Informe Técnico - Technical Report
DPTOIA-IT-2024-016
Julio, 2024

Aplicación de técnicas de búsqueda elástica e IA para la creación de un sistema de inteligencia de amenazas

Creado por Javier García Pechero
Revisado por Juan Manuel Corchado Rodríguez

Departamento de Informática y Automática
Universidad de Salamanca

Revisado por:
Dr. - Juan Manuel Corchado Rodríguez -

Información de los Autores:

Javier García Pechero
Ingeniero Informático
Estudiante del Máster en Sistemas Inteligentes
Universidad de Salamanca (USAL)
javigp@usal.es

Resumen

El presente informe técnico de fin de máster detalla el diseño, implementación y validación de un sistema avanzado de inteligencia de amenazas que integra técnicas de búsqueda elástica e inteligencia artificial. Este sistema, recopila, procesa y analiza información de reportes de ciberseguridad de empresas reconocidas del sector, mejorando significativamente la eficiencia y eficacia en la detección de amenazas cibernéticas. La contribución principal es la creación de una ontología comprehensiva para la inteligencia de amenazas en ciberseguridad, basada en el marco MITRE ATT&CK. Esta ontología estandariza y estructura el conocimiento en el área, facilitando la comunicación entre sistemas y profesionales, y mejorando significativamente la identificación y análisis de amenazas.

El sistema destaca por su capacidad para procesar datos no estructurados de reportes de ciberseguridad, utilizando un modelo de lenguaje avanzado con procesamiento del lenguaje natural para extraer entidades, como vulnerabilidades, tácticas, técnicas y procedimientos (TTP) de atacantes, e indicadores de compromiso (IoC). Estas entidades se almacenan en una base de datos que emplea técnicas de búsqueda elástica, permitiendo una recuperación eficiente de la información.

Para mejorar la precisión en el reconocimiento de entidades, se han aplicado técnicas de ajuste fino eficiente de parámetros (PEFT) al modelo de lenguaje, adaptándolo más precisamente al dominio de la ciberseguridad. El sistema ofrece una aplicación web que facilita el acceso gráfico a las funcionalidades de procesamiento, extracción y análisis de las entidades. Además, automatiza la creación de un grafo de conocimiento para representar visualmente la ontología con sus respectivas entidades, proporcionando una herramienta poderosa para la comprensión y análisis de las relaciones entre diferentes elementos de seguridad.

Validado en escenarios reales, el sistema demuestra su capacidad para mejorar significativamente la eficacia de los profesionales de ciberseguridad, minimizando los prolongados periodos necesarios para extraer entidades de los informes. Este proyecto representa un avance significativo en el campo de la inteligencia de amenazas, proporcionando a las organizaciones una herramienta poderosa para anticiparse y responder eficazmente a las amenazas cibernéticas en un entorno digital cada vez más complejo y dinámico. La integración del marco MITRE ATT&CK asegura que el sistema esté alineado con las mejores prácticas de la industria, facilitando la evaluación y el fortalecimiento de las estrategias de seguridad a través de una terminología común y un enfoque estructurado para la comprensión de las tácticas y técnicas de los atacantes.

Abstract

This technical report details the design, implementation and validation of an advanced threat intelligence system that integrates elastic search and artificial intelligence techniques. This system collects, processes and analyzes information from cybersecurity reports of well-known companies in the industry, significantly improving the efficiency and effectiveness of cyber threat detection. A major contribution is the creation of a comprehensive ontology for cybersecurity threat intelligence, based on the MITRE ATT&CK framework. This ontology standardizes and structures knowledge in the area, facilitating communication between systems and professionals, and significantly improving threat identification and analysis.

The system stands out for its ability to process unstructured data from cybersecurity reports, using an advanced language model with natural language processing to extract key entities such as vulnerabilities, tactics, techniques and procedures (TTP) of attackers, and indicators of compromise (IoC). These entities are stored in a database that employs elastic search techniques, enabling efficient information retrieval.

To improve entity recognition accuracy, parameter efficient fine tuning (PEFT) techniques have been applied to the language model, adapting it more precisely to the cybersecurity domain. The system provides a web application that facilitates graphical access to entity processing, extraction and analysis functionalities. In addition, it automates the creation of a knowledge graph to visually represent the ontology with its respective entities, providing a powerful tool for understanding and analyzing the relationships between different security elements.

Validated in real-world scenarios, the system demonstrates its ability to significantly improve the efficiency of cybersecurity professionals, minimizing the lengthy periods required to extract entities from reports. This project represents a significant advancement in the field of threat intelligence, providing organizations with a powerful tool to effectively anticipate and respond to cyber threats in an increasingly complex and dynamic digital environment. The integration of the MITRE ATT&CK framework ensures that the system is aligned with industry best practices, facilitating the assessment and strengthening of security strategies through common terminology and a structured approach to understanding attacker tactics and techniques.

Índice

Índice de figuras

Indice de tablas

1. Introducción

En el panorama actual de la ciberseguridad, la inteligencia de amenazas se ha convertido en un elemento crítico para la protección de los activos digitales organizacionales. En respuesta a esta necesidad, se ha desarrollado una plataforma innovadora que integra una ontología especializada en el dominio de la ciberseguridad, basada en la terminología del marco MITRE ATT&CK, con un sistema de procesamiento del lenguaje natural entrenado con corpus específicos del sector. Esta plataforma incorpora un extractor de entidades que permite asociar información relevante a la ontología creada, y utiliza un motor de búsqueda elástica para almacenar y recuperar datos de manera eficiente. El resultado es un repositorio de inteligencia de amenazas que proporciona información valiosa para los Centros de Operaciones de Seguridad (SOC), permitiendo anticiparse a las metodologías de los atacantes y mejorar la postura de seguridad de las organizaciones.

El presente trabajo se enfoca en el diseño, implementación y evaluación de esta plataforma de inteligencia de amenazas, explorando cómo la combinación de ontologías especializadas, técnicas avanzadas de procesamiento del lenguaje natural y sistemas de almacenamiento y recuperación de alta velocidad pueden mejorar significativamente la capacidad de las organizaciones para detectar, analizar y responder a las amenazas cibernéticas emergentes.

1.1. Objetivos

El objetivo principal de este proyecto es el diseño, implementación y evaluación de una plataforma innovadora de inteligencia de amenazas en ciberseguridad, enfocada en mejorar la capacidad de las organizaciones para anticipar, detectar y responder a amenazas cibernéticas emergentes.

Los objetivos específicos del proyecto incluyen:

- Desarrollo de una ontología especializada: Diseñar y construir una ontología robusta centrada en el dominio de la ciberseguridad, incorporando la terminología estandarizada del marco MITRE ATT&CK, para facilitar la compartición de información entre sistemas automatizados y analistas humanos.

- Implementación de un sistema de procesamiento del lenguaje natural (NLP) con extractor de entidades: Desarrollar y entrenar un modelo de NLP con corpus específicos de ciberseguridad, integrando un componente capaz de identificar y extraer automáticamente entidades relevantes a partir de fuentes de datos no estructuradas, permitiendo su asociación dinámica con los conceptos definidos en la ontología desarrollada.

- Incorporación de un motor de búsqueda elástica y repositorio centralizado: Implementar un sistema de almacenamiento y recuperación de alta velocidad que facilite el análisis en tiempo real y la correlación compleja de los datos,

construyendo una base de conocimientos que albergue indicadores de compromiso, tácticas, técnicas y procedimientos (TTPs) de atacantes, y tendencias del panorama de amenazas cibernéticas.

- Optimización de la integración de componentes: Asegurar la sinergia efectiva entre la ontología, el sistema de NLP con extractor de entidades y el motor de búsqueda, maximizando la eficiencia y precisión del sistema en su conjunto.

- Validación en escenarios reales: Realización de pruebas de la plataforma en entornos operativos de ciberseguridad, evaluando su capacidad para proporcionar inteligencia accionable y mejorar la toma de decisiones en la detección, respuesta y mitigación de incidentes de seguridad.

2. Estado de la cuestión: Mapping Sistemático de la Literatura

En el contexto actual de rápida evolución tecnológica y amenazas cibernéticas emergentes, los sistemas de inteligencia de amenazas requieren una capacidad de adaptación y respuesta sin precedentes. La integración de técnicas avanzadas de búsqueda elástica e inteligencia artificial se ha transformado en una necesidad para el desarrollo de estrategias de defensa más efectivas y proactivas.

La inteligencia de amenazas se fundamenta en la habilidad de identificar y analizar con precisión las amenazas potenciales a partir de grandes volúmenes de datos no estructurados. En este campo, las herramientas de Procesamiento de Lenguaje Natural (NLP), particularmente las destinadas a la extracción y gestión de conocimientos basados en ontologías, desempeñan un papel fundamental. Estas herramientas no solo facilitan la identificación de entidades específicas en textos complejos, sino que también enriquecen la comprensión contextual necesaria para interpretar correctamente las amenazas.

La incorporación de ontologías en sistemas basados en NLP proporciona un marco estructurado que define con precisión las relaciones entre los conceptos clave del dominio de las amenazas cibernéticas. Esta base sólida mejora significativamente la generación de alertas precisas y la mitigación de riesgos.

Esta sección propone un mapeo sistemático de la literatura existente y las metodologías aplicadas en la intersección de la inteligencia artificial y el procesamiento de lenguaje natural. El objetivo es desarrollar un sistema robusto de inteligencia de amenazas que no solo avance en la comprensión académica y práctica de las técnicas eficaces, sino que también establezca un fundamento teórico y práctico para futuras investigaciones en este campo vital.

2.1. Proceso de Mapping

El propósito fundamental de esta revisión sistemática [9] es compilar y examinar exhaustivamente la literatura existente relacionada con la implementación de técnicas de inteligencia artificial en el desarrollo de sistemas de inteligencia de amenazas. El análisis se centra en tres ejes principales: las metodologías aplicadas, la tipología de técnicas empleadas y los contextos específicos de implementación, abarcando aspectos como el nivel de datos y los tipos de amenazas abordadas.

Para lograr este objetivo, el proceso se estructura en dos fases:

1. **Planificación**
2. **Ejecución**

La implementación de la herramienta Parsifal[1] en este proceso de revisión sistemática permitirá la construcción de un mapping organizado y estructurado.

2.1.1. Fase de planificación

El primer paso del procedimiento metodológico incluye una planificación detallada de la revisión sistemática. Esta fase se compone de los siguientes puntos:

- Formulación de preguntas de mapping (MQs): Se definen con claridad las preguntas que la revisión sistemática busca responder, estableciendo el marco del estudio.
- Construcción de las cadenas de búsqueda: Se desarrollan estrategias de búsqueda efectivas, utilizando términos clave relacionados con el tema del mapping.
- Identificación de fuentes bibliográficas: Se seleccionan las bases de datos y repositorios académicos donde el Proceso de Mapping Sistemático (SMP) realizará las búsquedas de literatura.
- Establecimiento de criterios de selección: Se determinan criterios claros de inclusión y exclusión para seleccionar los estudios relevantes, asegurando la calidad de la investigación.
- Definición de parámetros de evaluación de calidad: Se establecen criterios para evaluar la solidez metodológica y la relevancia de los estudios seleccionados.

En esta fase de planificación, la plataforma Parsifal ofrece funcionalidades especializadas que optimizan el proceso de creación de los puntos anteriores.

[1] https://parsif.al

2.1.1.1. Preguntas de mapeo (MQs)

Para llevar a cabo la revisión en base al objetivo principal definido en la introducción, se formularon las siguientes preguntas:

- MQ1: ¿Cómo se pueden integrar las ontologías en sistemas de procesamiento de lenguaje natural para mejorar la identificación de amenazas cibernéticas?
- MQ2: ¿Cuáles son las ventajas y limitaciones de las técnicas de búsqueda elástica en la inteligencia de amenazas de ciberseguridad?
- MQ3: ¿Qué modelos de inteligencia artificial son más efectivos para analizar y predecir amenazas en sistemas de seguridad informática?
- MQ4: ¿Cómo contribuyen las técnicas avanzadas de NLP, como el análisis semántico y la comprensión del lenguaje natural, al procesamiento de datos en ciberseguridad?
- MQ5: ¿Qué papel tiene la minería de datos en la identificación de patrones ocultos en grandes conjuntos de datos de ciberseguridad y cómo se puede mejorar su aplicación?

Estas preguntas se pueden descomponer para responder a quién, cómo y qué, lo que ayuda a identificar elementos fundamentales de cada pregunta establecida.

Para formalizar estas preguntas, se ha empleado el marco PICOC, que permite organizar las preguntas en una estructura formal, descomponiéndolas en los conceptos que las conforman:

- **P**oblación (Population): Sistemas de inteligencia de amenazas en ciberseguridad.
- **I**ntervención (Intervention): Aplicación de técnicas de procesamiento de lenguaje natural, búsqueda elástica, inteligencia artificial y minería de datos.
- **C**omparación (Comparison): Comparación entre diferentes enfoques y técnicas dentro de cada área (ontologías, búsqueda elástica, IA, NLP, minería de datos).
- **O**utcome (Resultado): Mejora en la identificación y predicción de amenazas cibernéticas, eficacia en el procesamiento de datos de seguridad.
- **C**ontexto (Context): Entornos de ciberseguridad y defensa de redes.

A partir de las interrogantes MQs y empleando el marco PICOC, se derivan las siguientes palabras clave junto con sus sinónimos correspondientes.

Keyword	Synonyms	Related to
Cybersecurity	Cyber defense Information security Network security	Population
Data extraction	Content extraction Data harvesting Information retrieval	Outcome
Elastic search	Dynamic search Flexible querying Scalable search	Comparison
Natural Language Processing (NLP)	Computational linguistics Language analysis Text mining	Intervention
Ontology	Conceptual model Knowledge model Semantic framework	Intervention
Pattern recognition	Data pattern analysis Pattern detection Trend analysis	Outcome
Query processing	Data retrieval Query optimization Search algorithm	Comparison
Threat intelligence	Cyber threat intelligence Security intelligence Threat detection	Comparison

Tabla 1: Palabras clave PICOC

2.1.1.2. Cadena de búsqueda

Tras la definición de las preguntas de mapping, es esencial establecer la cadena de búsqueda. Esta debe incluir todos los términos relevantes que se utilizarán en el proceso de búsqueda para responder a las MQs (ver sección 2.1.1.1). En este caso, los términos seleccionados son:

- ”Ontología” o ”Modelo de conocimiento” en combinación con términos como sistemas ”NLP”.
- ”Inteligencia artificial” en combinación con ”modelos” e ”inteligencia de amenazas”.
- ”Procesamiento de lenguaje natural” o NLP” combinado con ”análisis semántico”, ”comprensión del lenguaje natural”.
- ”Minería de datos” en combinación con ”identificación de patrones”.

Utilizando combinaciones de operadores booleanos (AND y OR), definimos una cadena de búsqueda que puede variar dependiendo de la fuente. La cadena de búsqueda genérica utilizada es la siguiente:

```
("Cybersecurity" OR "Information security" OR "Cyber defense"
OR "Network security")
AND
("Ontology" OR "Semantic framework" OR "Knowledge model" OR "Conceptual model")
AND
("Threat intelligence" OR "Cyber threat intelligence" OR "Threat detection"
OR "Security intelligence")
AND
("Natural Language Processing" OR "NLP" OR "Computational linguistics")
AND
("Data extraction" OR "Information retrieval" OR "Data harvesting"
OR "Content extraction")
```

2.1.1.3. Selección de fuentes

La siguiente decisión a tomar fue seleccionar dónde buscar estudios, lo cual determinará el alcance del mapeo. En este caso, consideramos algunas de las bibliotecas más populares y completas, teniendo en cuenta los temas de este estudio:

- ACM Digital Library[2]
- IEEE Digital Library[3]
- ScienceDirect[4]
- Scopus[5]

Estas plataformas son reconocidas por su rigor académico y la calidad de las publicaciones que indexan, lo que asegura la obtención de un corpus de investigación exhaustivo y relevante para abordar las preguntas de investigación planteadas en el estudio.

2.1.1.4. Criterios de inclusión y exclusión

Una vez determinadas las fuentes para la búsqueda bibliográfica, es importante establecer criterios de selección que permitan filtrar los estudios relevantes basándose en sus metadatos, título y resumen. Estos criterios son esenciales para determinar qué artículos se incluirán en el proceso de revisión y cuáles serán descartados.

[2]http://portal.acm.org
[3]http://ieeexplore.ieee.org
[4]http://www.sciencedirect.com
[5]http://www.scopus.com

Los criterios de inclusión son los siguientes:

- **IC1**: Artículos en revistas indexadas y arbitradas.
- **IC2**: Artículos que reporten pruebas empíricas y resultados verificables.
- **IC3**: Estudios publicados en los últimos 15 años.
- **IC4**: Estudios que desarrollen o utilicen ontologías para la inteligencia.
- **IC5**: Publicaciones en inglés o español.
- **IC6**: Trabajos que utilicen NLP en ciberseguridad.

Los criterios de exclusión son los siguientes:

- **EC1**: Artículos no arbitrados o publicados en revistas no indexadas.
- **EC2**: Artículos que no incluyan ontologías aplicadas a la inteligencia.
- **EC3**: Estudios que no apliquen NLP específicamente a ciberseguridad.
- **EC4**: Estudios sin datos empíricos o resultados no verificables.
- **EC5**: Literatura gris o documentos no científicos.
- **EC6**: Publicaciones anteriores a los últimos 15 años.

2.1.1.5. Evaluación de calidad

Tras la inclusión de artículos que cumplen con los criterios establecidos, el proceso continúa con la verificación de la calidad de los estudios, reteniendo solo aquellos que presentan una calidad aceptable. Para esta evaluación, se ha elaborado una lista de verificación de cinco preguntas. Todas las preguntas admiten respuestas de "SÍ", "PARCIAL" o "NO", asignándoseles puntajes de 1, 0.5 y 0, respectivamente. Las preguntas de calidad son las siguientes:

- **QC1**: ¿Los objetivos del estudio están claramente especificados y son relevantes para el tema de investigación?
- **QC2**: ¿Se identifican claramente las metodologías utilizadas, incluyendo técnicas de NLP y ontologías?
- **QC3**: ¿Se describe detalladamente el proceso de NLP utilizado en el estudio?
- **QC4**: ¿Se proporciona una base teórica o empírica sólida para la elección y aplicación de las ontologías?
- **QC5**: ¿Se proporcionan suficientes detalles sobre los datos, metodologías y análisis para permitir la replicación del estudio?

La nota de corte se ha establecido en 4.0, optando por un puntaje elevado que refleja la complejidad del estudio. Este umbral garantiza la selección exclusiva de investigaciones que se adhieren rigurosamente a las características fundamentales del tema investigado.

2.1.2. Fase de ejecución

Durante la segunda fase del proceso de mapeo sistemático, se procede a realizar la realización de cada una de las etapas definidas durante la fase de planificación (ver sección 2.1.1). Para este propósito, se ha adoptado el marco PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analysis) en su versión de 2020 [25], cuyo objetivo es asistir a los investigadores en la mejora de la calidad de las revisiones sistemáticas. Se ha utilizado específicamente el diagrama de flujo que PRISMA sugiere, organizando el proceso en las siguientes etapas claramente definidas:

Identificación: Se lleva a cabo la implementación de la cadena de búsqueda en las fuentes seleccionadas, la importación de estudios y la eliminación de duplicados.

Selección: Se aplican los criterios de inclusión y exclusión para filtrar los artículos pertinentes a la investigación.

Elegibilidad: En esta fase se evalúa la calidad de los estudios que avanzaron del filtrado anterior. Se determina que artículos serán analizados detalladamente.

2.1.2.1. Identificación

Esta etapa implica la aplicación de la cadena de búsqueda en cada una de los repositorios de estudios científicos seleccionados (ver sección 2.1.1.2). Para garantizar una cobertura exhaustiva y precisa de la literatura relevante, se ajustó y refinó la cadena de búsqueda en cada una. El total de estudios obtenidos fue de 155. Se obtuvieron los siguientes resultados:

Repositorio	**Num Estudios**
Scopus	53
IEEE Xplore	4
ScienceDirect	45
ACM Digital Library	54
Total	**155**

Tabla 2: Distribución de estudios por repositorio

Para presentar la distribución de los estudios recopilados por cada repositorio, se utiliza un gráfico que ilustra la proporción de estudios correspondiente a cada fuente. Este tipo de visualización permite una comprensión rápida y clara de dónde se han obtenido la mayoría de los estudios y cómo se comparan los diferentes repositorios en términos de contribución al conjunto de datos total.

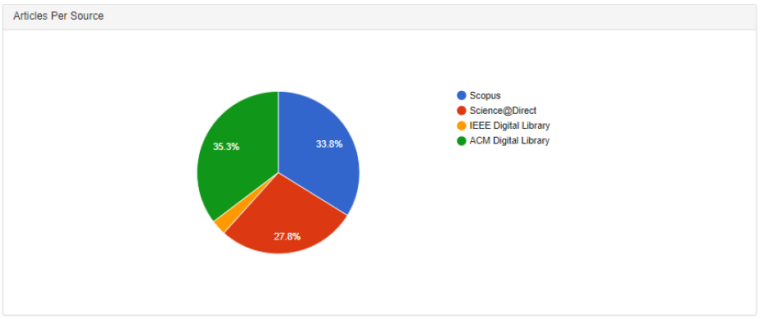

Figura 1: Articulos por repositorio

Además, se muestra un gráfico que detalla el año de publicación de los artículos. La observación de este gráfico revela una tendencia alcista en la cantidad de publicaciones a lo largo de los años, lo que indica un creciente interés y desarrollo en el campo estudiado. Esta tendencia alcista es un indicativo de que el área de investigación está enfocada en tecnologías o temas emergentes, reflejando un dinamismo en la innovación y en la producción académica relacionada con nuevas tecnologías.

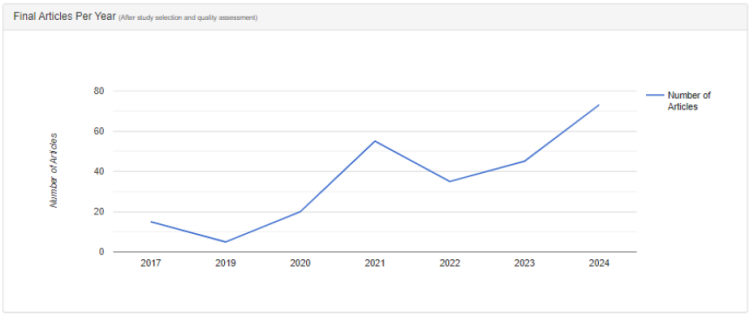

Figura 2: Articulos publicados por año

2.1.2.2. Selección

Durante la fase de selección, se aplicaron los criterios de inclusión y exclusión previamente establecidos a todos los estudios recuperados de las distintas fuentes. Este proceso crítico facilitó la filtración y selección de aquellos estudios que cumplían con los requisitos definidos, garantizando la relevancia directa de cada artículo respecto a las preguntas de investigación planteadas. De los 155 estudios seleccionados tras eliminar duplicados, se obtuvieron 133.

Al concluir el proceso de selección y filtrado aplicando los criterios de inclusión y exclusión, se obtuvieron un total de 50 estudios pertinentes para el análisis. Este resultado refleja la exhaustividad del método empleado para garantizar la relevancia y calidad de las investigaciones consideradas en el estudio.

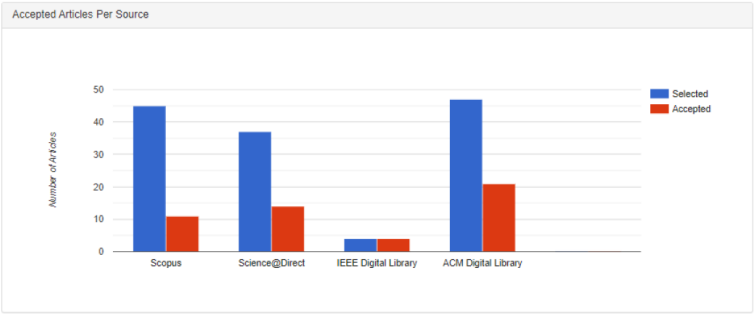

Figura 3: Articulos aceptados por fuente

2.1.2.3. Elegibilidad

En esta etapa se lleva a cabo una evaluación detallada de la calidad mediante preguntas específicas, lo que requiere una lectura en profundidad de cada uno de los artículos seleccionados. Se han identificado un total de n=14 estudios que alcanzan o superan el umbral de puntuación establecido en **4.0**.

El siguiente diagrama representa el flujo de trabajo aplicado en las secciones anteriores, que sirve para entender los diversos filtros y criterios aplicados a lo largo del mapping, asegurando que solo los datos más relevantes sean considerados para el análisis final.

