user root
24 2024-04-23T07:15:02.149496+04:00 kali CRON[40127]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
25 2024-04-23T07:15:02.169090+04:00 kali CRON[40122]: pam_unix(cron:session): session closed for user root
26 2024-04-23T07:15:02.169090+04:00 kali CRON[41238]: pam_unix(cron:session): session closed for user root
27 2024-04-23T07:15:02.169090+04:00 kali CRON[41338]: pam_unix(cron:session): session closed for user root
28 2024-04-23T07:15:02.169090+04:00 kali CRON[41338]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
29 2024-04-23T07:25:01.537336+04:00 kali CRON[44688]: pam_unix(cron:session): session closed for user root
30 2024-04-23T07:35:01.660787+04:00 kali CRON[49180]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
31 2024-04-23T07:35:01.660787+04:00 kali CRON[49180]: pam_unix(cron:session): session closed for user root
32 2024-04-23T07:39:01.795607+04:00 kali CRON[51411]: pam_unix(cron:session): session opened for user root
33 2024-04-23T07:39:01.795607+04:00 kali CRON[51411]: pam_unix(cron:session): session closed for user root
34 2024-04-23T07:45:01.417248+04:00 kali CRON[53765]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
35 2024-04-23T07:45:01.430223+04:00 kali CRON[53765]: pam_unix(cron:session): session closed for user root
36 2024-04-23T07:45:01.430223+04:00 kali CRON[57540]: pam_unix(cron:session): session closed for user root
37 2024-04-23T08:15:01.722616+04:00 kali CRON[57540]: pam_unix(cron:session): session closed for user root
38 2024-04-23T08:17:01.739878+04:00 kali CRON[58566]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
39 2024-04-23T08:17:01.739878+04:00 kali CRON[58566]: pam_unix(cron:session): session closed for user root
40 2024-04-23T08:25:01.811331+04:00 kali CRON[62168]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
41 2024-04-23T08:25:01.828790+04:00 kali CRON[62168]: pam_unix(cron:session): session closed for user root
42 2024-04-25 09:35:41.971794+04:00 kali CRON[66003]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)

```

Figura 3.3: Contenido del registro de *auth.log*

```

File: /var/log/dpkg.log.1
1 2024-02-08 18:38:58 startup archives unpack
2 2024-02-08 18:39:01 install għamdm64 <none> 2.43.1
3 2024-02-08 18:39:01 status half-installed għamdm64 2.43.1
4 2024-02-08 18:39:01 triggers-pending kali-menu:all 2023.4.6
5 2024-02-08 18:39:01 triggers-pending man-db:amd64 2.12.0-1
6 2024-02-08 18:39:02 status unpacked għamdm64 2.43.1
7 2024-02-08 18:39:02 startup packages config
8 2024-02-08 18:39:02 configure għamdm64 2.43.1 <none>
9 2024-02-08 18:39:02 status half-installed għamdm64 2.43.1
10 2024-02-08 18:39:02 status half-configured għamdm64 2.43.1
11 2024-02-08 18:39:02 status installed għamdm64 2.43.1
12 2024-02-08 18:39:02 triggproc man-db:amd64 2.12.0-1 <none>
13 2024-02-08 18:39:02 status half-configured man-db:amd64 2.12.0-1
14 2024-02-08 18:39:02 triggers-pending kali-menu:all 2023.4.6
15 2024-02-08 18:39:02 triggers-pending man-db:amd64 2.12.0-1
16 2024-02-08 18:39:02 triggers-pending kali-menu:all 2023.4.6
17 2024-02-08 18:39:02 triggers-pending man-db:amd64 2.12.0-1
18 2024-02-08 18:39:02 startup archives unpack
19 2024-02-08 18:39:02 install nodejs:amd64 <none> 4.0.2-3
20 2024-02-08 18:39:02 status half-installed nodejs:amd64 4.0.2-3
21 2024-02-08 12:16:02 status unpacked node-xtend:all 4.0.2-3
22 2024-02-08 12:16:02 install nodejs:amd64 <none> 18.19.0+dfsg-6
23 2024-02-08 12:16:02 status f-installed nodejs:amd64 18.19.0+dfsg-6
24 2024-02-08 12:16:02 status unpacked node-xtend:all 4.0.2-3
25 2024-02-08 12:16:02 status triggers-pending node-xtend:all 4.0.2-3
26 2024-02-08 12:16:02 status unpacked nodejs:amd64 18.19.0+dfsg-6
27 2024-02-08 12:16:02 install node-acorn:nodejs <none> 8.8.1+ds-cs23.11.12.3-6
28 2024-02-08 12:16:02 status half-installed node-acorn:nodejs 8.8.1+ds-cs23.11.12.3-6
29 2024-02-08 12:16:02 status unpacked node-acorn:nodejs 8.8.1+ds-cs23.17.7-2
30 2024-02-08 12:16:02 install node-cjs-module-lexical:nodejs 1.2.2+dfsg-5
31 2024-02-08 12:16:02 status half-installed node-cjs-module-lexical:nodejs 1.2.2+dfsg-5
32 2024-02-08 12:16:02 status unpacked node-cjs-module-lexical:nodejs 1.2.2+dfsg-5
33 2024-02-08 12:16:02 status triggers-pending node-cjs-module-lexical:nodejs 1.2.2+dfsg-5
34 2024-02-08 12:16:03 status unpacked node-undici:nodejs 5.28.2+dfsg1+cs23.11.12.3-6
35 2024-02-08 12:16:03 status unpacked node-undici:nodejs 5.28.2+dfsg1+cs23.11.12.3-6
36 2024-02-08 12:16:03 install libnode108:amd64 18.19.0+dfsg-6
37 2024-02-08 12:16:03 status unpacked libnode108:amd64 18.19.0+dfsg-6
38 2024-02-08 12:16:03 status half-installed libnode108:amd64 18.19.0+dfsg-6
39 2024-02-08 12:16:04 status unpacked libnode108:amd64 18.19.0+dfsg-6
40 2024-02-08 12:16:05 install nodejs-doc:all <none> 18.19.0+dfsg-6
41 2024-02-08 12:16:05 status half-installed nodejs-doc:all 18.19.0+dfsg-6
42 2024-02-08 12:16:05 status triggers-pending doc-base:all 0.11.1

```

Figura 3.4: Contenido del registro de *dpkg.log*

En estos ejemplos, ambos ficheros contienen los campos *timestamp*, es decir, la fecha y la hora a la que sucedió cada evento, y *message*, pero propiedades como la separación entre campos o la precisión que presentan son distintas. A pesar de estas diferencias, la información puede llegar a combinarse a través de técnicas de preprocesamiento de modo que sea utilizable dentro de un mismo formato de *datasets*.

Además de los ficheros comentados anteriormente, existen otras herramientas importantes para el manejo de *logs* en Linux que contribuyen al análisis de seguridad y al diagnóstico de problemas:

- **dmesg**: muestra mensajes del kernel útiles para diagnosticar problemas de hardware y de carga de controladores durante el arranque (puede visualizarse en la Figura 3.5).
  - **journalctl**: herramienta para consultar y administrar *logs* del sistema gestionados por **systemd** (puede visualizarse en la Figura 3.6).

Figura 3.5: Contenido de los logs de *dmesq*

Figura 3.6: Contenido de los logs de *journalctl*

### 3.2. El marco de MITRE ATT&CK

Con el fin de proporcionar una guía detallada y práctica que permita entender y combatir mejor los ciberataques, nació en 2013 [33] el marco de técnicas adversarias de MITRE ATT&CK (*Adversarial Tactics, Techniques, and Common Knowledge*). Este fue creado por el NIST, y clasifica las diferentes TTPs (*Tactics, Techniques and Procedures*) seguidas por los cibercriminales en cada una de las fases de un ataque.

- **Tácticas:** Describen el *qué* y el *porqué* de las acciones de los adversarios, refiriéndose a su objetivo general y estrategia durante un ataque.
  - **Técnicas:** Revelan el *cómo*, especificando el método específico empleado para alcanzar un objetivo táctico.
  - **Procedimientos:** Son los *detalles exactos* de la técnica utilizada, incluyendo los comandos ejecutados, la secuencia de operaciones y cualquier configuración aplicada.

Por ejemplo, en un ataque de *phishing*, la **táctica** empleada sería la recopilación inicial de acceso, la técnica podría ser el envío de correos electrónicos engañosos y por último el **procedimiento** consistiría en el uso de un correo electrónico que aparenta ser una alerta legítima de seguridad de una institución conocida para engañar a los usuarios y hacer que revelen sus credenciales.

MITRE ATT&CK está compuesto por una lista estructurada de comportamientos expresados en forma matricial que, tal y como se ilustra en la Figura 3.7, proporcionan una taxonomía común de acciones tanto del propio adversario como de contra-medidas para defenderse del mismo.

Figura 3.7: Matriz de técnicas y tácticas de MITRE ATT&CK para enterprise [2]

Actualmente es muy utilizado tanto a nivel empresarial como gubernamental a modo de modelo base para el desarrollo de productos y servicios de ciberseguridad. La principal causa de su éxito podría ser la forma en la que se describe cada procedimiento de forma detallada y con una terminología correcta y entendible, pero sin llegar a mencionar el uso de herramientas específicas que lleven a cabo las acciones descritas.

Una gran ventaja de MITRE ATT&CK [2] es que se encuentra en una constante evolución, incorporando a la matriz técnicas y sub-técnicas vanguardistas que van apareciendo en la escena del cibercrimen. Cada una de las técnicas y sub-técnicas 3.2.2 de la matriz de MITRE está asociada a un identificador único y cuenta con su correspondiente descripción, además de otra información de relevancia como un listado de formas de mitigación y/o detección.

Estos identificadores se estructuran siguiendo un patrón muy similar: uno o dos caracteres seguidos de cuatro dígitos. Esta forma de identificarlos permite reservar el suficiente espacio como para añadir durante muchos años nuevas tácticas, técnicas, sub-técnicas, procedimientos, etc.

Identificador	Categoría	Descripción
TA####	Tácticas	TA indica que es un identificador de táctica y va seguido de 4 dígitos. Ejemplo: TA0001.
T####	Técnicas	T indica que es un identificador de técnica y va seguido de 4 dígitos. Ejemplo: T1059.
T###.##	Sub-técnicas	Parecido a los identificadores de técnicas, pero cuentan con caracteres adicionales ya que se agrupan dentro de los anteriores. Ejemplo: T1059.001.
M####	Mitigaciones	M indica que es un identificador de mitigación y va seguido de 4 dígitos. Ejemplo: M1050.
DS####	Detecciones	DS indica que es un identificador de detección y va seguido de 4 dígitos. Ejemplo: DS0029.
S### — G###	Procedimientos	Identificadores de procedimientos específicos usados por actores de amenazas o grupos específicos. Ejemplo: S0003 para procedimientos y G0007 para grupos.

Tabla 3.6: Identificadores del marco de *MITRE ATT&CK*

Hace relativamente poco, se publicaron versiones más específicas de esta matriz tanto para móviles como para ICS (*Industrial control system*). Estas contienen menos volumen de tácticas y técnicas pero se adaptan mejor a un escenario real de ataque sobre este tipo de entornos. Además, desde MITRE han desarrollado también matrices específicas para distintos Sistemas Operativos y áreas de conocimiento, también más pequeñas pero más precisas al entorno de simulación.

Categoría	Plataformas / Sistemas
Enterprise	PRE Windows macOS Linux Cloud Network Containers
Mobile	Android iOS
ICS	PLC, RTU, SCADA, DCS

Tabla 3.7: Matrices de MITRE ATT&CK

En el caso de este proyecto se hará uso de la matriz de Linux ya que es la que mejor se adapta al escenario donde se llevarán a cabo las simulaciones de ataques.

### 3.2.1. Tácticas

MITRE ATT&CK [2] agrupa las distintas técnicas de intrusión dentro de catorce tácticas distintas ordenadas secuencialmente, aunque este orden puede orden variar en determinadas operaciones.

Código	Fase del Ataque	Descripción
1	Reconocimiento	Recopilación de información para la planificación futura un ataque.
2	Desarrollo de recursos	Creación de recursos que respalden el desarrollo de un ataque.
3	Acceso inicial	Intento inicial de acceder al objetivo a comprometer.
4	Ejecución	Introducción de código malicioso en el sistema atacado.
5	Persistencia	Asentar un acceso continuo al sistema objetivo a través de reinicios, cambios de credenciales y supresión de interrupciones que podrían cancelar su acceso.
6	Escalada de privilegios	Obtención de permisos de nivel superior para poder acceder a más recursos del sistema atacado.
7	Evasión de defensa	Ocultación frente a la detección mientras se está efectuando la operación de compromiso.
8	Acceso a credenciales	Robo de credenciales para acceder al sistema de destino.
9	Descubrimiento	Obtención de más conocimientos sobre el sistema objetivo y su entorno.
10	Movimiento lateral	Desplazamiento a través del entorno interno del conjunto de sistemas objetivo.
11	Recopilación	Búsqueda de información relevante del objetivo.
12	C2 - Command & Control	Comunicación remota con el sistema comprometido para controlar y operar sobre este.
13	Exfiltración	Robo de datos sensibles almacenados en el sistema comprometido.
14	Impacto	Manipulación, ocultación o destrucción del sistema comprometido y sus datos.

Tabla 3.8: Conjunto de tácticas ofensivas del marco de *MITRE ATT&CK*

### 3.2.2. Técnicas

Cada una de las tácticas indicadas en la subsección anterior (3.2.1) contiene un listado técnicas que pueden utilizarse para llevar a cabo cada objetivo. Estas técnicas a su vez agrupan una serie de subtécnicas, proporcionando un mayor nivel de abstracción que aglutina una mayor base de conocimiento con respecto a todas las formas de llevar a cabo una acción.

Todo esto puede resultar inicialmente algo confuso de entender, por lo que una manera útil de comprenderlo es imaginarlo como una especie de matriz tridimensional en la que cada táctica es una fila (abscisas), cada técnica una columna (ordenadas) y por último cada subtécnica una altura o cota. Por ejemplo, dentro de la táctica de Reconocimiento (3.8) existen actualmente diez técnicas distintas, cada una de ellas con una serie de subtécnicas más específicas.



Figura 3.8: Listado de técnicas de la táctica de *Reconnaissance* [2]

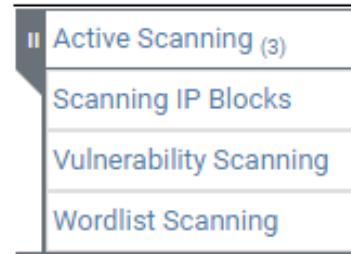


Figura 3.9: Sub-técnicas de *Active Scanning* [2]

Según se ilustra en las Figuras 3.8 y 3.9, la primera técnica de la táctica de Reconocimiento, *Active Scanning*, cuenta con 3 subtécnicas: *Scanning IP Blocks*, *Vulnerability Scanning* y *Wordlist Scanning*.

### 3.2.3. Procedimientos

Por último, se encuentran los procedimientos, que ofrecen una guía detallada de cómo se han de realizar las las técnicas o subtécnicas. En el ejemplo ilustrado en la Figura 3.10, se explica el procedimiento para realizar escaneos de bloques IP como técnica de reconocimiento, en la que los adversarios pueden escanear bloques de direcciones IP para recabar información sobre redes objetivo que posteriormente puede ser usada en ataques. Dichos escaneos varían desde *pings* simples hasta métodos más sofisticados que revelan versiones de software hasta otros más sencillos.

Home > Techniques > Enterprise > Active Scanning > Scanning IP Blocks

**Active Scanning: Scanning IP Blocks**

Other sub-techniques of Active Scanning (3)

Adversaries may scan victim IP blocks to gather information that can be used during targeting. Public IP addresses may be allocated to organizations by block, or a range of sequential addresses.

Adversaries may scan IP blocks in order to Gather Victim Network Information, such as which IP addresses are actively in use as well as more detailed information about hosts assigned these addresses. Scans may range from simple pings (ICMP requests and responses) to more nuanced scans that may reveal host software/versions via server banners or other network artifacts.<sup>[1]</sup> Information from these scans may reveal opportunities for other forms of reconnaissance (ex: Search Open Websites/Domains or Search Open Technical Databases), establishing operational resources (ex: Develop Capabilities or Obtain Capabilities), and/or initial access (ex: External Remote Services).

ID: T1595.001  
Sub-technique of: T1595  
Tactic: Reconnaissance  
Platforms: PRE  
Version: 1.0  
Created: 02 October 2020  
Last Modified: 15 April 2021  
[Version Permalink](#)

Figura 3.10: Información referente al procedimiento asignado *Scanning IP Blocks* [2]

Adicionalmente, como se observa en la Figura 3.11, se puede encontrar justo debajo ejemplos de técnicas de Detección y Mitigación, también identificadas bajo el criterio mencionado anteriormente.

Procedure Examples

ID	Name	Description
60139	TeamTNT	TeamTNT has scanned specific lists of target IP addresses.

Mitigations

ID	Mitigation	Description
M1056	Pre-compromise	This technique cannot be easily mitigated with preventive controls since it is based on behaviors performed outside of the scope of enterprise defenses and controls. Efforts should focus on minimizing the amount and sensitivity of data available to external parties.

Detection

ID	Data Source	Data Component	Detected
DS0029	Network Traffic	Network Traffic Flow	Monitor network data for uncommon data flows. Processes utilizing the network that do not normally have network communication or have never been seen before are suspicious.

Figura 3.11: Ejemplos de mitigación y detección frente a *Scanning IP Blocks* [2]

## Frameworks basados en MITRE ATT&CK

Actualmente existen numerosos *frameworks* basados en el marco adversario de MITRE para orquestar simulaciones de ataques en auditorías de seguridad tanto internas como externas. Algunos de estos *frameworks* son:

- **CALDERA** [34]: Desarrollado por el propio MITRE, este *framework* permite planificar, ejecutar y evaluar campañas adversarias basadas en las técnicas definidas en la matriz de ATT&CK. Con más de 800 artículos y libros publicados en plataformas académicas como Google Scholar y Scopus, puede considerarse el más utilizado en la actualidad y por tanto el elegido para la implementación de este proyecto, por lo que será explicado con mayor detalle en secciones posteriores.
- **Red Team Automation (RTA)** [35]: Un conjunto de *scripts* diseñados para generar tráfico malicioso que simula comportamientos específicos basados en MITRE ATT&CK.
- **Atomic Red Team** [36]: Es un conjunto de pruebas pequeñas y altamente portátiles, cada una diseñada para ejecutar una técnica única de MITRE ATT&CK. Es útil para equipos de seguridad que desean probar sus defensas de manera incremental y sistemática.
- **Infection Monkey** [37]: Una herramienta de seguridad de código abierto que simula ataques en la red para probar la resiliencia de los sistemas y la eficacia de las medidas de defensa en profundidad. Utiliza técnicas de MITRE ATT&CK para evaluar cómo los adversarios podrían avanzar dentro de la red.
- **Scythe** [38]: Esta plataforma permite a los usuarios crear y ejecutar campañas de simulación de amenazas que replican el comportamiento de los adversarios basado en ATT&CK.

### **3.3. Modelos de IA para la detección de vectores de ataque**

Es innegable que en todos los campos tecnológicos se quiere aprovechar al máximo las novedades que está trayendo consigo la Inteligencia Artificial. Concretamente, en el sector de la ciberseguridad muchas empresas de desarrollo de software están tratando de acoplar el análisis e interpretación de datos masificados a través de modelos de ML (*Machine Learning*) y DL (*Deep Learning*) con el fin de agilizar diversas tareas que hasta la fecha han tenido que llevarse a cabo manualmente por personas, ocupando gran parte del horario laboral, y por tanto no pudiendo ser invertido en otras actividades en las que el desempeño humano es totalmente necesario para la toma de decisiones.

Es por ello que está incrementando considerablemente la cantidad de artículos científicos y de investigación publicados en los que se exploran distintas técnicas y se realizan pruebas haciendo uso de *datasets* generados de forma natural o sintética.

Además de las propias empresas, es bien conocido que algunos grupos de cibercriminales o grupos de ciberterrorismo que operan a la escala de grandes empresas tienen sus propias áreas de I+D (*Investigation & Development*) en las que investigan cómo a través de la IA [39] pueden refinar ataques a gran escala, generar *phising*s sofisticados o *malware* simplemente a través de la correcta redacción de un *prompt* enviado a un LLM que no está correctamente sanitizado o que incluso no tiene limitaciones éticas.

Existen distintos enfoques y técnicas para utilizar esta tecnología: Por un lado, el *machine learning* permite resolver problemas concretos utilizando un volumen de datos más pequeño, pero solo aplica a problemas de dimensionalidad baja. Por el contrario, el *deep learning* necesita más datos así como unos recursos de computación mayores (LeCun et. al [40]), pero puede utilizarse para cuestiones con un mayor nivel de abstracción.

Asimismo, la forma en la que estos modelos aprenden acerca de un problema difiere en función del nivel de etiquetado de los datos. Pueden distinguirse los siguientes tipos de aprendizaje:

- *Supervised Learning* (SL): utiliza *datasets* etiquetados y su finalidad es predecir un valor real (regresión) o bien un valor que esté contenido dentro de un conjunto (clasificación).
- *Semi-Supervised Learning* (SSL): el *dataset* solo tiene algunas de sus partes etiquetadas, por lo que se extraen características relevantes y se representan los datos no etiquetados de forma que luego sea posible emplear el conocimiento adquirido en modelos de aprendizaje supervisado.
- *Unsupervised Learning* (UL): los datos no tienen que estar etiquetados, de modo que se trata de encontrar relación entre las diferentes instancias del conjunto de datos.
- *Reinforcement Learning* (RL): se basa en el ajuste de los parámetros para optimizar la llamada función de recompensa, la cual beneficiará al modelo al realizar una acción determinada.

Una técnica ampliamente utilizada en el aprendizaje no supervisado y semi-supervisado es el *clustering*, que permite agrupar datos no etiquetados basándose en sus características y similitudes, como patrones inusuales o comportamientos similares que podrían indicar actividades maliciosas. Esta técnica se puede llevar a cabo utilizando distintos tipos de algoritmos, entre los más comunes se encuentran:

- **K-means:** Es uno de los métodos más sencillos y populares. Funciona dividiendo el conjunto de datos en  $k$  grupos (*clusters*) definidos por  $k$  centroides, que se ajustan iterativamente para minimizar la variación dentro de cada *cluster*. A pesar de su simplicidad, K-means puede ser muy efectivo para detectar agrupaciones naturales en los datos.
- **DBSCAN** (*Density-Based Spatial clustering of Applications with Noise*): Es útil para identificar *clusters* de cualquier forma y manejar ruido (datos atípicos). Agrupa puntos que están densamente conectados, es decir, puntos que están a una distancia mínima unos de otros. Esto permite encontrar estructuras complejas en los datos que K-means podría no detectar.
- **HC** (*Hierarchical clustering*): Este algoritmo crea una jerarquía de *clusters*, que se puede representar mediante un dendrograma. Existen dos enfoques principales:
  - **AHC** - Aglomerativo (*bottom-up*), que comienza con puntos individuales y los fusiona en *clusters* más grandes
  - **DHC** Divisivo (*top-down*), que empieza con todo el conjunto de datos y los divide en *clusters* más pequeños.

### 3.3.1. Principales *Datasets* de logs

Al ser una gran fuente de información sensible, es altamente complejo encontrar *datasets* de ficheros *log* que no hayan sido generados de forma sintética. Del mismo modo, gran parte de los *logfiles* de Linux utilizados como conjuntos de datos para IA solo pueden ser accesibles pagando.

Gracias a la contribución de Jieming Zhu et. al [41], se desarrolló un repositorio en *Github* llamado LogHub, que contiene un gran conjunto de *datasets* de logs de distintos ámbitos como sistemas distribuidos, supercomputadores, sistemas operativos, sistemas móviles, servicios y software.

### 28 3.3. Modelos de IA para la detección de vectores de ataque

Dataset	Descripción	Etiquetado	Duración	Nº Líneas
<b>Sistemas Distribuidos</b>				
HDFS_v1	Logs del sistema de archivos distribuido Hadoop	Sí	38.7 horas	11,175,629
HDFS_v2	Logs del sistema de archivos distribuido Hadoop	-	-	71,118,073
HDFS_v3	Log trazado instrumentado de HDFS (TraceBench)	Sí	-	14,778,079
Hadoop	Logs de trabajos de Hadoop MapReduce	Sí	-	394,308
Spark	Logs de trabajos de Spark	-	-	33,236,604
Zookeeper	Logs del servicio ZooKeeper	-	26.7 días	74,380
OpenStack	Logs de infraestructura de OpenStack	Sí	-	207,820
<b>Supercomputadores</b>				
BGL	Logs del supercomputador Blue Gene/L	Sí	214.7 días	4,747,963
HPC	Logs de clúster de alto rendimiento	-	-	433,489
Thunderbird	Logs del supercomputador Thunderbird	Sí	244 días	211,212,192
<b>Sistemas Operativos</b>				
Windows	Logs de eventos de Windows	-	226.7 días	114,608,388
Linux	Logs de Linux	-	263.9 días	25,567
Mac	Logs de Mac OS	-	7.0 días	117,283
<b>Sistemas Móviles</b>				
Android_v1	Logs de framework de Android	-	-	1,555,005
Android_v2	Logs de framework de Android	-	-	30,348,042
HealthApp	Logs de aplicación de salud	-	10.5 días	253,395
<b>Aplicaciones de Servidor</b>				
Apache	Logs de errores del servidor web Apache	-	263.9 días	56,481
OpenSSH	Logs del servidor OpenSSH	-	28.4 días	655,146
<b>Software Autónomo</b>				
Proxifier	Logs del software Proxifier	-	-	21,329

Tabla 3.9: Conjunto de *Datasets* de logs para el análisis impulsado por IA

Entre los anteriores, algunos de los más potentes son BGL (*Blue Gene/L*) y HDFS (*Hadoop Distributed File System*), que se han utilizado extensamente en investigaciones de detección de anomalías y fallos en sistemas de alta computación, debido a su gran variedad de eventos y la duración prolongada del registro de los *logs* que contienen.

Otro *dataset* destacable a pesar de su menor volumen de datos, es el de Linux. Este puede ser generado con cierta facilidad ya que simplemente es necesario configurar un entorno en el cual utilizar la herramienta *syslog* y automatizar el almacenamiento de distintos tipos de *logs* dentro de un mismo fichero.

### 3.3.1.1. BGL

De las siglas "Blue Gene/L", BGL es un *dataset* que contiene *logs* generados por el supercomputador *Blue Gene/L*, desarrollado por IBM. Este *dataset* se ha utilizado ampliamente en la investigación de técnicas de detección de anomalías y fallos en sistemas. Los *logs* de BGL incluyen una gran variedad de eventos de sistema, lo que permite a los probar diferentes enfoques de ML y DL para identificar patrones inusuales que podrían indicar un ataque o un fallo en el sistema.

### 3.3.1.2. HDFS

El Hadoop Distributed File System (HDFS) es otro *dataset* significativo que consiste en *logs* generados por sistemas distribuidos de *Hadoop*. Este es útil para estudiar el comportamiento de sistemas de archivos distribuidos y detectar anomalías que podrían sugerir intentos de intrusión o errores en el sistema. Los *logs* de HDFS incluyen información detallada sobre operaciones de archivos, accesos y errores, proporcionando una rica fuente de datos para el entrenamiento y evaluación de modelos de detección de anomalías.

### 3.3.1.3. Linux\_2k

El *dataset* Linux\_2k es una colección de *logs* generados por sistemas Linux, específicamente recolectados desde el archivo */var/log/messages* en un servidor Linux durante un período de más de 260 días, como parte del proyecto *Public Security Log Sharing Site* liderado por el Dr. Anton Chuvakin [42]. Este sitio proporciona una variedad de muestras de logs gratuitas de diferentes sistemas, dispositivos de seguridad y red, aplicaciones, etc., recolectados de sistemas reales y que en muchos casos contienen evidencias de compromisos y otras actividades maliciosas.

El proyecto, iniciado el 23 de junio de 2009 y actualizado el 11 de agosto de 2010, busca ofrecer *logs* sin sanitización ni anonimización, permitiendo así un análisis genuino de los datos tal y como fueron registrados por los sistemas de *logging*, lo que lo convierte en un *dataset* especialmente valioso para este proyecto de investigación.

### 3.4. Sistemas de Gestión de Información y Eventos de Seguridad (SIEM)

Actualmente existen distintos tipos de software de seguridad orientados a la prevención, detección y respuesta a cualquier tipo de incidentes de seguridad. Estos trabajan, por lo general, con cantidades ingentes de datos, por lo que están diseñadas de forma óptima para minimizar tiempos de espera.

Sin embargo, a pesar de la funcionalidad que ofrecen, estos presentan un cuello de botella con respecto a la toma de determinadas decisiones. A día de hoy es necesario en muchos casos que un profesional determine manualmente si hay algún tipo de intento de intrusión maliciosa, si hay un falso positivo o si simplemente no hay actividad inusual.

Está claro que es necesaria la intervención humana, pero el hecho de automatizar algunas tareas puede impulsar la inversión de ese tiempo de esfuerzo en otras actividades de mayor interés. Por este hecho, surgen distintas herramientas (Figura 3.10) de seguridad que llevan a cabo este proceso:

Nombre	Descripción
SIEM	Gestión de eventos e información de seguridad.
Firewall	Control de tráfico de red.
WAF	Firewall de aplicaciones web.
EDR	Monitorización y respuesta en <i>endpoints</i> <sup>2</sup> .
Antivirus	Protección frente a software malicioso.
Antimalware	Protección avanzada contra malware.

Tabla 3.10: Herramientas software de seguridad

De los anteriores, uno de los más completos y utilizados es el SIEM [43], ya que puede integrar las funcionalidades de distintas herramientas software de seguridad (Figura 3.11). Inicialmente eran dos herramientas distintas: SIM (*Security Information Management*) y SEM (*Security Events Management*) pero a partir de 2005 se hizo mención por primera vez como una herramienta conjunta por *Williams y Niccoletti* [44]. Son principalmente utilizados en los SOC para supervisar y monitorizar la actividad para poder detectar ataques y mitigarlos eficientemente.

Nombre	Descripción
SOAR	Orquestación, automatización y respuesta.
IDS	Sistemas de detección de intrusiones.
NIDS	Detección de intrusiones en la red.
HIDS	Detección de intrusiones en <i>hosts</i> .
IPS	Sistemas de prevención de intrusiones.
IRS	Sistemas de respuesta de intrusiones.

Tabla 3.11: Herramientas integrables o complementarias al SIEM

---

<sup>2</sup> Dispositivo final que proporciona un punto de entrada en un activo, como un dispositivo conectado a red o un servicio web.

Tal y como define González-Granadillo et al. [45], los SIEM han sido desarrollados en respuesta para ayudar a los administradores a diseñar políticas de seguridad y gestionar eventos de diferentes fuentes. Generalmente, están compuestos de bloques separados (por ejemplo, dispositivo fuente, recolección de registros, normalización de análisis, motor de reglas, almacenamiento de registros, monitoreo de eventos) que pueden trabajar independientemente entre sí, pero sin su coordinación, el SIEM no funcionaría adecuadamente.

Las dos principales métricas utilizadas [46] para medir su funcionamiento son el tiempo medio hasta la detección (MTTD) y el tiempo medio hasta la respuesta (MTTR).

$$\text{MTTD} = \frac{\text{Total Sum of Detection time (TSD)}}{\text{Total Number of Incidents (TNI)}} \quad (3.1)$$

Donde TSD es la suma total del tiempo acumulado que ha pasado desde el inicio de los incidentes hasta su descubrimiento, y la TNI es el número total de incidentes detectados.

$$\text{MTTR} = \frac{\text{Total Sum of Detection to Remediation time (TSR)}}{\text{Total Number of Incidents Remediated (TNIR)}} \quad (3.2)$$

Donde TSR es el tiempo acumulado que ha pasado desde que se descubrieron los incidentes hasta su remediación, y la TNIR es el número total de incidentes remediados.

### 3.4.1. Principales SIEM en el escenario actual

En la actualidad, existen SIEM de pago y también alternativas *opensource*. La principal diferencia reside, además del precio, en la personalización y el soporte técnico. Las soluciones de pago generalmente ofrecen una mayor integración con otros productos y un soporte técnico más completo y directo por parte de los proveedores, como actualizaciones automáticas, asistencia para la configuración y solución rápida de problemas.

Por otro lado, las alternativas *opensource* como las mencionadas por Ángel Vélez et. al [47], si bien pueden ser menos costosas, a menudo requieren un mayor conocimiento técnico para su implementación y mantenimiento, y el soporte suele depender de la comunidad de usuarios, lo cual puede no ser tan inmediato o accesible.

Además, estas pueden tener menos características *out-of-the-box* comparadas con las versiones de pago, lo que podría requerir más desarrollo personalizado para integrarlas completamente en entornos empresariales complejos.

Es posible clasificar los principales SIEM a través del Cuadrante Mágico de Gartner [7], una herramienta analítica que clasifica a los proveedores en un mercado tecnológico específico en cuatro categorías: Líderes, Visionarios, Desafiantes y Jugadores de Nicho.

SIEM	Empresa	OSS	Categoría Gartner
Splunk Enterprise Security	Splunk Inc.	No	Líder
IBM QRadar	IBM	No	Líder
LogRhythm Next-Gen SIEM Platform	LogRhythm, Inc.	No	Líder
ArcSight ESM	Micro Focus	No	Desafiante
AlienVault OSSIM	AT&T Cybersecurity	Sí	Jugador de Nicho
ELK Stack (Elasticsearch, Logstash, Kibana)	Elastic	Sí	Visionario
Wazuh	Wazuh, Inc.	Sí	Jugador de Nicho
Graylog	Graylog, Inc.	Sí	Jugador de Nicho

Tabla 3.12: Principales SIEM en escenario actual según Cuadrante Mágico de Gartner [7]

Los precios de los SIEM de pago como Splunk, perteneciente a CISCO [30], o IBM varían según el volumen de datos y la capacidad de cómputo que necesita utilizarse. De ello surgen varios modelos.

#### Modelo basado en el volumen de datos de Ingesta diaria

Calcula el coste basado en la cantidad de datos procesados diariamente:

$$C_{\text{datos}} = V_{\text{GB}} \times P_{\text{GB}} \quad (3.3)$$

Donde

- $C_{\text{datos}}$  es el coste total basado en el volumen de datos.
- $V_{\text{GB}}$  es el volumen de datos en gigabytes por día.
- $P_{\text{GB}}$  es el precio por gigabyte por día.

#### Modelo basado en unidades de cómputo virtual (SVC)<sup>3</sup>

Determina el coste a partir del nº de unidades de cómputo virtual asignadas:

$$C_{\text{SVC}} = N_{\text{SVC}} \times P_{\text{SVC}} \quad (3.4)$$

Donde:

- $C_{\text{SVC}}$  es el coste total basado en las unidades de cómputo.
- $N_{\text{SVC}}$  es el número de unidades de cómputo virtual de Splunk.
- $P_{\text{SVC}}$  es el precio por unidad de cómputo.

---

<sup>3</sup> Específicamente utilizado por Splunk Enterprise Security de Splunk Inc [48].

### 3.4.2. Nuevas tendencias y mejoras en SIEM

Los SIEM son, como se comenta anteriormente, un pilar fundamental en el modelo de infraestructura de ciberseguridad actual, no solo por su capacidad de centralizar y sintetizar datos, sino también por su evolución constante en la integración con tecnologías emergentes.

Según González-Granadillo et al. [45], estos están convergiendo gradualmente con herramientas de análisis de *big data*, lo que amplía su capacidad para manejar y analizar un volumen más extenso de datos en tiempo real. Esta convergencia permite a los SIEMs detectar patrones complejos de comportamiento que podrían indicar la presencia de amenazas sofisticadas.

Sin embargo, es crucial adaptarse a la GDPR. Por ello, los proveedores de SIEM están mejorando sus plataformas para ofrecer mayores capacidades de anonimización y gestión de la privacidad. Esto incluye funciones que permiten a las organizaciones cumplir con los derechos de acceso, rectificación, limitación del procesamiento, y supresión de datos personales esta garantiza a los individuos [49].

Además, algunas soluciones SIEM modernas incorporan módulos específicos diseñados para gestionar el cumplimiento de la GDPR, facilitando así la auditoría y generación de informes necesarios para demostrar el cumplimiento a los reguladores. Por este motivo, resulta realmente complejo encontrar *datasets raw* de eventos de seguridad que sean públicos o que no hayan pasado previamente por algún tipo de proceso de anonimización antes de haber sido liberados.

---

En base al estudio llevado a cabo de todas las tecnologías anteriores, se establecen las siguientes conclusiones:

Se llevará a cabo la simulación de ataques mediante el *framework* CALDERA dentro del marco táctico de MITRE ATT&CK y se preprocesarán los *logs* obtenidos con el fin de generar un *dataset*, extrayendo las características principales que permitan realizar dicha clasificación. A continuación se realizará un estudio de dicho *dataset* y otros diferentes que hayan sido preprocesados de manera equivalente. Seguidamente, se aplicarán técnicas de ML, más concretamente algoritmos de *clustering* combinados con distintas técnicas de extracción de características y de reducción de la dimensionalidad, para la detección de patrones de ataque. Por último se llevará a cabo un análisis de los resultados obtenidos mediante distintas métricas de evaluación y se efectuará una comparativa con otros proyectos similares, así como un esbozo de cómo puede integrarse esta tecnología en los SIEM actuales.



## Capítulo 4

# Metodología de trabajo

En este capítulo se profundizará en el dominio del problema que se quiere resolver, examinando la metodología a seguir para poder llevar a cabo este proceso, así como las tecnologías y herramientas que se utilizarán para que sea posible su desarrollo.

### 4.1. Complejidad del problema a resolver

La principal complejidad que presenta el análisis de *logs* es, sin duda, la gran heterogeneidad que tiene su representación, así como la estructura abstracta de algunos de sus campos. En el caso de Linux, esta abstracción viene dada por el campo *message*, probablemente el más importante, ya que este contiene la descripción del evento registrado.

Fecha	Hora					
AAAA	MM	DD	hh	mm	ss	Timezone
2024-05-05	09:38:32+02:00	f6194a3d2d4d	sshd[90]	pam_unix(sshd:auth)	: authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=172.17.0.1	

Figura 4.1: Sintaxis de un *log* de Linux

Como se puede observar en la Figura 4.1, los *logs* de Linux presentan un total de 4 campos: *timestamp*, *hostname*, *service* y *message*. Este último representa un claro aumento en la complejidad de preprocesamiento ya que no sigue un patrón concreto sino que esta elección se delega al servicio que está sirviendo la información, como ocurre en la Figura 4.2.

pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=172.17.0.1
Message

Figura 4.2: Sintaxis del *message* de un *log* de Linux

Dentro del mismo puede extraerse otra información útil; en el ejemplo proporcionado se informa de un fallo en la autenticación, así como la TTY utilizada, la IP del host remoto involucrado o el identificador de usuario asociado.

Por otro lado, la sintaxis del *timestamp* sigue el formato ISO 8601 [8], el cual consta de la estructura representada en la Figura 4.3. Analizando los elementos por lo que se compone, se extrae la siguiente información:

Campo	Descripción
Fecha	La parte de la fecha está representada en el formato YYYY-MM-DD, donde: <ul style="list-style-type: none"> <li>■ YYYY representa el año en cuatro dígitos.</li> <li>■ MM representa el mes en dos dígitos (01 a 12).</li> <li>■ DD representa el día del mes en dos dígitos (01 a 31).</li> </ul>
Hora	La parte de la hora sigue el formato hh:mm:ss, donde: <ul style="list-style-type: none"> <li>■ hh representa la hora en formato de 24 horas (00 a 23).</li> <li>■ mm representa los minutos (00 a 59).</li> <li>■ ss representa los segundos (00 a 59).</li> </ul>
Separador	La letra T se utiliza como delimitador entre la fecha y la hora, indicando que la información que sigue es la hora correspondiente a la fecha especificada previamente.
Zona horaria	La parte final del <i>timestamp</i> indica la diferencia con respecto al Tiempo Universal Coordinado (UTC), expresada en el formato ±hh:mm. Por ejemplo, +02:00 indica que la hora está dos horas adelantada con respecto a UTC.

Tabla 4.1: Descripción del formato de *timestamp* según el estándar ISO 8601 [8]

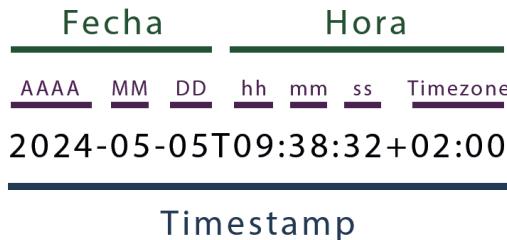


Figura 4.3: Estructura del *timestamp* de un *log* de Linux

Para este proyecto, se hará uso de otros *datasets* de *logs* que contienen una sintaxis que difiere bastante con la ofrecida por *syslog*, por lo que será necesario llevar a cabo un pre-procesamiento de todos los tipos de eventos para que sean homogéneos, de modo que el modelo pueda interpretarlos de forma correcta.

En el caso de BGL, como se puede observar en la Figura 4.4, existen nueve campos, aunque algunos de ellos redundantes: *timestamp* (*epoch-time*), *date*, *service*, *detailed-timestamp*, *hostname*, *component*, *severity*, *type* y *message*. De aquí se puede extraer que se repiten los cuatro campos de los *logs* de Linux, y se añaden otros que pueden resultar muy útiles: la severidad y el tipo de evento.

<u>ssssssss</u>	<u>AAAA MM DD</u>	<u>AAAA MM DD hh mm ss ffffff</u>	<u>Hostname</u>	<u>Com.</u>	<u>Severity</u>	<u>Type</u>	<u>Message</u>
Timestamp (epoch-time)	Date	Service	Detailed Timestamp				
- 1119979190	2005.06.28	R26-M0-NB-C:J07-U01	2005-06-28-10.19.50.589372	R26-M0-NB-C:J07-U01	RAS KERNEL INFO	total of 21 ddr error(s) detected and corrected	

Figura 4.4: Sintaxis de un *log* de BGL

Otro factor a tener en cuenta es que cambia la estructura del *timestamp*, analizando la Figura 4.5, se observa que los delimitadores son guiones y puntos, y que también contiene un campo extra asociado a una mayor precisión en el momento de captura del evento, ya que también mide a nivel de microsegundos, pero se pierde la precisión de la zona horaria.

<u>AAAA</u>	<u>MM</u>	<u>DD</u>	<u>hh</u>	<u>mm</u>	<u>ss</u>	<u>fffffff</u>
2005	-06	-28	-10.19.50.589372			

Detailed Timestamp

Figura 4.5: Estructura del *timestamp* de un *log* de BGL

Por último, en el caso de HDFS, según se ilustra en la Figura 4.6 los eventos se organizan en cuatro campos: *timestamp*, *thread ID*, *type* y *message*. El único campo nuevo es el identificador de la hebra, aunque no es relevante para el estudio que se está llevando a cabo.

<u>Fecha</u>	<u>Hora</u>	<u>Message</u>
<u>DD MM AA</u>	<u>hh mm ss</u>	
Timestamp	ID Thread	Type
081109 203518	143	INFO dfs.DataNode\$DataXceiver: Receiving block blk_-1608999687919862906 src: /10.250.19.102:54106 dest: /10.250.19.102:50010

Figura 4.6: Sintaxis de un *log* de HDFS

También varía cómo se organiza el campo de *timestamp*. Según se ve en la Figura 4.7, este es menos preciso en comparación con los anteriores, pero sigue siendo útil para llevar a cabo el estudio. Además, también se mantiene el campo *message* con una estructura irregular.

<u>Fecha</u>	<u>Hora</u>
<u>DD MM AA</u>	<u>hh mm ss</u>
081109	203518

Timestamp

Figura 4.7: Estructura del *timestamp* de un *log* de HDFS

Normalmente, todos los *logs* de servicios software, dispositivos y otros sistemas operativos contienen los mismos campos. De hecho, para que un producto de este tipo pueda pasar una evaluación del tipo CC (*Common Criteria*) (reconocida a nivel mundial), es necesario que los registros de *logs* cumplan con el SFR FAU\_GEN.1 [50], el cual especifica el contenido mínimo que debe tener un evento registrado.

En conclusión, puede diferirse que los *datasets* de eventos son de tipo no estructurado, por lo que pueden llevarse a cabo varios enfoques. Para este trabajo se hará un acercamiento mediante diferentes métodos de *clustering* para poder detectar vectores de ataque y llevar a cabo agrupaciones de *logs* para su posterior análisis. Previamente será necesario el preprocesado de los datos para unificar su formato, extrayendo más características importantes del campo *message* junto con aquellas anteriormente mencionadas a través del uso de técnicas de *parsing*.

## 4.2. Algoritmos de *machine learning* escogidos

Existen diversos métodos que abordan la clasificación y agrupamiento de *logs* para poder detectar posibles patrones de ataque en sistemas Linux. Este Trabajo se centrará principalmente en el *clustering* [51].

En primer lugar es necesario establecer una definición para esta técnica; en base a la proporcionada por *Barreno Recio et. al* [52].

### Definición 1.

El *clustering* se refiere al método de agrupación no supervisada de un conjunto de datos  $D = \{F_1, F_2, \dots, F_n\}$  en  $k$  grupos  $C = \{C_1, C_2, \dots, C_k\}$  utilizando una métrica de similitud diseñada para maximizar la homogeneidad entre los elementos de un mismo grupo y aumentar la heterogeneidad con respecto a los elementos de los otros grupos.

De tal modo, la idea consiste en poder agrupar conjuntos de *logs* que tengan características similares, de modo que entre dichas agrupaciones se encuentren vectores de ataque distintos como DoS o fuerza bruta a servicios, como es el caso de la Figura 4.8. Normalmente cuando se producen este tipo de ataques se genera una secuencia significativamente grande de eventos prácticamente equivalentes entre sí.