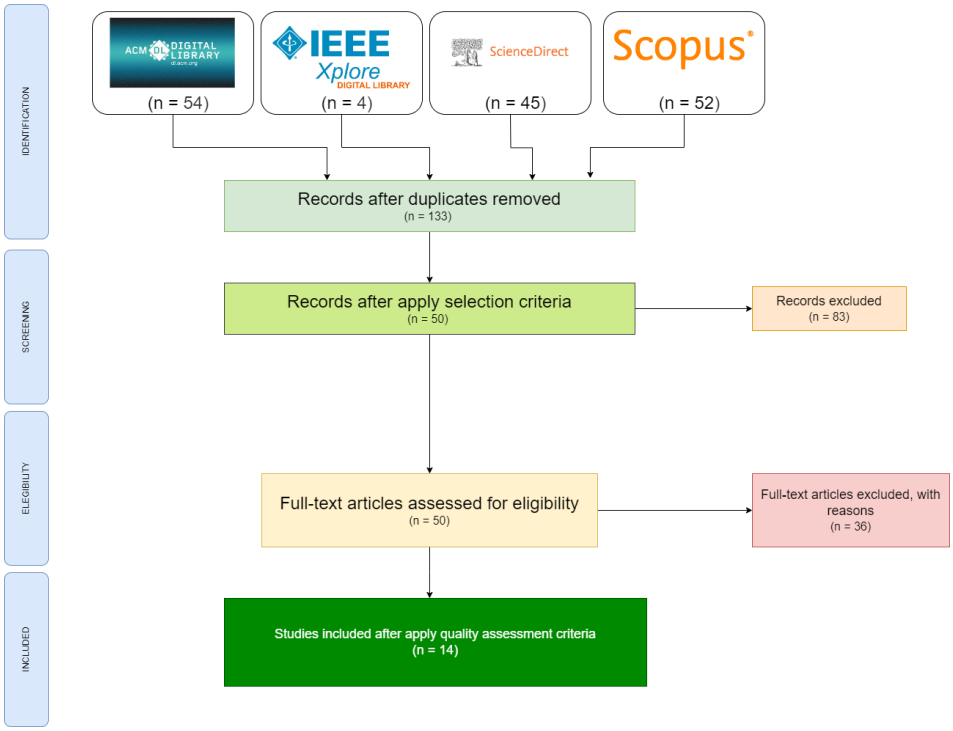

Figura 4: Diagrama mapping

2.1.3. Extracción y análisis de resultados

En esta sección se presentan los resultados obtenidos a partir de la aplicación de la fase de ejecución (ver sección 2.1.2), es decir, los estudios elegidos permiten formular las respuestas a las preguntas de mapeo (MQs) establecidas.

Los resultados finales del mapping sobre todo se han centrado en:

- El tipo de tecnología de procesamiento de lenguaje natural (NLP) utilizada.
- Las técnicas de inteligencia artificial empleadas.
- El uso de ontologías para mejorar la identificación de amenazas cibernéticas.

Cada estudio seleccionado ha sido examinado al detalle para determinar su contribución específica en estos ámbitos, permitiendo así una comprensión detallada y estructurada de cómo las diferentes tecnologías están siendo aplicadas y desarrolladas dentro del campo de la ciberseguridad.

2.1.3.1. MQ1: ¿De qué manera las ontologías pueden ser eficazmente integradas en los sistemas de procesamiento de lenguaje natural para mejorar la precisión en la identificación de amenazas cibernéticas?

La integración de ontologías en los sistemas de procesamiento de lenguaje natural (NLP) es esencial para la identificación precisa de amenazas cibernéticas. Estas estructuras permiten organizar y relacionar meticulosamente la información de inteligencia de amenazas, mejorando así la comprensión semántica y contextual de los datos. Un ejemplo de esto es el uso de una terminología basada en el conocido marco MITRE ATT&CK[6], como se propone en el trabajo [15]. La creación de la ontología facilita la extracción y almacenamiento de tripletas de inteligencia mediante técnicas de extracción de información, y potencia la capacidad de los sistemas para realizar razonamientos automáticos y consultas complejas usando SPARQL[7].

Además, la creación de grafos de conocimiento en el ámbito de la ciberseguridad, explorada en [44], emplea ontologías para estructurar información sobre amenazas cibernéticas. La integración de tecnologías avanzadas de NLP, como el modelo de lenguaje BERT[8] [8], que es un modelo de procesamiento de lenguaje natural desarrollado por Google, se caracteriza por su capacidad de entender el contexto de una palabra en una frase a partir de todos los lados (bidireccionalmente), no solo desde la palabra anterior o siguiente, lo que representa una mejora significativa respecto a modelos anteriores. Esto permite una eficaz extracción de relaciones semánticas, esenciales para la precisión en la identificación de amenazas. En un enfoque relacionado por L.Li [20], muestra cómo la terminología de MITRE ATT&CK se utiliza

[6]https://attack.mitre.org/
[7]SPARQL Protocol and RDF Query Language
[8]Bidirectional Encoder Representations from Transformers

junto con modelos de NLP para afinar la clasificación de tácticas y técnicas en informes de inteligencia de amenazas, mejorando así la precisión de la detección.

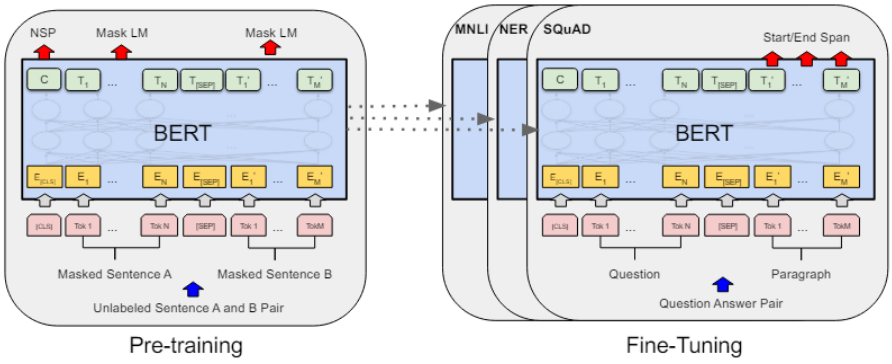

Figura 5: Arquitectura BERT

Investigaciones adicionales, como las citadas en "CSKG4APT" [33] y por W. Shang [35], destacan el uso de ontologías para capturar relaciones en contextos específicos, como IoT[9] e IoV[10]. Estos estudios demuestran que la combinación de ontologías y técnicas avanzadas de NLP mejora significativamente la identificación y clasificación de amenazas cibernéticas. Estas investigaciones subrayan la importancia de las ontologías para estructurar datos de amenazas, capturar intenciones de adversarios y refinar los modelos de clasificación a través de razonamientos avanzados.

2.1.3.2. MQ3: Considerando las técnicas de inteligencia artificial, ¿cuáles son los modelos más efectivos para el análisis y la identificación de amenazas en sistemas de seguridad informática?

El uso de técnicas de inteligencia artificial ha transformado significativamente el campo de la ciberseguridad, especialmente en el análisis y la identificación de amenazas. Modelos basados en redes neuronales profundas, como LSTM[11] [13]. Esta, es un tipo de red neuronal recurrente especializada en aprender dependencias a largo plazo, lo que la hace ideal para analizar secuencias de datos como texto o series de tiempo y Bi-LSTM[12] [6] mejora este concepto al procesar la información en dos direcciones, hacia adelante y hacia atrás, proporcionando un contexto más amplio y mejorando la captación de relaciones en los datos. Esta última junto con las redes de atención, han demostrado una eficacia particular. Un ejemplo destacado es el modelo SMIEN, mencionado en [35], que combina BERT con Bi-LSTM y capas de atención para analizar efectivamente la semántica y las dependencias temporales

[9]Internet of Things
[10]Internet of Vehicles
[11]Long short-term memory
[12]Bidirectional Long Short-Term Memory

en textos. Esta capacidad lo hace ideal para identificar relaciones complejas entre entidades de amenazas.

Por otro lado, modelos como BERT y sus variantes, utilizados en estudios como "KnowCTI" [42] y "TTPHunter" [32], facilitan la extracción y clasificación de entidades y relaciones en textos de inteligencia de amenazas. Estos modelos integran conocimientos específicos de ciberseguridad, mejorando significativamente tanto la comprensión como la precisión en la identificación de amenazas.

Además, los modelos de embedding, como el "Cyber-Phrase Embedding" citado en [7], permiten capturar relaciones semánticas a nivel de frase, lo cual sirve para identificar y clasificar patrones de ataque en textos de ciberseguridad. Asimismo, las redes neuronales gráficas y los grafos de conocimiento, discutidos en [33] y [44], son fundamentales en la representación y análisis de entidades y relaciones dentro de grandes conjuntos de datos de ciberseguridad, mejorando así la comprensión y la predicción de amenazas.

2.1.3.3. MQ4: ¿Cómo contribuyen las técnicas avanzadas de NLP, como el análisis semántico y la comprensión del lenguaje natural, al procesamiento y análisis de datos en ciberseguridad?

Las técnicas avanzadas de procesamiento de lenguaje natural (NLP) se han establecido como un pilar fundamental en el campo de la ciberseguridad, especialmente en el procesamiento y análisis de datos no estructurados. Estas herramientas permiten una comprensión profunda y facilitan la extracción de información clave para la identificación y mitigación de amenazas cibernéticas. Por ejemplo, el análisis semántico y los modelos de embedding capturan relaciones contextuales entre términos utilizados en descripciones de amenazas, como demuestra el modelo "Cyber-Phrase Embedding" [7]. Este enfoque mejora la interpretación de textos y alinea conceptos del sector con tácticas conocidas en bases como MITRE ATT&CK.

La extracción de entidades y relaciones mediante NLP sirve para identificar y clasificar componentes específicos de las amenazas, tales como actores, técnicas y procedimientos. La combinación de tecnologías como BERT, LSTM y CRF[13] [42] ha probado ser efectiva en el reconocimiento y categorización de elementos críticos a partir de grandes volúmenes de datos, esencial para la generación de alertas precisas y la implementación de estrategias de mitigación efectivas.

Además, el NLP facilita la clasificación y el mapeo estructurado de tácticas y técnicas a partir de textos de inteligencia sobre amenazas, ilustrado por el uso de DistilBERT en estudios como [20]. Esta capacidad para procesar información a nivel de frase y mapearla con precisión con la ontología de MITRE ATT&CK subraya cómo el NLP puede mejorar significativamente los sistemas de correlación y clasificación automática de amenazas.

[13] Conditional random fields

2.1.3.4. Resto de preguntas del mapping

En el ámbito de la ciberseguridad, tanto las técnicas de búsqueda elástica como la minería de datos son herramientas útiles para el manejo y análisis de grandes volúmenes de datos. No obstante, se debe señalar que el foco predominante de los estudios revisados ha estado en el uso de tecnologías de procesamiento de lenguaje natural (NLP) y la integración de ontologías para la identificación y clasificación de amenazas cibernéticas. Esto indica que, si bien las técnicas de búsqueda elástica y la minería de datos son importantes, podría haber sido provechoso orientar el mapeo de la investigación hacia enfoques que incorporaran de manera más explícita estas tecnologías, especialmente dadas sus aplicaciones potenciales en la inteligencia sobre amenazas.

2.1.3.5. MQ2: ¿Cuáles son las ventajas y limitaciones de las técnicas de búsqueda elástica cuando se aplican a la inteligencia de amenazas en ciberseguridad?

Las técnicas de búsqueda elástica, ampliamente empleadas en la recuperación de información, permiten localizar y acceder rápidamente a datos específicos dentro de extensos repositorios de información de ciberseguridad. Esta habilidad resulta esencial cuando la información está debidamente normalizada y etiquetada, facilitando así que los sistemas de seguridad respondan de manera eficaz a incidentes cibernéticos. Sin embargo, la efectividad de estas técnicas depende en gran medida de la estructura y etiquetado de los datos, lo cual puede constituir una limitación si la información no está correctamente preparada o actualizada.

2.1.3.6. MQ5: ¿Qué papel juega la minería de datos en la identificación de patrones ocultos en grandes conjuntos de datos de ciberseguridad y cómo se puede mejorar su aplicación?

El objetivo principal de estas técnicas es revelar correlaciones y tendencias que no son inmediatamente perceptibles. La integración de la minería de datos con modelos avanzados de procesamiento de lenguaje natural (NLP) y algoritmos de aprendizaje automático, que toman en cuenta el contexto y la complejidad de los datos de ciberseguridad, puede mejorar significativamente su rendimiento. No obstante, para alcanzar su máxima efectividad, es importante aplicar la minería de datos en conjuntos de datos bien estructurados y enriquecidos con metadatos precisos. Esto significa que la preparación y el preprocesamiento de los datos son etapas críticas, cuya inadecuada gestión puede restringir la aplicabilidad y efectividad de la minería de datos.

2.2. Conclusiones del mapping sistemático

La integración de ontologías en los sistemas de procesamiento de lenguaje natural (NLP) no solo mejora la precisión en la identificación de amenazas cibernéticas, sino que también proporciona una base sólida para un análisis y comprensión profundos de la información de seguridad. Este enfoque mejora la rapidez y precisión en la identificación de amenazas, lo que, a su vez, permite una respuesta más efectiva ante incidentes de ciberseguridad.

Además, la combinación de modelos avanzados de inteligencia artificial especificos en NLP son esenciales para el procesamiento y análisis de datos en ciberseguridad, mejorando la capacidad de los sistemas para realizar análisis semántico, extraer entidades y relaciones relevantes, y clasificar y mapear tácticas y técnicas. El NLP facilita una respuesta más rápida y precisa a las amenazas cibernéticas, fortaleciendo el desarrollo de defensas más robustas y adaptativas en un entorno de ciberseguridad cada vez más complejo y dinámico.

Aunque la búsqueda elástica y la minería de datos son herramientas potencialmente útiles en la ciberseguridad, su efectividad depende de una cuidadosa consideración de la estructura de los datos y su integración con tecnologías que puedan interpretar y analizar el contenido semántico y contextual.

Durante el desarrollo de este mapping, se han utilizado herramientas como Parsifal para el mapeo sistemático y la organización de la literatura, Scholarcy[14] para la síntesis y el resumen de información relevante de los artículos, y ChatGPT-4[15] para una síntesis efectiva y la formulación de preguntas pertinentes. Adicionalmente, el uso de Obsidian[16] para la toma de notas ha facilitado la organización eficiente de la información y las ideas clave durante el proceso de investigación, mejorando el acceso y la gestión del conocimiento adquirido. Estas herramientas tecnológicas han demostrado ser extremadamente útiles en la conducción de investigaciones complejas, permitiendo un análisis más riguroso y sistemático de grandes volúmenes de datos

3. Ontologías: Documentos que definen el conocimiento

Una ontología, según [24], es un documento estructurado que se compone de tres elementos que interactúan entre sí para proporcionar una comprensión profunda y una manipulación efectiva del conocimiento dentro de un dominio específico. Los tres elementos que interactúan entre sí son los siguientes:

- **Vocabulario del dominio:** El primer componente de una ontología es un vocabulario seleccionado que describe el dominio de interés. Este vocabulario

[14]https://www.scholarcy.com/
[15]https://chatgpt.com/
[16]https://obsidian.md/

incluye términos y conceptos clave que sirven para capturar la esencia del dominio. El propósito de este vocabulario es ofrecer una terminología estándar que pueda ser utilizada de manera consistente, no solo en discusiones teóricas sino también en implementaciones prácticas. Esto asegura que todos los usuarios de la ontología tengan un entendimiento común y claro de los términos utilizados, lo cual es útil para la eficacia comunicativa y operativa.

- **Anotaciones del vocabulario:** El segundo elemento son las anotaciones, que sirven para documentar el vocabulario proporcionando clarificaciones y contextos adicionales. Estas anotaciones explican el significado y el uso de los términos dentro del vocabulario y pueden incluir definiciones, ejemplos, y notas explicativas. Las anotaciones son vitales porque enriquecen el vocabulario con información que puede no ser inmediatamente evidente a través de los términos por sí solos, ayudando así a los usuarios a aplicar la ontología de manera más efectiva y precisa.

- **Teoría lógica:** El tercer componente de una ontología es una teoría lógica, que consiste en axiomas y definiciones que establecen cómo los términos del vocabulario se relacionan entre sí y cómo deben ser interpretados dentro del marco del dominio. Esta teoría lógica no solo define las reglas formales para el manejo de la información dentro del dominio sino que también soporta el razonamiento automatizado y las inferencias basadas en los datos representados por la ontología.

La utilidad de las ontologías abarca una amplia gama de aplicaciones en diversos campos, proporcionando una base estructurada para la organización, recuperación, y análisis del conocimiento, como puede ser en áreas como la medicina, la creación de la "Gene ontology: Tool for the unification of biology" [4] es un ejemplo prominente que facilita la búsqueda y el análisis de información genética, ayudando a los investigadores a entender las funciones de los genes y sus interacciones en diferentes organismos. En la gestión empresarial se estudia como el desarrollo de una ontología específica en una compañía su integración en los sistemas de información mejora la comunicación entre departamentos, y permite análisis más sofisticados de operaciones y estrategias de mercado [43].

Otro área como son las ontologías en la web semántica como muy bien explican en [10], permiten a las máquinas "entender" el contenido de las páginas web de manera que puedan ofrecer a los usuarios resultados más relevantes y personalizados. En inteligencia artificial, las ontologías permiten la representación de conocimiento en sistemas que necesitan realizar razonamientos complejos, como en los asistentes virtuales, que pueden usar ontologías para interpretar mejor las preguntas de los usuarios y proporcionar respuestas más precisas.

Los asistentes como Siri, Alexa, o Google Assistant utilizan ontologías para desglosar y analizar las preguntas de los usuarios. Esto no solo implica captar las palabras claves, sino también entender las relaciones entre los conceptos mencionados y cómo estos se relacionan con las necesidades o comandos del usuario.

Por ejemplo, si un usuario pregunta: "¿Cuál es el pronóstico del tiempo para mañana?", el asistente utiliza la ontología para interpretar que "pronóstico del tiempo"se relaciona con información meteorológica y que "mañanaïndica una consulta sobre un momento específico en el tiempo. La ontología ayuda al asistente a conectar estos conceptos con datos relevantes para proporcionar una respuesta precisa.

3.1. Ontologías en ciberseguridad

En el dinámico ámbito de la ciberseguridad, las ontologías emergen como herramientas esenciales para estructurar y clarificar la complejidad inherente a la seguridad de la información. Esta subsección se enfoca en el papel crítico que desempeñan las ontologías dentro de la ciberseguridad, destacando cómo contribuyen a la identificación, análisis y mitigación de amenazas cibernéticas. A través de la formalización de vocabularios y la estructuración de conocimientos, las ontologías permiten a las organizaciones mejorar sus capacidades de respuesta y comprensión ante incidentes de seguridad, facilitando así la implementación de estrategias de defensa más efectivas y adaptativas. Como se ha mencionado en el mapping sistemático de la literatura (ver sección 2.1).

Una de las más destacadas del campo es la Ontología Unificada de Ciberseguridad (UCO) [38] se creó con el objetivo de mejorar la integración de la información y la conciencia situacional en ciberseguridad. Esta ontología ha sido diseñada para respaldar la integración de datos heterogéneos y esquemas de conocimiento de diferentes sistemas y estándares de ciberseguridad comúnmente utilizados para el intercambio y la compartición de información.

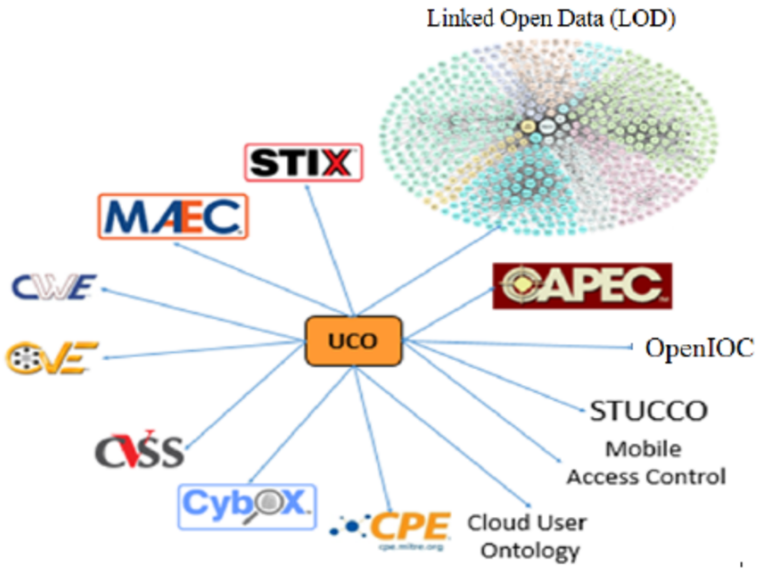

Figura 6: Ontología UCO

La principal finalidad de UCO es facilitar un marco común que permita la compartición, integración y reutilización de información a través de diferentes aplica-

ciones y sistemas, lo que resulta esencial para manejar amenazas y vulnerabilidades en un entorno de seguridad cibernética cada vez más complejo y dinámico. UCO sirve como una ontología central que conecta diversas fuentes de datos y ontologías existentes en ciberseguridad, funcionando de manera similar a cómo DBpedia [19] actúa. Como el núcleo para el conocimiento general en la nube de datos abiertos vinculados.

Estas capacidades reflejan la importancia crítica de las ontologías para estructurar los datos de amenazas, capturar intenciones de adversarios y mejorar los modelos de clasificación a través de razonamientos avanzados, asegurando así que las infraestructuras de ciberseguridad puedan mantenerse al día con el panorama de amenazas rápidamente cambiante.

A día de hoy se está desarrollando D3FEND[16], que se presenta como un marco de trabajo para codificar una base de conocimiento sobre contramedidas, pero más específicamente, un grafo de conocimiento. Este grafo incluye tipos y relaciones semánticamente rigurosas que definen los conceptos clave en el dominio de las contramedidas de ciberseguridad y las relaciones necesarias para vincular estos conceptos entre sí. Cada concepto y relación se basa en referencias específicas de la literatura de ciberseguridad.

A través del análisis de una muestra dirigida de más de 500 patentes de contramedidas extraídas del corpus de la Oficina de Patentes de EE. UU., los autores demuestran cómo el grafo puede apoyar consultas que mapean inferencialmente las contramedidas de ciberseguridad a las técnicas, tácticas y procedimientos ofensivos. Además, se esboza el trabajo futuro para el proyecto D3FEND, que incluye la utilización de datos abiertos y la aplicación de métodos de aprendizaje automático, particularmente métodos semi-supervisados, para mantener el grafo de conocimiento a lo largo del tiempo.

3.2. Ingeniería de la Ontología

En el desarrollo de la ontología centrada en la ciberseguridad, se ha adoptado la metodología Linked Open Terms (LOT) [26], la cual se caracteriza por su orientación hacia aplicaciones industriales y tecnologías del web semántico. Esta metodología se destaca por su capacidad para integrar y armonizar las prácticas de desarrollo ágil de software con las exigencias de proyectos de investigación y desarrollo académico. El enfoque de LOT facilita una estructuración flexible y adaptable del proceso de creación ontológica, permitiendo iteraciones y revisiones continuas que son esenciales en el ámbito dinámico de la ciberseguridad.

La metodología LOT se fundamenta en principios de desarrollo ligero y abierto, lo que permite su aplicación tanto en entornos académicos como industriales. Se basa en la experiencia acumulada a lo largo de más de dos décadas de ingeniería ontológica y ha sido refinada a través de su aplicación en dieciocho proyectos distintos. Estos proyectos no solo han implicado a equipos de autores de este documento, sino también a colaboraciones externas que abarcan expertos en dominios especí-

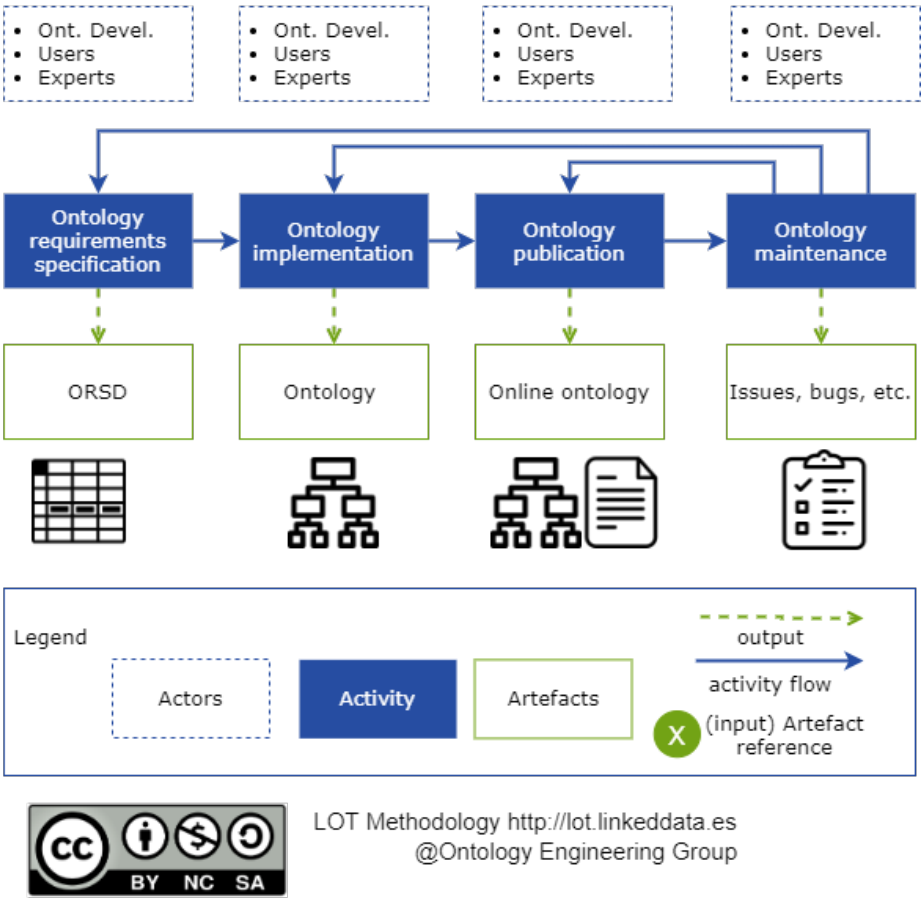

Figura 7: LOT workflow

ficos y equipos de ingenieros de software, lo que ha enriquecido y diversificado las perspectivas y aproximaciones metodológicas.