```

535 2024-05-05T10:31:05+02:00 f6194a3d2d4d sshd[341]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=172.17.0.1
536 2024-05-05T10:31:05+02:00 f6194a3d2d4d sshd[344]: pam_unix(sshd:auth): check pass; user unknown
537 2024-05-05T10:31:05+02:00 f6194a3d2d4d sshd[344]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=172.17.0.1
538 2024-05-05T10:31:05+02:00 f6194a3d2d4d sshd[345]: pam_unix(sshd:auth): check pass; user unknown
539 2024-05-05T10:31:05+02:00 f6194a3d2d4d sshd[345]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=172.17.0.1
540 2024-05-05T10:31:05+02:00 f6194a3d2d4d sshd[348]: pam_unix(sshd:auth): check pass; user unknown
541 2024-05-05T10:31:05+02:00 f6194a3d2d4d sshd[348]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=172.17.0.1
542 2024-05-05T10:31:05+02:00 f6194a3d2d4d sshd[348]: pam_unix(sshd:auth): check pass; user unknown
543 2024-05-05T10:31:05+02:00 f6194a3d2d4d sshd[348]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=172.17.0.1

```

Figura 4.8: Ejemplo de secuencia de *logs* generados por fuerza bruta a SSH

Por tanto, basándonos en la definición anterior, el conjunto de datos D será un fichero que contenga una secuencia de eventos y estos serán agrupados en  $k$  grupos de tipos de eventos distintos.

### Factores de un modelo de *clustering*

Factor	Descripción
Representación de los datos	<ul style="list-style-type: none"> <li>▪ Adaptados: los datos son transformados para mejorar la estructura del modelo.</li> <li>▪ No adaptados: los datos son utilizados en su forma original.</li> </ul>
Medida de similitud	<ul style="list-style-type: none"> <li>▪ Tiempo: considerando las variaciones temporales.</li> <li>▪ Cambio: observando las diferencias entre estados.</li> <li>▪ Forma: evaluando la estructura de los datos.</li> </ul>
Algoritmo de <i>clustering</i>	<ul style="list-style-type: none"> <li>▪ Particional: K-Means, Fuzzy C-Means</li> <li>▪ Jerárquico: AHC, DHC, HDBSCAN, BIRCH, FINCH</li> <li>▪ Densidad: DBSCAN, OPTICS, DENCLUE, SNN, DenMune</li> <li>▪ Cuadrícula: STING, CLIQUE</li> </ul>
Medidas de Evaluación	<ul style="list-style-type: none"> <li>▪ Interna: KFCV, LOOCV</li> <li>▪ Externa: uso de otros <i>datasets</i> distintos al usado para el entrenamiento como HDFS, BGL o Linux_2k.</li> </ul>

Tabla 4.2: Factores de un modelo de *clustering*

### Tipos de agrupamiento en técnicas de *clustering*

Además de los distintos factores indicados en la Tabla 4.2, se puede distinguir entre dos tipos de agrupamiento para el *clustering*: el primero de ellos es el compacto, en el cual los elementos del grupo son muy parecidos y se representan a través de su punto central, mientras que el segundo es el agrupamiento encadenado [53], en el cual un elemento de un grupo es más similar a otro de los elementos del grupo que los demás, produciéndose un efecto de cadena entre los elementos a través de una especie de ruta. Se puede ver en más detalle en la Figura 4.9.

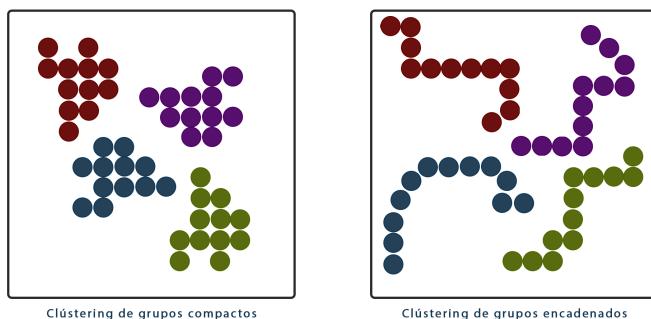


Figura 4.9: Tipos de agrupamiento en técnicas de *clustering*

### 4.2.1. K-means

Se trata de un algoritmo de *clustering* basado en partición, es decir, este divide los datos en  $k$  grupos de forma que cada grupo está compuesto por, al menos, un elemento. La forma de generar estos grupos consiste en tomar una serie de elementos que sean representativos, también conocidos como prototipos.

La dificultad principalmente reside en escoger estos elementos representativos de manera que la partición sea lo más óptima posible. Para ello se lleva a cabo un proceso de iteración dividido en dos pasos:

- Asignación: a partir de los prototipos iniciales se asignan los elementos al *cluster* del prototipo que esté a menor distancia.
- Optimización: se vuelve a repetir el mismo proceso hasta que se cumpla una condición establecida, como un máximo de iteraciones o un error.

#### Pseudocódigo del algoritmo K-means

---

##### Algorithm 1 K-means

---

**Datos:**  $D$  Agrupación de datos a amontonar,  $k$  número de clúster

**Resultado:** Total de datos que forman  $D$  agrupados en uno de los  $k$  clúster

```

1: Inicio;
2:  $centroides =$  lista de  $k$  elementos aleatorios no repetidos de  $D$ ;
3:  $clusters =$  lista de  $k$  clúster;
4: repetir
5:   para cada punto  $P$  de  $D$  hacer
6:      $centroideMasCercano = centroides[0]$ ;
7:     para  $c$  en  $centroides$  hacer
8:       si  $dist(P, c) < dist(P, centroideMasCercano)$  entonces
9:          $centroideMasCercano = c$ ;
10:      Asignar  $P$  al clúster al que pertenece  $centroideMasCercano$ ;
11:      para  $clust$  en  $clusters$  hacer
12:        Calcular Promedio del  $clust$ ;
13:        Actualizar  $centroide$ ;
14:    hasta que se cumplan un número de iteraciones, que todos los grupos cumplan una
       determinada tolerancia, que los grupos no cambien en esta iteración u otra condición
       de parada.

```

---

#### Parámetros del algoritmo K-means

Este requiere de un único parámetro  $k$ :

$$Kmeans (k) \quad (4.1)$$

que es el número de *clusters* en los que se quiere llevar a cabo el agrupamiento.

Su elección es crucial para obtener buenos resultados, es por ello que cuando no se conoce este valor es necesario aplicar una serie de heurísticas en base a la calidad de los grupos, como los gráficos de silueta [54] o el método del codo (Figura 4.10), explicado por Syakur et. al [55] en su artículo a través de los siguientes pasos:

1. Determinar el número de *clusters*  $K$  y el número máximo de iteraciones.
2. Realizar el proceso de inicialización de los centroides de los clusters. La ecuación para calcular los centroides es:

$$C_i = \frac{1}{M} \sum_{j=1}^M x_j \quad (4.2)$$

siendo  $M$  el número de datos asignados al cluster  $i$ .

3. Asignar cada dato al *cluster* más cercano utilizando la distancia euclídea:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (4.3)$$

4. Reasignar los datos a cada grupo basándose en la comparación de la distancia entre los datos y el centroide de cada grupo:

$$a_{ij} = \begin{cases} 1 & \text{si } d = \min\{D(x_i, c_i)\} \\ 0 & \text{en caso contrario} \end{cases} \quad (4.4)$$

5. Recalcular la posición del centroide del cluster. La función objetivo  $J$  utilizada por este método se basa en la distancia y en el valor de la pertenencia de los datos al grupo:

$$J = \sum_{i=1}^n \sum_{l=1}^k a_{il} D(x_i, c_l)^2 \quad (4.5)$$

donde  $n$  es la cantidad de datos,  $k$  es el número de grupos,  $a_{il}$  es el valor de la pertenencia del punto de datos  $x_i$  al grupo  $c_l$ .

6. Si hay un cambio en la posición del centroide del cluster o en el número de iteraciones, volver al paso 3. Si no, devolver el resultado del *clustering*.

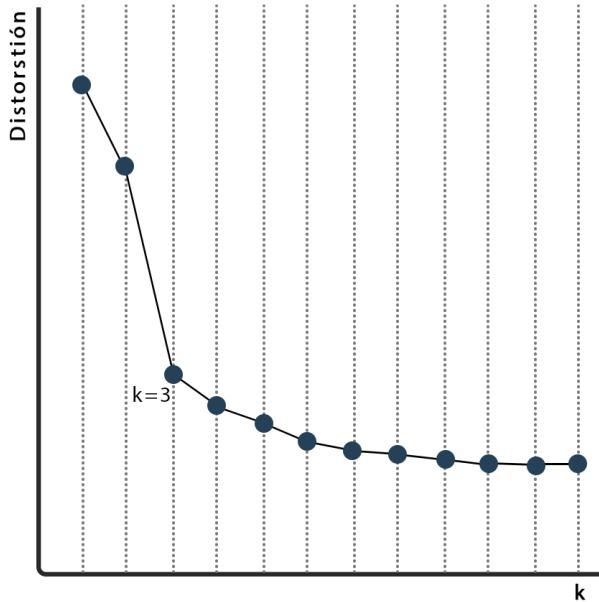


Figura 4.10: Método del codo para la elección del parámetro  $k$

### Orden de eficiencia de K-means

Su orden de eficiencia es:

$$O(nkdi) \quad (4.6)$$

donde  $n$  es el número de puntos de datos,  $k$  es el número de *clusters*,  $d$  es la dimensión de los datos y  $i$  es el número de iteraciones hasta la convergencia.

### Librería sklearn para la implementación de K-means

Puede aprovecharse la implementación del algoritmo K-means de la librería scikit-learn de Python [56], la cual permite la normalización, ajuste y evaluación del modelo, así como la visualización de los resultados con un gran margen de personalización.

Consta de los siguientes parámetros de entrada y atributos de salida:

Tipo	Nombre	Descripción
<b>Parámetros de entrada</b>		
Parámetro	n_clusters	Número de grupos en los que se dividen los datos.
Parámetro	init	Método que determina los centroides iniciales; permite las siguientes opciones: <ul style="list-style-type: none"> <li>■ <b>k-means++</b> [57]: método por defecto.</li> <li>■ <b>random</b>: selecciona los centroides aleatoriamente.</li> <li>■ Matriz de centroides iniciales: se pueden pasar directamente.</li> </ul>
Parámetro	n_init	Número de repeticiones del algoritmo con diferentes centroides iniciales para obtener la mejor agrupación.
Parámetro	max_iter	Número máximo de iteraciones para un conjunto dado de centroides iniciales.
<b>Atributos de salida</b>		
Atributo	cluster_centers_	Array con los centroides de cada grupo, de dimensiones $(k, m)$ .
Atributo	labels_	Etiquetas de los elementos del conjunto agrupado, donde cada etiqueta $i$ corresponde al centroide $i$ en cluster_centers_.
Atributo	inertia_	Suma de las distancias al cuadrado de cada elemento a su centroide más cercano.
Atributo	n_iter_	Número de iteraciones realizadas.

Tabla 4.3: Parámetros y atributos del algoritmo K-Means en sklearn

#### 4.2.2. DBSCAN

Es el algoritmo de *clustering* basado en densidad más utilizado en la actualidad [58] gracias a que no hace falta tener una gran base de conocimiento sobre los datos que se manejan y su uso es compatible con *datasets* de gran tamaño.

Conceptualmente, DBSCAN busca zonas que tengan una densidad alta y que estén alejadas de otras con una densidad más baja. Para ello se utilizan los términos *neighbor*, que es un punto dentro del radio  $\epsilon$  de otro punto, y *neighborhood*, que es el conjunto de todos los puntos dentro del radio  $\epsilon$  de un punto específico, incluyendo al punto mismo.

##### Definición 2.

El *neighborhood* de un punto  $p$  conocido como  $N(p)$  puede definirse como:

$$N(p) = \{q \in D \mid \text{dist}(p, q) \leq \epsilon\}$$

donde  $D$  es el conjunto de datos,  $\text{dist}(p, q)$  es la distancia entre los puntos  $p$  y  $q$ , y  $\epsilon$  es el umbral de distancia.

#### Pseudocódigo del algoritmo DBSCAN

##### Algorithm 2 DBSCAN

**Datos:**  $D$  conjunto de datos a agrupar,  $\text{eps}$  distancia máxima entre dos puntos vecinos,  $\text{minPts}$  mínimo número de puntos para formar un *cluster*

**Resultado:** Datos agrupados en clusters

```

1: Inicio;
2:  $C = 0$ ;
3: para cada punto  $P$  de  $D$  hacer
4:   si  $P$  no está en visitados entonces
5:     Marcar  $P$  como visitado;
6:      $ptosVecinos = calcularRegion(P, \text{eps})$ ;
7:     si tamaño( $ptosVecinos$ ) <  $\text{minPts}$  entonces
8:       marcar  $P$  como ruido;
9:     si no
10:       $C = \text{próximocluster}$ ;
11:      expandirCluster( $P, ptosVecinos, C, \text{eps}, \text{minPts}$ );
12:    fin

```

Como se puede observar, se inicia desde un punto aleatorio y se verifica si en el radio  $\text{eps}$  de ese punto existe un mínimo de puntos  $P$ . Si es así, se forma un *cluster* y se vuelve a la primera etapa con uno de los puntos encontrados. Si no se alcanza el mínimo de puntos pero se ha llegado a través de un punto que sí cumplía esta condición, este formará parte del mismo *cluster*. En caso de que no se pueda llegar al punto por medio de otros, no formará parte del *cluster* y se clasificará como un nodo de tipo *noise*.

### Parámetros del algoritmo DBSCAN

Los dos parámetros utilizados por este algoritmo son:

$$DBSCAN(\epsilon, MinPts) \quad (4.7)$$

donde

- $\epsilon$  es el radio máximo de un vecindario de un punto, es decir, la distancia máxima que se considera para definir los puntos vecinos.
- $MinPts$  es el número mínimo de puntos requeridos para formar un punto **core**. Un punto se considera un punto *core* si hay al menos  $MinPts$  puntos dentro de su vecindario (incluyendo el mismo punto).

### Clasificación de puntos en un *cluster* de DBSCAN

Los puntos se clasifican en tres tipos posibles, tal y como se describe en la Tabla 4.4 y se ilustra en la Figura 4.11.

Tipo	Descripción
<b>core</b>	Puntos interiores de un <i>cluster</i> con un vecindario de radio $\epsilon$ .
<b>border</b>	Tienen menos de $MinPts$ puntos en su vecindario de radio Epsilon, estando en el vecindario de algún punto <b>core</b> .
<b>noise</b>	Cualquier punto que no forma parte del <b>core</b> de un <i>cluster</i> ni está en su frontera o <b>border</b> .

Tabla 4.4: Clasificación de puntos en DBSCAN

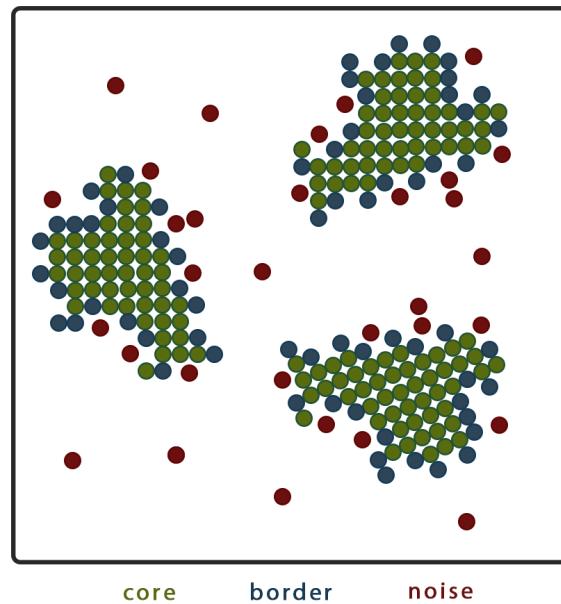


Figura 4.11: Ejemplo de clasificación de puntos de algoritmo de *clustering* DBSCAN

### Orden de eficiencia de DBSCAN

Su orden de eficiencia es:

$$O(n \log n) \quad (4.8)$$

donde  $n$  es el número de puntos de datos.

#### 4.2.3. Clustering Jerárquico

También conocido como *Agglomerative Hierarchical Clustering* [51], este algoritmo jerárquico que agrupa datos de manera aglomerativa. Es decir, comienza tratando cada punto de datos como un clúster individual y, en cada paso, combina los dos clústeres más cercanos hasta que todos los puntos de datos pertenecen a un solo clúster.

El proceso del algoritmo se basa en la siguiente serie de pasos:

Paso	Descripción
1	Calcular la matriz de disimilitud entre todos los puntos de datos.
2	Repetir los siguientes pasos hasta que quede un solo clúster:
3	Fusionar los <i>clusters</i> más cercanos $C_a$ y $C_b$ . El nuevo <i>cluster</i> tendrá una cardinalidad $N_{a\cup b} = N_a + N_b$ .
4	Insertar una nueva fila y columna en la matriz de disimilitud que contenga las distancias entre el nuevo <i>cluster</i> $C_{a\cup b}$ y los <i>clusters</i> restantes.
5	Terminar cuando solo quede un <i>cluster</i> máximo.

Tabla 4.5: Pasos del algoritmo de Clustering Jerárquico Aglomerativo

### Pseudocódigo del algoritmo AHC

---

#### Algorithm 3 AHC

---

**Datos:**  $\mathbf{D}$  Agrupación de datos a amontonar

**Resultado:** Datos agrupados en un solo *cluster* jerárquico

- 1: **Inicio;**
  - 2: Calcular la matriz de disimilitud entre todos los puntos de datos;
  - 3: **repetir**
  - 4:     Fusionar los *clusters* más cercanos  $C_a \cup C_b$ . Establecer la cardinalidad del nuevo *cluster* como  $N_{a\cup b} = N_a + N_b$ ;
  - 5:     Insertar una nueva fila y columna que contenga las distancias entre el nuevo clúster  $C_{a\cup b}$  y los *clusters* restantes;
  - 6: **hasta que** solo quede un *cluster* máximo.
- 

### Parámetros del algoritmo AHC

A diferencia de los anteriores, este algoritmo no requiere parámetros explícitos como K-means o DBSCAN. En su lugar, la principal decisión que hay que tomar es la métrica de disimilitud <sup>1</sup>utilizada para calcular las distancias entre *clusters*, que puede ser:

- Distancia simple (*single linkage*): La distancia mínima entre puntos de diferentes *clusters*.
- Distancia completa (*complete linkage*): La distancia máxima entre puntos de diferentes *clusters*.
- Distancia promedio (*average linkage*): El promedio de todas las distancias entre puntos de diferentes *clusters*.

### Orden de eficiencia de AHC

El orden de eficiencia del algoritmo AHC es:

$$O(n^3) \quad (4.9)$$

donde  $n$  es el número de puntos de datos. Este orden de eficiencia se debe al cálculo repetido de distancias entre todos los pares de clústeres a medida que se fusionan durante la construcción del árbol jerárquico.

Cuando se habla de *clustering* jerárquico, debe definirse qué es un dendrograma. Se trata de una representación gráfica de los resultados del *clustering* tal que en el eje vertical se muestra la distancia o disimilitud en la que se combinan los *clusters*, mientras que en el eje horizontal se encuentran los puntos de datos, como se muestra en la Figura 4.12. Esta visualización permite identificar la estructura del *cluster* y decidir el número óptimo de *clusters* al cortar el dendrograma a una altura específica.

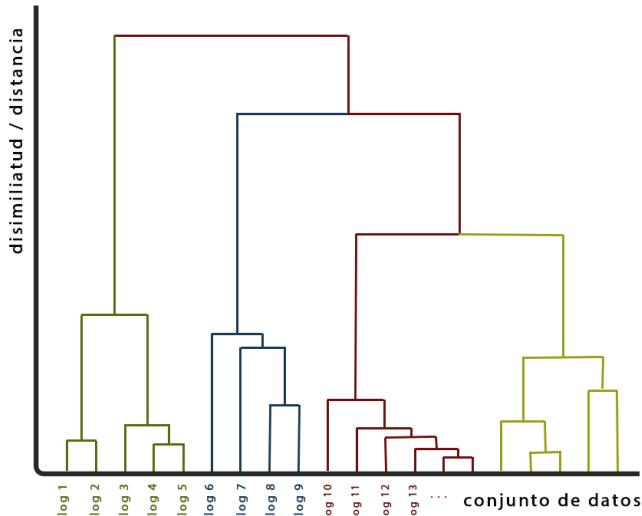


Figura 4.12: Ejemplo de un dendrograma para AHC

---

<sup>1</sup>Es una medida que cuantifica cuán diferentes son dos elementos. En el contexto del *clustering*, se utiliza para calcular la distancia entre puntos de datos o *clusters*.

### 4.3. *Set-up experimental: tecnologías y herramientas*

Para llevar a cabo la implementación del proyecto con la mayor precisión posible, se ha llevado a cabo un proceso minucioso de selección de las tecnologías y herramientas que se iban a utilizar.

#### 4.3.1. Herramientas para la simulación de ataques

En primer lugar, ha sido necesario seleccionar uno de los múltiples *frameworks* existentes basados en el marco adversario de MITRE ATT&CK para llevar a cabo la simulación de ataques a las máquinas y la posterior generación de grandes cantidades de *logs* que pudiesen ser procesadas para generar un *dataset*. Complementariamente, se han utilizado otras herramientas de explotación para enriquecer la cantidad de eventos generados a raíz de distintos ataques.

#### Framework CALDERA

El *framework* escogido ha sido **CALDERA** [34], ya que permite emular la totalidad de técnicas y tácticas contempladas por el marco de MITRE, además de ofrecer una gran personalización. La versión utilizada ha sido la v5.0.0 "Magma" [59], publicada el pasado 14 de febrero. A diferencia de las alternativas propuestas en la segunda sección del Estado del Arte (3.2.3), esta ha sido directamente desarrollada por MITRE, lo que la convierte en la solución opensource más estandarizada y utilizada en el entorno empresarial.

Su funcionamiento está basado en la combinación de dos sistemas: ViRTS, una infraestructura software usada para crear y emular adversarios de *red-team*, y un modelo lógico denominado LAVA que se encarga de determinar qué acciones adversarias llevar a cabo.

Por otro lado, la infraestructura de CALDERA, instanciada por ViRTS, se compone de dos elementos principales: el servidor maestro (ExtroViRTS) y los clientes de herramientas de acceso remoto (RAT) en hosts ya infectados (IntroViRTS), como se observa en la Figura 4.13.

El proceso consta de los siguientes pasos [3]:

1. **Configuración inicial:** CALDERA se configura de modo que un único host esté infectado con un RAT de IntroViRTS, asegurando que la comunicación entre este RAT y el servidor maestro ExtroViRTS sea fluida.
2. **Ampliación del control:** A partir de la infección inicial, CALDERA utiliza el motor LAVA para determinar las acciones a tomar. El servidor maestro ejecuta una instancia de LAVA para seleccionar una acción específica a ejecutar.
3. **Ejecución de acciones:** El servidor maestro envía un comando a un RAT específico en el campo, que ejecuta la acción seleccionada.
4. **Retroalimentación y actualización:** El RAT envía todos los detalles relevantes de la acción ejecutada de vuelta al servidor maestro.

5. **Actualización de la base de conocimientos:** El servidor ExtroViRTS lleva a cabo la actualización de su base de conocimientos interna.
6. **Selección de futuras acciones:** El servidor maestro continúa utilizando LAVA para seleccionar futuras acciones que serán ejecutadas por los clientes IntroViRTS.

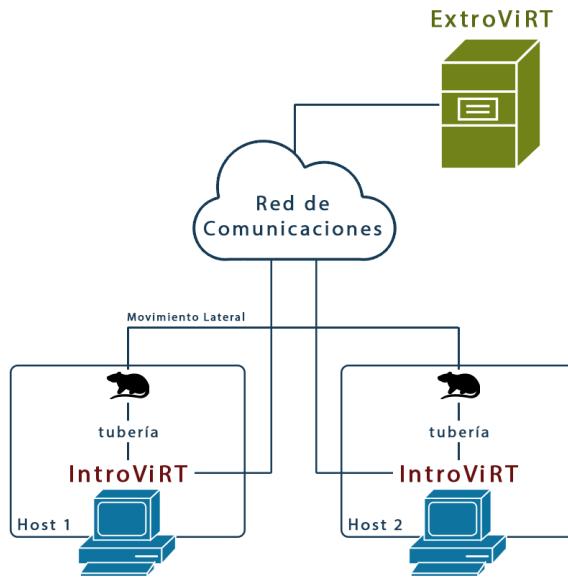


Figura 4.13: Infraestructura del framework CALDERA [3]

Este *framework* emplea una terminología específica, cuyo entendimiento es vital para poder llevar a cabo un ataque. Como se muestra en la Tabla 4.6, distinguimos entre cuatro elementos principales:

Término	Descripción
Agentes	Aplicación software que se conecta de a CALDERA en intervalos regulares para obtener instrucciones. Cada agente se comunica con el servidor de CALDERA y se le asigna un ID único llamado <i>paw</i> . Algunos ejemplos de agentes disponibles son Sandcat, Manx y Ragdoll.
Habilidades	Tácticas, técnicas y procedimientos específicos de ATT&CK que se ejecutan en agentes activos. Cada habilidad incluye los comandos a ejecutar, las plataformas/ejecutores para el comando, cargas útiles y una referencia a los resultados de salida en el servidor de CALDERA.
Adversarios	Perfiles de amenazas predefinidos que consisten en un grupo de habilidades (TTPs). Los perfiles pueden ayudar a determinar qué habilidades deberían ejecutarse al llevar a cabo una operación.
Operaciones	Ejecutan habilidades en grupos de agentes.

Tabla 4.6: Terminología utilizada por el *framework* de CALDERA

En el próximo capítulo se documentará el proceso seguido para llevar a cabo la instalación y configuración del *framework*.

### Herramientas de ataque adicionales

CALDERA, según lo comentado anteriormente, basa su funcionamiento en la infección del host atacado mediante un *payload* conocido como agente, es por ello que todos los ataques generados por este *framework* están orientados a reconocimiento, movimiento lateral y otras operaciones de explotación y post-exploitación. Por consiguiente, con la finalidad de aumentar la variedad de eventos recopilados en el *dataset*, se hará uso de las siguientes herramientas de *hacking* ético:

- **Metasploit** [60]: se trata del *framework* más utilizado para ejercicios de *pentesting*, es ampliamente conocido por facilitar la automatización de una amplia gama de ataques asociados a CVEs. Cuenta con una gran base de datos, y su popularidad debe a la facilidad de uso para realizar ataques realistas y eficaces en entornos controlados.
- **Hydra** [61]: es una herramienta de *cracking* de contraseñas que destaca por su velocidad a través de opciones de ataques concurrentes mediante hebras, y su versatilidad, soportando múltiples protocolos de red. Es altamente efectiva en técnicas de fuerza bruta y ataques por diccionario.

#### 4.3.2. Virtualización de las máquinas víctima

Para llevar a cabo la simulación de ataques, se investigó acerca de qué tecnología utilizar de cara al despliegue de *hosts* que simularan una infraestructura operacional que hubiera sido vulnerada y tuviera un servidor central que almacenara internamente todos los eventos registrados. Las dos principales opciones eran las siguientes:

Tecnología	Descripción
Máquinas virtuales	Ofrecen un aislamiento completo del sistema operativo subyacente, lo que resulta en una mayor seguridad y estabilidad. Sin embargo, consumen más recursos y requieren más tiempo para configurarse y mantenerse en comparación con los contenedores.
Contenedores Docker	Permiten un despliegue rápido y eficiente, consumiendo menos recursos que las máquinas virtuales. Además, facilitan la portabilidad entre diferentes sistemas y plataformas. Sin embargo, ofrecen un aislamiento menor que las máquinas virtuales.

Tabla 4.7: Comparación de tecnologías de virtualización para la simulación de ataques

Una tercera opción sería el uso de WSL para realizar desde ahí la simulación de ataques, ya que desde el lanzamiento de la versión 2.0 se han implementado numerosas mejoras a nivel de aislamiento con respecto a Windows, y al mismo tiempo proporcionando cada vez una mayor funcionalidad, que previamente estaba muy limitada en la versión 1.0.

Esto fue analizado el pasado 18 de mayo de 2024 en Hack-én por el investigador Sergio De Los Santos en su ponencia titulada *WSL: Windows Subsystem for Linux. Del odio al amor y del amor al odio*. En su libro recientemente publicado [62] puede obtenerse mucha más información sobre el tema.

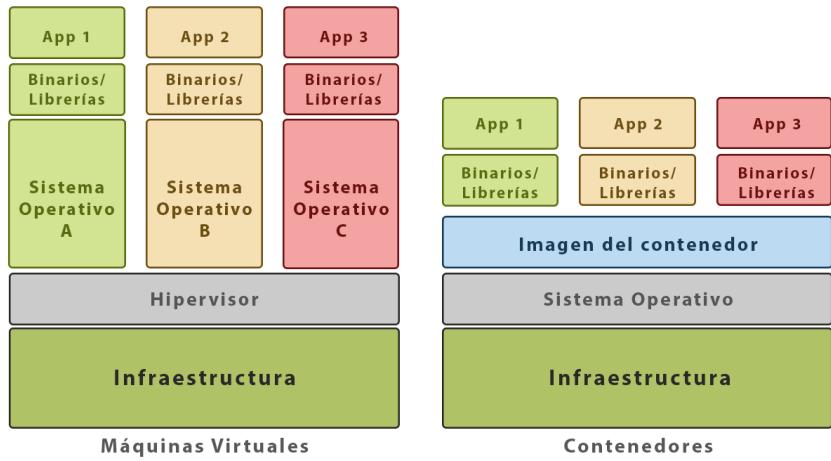


Figura 4.14: Comparativa estructural de Máquinas Virtuales vs Contenedores

Finalmente, de las dos opciones ilustradas en la Figura 4.14 ha sido elegida la versión más ligera y simple: los contenedores [63]. Para el contexto de la simulación de ataques no suponía un problema que hubiese un menor aislamiento. Se llevaron a cabo pruebas de funcionalidad y rendimiento utilizando dos tipos de contenedores distintos.

El primero de ellos fue Alpine Linux [64], que era la opción más ligera y rápida de generar a través de un Dockerfile, sin embargo tenía notables limitaciones y no representaba un entorno realista, ya que no se suele utilizar para servidores. La segunda opción fue un contenedor de Ubuntu Server [65], que ha resultado ser la elección definitiva, ya que sigue siendo una alternativa ligera pero con una utilidad suficiente como para llevar a cabo la simulación de ataques, y además es una distribución muy utilizada en cuanto a ingeniería de servidores respecta.

En el próximo capítulo se detallará cómo se ha llevado a cabo la implementación de los ficheros Dockerfile para levantar dichos contenedores.

#### 4.3.3. Librerías de Python utilizadas

Python es actualmente el principal lenguaje de programación de alto nivel utilizado para cualquier actividad relacionada con análisis de datos, ML o DL [66]. Es por ello que existen multitud de librerías que ofrecen una capa de abstracción para hacer uso de funcionalidades que se basan en el álgebra lineal y su implementación desde cero tendría un nivel de complejidad exponencial.

### Librerías para el preprocesado de *datasets*

En primer lugar, para llevar acabo el preprocesado de *datasets*, se han utilizado las siguientes librerías:

- **Pandas:** Esta librería es una de las más conocidas en el análisis de datos. Proporciona estructuras de datos flexibles y potentes, conocidas como DataFrames, que facilitan la limpieza, transformación y análisis de grandes volúmenes de datos. Pandas permite cargar datos desde diversos formatos (CSV, Excel, SQL, etc.) y realizar operaciones complejas de agrupación, filtrado y agregación de manera eficiente.
- **RE:** El módulo *re* de Python proporciona soporte para expresiones regulares. Este permite buscar, sustituir y dividir cadenas de texto basándose en patrones específicos, lo cual resulta especialmente útil para el procesamiento de campos de *logs*.
- **CSV:** La librería *csv* se utiliza para leer y escribir archivos en formato CSV (Comma-Separated Values). Este formato es el más utilizado para el análisis de datos y es compatible con muchas herramientas de análisis como las utilizadas en este proyecto para la implementación del *clustering*.
- **io:** El módulo *io* de Python sirve para trabajar con flujos de datos en memoria, permitiendo la lectura y escritura de datos en diversos formatos sin necesidad de interactuar directamente con el sistema de archivos. Esto es útil para la manipulación temporal de datos durante el preprocesado.

El uso de estas librerías ha sido más que suficiente para realizar el preprocesado de los *datasets* en un formato inicial de *.log*, ya que han permitido limpiar y posteriormente transformar los datos de manera eficiente de cara a su posterior análisis.

### Librerías para la implementación del *clustering*

En segundo lugar, para la implementación de técnicas de *clustering* en Python, se ha hecho uso de varias librerías que ofrecen un uso a alto nivel de algoritmos utilizados para la agrupación de datos como los estudiados anteriormente 1. Estas son:

- **Scikit-learn (sklearn):** Esta librería es una de las más populares para el aprendizaje automático en Python. Incluye una amplia gama de algoritmos de *clustering*, como K-means, DBSCANs y *clustering* jerárquico. Scikit-learn proporciona una interfaz sencilla y consistente para entrenar, evaluar y ajustar modelos, lo que facilita enormemente la integración de técnicas de *clustering* en *pipelines* de procesamiento de datos.
- **NumPy:** Ofrece una poderosa estructura de datos en forma de arrays y matrices multidimensionales, y además permite realizar cálculos numéricos de forma eficiente sobre grandes volúmenes de datos, facilitando así la implementación de algoritmos de *clustering* y otras técnicas de procesamiento de datos.

- **Matplotlib**: Aunque no es una librería de *clustering*, *matplotlib* es fundamental para la visualización de los resultados del *clustering*. Permite crear gráficos Y por tanto visualizar los resultados de los algoritmos de agrupación de cara a su análisis.
- **Scipy**: Esta librería complementa a *scikit-learn* proporcionando funciones adicionales para el análisis y manipulación de datos. Incluye herramientas para la implementación de *clustering* jerárquico y otras técnicas similares.
- **UMAP-learn**: Tal y como indica su nombre, implementa la técnica de reducción de dimensionalidad que también se puede utilizar para el *clustering*. UMAP es especialmente útil para visualizar datos de alta dimensionalidad y descubrir estructuras subyacentes en los datos. Esto ha optimizado considerablemente la implementación del proyecto.
- **Tabulate**: Facilita la creación de tablas con diferentes estilos de formato de manera muy sencilla y rápida a partir de listas de datos o estructuras similares. Esta soporta texto simple, HTML, LaTeX, y Markdown, lo cual resulta especialmente de cara al análisis de resultados.

El uso combinado de estas librerías ha permitido implementar de manera eficiente y robusta diversas técnicas de *clustering*, que luego han sido utilizadas en los *datasets* de *logs* generados, para llevar a cabo el estudio de vectores de ataque.

---

En la Figura 4.15 se muestra el diagrama estructural utilizado para llevar a cabo este Trabajo.

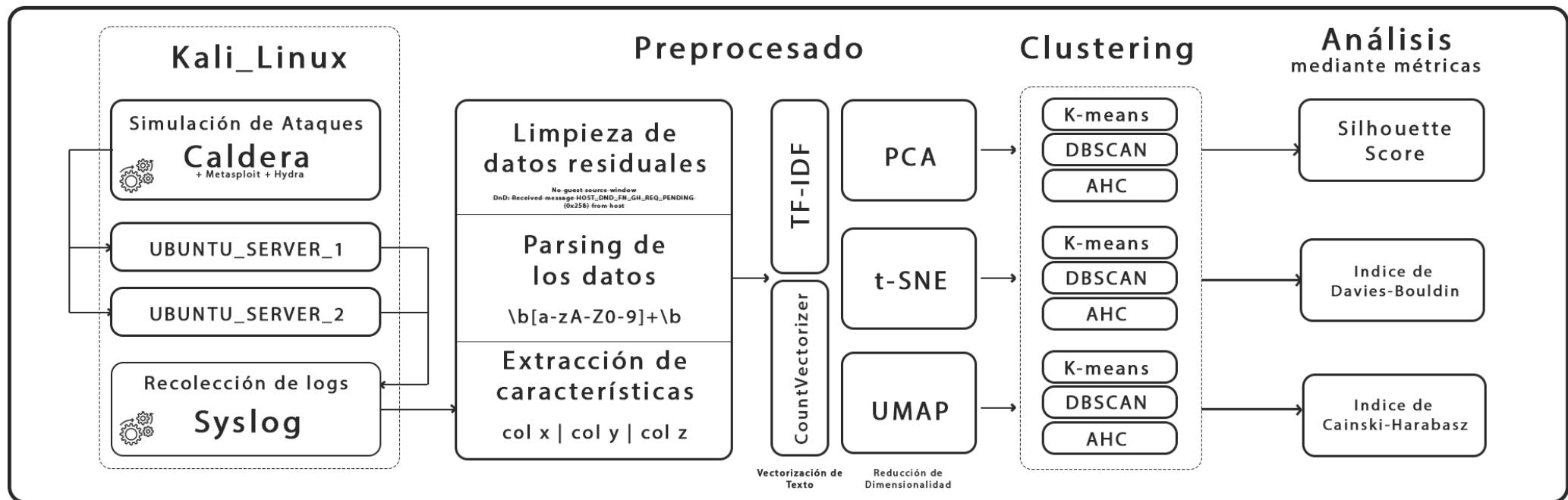


Figura 4.15: Diagrama estructural del Trabajo de Fin de Grado



# Capítulo 5

## Desarrollo e implementación

En este capítulo se abordará cómo se ha llevado a cabo el proceso de implementación del proyecto, partiendo desde la preparación del entorno para la realización de ataques y posterior generación del *dataset* de logs, siguiendo por cómo se ha preprocesado dicho *dataset* y otros extraídos de fuentes públicas para la comparación de resultados, y finalmente la implementación de las distintas técnicas de *clustering* y la selección de métricas para evaluar los resultados obtenidos.

### 5.1. Simulación de ataques y generación de logs

A continuación, se verá en detalle cómo se han desarrollado los conjuntos de datos de *logs* de Linux. En primer lugar, se ha realizado un *dataset* de manera totalmente manual, y finalmente se ha hecho uso de un LLM para la generación de un *dataset* sintético y ya estructurado que simula dichos *logs*.

#### 5.1.1. Dataset Manual

Como se ha comentado en capítulos anteriores, para la simulación de ataques se hará uso del framework de CALDERA [34]. Antes de mostrar este proceso paso a paso, es importante recalcar que toda la implementación de esta sección se hará sobre el sistema operativo Kali Linux 6.5.0-kali3-amd64 Debian GNU/Linux tal y como se indica en la Figura 5.1, aunque podría ser emulado de forma equivalente en otras distribuciones de Linux.

```
(kali㉿kali)-[~]
└─$ uname -a
Linux kali 6.5.0-kali3-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.5.6-1kali1 (2023-10-09) x86_64 GNU/Linux
```

Figura 5.1: Distribución de Linux utilizada para la Simulación de ataques

Sobre este, colgarán 2 contenedores Docker con la distribución de Ubuntu Server que serán utilizados como máquinas víctima para llevar a cabo los ataques. Además del uso de este *framework* de MITRE, se hará uso de otros servicios como Syslog, SFTP o SSH. La máquina Kali actuará como servidor central y por tanto los *logs* de las máquinas que son objeto de ataque tendrán un flujo de salida hacia este, de

modo que a través de la configuración de Syslog se irá formando un fichero log que dará forma al *dataset* en formato *raw*. Al mismo tiempo, por simplicidad, actuará como máquina atacante dentro de la misma subred que simula la infraestructura empresarial.

### Configuración del Servidor Syslog y Máquinas Víctima

Con el fin de llevar a cabo la emulación de ataques sobre los contenedores y que los eventos generados se almacenaran en un único fichero, fue necesario diseñar la arquitectura del entorno, incluyendo los servicios utilizados, ficheros de configuración necesarios, así como el orden a seguir para preparación. En primer lugar, se definió la arquitectura de modo que las máquinas involucradas fueran:

Máquina	Descripción
KALI_LINUX	Máquina principal con el framework de CALDERA instalado y servidor de Syslog para monitorizar los logs. Virtualizado con Oracle VM VirtualBox.
UBUNTU_SERVER.1	Primera máquina víctima, contenedor de Docker levantado sobre máquina Kali.
UBUNTU_SERVER.1	Segunda máquina víctima, contenedor de Docker levantado sobre máquina Kali.

Tabla 5.1: Máquinas utilizadas para la simulación de ataques y recolección de *logs*

Cada una de las máquinas reenviará el flujo de *logs* al servidor de Syslog ubicado en la máquina Kali. De tal modo, será posible almacenar en un mismo fichero los *logs* de cada una, como se ilustra en la Figura 5.2.

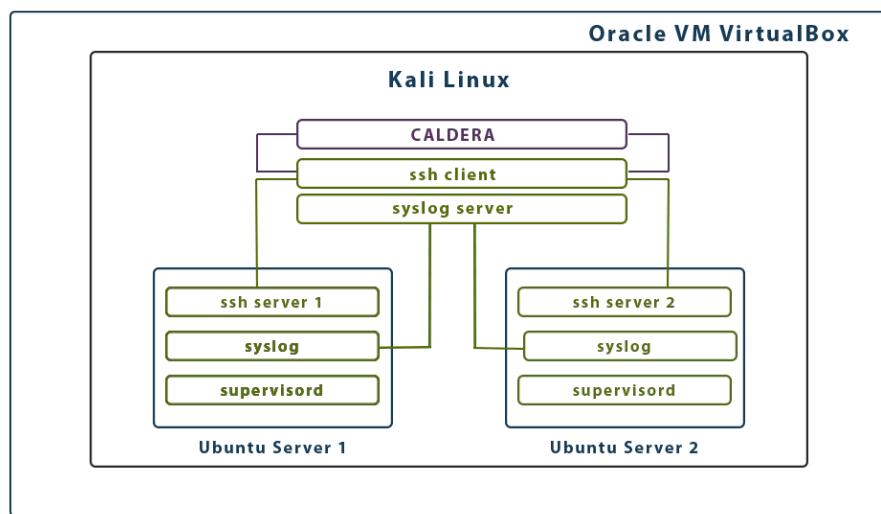


Figura 5.2: Arquitectura del entorno de simulación de ataques

Se realizaron los siguientes pasos:

1. Acceder a la ruta donde se ubica el fichero Dockerfile y ejecutar el siguiente comando para construir la imagen `ubuntu-logger`:

```
docker build -t ubuntu-logger .
```

2. Crear las máquinas a partir de la imagen generada:

```
docker run -d --name ubuntu-logger-1 ubuntu-logger
docker run -d --name ubuntu-logger-2 ubuntu-logger
```

3. Iniciar una shell en ambas máquinas:

```
docker exec -it ubuntu-logger-1 /bin/bash
docker exec -it ubuntu-logger-2 /bin/bash
```

Para construir las imágenes de los contenedores de Ubuntu Server [65] se utilizó el siguiente fichero Dockerfile:

```
# Se utiliza la imagen oficial de Ubuntu
FROM ubuntu:latest

# Evita preguntas al instalar paquetes
ARG DEBIAN_FRONTEND=noninteractive

# Actualiza los repositorios
RUN apt-get update
#Instala rsyslog, nano, iproute2, openssh-server y supervisor
RUN apt-get install -y rsyslog nano iproute2 curl openssh-server \
supervisor

# Configura rsyslog para enviar logs a máquina host y desactivar imklog
RUN sed -i 's/module(load="imklog")/#module(load="imklog")/' \
/etc/rsyslog.conf && echo '.* @@172.17.0.1:514' >> /etc/rsyslog.conf

# Configura SSH para permitir acceso root con contraseña
RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' \
/etc/ssh/sshd_config && echo 'root:root' | chpasswd

# Configuración de supervisord
COPY supervisord.conf /etc/supervisor/conf.d/supervisord.conf

# Expone los puertos necesarios para syslog y SSH
EXPOSE 514 22

# Configura supervisord como el comando principal
CMD ["/usr/bin/supervisord"]
```

A continuación, es necesario editar el fichero `supervisord.conf` en los contenedores. Este, utilizado para la configuración de `supervisord`, gestiona los procesos del sistema, asegurando que los servicios definidos se inicien y mantengan en ejecución, concretamente `ssh` y `rsyslog`.

```
[supervisord]
nodaemon=true

[program:sshd]
command=/usr/sbin/sshd -D

[program:rsyslogd]
command=/usr/sbin/rsyslogd -n
```

Por otro lado, el fichero de configuración `rsyslog.conf` utilizado en la máquina Kali Linux define cómo se manejan los registros de las máquinas Ubuntu Server 1 y Ubuntu Server 2.

Este permite la recolección y almacenamiento de *logs* provenientes de estas, que están dentro de la misma subred. Dicho fichero debe de tener activos los módulos de TCP y UDP para que la transferencia de *logs* se realice correctamente:

```
# /etc/rsyslog.conf configuration file for rsyslog

module(load="imuxsock")
module(load="imklog")

module(load="imudp")
input(type="imudp" port="514")

module(load="imtcp")
input(type="imtcp" port="514")
```

Además, es necesario añadir la siguiente línea al final del fichero para que almacene todos los tipos de ficheros de *log* en un único archivo `dataset-raw.log`:

```
.*.* /var/log/dataset-raw.log
```

Finalmente, es necesario que el fichero `rsyslog.conf` de los contenedores tenga el siguiente contenido, para que se envíen los *logs* a la máquina Kali:

```
# /etc/rsyslog.conf configuration file for rsyslog

module(load="imuxsock") # provides support for local system logging
module(load="imklog" permitnonkernelfacility="on")

$IncludeConfig /etc/rsyslog.d/*.conf
```

Es necesario añadir al final del fichero las siguientes líneas:

```
*.* @@172.17.0.1:514
cron,kern,mail,user,auth,authpriv.* @@172.17.0.1:514
```

Estas especifican la redirección de los *logs* generados en los contenedores hacia el servidor de syslog centralizado, en este caso, la máquina Kali Linux con la dirección IP 172.17.0.1 a través del puerto 514.

La primera línea indica que todos los mensajes de *log*, independientemente de su nivel de severidad o su origen, deben ser enviados al servidor central. Por otro lado, la segunda, refuerza esta configuración para logs específicos generados por las *facilities* 3.5 cron, kern, mail, user, auth, y authpriv.

### Simulación de Ataques con CALDERA

Una vez se ha instalado y configurado correctamente el *framework* de CALDERA siguiendo la documentación del Anexo C, puede llevarse a cabo la simulación de ataques sobre los contenedores UBUNTU\_SERVER\_1 y UBUNTU\_SERVER\_2. Dentro de la interfaz web del *framework*, simplemente basta con acceder a la sección de operaciones y hacer click sobre el botón de **Start** para iniciar la operación. La principal ventaja de este *framework* es que la interfaz proporciona una monitorización en tiempo real de la ejecución de las operaciones, de modo que puedan verse los resultados que van obteniéndose, así como el orden de las tácticas y técnicas empleadas en dicho proceso de ataque.

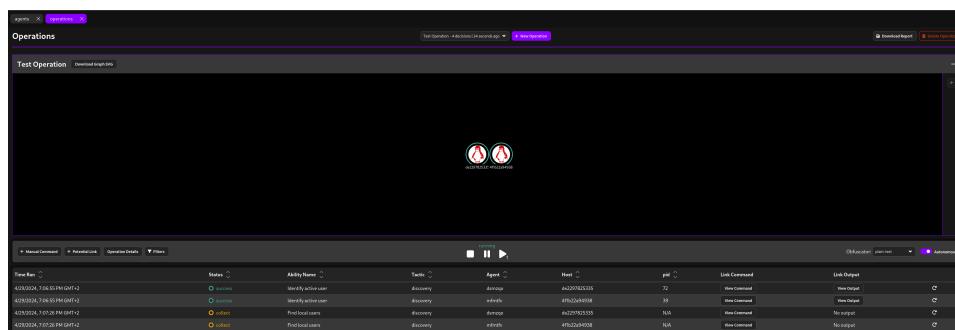
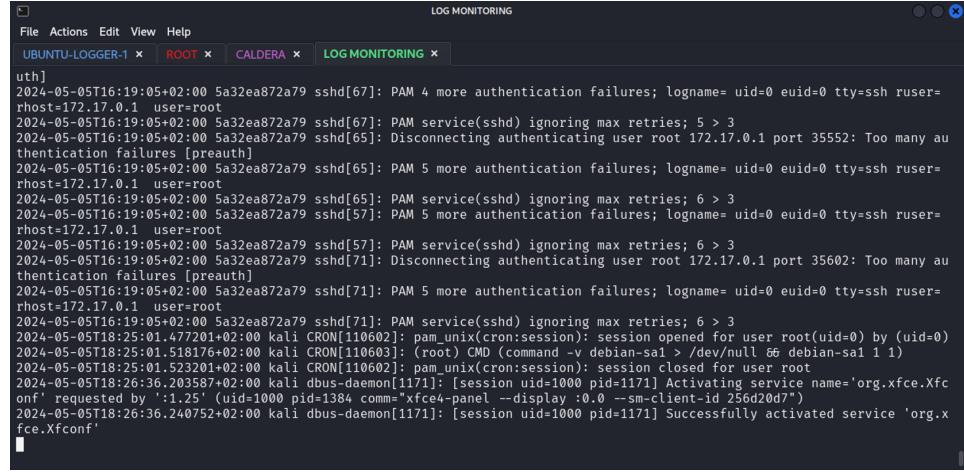


Figura 5.3: Ejemplo de ejecución de una operación de Discovery con CALDERA

Tal y como se muestra en la Figura 5.3, el panel de operaciones una vez iniciado el ataque se divide en dos zonas. En primer lugar, la zona superior muestra las máquinas que están siendo atacadas y el tipo de sistema operativo de estas. En este caso son los dos contenedores de Ubuntu Server. En segundo lugar, la zona inferior detalla la información relativa al ataque, como es el tiempo de ejecución, el estado de la operación (p.e. *success* o *collect*), el nombre de la habilidad asociada, la táctica utilizada, el agente asociado al host, el identificador del host, el ID del proceso e incluso el comando utilizado y su *output* generado.

Al mismo tiempo que se llevaban a cabo las operaciones, se fueron almacenando los eventos de *logs* en la máquina víctima y enviándose de forma síncrona al servidor Syslog de la máquina KALI LINUX, como se muestra en la Figura 5.4.



The screenshot shows a window titled "LOG MONITORING" with three tabs: "UBUNTU-LOGGER-1", "ROOT", and "CALDERA". The "LOG MONITORING" tab is active, displaying log entries from the "sshd" service. The logs show multiple failed SSH login attempts from the IP address 172.17.0.1, where the user "root" is trying to log in. The logs also mention cron jobs running as root and dbus-daemon activating services.