Un aspecto distintivo de la metodología LOT es su énfasis en la alineación con las prácticas de desarrollo de software. Esto se manifiesta en la adopción de sprints y ciclos iterativos de desarrollo que son característicos de los métodos ágiles. Tal estructura promueve una integración eficaz y eficiente entre el desarrollo ontológico y los ciclos de desarrollo de software, asegurando que la ontología no solo se mantenga relevante frente a los avances tecnológicos, sino que también responda de manera efectiva a las necesidades emergentes en ciberseguridad.

Además, LOT pone un fuerte énfasis en la reutilización de términos y vocabularios ya publicados, favoreciendo así la interoperabilidad y la estandarización. Esta práctica no solo reduce el tiempo de desarrollo al evitar la duplicidad de esfuerzos, sino que también mejora la calidad y la coherencia de la ontología resultante. Al mismo tiempo, la metodología facilita la publicación de las ontologías desarrolladas bajo principios de datos vinculados abiertos, lo que promueve una mayor accesibilidad y reusabilidad por parte de la comunidad global, un aspecto importante en el dominio de la ciberseguridad donde la colaboración y el intercambio de información son clave para el avance y la innovación. Esta afirmación respalda el uso de la terminología bajo el marco MITRE ATT&CK, como se han mostrado en estudios anteriormente,

que puede ser un buen estándar aplicado a la hora de crear una ontología.

3.2.1. Formulación de preguntas de competencia

La especificación de requisitos ontológicos constituye una fase crítica en el desarrollo de la ontología centrada en ciberseguridad, ya que establece las bases para una modelización precisa de los conceptos relevantes en este dominio. Esta etapa comienza con la definición de requisitos claros y detallados que guiarán todas las fases subsecuentes del desarrollo ontológico.

Las preguntas de competencia (CQs) son formuladas para capturar y reflejar las necesidades informativas específicas que la ontología debe satisfacer. En el contexto de la ciberseguridad, estas preguntas son esenciales para entender no solo las características de los ataques, sino también para establecer relaciones causales y procedimentales entre diferentes entidades y eventos. Se muestran las preguntas de competencia formuladas:

1. **¿Qué tipo de herramientas han empleado los atacantes?**
2. **¿Qué tipo de malware ha sido distribuido en el ataque?**
3. **¿Qué vulnerabilidades se han explotado?**
4. **¿Qué TTPs[17] han sido efectuadas por el atacante?**
5. **¿Quién o quiénes son el objetivo del ataque?**
6. **¿Desde dónde se ha efectuado el ataque?**
7. **¿Qué ficheros están implicados en el ataque?**

Las preguntas se han formulado en base a la lectura de una serie de reportes de ciberseguridad de repositorios de empresas expertas en el sector. Cada reporte describe un escenario específico donde la ontología puede ser aplicada para mejorar la comprensión y el manejo de ataques cibernéticos. Por ejemplo, un caso de uso podría involucrar el análisis de un ataque de ransomware a una infraestructura crítica, detallando cómo los atacantes ganaron acceso, qué vulnerabilidades fueron explotadas, y qué herramientas específicas fueron utilizadas.

El proceso de especificación de requisitos concluye con la formalización de un Documento de Especificación de Requisitos Ontológicos (ORSD) (ver A), que detalla todos los requisitos recopilados, las preguntas de competencia, y los casos de uso. Este documento es esencial para las fases de implementación y validación de la ontología, asegurando que todos los elementos de la ontología se desarrollen de manera coherente y orientada a objetivos claros.

[17] Técnicas, Tácticas y Procedimientos

3.2.2. Implementación de la Ontología

La implementación de la ontología de ciberseguridad bajo la metodología LOT se estructuró para incorporar la terminología general del marco MITRE ATT&CK como términos específicos del dominio de la ciberseguridad. Este proceso se dividió en varias fases: conceptualización y codificación.

3.2.2.1. MITRE ATT&CK

MITRE ATT&CK[37] es una base de conocimientos de acceso global que compila tácticas y técnicas de adversarios basadas en observaciones del mundo real. La base de conocimientos ATT&CK sirve como fundamento para el desarrollo de modelos de amenazas específicos y metodologías tanto en el sector privado como en el gubernamental, y en la comunidad de productos y servicios de ciberseguridad. ATT&CK proporciona una taxonomía común para la ofensa y la defensa, y se ha convertido en una herramienta conceptual útil en múltiples disciplinas de ciberseguridad para transmitir inteligencia sobre amenazas, realizar pruebas a través de equipos rojos o emulación de adversarios, y mejorar las defensas de redes y sistemas contra intrusiones. El proceso utilizado por MITRE para crear ATT&CK y la filosofía que se ha desarrollado para curar nuevo contenido son aspectos críticos de este trabajo, y resultan útiles para otros esfuerzos que buscan crear modelos de adversarios e información repositorios similares.

MITRE | ATT&CK®

Figura 8: MITRE ATT&CK

3.2.2.2. Conceptualización

La fase de conceptualización implicó la creación de un modelo conceptual que sirve como esqueleto para la ontología. Dado que la terminología y las clasificaciones del marco MITRE ATT&CK forman la base del vocabulario de la ontología, se detalló mapear estos términos a clases y relaciones ontológicas de manera que reflejen fielmente las TTPs observados en el mundo real. Se emplearon diagramas y modelos UML, creados mediante la herramienta drawio[18] para visualizar las relaciones entre las entidades y para asegurar que todos los aspectos del conocimiento de ciberseguridad estuvieran representados adecuadamente.

Durante esta fase, se identificaron las principales clases/entidades como:

- **"IP"**
- **"Técnica"**

[18] https://app.diagrams.net/

- **"Táctica"**
- **"Herramienta"**
- **"Vulnerabilidad"**
- **"APT**[19]**"**
- **"Operación"**
- **"geoPunto"**
- **"Malware"**
- **"Hash"**
- **"Servidor"** (Incluyendo Servidor Adversario y Servidor Víctima)

Cada entidad fue definida con atributos específicos que permiten una descripción detallada de las instancias correspondientes. Por ejemplo, la clase Técnica incluye atributos para la descripción, su nombre y su id propio. Se procede a explicar los dominios y entidades conceptualizadas con más detalle en las siguientes secciones.

3.2.2.3. Dominios

La estructura de los dominios y sus interconexiones sirven para modelar de manera efectiva las relaciones y características de las entidades implicadas en los ciberataques. La integración de múltiples dominios en una única ontología es una práctica común en contextos donde los fenómenos a modelar son complejos y multidimensionales.

El dominio de **"Server"** describe las entidades que son servidores en la red, que pueden ser utilizados tanto por los adversarios como por las víctimas. Estos servidores son representados con atributos como dirección IP, proveedor de servicios, y llevan asociados una entidad que los geolocaliza. Este dominio sirve para identificar la infraestructura de red utilizada en los ataques y defensas.

El dominio asociado es **"Geo"**, se utiliza para describir la ubicación física o virtual de servidores, infraestructuras y actores involucrados en el ciberespacio. Incorporar el dominio de geolocalización permite a los analistas entender y correlacionar las actividades cibernéticas con ubicaciones geográficas específicas, lo que permite la identificación de patrones de ataque, la atribución de ataques a actores específicos, y la comprensión de la dinámica geopolítica asociada a la ciberseguridad.

Conexos al dominio de "Server", se han definido los dominios **"Adversary"** y **"Victim"**:

[19] Advanced Persistent Thread

- **Adversary**: los adversarios utilizan servidores específicos para lanzar ataques. Por esto, la entidad .Adversary Server.en la ontología comparte atributos con la entidad genérica "Server", como la dirección IP y detalles del servidor, permiten entender desde dónde y cómo se están llevando a cabo los ataques.
- **Victim**: Similarmente, la entidad "Victim Serverrefleja la parte de infraestructura que ha sido atacada o está en riesgo de ser atacada, utilizando también atributos de "Server". Esto permite modelar y analizar la infraestructura de la víctima y determinar posibles vulnerabilidades o el impacto de un ataque.

Además, **"Adversary"** cuenta con otras entidades que en conjunto a la entidad "Adversary Server" permiten la identificación única del grupo de cibercriminales.

El dominio **"Capabilities"** incluye las capacidades ofensivas empleadas durante el ciberataque.

El dominio **"Security"** se centra en aspectos de seguridad que permiten verificar la integridad de los datos.

El último dominio es **"Strategy"**, que cuenta con dos entidades que permiten modelar el comportamiento efectuado durante el ciberataque.

3.2.2.4. Entidades

En el diseño de ontologías, una entidad se refiere a un objeto o concepto que tiene existencia distinguible dentro de un dominio de estudio específico. En el campo de la ciberseguridad, por ejemplo, entidades como "APT", "Servidor" o "Herramienta" representan componentes concretos o abstractos claves para entender y analizar la naturaleza y dinámicas de los ataques cibernéticos.

Las entidades deben ser definidas de manera clara y precisa para asegurar que el modelo sea tanto comprensible como útil para los usuarios y sistemas que dependen de él para el análisis y la toma de decisiones.

Los atributos son las características específicas que describen o califican a una entidad. En la práctica, los atributos proporcionan la información detallada necesaria para caracterizar una entidad en su contexto y permiten diferenciar entre instancias de la misma entidad basándose en esos detalles.

Se procede a profundizar en cada entidad con sus atributos aportando información del porqué de estas entidades:

APT (Advanced Persistent Thread): La primera entidad mostrada forma parte del dominio "Adversary", un dominio centrado en los atacantes, permite identificarlos y posicionarlos como entidades únicas en el conjunto numeroso de grupos de cibercriminales. En este dominio también se incluyen las operaciones llevadas a cabo por ellos, que son las campañas de ciberataques que efectúan a objetivos concretos para lucrarse. Por último se incluye aquella infraestructura usada para

llevar a cabo el ciberataque, englobando en un dominio todas las características que permiten identificar a un grupo.

Esta entidad central representa a un actor de amenazas, ya sea individual o grupal. Los atributos incluyen motivación, nombre, origen. La motivación es crítica para entender el 'por qué' detrás de un ataque.

Cuenta con tres relaciones formales definidas:

1. Relación de pertenencia plural: Con otra entidad del dominio (Operación).
2. Relación de mantenimiento plural: Con la entidad "adversary server" que forma parte de su dominio.
3. Relación de ataque plural: Con la entidad "victim server" que forma parte del dominio "Victim".
4. Relación de aplicación de estrategias: Conecta las estrategias realizadas por los atacantes.

Operación: La entidad representa una serie de acciones coordinadas o una campaña emprendida por un adversario para alcanzar un objetivo específico. Cuenta con atributos como el nombre de la operación, el objetivo del ataque (robo de datos, destruir la infraestructura, etc), el nombre del sector al que va dirigido el ataque y el nombre del objetivo afectado.

Servidor Adversario y Servidor Víctima: Estas entidades facilitan una representación clara de los nodos de ataque y los atacados. Los atributos incluyen detalles como si la url cuenta con un servicio web, el proveedor de servicios, el sistema operativo del servidor, el dominio asociado, el tipo de servidor si es en "cloud" o "self-hosted", y si tiene un nombre identificativo, proporcionando así un contexto detallado sobre la infraestructura implicada.

Un servidor cuenta con dos relaciones:

1. Relación entidad dirección IP: Permite asociar las posibles direcciones IP que puede tomar a lo largo del tiempo un servidor.
2. Relación con la entidad Malware: Permite relacionar los repositorios públicos que distribuyen el malware en los ciberataques.

IP y Point: En cuanto a la entidad IP, esta se centra en la representación de las direcciones IP. Con atributos como su valor de dirección, sus puertos habilitados al exterior y el tipo que es (IPv4 o IPv6).

Toda dirección IP lleva asociada una geolocalización, por eso existe la relación la entidad "Point" del dominio geolocalización. Con atributos de representacion de coordenadas y región.

Malware: Se concentra en el software malicioso utilizado en ataques. Los atributos como type y name. Valores como su comportamiento definidos por el primer atributo y el nombre de la familia de malware a la que pertenecen permiten la identificación y el análisis del malware.

Esta entidad posee dos relaciones:

1. Relación explotar vulnerabilidades: Cada malware tiene asociadas unas vulnerabilidades que explota.

2. Relación hash: Proporciona un hash de seguridad para verificar la integridad del malware.

Vulnerability: Esta entidad lleva presente desde los inicios en la ciberseguridad, ya que identifica las debilidades que pueden ser explotadas por los adversarios a través del malware y las herramientas. Atributos como severity, cve_id, y desc (descripción), ofrecen información detallada sobre la gravedad, identificación y detalles de cada vulnerabilidad. El id está definido por la CVE[20]

La misión del Programa CVE es identificar, definir y catalogar las vulnerabilidades de ciberseguridad divulgadas públicamente. Hay un registro CVE por cada vulnerabilidad del catálogo. Las vulnerabilidades son descubiertas, asignadas y publicadas por organizaciones de todo el mundo que se han asociado al Programa CVE. Los socios publican Registros[21] CVE para comunicar descripciones coherentes de las vulnerabilidades. Los profesionales de las tecnologías de la información y la ciberseguridad utilizan los registros CVE para asegurarse de que están hablando del mismo tema y para coordinar sus esfuerzos para priorizar y abordar las vulnerabilidades.

Herramientas: Refleja las herramientas técnicas utilizadas por atacantes. Los atributos como *type* y *name* permiten una descripción detallada de la herramienta, incluyendo su funcionalidad y propósito. La relación con la entidad "Malware", proporciona un enlace directo entre las herramientas usadas para desplegar el malware y el malware mismo, lo que es vital para comprender los vectores de ataque.

Técnicas: Se refiere a los métodos concretos empleados para lograr los objetivos detallados en las tácticas. Las técnicas son las acciones reales realizadas por los adversarios, como el uso de un exploit específico, phishing, o técnicas de evasión. Las técnicas también utilizan identificadores estándar bajo el marco MITRE ATT&CK y nombres descriptivos que ayudan en la clasificación y el análisis.

La única relación con la que cuenta esta entidad es con las Herramientas, conecta las técnicas específicas con las herramientas utilizadas para llevarlas a cabo.

[20]Common Vulnerabilities and Exposures
[21]https://www.cve.org/

Tácticas: Representa el nivel más alto de planificación estratégica en ciberseguridad. Cada táctica encapsula un grupo de técnicas relacionadas que apuntan hacia un objetivo común, permitiendo a los adversarios coordinar sus esfuerzos en una dirección estratégica coherente. Las técnicas pueden vincularse a un identificador en el marco de MITRE ATT&CK, proporcionando una referencia estándar para identificar y comunicar sobre acciones específicas dentro de la comunidad de ciberseguridad.

Destacar las tres relaciones con las que cuenta esta entidad:

1. Relación con las técnicas: Como se ha explicado en la descripción de la entidad, una táctica engloba un listado de técnicas.

2. Relación con las vulnerabilidades: Una táctica estructura la explotación de vulnerabilidades en función de la etapa del ciberataque.

3.2.2.5. Codificación

La codificación de la ontología se realizó utilizando el lenguaje OWL[22], que permite una representación rica y compleja de las relaciones entre las entidades. El uso de OWL facilitó la implementación de restricciones y reglas lógicas que definen el comportamiento de las relaciones entre las diferentes técnicas, tácticas y herramientas descritas en MITRE ATT&CK.

Además, se desarrollaron reglas SWRL[23] para inferir nuevas relaciones y propiedades a partir de los datos existentes. Finalmente se muestra el diagrama de la ontología definida y creada bajo el marco LOT:

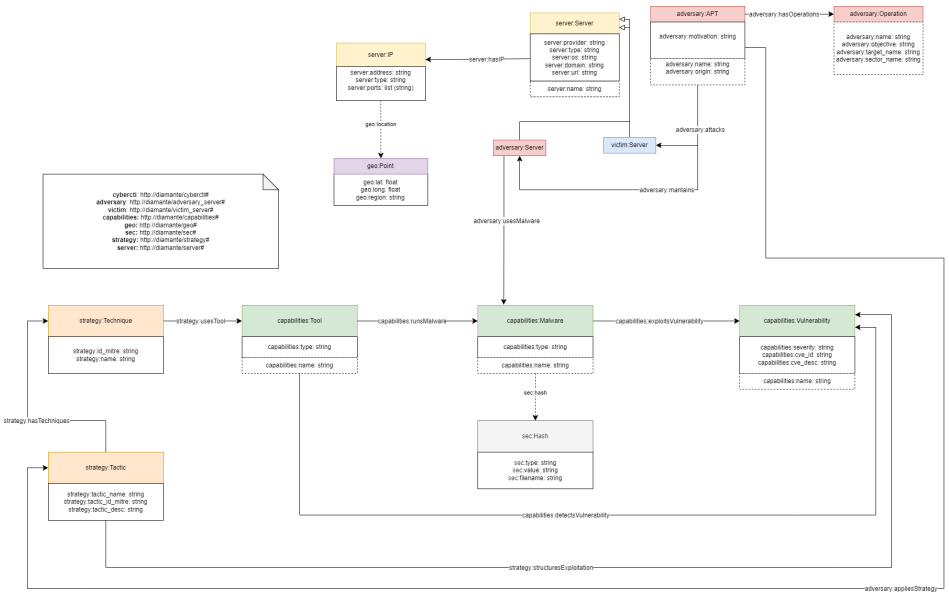

Figura 9: Ontologia ciberseguridad

[22]Web Ontology Language
[23]Semantic Web Rule Language

3.2.2.6. Revisión continua

La implementación de esta ontología en el dominio de la ciberseguridad, guiada por la metodología LOT, no solo destaca la necesidad de una representación estructurada de conocimientos complejos y dinámicos, sino que también subraya la importancia de la revisión y actualización continua.

El marco LOT[26] sugiere que cualquier ontología debe estar abierta a modificaciones continuas para reflejar los cambios en el dominio que representa. Esto es muy importante en la ciberseguridad, donde las nuevas amenazas, técnicas y tecnologías emergen a un ritmo acelerado. Por lo tanto, se recomienda establecer un protocolo de revisión regular, donde los expertos en ciberseguridad revisen y actualicen la ontología periódicamente. Este proceso no solo debe enfocarse en añadir nuevos elementos, sino también en reevaluar y posiblemente rediseñar partes existentes de la ontología para mejorar la precisión y la relevancia. La capacidad de cambio modular de la ontología es un principio en el marco LOT, permite que estas actualizaciones sean manejables y menos disruptivas, asegurando que la ontología pueda evolucionar sin necesidad de reestructuraciones masivas que podrían desestabilizar los sistemas existentes.

4. Reportes de ciberseguridad

Los reportes de ciberseguridad constituyen información relevante en el ámbito de la seguridad informática. Estos documentos son elaborados para proporcionar una visión detallada y comprensiva de incidentes específicos de ciberataques. Los reportes pueden variar en formato y detalle, pero típicamente incluyen información sobre las tácticas, técnicas y procedimientos (TTPs) utilizados por los atacantes, así como detalles sobre las vulnerabilidades explotadas y los tipos de daños ocasionados.

4.1. Características de los reportes

Un reporte de ciberseguridad se distingue por incorporar datos no estructurados que revelan aspectos concretos y técnicos de un incidente de ciberseguridad. Estos datos suelen comprender:

- **Descripción del ataque:** Detalle narrativo del incidente, incluyendo el vector de ataque inicial, la progresión del ataque dentro de la red afectada y las técnicas específicas empleadas por los atacantes.
- **Identificación del APT:** Análisis del grupo de amenazas persistente avanzado (APT) responsable del ataque, incluyendo su posible origen geográfico y sus motivaciones.
- **Impacto en las Víctimas:** Evaluación del impacto operativo y financiero sobre las entidades afectadas. Esto puede incluir la cantidad y tipo de datos

comprometidos, así como una descripción de las interrupciones causadas en las operaciones normales de la víctima.

- **Indicadores de Compromiso (IoCs):** Lista de artefactos como direcciones IP, URLs, hashes de archivos y otros identificadores que pueden ser utilizados para detectar actividades maliciosas relacionadas con el incidente.

- **Recomendaciones y Mitigaciones:** Estrategias y medidas recomendadas para prevenir futuros incidentes o para mitigar el impacto de un ataque en curso.

Los reportes de ciberseguridad no solo sirven para documentar un incidente específico, sino que también cumplen una función educativa y preventiva dentro de la comunidad de ciberseguridad. Permiten a las organizaciones entender mejor las amenazas emergentes y adaptar sus estrategias de defensa en consecuencia. Además, estos reportes fomentan una cultura de transparencia y colaboración entre las entidades afectadas y la comunidad de seguridad en general.

4.2. Fuentes del sector

Sitios web como The Hacker News[24], Securelist[25], Talos Intelligence Blog[26] y The DFIR Report[27] son ejemplos destacados de plataformas que regularmente publican reportes de ciberseguridad. Estas plataformas son reconocidas por su profundidad analítica y su capacidad para proporcionar información oportuna y relevante sobre las últimas amenazas de ciberseguridad.

Cada uno de estos sitios tiene un enfoque particular:

- **The Hacker News** proporciona una amplia cobertura de las últimas vulnerabilidades, hacks y brechas de seguridad.

- **Securelist**, asociado con Kaspersky, ofrece análisis detallados sobre malware y campañas de espionaje cibernético.

- **Talos Intelligence Blog**, de Cisco, se especializa en análisis de amenazas y la divulgación de vulnerabilidades.

- **The DFIR Report** se centra en reportes de análisis forense digital y respuestas a incidentes, proporcionando detalles técnicos sobre las investigaciones de ciberataques.

[24]https://thehackernews.com/
[25]https://securelist.com/
[26]https://blog.talosintelligence.com/
[27]https://thedfirreport.com/

4.3. Datos no estructurados

Los datos no estructurados son aquellos que no siguen un modelo o formato predefinido, lo que dificulta su análisis y procesamiento utilizando métodos tradicionales de bases de datos o análisis estructurado. Estos datos pueden incluir texto, imágenes, audio y video, y son predominantemente generados por fuentes como documentos, correos electrónicos, redes sociales, blogs, transcripciones de llamadas y más. A diferencia de los datos estructurados, que se almacenan en campos específicos dentro de una base de datos como tablas con filas y columnas, los datos no estructurados son almacenados en formatos más libres y flexibles.

4.3.1. Extracción de datos

El procesamiento de datos no estructurados, especialmente provenientes de fuentes como documentos HTML y PDF, requiere una serie de pasos para transformarlos en un formato más estructurado y analizable. A continuación se detallan las etapas llevadas a cabo en el procesamiento de estos datos:

- **HTML**: La extracción de datos de documentos HTML usualmente se realiza a través de técnicas de web scraping. Herramientas y bibliotecas como Beautiful Soup o Scrapy en Python son utilizadas para parsear el HTML y extraer la información relevante, como textos, enlaces y otros datos contenidos en etiquetas específicas.
- **PDF**: Los archivos PDF pueden contener tanto texto como imágenes. La extracción de texto se puede realizar utilizando herramientas como PDFMiner o PyPDF2 en Python, que permiten acceder al contenido textual de los PDFs. En el caso de que el PDF contenga imágenes de texto, se requiere un paso adicional de OCR[28] para extraer el texto de estas imágenes.

En el caso de este proyecto, se emplea el lenguaje de programación Python, donde esta función se realiza mediante la biblioteca Beautiful Soup[29]

4.3.2. Limpieza de datos

Una vez extraídos, los datos suelen requerir una limpieza para eliminar incoherencias, errores o información irrelevante. Este proceso puede incluir:

- Eliminación de etiquetas HTML.
- Corrección de errores de codificación o formatos.

[28] Optical Character Recognition
[29] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

- Normalización de textos, como la conversión a minúsculas, eliminación de puntuación y eliminación de palabras de parada (stop words).
- Detección y corrección de errores tipográficos o gramaticales.

Este proceso se ha llevado a cabo mediante la biblioteca **html2text**[30] que es un script de Python que convierte una página HTML en texto estructurado Markdown equivalente.

El **Markdown** es un lenguaje de marcado ligero, que facilita la escritura de texto en la web. Se diseñó para ser lo más legible posible en su forma sin procesar, lo que significa que el texto formateado con Markdown puede leerse sin distractores visuales típicos de los lenguajes de marcado más complejos como HTML. La sintaxis de Markdown incluye elementos sencillos para dar formato al texto, como asteriscos para el énfasis (cursiva o negrita), guiones o asteriscos para listas, almohadillas para títulos, y corchetes junto con paréntesis para enlaces. Este lenguaje es ampliamente utilizado en plataformas de documentación, sistemas de gestión de contenido, y más recientemente, en aplicaciones de notas y foros en línea, debido a su simplicidad y facilidad de conversión a HTML. La sencillez y la legibilidad del Markdown permiten trabajar con frases estructuradas para el proyecto planteado.

4.4. Procesamiento de imágenes

El Reconocimiento Óptico de Caracteres (OCR) es una tecnología que permite convertir diferentes tipos de documentos impresos o escritos a mano en datos codificados por máquina y editables. Su utilidad se extiende a múltiples aplicaciones, desde la digitalización de documentos históricos hasta la automatización de entrada de datos y la accesibilidad, permitiendo a las máquinas leer y procesar textos humanos. En el caso del proyecto, se emplea para extraer texto de las imágenes y añadirlo junto al texto relacionado con la imagen en la sección del Markdown que le corresponda.

4.4.1. Tesseract

Tesseract OCR[36] se distingue como uno de los motores de OCR de código abierto más poderosos y versátiles en el mundo de la tecnología. Desarrollado originalmente por HP Labs en 1984 y más tarde adoptado por Google. Tesseract ha evolucionado desde sus inicios, aunque inicialmente fue un proyecto dentro de un ambiente académico y luego industrial, su liberación como software de código abierto en 2005 marcó un antes y un después, permitiendo a los desarrolladores de todo el mundo contribuir y utilizar esta tecnología en una variedad de aplicaciones.

Para emplear la tecnología de OCR en Python, se utiliza la biblioteca Python-tesseract, también conocida como **pytesseract**. Esta biblioteca actúa como un en-

[30]https://pypi.org/project/html2text/

voltorio (wrapper) para el motor Tesseract-OCR de Google, permitiendo a los desarrolladores integrar de manera sencilla la capacidad de reconocimiento óptico de caracteres en sus aplicaciones Python. A continuación, se muestra una imagen con su respectivo código de ejemplo de uso de pytesseract:

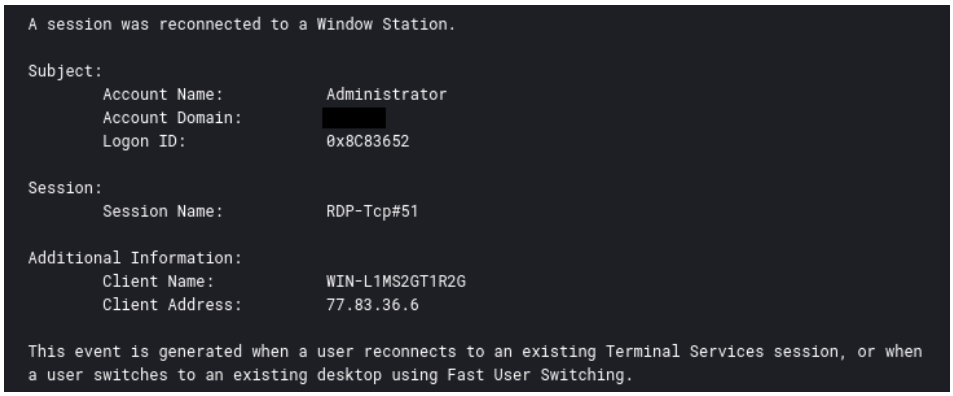

Figura 10: Pytesseract ejemplo

```python
from PIL import Image
import pytesseract
pytesseract.pytesseract.tesseract_cmd = r'F:\tesseract-ocr\
    ↪ tesseract.exe'
img = Image.open(r"C:\Users\javi\Downloads\pytesseract.png")
img.load()
text = pytesseract.image_to_string(img)
print(text)
```

Listing 1: Python pytesseract

```
‘A session was reconnected to a Window Station.

Subject:
Account Name Administrator
Account Domain:
Logon ID: 0x8¢83652
Session:
Session Name RDP-Top#51

Additional Information:
Client Name WIN-L1MS26T1R26
Client Address: 77.83.36.6

This event is generated when a user reconnects to an existing Terminal Services session, or when
a user switches to an existing desktop using Fast User Switching.
```

5. Modelos de Procesamiento del Lenguaje Natural

Los modelos de procesamiento de lenguaje natural (NLP[31]) han evolucionado desde sus inicios, donde predominaban métodos estadísticos simples. Con el tiempo,

[31]Natural Language Processing

la adopción de modelos basados en redes neuronales ha permitido un tratamiento más sofisticado y efectivo del lenguaje natural. Estos modelos aprenden representaciones de palabras y frases en espacios vectoriales densos, capturando así el contexto y los matices semánticos de manera más efectiva que los enfoques anteriores.

Inicialmente, los modelos de lenguaje utilizaban arquitecturas recurrentes (RNN[32]) para procesar secuencias de texto, siendo Long Short-Term Memory (LSTM)[13] y Gated Recurrent Units (GRU) [5] las variantes más efectivas y populares. Estos modelos procesan texto secuencialmente, lo cual presenta desafíos en términos de eficiencia computacional y dificultad para captar dependencias a largo plazo debido a problemas como la desaparición del gradiente.

El verdadero cambio paradigmático llegó con el modelo Transformer [40], donde la "atención" se refiere a un mecanismo que permite a los modelos de Procesamiento de Lenguaje Natural ponderar diferentes partes de una secuencia de entrada de manera selectiva, mejorando la comprensión y generación del lenguaje. Este enfoque permite a los modelos centrarse en la información relevante y establecer relaciones contextuales complejas a lo largo de toda la secuencia de texto, sin depender de la secuencia en el procesamiento de datos.

5.1. Transformers

El Transformer se estructura en torno a dos componentes principales: el codificador y el decodificador. Cada uno consta de una serie de capas que, en el caso del codificador, procesan la entrada para generar un conjunto de representaciones vectoriales; el decodificador, luego, utiliza estas representaciones para generar la salida secuencialmente.

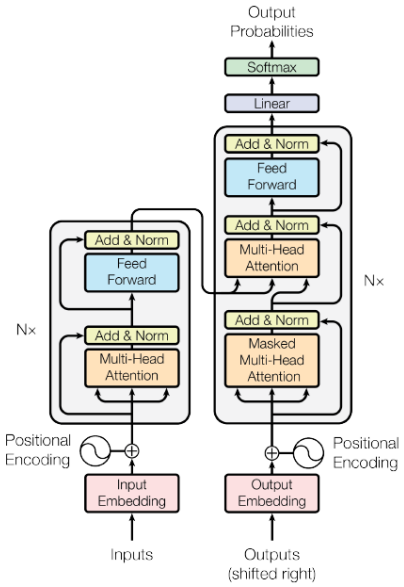

Figura 11: Arquitectura Transformer

[32]Recurrent Neural Networks

La importancia reside en su propuesta de que la atención, y no la secuencia de procesamiento, es vital para el rendimiento en tareas de NLP. Los Transformers logran esto a través de lo que se denomina ”atención de cabezas múltiples” (multi-head), que permite al modelo poder gestionar diferentes partes de la entrada simultáneamente, proporcionando una comprensión mejor del contexto y las relaciones entre palabras. Este enfoque además de mejorar la precisión también aumenta significativamente la eficiencia computacional, facilitando el entrenamiento de modelos más grandes y complejos.

Desde su introducción, los Transformers han dominado el campo del NLP, siendo la base para modelos posteriores como BERT [8] (ver sección 2), GPT [28] y otros, que han establecido nuevos estándares en una amplia gama de tareas de NLP. La capacidad de procesar todas las palabras de una secuencia de entrada en paralelo ha resultado en mejoras significativas tanto en eficiencia de entrenamiento como en la calidad del modelado del lenguaje.

5.2. Reconocimiento de Entidades

Los inicios de la tarea NER[17] se centraban en extraer información específica de textos limitados, como artículos de periódicos, y estaban muy influenciados por necesidades específicas de la industria, como la extracción de datos para bases de conocimientos o sistemas de consulta. En estos primeros días, NER dependía en gran medida de técnicas basadas en reglas que utilizaban listas de nombres (gazetteers) y patrones definidos manualmente para identificar y clasificar nombres de personas, organizaciones y ubicaciones. Estas reglas eran específicas para cada dominio y dependían en gran medida del conocimiento del experto en lingüística. Por ejemplo, una regla podría identificar nombres de personas si iban precedidos por títulos como ”Sr.” o ”Dr.”.

En términos más prácticos, para aplicar la tarea NER más simple, se puede usar un enfoque basado en un modelo de clasificación para cada palabra de la secuencia. Cada palabra w_i en la secuencia es clasificada individualmente en una etiqueta y_i, que puede representar entidades como personas, lugares, organizaciones, etc.

Dada una secuencia de palabras:

$w = (w_1, w_2, \ldots, w_n)$

Y teniendo una secuencia de etiquetas

$y = (y_1, y_2, \ldots, y_n)$

Una versión muy simplificada de este modelo puede asumir independencia condicional entre las etiquetas, y se puede formular como:

$$P(y \mid w) = \prod_{i=1}^{n} P(y_i \mid w_i) \tag{1}$$

Aquí, cada $P(y_i \mid w_i)$ es la probabilidad de que la etiqueta y_i sea asignada a la

palabra w_i.

Posteriormente, con la llegada del aprendizaje automático, estos modelos eran principalmente algoritmos de aprendizaje supervisado como árboles de decisión, sistemas basados en reglas de inducción y, eventualmente, modelos de máquina de vectores de soporte (SVM) [12]. Estos modelos mejoraron la capacidad de generalizar desde el entrenamiento a datos de prueba no vistos anteriormente, aunque seguían dependiendo en gran medida de características diseñadas manualmente.

La introducción de modelos basados en redes neuronales, como se menciona antes, las redes neuronales recurrentes (RNN) y luego las Long Short-Term Memory (LSTM), marcaron un antes y un después en la tarea NER por su capacidad de modelar dependencias a largo plazo y su eficacia en el manejo del contexto dentro de secuencias de texto. A pesar de sus ventajas, estos modelos seguían enfrentando desafíos relacionados con el desvanecimiento del gradiente y la necesidad de grandes cantidades de datos anotados para su entrenamiento efectivo.

Los Transformers han permitido el desarrollo de modelos pre-entrenados de lenguaje, como BERT [8] y GPT [28], que se adaptan eficazmente a la tarea de NER mediante un ajuste fino posterior (fine-tuning). Este método de "especializar" a los modelos de lenguaje, ha demostrado ser especialmente poderoso, logrando nuevos estándares en el rendimiento de apliación de la tarea NER al beneficiarse de representaciones lingüísticas profundas y contextualizadas generadas durante el preentrenamiento en grandes corpus de texto.

La tarea de Reconocimiento de Entidades (NER) utilizando el modelo BERT se puede expresar mediante la siguiente ecuación:

$$P(y|x) = \text{softmax}(W \cdot \text{BERT}(x) + b) \tag{2}$$

Donde:

- $x \in R^n$: Secuencia de entrada de n tokens.
- $y \in \{1, \ldots, K\}^n$: Etiquetas de entidades nombradas predichas para cada token, donde K es el número de clases de entidades.
- $\text{BERT}(x) \in R^{n \times d}$: Representaciones vectoriales de dimensión d producidas por BERT para cada token.
- $W \in R^{K \times d}$: Matriz de pesos aprendida durante el entrenamiento.
- $b \in R^K$: Vector de sesgo aprendido durante el entrenamiento.
- $\text{softmax} : R^K \to [0, 1]^K$: Función de activación que convierte los puntajes en distribuciones de probabilidad.

La ecuación modela la probabilidad condicional de las etiquetas y dado el texto de entrada x. El proceso se puede desglosar en los siguientes pasos:

1. BERT codifica la secuencia de entrada x en representaciones contextuales de alta dimensión.

2. Se aplica una transformación lineal ($W \cdot \text{BERT}(x) + b$) para proyectar estas representaciones al espacio de etiquetas.

3. La función softmax normaliza los puntajes resultantes en distribuciones de probabilidad sobre las posibles etiquetas para cada token.

Esta formulación permite que BERT capture efectivamente el contexto global y las dependencias complejas necesarias para realizar la tarea NER de forma precisa, superando las limitaciones del modelo simple de clasificación independiente.

5.3. Aplicación de la tarea NER

Para el desarrollo del proyecto se han investigado y probado diversidad de bibliotecas del lenguaje de programación Python para la realización de la tarea NER. La más básica es **NLTK**[33] [22] lanzado en 2001, fue una de las primeras bibliotecas de Python para NLP que incluyó capacidades de NER. NLTK utiliza principalmente métodos estadísticos y basados en reglas para el NER, como se menciona al principio de la sección 5.

spaCy[34], introducido en 2015, representó un salto cualitativo en el rendimiento y la facilidad de uso para tareas de NLP, incluyendo NER. A diferencia de NLTK, spaCy se diseñó desde el principio para ser eficiente y orientado a la producción. spaCy ha incorporado el uso de Transformers para aprovechar el poder de mejora. Su última versión que mejor realiza la tarea NER, es "spaCy RoBERTa (2020)". Utilizando la arquitectura del Transformer **RoBERTa** [21], es un modelo de lenguaje desarrollado por Meta AI[35] en 2019, consiste una versión optimizada y mejorada de BERT, donde se ha entrenado con más datos, durante más tiempo, con lotes más grandes y secuencias de tokens más larga.

Stanza [27] es una biblioteca de procesamiento de lenguaje natural (NLP) desarrollada por el Grupo de Procesamiento de Lenguaje Natural de Stanford. Fue lanzada en 2020 como una evolución de las herramientas de NLP de Stanford, ofreciendo una interfaz de Python moderna y eficiente para una variedad de tareas de NLP. Utiliza principalmente redes neuronales recurrentes (RNNs), específicamente LSTMs bidireccionales, para la mayoría de sus tareas.

Flair [2] es un framework NLP que utiliza embeddings contextuales a nivel de caracteres [3] para etiquetado de secuencias. Cuenta con modelos de lenguaje bidireccionales a nivel de caracteres (forward y backward) pre-entrenados en grandes corpus. Realiza una extracción de embeddings contextuales combinando estados

[33] Natural Language Toolkit
[34] https://spacy.io/
[35] https://ai.meta.com/meta-ai/

ocultos de estos modelos para cada palabra, para usarlos finalmente en un modelo BiLSTM-CRF para tareas como NER y PoS[36] tagging.

En la web de spaCy se puede encontrar un Benchmarking comparativo entre estos tres frameworks. Las métricas que se muestran se refieren a dos conjuntos de datos estándar para la evaluación de sistemas NER.

- **OntoNotes**[37]: Un corpus mulitlingüe y de dominio general.
- **CoNLL'03** [39]: Un conjunto de datos específico para NER en inglés.

Los números representan la puntuación F1 (explicación métrica en sección 5.6.1), una medida que combina precisión y exhaustividad (recall). Las puntuaciones son:

Sistema	OntoNotes	CoNLL '03
spaCy RoBERTa (2020)	89.8	91.6
Stanza (StanfordNLP)	88.8	92.1
Flair	89.7	93.1

Tabla 3: Comparación de rendimiento de sistemas NER

5.4. Etiquetas de Entidades

Para que los modelos comprendan la información que se les transmite para su procesamiento, primero deben haber "aprendido" en base a ejemplos la tarea NER que deben desempeñar. Estos ejemplos se definen en "**datasets**" que son una colección organizada de datos centrados en un área.

En relación a la tarea NER el etiquetado debe ser mediante "Gazetteers"[23] se refiere al etiquetado de entidades para la tarea de reconocimiento de entidades. El formato **IOB**[38], presentado por Ramshaw y Marcus en su artículo "Text Chunking using Transformation-Based Learning" [31] en 1995, es un esquema de etiquetado común para clasificar tokens en una tarea de segmentación. Este formato asigna etiquetas a cada token en una secuencia para indicar su posición relativa dentro de una entidad o segmento.

- B- (Beginning): El prefijo B- antes de una etiqueta indica que el token marca el inicio de un segmento o entidad. Específicamente, se utiliza cuando un segmento sigue inmediatamente a otro sin etiquetas O entre ellos.
- I- (Inside): El prefijo I- indica que el token está dentro de un segmento o entidad.

[36] Part-of-Speech
[37] https://catalog.ldc.upenn.edu/LDC2013T19
[38] Inside,Outside,Beginning

- O (Outside): La etiqueta O indica que el token no pertenece a ningún segmento o entidad.

Un ejemplo de una frase en un dataset con "Gazetteers" con el etiquetado "IOB", es el siguiente:

Alex	is	going	with	Marty	A.	Rick	to	Los	Angeles
B-PER	O	O	O	B-PER	I-PER	I-PER	O	B-LOC	I-LOC

En el Benchmarking anterior (ver sección 5.3) los datasets "Ontonotes" y "CoNLL '03", cuentan con este tipo de etiquetado.

Con el tiempo se han desarrollado nuevos formatos de etiquetado, como el empleado en este proyecto que es el "BIOES". Que añade las siguientes etiquetas:

- E- (End): Marca el final de una entidad.
- S- (Single): Indica que el token es una entidad de un solo elemento.

5.5. Creación del Dataset

En este capítulo se describe el proceso de construcción del dataset utilizado para entrenar un modelo de lenguaje enfocado en la tarea de Reconocimiento de Entidades(NER). Se ha utilizado el esquema de etiquetado BIOES mencionado anteriormente, que permite una clasificación detallada de las entidades en un texto, facilitando su identificación precisa.

Primero se deben establecer las etiquetas necesarias para el proyecto en el ámbito de la ciberseguridad y del proyecto desarrollado. Más adelante, en el prompt de sistema se especifica estas etiquetas con una descripción de lo que identifican.

Para llevar a cabo la construcción del dataset se desarrolló un GPT[39] basado en el modelo ChatGPT-4[40], personalizado para la tarea de etiquetado NER que su objetivo es generar frases con el nuevo etiquetado propio.

Como se cuenta en el artículo de creación de un gpt personalizado, se pueden establecer las siguientes características:

- Prompt de sistema: Es una instrucción o conjunto de instrucciones que se proporcionan a un modelo de inteligencia artificial, para guiar su comportamiento

[39] Generative Pre-trained Transformer
[40] https://help.openai.com/en/articles/8554397-creating-a-gpt

y respuestas en una tarea específica. Este prompt define el contexto, los roles, las reglas y los objetivos que el modelo debe seguir al procesar y generar respuestas.

- Ficheros de conocimiento: Son documentos que contienen información específica y detallada sobre un tema particular, utilizados para entrenar o enriquecer un modelo de inteligencia artificial con datos relevantes.
- Capacidades: Las habilidades o funciones que un modelo de inteligencia artificial puede ejecutar, en este caso, serían tres: Navegación por Internet, Generación de imágenes de DALL-E e Intérprete de código y análisis de datos.

En el caso del gpt personalizado, se aportaron ficheros de conocimiento del área de ciberseguridad con un amplio corpus de las entidades que se deben etiquetar.

Y en cuanto a las capacidades, solamente se estableció la función de análisis de datos.

El siguiente prompt de sistema fue utilizado para la tarea de cración del dataset:

```
Eres un experto NLP. Se encarga de realizar la tarea de un NER
(Text classification).

Se proporciona un fichero el cual leeras
y debes devolver un output de tal manera que crearas frases proporcionadas
del reporte, la divides en palabra por palabra junto con
la etiqueta correspondiente.

Todo esto lo devuelves linea por linea con una separacion de un espacio en la
etiqueta correspondiente. No aportes informacion extra a la conversación,
ni código adjunto, únicamente realiza la tarea de etiquetado.

El etiquetado tiene la siguiente terminologia como:

**S- = Single (Es una unica palabra la que representa la entidad).**

**B- = Begin (Es una palabra que comienza el grupo
de palabras que conforman la entidad)**

**I- = Intermediate (Es una palabra intermedia en el grupo
de palabras que conforman la entidad)**


**E- = End (Es la ultima palabra del grupo de
palabras que conforman la entidad)**
```

```
**La etiqueta O es para cualquier palabra que no tenga etiqueta,
sin el el formato previamente explicado**

Las entidades son acompañadas por ejemplos entre parentesis:
**IP = Se refiere a direcciones IP de todo tipo (IPv4:"192.168.1.120",
IPv6:2001:0000:130F:0000:0000:09C0:876A:130B)**

**APT = Se refiere a grupos cibercriminales (admin@338,APT12,Deep Panda)**

**MAL = Se refiere a familias de malware
(3PARA RAT,Agent Tesla,AsyncRAT,Bumblebee,Conti,Cobalt)**

**DOM = Se refiere a un nombre de dominio
(subdomain.domain.com, ej.ejemplo.ru, ejemp.ejemplo2.es)**

**URL = Se refiere a una url cualquiera
(https://www.ejemplo.es, http://ejemplo.ru, https://ejem.ejemplo.com)**

**VULN = Se refiere a vulnerabilidades
(Buffer Overflow,CRLF Injection,Memory leak,Improper Data Validation)**

**TOOL = Se refiere a herramientas software
(Cobalt Strike,bloodhound,impacket,poshc2,cmd,rundll)**

**HASH = Se refiere a cualquier tipo de hash y se detecta formato de hash
(sha1, sha2, md5,
098f6bcd4621d373cade4e832627b4f6,a94a8fe5ccb19ba61c4c0873d391e987982fbbd3)**

**LOC = Se refiere a localizaciones como ciudades, paises, etc
(Spain, Madrid, China,etc)**

**FILE = Se refieres a ficheros con todo tipo de extension y formato
(test.exe, data.txt, mal.dll)**

**OS = Se refiere a sistemas operativos
(Android, Windows, MacOS, IOS, Linux, Debian)***

**CVE = Se refiere a cualquier CVE-ID encontrado
(CVE-2023-36563,CVE-2023-4863,CVE-2012-0143)**

**SECTEAM = Se refiere a grupos de ciberseguridad aliados
(kaspersky,symantec,mandiant,cisco, palo alto unit 42, dell secure)**

Se proporciona ficheros con diversidad de ejemplos para que se comprueben
al realizar la tarea NER para las labels MAL,SECTEAM,APT,OPT,TOOL.
```

```
mal.txt,secteam.txt,apt.txt,opt.txt,tact.txt,tact.txt,tech.txt,tool.txt
respectivamente.

Devuelve la informacion para que sea accesible al copiado rapido.
Output linea por linea formato "Palabra Label"
```

Se muestra un ejemplo de una frase generada y etiquetada por el gpt personalizado:

Token	Etiqueta
”	O
kaspersky	S-SECTEAM
found	O
the	O
group	O
was	O
exploiting	O
a	O
adobe	B-TOOL
flash	I-TOOL
player	E-TOOL
zero-day	S-VULN
vulnerability	O
(	O
cve-2016-4117	S-CVE
)	O
to	O
remotely	O
deliver	O
the	O
latest	O
version	O
of	O
“	O
finspy	S-MAL
”	O
malware	O
,	O
according	O
to	O
a	O
new	O
blog	O
post	O
published	O
monday	S-TIME
.	O

Tabla 5: Etiquetado de ejemplo

Como se ha decidido emplear la arquitectura Transformers para la creación del modelo centrado en la tarea NER, se procede a asignar un identificador a cada etiqueta, pues no procesan texto directamente, en su lugar, trabajan con representaciones numéricas.

Cada palabra (token) y etiqueta (NER tag) se mapea a un identificador numérico para que el modelo pueda procesarlos de manera eficiente.

Etiqueta	ID	Etiqueta	ID	Etiqueta	ID
O	0	S-MAL	1	S-URL	2
B-TOOL	3	E-TOOL	4	S-TOOL	5
B-FILE	6	I-FILE	7	E-FILE	8
S-VULN	9	S-CVE	10	S-DOM	11
S-FILE	12	S-SECTEAM	13	B-URL	14
I-URL	15	E-URL	16	S-APT	17
S-OS	18	S-IP	19	B-OS	20
I-OS	21	E-OS	22	S-HASH	23
B-APT	24	I-APT	25	E-APT	26
B-MAL	27	E-MAL	28	B-OPT	29
E-OPT	30	S-OPT	31	I-OPT	32
S-EMAIL	33	B-SECTEAM	34	I-SECTEAM	35
E-SECTEAM	36	I-MAL	37	I-VULN	38
E-VULN	39	B-VULN	40	B-TIME	41
I-TIME	42	E-TIME	43	S-LOC	44
B-LOC	45	E-LOC	46	I-TOOL	47
S-TIME	48	B-IP	49	E-IP	50
B-HASH	51	E-HASH	52	I-LOC	53
B-EMAIL	54	E-EMAIL	55		

Tabla 7: Etiquetas y sus identificadores.

La frase de ejemplo anterior, con la transformación del etiquetado con los IDs aplicado para Transformers quedaría de la siguiente manera:

```
{"tokens":
    ["\u201d", "kaspersky", "found", "the", "group", "was", "
    ↪ exploiting", "a", "adobe", "flash", "player", "zero-day", "
    ↪ vulnerability", "(", "cve-2016-4117", ")", "to", "remotely",
    ↪ "deliver", "the", "latest", "version", "of", "\u201c", "
    ↪ finspy", "\u201d", "malware", ",", "according", "to", "a", "
    ↪ new", "blog", "post", "published", "monday", "."],
"ner_tags":
    [0, 13, 0, 0, 0, 0, 0, 0, 3, 47, 4, 9, 0, 0, 10, 0, 0, 0, 0, 0,
    ↪  0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 48, 0]}
```

El tamaño de los datasets finales de entrenamiento y validación son los siguientes:

- **Dataset de entrenamiento**: 10,584 líneas.
- **Dataset de validación**: 1,631 líneas.