```

LOG MONITORING
File Actions Edit View Help
UBUNTU-LOGGER-1 × ROOT × CALDERA × LOG MONITORING ×
uth]
2024-05-05T16:19:05+02:00 5a32ea872a79 sshd[67]: PAM 4 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser=
rhost=172.17.0.1 user=root
2024-05-05T16:19:05+02:00 5a32ea872a79 sshd[67]: PAM service(sshd) ignoring max retries; 5 > 3
2024-05-05T16:19:05+02:00 5a32ea872a79 sshd[65]: Disconnecting authenticating user root 172.17.0.1 port 35552: Too many au-
thentication failures [preauth]
2024-05-05T16:19:05+02:00 5a32ea872a79 sshd[65]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser=
rhost=172.17.0.1 user=root
2024-05-05T16:19:05+02:00 5a32ea872a79 sshd[65]: PAM service(sshd) ignoring max retries; 6 > 3
2024-05-05T16:19:05+02:00 5a32ea872a79 sshd[57]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser=
rhost=172.17.0.1 user=root
2024-05-05T16:19:05+02:00 5a32ea872a79 sshd[57]: PAM service(sshd) ignoring max retries; 6 > 3
2024-05-05T16:19:05+02:00 5a32ea872a79 sshd[71]: Disconnecting authenticating user root 172.17.0.1 port 35602: Too many au-
thentication failures [preauth]
2024-05-05T16:19:05+02:00 5a32ea872a79 sshd[71]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser=
rhost=172.17.0.1 user=root
2024-05-05T16:19:05+02:00 5a32ea872a79 sshd[71]: PAM service(sshd) ignoring max retries; 6 > 3
2024-05-05T16:19:05+02:00 5a32ea872a79 pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
2024-05-05T18:25:01:518176+02:00 kali CRON[110602]: (root) CMD (command -v debian-sai > /dev/null && debian-sai 1 1)
2024-05-05T18:25:01:523201+02:00 kali CRON[110602]: pam_unix(cron:session): session closed for user root
2024-05-05T18:26:36:203587+02:00 kali dbus-daemon[1171]: [session uid=1000 pid=1171] Activating service name='org.xfce.Xfc
onf' requested by '1.25' (uid=1000 pid=1384 comm='xfce4-panel --display :0.0 --sm-client-id 256d20d7')
2024-05-05T18:26:36:240752+02:00 kali dbus-daemon[1171]: [session uid=1000 pid=1171] Successfully activated service 'org.x
fce.Xfcconf'

```

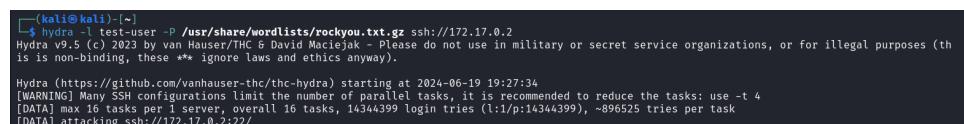
Figura 5.4: Monitorización de *logs* generados en máquinas víctima por operaciones

### Simulación de Ataques con otras herramientas

Una vez efectuados los ataques en la infraestructura de red interna a través de CALDERA, se llevó a cabo adicionalmente una serie de ataques adicionales a través de intentos de acceso externo mediante varias herramientas distintas indicadas anteriormente en 4.3.1: Metasploit e Hydra. El proceso para cada una de ellas fue el siguiente:

En el caso de Hydra, se llevó a cabo un ataque por diccionario al servicio de SSH como el mostrado en la Figura 5.5, utilizando uno de los disponibles en la carpeta */usr/share/wordlists/* de cualquier distribución Kali, en este caso el famoso diccionario *Rockyou* [67]. El comando utilizado fue:

```
hydra -l test-user -P /usr/share/wordlists/rockyou.txt.gz ssh://172.17.0.2
```



The terminal session shows the execution of the Hydra command to perform a dictionary attack on the SSH service. It includes a warning about the use of the tool for illegal purposes and details about the attack parameters: target is 172.17.0.2, user is "test-user", and password list is "/usr/share/wordlists/rockyou.txt.gz".

```

(kali㉿kali)-[~]
$ hydra -l test-user -P /usr/share/wordlists/rockyou.txt.gz ssh://172.17.0.2
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-06-19 19:27:34
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 16344399 login tries (l:/p:14344399), ~896525 tries per task
[DATA] attacking ssh://172.17.0.2:22/

```

Figura 5.5: Ataque por diccionario a servicio SSH mediante Hydra

A continuación, se utilizó un *scanner* de Metasploit para intentar enumerar usuarios de SSH de las máquinas víctima. Para ello, se utilizaron los siguientes comandos:

```
msfconsole # abrir una instancia de Metasploit
use auxiliary/scanner/ssh/ssh_enumusers # seleccionar script
set rhosts 172.17.0.2 # indicar la IP objetivo
set user_file /usr/share/wordlists/rockyou.gz # indicar el diccionario utilizado
run # ejecutar el ataque
```

Finalmente, como se muestra en la Figura 5.6, se usa la técnica *Malformed Packet*, que consiste en enviar paquetes formados incorrectamente a un servidor para probar forzar errores en el manejo de dichos paquetes y obtener información aparentemente oculta o inaccesible.

```
msf6 auxiliary(scanner/ssh/ssh_enumusers) > set rhosts 172.17.0.2
rhosts → 172.17.0.2
msf6 auxiliary(scanner/ssh/ssh_enumusers) > set user_file /usr/share/wordlists/rockyou.txt.gz
user_file ⇒ /usr/share/wordlists/rockyou.txt.gz
msf6 auxiliary(scanner/ssh/ssh_enumusers) > run
[*] 172.17.0.2:22 - SSH - Using malformed packet technique
[*] 172.17.0.2:22 - SSH - Checking for false positives
[*] 172.17.0.2:22 - SSH - Starting scan
```

Figura 5.6: Ataque con Metasploit para enumeración de usuarios en un servidor SSH

### 5.1.2. Dataset Sintético

En el caso del conjunto de datos generado de manera sintética, se ha realizado por medio de la funcionalidad de análisis de datos utilizada por el LLM de OpenAI multimodal (GPT4o). Para ello se ha hecho uso de técnicas de *prompt engineering*, de modo que su estructura fuese equivalente a la del anterior *dataset* ya preprocesado tal y como se especifica al final de la próxima sección 5.9.

Este modelo es capaz de generar como salida un fichero de texto. Además, el *notebook* que utiliza para generarlo puede ir readaptándose dinámicamente en base a las indicaciones que se van haciendo para ajustarlo al resultado deseado. Otra ventaja que presenta es la opción de adjuntar en el *prompt* un fichero, de modo que si se añade un ejemplo del resultado esperado del conjunto de datos, como el generado anteriormente, el modelo razonará con más claridad como debe implementar las celdas de código.

#### Prompt GPT4o - Generación Dataset Sintético

*"Genera un archivo en formato CSV que contenga 5000 registros, cada uno simulando un log de Linux orientado a la detección de vectores de ataque en sistemas de seguridad. Este dataset debe incluir una variedad ampliada de mensajes típicos encontrados en logs reales, adaptados para reflejar diversos tipos de actividades sospechosas y ataques, tales como intentos de acceso SSH, fallos de autenticación, alertas de kernel, y errores en aplicaciones y servicios de red. Cada log debe estar compuesto por las siguientes columnas, ajustándose al formato especificado en tu archivo de referencia: YYYY, MM, DD, hh, mm, Hostname, Service, User, IP, Port, Keyword, Interface, UID, Action, Protocol, Component, Severity, Type, Thread ID, Message."*

**Prompt GPT4o - Generación Dataset Sintético II**

Incluye en los registros una mezcla de eventos legítimos y maliciosos, asegurando que los mensajes y tipos de errores sean coherentes con el protocolo y el servicio implicado (por ejemplo, mensajes ICMP relacionados con problemas de red y mensajes SSH relacionados con la autenticación). Asegúrate de que los datos sean variados y realistas para facilitar el análisis y la detección de patrones anómalos en aplicaciones de aprendizaje automático.

**Prompt GPT4o - Generación Dataset Sintético III**

El dataset debe ser guardado como '06\_Synthetic-logs-5k.csv', y cada entrada debe ser única para evitar redundancias y mejorar la calidad del mismo de cara a pruebas y entrenamiento de modelos."

Por ejemplo:

```
2024 05 05 09:38 f6194a3d2d4d sshd[90]: test 172.17.0.1 Invalid
user test from 172.17.0.1 port 34132
```

Además de los anteriores, fue necesario indicar 4 *prompts* adicionales para ajustar más el formato del *dataset* obtenido. Finalmente, se obtuvo el conjunto de datos que se muestra en la Figura 5.7, el cual incluye los veinte campos indicados, con una gran variedad de eventos y con una completitud significativa.

Timestamp	User	Event Type	Source IP	Port	Protocol	Interface	IP	Action	Protocol Component	Severity	Type	Thread ID	Message
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.1 2679 error	eth1	1000 failed	UDP	application	high	warning	8833	fg connection from 192.168.2.1:2679 failed			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 error	eth1	1000 open	ICMP	system	high	error	2736	Possible SYN flooding from 192.168.2.3.151 detected, sending cookies			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 error	eth1	1000 succeeded	ICMP	system	high	warning	8834	fg connection from 192.168.2.3.151 port 56078 succeeded			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 error	eth1	0 close	TCP	network	medium	error	5741	error: maximum authentication attempts exceeded for test from 192.168.2.3.151 port 56078 [preauth]			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 error	eth1	0 failed	TCP	network	medium	warning	8831	IPW unreachable from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 open	TCP	network	medium	warning	8832	IPW unreachable from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	1000 failed	HTTP	application	high	error	2549	IPW network unreachable from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	1000 open	HTTP	application	high	warning	7254	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7255	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7256	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7257	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7258	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7259	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7260	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7261	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7262	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7263	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7264	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7265	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7266	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7267	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7268	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7269	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7270	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7271	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7272	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7273	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7274	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7275	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7276	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7277	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7278	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7279	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7280	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7281	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7282	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7283	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7284	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7285	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7286	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7287	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7288	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7289	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7290	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7291	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7292	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7293	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7294	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7295	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7296	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7297	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7298	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7299	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7300	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7301	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7302	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7303	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7304	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7305	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7306	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7307	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 failed	HTTP	application	high	warning	7308	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3.151 56078 warning	eth1	0 succeeded	HTTP	application	high	warning	7309	Received malformed packet from 192.168.2.3.151 to 192.168.2.201			
2024 02 28 00:41 server9 fg[34]:	root	192.168.2.3											

## 5.2. Preparación de los *datasets*

Una vez generado el *dataset* original en formato .log, es necesario llevar a cabo un proceso de preprocesamiento de modo que pueda ser interpretado correctamente por los modelos de análisis de datos. Este proceso se tiene que realizar de manera equivalente para los otros *datasets* que se han utilizado para hacer una comparativa y así enriquecer el análisis de posibles vectores de ataque. Se ha dividido en los siguientes pasos:

- Limpieza del conjunto de datos: se eliminarán los eventos que se consideren residuales y no resulten útiles para el proceso de entrenamiento. Sin embargo, no se eliminarán todos los *logs* que no estén relacionados con un ataque simulado, ya que la idea es poder generar agrupaciones de distintos *logs* y ver cuáles de estas pertenecen a ataques, para que todo sea del modo más fidedigno posible a la realidad.
- Preprocesamiento de los datos: el formato original de los *logs* no es práctico para su análisis, por lo que será necesario transformarlo a un fichero CSV, un formato más estandarizado para estas técnicas de *clustering* y hacer técnicas de *parsing* y normalización del texto.
- Extracción de características: en línea con el ítem anterior, será necesario extraer nuevos campos para facilitar el agrupamiento de *logs* en base a características comunes.

A continuación se detalla cómo se han hecho cada uno de los pasos anteriores.

### Limpieza del conjunto de datos

En primer lugar, se suprimieron una gran cantidad de *logs* redundantes y a su vez secuencialmente repetitivos para el problema a resolver, es decir, aquellos eventos generados en varias ocasiones que no aportaban al conjunto de datos. Este proceso se hizo manualmente examinando las aproximadamente nueve mil líneas de *logs* y parseando a través de la técnica de *search & replace*. En la Figura 5.8 se muestra un ejemplo de *logs* etiquetado como residual.

```

1889 2024-05-05T18:00:38.178135+02:00 kali kernel: 16:00:38.173348 dndHGM DnD: Received message HOST_DND_FN_GH_REQ_PENDING (0x258) from host
1890 2024-05-05T18:00:38.224087+02:00 kali kernel: 16:00:38.221510 dnd No guest source window
1891 2024-05-05T18:00:38.227224+02:00 kali kernel: 16:00:38.224418 dndHGM DnD: Received message HOST_DND_FN_GH_REQ_PENDING (0x258) from host
1892 2024-05-05T18:00:38.238905+02:00 kali kernel: 16:00:38.237825 dnd No guest source window
1893 2024-05-05T18:00:38.252907+02:00 kali kernel: 16:00:38.251288 dndHGM DnD: Received message HOST_DND_FN_GH_REQ_PENDING (0x258) from host
1894 2024-05-05T18:00:38.295201+02:00 kali kernel: 16:00:38.293651 dnd No guest source window
1895 2024-05-05T18:00:38.299803+02:00 kali kernel: 16:00:38.295948 dndHGM DnD: Received message HOST_DND_FN_GH_REQ_PENDING (0x258) from host
1896 2024-05-05T18:00:38.306434+02:00 kali kernel: 16:00:38.302382 dnd No guest source window
1897 2024-05-05T18:00:38.311457+02:00 kali kernel: 16:00:38.310826 dndHGM DnD: Received message HOST_DND_FN_GH_REQ_PENDING (0x258) from host
1898 2024-05-05T18:00:38.355959+02:00 kali kernel: 16:00:38.353306 dnd No guest source window
1899 2024-05-05T18:00:38.373860+02:00 kali kernel: 16:00:38.370959 dndHGM DnD: Received message HOST_DND_FN_GH_REQ_PENDING (0x258) from host
1900 2024-05-05T18:00:38.373947+02:00 kali kernel: 16:00:38.371507 dnd No guest source window
1901 2024-05-05T18:00:38.378342+02:00 kali kernel: 16:00:38.374865 dndHGM DnD: Received message HOST_DND_FN_GH_REQ_PENDING (0x258) from host
1902 2024-05-05T18:00:38.397929+02:00 kali kernel: 16:00:38.379264 dnd No guest source window
1903 2024-05-05T18:00:38.410911+02:00 kali kernel: 16:00:38.408917 dndHGM DnD: Received message HOST_DND_FN_GH_REQ_PENDING (0x258) from host

```

Figura 5.8: Ejemplo de *logs* residuales del conjunto de datos

```
No guest source window
DnD: Received message HOST_DND_FN_GH_REQ_PENDING (0x258) from host
```

Estos eventos están relacionados con un intento reiterado de realizar la operación de DnD (*drag & drop*), pero que no se ha podido realizar debido a que no se ha encontrado la ventana de origen en el sistema invitado o *guest*, un *log* muy común

en máquinas virtuales. También se han eliminado otros eventos relacionados con actividades de *booting* de VMs.

A través de este proceso, se consigue llevar a cabo un cierto balanceo de clases, lo cual es crucial para un buen rendimiento en el análisis de un *dataset*. En el contexto de los conjuntos utilizados para este proyecto, el volumen de datos no es significativamente grande, sin embargo sería interesante usar algún algoritmo de balanceo [69] en caso de enfrentarnos a conjuntos de datos mayores. Para ello se podrían utilizar algunos de los siguientes:

Tipo	Descripción
Sobremuestreo ( <i>Oversampling</i> )	Aumenta la cantidad de muestras en las clases menos representadas. El método más conocido es SMOTE [70], donde se generan ejemplos sintéticos (no duplicados) mediante la interpolación entre ejemplos existentes de la clase minoritaria.
Submuestreo ( <i>Undersampling</i> )	Reduce la cantidad de muestras en las clases sobrerepresentadas. Este método puede llevar a una pérdida de información importante, pero es útil cuando hay un volumen muy alto de datos.
Combinación de sobre-muestreo y submuestreo	Utiliza ambos métodos para equilibrar el conjunto de datos. Por ejemplo, se puede aplicar submuestreo a la clase mayoritaria y sobremuestreo a la clase minoritaria para lograr un balance.
Ensemble de modelos ( <i>Ensemble methods</i> )	Crea múltiples modelos (como árboles de decisión) en diferentes subconjuntos del conjunto de datos original, donde cada subconjunto refleja un balance de clases más equitativo. Técnicas como Balanced Random Forest o EasyEnsemble aplican esta idea.
Modificar pesos de clases ( <i>Class weights modification</i> )	Da más peso a la clase minoritaria durante el entrenamiento del modelo, lo cual puede influir en el algoritmo de aprendizaje para que preste más atención a la clase minoritaria.

Tabla 5.2: Técnicas de balanceo de clases para grandes conjuntos de datos

## Preprocesamiento de los datos

Una vez los *datasets* habían sido limpiados de *logs* residuales, se pasó a la fase de preprocesamiento. Esta fase tuvo un primer enfoque mediante el cual los ficheros *raw* se transformaron a un fichero JSON.

El proceso fue muy similar para los conjuntos de datos utilizados. En el caso de los *datasets* Linux\_2k 3.3.1 y el generado manualmente, se hizo uso de un mismo *script*. Dicho código se adaptó ligeramente para poder utilizarse en los *datasets* de BGL (3.3.1) y HDFS (3.3.1). La principal diferencia entre estos fue el tratamiento del campo *timestamp*, ya que los formatos eran ligeramente distintos, tal y como se explica en el capítulo cuarto (4.1).

Para el primer paso, el *script* realizaba un *parsing* contenido para crear un diccionario con cuatro etiquetas por cada *log*, de modo que se consiguiera la siguiente estructura:

```
{  
    "timestamp": "2024-05-05T09:38:32+02:00",  
    "hostname": "f6194a3d2d4d",  
    "service": "sshd[90]:",  
    "message": "pam_unix(sshd:auth): check pass; user unknown"  
}
```

Una vez todos los *datasets* tenían este formato unificado, ya resultaba más fácil realizar el tratamiento de los campos para la extracción de características y generar nuevas columnas.

## Extracción de características

Partiendo de los conjuntos de datos en formato JSON, se continuó con el preprocesado. Se llevaron a cabo simultáneamente varios pasos <sup>1</sup>para la extracción de características:

- Normalización de los campos: principalmente, se normalizó el contenido del *timestamp*, de modo que se mostrase con un formato equivalente. Este paso significó una leve pérdida de precisión en algunos de los datasets, que registraban también los segundos, microsegundos o bien la zona horaria asociada a la hora.
- Paso de JSON a CSV: mediante un *notebook* de *Jupyter* se llevó a cabo la transformación de los *datasets* al formato matricial separado por comas. De esta manera, cada fila representaba un evento y cada columna una característica.
- Extracción de nuevos campos: a partir de los campos de *timestamp* y *message* se generaron nuevas columnas que mejorarían el potencial de estos para ser analizados y estudiados a continuación.

En este proceso se siguió una metodología equivalente al paso anterior, es decir, para los conjuntos de datos de HDFS y BGL se utilizaron *notebooks* distintos al principal, que sí podía ser doblemente aprovechado por el *dataset* generado manualmente y Linux\_2k, extraído de LogHub [41]. Asimismo, los *dataset* resultantes tienen un total de veinte columnas, siendo las primeras las relativas a los datos correspondientes al *timestamp* separados en año, mes, día, hora, minutos y segundos, y la última columna el *message* intacto. Las columnas intermedias son, además del servicio y el *hostname*, aquellas que se han extraído de *message*.

---

<sup>1</sup>Además de estos pasos, se hace uso de IDF, pero esto será explicado más adelante en la próxima subsección (5.3.1)

YYYY	MM	DD	hh:mm	Hostname	Service
2024	05	05	11:09	kali	sshd[90]:
User	IP	Port	Keyword	Interface	
test	172.17.0.1	22	warning	eth0	
UID	Action	Protocol	Component	Severity	
1000	Started	TCP	RAS	HIGH	
Type	Thread ID	Message			
INFO	357	Invalid user test from 172.17.0.1 port 22			

Tabla 5.3: Estructura del *dataset* preprocesado

En algunos casos, las columnas tienen más posibilidades de completarse en función del *dataset* al que aplican, pero eso no significa que no puedan llegar a resultar de utilidad para la totalidad de los mismos. Por ese motivo y los mencionados anteriormente, he llegado a la conclusión de que todos los *datasets* debían hacer uso de la misma estructura.

El *parsing* se ha efectuado haciendo uso de los DFs de la librería pandas, de modo que se aplicaban expresiones regulares sobre la columna de *message* para identificar patrones que pudieran corresponderse con los nuevos campos extraídos. Por ejemplo, para llevar a cabo la extracción de una dirección IP se implementó una función y luego esta era aplicada sobre el *message* del *dataframe*:

```
def extract_ip(message):
    match = re.search(r'\b(?:\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})\b', message)
    return match.group(0) if match else None

df['IP'] = df['message'].apply(extract_ip)
```

Otro ejemplo más sencillo es el de extracción del campo *type*, donde se recorre el contenido de *message* en busca de alguna de las palabras identificadas como los niveles de prioridad (3.4) o tipo de evento:

```
df['Type'] = df['message'].apply(lambda x:
    re.findall(r'\b(INFO|FATAL|SEVERE|WARNING|ERROR)\b', x, re.IGNORECASE)[0] if
    re.findall(r'\b(INFO|FATAL|SEVERE|WARNING|ERROR)\b', x, re.IGNORECASE) else None)
```

Finalmente, se reorganiza el *dataframe* añadiendo las nuevas columnas y utilizando una separación por tabulados. Dicha separación se podría llevar a cabo también con comas, de hecho es lo más común, sin embargo es visualmente más sencillo interpretar el contenido con columnas separadas con tabulaciones. Por tanto, las características extraídas de los *datasets* de eventos son las siguientes:

- **YYYY:** Año de registro del log. Indica el año de calendario en formato de cuatro dígitos.
- **MM:** Mes del registro en formato numérico. Representa el mes del evento del 01 (enero) al 12 (diciembre).
- **DD:** Día del mes del registro. Número del día del mes en que se registró el evento, de 01 hasta 31, dependiendo del mes.
- **hh:** Hora del registro en formato 24 horas, de 00 a 23.
- **mm:** Minuto del registro, de 00 a 59, detallando el minuto exacto de la hora en que ocurrió el evento.
- **Hostname:** Nombre del host donde se generó el registro, que identifica la máquina o servidor específico.
- **Service:** Nombre e identificador del servicio implicado en el registro, proporcionando detalles sobre el proceso o aplicación involucrada.
- **User:** Usuario asociado al evento del log. Detalla el nombre de usuario que ejecutó o desencadenó el evento, si aplica.
- **IP:** Dirección IP asociada al registro. Especifica la dirección IP desde la cual se originó el evento, facilitando el rastreo de la actividad en la red.
- **Port:** Puerto de red relacionado con el evento. Indica el puerto de comunicación utilizado durante el incidente, si es relevante.
- **Keyword:** Palabras clave específicas identificadas en el mensaje. Ayuda a categorizar y buscar registros específicos basados en términos comunes.
- **Interface:** Interfaz de red implicada en el registro, proporcionando detalles sobre la interfaz física o virtual utilizada.
- **UID:** Identificador único del usuario asociado al registro, especialmente útil para correlacionar actividades en sistemas multiusuario.
- **Action:** Acción que describe el evento registrado, detallando la operación realizada, como iniciar, suspender, reanudar o finalizar una acción.
- **Protocol:** Protocolo de red relacionado con el registro, indicando el protocolo de comunicación utilizado (TCP, UDP, etc.), si aplica.
- **Component:** Componente del sistema implicado en el log, como un módulo específico del software o hardware involucrado.
- **Severity:** Nivel de severidad del evento, clasificado en términos de impacto o prioridad, como crítico, mayor, menor, o informativo.

- **Type:** Tipo de registro, basado en la naturaleza del evento, como error, advertencia, información, o depuración.
  - **Thread ID:** Identificador del hilo de procesamiento asociado al evento, útil para el seguimiento de problemas en aplicaciones multihilo.
  - **Message:** Mensaje detallado del registro, describiendo el evento, las acciones tomadas, y cualquier resultado relevante.

Figura 5.9: Estructura del *dataset* tras fase de preprocesamiento

Tras este proceso de limpieza, preprocesamiento y extracción de características, los conjuntos de datos resultantes que serán usados para realizar el *clustering* tienen las siguientes características:

Nombre	Nº de eventos
01_linux-logs-9k.csv	8890
02_linux-logs-2k.csv	1546
03_HDFS-logs-20k.csv	20000
04_BGL-logs-20k.csv	20000
05_Synthetic-logs-5k.csv	5000
06_auth-logs-20k.csv	20000

Tabla 5.4: Listado de conjuntos de datos preprocesados

De los conjuntos de datos presentados en la Tabla 5.4, tres de ellos (03\_HDFS-logs-20k.csv, 04\_BGL-logs-20k.csv y 06\_auth-logs-20k.csv) son fragmentos de otros *datasets* de mayor tamaño, que se han limitado a 20000 eventos. La decisión de no tomar los originales se ha sustentado en las limitaciones de cómputo disponibles para este Trabajo. Sin embargo, podría realizarse el mismo proceso utilizando sus versiones originales en caso de disponer de los recursos necesarios.

## 5.3. Implementación del modelo

Seguidamente, se llevó a cabo la implementación de distintos algoritmos de *clustering* para clasificar los conjuntos de datos generados, y en base a los resultados obtenidos extraer potenciales vectores de ataque. A continuación se muestran las técnicas de vectorización de texto, reducción de dimensionalidad y extracción de características utilizadas, así como los pasos seguidos para implementar los algoritmos de agrupación.

### 5.3.1. Técnicas utilizadas

Se ha hecho uso de distintas técnicas que mejorasen la creación de agrupaciones de eventos para que estas fueran procesadas de forma más precisa, y otras técnicas para minimizar la cantidad de dimensiones y simplificar los datos sin perder la representatividad.

#### Técnicas de Vectorización de Texto

La vectorización de texto se utiliza para transformar cada uno de los eventos de *logs* en representaciones numéricas, de modo que estos puedan ser procesados por algoritmos de ML. Para ello, se han utilizado las siguientes herramientas:

**TF-IDF:** (*Term Frequency-Inverse Document Frequency*): convierte los *logs* en una matriz de características TF-IDF. Como indica su acrónimo, puede desglosarse en dos partes con distinta funcionalidad:

- **TF** (*Term Frequency*): en primer lugar, este mide la frecuencia de cada una de las palabras existentes en una línea y posteriormente las vectoriza, convirtiéndolas en un conjunto de números que representan la frecuencia de cada palabra en dicho documento<sup>2</sup>.
- **IDF** (*Inverse Document Frequency*) : en segundo lugar, se reduce la escala de las palabras que aparecen con mayor frecuencia en el conjunto de documentos o *corpus*, y que por consiguiente son más discriminatorias con respecto a las demás palabras. Esto ayuda a resaltar aquellas palabras que son más únicas para cada *dataset* de eventos de Linux.

A efectos matemáticos, según Ho Chung et. al [71] se interpreta como el producto de las dos medidas, TF e IDF. Su valor puede calcularse de varias formas:

Por un lado, **TF**, representado como  $tf(t, d)$ , se suele usar la *frecuencia bruta* del término  $t$  en el *dataset*  $d$ , es decir, el nº de veces que una palabra o término  $t$  aparece en  $d$ . Denotando la frecuencia bruta de  $t$  por  $f(t,d)$ , la función  $tf$  será  $tf(t, d) = f(t,d)$ . También se pueden utilizar otras alternativas como las frecuencias booleanas, escalada logarítmicamente o la normalizada.

---

<sup>2</sup>Cuando se habla de documentos en este contexto, nos referimos a una línea o conjunto de líneas del *dataset*.

Esta última se calcula por el cociente de la frecuencia bruta entre la frecuencia máxima de alguna palabra ( $t$ ) en el documento:

$$\text{tf}(t, d) = \frac{f(t, d)}{\max\{f(t', d) : t' \in d\}} \quad (5.1)$$

Por otro lado, **IDF** mide si la palabra es común o no en el conjunto de *corpus*. Se obtiene por medio del cociente del nº total de *datasets* entre el número de *datasets* que contienen la palabra, y se toma el logaritmo de dicho cociente:

$$\text{idf}(t, D) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right) \quad (5.2)$$

donde:

- $|D|$ : cardinalidad de  $D$ , o número de *datasets* en el conjunto.
- $|\{d \in D : t \in d\}|$ : número de *datasets* donde aparece la palabra o término  $t$ . Si no está en la colección se producirá una división-por-cero. Por lo tanto, es común ajustar esta fórmula a  $1 + |\{d \in D : t \in d\}|$ .

Por último, en cuanto a la base de la función *log* no es relevante con respecto al resultado final obtenido. Por tanto, TF-IDF se calcula de la siguiente forma:

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D) \quad (5.3)$$

La combinación de estos permite por tanto la vectorización de los *logs* y al mismo tiempo la extracción de sus características. De cara a su uso en el *notebook*, se han importado de la librería Sci-kit Learn (4.3.3):

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

La segunda técnica utilizada ha sido **Count-Vectorizer** [72]. Esta herramienta se utiliza para transformar un *dataset* de texto en una matriz numérica de recuentos de palabras.

A diferencia de TF-IDF, Count-Vectorizer simplemente cuenta la frecuencia de aparición de cada término en el documento, sin considerar la frecuencia inversa de los documentos. Esta es útil cuando se desea obtener una representación simple y directa de la frecuencia de los términos.

Su proceso puede describirse de la siguiente manera:

1. Primero, se construye un vocabulario con todos los términos únicos presentes en el *corpus*.
2. Luego, se cuenta la frecuencia de aparición de cada término en cada línea, creando así una matriz en la que las filas representan líneas y las columnas representan los términos del vocabulario.
3. Cada entrada en la matriz corresponde al número de veces que un término aparece en una línea específica.

Este método es más sencillo y rápido de computar en comparación con TF-IDF, pero no proporciona la misma capacidad para diferenciar términos comunes y poco comunes en el conjunto de documentos. En el *notebook*, se ha importado y utilizado de la librería Sci-kit Learn (4.3.3) de la siguiente manera:

```
from sklearn.feature_extraction.text import CountVectorizer
```

De cara a futuras mejoras sería interesante utilizar otras alternativas como Word2Vec que aprende asociaciones de palabras a partir de grandes cantidades de datos en texto plano, y BERT, o su versión adaptada a detección de anomalías de seguridad como BERT-log [73].

## Técnicas de Reducción de Dimensionalidad

La reducción de dimensionalidad en los *corpus* de *logs* ha implicado transformar datos complejos en una representación más simple y manejable, sin perder mucha información importante. Para lograr esto, se han utilizado tres técnicas distintas.

En primer lugar, **PCA** (*Principal Component Analysis*) [74] transforma los datos a un nuevo sistema de coordenadas donde las mayores varianzas de los datos se proyectan en las primeras coordenadas. Esto permite reducir la dimensionalidad del *corpus* de *logs* manteniendo la mayor cantidad de varianza posible. Esta técnica es escalable a conjuntos de datos de un mayor tamaño, pero puede no capturar adecuadamente las complejas relaciones no lineales presentes en los *logs* de eventos de seguridad, lo que puede afectar la precisión en la detección de patrones de ataque.

Para utilizarla en el *notebook*, se realiza el siguiente *import*:

```
from sklearn.decomposition import PCA
```

En segundo lugar, **t-SNE** (*t-distributed Stochastic Neighbor Embedding*) [75], convierte similitudes entre datos en alta dimensión en probabilidades conjuntas y minimiza la divergencia de Kullback-Leibler entre estas probabilidades en el espacio de baja dimensión. Esta es particularmente efectiva para la agrupación de eventos de *logs*, ya que puede capturar relaciones no lineales y resaltar estructuras de *clusters* que son críticas para la identificación de vectores de ataque. Sin embargo, t-SNE puede ser computacionalmente costoso y no escalar bien con conjuntos de datos muy grandes, lo que podría limitar su aplicabilidad en escenarios con grandes volúmenes de datos de *logs*. Se importa de la siguiente forma:

```
from sklearn.manifold import TSNE
```

Por último, **UMAP** (*Uniform Manifold Approximation and Projection*) [76] está basada en la teoría de grafos y la topología algebraica, por lo que preserva tanto la estructura local como global de los datos. UMAP puede manejar grandes conjuntos de datos de manera eficiente y mantener la integridad de las relaciones tanto locales como globales. Esto facilita la detección de patrones complejos y anómalos en los *logs* con relativa rapidez pero, en consecuencia, solamente es útil para un número de vecinos pequeño. Para hacer uso de esta, primero es necesario instalar la dependencia de `umap-learn` y luego hacer el *import*:

```
!pip install umap-learn
import umap.umap_ as umap
```

A través de todas las técnicas que han sido mencionadas, se tratará de mejorar el tratamiento de los *dataset* y así obtener unos mejores resultados. Algunas de ellas serán combinadas y se analizará su rendimiento por medio de comparativas.

### 5.3.2. Desarrollo del *clustering*

Una vez aplicadas las técnicas mencionadas anteriormente tanto para la vectorización de texto como para la reducción de la dimensionalidad, se ha llevado a cabo la implementación de los algoritmos de *clustering* escogidos en el capítulo anterior. En el caso del *clustering* jerárquico, se han utilizado tanto el algoritmo HC, como la variante aglomerativa: AHC. A continuación se detalla el procedimiento seguido para cada uno de ellos.

Con el propósito de ir validando e ir haciendo un primer análisis sobre los resultados obtenidos, se ha ido imprimiendo un conjunto de gráficas, correspondientes a cada una de las combinaciones entre técnicas y algoritmos de *clustering*, de modo que se estimara visualmente qué combinación respondía mejor frente a este tipo de *corpus*.

En esta sección, se mostrará el proceso para el tratamiento del conjunto de datos generado manualmente, es decir: *01\_linux-logs-9k.csv*. En el caso de los demás *datasets* se ha seguido estrictamente el mismo procedimiento. Por lo que en el próximo capítulo se analizarán y compararán los resultados obtenidos.

#### Implementación de K-means

En primer lugar, para K-means se ha realizado la estimación del número óptimo de *clusters* asociado al parámetro  $k$  por medio del Algoritmo del Codo<sup>3</sup>, mencionado en el capítulo cuarto 4.10. En la Figura 5.10 se muestran resultados obtenidos para el *dataset* *01\_linux-logs-9k.csv*.

Como se observa, pueden diferenciarse ligeramente tres codos. El primero de ellos en torno a  $k=5$ , el segundo alrededor de  $k=10$  y el último próximo a  $k=16$ . Con el fin de verificar cual de estos valores es el más óptimo, se ha utilizado *Silhouette Score* en el rango [2,31].

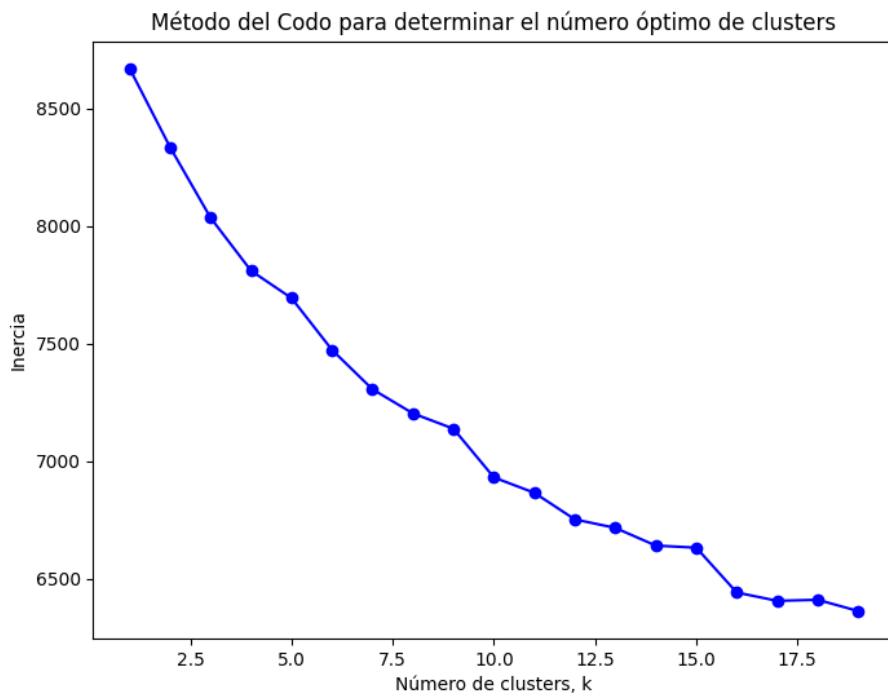


Figura 5.10: Aplicación del Algoritmo del Codo sobre *dataset* generado manualmente

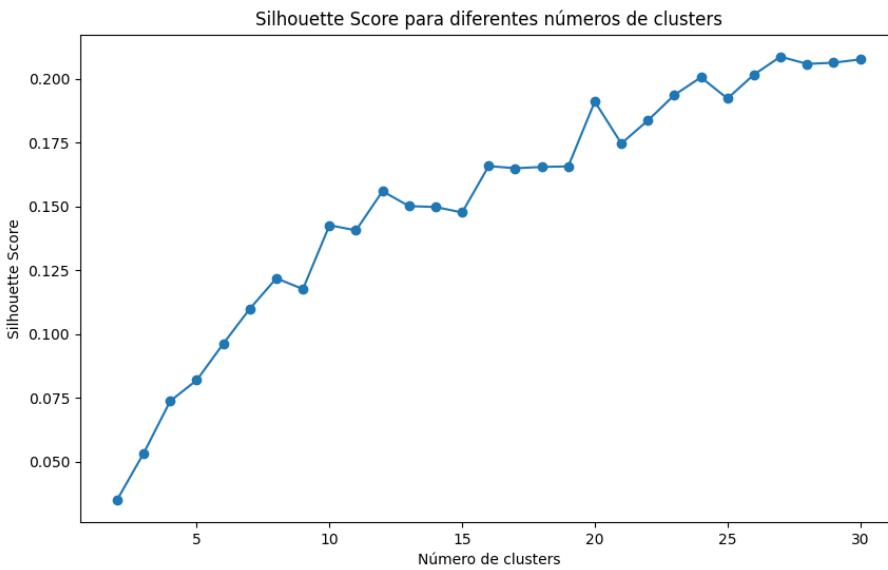


Figura 5.11: Cálculo del valor  $k$  óptimo mediante Silhouette Score

<sup>3</sup>En la Figura 5.10 el eje  $y$  está sociado a la Inercia, que es la suma de las distancias cuadradas de cada punto al centro de su *cluster* más cercano.

En base a los resultados visibles en la Figura 5.11, no se observa ningún pico especialmente significativo salvo uno cercano a  $k=10$ . Esto se debe a la heterogeneidad de los *logs*, incluso aquellos asociados a ataques de fuerza bruta. Si el rango utilizado para aplicar *Silhouette Score* aumentara, se ampliaría la cantidad de *clusters*, pero lo más conveniente es seleccionar un número realista, por tanto se fijará  $k=10$ .

Ahora se debe de llevar a cabo la inicialización del algoritmo utilizando 10 *clusters* y un valor fijo para la semilla (en este caso 42), que será utilizada posteriormente para poder comparar todos los resultados obtenidos. Seguidamente se aplicará el algoritmo al *dataset X*, de modo que se guarden todas las etiquetas de *cluster* asignadas a cada punto en *kmeans\_labels*.

```
kmeans = KMeans(n_clusters=10, random_state=42)
kmeans_labels = kmeans.fit_predict(X)
```

Por último se añadirá al DF una columna adicional que identificará la etiqueta de *cluster* que le corresponda y se extraerán las características más representativas de cada uno de estos diez *clusters*, de modo que pueda visualizarse posteriormente.

```
data['kmeans_cluster'] = kmeans_labels
kmeans_cluster_terms = np.array([extract_cluster_terms(kmeans_labels,
features, X.toarray())[i] for i in range(10)])
```

## Implementación de DBSCAN

Para el caso de DBSCAN, no se especifica directamente el número de *clusters*, sino que se calcula en base a los parámetros *eps* y *min\_samples*. Tras realizar pruebas prácticas para ver qué valores respondían mejor, se han establecido los siguientes:

$$\text{eps} = 0.5, \text{min\_samples} = 5$$

```
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(X)
```

Tras inicializar el algoritmo con los valores especificados, y a continuación aplicarlo al *dataset*, se procede a añadir al *dataframe* la nueva etiqueta y extraer las características.

```
data['dbscan_cluster'] = dbscan_labels
dbscan_cluster_terms = extract_cluster_terms(dbscan_labels, features,
X.toarray())
```

## Implementación de AHC y HC

Para los algoritmos de *clustering* jerárquico y aglomerativo, se ha optado por emplear el método de enlace *ward*, que minimiza la suma de diferencias cuadradas dentro de todos los *clusters*. Este método es menos susceptible a ruido y *outliers* en comparación con otros métodos de enlace como el completo o el simple. Este utiliza la siguiente fórmula para calcular la distancia entre dos *clusters* *A* y *B*:

$$D(A, B) = \sqrt{\frac{|A| \times |B|}{|A| + |B|} \times ||\text{centroide}_A - \text{centroide}_B||^2}$$

Después de calcular las distancias utilizando el método de enlace *ward*, el siguiente paso es aplicar la función *linkage* para construir la jerarquía de agrupaciones y luego usar *fcluster* para formar los *clusters*. De esta forma, será posible visualizar los dendogramas generados para poder hacer el análisis.

```
linked = linkage(X.toarray(), method='ward')
hierarchical_labels = fcluster(linked, 10, criterion='maxclust')
```

Una vez etiquetados los *clusters*, se deben asignar estas etiquetas a los datos originales, de modo que puedan identificarse características comunes dentro de cada *cluster*.

```
data['hierarchical_cluster'] = hierarchical_labels
hierarchical_cluster_terms = extract_cluster_terms(hierarchical_labels,
features, X.toarray())
```

En el caso del *clustering* aglomerativo es similar al jerárquico pero con un enfoque más directo en la combinación progresiva de *clusters* hasta alcanzar el número deseado de agrupaciones. Este método tenderá a dar mejores resultados con los *datasets* con mayor volumen de datos, como se demostrará en el próximo capítulo.

```
agglomerative_clustering = AgglomerativeClustering(n_clusters=9)
agglomerative_labels = agglomerative_clustering.fit_predict(X.toarray())
```

Finalmente, los resultados del *clustering* aglomerativo se asignan a los datos, y se extraen términos representativos para cada *cluster*.

```
data['agglomerative_cluster'] = agglomerative_labels  
agglomerative_cluster_terms = extract_cluster_terms(agglomerative_labels,  
features, X.toarray())
```

## 5.4. Selección de métricas de evaluación

El análisis del *clustering* en el contexto de la detección de patrones de ataque en *logs* requiere unas métricas específicas que reflejen la calidad de la agrupación más que la precisión de clasificación. Por este motivo, métricas comunes en otros contextos de aprendizaje supervisado como RECALL, PRECISION, o F1 Score no son aplicables aquí, ya que presuponen la existencia de etiquetas verdaderas previas, lo cual no se alinea con la naturaleza no supervisada del *clustering* [77].

### Silhouette Score

El coeficiente de silueta mide cuán bien un objeto está emparejado a su propio *cluster* comparado con otros clusters. Este coeficiente varía de -1 a 1, donde valores cercanos a 1 indican una asignación adecuada del objeto a su *cluster*. La fórmula del coeficiente de silueta es:

$$s = \frac{b - a}{\max(a, b)} \quad (5.4)$$

donde  $a$  es la distancia media dentro del *cluster* y  $b$  es la menor distancia media a cualquier otro cluster del que el objeto no es miembro.

Esta métrica viene implementada en Python a través de la librería **scikit-learn** (4.3.3). Para ello, se utiliza el siguiente bloque de código:

```
from sklearn.metrics import silhouette_score
```

### Índice de Davies-Bouldin

El índice de Davies-Bouldin [78] es una métrica que evalúa la separación entre *clusters*. Valores más bajos de este índice indican que los *clusters* están bien separados y son compactos. La fórmula para calcular el índice de Davies-Bouldin [79] es la siguiente:

$$DB = \frac{1}{n} \sum_{i=1}^n \max_{j \neq i} \left( \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) \quad (5.5)$$

donde  $\sigma$  es la dispersión dentro del *cluster* y  $d(c_i, c_j)$  es la distancia entre los centroides de los clusters  $i$  y  $j$ .

Al igual que la anterior, esta métrica viene implementada en Python por medio de **scikit-learn**. Puede importarse con:

```
from sklearn.metrics import davies_bouldin_score
```

## Índice de Calinski-Harabasz

Por último, el índice de Calinski-Harabasz [80] compara la dispersión entre los *clusters* con la dispersión existente dentro de los *clusters*. Valores más altos indican una mejor definición de los *clusters*. La fórmula para el índice de Calinski-Harabasz es:

$$CH = \frac{SSB/(k-1)}{SSW/(n-k)} \quad (5.6)$$

donde *SSB* es la suma de cuadrados entre los *clusters*, *SSW* es la suma de cuadrados dentro de los *clusters*, *k* es el número de *clusters*, y *n* es el número de puntos.

Para importar esta métrica se usa una vez más la implementación proporcionada por **scikit-learn**:

```
from sklearn.metrics import calinski_harabasz_score
```

Para calcular en el *notebook* las distintas métricas utilizadas, se ha hecho uso del siguiente fragmento de código:

```
metrics = {
    'Método de Clustering': ['K-Means', 'DBSCAN', 'Clustering Jerárquico',
    'Clustering Jerárquico Aglomerativo'],
    'Silhouette Score': [
        silhouette_score(X, kmeans_labels),
        silhouette_score(X, dbscan_labels) if len(set(dbscan_labels)) > 1
        else "No aplicable",
        silhouette_score(X, hierarchical_labels),
        silhouette_score(X, agglomerative_labels)
    ],
    'Índice Davies-Bouldin': [
        davies_bouldin_score(X.toarray(), kmeans_labels),
        davies_bouldin_score(X.toarray(), dbscan_labels)
        if len(set(dbscan_labels)) > 1 else "No aplicable",
        davies_bouldin_score(X.toarray(), hierarchical_labels),
        davies_bouldin_score(X.toarray(), agglomerative_labels)
    ],
    'Índice Calinski-Harabasz': [
        calinski_harabasz_score(X.toarray(), kmeans_labels),
        calinski_harabasz_score(X.toarray(), dbscan_labels)
        if len(set(dbscan_labels)) > 1 else "No aplicable",
        calinski_harabasz_score(X.toarray(), hierarchical_labels),
        calinski_harabasz_score(X.toarray(), agglomerative_labels)
    ]
}
```

Por tanto, los valores que pueden tomar los resultados utilizando las métricas mencionadas pueden corresponder con los de la Tabla 5.5.

Métrica	Mínimo	Máximo
Coeficiente de Silueta	-1	1
Índice de Davies-Bouldin	0	$\infty$
Índice de Calinski-Harabasz	0	$\infty$

Tabla 5.5: Intervalos de oscilación para las métricas de evaluación del *clustering*

### Extracción de términos principales

Además de llevar a cabo el análisis del *clustering* por medio de las tres métricas mencionadas anteriormente, se han implementado varias funciones que permitan extraer las palabras clave asociadas a cada *cluster* y graficarlas junto a la imagen donde se visualizan los puntos.

```
# Función para extraer palabras clave por cluster
def extract_cluster_terms(labels, features, X, n_words=10):
    unique_labels = np.unique(labels)
    cluster_terms = {}
    for label in unique_labels:
        if label != -1: # Excluir clusters marcados como ruido por DBSCAN
            indices = np.where(labels == label)[0]
            if len(indices) > 0:
                cluster_data = X[indices]
                centroid = np.mean(cluster_data, axis=0).flatten()
                top_indices = np.argsort(-centroid)[:n_words]
                cluster_terms[label] = [features[i] for i in top_indices]
    return cluster_terms
```

De esta forma, es posible entender cómo cada algoritmo está agrupando los distintos conjuntos de eventos, y por consiguiente identificar aquella/s agrupaciones asociadas a posibles riesgos de seguridad en el sistema operativo.

```
# Función para pintar y mostrar términos de cluster
def plot_and_print_clusters(X_pca, labels, title, cluster_terms, n_clusters=None):
    plt.figure(figsize=(10, 8))
    scatter = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, cmap='tab10',
                          marker='o', edgecolor='black')
    plt.title(title)
    plt.colorbar(scatter, ticks=np.unique(labels))
    plt.show()
    if n_clusters:
        for i in range(n_clusters):
            print(f"Cluster {i}: {', '.join(cluster_terms[i])}")
    else:
        for label, terms in cluster_terms.items():
            print(f"Cluster {label}: {', '.join(terms)})")
```

# Capítulo 6

## Análisis de resultados

En este capítulo se hará un estudio de los resultados obtenidos tras aplicar *clustering* a los distintos *datasets* de logs, y en base a las métricas utilizadas y las gráficas obtenidas medir cuales de ellos responden mejor de cara a la detección de potenciales vectores de ataque.

### 6.1. Evaluación de la efectividad de los modelos

Con el objetivo de evaluar cómo han respondido cada uno de los *datasets* anteriores, se va a proceder a analizar los valores calculados a través de las métricas de Silhouette Score, el índice de Davies-Bouldin y el índice de Calinski-Harabasz. Además, se hará un análisis de las gráficas en las que se ha mapeado cada *cluster* a los conjuntos de eventos y palabras contenidas en ellos.

Comenzando con el *dataset* generado manualmente, `01_linux-logs-9k.csv`, según las métricas este ha respondido mejor aparentemente con DBSCAN usando ambas técnicas de vectorización. Sin embargo, esto se debe a que el número de *clusters* ha sido muy elevado. Observando la gráfica de la Figura 6.1, puede observarse un resultado ampliamente mejor para el caso de K-means con TF-IDF y la técnica de reducción de dimensionalidad t-SNE. En dicho ejemplo el vector de ataque se encuentra en el tercer *cluster*, cuyas palabras asociadas son las siguientes:

```
Cluster 3: port, 172, 17, ssh2, password, failed, invalid, anonymous,  
user, root
```

En este ejemplo hay cierto margen de mejora, ya que se está agrupando en el *cluster* 8 otro conjunto asociado a un vector de ataque que debería haberse unificado con el anterior. Lo mismo sucede con el *cluster* 2, aunque este en menor medida.

```
Cluster 8: euid, ruser, logname, rhost, tty, ssh, authentication, uid,  
172, failure  
Cluster 2: pass, unknown, check, auth, sshd, pam_unix, user, device, devel
```

Igualmente, el resto de *clusters* no presentan ninguna palabra que esté potencialmente asociada a ningún tipo de ataque, por lo que puede concluirse un resultado bastante positivo. El vector de ataque detectado en este conjunto de datos está directamente relacionado con un intento de inicio de sesión fallido a través del servicio de SSH por parte de varios usuarios en reiteradas ocasiones: **root** y **anonymous**, lo cual podría estar directamente relacionado con un ataque del tipo *Malformed Packet* para descubrir usuarios comunes listados en un diccionario, o bien con un ataque por diccionario para esos dos usuarios en concreto.

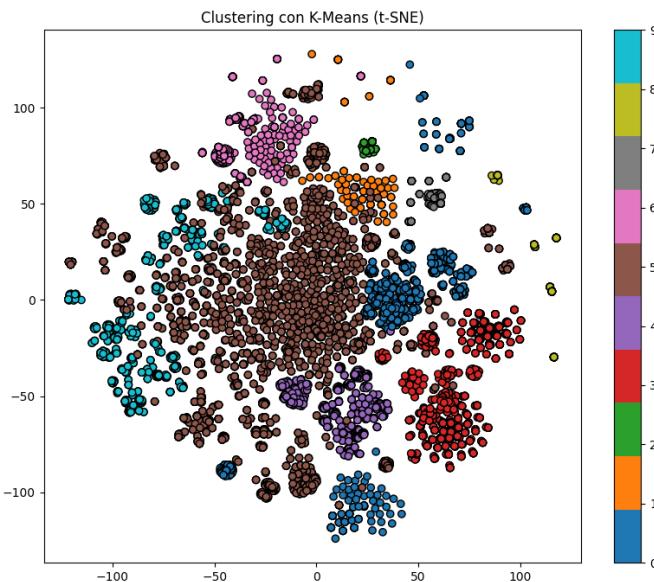


Figura 6.1: Clustering con K-Means utilizando TF-IDF y t-SNE

El resto de *clusters*, como se ha indicado anteriormente, no contienen ninguna palabra que pueda estar asociada algún ataque:

```

Cluster 0: main, 09, 08, dnd, x11, received, events, debian, host, 47
Cluster 1: retries, ignoring, max, pam, sshd, service, devices, device, devel
Cluster 4: containererd, 05, info, io, msg, 2024, time, level, 02, v1
Cluster 5: service, target, pipewire, agent, device, entered, socket, starting
Cluster 6: session, uid, pam_unix, cron, root, user, opened, closed, sudo, lightdm
Cluster 7: supervising, processes, threads, users, 10, 12, 13, 11, 14, 15
Cluster 9: service, successfully, org, bus, pid, uid, deactivated, 1000, dbus
  
```

Comparando el resultado obtenido con TF-IDF en K-means y a continuación, con *Count Vectorizer* en AHC (Figura 6.2), se observa una ligera mejora, ya que la amplia mayoría de *logs* asociados a ataques se agrupan en el *cluster* dos, y el resto en el quinto y el séptimo.

```

Cluster 2: port, 172, 17, ssh2, password, failed, invalid, anonymous, user, root
Cluster 5: euid, ruser, logname, rhost, tty, ssh, authentication, uid, 172,
failure
Cluster 7: pass, unknown, check, auth, sshd, pam_unix, user, device, devel,
failure
  
```

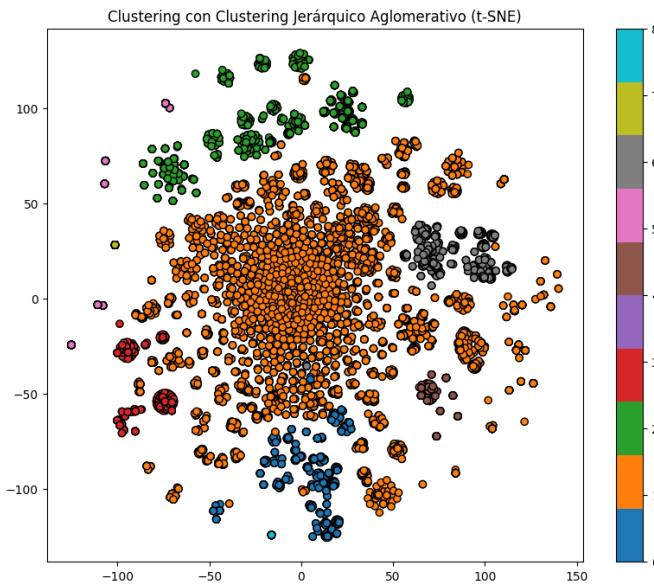


Figura 6.2: Clustering con AHC utilizando *Count Vectorizer* y t-SNE

El hecho de que DBSCAN haya tenido buenos resultados con *Silhouette Score* y al mismo tiempo unos muy bajos en las otras dos métricas como se muestra en la Figura 6.3, se debe a la gran cantidad de *clusters* que tiene, de modo que reajustando sus parámetros podría mejorar el resultado en estas últimas. No obstante, se tendrá en cuenta este resultado para ver qué comportamiento presenta este algoritmo de *clustering* en los demás conjuntos de datos.

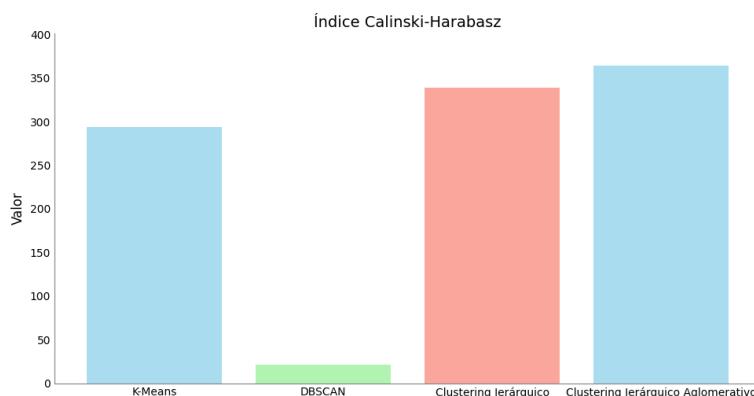


Figura 6.3: Resultados obtenidos en *dataset* manual según índice de Calinski-Harabasz

En el caso del segundo conjunto de datos: `02_linux-logs-2k.csv`, el cual contiene también eventos generados por *syslog* en una máquina Linux del año 2016, se incluyen principalmente intentos de inicio de sesión fallidos. Al aplicar los cuatro algoritmos de *clustering*, se han obtenido algunos resultados de interés.

Para el algoritmo *K-means* se aplicó el Algoritmo del Codo, y se obtuvo un valor de codo cercano a  $k=4$  tal y como se muestra en la Figura 6.4. En este caso el resultado es claro, por lo tanto ha sido necesario aproximar de nuevo el valor de  $k$  por medio de Silhouette Score.

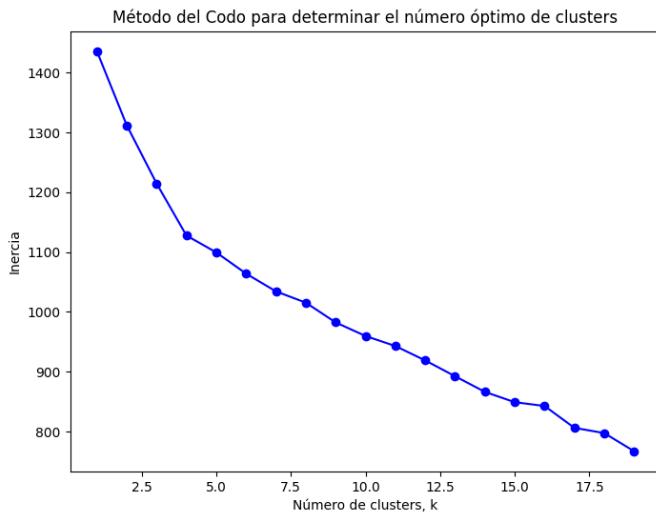


Figura 6.4: Cálculo del método del codo para el *dataset 02\_linux-logs-2k.csv*

Utilizando ambas técnicas de reducción de dimensionalidad se obtienen unos resultados muy similares. Los *logs* asociados a eventos críticos de seguridad se han podido agrupar en un mismo *cluster*: el número uno.

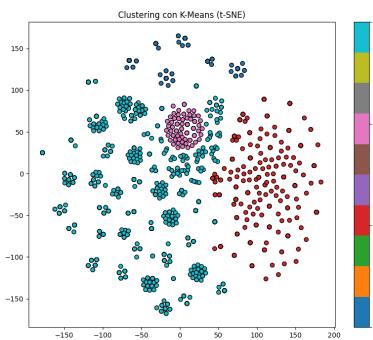


Figura 6.5: Clustering de *Linux\_2k* con K-means utilizando TF-IDF y t-SNE

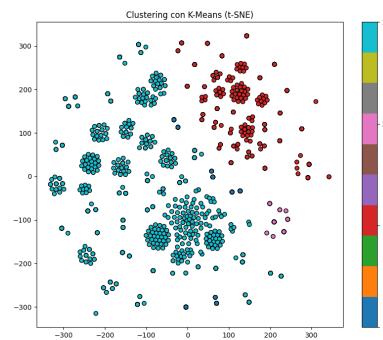


Figura 6.6: Clustering de *Linux\_2k* con K-means utilizando CountVectorizer y t-SNE

En la Figura 6.5 se agrupan bajo el *cluster* uno, al igual que en la Figura 6.6. Además su contenido es equivalente:

Cluster 1: ruser, rhost, euid, logname, tty, failure, nodevssh, authentication, uid, root

Para realizar una comparativa con *datasets* de eventos distintos a los de *syslog* en Linux, se ha seleccionado el conjunto de datos asociado a un fragmento de 20000 líneas de HDFS, generado en el año 2008. Los resultados obtenidos distan bastante con los anteriores, en primer lugar debido a que ha aumentado significativamente el volumen de datos *clusterizados*, y en segundo lugar por la información recopilada por estos *logs*, por tanto se presentan anomalías distintas, tal y como se muestra en la Figura 6.7 en el caso del algoritmo AHC. Al aplicar el método del codo, se ha obtenido un valor óptimo de *clusters* con  $k=13$ .

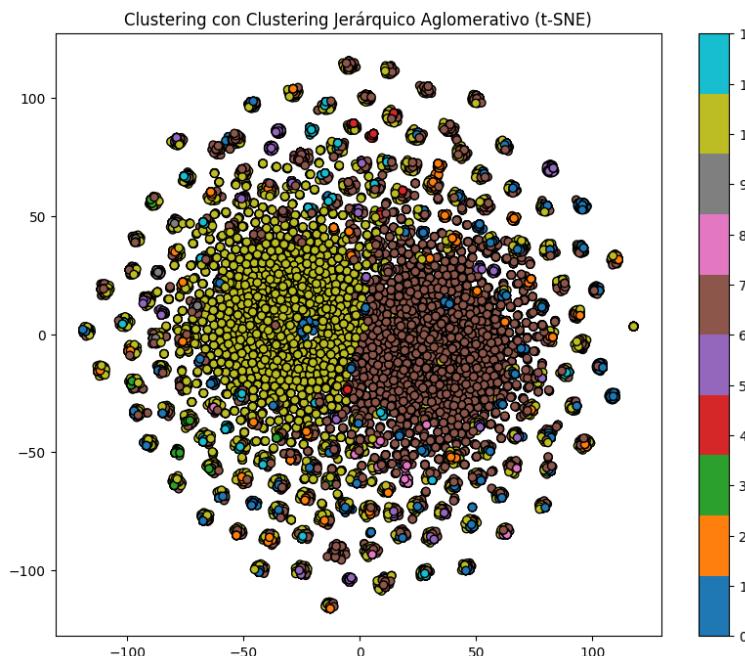


Figura 6.7: Clustering de HDFS con AHC utilizando TF-IDF y t-SNE

En la Figura 6.7, los principales *clusters* son el siete y el diez. Cada uno de ellos hace referencia a operaciones críticas sobre grandes bloques de datos, como finalizar una transferencia. Sin embargo, no se aprecia ninguna operación que llegue a resultar de utilidad para detectar algún tipo de acción indeseada, por lo que se demuestra una vez más la vital importancia de que los *logs* registren la información de la manera más clara posible en lenguaje natural, de modo que pueda ser clasificada de forma óptima por modelos de inteligencia artificial. Los *logs* de HDFS utilizan una terminología muy específica y ambigüa, que tendría que ser mapeada para poder facilitar al modelo su proceso de razonamiento y que pueda otorgar buenos resultados.

```
Cluster 7: terminating, block, updated, added, size, 67108864, 50010, 10, 251,250
Cluster 10: blk_, terminating, block, updated, added, 67108864, size, 50010, 10,25
```

En cuarto lugar, para BGL, que se trata de un conjunto de datos del supercomputador de IBM como se ha explicado en capítulos anteriores, se ha calculado

un valor óptimo de  $k=4$  *clusters*. Este valor, al igual que en los ejemplos anteriores, se ha calculado a partir de aplicar el Algoritmo del Codo. En este caso se han agrupado en el cluster cero (azul) *cluster* los eventos asociados a errores del *Kernel*, como se puede observar en la Figura 6.8.

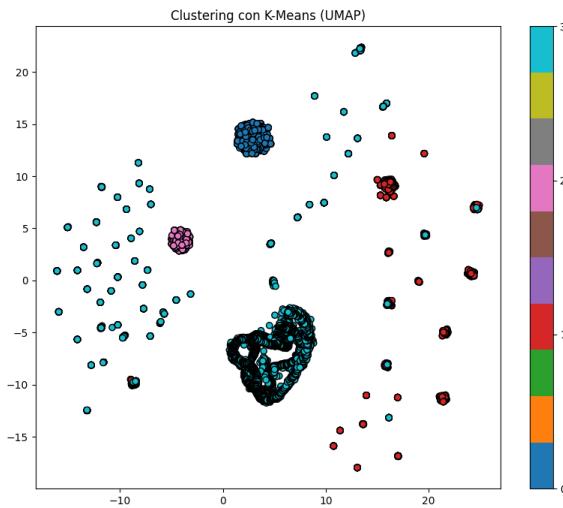


Figura 6.8: Clustering de BGL con K-means utilizando TF-IDF y UMAP

Se han obtenido unos resultados prácticamente equivalentes para el caso de los algoritmos de *clustering* jerárquico y su variante aglomerativa. De los cuatro *clusters*, se dividen varios tipos de alertas. Por un lado, en el grupo cero aquellas relacionadas con errores del *kernel* y la memoria caché. En el caso del *cluster* uno, se agrupan problemas relacionados con excepciones y alineamiento. Esto es positivo, ya que no simplemente se agrupan alertas en uno solo, sino que se fusionan en base a la tipología del error que ocasiona el evento.