5.6. Modelos elegidos

Como se ha visto tras la realización del mapping sistemático (ver sección 2, en la línea de investigación del proyecto se ha empleado sobre todo el modelo BERT y sus sucesores. Un punto a tener en cuenta, pues se han realizado intentos con una serie de modelos Transformer, quedando finalmente con el modelo que mejores métricas de entrenamiento y validación ha presentado y pruebas en el entorno real ha demostrado.

Para la elección del modelo de lenguaje Transformer, se establecieron tres criterios fundamentales:

- **Accesibilidad universal**: El modelo debe estar disponible para su uso por parte de cualquier investigador o desarrollador, independientemente de su ubicación o afiliación institucional. Esto implica que el modelo debe ser de código abierto o, al menos, tener una licencia que permita su uso académico y de investigación sin restricciones significativas.
- **Requisitos computacionales moderados**: El modelo debe poder ejecutarse sin la necesidad de recursos de cómputo de alto rendimiento. Esto es importante para garantizar que la investigación pueda replicarse en una variedad de entornos, desde ordenadores personales hasta servidores de gama media. La capacidad de ejecutar el modelo sin depender de hardware especializado o costoso amplía significativamente su aplicabilidad y accesibilidad.
- **Respaldo científico sólido:** El modelo debe estar respaldado por un equipo científico reconocido en el sector, con un historial probado en el desarrollo de modelos de lenguaje. Este criterio asegura que el modelo se basa en principios científicos sólidos, ha pasado por un riguroso proceso de desarrollo y validación, y cuenta con documentación y soporte adecuados.

Estos criterios no solo garantizan la viabilidad técnica del proyecto, sino que también aseguran su relevancia académica y su potencial para contribuir significativamente al campo del procesamiento del lenguaje natural, específicamente en la tarea de reconocimiento de entidades.

Uno de los repositorios por excelencia que cuenta con modelos subidos que cumpla los tres puntos definidos es *Hugging Face*[41]. Donde su lema literal es el siguiente: "*We are on a mission to democratize good machine learning, one commit at a time.*"

Hugging Face es una plataforma y comunidad líder en el campo del procesamiento del lenguaje natural, mantenida por Meta AI. Ofrece una biblioteca de código abierto llamada Transformers, que proporciona miles de modelos de lenguaje pre-entrenados, así como herramientas para entrenar, ajustar y desplegar estos modelos.

Para la búsqueda de modelos con renombre que desempeñan bien la tarea NER, se investigó específicamente en el apartado "Token Classification".

[41] https://huggingface.co/huggingface

Al explorar esta sección, se pudo identificar modelos que no solo cumplen con los tres criterios mencionados anteriormente (accesibilidad, requisitos computacionales moderados y respaldo científico sólido), sino que también han demostrado un rendimiento notable en tareas de NER.

Se muestra una tabla comparativa de parámetros y creadores entre los modelos seleccionados para entrenar:

Modelo	**Parámetros**	**Creador**
deBERTa	304M	Microsoft
GPT-2	137M	OpenAI
SECRoBERTa	84.1M	jackaduma
T5	223M	Google
ALBERT	223M	Google
BERT	110M	Google

Tabla 8: Modelos de lenguaje, sus parámetros y creadores

Todos estos modelos de lenguaje tienen la capacidad potencial de realizar la tarea NER, pero por defecto no están configurados específicamente para ello.

Para adaptar estos modelos se requiere el uso del tokenizador apropiado. El tokenizador[42] es una herramienta del procesamiento del lenguaje natural que divide el texto en unidades más pequeñas llamadas tokens. Para la tarea de "*Token Classification*", que incluye NER, se necesita un tokenizador específico. Este tokenizador no solo divide el texto en palabras o subpalabras, sino que también preserva la información necesaria para identificar entidades que pueden abarcar múltiples tokens.

Es necesario proporcionar al modelo el conjunto de etiquetas definido en la sección 5.4. Estas etiquetas representan las categorías de entidades que el modelo debe aprender a identificar.

Aunque estos modelos están pre-entrenados con vastas cantidades de texto, por sí solos no comprenden la tarea específica de NER ni cómo aplicar el esquema de etiquetado definido. Para ello debemos realizar un último paso: **Fine-Tune** o **Ajuste Fino** sobre el modelo.

5.6.1. Ajuste Fino

El ajuste fino, también conocido como fine-tuning, es una etapa importante en el aprendizaje por transferencia aplicado a modelos de lenguaje pre-entrenados. Este proceso permite adaptar un modelo general a una tarea específica, en este caso, el Reconocimiento de Entidades.

El objetivo es que el modelo aprenda a identificar y clasificar entidades nombradas en texto, este proceso aprovecha el conocimiento general del lenguaje adquirido durante el pre-entrenamiento y lo especializa para NER.

[42]https://docs.mistral.ai/guides/tokenization/

En cuanto a las configuraciones del fine-tune se deben especificar las siguientes:

- Se ajusta la capa de salida para que coincida con el número de etiquetas de NER
- Se configuran hiperparámetros como la tasa de aprendizaje, tamaño de lote, número de épocas, etc.

Durante el entrenamiento, los pesos del modelo se ajustan para optimizar el rendimiento en la tarea de NER y se utiliza una función de pérdida específica para tareas de clasificación de tokens, para tener en cuenta la capacidad del modelo y como mejora o empeora en el tiempo observan métricas que lo van evaluando.

En cuanto a las métricas, la más destacable es la medida **F1**, también conocida como F-score o F-measure, es una métrica de evaluación utilizada para medir el rendimiento de un modelo en problemas de clasificación, especialmente cuando se trata de un conjunto de datos desequilibrado. Combina la precisión (precision) y la exhaustividad (recall) en una sola métrica. Para poder entender esta métrica, se desglosa las medidas que la combinan:

Precisión

Es la proporción de instancias correctamente clasificadas como positivas sobre el total de instancias clasificadas como positivas.

$$\text{Precisión} = \frac{\text{Verdaderos Positivos (TP)}}{\text{Verdaderos Positivos (TP)} + \text{Falsos Positivos (FP)}} \tag{3}$$

Recall

Es la proporción de instancias correctamente clasificadas como positivas sobre el total de instancias que realmente son positivas.

$$\text{Exhaustividad} = \frac{\text{Verdaderos Positivos (TP)}}{\text{Verdaderos Positivos (TP)} + \text{Falsos Negativos (FN)}} \tag{4}$$

F1

La medida F1 es la media armónica de la precisión y el Recall. Se usa la media armónica en lugar de la media aritmética porque la media armónica penaliza más los valores extremos. Esto significa que un buen valor de F1 sólo se puede obtener si tanto la precisión como el Recall son altas.

$$\text{F1} = 2 \times \frac{\text{Precisión} \times \text{Recall}}{\text{Precisión} + \text{Recall}} \tag{5}$$

En la búsqueda de un método de fine-tune asequible en términos de coste computacional y tiempo, en la investigación se descubrió LoRa[43] [14].

[43]Low-Rank Adaptation

Low-Rank Adaptation es un método propuesto para adaptar grandes modelos de lenguaje preentrenados a tareas específicas de una manera más eficiente en términos de parámetros y computación. La idea principal es mantener congelados los pesos del modelo preentrenado y añadir matrices de descomposición de rango bajo (**A** y **B**) que sean entrenables en cada capa de la arquitectura del Transformer. Durante el entrenamiento, solo se actualizan estas matrices de rango bajo, reduciendo significativamente el número de parámetros entrenables y, por ende, el requerimiento de memoria GPU.

- **Reducción de Parámetros Entrenables:** LoRa reduce el número de parámetros entrenables al emplear matrices de rango bajo, lo que disminuye la necesidad de recursos computacionales y de memoria.
- **Mayor Eficiencia:** Como solo una pequeña fracción de los parámetros del modelo se entrena, LoRa permite una mayor eficiencia en términos de uso de GPU, reduciendo el consumo de VRAM hasta tres veces y permitiendo entrenar modelos más grandes con menos hardware.
- **Sin Latencia Adicional en Inferencia:** A diferencia de otros métodos como los adaptadores, LoRa no introduce latencia adicional durante la inferencia, ya que las matrices entrenables se combinan con los pesos preentrenados en el momento de la implementación.
- **Facilidad para Cambiar de Tarea:** Dado que se puede cambiar entre tareas simplemente sustituyendo las matrices **A** y **B**, LoRa facilita la creación y el despliegue de múltiples modelos específicos de tareas sin necesidad de almacenar múltiples instancias completas de los modelos finamente ajustados.

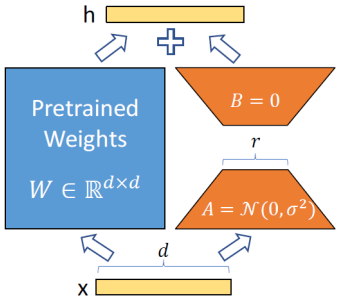

Figura 12: Entrenamiento LoRa

En el paper publicado de LoRa [14], se menciona la implementación en varios modelos de los cuales se han propuesto para el desarrollo de este proyecto, que son:

- RoBERTa
- DeBERTa

- GPT-2

Se proporcionan implementaciones y puntos de control para estos modelos, demostrando que LoRa puede mantener o incluso mejorar la calidad del modelo en comparación con el ajuste fino completo, pese a tener muchos menos parámetros entrenables.

5.7. Entrenamientos

La presente sección detalla el proceso de entrenamiento y evaluación de diversos modelos de lenguaje aplicados a la tarea NER. Se ha llevado a cabo un análisis comparativo de configuraciones en cada modelo para encontrar la mejor, con el objetivo de identificar el modelo que exhibe el rendimiento óptimo en términos de precisión, recall y medida F1. Se implementó la técnica LoRa para optimizar el proceso de fine-tuning, mostrando las configuraciones de cada modelo.

El protocolo experimental ha incluido una serie de iteraciones de entrenamiento, validación y prueba para cada modelo considerado. Se han empleado conjuntos de datos estandarizados y métricas de evaluación para garantizar la reproducibilidad y comparabilidad de los resultados obtenidos.

A continuación, se presenta una descripción detallada de cada modelo entrenado, incluyendo hiperparámetros de entrenamiento, porcentaje de parámetros entrenables, y resultados cuantitativos. Posteriormente, se ofrece un análisis crítico de los hallazgos, culminando en la selección justificada del modelo que ha demostrado el desempeño más sobresaliente en la tarea NER propuesta.

5.7.1. Hardware

Para el entrenamiento de los modelos se ha utilizado hardware de alto rendimiento, específicamente dos tarjetas gráficas NVIDIA RTX 6000 Ada Generation[44].

5.7.2. Parámetros LoRa y entrenamiento

Se muestra la explicación de los parámetros definidos en los modelos para efectuar el fine-tune y el entrenamiento que se ha llevado a cabo para realizar un estudio comparativo de resultados.

Los parámetros de LoRa son los siguientes:

- **r**: Define el rango de las matrices de adaptación LoRa. Un valor mayor permite más flexibilidad en el aprendizaje, pero aumenta el número de parámetros.
- **lora_alpha**: Es el factor de escala para los pesos LoRa. Controla la magnitud de la actualización LoRa en relación con los pesos originales.

[44]https://www.nvidia.com/es-es/design-visualization/rtx-6000/

- **lora_dropout**: Establece la tasa de dropout aplicada a las capas LoRa, ayudando a prevenir el sobreajuste.
- **target_modules**: Especifica las capas del modelo a las que se aplica LoRa, en este caso, las matrices de atención del transformador.
- **bias="all"**: Indica que todos los sesgos (bias) del modelo son entrenables.
- **modules_to_save**: Especifica qué módulos del modelo se guardarán completamente, no solo sus adaptadores LoRa.

Los parámetros de entrenamiento explicados, son los siguientes:

- **learning_rate**: Tasa de aprendizaje. Controla el tamaño de los pasos durante la optimización. Un valor de 4×10^{-4} es relativamente alto, lo que puede permitir un aprendizaje más rápido.
- **per_device_train_batch_siz**: Tamaño del lote de entrenamiento por dispositivo. Indica cuántas muestras se procesan simultáneamente durante el entrenamiento.
- **per_device_eval_batch_size**: Tamaño del lote de evaluación por dispositivo. Similar al anterior, pero para la fase de evaluación.
- **num_train_epochs**: Número de épocas de entrenamiento. El modelo recorrerá el conjunto de datos de entrenamiento 10 veces.
- **weight_decay**: Factor de regularización L2. Ayuda a prevenir el sobreajuste penalizando pesos grandes.
- **evaluation_strategy="epoch"**: Indica que la evaluación se realiza al final de cada época.
- **fp16=True**: Habilita el entrenamiento en precisión mixta (float16), lo que puede acelerar el entrenamiento y reducir el uso de memoria.
- **save_strategy="epoch"**: Guarda el modelo al final de cada época.
- **load_best_model_at_end=True**: Carga el mejor modelo (según la métrica de evaluación) al final del entrenamiento.

5.7.3. BERT

El modelo BERT[45] se propuso en "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" [8]. Es un modelo de lenguaje desarrollado por Google en 2018 que revolucionó el campo del procesamiento del lenguaje

[45]https://huggingface.co/docs/transformers/model_doc/bert

natural. Basado en la arquitectura Transformer, BERT se caracteriza por su capacidad de aprender representaciones contextuales bidireccionales de texto, lo que le permite comprender el contexto completo de una palabra en una frase. El modelo se pre-entrena en grandes corpus de texto sin etiquetar y luego se puede ajustar fácilmente para una amplia gama de tareas de NLP, como clasificación de texto, respuesta a preguntas y la tarea NER, logrando resultados de vanguardia en numerosos benchmarks.

A continuación, se muestra el entrenamiento del modelo detallando las configuraciones y parámetros utilizados:

Configuración LoRa

```
r=16
lora_alpha=16
lora_dropout=0.12
target_modules=["key", "query", "value"]
bias="all"
modules_to_save=["classifier"]
```

Listing 2: BERT LoRa

Al modificar los valores de LoRa ocurre que se define que cantidad de parámetros son entrenables del modelo para el fine-tune, con la configuración anterior se queda en:

- Parámetros entrenables: 1,039,172
- Total de parámetros: 109,880,968
- Porcentaje entrenable: 0.9457 %

Configuración del entrenamiento

```
learning_rate=4e-4
per_device_train_batch_size=8
per_device_eval_batch_size=16
num_train_epochs=10
weight_decay=0.03
evaluation_strategy="epoch"
fp16=True
save_strategy="epoch"
load_best_model_at_end=True
```

Listing 3: BERT Entrenamiento

A continuación, se presenta una tabla con los resultados del entrenamiento por época:

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	0.688600	0.347525	0.746056	0.697940	0.721196	0.906410
2	0.338200	0.350270	0.721254	0.745376	0.733117	0.904852
3	0.290100	0.322183	0.745688	0.749976	0.747826	0.911142
4	0.233600	0.331471	0.746750	0.787254	0.766468	0.913958
5	0.212700	0.332425	0.742793	0.763009	0.752765	0.909425
6	0.192600	0.356532	0.736161	0.773551	0.754393	0.907408
7	0.185200	0.350614	0.732271	0.779780	0.755279	0.907149
8	0.164400	0.355656	0.742879	0.774892	0.758501	0.909585
9	0.154100	0.361981	0.744423	0.764255	0.754208	0.907368
10	0.155000	0.362567	0.743427	0.774988	0.758880	0.909205

Tabla 9: Resultados del Entrenamiento de BERT

El tiempo total de entrenamiento fue de 19:24 minutos para las 10 épocas. Estos resultados demuestran la efectividad de la implementación de LoRa en el fine-tuning del modelo BERT para la tarea de NER, logrando una convergencia rápida y un rendimiento estable a lo largo de las épocas de entrenamiento.

5.7.4. ALBERT

ALBERT[46] (A Lite BERT) es una variante optimizada de BERT diseñada para mejorar la eficiencia del modelo en términos de memoria y tiempo de entrenamiento. Desarrollado por investigadores de Google Research y Toyota Technological Institute at Chicago, ALBERT[18] introduce dos técnicas principales para reducir el número de parámetros: la parametrización de embeddings factorizados y el compartir parámetros entre capas. Estas técnicas permiten a ALBERT mantener un rendimiento competitivo con menos parámetros y una mayor velocidad de entrenamiento.

A continuación, se muestra el entrenamiento del modelo detallando las configuraciones y parámetros utilizados:

Configuración LoRa

```
r=16
lora_alpha=16
lora_dropout=0.15
target_modules=["key", "query", "value"]
bias="all"
modules_to_save=["classifier"]
```

Listing 4: ALBERT LoRa

Al modificar los valores de LoRa, se define qué cantidad de parámetros son entrenables del modelo para el fine-tune, con la configuración anterior se queda en:

- Parámetros entrenables: 721,092
- Total de parámetros: 206,764,680

[46]https://huggingface.co/docs/transformers/model_doc/albert

- Porcentaje entrenable: 0.3488 %

Configuración del entrenamiento

```
learning_rate=5e-4
per_device_train_batch_size=8
per_device_eval_batch_size=16
num_train_epochs=10
weight_decay=0.03
evaluation_strategy="epoch"
save_strategy="epoch"
load_best_model_at_end=True
```

Listing 5: ALBERT Entrenamiento

A continuación, se presenta una tabla con los resultados del entrenamiento por época:

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	0.544200	0.323921	0.784242	0.687633	0.732767	0.914334
2	0.267700	0.307875	0.766215	0.759034	0.762607	0.919877
3	0.230100	0.296119	0.787799	0.756135	0.771643	0.921798
4	0.170900	0.317090	0.749331	0.757005	0.753148	0.918103
5	0.142400	0.317234	0.761704	0.751401	0.756518	0.916072
6	0.121300	0.328367	0.757605	0.750725	0.754149	0.914042
7	0.108000	0.344386	0.754866	0.749469	0.752157	0.915432
8	0.076800	0.359593	0.751389	0.744928	0.748144	0.914572
9	0.063600	0.370407	0.753077	0.739034	0.745989	0.912560
10	0.056600	0.375747	0.746877	0.745217	0.746046	0.912779

Tabla 10: Resultados del Entrenamiento de ALBERT

El tiempo total de entrenamiento fue de 3 horas y 44 minutos para las 10 épocas. Aunque ALBERT está diseñado para ser más eficiente en términos de parámetros mediante la factoración matricial y el uso compartido de parámetros, estos cambios pueden introducir una complejidad adicional durante el entrenamiento. Diferentes tasas de aprendizaje, tamaños de batch, y estrategias de optimización pueden impactar significativamente los tiempos de entrenamiento. El proceso de compartir y descomponer matrices puede requerir más operaciones computacionales por iteración en comparación con BERT.

5.7.5. GPT-2

GPT-2[47] (Generative Pre-trained Transformer 2) es un modelo de lenguaje desarrollado por OpenAI que se destaca por su capacidad para realizar múltiples tareas de procesamiento de lenguaje natural sin necesidad de entrenamiento específico para cada tarea, como se explica en [29].

[47] https://huggingface.co/docs/transformers/model_doc/gpt2

A continuación, se muestra el entrenamiento del modelo detallando las configuraciones y parámetros utilizados:

Configuración LoRa

```
r=12
lora_alpha=32
lora_dropout=0.1
```

Listing 6: GPT-2 LoRa

Al modificar los valores de LoRa, definimos qué cantidad de parámetros son entrenables del modelo para el fine-tune, con la configuración anterior se queda en:

- Parámetros entrenables: 2,300,229
- Total de parámetros: 776,418,698
- Porcentaje entrenable: 0.2963 %

Configuración del entrenamiento

```
learning_rate=1e-4
per_device_train_batch_size=8
per_device_eval_batch_size=8
num_train_epochs=10
weight_decay=0.01
evaluation_strategy="epoch"
save_strategy="epoch"
load_best_model_at_end=True
```

Listing 7: GPT-2 Entrenamiento

A continuación, se presenta una tabla con los resultados del entrenamiento por época:

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	No log	0.495206	0.569272	0.517307	0.542047	0.863759
2	1.010400	0.461356	0.569958	0.549388	0.559484	0.863568
3	0.574800	0.447093	0.596949	0.574082	0.585292	0.870078
4	0.499700	0.432039	0.632128	0.575559	0.602519	0.875570
5	0.455400	0.437789	0.625710	0.604580	0.614964	0.877542
6	0.416900	0.432039	0.632651	0.606480	0.619289	0.877118
7	0.416900	0.441278	0.614695	0.607429	0.611040	0.874276
8	0.394000	0.445935	0.600000	0.618932	0.609319	0.872686
9	0.371000	0.446493	0.615024	0.622942	0.618958	0.875252
10	0.361500	0.448329	0.612119	0.619354	0.615715	0.875167

Tabla 11: Resultados del Entrenamiento de GPT-2

El tiempo total de entrenamiento fue de 28 minutos y 36 segundos para las 10 épocas. Estos resultados demuestran la efectividad de la implementación de LoRa en el fine-tuning del modelo GPT-2 para la tarea de NER, logrando una convergencia rápida y un rendimiento estable a lo largo de las épocas de entrenamiento.

5.7.6. SecRoBERTa

SecRoBERTa[48] es un modelo de lenguaje preentrenado desarrollado específicamente para textos de ciberseguridad. Es una variante del modelo RoBERTa adaptada para manejar y comprender mejor el contenido relacionado con la ciberseguridad. El modelo fue presentado en el trabajo "SecBERT: A Pretrained Language Model for Cyber Security Text"[1].

Fue entrenado utilizando un corpus especializado que incluye documentos y reportes de ciberseguridad y utiliza un vocabulario de piezas de palabras propio, llamado *secvocab*, que ha sido construido para coincidir de la mejor manera con el corpus de entrenamiento. Este vocabulario especializado permite al modelo manejar terminología específica y compleja relacionada con la ciberseguridad de manera más efectiva.

Esta propuesta es muy similar a la que se ha propuesto en este proyecto, pero habiendo desarrollado una propia ontologia para dicho cometido.

A continuación, se muestra el entrenamiento del modelo detallando las configuraciones y parámetros utilizados:

Configuración LoRa

```
r=lora_r
lora_alpha=32
lora_dropout=0.1
target_modules=["query", "value","key"]
```

Listing 8: secRoBERTa LoRa

Al modificar los valores de LoRa, definimos qué cantidad de parámetros son entrenables del modelo para el fine-tune, con la configuración anterior se queda en:

- Parámetros entrenables: 384,837
- Total de parámetros: 83,298,186
- Porcentaje entrenable: 0.4620 %

A continuación, se presenta una tabla con los resultados del entrenamiento por época:

[48] https://huggingface.co/jackaduma/SecRoBERTa

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	No log	0.587211	0.568402	0.411681	0.477511	0.856239
2	1.184000	0.522946	0.567318	0.548828	0.557920	0.864745
3	0.659000	0.479289	0.619968	0.545855	0.580555	0.871319
4	0.572700	0.451860	0.653703	0.543833	0.593728	0.875983
5	0.527900	0.445168	0.651452	0.573808	0.610169	0.877382
6	0.492300	0.444727	0.637534	0.589152	0.612389	0.877360
7	0.492300	0.436606	0.648175	0.585108	0.615029	0.878759
8	0.471800	0.439799	0.628118	0.607946	0.617868	0.878470
9	0.451700	0.435430	0.632797	0.608184	0.620246	0.879227
10	0.443600	0.436808	0.635051	0.606875	0.620644	0.879270

Tabla 12: Resultados del Entrenamiento de secRoBERTa

El tiempo total de entrenamiento fue de 3 minutos y 48 segundos para las 10 épocas. Estos resultados demuestran la efectividad de la implementación de LoRa en el fine-tuning del modelo secRoBERTa para la tarea NER.

5.7.7. T5

T5[49] (Text-to-Text Transfer Transformer) es un modelo de lenguaje desarrollado por Google Research que convierte todas las tareas de procesamiento de lenguaje natural (NLP) en problemas de "*texto a texto*". Este enfoque unificado permite que el mismo modelo y procedimiento de entrenamiento se apliquen a diversas tareas, como traducción, respuesta a preguntas, resumen de texto y clasificación de texto. T5 ha sido preentrenado en un enorme corpus de datos no etiquetados y luego ajustado para tareas específicas, logrando resultados de vanguardia en múltiples benchmarks de NLP [30].

Configuración LoRa

```
r=8
lora_alpha=16
lora_dropout=0.15
target_modules=["k", "q", "v", "o"]
bias="all"
modules_to_save=["classifier"]
```

Listing 9: T5 LoRa

Al modificar los valores de LoRa, se define qué cantidad de parámetros son entrenables del modelo para el fine-tune, con la configuración anterior se queda en:

- Parámetros entrenables: 642,500
- Total de parámetros: 110,322,952
- Porcentaje entrenable: 0.5824 %

[49] https://huggingface.co/docs/transformers/model_doc/t5

Configuración del entrenamiento

```
learning_rate=1e-3
per_device_train_batch_size=8
per_device_eval_batch_size=16
num_train_epochs=10
weight_decay=0.03
evaluation_strategy="epoch"
fp16=True
save_strategy="epoch"
load_best_model_at_end=True
```

Listing 10: T5 Entrenamiento

A continuación, se presenta una tabla con los resultados del entrenamiento por época:

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	1.822100	0.576468	0.697127	0.478573	0.567536	0.860068
2	0.656700	0.429477	0.718305	0.675308	0.696143	0.890179
3	0.496100	0.471696	0.751312	0.681808	0.714874	0.895927
4	0.368600	0.420835	0.731352	0.747714	0.739442	0.900333
5	0.332200	0.398957	0.733492	0.744690	0.739049	0.897451
6	0.297700	0.393413	0.754381	0.771144	0.762670	0.905327
7	0.286600	0.444431	0.768936	0.744237	0.756385	0.902897
8	0.251200	0.449325	0.767569	0.762754	0.765154	0.905947
9	0.239100	0.436914	0.767266	0.751493	0.759297	0.902277
10	0.240800	0.455708	0.761455	0.761167	0.761311	0.903115

Tabla 13: Resultados del Entrenamiento de T5

El tiempo total de entrenamiento fue de 22 minutos y 8 segundos para las 10 épocas. Estos resultados demuestran la efectividad de LoRa en el fine-tuning del modelo T5 para la tarea NER.

5.7.8. DeBERTa

DeBERTa[50] (Decoding-enhanced BERT with disentangled attention) es un modelo de lenguaje basado en Transformer que introduce mejoras significativas sobre BERT y RoBERTa [11]. Este modelo utiliza un mecanismo de atención desentrelazada que representa cada palabra con dos vectores distintos para codificar su contenido y posición. Los pesos de atención entre las palabras se calculan utilizando matrices desentrelazadas, basándose en sus contenidos y posiciones relativas, lo que mejora la precisión del modelo.

Además, incorpora un decodificador de máscara mejorado que utiliza posiciones absolutas en la capa de decodificación para predecir tokens enmascarados durante el preentrenamiento, lo que permite al modelo distinguir mejor entre contextos similares pero funcionalmente diferentes. Estas innovaciones permiten a DeBERTa

[50]https://huggingface.co/docs/transformers/model_doc/deberta-v2

mejorar tanto la eficiencia del pre-entrenamiento como el rendimiento en diversas tareas de procesamiento del lenguaje natural.

DeBERTa ha demostrado su superioridad en benchmarks como GLUE [41] y SuperGLUE [34], superando incluso el rendimiento humano en algunos casos. Con su capacidad para lograr altos niveles de rendimiento con menos datos de entrenamiento en comparación con otros modelos, DeBERTa representa un avance significativo en la arquitectura de modelos de lenguaje.

A continuación, se muestra el entrenamiento del modelo detallando las configuraciones y parámetros utilizados:

Configuración LoRa

```
inference_mode=False
r=16
lora_alpha=16
lora_dropout=0.15
target_modules=["key_proj", "query_proj", "value_proj"]
bias="all"
modules_to_save=["classifier"]
```

Listing 11: deBERTa LoRa

Al modificar los valores de LoRa, se define qué cantidad de parámetros son entrenables del modelo para el fine-tune, con la configuración anterior se queda en:

- Parámetros entrenables: 1,030,712
- Total de parámetros: 184,802,416
- Porcentaje entrenable: 0.5577 %

Configuración del entrenamiento

```
learning_rate=5e-4
per_device_train_batch_size=8
per_device_eval_batch_size=16
num_train_epochs=10
weight_decay=0.01
evaluation_strategy="epoch"
fp16=True
save_strategy="epoch"
```

Listing 12: deBERTa Entrenamiento

A continuación, se presenta una tabla con los resultados del entrenamiento por época:

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	0.524100	0.283164	0.770974	0.767845	0.769406	0.920171
2	0.271500	0.257808	0.814428	0.779032	0.796337	0.927559
3	0.222500	0.251960	0.802273	0.810613	0.806421	0.931047
4	0.179200	0.269056	0.814354	0.809920	0.812131	0.929611
5	0.162200	0.287415	0.791537	0.807445	0.799412	0.925363
6	0.154200	0.305727	0.788956	0.817543	0.802995	0.922654
7	0.131400	0.281950	0.807734	0.814771	0.811237	0.927189
8	0.123400	0.300885	0.790463	0.807445	0.798864	0.922654
9	0.115000	0.301779	0.800663	0.812791	0.806681	0.925260
10	0.106000	0.301117	0.800506	0.815167	0.807770	0.925671

Tabla 14: Resultados del Entrenamiento de deBERTa

El tiempo total de entrenamiento fue de 35 minutos y 49 segundos para las 10 épocas. Estos resultados demuestran la efectividad de la implementación de LoRa en el fine-tuning del modelo deBERTa para la tarea NER, logrando una convergencia rápida y un rendimiento estable a lo largo de las épocas de entrenamiento.

El código del entrenamiento del modelo está detallado en el apéndice E. La prueba con una frase donde se encuentran presentes entidades, se muestra en el apéndice F.

5.8. Comparativa y elección

La métrica de F1-score se utiliza en este estudio porque proporciona un balance entre precisión y recall, ofreciendo una medida más completa del rendimiento del modelo en comparación con métricas individuales. Para una explicación detallada de esta métrica, ver la sección 5.6.1.

Modelo	Época 1	Época 5	Época 10
BERT	0.721196	0.752765	0.758880
ALBERT	0.732767	0.756518	0.746046
GPT-2	0.542047	0.614964	0.615715
secRoBERTa	0.477511	0.610169	0.620644
T5	0.567536	0.739049	0.761311
deBERTa	0.769406	0.799412	0.807770

Tabla 15: Tabla comparación del F1-score

Al analizar los F1-scores de los distintos modelos a lo largo de las épocas de entrenamiento, se observa lo siguiente:

- **BERT** muestra un rendimiento constante y mejora gradual a lo largo de las épocas, alcanzando un F1-score de 0.758880 en la época 10.

- **ALBERT** también presenta un rendimiento sólido, aunque su F1-score disminuye ligeramente hacia la última época.

- **GPT-2** tiene el F1-score más bajo entre los modelos comparados, indicando un rendimiento inferior en la tarea NER.

- **secRoBERTa** presenta mejoras, pero su F1-score sigue siendo menor en comparación con otros modelos.

- **T5** muestra una mejora significativa, alcanzando un F1-score de 0.761311 en la última época.

- **deBERTa** tiene el mejor rendimiento, con un F1-score que comienza alto y mejora consistentemente, alcanzando 0.807770 en la época 10.

En base a los F1-scores observados, el modelo **deBERTa** es claramente el mejor modelo para la tarea NER. Este modelo no solo comienza con un F1-score alto, sino que también muestra una mejora constante y significativa a lo largo de las épocas, alcanzando el F1-score más alto de 0.807770. Esto indica que deBERTa es el modelo más eficaz y robusto para la tarea NER con el corpus de la ontología definido.

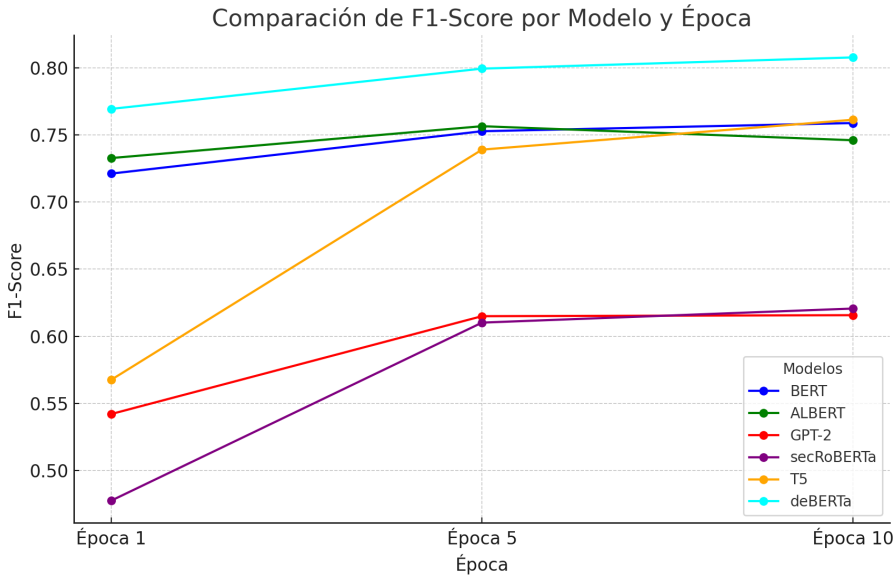

Figura 13: Comparativa F1 gráfico

6. Sistema de inteligencia de amenazas

La aplicación desarrollada es un sistema avanzado de inteligencia de amenazas que integra técnicas de búsqueda elástica e inteligencia artificial. El sistema está diseñado para recopilar, procesar y analizar información de reportes de ciberseguridad proporcionados por empresas reconocidas del sector, mejorando significativamente la eficiencia y eficacia en la detección de amenazas cibernéticas.

El sistema destaca por su capacidad para procesar datos no estructurados de reportes de ciberseguridad, utilizando un modelo de lenguaje avanzado con técnicas de procesamiento del lenguaje natural (NLP), descrito en los puntos anteriores de

este informe técnico.El cual permite extraer entidades clave como vulnerabilidades, tácticas, técnicas y procedimientos (TTP) de atacantes, e indicadores de compromiso (IoC). Estas entidades se almacenan en una base de datos que emplea técnicas de búsqueda elástica, lo que permite una recuperación eficiente y rápida de la información.

Se ha desarrollado una interfaz web segura y accesible para los analistas de seguridad, diseñada para ofrecer una experiencia de usuario optimizada. Esta interfaz permite la ejecución de tareas asíncronas gestionadas por Celery, un framework de Python especializado en la gestión de tareas distribuidas. Para la gestión de estas tareas, se utiliza Redis como broker de datos.

La tecnología Docker se ha elegido para el despliegue de esta aplicación debido a su capacidad para encapsular cada función del sistema en contenedores separados. Esta metodología garantiza la portabilidad, escalabilidad y seguridad del sistema.

6.1. Docker

Docker[51] se basa en la idea de los contenedores, que son unidades estándar de software que empaquetan el código y todas sus dependencias para que la aplicación se ejecute de manera rápida y confiable en diferentes entornos informáticos.

6.1.1. Arquitectura

Docker utiliza una arquitectura cliente-servidor. El cliente de Docker se comunica con el daemon de Docker (dockerd), que se encarga de las tareas pesadas de construcción, ejecución y distribución de contenedores Docker. El cliente y el daemon pueden ejecutarse en el mismo sistema o el cliente puede conectarse a un daemon remoto. Se comunican utilizando una API REST, a través de sockets UNIX o una interfaz de red.

[51] https://www.docker.com/

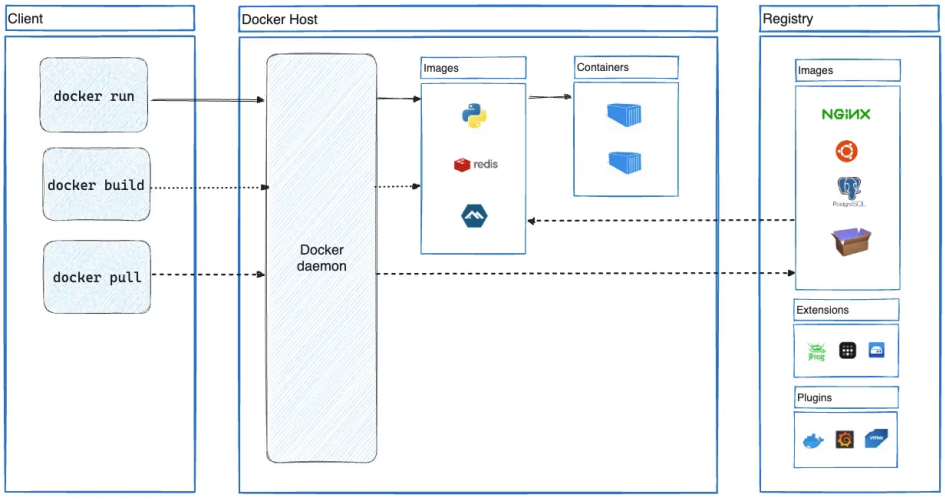

Figura 14: Arquitectura Docker

El daemon de Docker (dockerd) escucha las solicitudes de la API de Docker y gestiona objetos Docker como imágenes, contenedores, redes y volúmenes. También puede comunicarse con otros daemons para gestionar servicios Docker.

El cliente de Docker (docker) es la principal interfaz de los usuarios para interactuar con Docker. Utiliza comandos como `docker run`, enviando estas instrucciones al daemon para su ejecución. El cliente puede comunicarse con múltiples daemons.

Docker Desktop[52] es una aplicación fácil de instalar para entornos Mac, Windows o Linux que permite construir y compartir aplicaciones y microservicios en contenedores. Incluye el daemon de Docker, el cliente de Docker, Docker Compose, Docker Content Trust, Kubernetes y Credential Helper. Para el desarrollo del proyecto, se ha empleado esta aplicación sobre un sistema operativo Windows 11, con el Subsistema de Windows para Linux[53].

6.1.2. Componentes

- **Imágenes Docker**: Las imágenes son plantillas de solo lectura que contienen el sistema operativo y el software necesario para ejecutar una aplicación. Cada imagen está construida a partir de una serie de capas que representan cambios o adiciones al sistema de archivos.
- **Contenedores Docker**: Los contenedores son instancias en ejecución de imágenes Docker. Cada contenedor es una entidad aislada con su propio sistema de archivos, red y entorno de ejecución, lo que garantiza que las aplicaciones funcionen de manera consistente independientemente del entorno.
- **Dockerfile**: Un archivo Dockerfile es un script de texto que contiene instrucciones para construir una imagen Docker. Cada línea en un Dockerfile representa

[52]https://www.docker.com/products/docker-desktop/
[53]https://learn.microsoft.com/en-us/windows/wsl/install

una instrucción que añade una capa a la imagen.

- **Docker compose**: Un archivo de configuración en formato YAML utilizado por Docker Compose para definir y gestionar múltiples contenedores Docker. Este archivo especifica cómo deben ser configurados, construidos y conectados los diferentes servicios que componen una aplicación, permitiendo su despliegue y orquestación con un solo comando.
- **Docker Hub**: Docker Hub es un servicio de registro de Docker que permite a los usuarios buscar, compartir y almacenar imágenes Docker. Es un recurso clave para encontrar imágenes oficiales y personalizadas.

6.1.3. Ventajas de Uso

Docker ofrece numerosas ventajas que lo hacen ideal para el despliegue de aplicaciones modernas:

- **Portabilidad**: Los contenedores Docker pueden ejecutarse de manera consistente en cualquier entorno que soporte Docker, ya sea en una máquina local, en un servidor en la nube o en un entorno híbrido.
- **Escalabilidad**: Docker facilita la escalabilidad horizontal de las aplicaciones, permitiendo la creación rápida de múltiples instancias de contenedores para manejar aumentos en la carga de trabajo.
- **Eficiencia de Recursos**: Los contenedores son más ligeros que las máquinas virtuales tradicionales porque comparten el kernel del sistema operativo anfitrión, lo que reduce la sobrecarga de recursos.
- **Aislamiento y Seguridad**: Cada contenedor se ejecuta en un entorno aislado, lo que mejora la seguridad y evita conflictos entre aplicaciones.
- **Facilidad de Integración y Despliegue Continuo**: Docker se integra bien con herramientas de integración continua y despliegue continuo (CI/CD), facilitando el desarrollo ágil y la entrega continua de software.

En este proyecto, Docker se utiliza para desplegar y gestionar los diversos componentes del sistema de inteligencia de amenazas. Cada función del sistema se encapsula en un contenedor separado, lo que permite una administración eficiente y escalable de los servicios. Posteriormente a la arquitectura propuesta, en las siguientes subsecciones, se explica al detalle el funcionamiento de cada contenedor.

6.2. Arquitectura del sistema

La arquitectura del sistema se ilustra en la Figura 15. La aplicación consta de cinco contenedores principales que interactúan entre sí para ofrecer las funcionalidades requeridas. Para ver el código de despliegue implicado, tanto el fichero *docker compose*, como los ficheros *Dockerfile* de cada contenedor ir a los anexos B y C.

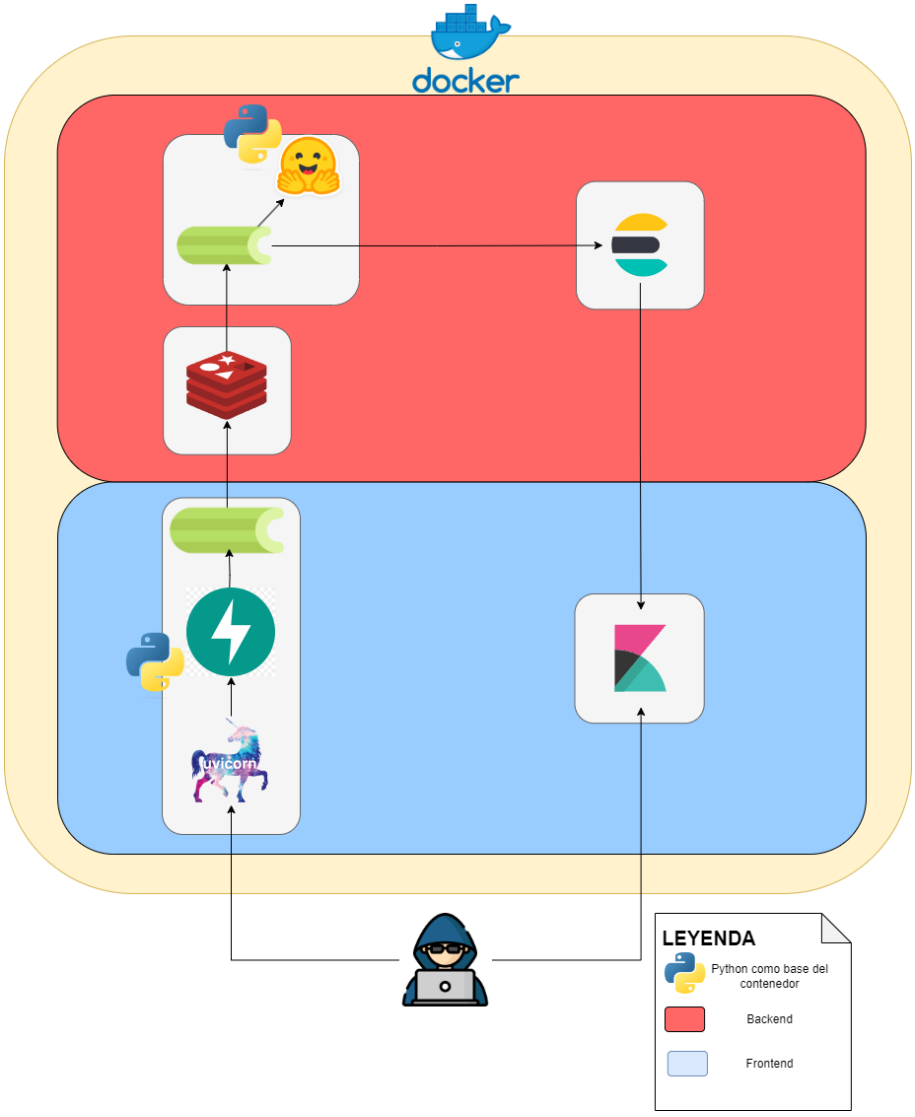

Figura 15: Arquitectura del sistema

6.3. Contenedores

En esta sección se detallan los diferentes contenedores Docker que componen el sistema de inteligencia de amenazas. Cada contenedor encapsula una función específica, facilitando su administración y escalabilidad. A continuación, se explican las características y roles de cada contenedor dentro del sistema.

6.3.1. Web

El contenedor web está basado en una imagen personalizada que utiliza como base **python:slim**. Esta imagen base es una versión ligera de Python, optimizada para tener un tamaño reducido y una mayor eficiencia en el despliegue. Sobre esta imagen base, se construyen e instalan diversas dependencias esenciales para el funcionamiento del servicio web.

El servicio principal dentro del contenedor es **FastAPI**[54], un framework de alto rendimiento para la creación de aplicaciones web y API en Python. FastAPI se destaca por su rapidez y capacidad de manejar un alto número de solicitudes concurrentes, lo que lo hace adecuado para aplicaciones modernas. Está ejecutado mediante **Uvicorn**, que es un servidor ASGI[55] ligero y de alto rendimiento que se utiliza para ejecutar aplicaciones FastAPI.

Las dependencias que se incluyen en este contenedor (para más detalle ir al apéndice C) son:

- *Ficheros python con la lógica de la API*: Con FastAPI, se construye el framework principal para la creación y gestión de la API web.
- *Ficheros html y css*: Permiten la renderización de la información en un navegador web.
- *Celery*: Cuenta con un cliente celery para añadir tareas a *Redis* y que se ejecuten de forma asíncrona sin que se quede en espera bloqueante la web.

Este contenedor, sirve como el front-end de la aplicación, gestionando todas las solicitudes HTTP entrantes a través de HTTPs, lo que garantiza la seguridad de las comunicaciones. Al iniciar, Uvicorn levanta la aplicación FastAPI y escucha en el puerto 443, manejando tanto solicitudes sincronas como asincronas.

FastAPI proporciona las rutas y la lógica de la aplicación, mientras que Celery se encarga de delegar tareas en segundo plano a Redis. Por ejemplo, cuando una solicitud requiere procesamiento intensivo o prolongado, FastAPI delega esta tarea a Celery, que la coloca en la cola de Redis para su procesamiento asíncrono, con el otro contenedor celery worker. Esta arquitectura permite una respuesta rápida a las solicitudes del usuario, mejorando la escalabilidad y el rendimiento de la aplicación.

El uso de certificados SSL generados mediante OpenSSL[56] asegura que todas las comunicaciones entre los clientes y el servidor sean cifradas, protegiendo los datos sensibles durante su transmisión.

[54] https://fastapi.tiangolo.com/
[55] Asynchronous Server Gateway Interface
[56] https://www.openssl.org/

6.3.2. Redis

El contenedor redis está basado en la imagen oficial de Redis, una base de datos en memoria de estructura de datos de código abierto, que se utiliza como un almacén de claves y valores. Redis es conocido por su alto rendimiento, soporte para una amplia gama de tipos de datos y operaciones atómicas, lo que lo convierte en una opción ideal para el almacenamiento en caché, la gestión de sesiones y la implementación de colas de mensajes.

Backend de mensajes: En combinación con Celery, Redis actúa como un backend de mensajes, gestionando la cola de tareas. Cuando el contenedor web delega tareas a Celery, estas se colocan en la cola gestionada por Redis, que luego son procesadas por los workers de Celery.

6.3.3. Celery

Celery[57] es un sistema de cola de tareas distribuido que permite el procesamiento de tareas en segundo plano en una aplicación web. En la arquitectura que se despliega, Celery se utiliza en conjunto con Redis (como broker y backend) para manejar tareas asincrónicas que pueden ser ejecutadas por uno o más workers.

```
from celery import Celery
app = Celery(
          'tasks',
          broker='redis://redis:6379/0',
          backend='redis://redis:6379/0'
          )
```

Listing 13: Celery code broker/backend

El funcionamiento de Celery dentro del sistema se puede entender en base a la siguiente arquitectura:

1. Cliente (FastAPI): Las acciones del usuario desencadenan solicitudes a la aplicación FastAPI. Algunas de estas acciones pueden requerir procesamiento intensivo o prolongado, que se delega a Celery para su ejecución en segundo plano.

2. Envío de tareas: FastAPI, actuando como cliente de Celery, envía mensajes que representan tareas a la cola de tareas "*task queue*" gestionada por Redis.

3. Broker: Redis actúa como intermediario para la cola de tareas, gestionando y distribuyendo las tareas a los workers disponibles.

4. Worker: El contenedor worker ejecuta las tareas asignadas por Redis. Pueden haber múltiples workers ejecutando en paralelo, lo que mejora la capacidad de procesamiento y la escalabilidad del sistema. En el caso del proyecto, solo se cuenta con uno.

[57] https://docs.celeryq.dev/en/stable/

5. Backend: Redis también actúa como backend de resultados, almacenando los resultados de las tareas ejecutadas por el worker.

6. Lectura de Resultados: FastAPI puede leer los resultados de las tareas almacenadas en Redis y proporcionar la respuesta final al usuario.