```
Cluster 0: cache, instruction, parity, corrected, error, info, kernel, 35737, 35717, 35716
Cluster 1: hummer, exceptions, double, alignment, kernel, info, 182, 162, 141, 142
Cluster 2: message, ciostream, control, socket, prefix, stream, read, 96, 116, 172
Cluster 3: info, core, generating, kernel, mask, ce, sym, 0x08, detected, ddr
```

En quinto lugar, el *dataset* que ha sido generado de forma sintética, estaba formado de forma intencionada únicamente por alertas de seguridad asociadas a ataques de distinto ámbito. Si se obtienen unos buenos resultados, sería indicativo de que los algoritmos de *clustering* responden de una forma adecuada cuando los conjuntos de datos de *logs* son robustos, es decir, contienen eventos con una información completa y clara.

Los mejores resultados se han obtenido a través de la reducción de dimensionalidad con *Count Vectorizer* y la vectorización de texto con t-SNE. Como se observa en la Figura 6.9, se agrupan anomalías principalmente en los *clusters* tres, uno y ocho. Su contenido difiere principalmente por los usuarios involucrados en el proceso de autenticación a un servidor SSH.

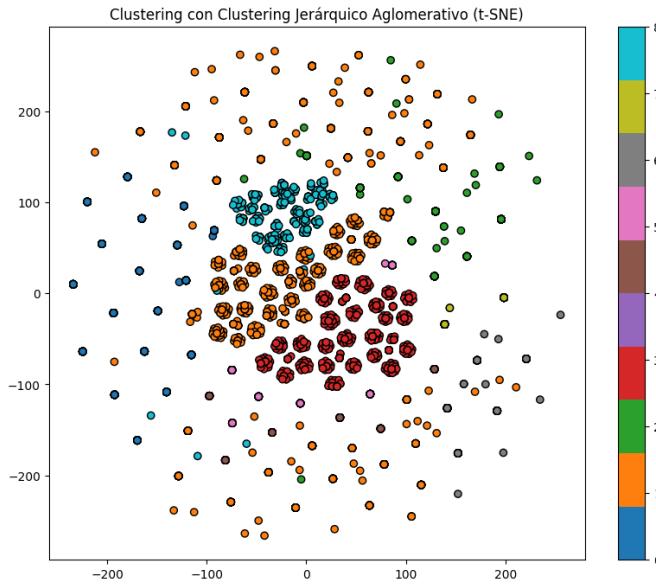


Figura 6.9: Clustering de AHC en *dataset* sintético con *Count Vectorizer* y t-SNE

En el siguiente fragmento de código se muestra el conjunto de palabras asociado a cada *cluster*. El primero de ellos hace referencia a varios elementos como la desconexión de un suario y un intento de sesión fallida. El segundo (*cluster 3*, se asocia con un *login* inválido a causa de un usuario inexistente y otros exitosos pero de riesgo debido a que están relacionados con usuarios con permisos de administrador. Por último, el *cluster* ocho contiene palabras relacionadas con un posible ataque de fuerza bruta que ha alcanzado un máximo de intentos de autenticación para un usuario *admin*.