El diagrama proporcionado muestra esta arquitectura de manera visual:

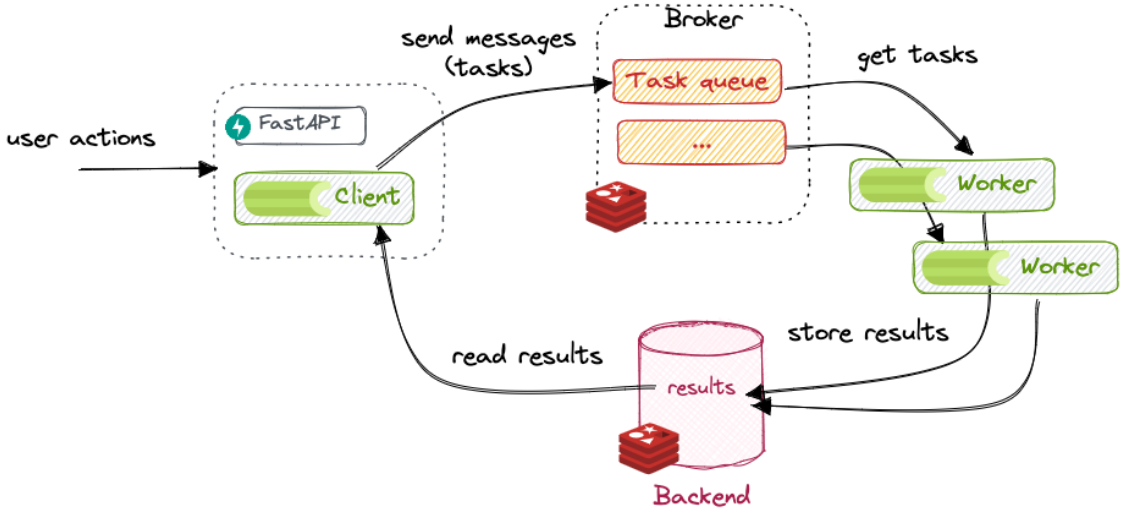

Figura 16: Arquitectura Celery

6.3.4. Elasticsearch

El contenedor elasticsearch se basa en la imagen oficial de Elasticsearch[58], una potente plataforma de búsqueda y análisis de código abierto. Elasticsearch es una evolución de Apache Lucene[59], que amplía las capacidades de este motor de búsqueda para proporcionar una solución escalable y distribuida. Al ser una base de datos NoSQL, Elasticsearch almacena y recupera datos de manera eficiente utilizando una estructura de documentos en lugar de las tradicionales tablas relacionales.

Los datos almacenados pueden ser consultados utilizando las poderosas capacidades de búsqueda y agregación de Elasticsearch. Esto es particularmente útil para analizar relaciones entre diferentes IoCs[60] y detectar patrones de amenazas. El contenedor elasticsearch sirve como el núcleo del sistema de búsqueda y almacenamiento de datos de la aplicación. Su principal función es almacenar y gestionar datos estructurados procesados de los reportes con la ontología definida, facilitando su búsqueda y análisis.

El uso de volúmenes persistentes asegura que los datos indexados no se pierdan entre reinicios del contenedor, manteniendo la integridad y disponibilidad de la información.

[58]https://www.elastic.co/es/elasticsearch
[59]https://lucene.apache.org/
[60]Indicator of Compromise

6.3.5. Kibana

El contenedor kibana está basado en la imagen oficial de Kibana, una herramienta de visualización de datos y análisis diseñada para trabajar con Elasticsearch. Kibana permite explorar y visualizar datos almacenados en Elasticsearch mediante dashboards interactivos, gráficos y otras representaciones visuales.

7. Aplicación

En esta sección se presentará una descripción detallada de las principales pantallas que conforman la aplicación web desarrollada con FastAPI. Esta aplicación ha sido diseñada para ofrecer una experiencia de usuario intuitiva y eficiente, permitiendo a los usuarios acceder y manejar datos de manera segura y organizada. A continuación, se proporcionará una breve explicación de las funcionalidades y el diseño de cada una de las pantallas principales, comenzando con la página de inicio de sesión y el dashboard principal.

7.1. Login

En esta imagen se observa la página de inicio de sesión de la aplicación web. Esta interfaz permite a los usuarios autenticarse proporcionando un nombre de usuario y una contraseña. El diseño es simple y directo, con campos de texto para ingresar las credenciales y un botón para enviar la información. Este proceso es gestionado por el backend implementado con FastAPI, que maneja la autenticación y autorización de los usuarios, asegurando que solo personas autorizadas puedan acceder a la aplicación.

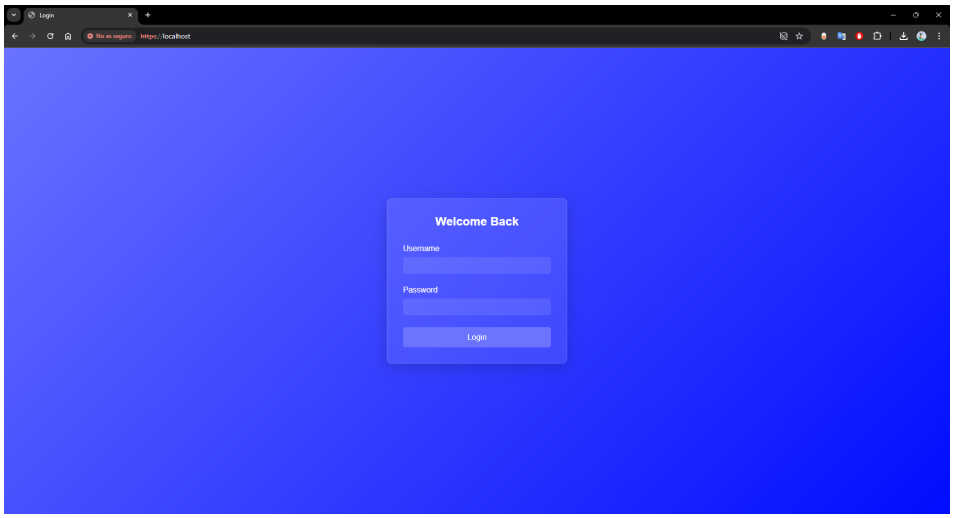

Figura 17: Login app

7.2. Dashboard

Se muestra el dashboard principal de la aplicación, que es accesible tras la autenticación exitosa del usuario. En el panel lateral izquierdo (sidebar) se pueden observar varias secciones disponibles: Inicio, Analizar, Reportes, Kibana y Logout. Este diseño proporciona una navegación intuitiva para los usuarios, permitiéndoles acceder fácilmente a diferentes funcionalidades de la aplicación. La implementación con FastAPI facilita la gestión de estos diferentes endpoints, proporcionando respuestas rápidas y eficientes a las solicitudes del usuario.

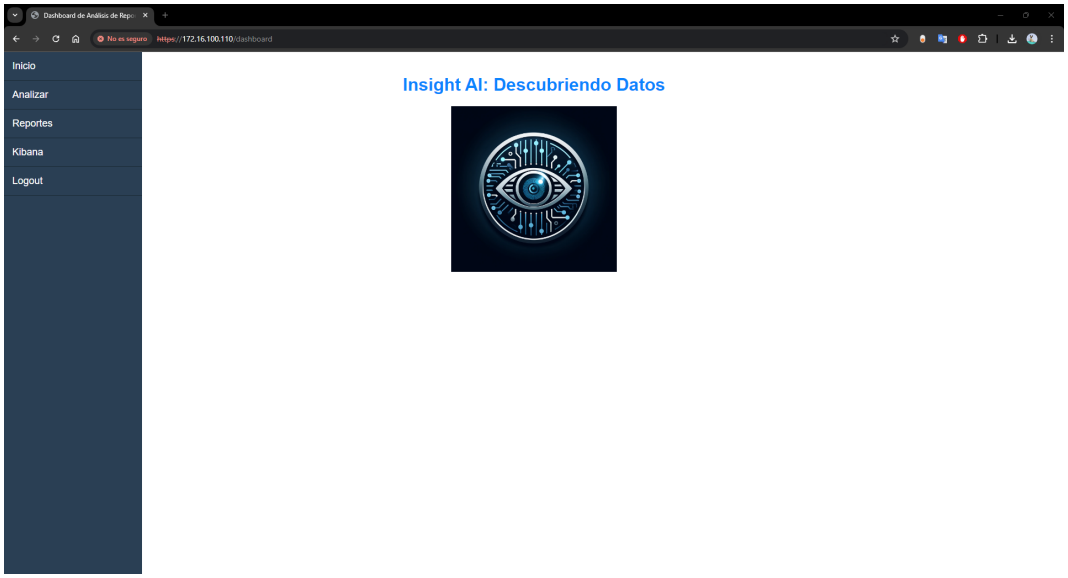

Figura 18: Dashboard app

7.3. Analizar

La sección ”Analizar” de la aplicación, permite a los usuarios añadir la URL de un reporte de ciberseguridad para su análisis. El diseño de la interfaz es sencillo, contando con un campo de texto donde el usuario puede ingresar la dirección URL del reporte y un botón para enviar esta información. Al hacer clic en ”Enviar URL”, la aplicación procesa el enlace y extrae los datos necesarios para su posterior análisis. En esta fase ocurre lo explicado de la limpieza de los datos no estructurados en la sección 4.3

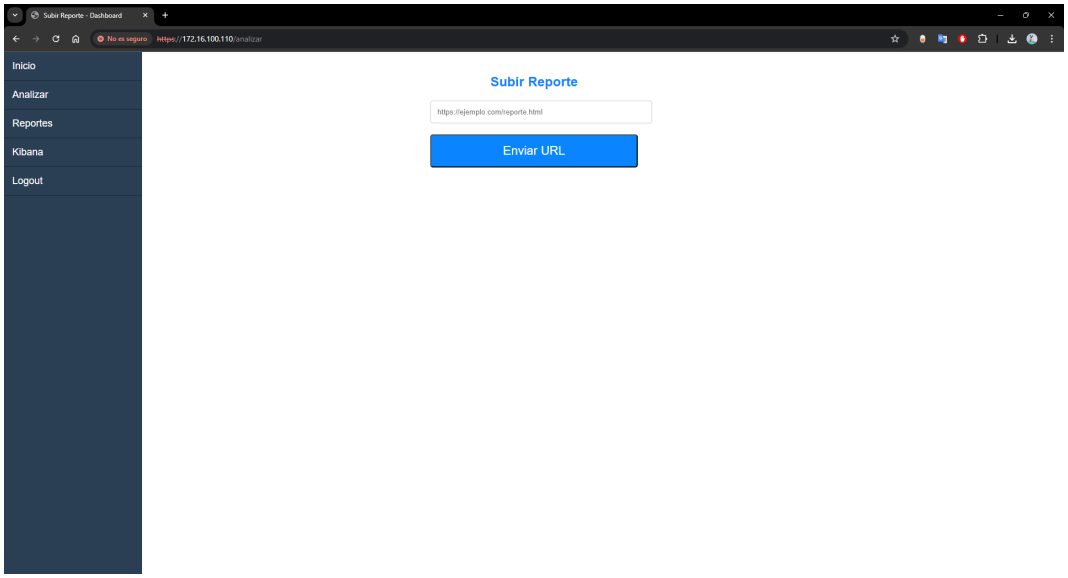

Figura 19: Analizar app

7.4. Reportes

En esta pantalla se muestra el estado del análisis de los reportes de ciberseguridad previamente enviados a través de la sección ”Analizar”. Esta pantalla presenta una lista de reportes con sus correspondientes estados de procesamiento, destacando en amarillo aquellos que están en estado ”pending” (pendiente) y en verde aquellos cuyo procesamiento ha sido ”completed” (completado).

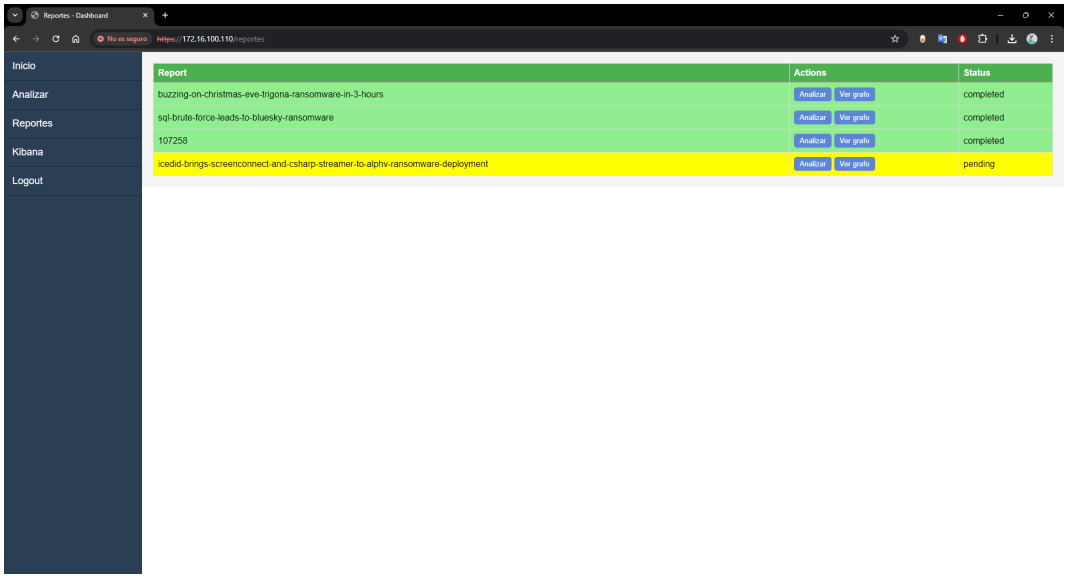

Figura 20: Reportes app

Cada reporte incluye varias opciones de acción:

- **Analizar**: Este botón permite al usuario acceder a las entidades extraídas del reporte. La explicación detallada de esta funcionalidad se presentará en la siguiente sección.
- **Ver grafo**: Este botón proporciona una visualización gráfica de las conexiones entre las entidades identificadas en el reporte. La descripción detallada de esta funcionalidad se presentará posteriormente.

El proceso de análisis se lleva a cabo utilizando tareas de Celery, que permiten el procesamiento asíncrono de los reportes mediante el modelo entrenado para extraer entidades relevantes del contenido de los reportes de ciberseguridad. Esta arquitectura asegura que los reportes sean procesados de manera eficiente y en paralelo, mejorando la capacidad de la aplicación para manejar múltiples solicitudes de análisis simultáneamente.

7.5. Analizar Reporte

En esta imagen se presenta la pantalla de análisis de entidades, accesible tras pulsar el botón *Analizar* en un reporte de la sección "Reportes". Una vez que el estado del reporte está en verde (completado), el usuario puede visualizar cada una de las secciones de texto correspondientes a las tácticas llevadas a cabo durante el ciberataque.

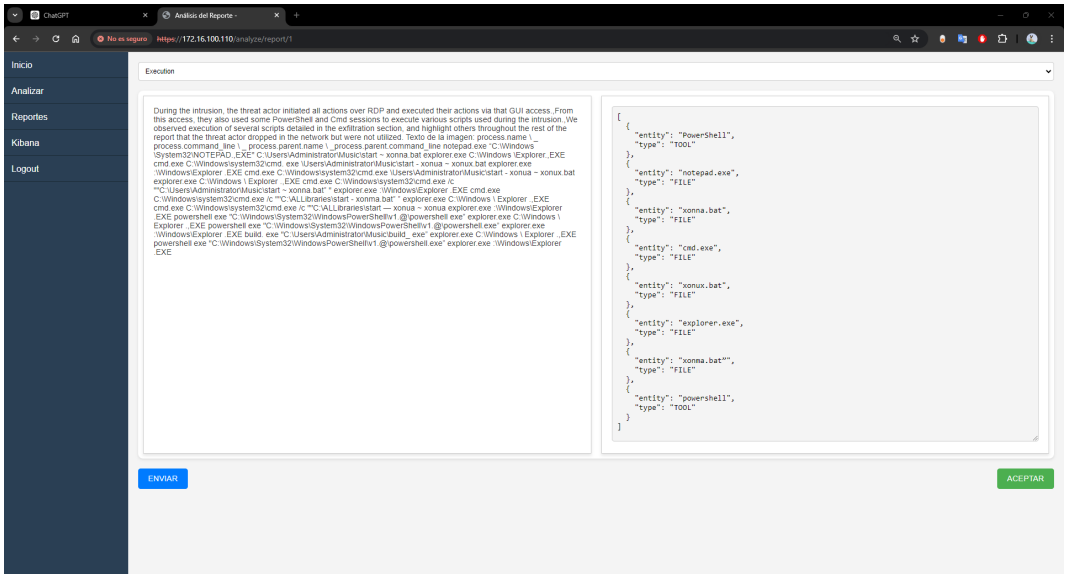

Figura 21: Entidades analizar app

- Texto del Reporte: En el panel izquierdo se muestra el texto del reporte, detallando las acciones del actor de la amenaza durante la intrusión.
- Entidades Encontradas: En el panel derecho se listan las entidades encontradas por el modelo entrenado. Estas entidades están clasificadas por tipo, como archivos, herramientas, y otros elementos relevantes.

El analista tiene la posibilidad de verificar la exactitud de las entidades identificadas utilizando el botón "ACEPTAR" en la parte inferior derecha. Además, puede editar, eliminar o añadir nuevas entidades según sea necesario. Esta funcionalidad asegura que la información extraída sea precisa y completa.

Una vez que el analista ha revisado y aceptado todas las entidades, estas son enviadas a una tarea de Celery que las procesa para transformarlas en una ontología compatible con Elasticsearch. Además, se genera un grafo que visualiza las conexiones entre las entidades extraídas, proporcionando una representación gráfica del análisis.

7.6. Grafo

Las imágenes proporcionadas corresponden a el grafo de entidades generado tras el procesamiento de datos de entidades aceptadas por un analista. Este grafo representa las relaciones y conexiones entre diferentes entidades y eventos dentro del reporte. El grafo mostrado se corresponde al análisis del reporte: *Buzzing on christmas eve trigona ransomware in 3 hours*[61]. Para acceder a ello se debe ir a la sección de "Reportes" y pulsar el botón "Ver grafo" correspondiente al reporte deseado.

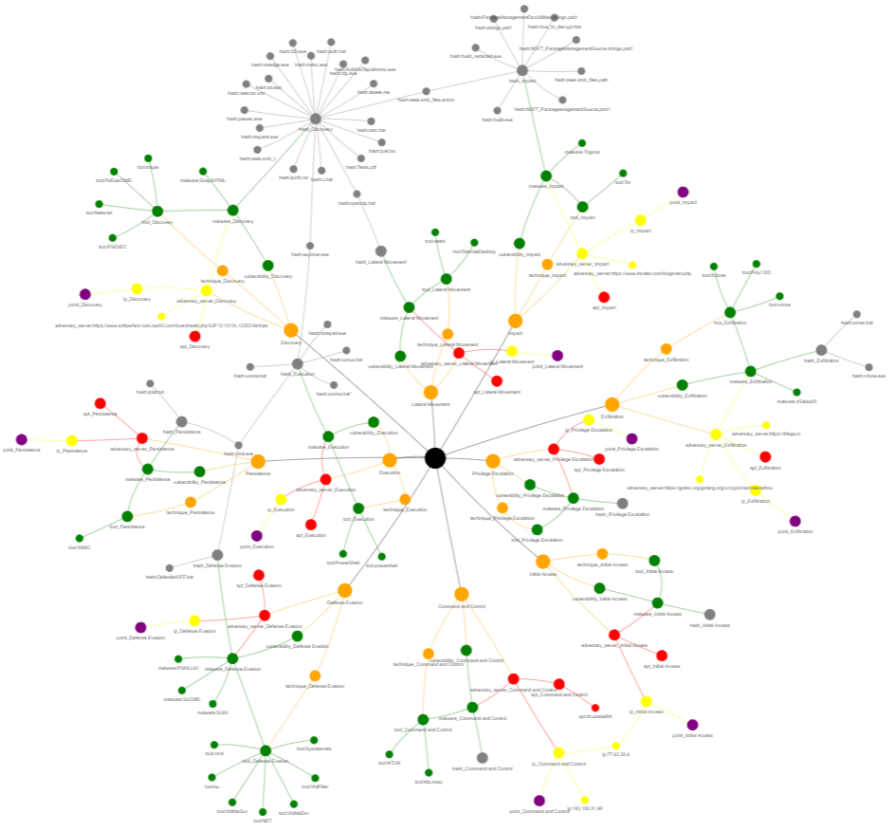

Figura 22: Grafo app

[61]https://thedfirreport.com/2024/01/29/buzzing-on-christmas-eve-trigona-ransomware-in-3-hours/

Este grafo global integra las diferentes fases del ciclo de vida del ataque cibernético, desde el acceso inicial hasta el impacto final, pasando por todas las técnicas.

Las conexiones multidimensionales entre los nodos resaltan la complejidad de los ataques cibernéticos modernos, donde múltiples técnicas y herramientas pueden ser utilizadas en diferentes combinaciones para lograr el objetivo del atacante.

La representación holística del grafo global permite al analista tener una visión completa del ecosistema de amenazas, facilitando la identificación de patrones y la comprensión de las estrategias de ataque utilizadas.

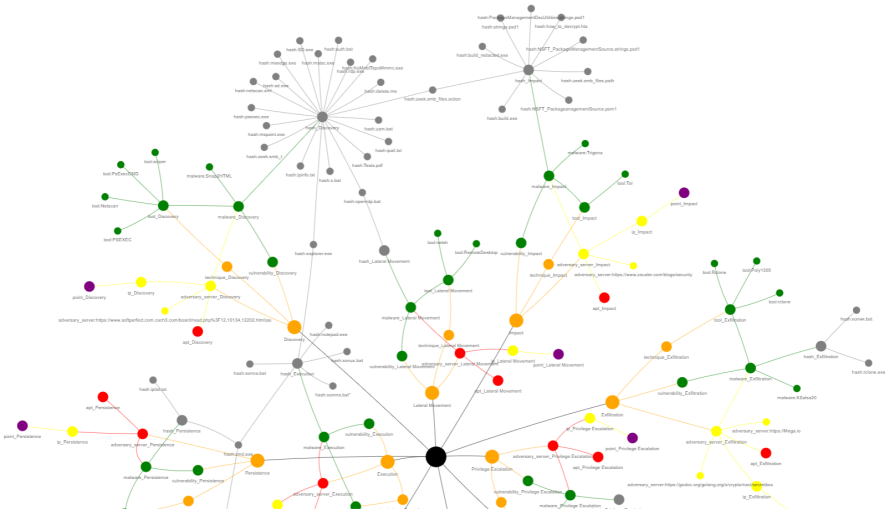

Figura 23: Grafo norte app

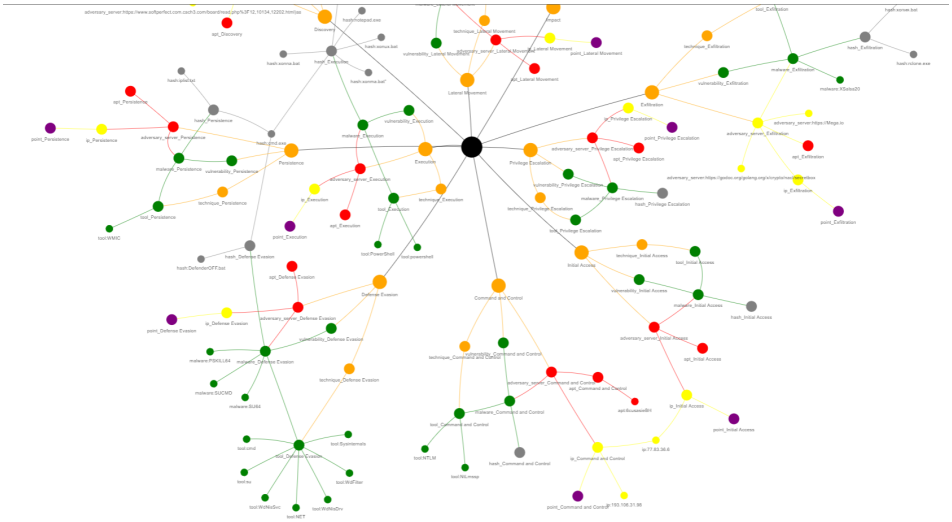

Figura 24: Grafo sur app

7.7. Búsqueda elástica

Las capturas de pantalla proporcionadas muestran diferentes índices creados en Kibana como resultado del procesamiento de datos mediante la ontología. Esta ontología permite estructurar y categorizar las entidades de manera eficiente. Kibana, facilita el acceso y análisis de estos datos categorizados.

En el índice de malware, se almacenan documentos que identifican y categorizan diferentes tipos de malware.