```

Cluster 1: port, 168, 192, ssh2, failed, transfer, completed, user, disconnected
Cluster 3: logged, 168, 192, test, admin, guest, root, invalid_user, 51, 101
Cluster 8: exceeded, authentication, attempts, preauth, ssh2, port, 192, 168,
          test, admin
  
```

Una vez más, las métricas de Índice de Davies-Bouldin [78] e Índice de Calinski-Harabasz [80] otorgan una mayor puntuación a los algoritmos K-means, HC y AHC, mientras que *Silhouette Score* da la valoración más positiva a DBSCAN.

Finalmente, para el conjunto de datos *06\_auth-logs-20k.csv*, se ha calculado de nuevo mediante el Algoritmo del Codo el valor de *k* para utilizarlo en los algoritmos jerárquicos y K-means, obteniendo un valor de 12. El tamaño original de este era de 47000 eventos de seguridad, sin embargo debido a las limitaciones de computación ha sido necesario tomar un fragmento de 20000.

Los resultados obtenidos han sido, en comparación con los *datasets* generados tanto de forma manual como sintética, significativamente peores, tal y como puede observarse en la Figura 6.10.

Esto se debe principalmente a que los eventos capturados en este fichero provienen del fichero `auth.log`, que únicamente almacena logs a sociados a intentos de inicio de sesión fallidos. Es por ello, que sería interesante tratar de otro modo el *corpus* para que se realizara el *clustering* en base a los usuarios que han llevado a cabo el proceso de autenticación.

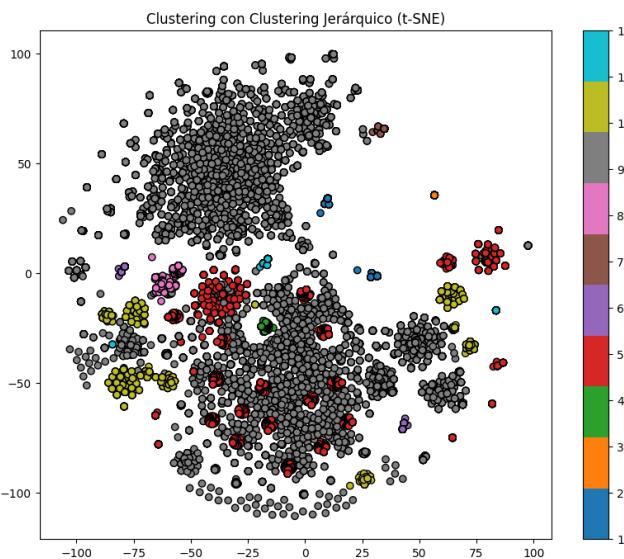


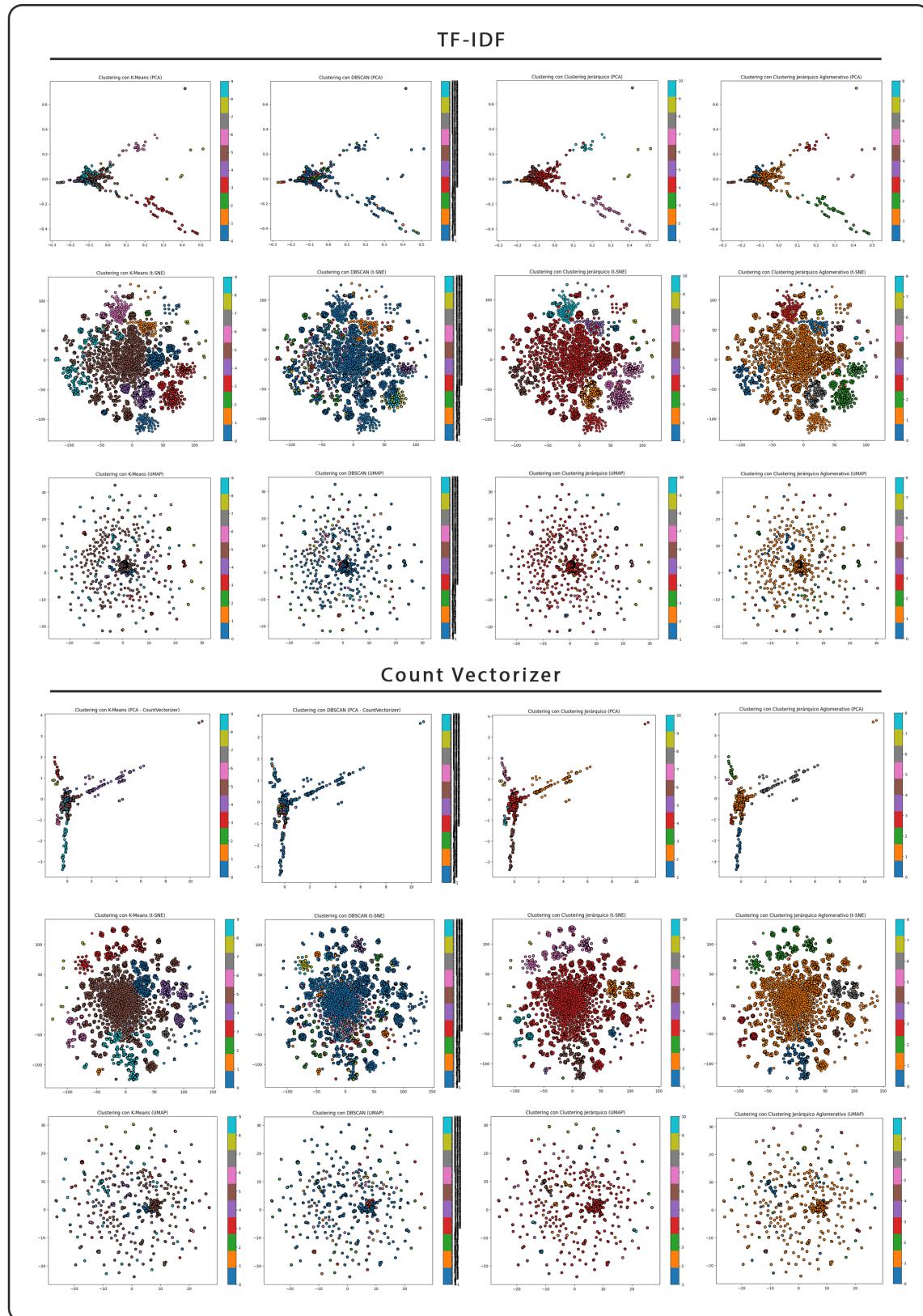
Figura 6.10: Clustering de HC en `06_auth-logs-20k.csv` con *Count Vectorizer* y t-SNE

Todas las gráficas anteriores que muestran los resultados de un análisis de *clustering* después de aplicar alguna técnica de reducción de dimensionalidad se representan por los siguientes ejes:

- **Eje X (horizontal):** Corresponde a la primera componente principal, que capta la dirección de máxima varianza de los datos, mostrando dónde los datos varían más.
- **Eje Y (vertical):** Corresponde a la segunda componente principal, que capta la siguiente dirección de máxima varianza, perpendicular a la primera.

En conclusión, los algoritmos de *clustering* que mejor han respondido ante estos conjuntos de datos han sido, por amplia diferencia K-means y los algoritmos jerárquicos HC y AHC. En cuanto a la técnica de vectorización de texto que mejores prestaciones ha tenido en base a las métricas utilizadas ha sido *Count Vectorizer*. Por último, la técnica de reducción de dimensionalidad que ha permitido graficar de forma más óptima los conjuntos de *logs* ha sido t-SNE.

En la Figura 6.11 pueden visualizarse los resultados obtenidos para el *dataset* generado manualmente (`01_linux-logs-9k.csv`) aplicando los distintos algoritmos de *clustering* en base a las técnicas utilizadas para vectorización de texto y reducción de dimensionalidad. Además, en la Tabla 6.1 pueden consultarse los valores obtenidos en base a las tres métricas utilizadas y clasificados por algoritmo utilizado y técnica de vectorización de texto.

Figura 6.11: Resultados del *clustering* obtenidos en el *Dataset* manual

Métrica Utilizada		TF-IDF				CountVectorizer			
Dataset	Número Eventos	K-means	DBSCAN	HC	AHC	K-means	DBSCAN	HC	AHC
<b>Silhouette Score</b>									
01_linux-logs-9k.csv	8890	0,142565	<b>0,327076</b>	0,148059	0,138362	0,045752	0,223888	0,056558	0,047342
02_linux-logs-2k.csv	1546	0,186831	<b>0,856451</b>	0,186831	0,186831	0,120715	0,834370	0,120715	0,120715
03_HDFS-logs-20k.csv	20000	0,055194	-0,01371	0,020737	0,018550	<b>0,129202</b>	-0,10574	0,026898	0,024624
04_BGL-logs-20k.csv	20000	0,367028	0,476697	0,363458	0,363458	0,596589	0,345334	<b>0,582639</b>	<b>0,582639</b>
05_Synthetic-logs-5k.csv	5000	0,183675	<b>0,327810</b>	0,168854	0,177956	0,166143	0,254842	0,172808	0,153171
06_auth-logs-20k.csv	20000	0,337545	0,661056	0,342969	0,342969	0,351366	<b>0,645813</b>	0,277735	0,277735
<b>Indice de Davies-Bouldin</b>									
01_linux-logs-9k.csv	8890	<b>3,160860</b>	1,101850	2,215990	2,427580	2,543825	1,217092	1,697760	1,813962
02_linux-logs-2k.csv	1546	<b>2,042738</b>	1,040805	<b>2,042738</b>	<b>2,042738</b>	1,638323	0,840527	1,638323	1,638323
03_HDFS-logs-20k.csv	20000	3,914300	1,052135	5,081278	<b>5,365123</b>	2,195493	1,426535	3,877497	4,802849
04_BGL-logs-20k.csv	20000	1,830761	0,952598	<b>1,883499</b>	<b>1,883499</b>	0,728352	0,854809	0,759906	0,759906
05_Synthetic-logs-5k.csv	5000	<b>2,471757</b>	1,028901	2,254009	2,359428	1,789288	1,127338	1,673542	1,722449
06_auth-logs-20k.csv	20000	<b>2,067445</b>	1,035235	1,589178	1,589178	1,695114	0,982265	1,317203	1,317203
<b>Indice de Calinski-Harabaz</b>									
01_linux-logs-9k.csv	8890	247,64400	24,20370	254,66300	267,61000	293,662881	21,363775	338,759653	<b>364,631418</b>
02_linux-logs-2k.csv	1546	140,494849	156,210995	140,494849	140,494849	205,211041	<b>235,150142</b>	205,211041	205,211041
03_HDFS-logs-20k.csv	20000	254,81996	170,938375	173,080575	180,749527	1303,516335	10,251355	607,210643	<b>664,011968</b>
04_BGL-logs-20k.csv	20000	4263,557621	159,093892	4176,9412	4176,9412	<b>18712,75285</b>	182,430667	17821,29833	17821,29833
05_Synthetic-logs-5k.csv	5000	249,890189	58,079644	255,864348	263,576459	530,687883	60,799303	554,721452	<b>565,499024</b>
06_auth-logs-20k.csv	20000	1443,017509	112,332262	1431,08245	1431,08245	<b>3297,852755</b>	148,411853	2295,506274	2295,506274

Tabla 6.1: Métricas obtenidas en los distintos conjuntos de datos

## 6.2. Análisis comparativo. Estudios previos para la detección de amenazas

En la misma línea que este Trabajo de Fin de Grado, a lo largo de estos últimos años han emergido numerosas propuestas para la detección proactiva de actividades maliciosas por medio de técnicas que apliquen de un modo u otro inteligencia artificial. A lo largo de esta sección se abordarán las principales alternativas publicadas en distintas fuentes institucionales como *Google Scholar*, *Scopus* o *Science Direct*.

### Primer caso de Estudio

Comenzando por el artículo publicado por Kayhan et. al [81], se propone el uso de UHAC (*Unsupervised hunting of anomalous commands*), que precisamente sigue el enfoque de detectar anomalías a través de conjuntos de datos no etiquetados en SIEMs mediante *clustering*. En concreto, hace uso de TF-IDF para la creación de la matriz de características y DBSCAN para el *clustering*. Además, emplea una función de pérdida personalizada que le permite manejar de forma adecuada los pesos binarios<sup>1</sup>. Los comandos se van ordenando en base a su error de reconstrucción, de modo que los errores más altos son indicativo de una mayor probabilidad de la existencia de anomalías.

En esta investigación, se hizo uso de OC-SVM (*One-Class Support Vector Machine*), que se trata de una técnica de clasificación empleada para la detección de anomalías, especialmente efectiva en contextos donde los datos etiquetados son escasos o inexistentes. Este construye un modelo a partir de un conjunto de datos no etiquetados, diferenciando entre patrones normales y aquellos que se consideran anomalías, mediante la maximización del margen de separación en un espacio de características transformado.

### Segundo caso de Estudio

En el *paper* publicado por Tendikov et. al [82], se lleva a cabo una evaluación de la efectividad de distintos modelos de ML con el propósito principal de detectar intrusiones en redes. Para ello, se levó a cabo una recopilación datos de máquinas virtuales Windows capturados por un SIEM, más concretamente ataques web procedentes del *dataset* CICIDS2017 [83]. En cuanto a las técnicas de *clustering* utilizadas, se incluyen *Random Forest* y K-means, así como técnicas de vectorización de texto como TF-IDF y de reducción de dimensionalidad como PCA y t-SNE. Por último, para medir los resultados obtenidos se hizo uso de la métrica F1 Score, mediante la cual se obtuvo un 0,97 utilizando *Random Forest*.

---

<sup>1</sup>Pesos que indican la presencia (1) o ausencia (0) de características específicas en los comandos.

### Tercer caso de Estudio

En tercer lugar, en el libro de Skopik et. Al [84], se lleva a cabo una clasificación de los tipos de anomalías que pueden llegar a ser detectadas a través de algoritmos de *clustering*. Entre estas destacan:

- **Outliers:** también conocidos como valores atípicos, son líneas de registro individuales que no coinciden con ninguna de las plantillas existentes o son disímiles a todos los *clusters* identificados que representan el comportamiento normal del sistema. Un ejemplo podría ser un mensaje de error en un archivo de registro que generalmente solo contiene mensajes informativos y de depuración.
- **Anomalías de Frecuencia:** son eventos de registro que aparecen con una frecuencia inesperadamente alta o baja durante un intervalo de tiempo dado. Esto incluye casos donde los componentes dejan de registrar eventos o la detección de ataques que implican la ejecución de muchos eventos, como los escaneos de vulnerabilidades.
- **Anomalías de Correlación:** eventos que se espera que ocurran en pares o grupos pero no lo hacen. Un claro ejemplo serían anomalías de coocurrencia y de implicación, es decir, aquellos eventos que deberían implicar la ocurrencia de otros.
- **Anomalías de Tiempo entre Llegadas:** Causadas por intervalos de tiempo desviados entre la ocurrencia de *logs*. Estas están relacionadas con las de correlación y pueden proporcionar capacidades de detección adicionales, por ejemplo, si un evento esperado no ocurre dentro de una ventana de tiempo específica.
- **Anomalías de Secuencia:** son provocadas por eventos faltantes o adicionales, así como por órdenes desviadas en secuencias de eventos de registro que se esperan ocurran en ciertos patrones.

Estos tipos de anomalías pueden ser detectadas o apoyadas por técnicas de *clustering* tanto estáticas como dinámicas, dependiendo del tipo específico de anomalía. Los *outliers* pueden ser identificados mediante algoritmos de *clustering* estático, mientras que las demás anomalías requieren técnicas de *clustering* dinámico y análisis estadístico para detectar violaciones en las correlaciones esperadas.

### Cuarto caso de Estudio

Otro caso digno de análisis es el del *paper* publicado por Tharunika et. al [4], titulado *Detection and Prevention of Advanced Persistent Threat (APT) Activities in Heterogeneous Networks using SIEM and Deep Learning*. En este se comenta el uso de modelos de ML y DL para la detección y prevención de anomalías de red en SIEMs. Para ello, se hace uso de la arquitectura de la Figura 6.12.

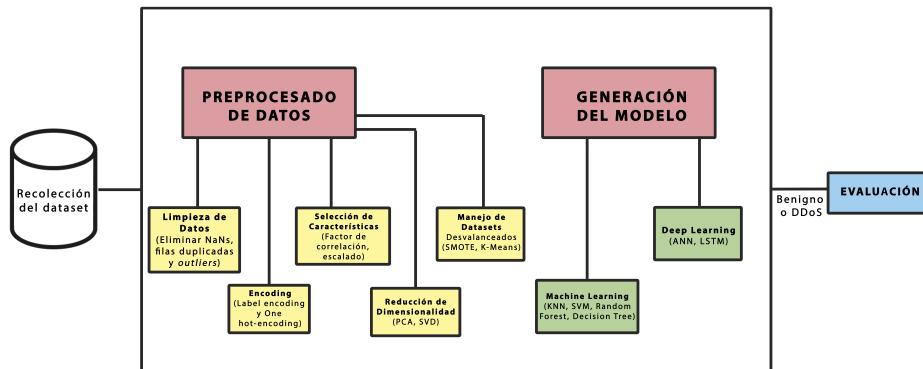


Figura 6.12: Diagrama de la arquitectura propuesta por Tharunika et. al [4]

Tal y como se puede observar, tras el proceso de recopilación del conjunto de datos, se lleva a cabo una fase de preprocesamiento en la que se eliminan NaNs, líneas duplicadas y *outliers*. A continuación se lleva a cabo el *encoding* y la selección de características. Seguidamente, se hace uso de técnicas de reducción de la dimensionalidad como PCA y SVD. Se emplean además algoritmos de *clustering* como SMOTE o K-means. Finalmente se lleva a cabo la generación de modelos, más concretamente se utilizan los siguientes:

- Modelos de ML: KNN, SVM, *Random Forest* y Árboles de decisión.
- Modelos de DL: ANN y LSTM.

Por medio del uso de estos modelos, se trataba de detectar si un *log* o conjunto de *logs* estaban asociados a un ataque de tipo DDoS o si eran benignos. Para evaluar los resultados obtenidos se hizo uso de cuatro métricas distintas. Estas métricas fueron: *Precision*, *Recall*, *F1-Score* y *Accuracy*. Según las tablas obtenidas se obtuvieron unos resultados muy positivos, lo cual demuestra de nuevo la importancia de llevar a cabo un buen preprocesamiento de los conjuntos de datos de *logs*.

## Quinto caso de Estudio

En línea con los estudios previos, el trabajo realizado por Datta et al. [9] en su artículo titulado *Real-time Threat Detection in UEBA using Unsupervised Learning Algorithms* se enfoca en la detección de amenazas en tiempo real mediante el uso de algoritmos de aprendizaje no supervisado en el análisis del comportamiento de usuarios y entidades (UEBA). En este estudio, se comparan cuatro algoritmos de *clustering* no supervisados para detectar amenazas internas de manera eficiente.

Para la reducción de dimensionalidad, se utilizó el PCA, permitiendo reducir las características del conjunto de datos a las más relevantes. En cuanto a las técnicas de preprocesamiento incluyeron *One-Hot Encoding* para transformar datos categóricos en formato binario, estandarización para ajustar los valores de los datos, y la eliminación de duplicados y valores atípicos y así mejorar la calidad del conjunto de datos.

Los modelos de *Machine Learning* utilizados fueron:

- K-Means (4.2.1)
- AHC (4.2.2)
- DBSCAN (4.2.3)
- GMM (*Gaussian Mixture Model*): este es un modelo probabilístico que asume la existencia de un número finito de distribuciones Gaussianas, cada una representando un cluster.

Para evaluar el rendimiento de los algoritmos utilizados, se emplearon las mismas métricas que se han empleado para este Trabajo de Fin de Grado, es decir, Silhouette Score, índice de Calinski-Harabasz e Índice de Davies-Bouldin. Los resultados obtenidos en esta investigación fueron los mostrados en la Tabla 6.2.

Modelo	Silhouette Score	Calinski-Harabasz	Davies-Bouldin
K-Means	0.2372	887.8355	1.2291
Agglomerative Clustering	0.2291	718.8155	1.4191
DBSCAN	0.3782	345.2103	2.2108
Gaussian Mixture Model (GMM)	0.0697	191.3066	2.4507

Tabla 6.2: Resultados obtenidos por Datta et al. [9] empleando algoritmos de *clustering* para la detección de amenazas

De este estudio se concluye que los algoritmos K-Means y *Agglomerative Clustering* fueron los más eficientes para la detección de amenazas internas. Una vez más, se prueba que funcionan los métodos de *clustering* no supervisados de cara a la detección de comportamientos anómalos en entornos de ciberseguridad.

Otro artículo estrechamente relacionado con este, es el de Artioli et al. [85], en el cual se lleva a cabo una investigación exhaustiva sobre algoritmos de *clustering* tanto tradicionales como emergentes aplicados a la UEBA. En este estudio, se abordan diferentes contextos de aplicación y se consideran múltiples escenarios de interacción usuario-entidad.

Artioli et al. comparan quince algoritmos de *clustering* en tres conjuntos de datos diferentes, en los que destaca principalmente el rendimiento prometedor de HDBSCAN y DenMune en el conjunto de datos de comportamiento CERT, generando grupos con una densidad muy cercana al número de usuarios.

## Sexto caso de Estudio

En contraste con los casos de estudio anteriores, se ha tenido en cuenta el desarrollo de un LLM orientado a ciberseguridad llamado *Odai* [86]. A lo largo del año 2023 y este 2024, ha ido optimizándose y mejorando su capacidad de resolución de actividades relacionadas con el *hacking* ético y la ciberseguridad. *Odai*, además del uso genérico de LLMs como *GPT* [87] o *Gemini* [88], propone el uso de agentes capaces de llevar a cabo ejercicios de *pentesting* en hasta 15 minutos.

A diferencia de otros *large model languages*, este no solo evita las limitaciones

convencionales de su uso para actividades relacionadas con el *hacking*, sino que está mejorado para implementar *scripts* que utilicen herramientas de seguridad comunes e incluso implementar *malware*.

Además, *Odai* ha ido adquirido diversas funcionalidades adicionales. Ahora permite la integración con IDSs y EDRs, herramientas directamente relacionadas con un SIEM. También ofrece acceso a una *API key* que permite usar el modelo de forma más personalizada, y ha logrado la integración con más de 50 herramientas de ciberseguridad.

La reciente actualización *Odai* v2.0 incluye un modelo con 50 mil millones de parámetros y un sistema de registro para el FS. Entre las herramientas utilizadas por sus usuarios se encuentran la implementación de *bypass* de *Cloudflare*, la herramienta Auto-XSS para detectar vulnerabilidades XSS, y mejoras en herramientas de OSINT para buscar contraseñas y datos en brechas de seguridad.

Más allá de la ética relacionada con el uso que se le puede dar a este tipo de soluciones tecnológicas, pueden surgir múltiples soluciones que hasta ahora no se habían llevado a cabo. En base a una reunión con su entonces director de tecnología Luis Javier Navarrete, el pasado 8 de junio, se comentó que una de estas soluciones sería la creación de *datasets* sintéticos a partir de agentes preparados con una serie de instrucciones muy específicas, de modo que se formaran conjuntos de datos verdaderamente útiles para entrenar modelos capaces de detectar anomalías de seguridad en distintos sistemas como Linux.

Sin embargo, podría ser de mayor utilidad cambiar el enfoque de la metodología utilizada para la detección de vectores de ataque en este tipo de sistemas, de modo que se prepararan conjuntos de datos completamente etiquetados y que contuvieran información adicional, como mapear para cada *log* si existe algún tipo de riesgo o no, o asignarle un potencial tipo de ataque (p.e. DDoS o fuerza bruta al *login* de algún servicio).

### 6.3. Sinergias e integración con sistemas SIEM

En el contexto de la amenaza constante por parte de los cibercriminales y ciberterroristas, la integración de la IA con sistemas SIEM es crucial para la detectar de forma efectiva incidentes y dar respuesta a estos en el menor tiempo posible. Según Ban et al.[89], los SIEMs modernos deben abordar con diversos problemas, como es el caso de la fatiga de alertas, que puede llegar a resolverse mediante el uso de técnicas avanzadas de ML y visualización de datos en tiempo real.

#### 6.3.1. Beneficios de la integración

La integración con sistemas SIEM ya está proporcionando múltiples beneficios que pueden ser ampliados, entre ellos destacan:

- Centralización de datos: la consolidación de *logs* y eventos de seguridad en una plataforma centralizada facilita su análisis y correlación a los analistas de seguridad.
- Automatización de respuestas: Los SIEMs pueden incorporar capacidades de SOAR para automatizar tareas de respuesta a incidentes, mejorando la eficiencia operativa.

- Reducción de falsos positivos: utilizando algoritmos de aprendizaje no supervisado o supervisado, como IWSVM (*Instance-Weighted Support Vector Machine*), propuesto por Ban et al.[89]<sup>2</sup>, es posible filtrar alertas irrelevantes y priorizar incidentes críticos.

### 6.3.2. Desafíos y soluciones

Uno de los mayores desafíos en la gestión de eventos de seguridad actualmente es la cantidad abrumadora de alertas generadas diariamente. Como se menciona en el trabajo de Ban et al.[89], se propuso un marco de SIEM de próxima generación que integra capacidades avanzadas de SOAR y utiliza una estrategia de DAC para mitigar el impacto de las alertas de baja calidad.

A pesar de la funcionalidad que ofrecen los SIEMs, estos presentan un cuello de botella con respecto a la toma de determinadas decisiones. Actualmente, en muchos casos es necesario que un profesional determine manualmente si hay algún tipo de intento de intrusión maliciosa, si hay un falso positivo o si simplemente no hay actividad inusual. Por lo tanto, a pesar de que la intervención humana seguirá siendo necesaria, la automatización de múltiples tareas permitirá que los analistas inviertan su tiempo únicamente en decisiones críticas e I+D.

La integración propuesta en este Trabajo de Fin de Grado implica la adopción por parte de un SIEM de técnicas de ML y visualización avanzada de los datos, con el principal propósito de mejorar la detección y respuesta a incidentes de seguridad. Entre estas técnicas, se propone especialmente utilizar algoritmos de *clustering* como K-means y AHC, ya que permiten una clasificación mucho más precisa que otras como DBSCAN según los resultados obtenidos.

De tal modo, se conseguirá una sinergia con sistemas SIEM que permitirá no solo la centralización y correlación de datos, sino también la automatización de tareas propias de un analista, así como la reducción de falsos positivos. En conclusión, los SIEMs han sido desarrollados para ayudar a los administradores, no a sustituir su trabajo, y a facilitar el diseño de políticas de seguridad y la gestión de eventos con la mayor celeridad posible, aumentando su efectividad frente métricas como el MTTD y MTTR.

---

<sup>2</sup>IWSVM es una mejora del SVM tradicional que asigna diferentes pesos a las instancias basándose en su importancia, mejorando el rendimiento en conjuntos de datos desbalanceados.

## Capítulo 7

# Conclusiones y trabajo futuro

### 7.1. Resumen de los logros obtenidos

Tras llevar a cabo este Trabajo de investigación acerca del análisis de *logs* de Linux por medio de algoritmos de *clustering* para detectar potenciales vectores de ataque, y con la finalidad principal de integrar esta funcionalidad en SIEMs para que estos mejoren los tiempos de respuesta, se han extraído las siguientes conclusiones:

- **Acerca de los *logs* de Linux:** Los *logs* de este sistema operativo son suficientemente ricos como para ser aprovechados en la detección de anomalías de distintos tipos. Se pueden identificar desde ataques remotos como fuerza bruta y denegación de servicio, hasta ataques a nivel del kernel o más bajo nivel.
- **Acerca del preprocessamiento de *logs*:** Esta es la parte más crucial del proceso. Lograr un dataset bien etiquetado mejora significativamente los resultados obtenidos. Sin embargo, también puede ser muy positivo el uso datasets no etiquetados como se ha llevado a cabo en este Trabajo.
- **Acerca de los modelos de ML utilizados:** Tras probar distintos algoritmos de *clustering*, los mejores resultados se obtuvieron con *k-means* y *clustering* jerárquico. Aun así, sería conveniente probar otros algoritmos de aprendizaje automático como KNN, SVM, *Random Forest*, *Decision Tree*, entre otros.
- **Acerca de la integración con SIEMs:** Desde hace algunos años se ha comenzado a integrar estas técnicas, pero el proceso es lento. Es vital seguir investigando para reducir los tiempos de detección de amenazas y generación de respuestas verdaderamente eficaces.

Con estos logros se ha demostrado que el análisis de *logs* de Linux mediante técnicas de *clustering* es una herramienta potente para la detección de vectores de ataque, y su integración en SIEMs es una vía prometedora para mejorar la ciberseguridad.

## 7.2. La importancia del Software Libre

El software libre juega un papel fundamental en el avance de la tecnología y la ciencia abierta. Permite el acceso libre y gratuito a información, herramientas y recursos que pueden ser utilizados y modificados por cualquier persona. En el contexto de la ciberseguridad, el software libre ofrece varias ventajas cruciales.

La transparencia es una de estas ventajas. Al ser accesible para todos, cualquier persona puede revisar el código fuente y asegurarse de que no contiene vulnerabilidades o puertas traseras. Esto aumenta la confianza en la seguridad y fiabilidad del software utilizado. Un ejemplo reciente es el de la biblioteca de compresión `xz` [90], donde se descubrió una puerta trasera en su código. La comunidad de código abierto reaccionó rápidamente, identificando y parcheando la vulnerabilidad, demostrando así la eficacia del escrutinio público en la detección y corrección de fallos de seguridad.

Otra ventaja es la colaboración. El software libre facilita la coordinación entre investigadores, desarrolladores y profesionales de la seguridad, quienes pueden contribuir con mejoras y nuevas funcionalidades. Esta colaboración abierta fomenta un entorno de innovación constante y rápida evolución tecnológica. Desde el ámbito personal, ha sido de gran utilidad para entender qué metodologías seguir para el proceso de análisis de *logs*. Un ejemplo de esta colaboración en el ámbito de la Universidad de Granada, es la labor realizada por la Oficina de Software Libre [91], que promueve la enseñanza a jóvenes en estas tecnologías de libre acceso.

El software libre también fomenta la innovación. Cualquiera puede tomar un proyecto existente y modificarlo para adaptarlo a nuevas necesidades o desarrollar nuevas soluciones. Esto hace que las herramientas evolucionen rápidamente y se adapten a los cambios y desafíos del entorno tecnológico. La comunidad de Linux es el principal ejemplo de que la innovación viene de la colaboración altruista, ya que es algo que beneficia a todos. En esta línea, muchas grandes empresas [92] están empezando a contribuir económicamente para que se invierta en herramientas *opensource*.

La accesibilidad es otro aspecto importante. El software libre hace que las herramientas de seguridad estén disponibles para una audiencia más amplia, incluyendo organizaciones pequeñas y países en desarrollo que no pueden permitirse costosas licencias de software propietario. Esto democratiza el acceso a tecnologías avanzadas y promueve la equidad en el ámbito tecnológico.

En resumen, el software libre y la ciencia abierta permiten que los resultados de la investigación sean compartidos y verificados por la comunidad científica global. Esto no solo mejora la calidad y la fiabilidad de la investigación, sino que también acelera el desarrollo de nuevas tecnologías y soluciones, beneficiando a toda la sociedad.

### 7.3. Líneas de investigación y desarrollo futuras (I+D)

Para seguir avanzando en el campo de la detección de vectores de ataque a través del análisis de *logs*, se identifican varias líneas de I+D que resultan prometedoras y necesarias.

Primero, es fundamental ampliar la variedad de vectores de ataque utilizados en las simulaciones realizadas. Estos pueden incluir desbordamiento de buffer (*buffer overflow*), condiciones de carrera (*race conditions*), inyecciones SQL, XSS o ejecución de *scripts* con firma desconocida, entre otros. Entender y detectar una gama más amplia de vectores de ataque permitirá crear sistemas de defensa más robustos y adaptativos.

En segundo lugar, se debe evaluar el uso de otros algoritmos de aprendizaje automático (ML) como KNN, *Random Forest*, RRNNs y SVM. Probar estos algoritmos en diferentes contextos y con distintos tipos de datos permitirá identificar cuáles son más efectivos para la detección de anomalías en los *logs*.

Además, diversificar las fuentes de *logs* es esencial. Utilizar *logs* de otros sistemas y aplicaciones, como los de servidores Apache, bases de datos y sistemas SCADA, ayudará a evaluar la efectividad de los modelos en diferentes contextos. Cada tipo de *log* puede presentar desafíos de preprocesado específicos, y detectar patrones únicos, lo que enriquecerá sin duda el análisis y la capacidad de detección.

Desarrollar técnicas para realizar correlaciones temporales entre eventos de *logs* es otra área clave de investigación. Identificar patrones de ataque que ocurren a lo largo del tiempo puede proporcionar una visión más completa y precisa de las amenazas, permitiendo una detección más temprana y una respuesta más efectiva.

La implementación y optimización de los modelos adaptados para su uso en análisis de *logs* en tiempo real es también una potencial vía de mejora, de modo que sea posible detectar y responder a incidentes de seguridad en tiempo real puede significar la diferencia entre prevenir un ataque y sufrir una brecha de seguridad grave.

Por último, continuar desarrollando métodos para la integración avanzada con SIEMs mejorará la capacidad para asociar y analizar datos de *logs* provenientes de múltiples fuentes, dando lugar a un análisis más profundo y una respuesta más coordinada a las amenazas de seguridad.

Estas líneas de investigación no solo buscan mejorar la precisión y eficiencia de los modelos actuales, sino también expandir el alcance y la aplicabilidad del análisis de *logs* en el ámbito de la ciberseguridad, asegurando que las nuevas amenazas puedan ser detectadas y mitigadas de manera efectiva.



# Bibliografía

- [1] EnterpriseAppsToday. *Linux Usage Statistics 2024 By Market Share, Usage Data, Number Of Users and Facts*, 2023.
- [2] MITRE Corporation. MITRE ATT&CK. <https://attack.mitre.org/>.
- [3] Julie Lidahl Gjerstad. Generating labelled network datasets of APTs with the MITRE CALDERA framework. Master's thesis, 2022.
- [4] V S Tharunika, Shridhar T, K Veeresh, Senthil Kumar Thangavel, Karthik Srinivasan, Sulakshan VajipayaJula, and Anjali Tibrewal. Detection and prevention of advanced persistent threat (apt) activities in heterogeneous networks using siem and deep learning. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–8, 2023.
- [5] TryHackMe. Linux Fundamentals Part 2. Comprehensive guide to Linux fundamentals, including practical exercises and challenges. URL: <https://tryhackme.com/r/room/linuxfundamentalspart2>.
- [6] QuickRef. Chmod: Quick reference. Quick reference cheat sheet that provides a brief overview of file permissions, and the operation of the chmod command. <https://quickref.me/chmod>.
- [7] Gartner. 2024 gartner magic quadrant for security information and event management, 2024.
- [8] ISO 8601. *Date and time format. Data elements and interchange formats. Information interchange. Representation of dates and times*. <https://www.iso.org/iso-8601-date-and-time-format.html>.
- [9] Joyatee Datta, Rohini Dasgupta, Sayantan Dasgupta, and Karmuru Rohit Reddy. Real-time threat detection in ueba using unsupervised learning algorithms. In *2021 5th International Conference on Electronics, Materials Engineering Nano-Technology (IEMENTech)*, pages 1–6, Sep. 2021.
- [10] Andrew Ng, Kian Katanforoosh, and Younes Bensouda Mourri. Neural networks and deep learning. DeepLearning.AI. Course offered through Coursera, oct 2023. This course is part of the Deep Learning Specialization. URL: <https://www.coursera.org/learn/neural-networks-deep-learning>.
- [11] Hugging Face. NLP Course. Hugging Face. Course available on the Hugging Face platform, jun 2024. Comprehensive course on Natural Language Processing using transformers. URL: <https://huggingface.co/learn/nlp-course/>.

- [12] Centro Criptológico Nacional. Inicio. <https://www.ccn.cni.es/index.php/es/>.
- [13] Centro Criptológico Nacional. Catálogo de Productos de Seguridad de las Tecnologías de la Información y Comunicación. <https://cpstic.ccn.cni.es/>.
- [14] Victor Poitevin. Stormshield. Principales cifras y estadísticas clave en cuanto a ciberamenazas, 2023. <https://www.stormshield.com/es/noticias/ciberseguridad-las-cifras-clave-de-2023/>.
- [15] MITRE Corporation. Common vulnerabilities and exposures (CVE). <https://cve.mitre.org/>, 2024. *The standardized list of vulnerabilities and exposures related to computer security.*
- [16] European Union Agency for Cybersecurity (ENISA). *ENISA Threat Landscape 2020 - Cybersecurity Research.* <https://www.enisa.europa.eu/topics/cyber-threats/threats-and-trends/etl-review-folder/etl-2020-cybersecurity-research>, 2020.
- [17] AV-TEST Institute. *2023 Cyber-Incidents in Numbers.* [https://www.av-test.org/fileadmin/pdf/security\\_report/AV-TEST\\_Cyber\\_Incidents\\_Report\\_2023\\_en.pdf](https://www.av-test.org/fileadmin/pdf/security_report/AV-TEST_Cyber_Incidents_Report_2023_en.pdf), February 2023. Period Covered by Report January 01st - December 31st, 2023.
- [18] Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. PentestGPT: An llm-empowered automatic penetration testing tool. *arXiv preprint arXiv:2308.06782*, 2023.
- [19] Raj Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*, volume 1. Wiley New York, 1991.
- [20] Ganttpro: Online Gantt Chart software for Project Planning. <https://ganttpro.es>.
- [21] José Ramón Fernández de la Cigoña. Contabilizando la amortización de los equipos informáticos: Pasos a seguir, March 2023.
- [22] Universidad Francisco de Vitoria. ¿Cuánto gana un ciberseguridad en España? Preguntas sobre grados, August 2023.
- [23] Universidad de Granada. Retribuciones pdi - universidad de granada 2023, 2023. Acceso a datos sobre retribuciones del personal docente e investigador.
- [24] Offensive Security. Kali Linux. Advanced Penetration Testing Linux distribution used for Penetration Testing, Ethical Hacking and network security assessments. <https://www.kali.org/docs/>.
- [25] Parrot Security. Parrot Security is a Free and Open source GNU/Linux distribution based on Debian stable designed for security experts, developers and privacy aware people. <https://parrotsec.org/docs/introduction/what-is-parrot/>.

- [26] BlackArch Linux. Blackarch Linux is an Arch Linux-based penetration testing distribution for penetration testers and security researchers. <https://blackarch.org/>.
- [27] Alexander Urrego Morera. *Seguridad en Sistemas Operativos Linux.* [https://repository.unipiloto.edu.co/bitstream/handle/20.500.12277/11573/Seguridad\\_Sistemas\\_operativos\\_Linux.pdf?sequence=1](https://repository.unipiloto.edu.co/bitstream/handle/20.500.12277/11573/Seguridad_Sistemas_operativos_Linux.pdf?sequence=1). Documento académico sobre las medidas de seguridad aplicables en sistemas operativos Linux.
- [28] Elastic. Elastic stack. <https://www.elastic.co/es/elastic-stack>, 2024. Comprehensive suite for search, logging, security, and analytics.
- [29] Splunk Inc. *Splunk.* <https://www.splunk.com/>.
- [30] Inc. Cisco Systems. Cisco official website. <https://www.cisco.com>, 2024. Official site for Cisco Systems, providing networking hardware, software, and telecommunications equipment.
- [31] Zabbix. <https://www.zabbix.com>. Plataforma de monitoreo de redes y aplicaciones.
- [32] Rainer Gerhards. *The Syslog Protocol.* <https://datatracker.ietf.org/doc/html/rfc5424>, March 2009.
- [33] Nitin Naik, Paul Jenkins, Paul Grace, and Jingping Song. Comparing attack models for it systems: Lockheed martin's cyber kill chain, mitre attack framework and diamond model. In *2022 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–7, 2022.
- [34] MITRE Corporation. Caldera. <https://github.com/mitre/caldera>, 2024. Automated adversary emulation system.
- [35] Endgame Inc. Red Team Automation (RTA), 2024. Repository of adversary emulation scenarios. <https://github.com/endgameinc/RTA>.
- [36] Red Canary. Atomic Red Team, 2024. Collection of small, highly portable detection tests mapped to the MITRE ATT&CK framework. <https://github.com/redcanaryco/atomic-red-team>.
- [37] Guardicore. Infection monkey, 2024. Open-source breach and attack simulation tool. <https://www.guardicore.com/infectionmonkey/>.
- [38] Scythe. Scythe. <https://www.scythe.io/platform>, 2024. Adversary emulation platform for conducting cybersecurity assessments.
- [39] Christine Barry. 5 Ways cybercriminals are using AI: Malware generation, April 2024. <https://blog.barracuda.com/2024/04/16/5-ways-cybercriminals-are-using-ai-malware-generation>.
- [40] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [41] Jieming Zhu, Shilin He, Pinjia He, Jinyang Liu, and Michael R. Lyu. Loghub: A large collection of system log datasets for ai-driven log analytics. <https://github.com/logpai/loghub/tree/master>, 2023.
- [42] Anton Chuvakin. Public security Log Sharing Site. 2010.

- [43] IBM. *What Is SIEM?* <https://www.ibm.com/es-es/topics/siem>, 2024.
- [44] Amrit Williams and Mark Nicolett. Improve it security with vulnerability management. *Gartner ID*, (G00127481), 2005.
- [45] Gustavo González-Granadillo, Susana González-Zarzosa, and Rodrigo Diaz. Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical infrastructures. *Sensors*, 21(14), 2021.
- [46] PlexTrac. MTTD and MTTR in cybersecurity. <https://plextrac.com/blog/mttd-and-mttr-in-cybersecurity/>, 2024.
- [47] Ángel Luis Vely Mora. Ventajas e implementación de un sistema SIEM. 2019.
- [48] Splunk. Splunk Enterprise Security. the industry-defining SIEM that delivers comprehensive visibility, empowers accurate detection with context, and fuels operational efficiency. [https://www.splunk.com/en\\_us/products/enterprise-security.html](https://www.splunk.com/en_us/products/enterprise-security.html), 2024.
- [49] Parlamento Europeo y del Consejo de la Unión Europea. *Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos)*. <https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=celex%3A32016R0679>, May 2016. Texto pertinente a efectos del EEE.
- [50] *Protection Profile for Network Devices, Version 2.2E*, March 2020.
- [51] Charu C. Aggarwal and Chandan K. Reddy. *Data Clustering: Algorithms and Applications*. Chapman Hall/CRC, 1st edition, 2013.
- [52] Patricia Lucía Barreno Recio, Anibal Bregón Bregón, and Miguel Angel. *Escola de Ingeniería Informática de Segovia, Universidad de Valladolid Martínez Prieto. Estudio de técnicas de clustering y detección de anomalías aplicado a fresadoras CNC*. pages 80–92, 2021.
- [53] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *WIREs Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.
- [54] Andrzej Dudek. Silhouette index as clustering evaluation tool. In Krzysztof Jajuga, Jacek Batóg, and Marek Walesiak, editors, *Classification and Data Analysis*, pages 19–33, Cham, 2020. Springer International Publishing.
- [55] M A Syakur, B K Khotimah, E M S Rochman, and B D Satoto. Integration k-means clustering method and elbow method for identification of the best customer profile cluster. *IOP Conference Series: Materials Science and Engineering*, 336(1):012017, apr 2018.
- [56] Kevin Babitz. DataCamp, Introduction to K-means Clustering with scikit-learn in Python. <https://www.datacamp.com/tutorial/k-means-clustering-python>, mar 2023. 21 min read.
- [57] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

- [58] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Trans. Database Syst.*, 42(3), jul 2017.
- [59] Michael Kouremetis et al. CALDERA v5.0.0 Magma, feb 2023. <https://github.com/mitre/caldera/releases/tag/5.0.0>.
- [60] David Kennedy, Jim O’Gorman, Devon Kearns, and Mati Aharoni. *Metasploit: The Penetration Tester’s Guide*. No Starch Press, 2011.
- [61] Jay Beale, Aaron Baker, and Brian Caswell. *Real World Penetration Testing Using Hydra*. Syngress, 2004.
- [62] Sergio de los Santos. *WSL Handbook. Guía Práctica Definitiva para Windows Subsystem for Linux*. Autoedición, 2024.
- [63] Prateek Sharma, Lucas Chaufournier, Prashant Shenoy, and Y. C. Tay. Containers and virtual machines at scale: A comparative study. In *Proceedings of the 17th International Middleware Conference*, Middleware ’16, New York, NY, USA, 2016. Association for Computing Machinery.
- [64] Alpine Linux Development Team. Alpine Linux, 2024. Alpine Linux is a security-oriented, lightweight Linux distribution based on musl libc and busybox. <https://alpinelinux.org/about/>.
- [65] Canonical Ltd. Ubuntu server. Ubuntu Server is a version of the Ubuntu operating system designed and engineered as a backbone for the internet, 2024. <https://ubuntu.com/server/docs>.
- [66] Abdulhamit Subasi. *Practical machine learning for data analysis using Python*. Academic Press, 2020.
- [67] Aikaterini Kanta, Iwen Coisel, and Mark Scanlon. A novel dictionary generation methodology for contextual-based password cracking. *IEEE Access*, 10:59178–59188, 2022.
- [68] Viacheslav Belenko, Vasiliy Krundyshev, and Maxim Kalinin. Synthetic datasets generation for intrusion detection in vanet. pages 1–6, 09 2018.
- [69] Joaquín García Abad et al. Comparativa de técnicas de balanceo de datos. aplicación a un caso real para la predicción de fuga de clientes. Master’s thesis, 2021.
- [70] Osvaldo Mario Spositto, Gabriel Esteban Blanco, Lorena Matteo, Marcelo Levi, and Julio Bossero. Smote, algoritmo para balanceo de clases en un estudio aplicado a la ganadería. In *XXVI Congreso Argentino de Ciencias de la Computación (CACIC)(Modalidad virtual, 5 al 9 de octubre de 2020)*, 2020.
- [71] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Trans. Inf. Syst.*, 26(3), jun 2008.
- [72] Scikit-learn developers. *sklearn.feature\_extraction.text.CountVectorizer*, 2024.
- [73] Haixuan Guo, Shuhan Yuan, and Xintao Wu. Logbert: Log anomaly detection via bert. In *2021 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2021.

- [74] I. T. Jolliffe. Principal component analysis. *Springer Series in Statistics*, 2016.
- [75] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.
- [76] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *ArXiv e-prints*, 2018.
- [77] Peter J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.
- [78] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.
- [79] Slobodan Petrovic. A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters. In *Proceedings of the 11th Nordic workshop of secure IT systems*, volume 2006, pages 53–64. Citeseer, 2006.
- [80] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974.
- [81] Varol O. Kayhan, Manish Agrawal, and Shivendu Shivendu. Cyber threat detection: Unsupervised hunting of anomalous commands (uhac). *Decision Support Systems*, 168:113928, 2023.
- [82] Noyan Tendikov, Leila Rzayeva, Bilal Saoud, Ibraheem Shaye, Marwan Hadri Azmi, Ali Myrzatay, and Mohammad Alnakhli. Security information event management data acquisition and analysis methods with machine learning principles. *Results in Engineering*, 22:102254, 2024.
- [83] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, January 2018.
- [84] Florian Skopik, Markus Wurzenberger, and Max Landauer. *Smart Log Data Analytics*. Springer, 2021.
- [85] Pierpaolo Artioli, Antonio Maci, and Alessio Magrì. A comprehensive investigation of clustering algorithms for user and entity behavior analytics. *Frontiers in big Data*, 7:1375818, 2024.
- [86] Luis Javier Navarrete. 0dai, 2024. AI-driven cybersecurity-focused Software as a Service (SaaS) platform offering services like malware development, social engineering, exploit development, and SOC analysis.
- [87] Gokul Yenduri, M. Ramalingam, G. Chemmalar Selvi, Y. Supriya, Gautam Srivastava, Praveen Kumar Reddy Maddikunta, G. Deepthi Raj, Rutvij H. Jhaveri, B. Prabadevi, Weizheng Wang, Athanasios V. Vasilakos, and Thippa Reddy Gadekallu. Gpt (generative pre-trained transformer)—a comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions. *IEEE Access*, 12:54608–54649, 2024.

- [88] Muhammad Imran and Norah Almusharraf. Google gemini as a next generation ai educational tool: a review of emerging educational technology. *Smart Learning Environments*, 11(1):22, 2024.
- [89] Tao Ban, Takeshi Takahashi, Samuel Ndichu, and Daisuke Inoue. Breaking alert fatigue: Ai-assisted siem framework for effective incident response. *Applied Sciences*, 13(11):6610, 2023.
- [90] Dan Goodin. Xz backdoor: Everything you need to know. *Ars Technica*, apr 2024.
- [91] Universidad de Granada. Oficina de Software Libre de la Universidad de Granada, 2023.
- [92] Recluit. Tendencias que transformarán el Open Source, 2024.
- [93] CALDERA. SSH Tunneling. <https://caldera.readthedocs.io/en/latest/C2-Tunneling.htmlssh-tunneling>.
- [94] Jerome H Saltzer. The origin of the “mit license”. *IEEE Annals of the History of Computing*, 42(4):94–98, 2020.
- [95] Overleaf. Overleaf - An online collaborative LaTeX editor. <https://www.overleaf.com>, 2024.
- [96] Certificación LINCE. Centro Criptológico Nacional (CCN), 2024.



## Apéndice A

# Formación Inicial

Para la elaboración de este trabajo ha sido necesario un período de formación en el que aprender acerca de distintas áreas específicas de la informática. El principal reto ha sido la aplicación de técnicas de inteligencia artificial, ya que la mención elegida durante la carrera fue Tecnologías de la Información, por tanto nunca había tratado con nada relacionado con modelos de IA, *clustering*, etc. Sin embargo, simplemente ha sido un reto más a superar. Para ello, se han llevado a cabo las siguientes actividades:

### Curso Redes neurales y Aprendizaje profundo

Este curso, impartido por Andrew Ng. et. al [10] y con una duración estimada de 4 semanas, proporcionaba las bases del funcionamiento de las RRNN y el DL. Cada semana debían visualizarse una serie de vídeos, realizar uno o varios cuestionarios e implementar un notebook para poner en práctica lo aprendido. Estaba formado por los siguientes módulos:

Módulo	Descripción	Videos	Lecturas	Tareas
1	Análisis de las principales tendencias que impulsan el auge del aprendizaje profundo y ejemplos de dónde y cómo se aplica en la actualidad.	6	3	1
2	Planteamiento de un problema de aprendizaje automático con una mentalidad de red neuronal y utilización de la vectorización para acelerar sus modelos.	19	5	3
3	Construcción de una red neuronal con una capa oculta, utilizando la propagación hacia delante y la retropropagación.	12	1	2
4	Análisis de los cálculos clave subyacentes al aprendizaje profundo y su utilización para construir y entrenar RRNN profundas para tareas de VC.	8	7	3

Tabla A.1: Módulos del Curso de Deep Learning & Neural Networks, Andrew Ng. [10]

Finalmente, a pesar de no tener una relación tan estrecha con la implementación llevada a cabo en este proyecto, resultó de especial utilidad para entender conceptos básicos y poder hacer un uso consciente de la terminología, así como dar a conocer las principales librerías de Python utilizadas para el análisis de datos.

Fecha de finalización: 24 Octubre 2023

### Curso NLP Hugging Face

Este segundo curso, aprendí los fundamentos del NLP, qué son los encoders y los decoders, la librería de transformers y otra información de utilidad para el proyecto.

Módulo	Sección	Contenido
0	Setup	Introduction
1	Transformer models	Natural Language Processing; Transformers, what can they do?; How do Transformers work?; Encoder models; Decoder models Sequence-to-sequence models; Bias and limitations
2	Using transformers	Introduction; Behind the pipeline; Models; Tokenizers; Handling multiple sequences; Putting it all together; Basic usage completed!
3	Fine-tuning a pretrained model	Introduction; Processing the data; Fine-tuning a model with the Trainer API or Keras; A full training; Fine-tuning, Check!
4	Sharing models and tokenizers	Uploading model; Versions and models; Sharing across the ecosystem; DL and model card; Common concerns.
5	The datasets library	Introduction; What if my dataset isn't on the Hub?; Time to slice and dice; Big data? Datasets to the rescue!; Creating your own dataset; Semantic search with FAISS; Datasets, check!
6	The tokenizers library	Introduction; Training a new tokenizer from an old one; Fast tokenizers' special powers; Fast tokenizers in the QA pipeline; Normalization and pre-tokenization; Byte-Pair Encoding tokenization; WordPiece tokenization; Unigram tokenization; Building a tokenizer, block by block; Tokenizers, check!
7	Main NLP tasks	Introduction; Token classification; Fine-tuning a masked language model; Translation; Summarization; Training a causal language model from scratch; Question answering; Mastering NLP

Tabla A.2: Módulos del Curso de NLP de Hugging Face [11]

Fecha de finalización: 13 Enero 2024

### Resolución de módulos y máquinas de Capture The Flag

Para aumentar mi formación en el ámbito de ciberseguridad, a lo largo de estos meses he llevado a cabo la resolución de las siguientes máquinas de CTF y módulos en distintas plataformas:

Plataforma	Máquinas Resueltas
TryHackMe	109
HackTheBox	36
Atenea	1
PortSwigger	2

Tabla A.3: Máquinas de *Hacking* Ético resueltas por Plataforma

Entre estas máquinas, algunas me han resultado de especial interés, ya que me han proporcionado una base de conocimiento en relación a los Sistemas Operativos Linux y conceptos básicos de implementación de su seguridad:

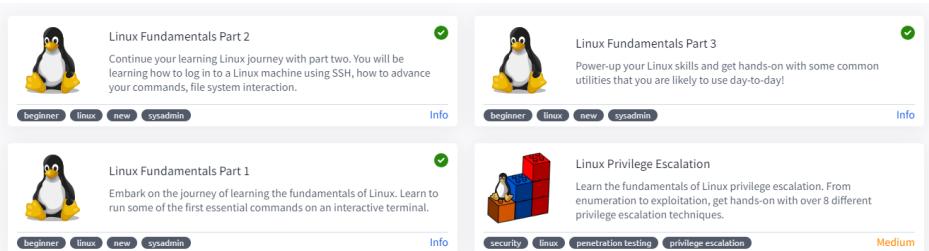


Figura A.1: Módulos de TryHackMe sobre Fundamentos de Seguridad en Linux [5]

En ellos, se explican elementos como la gestión de permisos, el sistema de archivos de Linux, automatización o técnicas de *hardening*. Otros módulos más específicos, como los que se muestran a continuación, ha resultado útiles para adquirir un mayor contexto acerca de los logs y su uso en sistemas de monitorización como los SIEM:

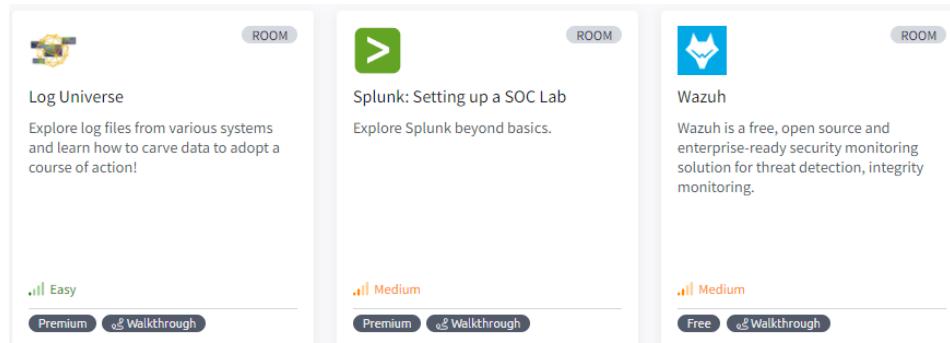


Figura A.2: Módulos de TryHackMe sobre Logs y Monitoreo de sistemas Linux I

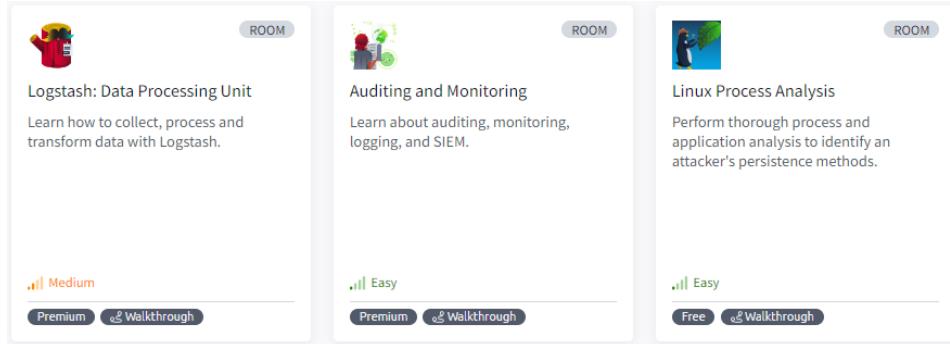


Figura A.3: Módulos de TryHackMe sobre Logs y Monitoreo de sistemas Linux II

## Apéndice B

# Repositorio del proyecto

En este capítulo se describe cómo se ha alojado este Trabajo de Fin de Grado en GitHub. Aunque se trata de un trabajo de investigación y no se ha seguido una metodología de CI/CD, se ha procurado aplicar buenas prácticas como trabajar en ramas separadas para cada parte del proyecto, utilizar *Issues*, *Milestones* y *PRs* para integrar los cambios en la rama principal de forma controlada.

### Licencia del proyecto: MIT

Se ha utilizado la licencia MIT [94] para este proyecto. Este tipo de licencia permite a los usuarios ejecutar, copiar, modificar, fusionar, publicar, distribuir, sublicenciar y/o vender copias del Software, siempre que se incluya el aviso de copyright y la renuncia de garantía.

### Metodología utilizada

Como se ha indicado anteriormente, este Trabajo se ha fundamentado en investigar acerca de cómo trabajar con *logs* de Linux, el funcionamiento de los SIEM y cómo aplicar algoritmos de *clustering* para tratar de detectar vectores de ataque sobre este tipo de sistemas. Por consiguiente, no se ha llevado a cabo un proceso de desarrollo convencional, de modo que se ha trabajado utilizando herramientas distintas como Google Colab, y una vez finalizada la investigación, se ha alojado en la forja. Las actividades desarrolladas se han dividido en distintos *Milestone*, tal y como se muestra en la Figura B.1.

Milestone	Descripción
Configuración del entorno de simulación de ataques	Configuración y documentación del entorno de pruebas para la simulación de ataques.
Datasets	Creación y organización de los conjuntos de datos necesarios para el proyecto.
Scripts de preprocessado	Desarrollo y prueba de <i>scripts</i> para la limpieza y preparación de datos.
Implementación del clustering	Desarrollo e integración de algoritmos de <i>clustering</i> para el análisis de datos.

Tabla B.1: Listado de *Milestones* utilizados para agrupar objetivos a cumplir

## Estructura del repositorio

Este repositorio cuenta con un fichero `README.md` principal, que explica a alto nivel en qué se basa esta investigación, además de la licencia MIT y un `.gitignore` que incluye archivos de ámbito privado como es el caso de los *datasets* de terceros, aunque en caso de que el tribunal requiera ver el contenido de los mismos, este se pondrá a su entera disposición sin problema alguno.

A continuación se muestra en la Tabla B.2 la estructura completa del repositorio del Trabajo de Fin de Grado, que ha sido titulado como *Cyberlynx*, ya que esta investigación ha sido en el ámbito de ciberseguridad y, en España, el término Lince está relacionado con un tipo de certificaciones de Seguridad [96].

Archivo/Directorio	Descripción
<code>.gitignore</code>	Excluidos ficheros como los <i>datasets</i> de terceros.
<code>LICENSE</code>	Documento de licencia.
<code>README.md</code>	Documento base del proyecto.
SRC/	<ul style="list-style-type: none"> <li>■ <code>DATASETS/</code> - Datasets de entrada.           <ul style="list-style-type: none"> <li>• <code>RAW/</code> - Logs sin procesar.</li> <li>• <code>JSON/</code> - Logs en formato JSON.</li> <li>• <code>CSV/</code> - Logs en formato CSV.</li> </ul> </li> <li>■ <code>CONFIG/</code> - Configuración del entorno.           <ul style="list-style-type: none"> <li>• <code>KALI/</code> - Configuración máquina Kali Linux.</li> <li>• <code>CONTAINERS/</code> - Configuración contenedores Docker.</li> </ul> </li> <li>■ <code>SCRIPTS/</code> - Scripts de procesamiento de datos.</li> <li>■ <code>CLUSTERING/</code> - Notebooks para <i>clustering</i> y análisis de resultados.</li> </ul>
DOC/	<ul style="list-style-type: none"> <li>■ <code>APENDICES/</code> - Apéndices.           <ul style="list-style-type: none"> <li>• <code>A.Formacion.tex</code></li> <li>• <code>B.Repositorio.tex</code></li> <li>• <code>C.Caldera.tex</code></li> </ul> </li> <li>■ <code>BIBLIOGRAFIA/</code> <ul style="list-style-type: none"> <li>• <code>bibliografia.bib</code> - Referencias bibliográficas.</li> </ul> </li> <li>■ <code>CAPITULOS/</code> - Capítulos del TFG.           <ul style="list-style-type: none"> <li>• <code>01_Introduccion.tex</code></li> <li>• <code>02_Planificacion.tex</code></li> <li>• <code>03_EstadoDelArte.tex</code></li> <li>• <code>04_Metodologia.tex</code></li> <li>• <code>05_Desarrollo.tex</code></li> <li>• <code>06_Analisis.tex</code></li> <li>• <code>07_Conclusiones.tex</code></li> </ul> </li> <li>■ <code>GLOSARIO/</code> - Glosario de términos.           <ul style="list-style-type: none"> <li>• <code>entradas_glosario.tex</code></li> </ul> </li> <li>■ <code>IMAGENES/</code> - Imágenes utilizadas.</li> <li>■ <code>PORTADA/</code> - Documentos de portada.           <ul style="list-style-type: none"> <li>• <code>portada_2.tex</code></li> <li>• <code>portada.tex</code></li> </ul> </li> <li>■ <code>PREFACIOS/</code> - Prefacios.           <ul style="list-style-type: none"> <li>• <code>prefacio.tex</code></li> </ul> </li> <li>■ <code>Makefile</code> - Script para la construcción del proyecto.</li> <li>■ <code>projeto.pdf</code> - Documento PDF final.</li> <li>■ <code>projeto.tex</code> - Archivo principal de LaTeX.</li> <li>■ <code>PRESENTACION.pdf</code> - Presentación del proyecto.</li> </ul>

Tabla B.2: Estructura jerárquica del repositorio Cyberlynx

De la estructura jerárquica visible en la Tabla B.2, se establece una división en dos carpetas principales:

- SRC, que contiene los *scripts* utilizados para el preprocesamiento de los *datasets*, ficheros de configuración empleados para la simulación de ataques y documentación aclaratoria sobre los comandos a utilizar para replicar dicha simulación, así como los *notebooks* en los que se ha llevado a cabo la implementación del *clustering*.
- DOC, que almacena la documentación de la memoria. Esta se ha desarrollado a través de la versión gratuita de la plataforma *opensource* Overleaf [95], por lo que una vez finalizada la memoria ha sido alojada en el repositorio.

### Enlace al repositorio

Para acceder al repositorio de este Trabajo de Fin de Grado debe utilizarse el siguiente enlace:

<https://github.com/mario-ca/cyberlynx>



## Apéndice C

# Instalación y configuración de CALDERA

Como se ha comentado anteriormente, CALDERA es un *framework* desarrollado por MITRE basado en el marco de técnicas, tácticas y procedimientos de MITRE ATT&CK. Este permite la simulación de adversarios complejos utilizando agentes distribuidos que pueden ejecutar una serie de técnicas descritas en el marco ATT&CK para evaluar la robustez de la seguridad de una red o un *host*. La principal ventaja de este *framework* frente a otros reside en su capacidad de integrar múltiples *plugins* y módulos que extienden sus funcionalidades, proporcionando la posibilidad de personalizar las operaciones de simulación de amenazas según necesidades específicas.

### Instalación de CALDERA

Para descargar CALDERA, es necesario acceder al repositorio oficial de GitHub [34]. Simplemente se clonará el repositorio activando la opción de copia recursiva en la máquina Kali por medio del siguiente comando:

```
git clone https://github.com/mitre/caldera.git --recursive
```

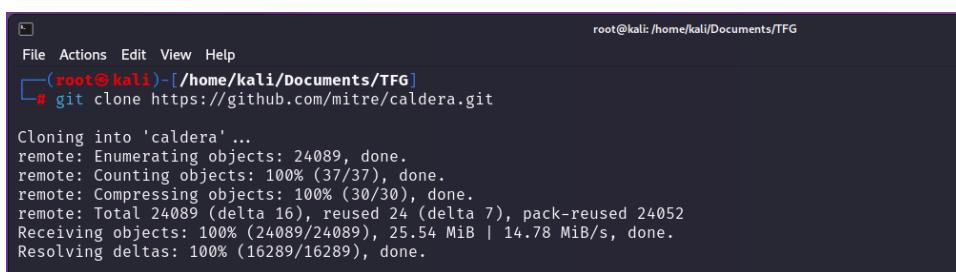
A terminal window titled 'root@kali: /home/kali/Documents/TFG' showing the command 'git clone https://github.com/mitre/caldera.git --recursive'. The output shows the cloning process, including object enumeration, counting, compressing, and receiving objects, and finally resolving deltas.

Figura C.1: Clonación del repositorio de CALDERA en la máquina atacante

A continuación, accedemos al directorio e instalamos las dependencias de Python por medio del siguiente comando:

```
cd caldera
pip install -r requirements.txt
```

```
root@kali:~/home/kali/Documents/TFG/caldera
# cd caldera
(root@kali) [~/home/kali/Documents/TFG/caldera]
# pip install -r requirements.txt

Collecting aiohttp-jinja2==1.5.1 (from -r requirements.txt (line 1))
  Downloading aiohttp_jinja2-1.5.1-py3-none-any.whl.metadata (8.1 kB)
Collecting aiohttp==3.9.3 (from -r requirements.txt (line 2))
  Downloading aiohttp-3.9.3-cp311-cp311-manylinux_2_17_x86_64_manylinux2014_x86_64.whl.metadata (7.4 kB)
Collecting aiohttp_session==2.12.0 (from -r requirements.txt (line 3))
  Downloading aiohttp_session-2.12.0-py3-none-any.whl.metadata (8.4 kB)
Collecting aiohttp_security==0.4.0 (from -r requirements.txt (line 4))
  Downloading aiohttp_security-0.4.0-py3-none-any.whl.metadata (2.9 kB)
Collecting aiohttpapispec==3.0.0#0 (from -r requirements.txt (line 5))
  Downloading aiohttpapispec-3.0.0#0.tar.gz (2.7 MB)
  Downloading aiohttpapispec-3.0.0#0.tar.gz (2.7 MB)

    2.7/2.7 MB 10.5 MB/s eta 0:00:00

Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing metadata (pyproject.toml) ... done
Collecting jinja2==3.1.3 (from -r requirements.txt (line 6))
  Downloading Jinja2-3.1.3-py3-none-any.whl.metadata (3.3 kB)
Requirement already satisfied: pyyaml==6.0.1 in /usr/lib/python3/dist-packages (from -r requirements.txt (line 7)) (6.0.1)
Collecting cryptography==42.0.2 (from -r requirements.txt (line 8))
  Downloading cryptography-42.0.2-cp39abi3-manylinux_2_28_x86_64.whl.metadata (5.3 kB)
Collecting websockets==11.0.3 (from -r requirements.txt (line 9))
  Downloading websockets-11.0.3-cp311-cp311-manylinux_2_5_x86_64_manylinux1_x86_64_manylinux2014_x86_64.whl.metadata (6.6 kB)
Collecting Sphinx==7.1.2 (from -r requirements.txt (line 10))
  Downloading sphinx-7.1.2-py3-none-any.whl.metadata (5.8 kB)
Collecting sphinx_rtd_theme==1.3.0 (from -r requirements.txt (line 11))
  Downloading sphinx_rtd_theme-1.3.0-py3-none-any.whl.metadata (4.5 kB)
Collecting myst-parser==2.0.0 (from -r requirements.txt (line 12))
  Downloading myst_parser-2.0.0-py3-none-any.whl.metadata (5.4 kB)
```

Figura C.2: Instalación de dependencias utilizadas por CALDERA

Por último, es necesario ejecutar el comando que levanta el servicio en el puerto 8888:

```
python server.py --build
```

```
root@kali:~/home/kali/Documents/TFG/caldera
# python server.py --build

2024-04-29 12:35:24 INFO     Creating new secure config in conf/local.yml
INFO     Log into Caldera with the following admin credentials:
INFO     Red
  USERNAME: red
  PASSWORD: L4McGH5543lh0L1I1YKGWa8sFFZgw9_qcSQuSquVYGo
  API_TOKEN: tfykyn7JG4bdkh9:HrmLiWLmTzpcAuhpFGstQK2bTE
Blue
  USERNAME: blue
  PASSWORD: 62cbMz9DBN0GUHsRaSN0vRWzoYd0rPRwvCN-MpzeKok
  API_TOKEN: ZK7kuZ2soEsP_rBCDK39wOUA2K60H9YFDt1eLX-k08
INFO     To modify these values, edit the conf/local.yml file.
INFO     Using main config from conf/local.yml
INFO     Building VueJS front-end.

2024-04-29 12:35:25 INFO     config_generator.py:55
INFO     config_generator.py:30
INFO     server.py:225
INFO     server.py:261
```

Figura C.3: Ejecución del comando para levantar el servicio de CALDERA



```

root@kali: /home/kali/Documents/TFG/caldera
File Actions Edit View Help
INFO    Enabled plugin: stockpile
INFO    Enabled plugin: compass
INFO    Enabled plugin: fieldmanual
INFO    Enabled plugin: maxn
INFO    Creating SSH listener on 0.0.0.0, port 8022
INFO    serving on 0.0.0.0:2222
2024-04-29 12:36:49 WARNING Ability referenced in adversary ef4d997c-a0d1-4067-9efa-87c58682db71 but not found: ff78708e0e18d31c0be7a2be295158ec
WARNING Ability referenced in adversary ef4d997c-a0d1-4067-9efa-87c58682db71 but not found: gfdc9037290299164d52b65219d628ef
WARNING Ability referenced in adversary ef4d997c-a0d1-4067-9efa-87c58682db71 but not found: ae21aeef2d2d993d7f45a4e55485fbcc33
WARNING Ability referenced in adversary ef4d997c-a0d1-4067-9efa-87c58682db71 but not found: 86abbd7ecc05b7dbe7699a96a0a173
WARNING Ability referenced in adversary ef4d997c-a0d1-4067-9efa-87c58682db71 but not found: 5c922d92f383656a01d5633ca23db497
WARNING Objective referenced in adversary ef4d997c-a0d1-4067-9efa-87c58682db71 but not found: c495a9828-cab1-44dd-a0ca-66e58177db8. Setting default objective.
INFO    Docs built successfully with the following warnings
/home/kali/Documents/TFG/caldera/plugins/fieldmanual/sphinx-docs/The-REST-API.md:3: WARNING: 'myst'
cross-reference target not found: '/api/docs'
2024-04-29 12:36:50 INFO    All systems ready.
server.py:101

```

Figura C.4: Ejecución del comando para levantar el servicio de CALDERA (II)

Además de abrir el puerto 8888 para la interfaz web, también tiene habilitados los puertos 2222 y 8022 para el servicio de SSH, que será posteriormente utilizado por los agentes como Sandcat para realizar operaciones de C2 [93].

Una vez el servicio está corriendo, simplemente basta con acceder a través de un navegador a la interfaz de `localhost:8888/login` e iniciar sesión a través de las credenciales por defecto, a las cuales se puede acceder a través de la ruta `caldera/conf/default.yml`:



```

File Actions Edit View Help
(kali㉿kali)-[~/Documents/TFG/caldera/conf]
$ batcat default.yml | tail -5
blue:
  blue: admin
red:
  admin: admin
  red: admin

```

Figura C.5: Fichero con credenciales por defecto de CALDERA

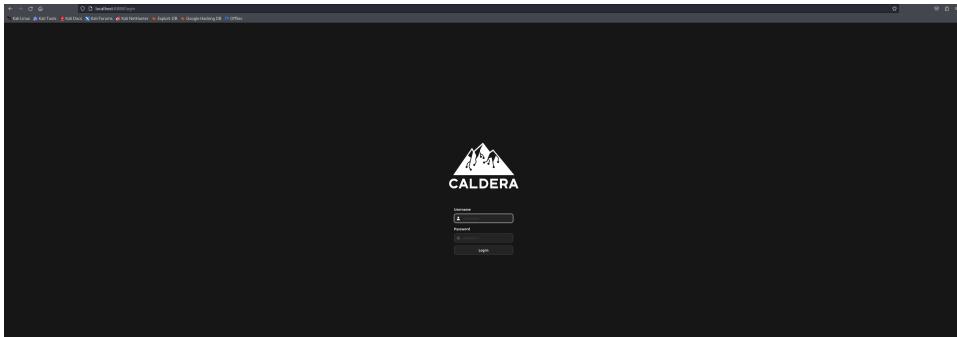


Figura C.6: Pestaña de Inicio de sesión del *framework*

## Configuración de CALDERA

Una vez se ha iniciado sesión, aparece una *dashboard* con un panel de navegación a la izquierda. Este contiene cuatro secciones, tal y como se aprecia en la Tabla C.1:

Sección	Descripción
<b>Campañas</b>	Permite a los usuarios crear y gestionar diferentes simulaciones de ataques, ajustando escenarios y monitorizando el progreso en tiempo real.
<b>Plugins</b>	Ofrece acceso a extensiones que pueden ser incorporadas para expandir la funcionalidad de CALDERA, incluyendo nuevas capacidades de inteligencia y herramientas de ataque.
<b>Configuración</b>	Permite a los administradores ajustar objetivos, contactos y archivos exfiltrados, así como gestionar cuentas de usuario.
<b>Recursos</b>	Proporciona enlaces a documentación, tutoriales y otros materiales de apoyo, así como herramientas de ofuscación de los <i>payloads</i> .

Tabla C.1: Descripción de las secciones principales en la interfaz de CALDERA

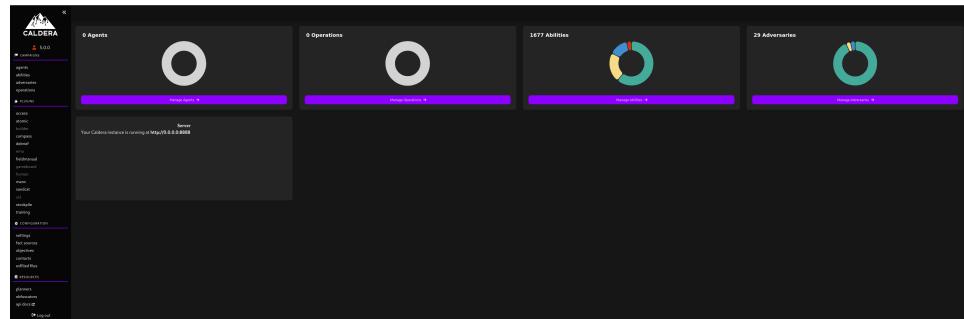


Figura C.7: Dashboard de la interfaz web del *framework*

Por otro lado, en el menú principal hay una serie de gráficas que refieren los cuatro elementos del *framework* utilizados para efectuar la simulación de los ataques, como son los agentes, operaciones, habilidades y adversarios. Inicialmente el número de agentes y operaciones está inicializado a cero, ya que estos deben de ser generados manualmente.

Se comenzará creando un agente. Para ello es necesario acceder a su sección en la barra de navegación lateral y posteriormente hacer click sobre el botón de **Deploy an agent**. Entonces aparecerá un menú emergente mediante el cual será posible seleccionar datos como el tipo de agente que se quiere utilizar, el nombre del agente y las extensiones (opcionalmente) a utilizar.

Conforme se van rellenando dichos campos se va generando un *payload* que servirá para infectar a las máquinas víctima. Se crearán además variantes de dicha carga útil que podrán ser igualmente utilizadas con el mismo objetivo.

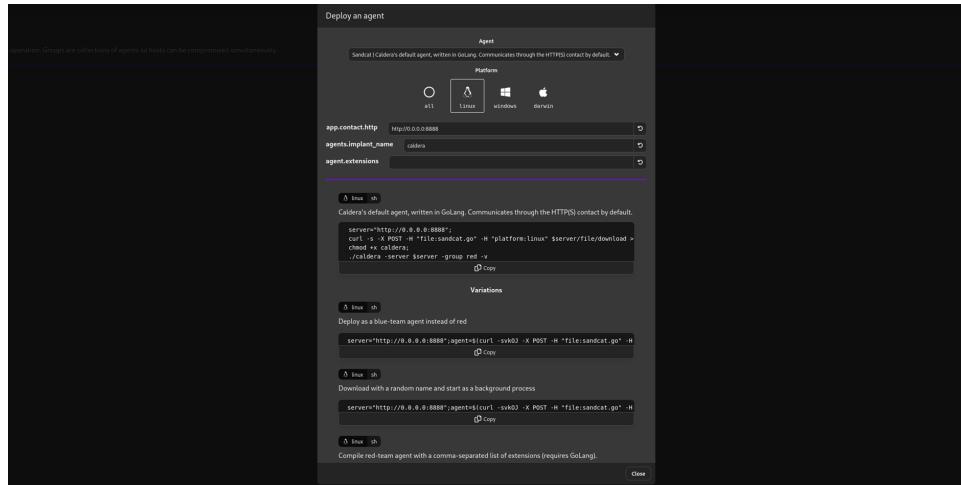


Figura C.8: Proceso de configuración de un agente de CALDERA

En este caso, las máquinas víctima son contenedores de Ubuntu Server, por ello se hará uso de la plataforma Linux y del agente recomendado por defecto: Sandcat, que realiza las comunicaciones por medio del protocolo HTTP(s).

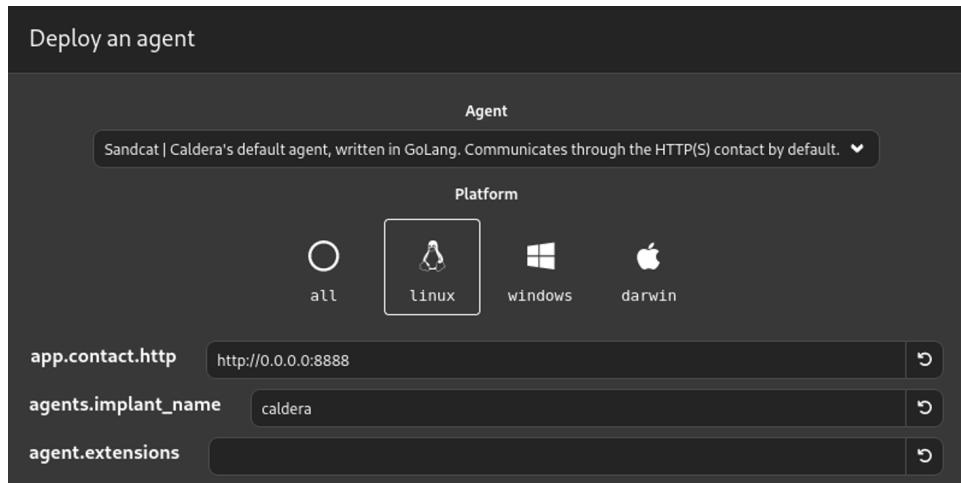
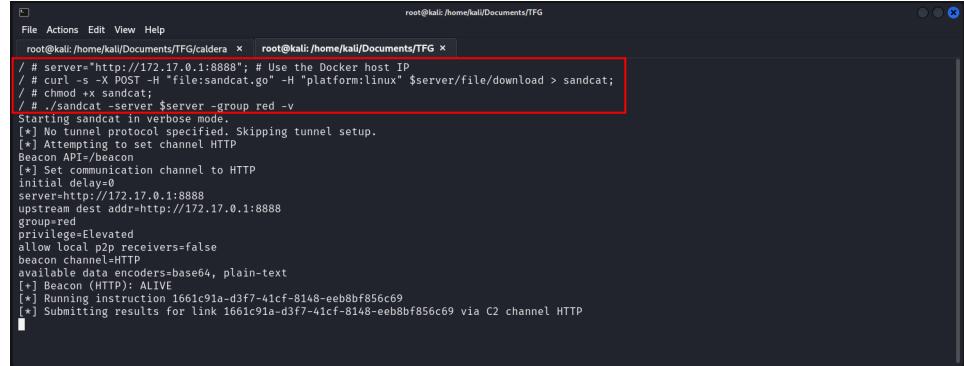


Figura C.9: Selección del agente Sandcat para la infección de las máquinas víctima

Una vez obtenido el *payload* que se va a utilizar simplemente es necesario ejecutarlo en la línea de comandos de cada una de las máquinas infectadas, y ya se creará el túnel por el cual se enviarán el resto de comandos a esta.



```

root@kali:/home/kali/Documents/TFG/caldera x root@kali:/home/kali/Documents/TFG x
/ # server="http://172.17.0.1:8888"; # Use the Docker host IP
/ # curl -s -X POST -H "file:sandcat.go" -H "platform:linux" $server/file/download > sandcat;
/ # chmod +x sandcat;
/ # ./sandcat -server $server -group red -v
Starting sandcat in verbose mode.
[*] No tunnel protocol specified. Skipping tunnel setup.
[*] Attempting to set channel HTTP
Beacon API / beacon
[*] Set the application channel to HTTP
initial delay=0
server=http://172.17.0.1:8888
upstream dest addr=http://172.17.0.1:8888
group=red
privilege=Elevated
allow local p2p receivers=false
beacon channel=HTTP
available data encoders=base64, plain-text
[+] Beacon (HTTP): ALIVE
[*] Running instruction 1661c91a-d3f7-41cf-8148-eeb8bf856c69
[*] Submitting results for link 1661c91a-d3f7-41cf-8148-eeb8bf856c69 via C2 channel HTTP
  
```

Figura C.10: Ejecución del *payload* para generar un agente conectado a la máquina

Es entonces cuando aparecerá un nuevo agente en la interfaz web con la siguiente información:

<b>id(paw)</b>	<b>host</b>	<b>group</b>	<b>platform</b>	<b>contact</b>
dsmzqo	de2297825335	red	linux	HTTP
<b>pid</b>	<b>privilege</b>	<b>status</b>	<b>last seen</b>	
66	Elevated	alive, trusted	4/29/2024, 7:02:13 PM	

Tabla C.2: Formato de la tabla de un agente de CALDERA

Una vez es posible visualizar dicho contenido, significa que los agentes han sido correctamente creados, y si el valor del campo **status** es **alive**, **trusted** significa que estos están preparados para adquirir habilidades, asignarlas a adversarios y posteriormente llevar a cabo las operaciones, es decir, los ataques.



Figura C.11: Verificación del estado de los agentes configurados

El siguiente paso es seleccionar aquellas habilidades que se quieran utilizar para las operaciones. Estas están directamente conectadas al marco de MITRE ATT&CK, por lo que se dispone de una gran cantidad de TTPs, concretamente más de 1600 habilidades. Además, es posible filtrar mediante una barra de búsqueda en función del tipo de táctica, técnica o procedimiento que se quiera utilizar, ya sea a través de lenguaje natural o bien mediante los códigos de identificación mencionados en el tercer capítulo 3.7. Además, también existe la posibilidad de agrupar un conjunto de habilidades bajo un identificador especificado manualmente, y posteriormente asignar dicho conjunto a un adversario.

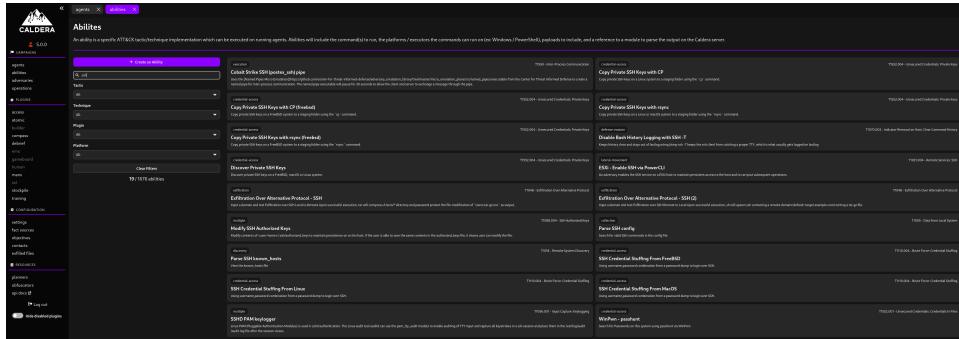


Figura C.12: Ejemplo de filtrado en el panel de navegación de habilidades de CALDERA

En tercer lugar, accediendo en el menú de navegación a la sección de adversarios debe seleccionarse entre las opciones del desplegable para crear un nuevo perfil, o bien importarlo a partir de un fichero.

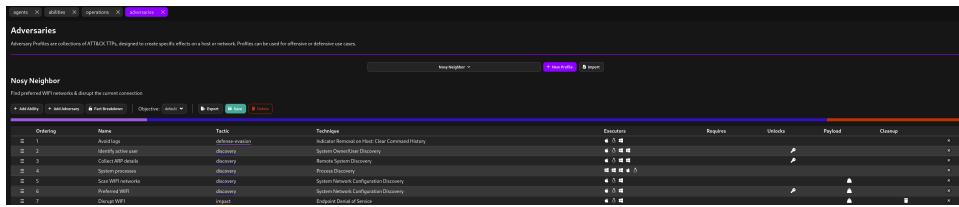


Figura C.13: Configuración de un perfil de adversario en CALDERA

Por último, para crear una nueva operación es necesario acceder a través del panel lateral y posteriormente hacer click en el botón de **+ New Operation**. Aparecerá una ventana donde se podrán llenar campos de interés como el nombre de la operación, el adversario utilizado (lo más importante), grupo al que pertenece, métodos de ofuscación (por defecto las operaciones se lanzan en texto plano salvo que se seleccione lo contrario) y otras opciones de personalización adicionales.

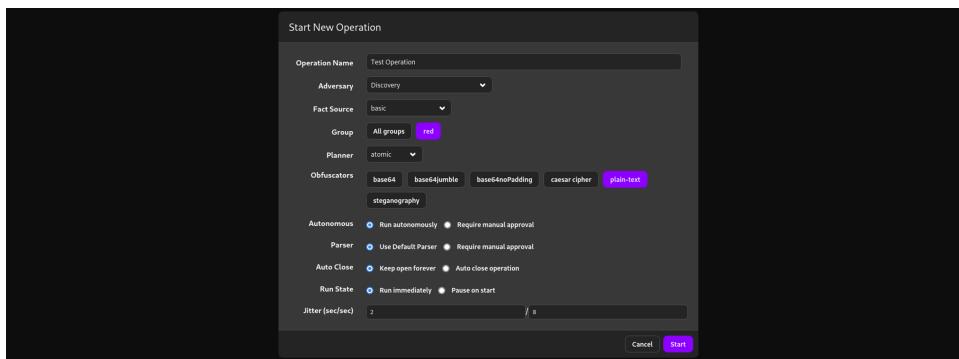


Figura C.14: Configuración de una operación de caldera a partir de un perfil adversario



# Glosario

- AHC** *Agglomerative Hierarchical Clustering.* III, IV, 27, 39, 45, 46, 72, 75, 80, 81, 83, 85, 86, 92, 94
- ANN** *Artificial Neural Network.* 91
- API** *Application Programming Interface.* 93
- APT** *Advanced Persistent Threat.* 99
- ATT&CK** *Adversarial Tactics, Techniques, and Common Knowledge.* 3–5, 13, 21, 25, 33, 47, 115, 120
- BERT** *Bidirectional Encoder Representations from Transformers.* 71
- BGL** *Blue Gene/L.* III, IV, 5, 28, 29, 36, 37, 39, 64, 65, 83, 84
- BIRCH** *Balanced Iterative Reducing and Clustering using Hierarchies.* 39
- C2** *Command & Control.* 23, 117
- CALDERA** *Cyber Adversary Language and Detection Engine for RTA.* II–V, 3–5, 25, 33, 47–49, 55, 56, 59, 60, 115–119, 121
- CC** *Common Criteria.* 38
- CCN** *Centro Criptológico Nacional.* 1
- CERT** *Computer Emergency Readiness Team.* 92
- CI/CD** *Continuous Integration & Continuous Delivery/Deployment.* 111
- CIA** *Confidencialidad, Integridad y Disponibilidad.* 1
- CLIQUE** *Clustering In QUEst.* 39
- CPSTIC** *Catálogo de Productos y Servicios de Seguridad de las TIC del CCN.* 1
- CSV** *Comma-Separated Values.* 51, 61, 63, 65
- CTF** *Capture The Flag.* 4, 5, 109
- CVE** *Common Vulnerabilities and Exposures.* 1, 49
- DAC** *Divide and Conquer.* 94
- DBSCAN** *Density-Based Spatial Clustering of Applications with Noise.* III, 6, 27, 39, 43–45, 51, 74, 79, 81, 85, 89, 92, 94, 124

- DCS** *Distributed Control System.* 22
- DDoS** *Distributed Denial-of-Service.* 16, 91, 93
- DENCLUE** *Density-based Clustering.* 39
- DenMune** *Density Peak Based Clustering Using Mutual Nearest Neighbors.* 39, 92
- DF** *Data-Frame.* 66, 74
- DHC** *Divisive Hierarchical Clustering.* 27, 39
- DL** *Deep Learning.* 4, 26, 29, 50, 90, 91, 107
- DnD** *Drag & Drop.* 63
- DoS** *Denial of Service.* 38
- EDR** *Endpoint Detection and Response.* 30, 93
- ELK** *Elasticsearch, Logstash, y Kibana.* 17, 32
- ENISA** *European Network and Information Security Agency.* 2
- ESM** *Enterprise Security Manager.* 32
- FINCH** *First Integer Neighbor Clustering Hierarchy.* 39
- FS** *File System.* 93
- FTP** *File Transfer Protocol.* 18
- GDDR** *Graphics Double Data Rate.* 8
- GDPR** *General Data Protection Regulation.* 33
- Gemini** *Generalized Multimodal Intelligence Network Interface.* 92
- GMM** *Gaussian Mixture Model.* 92
- GNU** *Gnu's Not Unix.* 55
- GPT** *Generative Pre-training Transformer.* 61, 92
- GTX** *Giga Texel Shader eXtreme.* 8
- HC** *Hierarchical Clustering.* IV, 27, 72, 75, 85, 86
- HDBSCAN** *Hierarchical DBSCAN.* 39, 92
- HDFS** *Hadoop Distributed File System.* III, IV, 5, 28, 29, 37, 39, 64, 65, 83
- HIDS** *Host Intrusion Detection System.* 30
- HTML** *HyperText Markup Language.* 52
- HTTP** *Hypertext Transfer Protocol.* 119
- I+D** *Investigation & Development.* 26, 94, 97
- IA** *Inteligencia Artificial.* 2, 3, 13, 27, 93, 107

- IBM** *International Business Machines.* 32, 83
- ICS** *Industrial Control System.* 22
- IDF** *Inverse Document Frequency.* iv, 3, 5, 65, 69–71, 79, 80, 82–84, 89
- IDS** *Intrusion Detection System.* 30, 93
- IP** *Internet Protocol.* iii, 24, 25, 35, 59, 66, 67
- IPS** *Intrusion Prevention System.* 30
- IRS** *Intrusion Response System.* 30
- ISO** *Internacional Organization for Standardization.* 36
- IT** *Information Technologies.* 17
- IWSVM** *Instance-Weighted SVM.* 94
- JSON** *JavaScript Object Notation.* 64, 65
- KFCV** *K-Fold Cross-Validation.* 39
- KNN** *K-Nearest Neighbors.* 91, 95, 97
- LAVA** *Logic for Advanced Virtual Adversaries.* 47, 48
- LLM** *Large Language Models.* 2, 26, 55, 92
- LOOCV** *Leave-One-Out Cross-Validation.* 39
- LSTM** *Long Short-Term Memory.* 91
- MIT** *Massachusetts Institute of Technology.* 111, 112
- ML** *Machine Learning.* 4, 26, 29, 33, 50, 69, 89–91, 93–95, 97
- MTTD** *Main Time Till Detection.* 31, 94
- MTTR** *Main Time Till Response.* 31, 94
- NaN** *Not a Number.* 91
- NFS** *Network File System.* 16
- NIDS** *Network Intrusion Detection System.* 30
- NIST** *National Institute of Standards and Technology.* 21
- NLP** *Natural Language Processing.* 4, 5, 108
- NTP** *Network Time Protocol.* 18
- NVMe** *Non-Volatile Memory Express.* 8
- OC-SVM** *One-Class Support Vector Machine.* 89
- ODS** *Objetivos de Desarrollo Sostenible.* v, 3
- ONU** *Organización de las Naciones Unidas.* 3

- OPTICS** *Ordering Points To Identify the Clustering Structure.* 39
- OSINT** *Open-Source Intelligence.* 93
- OSS** *Open-source software.* 32
- OSSIM** *Open Source Security Information Management.* 32
- Pandas** *Python Data Analysis.* 51
- PCA** *Principal Component Analysis.* 71, 89, 91
- PLC** *Programmable Logic Controller.* 22
- PR** *Pull Request.* 111
- RAT** *Remote Access Tool.* 47
- RE** *Regular Expressions.* 51
- RFC** *Request For Comments.* 17
- RL** *Reinforcement Learning.* 26
- RRNN** *Redes Neuronales.* 97, 107
- RTA** *Red Team Automation.* 25, 123
- RTU** *Remote Terminal Unit.* 22
- SCADA** *Supervisory Control and Data Acquisition.* 22, 97
- SEM** *Security Events Management.* 30
- SFR** *Security Functional Requirement.* 38
- SFTP** *Secure File Transfer Protocol.* 55
- SIEM** *Security Information and Event Management.* I, II, 3–5, 13, 30–33, 89, 90, 93–95, 97, 109, 111
- SIM** *Security Information Management.* 30
- SL** *Supervised Learning.* 26
- SMOTE** *Synthetic Minority Over-sampling Technique.* 64, 91
- SNN** *Shared Nearest Neighbor.* 39
- SOAR** *Security Orchestration, Automation, and Response.* 30, 93, 94
- SOC** *Security Operations Center.* 1, 30
- SQL** *Structured Query Language.* 97
- SSD** *Solid State Drive.* 8
- SSH** *Secure Shell.* III, 38, 55, 60, 61, 80, 84, 117
- SSL** *Semi-Supervised Learning.* 26
- STING** *Statistical Information Grid.* 39
- SUDO** *Super User DO.* 16

- SVD** *Singular Value Decomposition*. 91
- SVM** *Support Vector Machine*. 91, 94, 95, 97, 125
- t-SNE** *t-distributed Stochastic Neighbor Embedding*. IV, 71, 79–86, 89
- TCP** *Transmission Control Protocol*. 58
- TF** *Term Frequency*. IV, 3, 5, 69–71, 79, 80, 82–84, 89
- TNI** *Total Number of Incidents*. 31
- TNIR** *Total Number of Incidents Remediated*. 31
- TSD** *Total Sum of Detection time*. 31
- TSDR** *Total Sum of Detection to Remediation time*. 31
- TPP** *Tactics, Techniques and Procedures*. 21, 48, 120
- TTY** *Talk To You*. 35
- UDP** *User Datagram Protocol*. 58
- UEBA** *User and Entity Behavior Analytics*. 91, 92
- UHAC** *Unsupervised Hunting of Anomalous Commands*. 89
- UL** *Unsupervised Learning*. 26
- UMAP** *Uniform Manifold Approximation and Projection*. IV, 52, 71, 84
- UTC** *Universal Time Coordinated*. 36
- UUCP** *Unix to Unix Copy Protocol*. 18
- VC** *Computer Vision*. 107
- ViRTS** *Virtual Red Team System*. 47
- VM** *Virtual Machine*. 56, 64
- VPN** *Virtual Private Network*. 16
- WAF** *Web Application Firewall*. 30
- WSL** *Windows Subsystem for Linux*. 49
- XSS** *Cross Site Scripting*. 93, 97