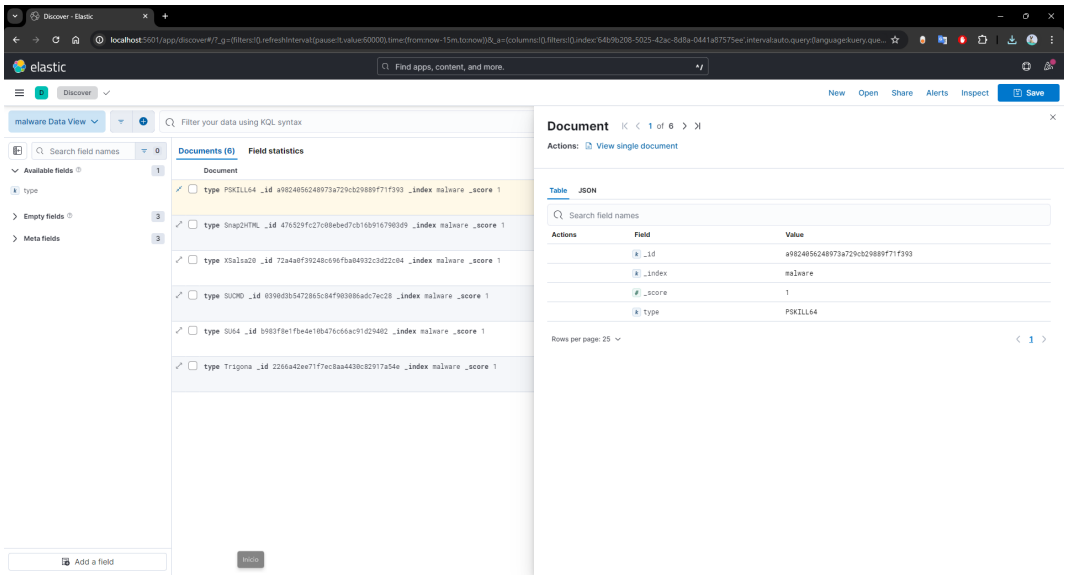

Figura 25: Kibana malware

El índice de direcciones IP contiene registros de direcciones IP relevantes. La estructuración de esta información permite a los analistas gestionar de manera efectiva las direcciones IP involucradas en posibles incidentes de seguridad.

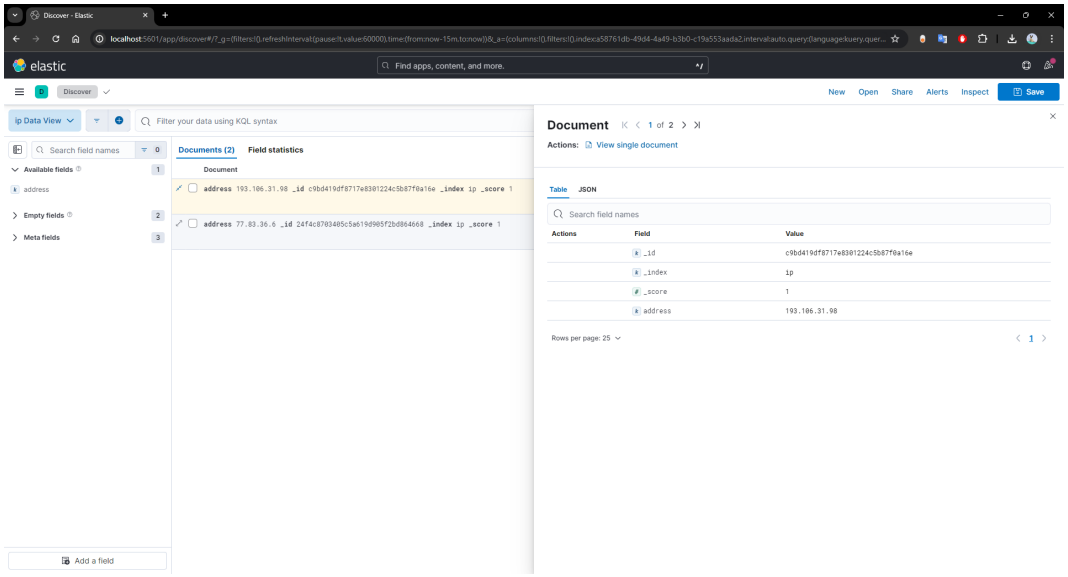

Figura 26: Kibana ips

El índice de hashes recopila información sobre los hashes de archivos significativos en los análisis de ciberseguridad.

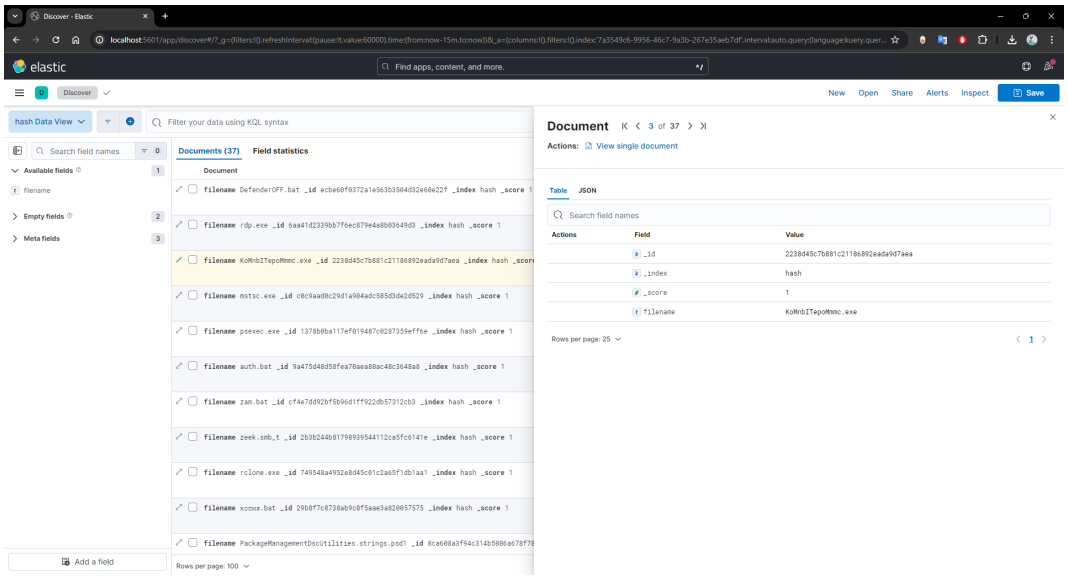

Figura 27: Kibana hash

8. Evaluación del Sistema

En esta sección se presenta una prueba de concepto realizada con expertos del sector, cuyo objetivo fue evaluar la eficacia del Sistema de inteligencia de amenazas en comparación con los métodos tradicionales. Para ello, se analizaron una serie de reportes tanto con la ayuda del sistema de inteligencia de amenazas como sin él, midiendo los tiempos necesarios para realizar estos análisis.

La prueba se centró en dos aspectos clave: el tiempo de análisis de los reportes por parte de analistas junior y senior, y el tiempo requerido para integrar indicadores de compromiso en las listas negras dentro de la cadena de valor. Los resultados obtenidos permitieron verificar que el uso del sistema de inteligencia de amenazas no solo reduce significativamente los tiempos de análisis, sino que también mejora la eficiencia en la integración de amenazas en las listas negras, demostrando así su utilidad y eficacia en entornos operacionales.

En las tablas siguientes se presenta una comparación detallada de los tiempos de análisis y de integración de IoCs con y sin el uso del sistema de inteligencia de amenazas, proporcionando una visión clara de los beneficios que ofrece esta herramienta tecnológica.

Rol del Analista	**Con Sistema (minutos)**	**Sin Sistema (minutos)**
Junior	30	120
Senior	20	90

Tabla 16: Comparativa de tiempos de análisis de un reporte

Proceso	**Con Sistema (minutos)**	**Sin Sistema (minutos)**
Identificación de IoCs	10	60
Validación de IoCs	15	90
Implementación en sistemas	20	120

Tabla 17: Comparativa de integración de IoCs en la cadena de valor

9. Conclusiones

En el presente Trabajo de Fin de Máster, se ha desarrollado un sistema de inteligencia de amenazas avanzado, cuyo objetivo principal es mejorar la eficiencia y efectividad en la detección y análisis de amenazas cibernéticas. El sistema, basado en ontologías y técnicas de procesamiento de lenguaje natural (NLP), representa un avance significativo en la integración de tecnología de inteligencia artificial en el campo de la ciberseguridad.

Se realizó un mapping sistemático de la literatura para identificar y analizar las tecnologías y métodos más relevantes aplicados en la ciberseguridad, específicamente

en la inteligencia de amenazas. Este proceso incluyó la identificación, selección, elegibilidad y extracción de resultados de estudios relevantes, lo que permitió formular respuestas a preguntas de mapeo específicas sobre el uso de ontologías y técnicas de procesamiento de lenguaje natural en la ciberseguridad.

Se desarrolló una ontología especializada basada en el marco MITRE ATT&CK, destinada a estructurar y categorizar la información relacionada con las amenazas cibernéticas. Esta ontología facilitó la organización de datos y mejoró la precisión en la identificación de amenazas. Además, se implementaron preguntas de competencia para asegurar la cobertura adecuada del dominio y se realizó una revisión continua para garantizar la calidad y relevancia de la ontología.

Se exploraron y compararon diversos modelos de lenguaje, incluyendo BERT, ALBERT, GPT-2, SecRoBERTa, T5 y deBERTa, todos adaptados para la tarea de reconocimiento de entidades nombradas en ciberseguridad. La elección del modelo final se basó en su accesibilidad, requisitos computacionales y respaldo científico. Se llevaron a cabo entrenamientos y ajustes finos para optimizar el rendimiento de los modelos seleccionados, demostrando mejoras significativas en la extracción y clasificación de entidades relacionadas con amenazas.

La plataforma desarrollada integra una ontología con técnicas avanzadas de NLP y un motor de búsqueda elástica. Se implementaron componentes como Docker, Elasticsearch y Kibana para asegurar una arquitectura robusta y eficiente. La plataforma permite el análisis de reportes, la extracción y clasificación de datos, y la visualización de resultados en gráficos interactivos, facilitando la identificación y gestión de amenazas cibernéticas.

Se realizó una evaluación del sistema mediante una prueba de concepto con expertos del sector. Los resultados mostraron una reducción significativa en los tiempos de análisis y en la integración de indicadores de compromiso (IoCs) en comparación con métodos tradicionales. Los analistas, tanto junior como senior, lograron realizar análisis más rápidos y precisos, destacando la utilidad y eficacia del sistema en entornos operacionales.

El desarrollo de esta plataforma de inteligencia de amenazas ha demostrado la efectividad de la integración de ontologías y técnicas de NLP en la ciberseguridad. La combinación de estas tecnologías ha permitido mejorar la precisión y rapidez en la identificación de amenazas, proporcionando una herramienta valiosa para los Centros de Operaciones de Seguridad (SOC). Este trabajo no solo contribuye al avance del conocimiento en el campo de la ciberseguridad, sino que también ofrece una solución práctica para enfrentar los desafíos crecientes en la protección de activos digitales.

A. Apéndice A - Documento de Especificación de Requisitos Ontológicos - ORSD

Elemento	Contenido
Propósito	Esta ontología tiene como propósito modelar y representar de manera estructurada los conocimientos y datos relevantes en el dominio de la ciberseguridad, basándose en el marco MITRE ATT&CK para la identificación y clasificación de tácticas, técnicas y procedimientos (TTPs) utilizados por adversarios.
Alcance	La ontología abarca la representación de adversarios, sus técnicas, las herramientas utilizadas, las vulnerabilidades explotadas, los tipos de archivos involucrados, la geolocalización de los ataques y los afectados por el ataque.
Nivel de Formalidad	La ontología se desarrollará con un alto nivel de formalidad utilizando OWL (Web Ontology Language), permitiendo así el uso de lógica descriptiva para inferencias complejas y validaciones.
Usuarios Previstos	Desarrolladores de software en seguridad, analistas de seguridad, investigadores en ciberseguridad y sistemas automatizados de respuesta a incidentes.
Usos Previstos	Utilizada para mejorar la detección de amenazas, análisis forense, educación en ciberseguridad y para mejorar los sistemas de respuesta automática a incidentes.
Grupos de Preguntas de Competencia	Incluye preguntas como: ¿Qué técnicas utilizó el adversario?, ¿Qué vulnerabilidades fueron explotadas?, ¿Cuáles herramientas fueron empleadas en el ataque? Estas preguntas ayudan a comprender la metodología del ataque para mejorar las estrategias de defensa y respuesta.
Pre-Glosario de Términos	Términos clave como "Técnica", "Táctica", "Vulnerabilidad", "Herramienta", "APT", "Archivo", "Geolocalización", "Servidor". Cada término incluirá una definición y su relevancia dentro del marco de ciberseguridad.

Tabla 18: Documento de Especificación de Requisitos de la Ontología de Ciberseguridad

B. Apéndice B - docker-compose.yml

```yaml
services:
  web:
    build: ./web
    hostname: web
    volumes:
      - ./web:/app
    ports:
      - "443:443"
    networks:
      - es-net
    depends_on:
      - worker

  redis:
    image: redis:latest
    hostname: redis
    ports:
      - "6379:6379"
    networks:
      - es-net

  worker:
    build: ./celery
    hostname: worker
    volumes:
      - ./celery:/celery
    networks:
      - es-net
    depends_on:
      - elasticsearch

  elasticsearch:
    image: elasticsearch:8.13.4
    environment:
      - discovery.type=single-node
      - ES_JAVA_OPTS=-Xms2g -Xmx2g
      - xpack.security.enabled=false
    volumes:
      - es_data:/usr/share/elasticsearch/data
    ports:
      - "9200:9200"
    hostname: elasticsearch
    depends_on:
```

```
      - redis
    networks:
      - es-net

  kibana:
    image: kibana:8.13.4
    environment:
      - "ELASTICSEARCH_URL=http://elasticsearch:9200"
    ports:
      - "5601:5601"
    hostname: kibana
    depends_on:
      - elasticsearch
    networks:
      - es-net

networks:
  es-net:
    driver: bridge

volumes:
  es_data:
    driver: local
```

C. Apéndice C - Dockerfiles

```
# Utilizar una imagen base oficial de Python
FROM python:slim

# Establecer el directorio de trabajo en el contenedor
WORKDIR /app

# Copiar el archivo de requisitos primero para aprovechar la cach
    ↪  de capas de Docker
COPY requirements.txt /app/

# Instalar las dependencias de Python
RUN pip install --no-cache-dir -r requirements.txt
RUN apt-get update && apt-get install tesseract-ocr -y

# Copiar el resto de la aplicaci n al directorio de trabajo
COPY . /app
# Exponer el puerto que FastAPI utilizar
EXPOSE 443

# Comando para ejecutar la aplicaci n usando uvicorn
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port",
    ↪ "443", "--ssl-keyfile", "/app/certs/key.pem", "--ssl-certfile
    ↪ ", "/app/certs/cert.pem"]
```

Listing 14: Contenedor Web

```
FROM python:slim

WORKDIR /celery

RUN mkdir -p /images
COPY requirements.txt .
RUN pip install -r requirements.txt
RUN apt-get update && apt-get install tesseract-ocr -y

COPY . .

CMD ["celery", "-A", "app.celery_worker.main", "worker", "--
    ↪ loglevel=INFO"]
```

Listing 15: Contenedor Celery Worker

D. Apéndice D - requirements.txt

```
fastapi
uvicorn
python-multipart
jinja2
passlib[bcrypt]
pydantic
pyjwt
aiofiles
celery
celery[redis]
SQLAlchemy
html2text
requests
pillow
pytesseract
torch
transformers
numpy
elasticsearch
pyvis
networkx
```

Listing 16: Librerías Python web

```
celery[redis]
html2text
requests
pillow
pytesseract
torch
transformers
numpy
elasticsearch
pyvis
networkx
```

Listing 17: Librerías Celery Worker

E. Apéndice E - Entrenamiento modelo deBERTa

```python
import os
import torch
from datasets import load_dataset
from transformers import (
    AutoModelForTokenClassification,
    AutoTokenizer,
    BitsAndBytesConfig,
    TrainingArguments,
    pipeline,
    DataCollatorForTokenClassification,
    AdamW,
    Trainer,
    TrainerCallback,
)
from tqdm.auto import tqdm
from datasets import Dataset, DatasetDict
import json
from peft import get_peft_model, LoraConfig, TaskType
from torch.utils.data import DataLoader
import evaluate
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Listing 18: Python imports

```python
base_model = "microsoft/deberta-v3-base"
label2id = {"O": 0, "S-MAL": 1, "S-URL": 2, "B-TOOL": 3, "E-TOOL":
    ↪ 4, "S-TOOL": 5, "B-FILE": 6, "I-FILE": 7, "E-FILE": 8, "S-
    ↪ VULN": 9, "S-CVE": 10, "S-DOM": 11, "S-FILE": 12, "S-SECTEAM"
    ↪ : 13, "B-URL": 14, "I-URL": 15, "E-URL": 16, "S-APT": 17, "S-
    ↪ OS": 18, "S-IP": 19, "B-OS": 20, "I-OS": 21, "E-OS": 22, "S-
    ↪ HASH": 23, "B-APT": 24, "I-APT": 25, "E-APT": 26, "B-MAL":
    ↪ 27, "E-MAL": 28, "B-OPT": 29, "E-OPT": 30, "S-OPT": 31, "I-
    ↪ OPT": 32, "S-EMAIL": 33, "B-SECTEAM": 34, "I-SECTEAM": 35, "E
    ↪ -SECTEAM": 36, "I-MAL": 37, "I-VULN": 38, "E-VULN": 39, "B-
    ↪ VULN": 40, "B-TIME": 41, "I-TIME": 42, "E-TIME": 43, "S-LOC":
    ↪  44, "B-LOC": 45, "E-LOC": 46, "I-TOOL": 47, "S-TIME": 48, "B
    ↪ -IP": 49, "E-IP": 50, "B-HASH": 51, "E-HASH": 52, "I-LOC":
    ↪ 53, "B-EMAIL": 54, "E-EMAIL": 55}
label_list = list(label2id.keys())
id2label = {v: k for k, v in label2id.items()}
cache_dir = "/home/javi/LLM_NER/DevBERTA_save"
tokenizer = AutoTokenizer.from_pretrained(base_model,cache_dir=
    ↪ cache_dir)
model = AutoModelForTokenClassification.from_pretrained(base_model,
    ↪  cache_dir=cache_dir,num_labels=len(label2id), id2label=
    ↪ id2label, label2id=label2id)
print(model)
```

Listing 19: Python carga modelo

```
peft_config = LoraConfig(
                         task_type=TaskType.TOKEN_CLS,
                         inference_mode=False,
                         r=16,
                         lora_alpha=16,
                         lora_dropout=0.15,
                         target_modules=["key_proj", "query_proj","
    ↪ value_proj"],
                         bias="all",
                         modules_to_save=["classifier"],
                         base_model_name_or_path=cache_dir
                        )
model = get_peft_model(model, peft_config)
model.print_trainable_parameters()
```

Listing 20: Python PEFT

```
def load_data(filename):
    texts, tags = [], []
    with open(filename, 'r', encoding='utf-8') as file:
        for line in file:
            entry = json.loads(line)
            texts.append(entry['tokens'])
            tags.append(entry['ner_tags'])
    return texts, tags

file_path = './datasets/trainS_new.jsonl'
data = pd.read_json(file_path, lines=True)
train_dataset = Dataset.from_pandas(data)
data_collator = DataCollatorForTokenClassification(tokenizer=
    ↪ tokenizer)
file_path = './datasets/testS_new.jsonl'
data = pd.read_json(file_path, lines=True)
test_dataset = Dataset.from_pandas(data)

def preprocess_data(examples):
    tokenized_inputs = tokenizer(examples['tokens'],
    ↪ is_split_into_words=True,add_special_tokens=True,padding='
    ↪ max_length', truncation=True, max_length=WINDOW_LENGTH,
    ↪ stride=STRIDE)
    labels = []
    for i, label in enumerate(examples['ner_tags']):
        word_ids = tokenized_inputs.word_ids(batch_index=i)
        label_ids = [-100 if id is None else label[id] for id in
    ↪ word_ids]
        labels.append(label_ids)
    tokenized_inputs['labels'] = labels
    return tokenized_inputs

train_dataset = train_dataset.map(preprocess_data, batched=True)
test_dataset = test_dataset.map(preprocess_data, batched=True)
```

Listing 21: Python carga datasets

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu
    ↪ ")
print(device)
model.to(device)
```

Listing 22: Python carga modelo en gráfica CUDA

```
seqeval = evaluate.load("seqeval")
from seqeval.metrics import accuracy_score, f1_score,
    ↪ precision_score, recall_score
def compute_metrics(p):
    predictions, labels = p
    predictions = np.argmax(predictions, axis=2)

    true_predictions = [
        [label_list[p] for (p, l) in zip(prediction, label) if l !=
    ↪  -100]
        for prediction, label in zip(predictions, labels)
    ]
    true_labels = [
        [label_list[l] for (p, l) in zip(prediction, label) if l !=
    ↪  -100]
        for prediction, label in zip(predictions, labels)
    ]

    results = seqeval.compute(predictions=true_predictions,
    ↪ references=true_labels)
    return {
        "precision": results["overall_precision"],
        "recall": results["overall_recall"],
        "f1": results["overall_f1"],
        "accuracy": results["overall_accuracy"],
    }
```

Listing 23: Python evaluación modelo

```
training_args = TrainingArguments(
    output_dir="/home/javi/LLM_NER/DeBERTA_fine",
    learning_rate=5e-4,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=16,
    num_train_epochs=10,
    weight_decay=0.01,
    evaluation_strategy="epoch",
    fp16=True,
    save_strategy="epoch",
    load_best_model_at_end=True,
    push_to_hub=False,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
```

```
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics
)

train_result = trainer.train()
```

Listing 24: Python entrenamiento

```
save_path = "/home/javi/LLM_NER/DeBERTA_save_fine"
trainer.save_model(save_path)
tokenizer.save_pretrained(save_path)
tokenizer = AutoTokenizer.from_pretrained(save_path)
from peft import PeftModel, PeftConfig
peft_config = PeftConfig.from_pretrained(save_path)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu
    ↪ ")
model = AutoModelForTokenClassification.from_pretrained(peft_config
    ↪ .base_model_name_or_path, num_labels=len(label2id), id2label=
    ↪ id2label, label2id=label2id,cache_dir=save_path).to(device)
model = PeftModel.from_pretrained(model, save_path)
model = model.merge_and_unload()
save_compile = "/home/javi/LLM_NER/DeBERTA_merge"
model.save_pretrained(save_compile)
tokenizer.save_pretrained(save_compile)
```

Listing 25: Python guardado modelo

F. Apéndice F - Prueba del modelo

```
save_compile = "/home/manu/javi/LLM_NER/DeBERTA_merge"
model = AutoModelForTokenClassification.from_pretrained(
    ↪ save_compile, num_labels=len(label2id), id2label=id2label,
    ↪ label2id=label2id)
tokenizer = AutoTokenizer.from_pretrained(save_compile)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu
    ↪ ")
model = model.to(device)
token_classifier = pipeline("token-classification", model=model,
    ↪ tokenizer=tokenizer,aggregation_strategy='max')
sentence4 = "hash 8
    ↪ dde190b1a694f20a9b74fdef55a34d24402fc80ad642ed7eb55e7dfd65c4293
    ↪  from malware agent tesla was download from http://www.
    ↪ malware.com and http://113.9.132.44:52848/bin.sh"
tokens = token_classifier(sentence4)
for token in tokens:
    print(token)
```

Listing 26: Python prueba modelo DeBERTa

```
{'entity_group': 'HASH', 'score': 0.9999105,
'word':'8dde190b1a694f20a9b74fdef55a34d24402fc80ad642ed7eb55e7dfd65c4293',
'start': 4, 'end': 69}
{'entity_group': 'MAL', 'score': 0.90634716, 'word': 'agent', 'start': 82, 'end': 88}
{'entity_group': 'MAL', 'score': 0.97336245, 'word': 'tesla', 'start': 88, 'end': 94}
{'entity_group': 'URL', 'score': 0.99652267, 'word': 'http://www.malware.com', 'start': 112, 'end': 135}
{'entity_group': 'URL', 'score': 0.99840194, 'word': 'http://113.9.132.44:52848/bin.sh', 'start': 139, 'end': 172}
```

Referencias

[1] Ehsan Aghaei, Xi Niu, Waseem Shadid, and Ehab Al-Shaer. Securebert: A domain-specific language model for cybersecurity, 2022. [Citado en pág. 53.]

[2] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. FLAIR: An easy-to-use framework for state-of-the-art NLP. In Waleed Ammar, Annie Louis, and Nasrin Mostafazadeh, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. [Citado en pág. 35.]

[3] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In Emily M. Bender, Leon Derczynski, and Pierre Isabelle, editors, *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. [Citado en pág. 35.]

[4] M. Ashburner, C.A. Ball, Judith Blake, David Botstein, Heather Butler, and J. Cherry. Gene ontology: Tool for the unification of biology. *The Gene Ontology Consortium. Nat Genet*, 25:25–29, 01 2000. [Citado en pág. 16.]

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. [Citado en pág. 32.]

[6] Savelie Cornegruta, Robert Bakewell, Samuel Withey, and Giovanni Montana. Modelling radiological language with bidirectional long short-term memory networks, 2016. [Citado en pág. 12.]

[7] Moumita Das Purba, Bill Chu, and Ehab Al-Shaer. From word embedding to cyber-phrase embedding: Comparison of processing cybersecurity texts. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6, Nov 2020. [Citado en pág. 13.]

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. [Citado en págs. 11, 33, 34 y 48.]

[9] Francisco García-Peñalvo. Developing robust state-of-the-art reports: Systematic literature reviews. *Education in the Knowledge Society (EKS)*, 23:e28600, 04 2022. [Citado en pág. 3.]

[10] Asuncion Gomez-Perez and Oscar Corcho. Ontology languages for the semantic web. *Intelligent Systems, IEEE*, 17:54 – 60, 02 2002. [Citado en pág. 16.]

[11] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention, 2021. [Citado en pág. 55.]

[12] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998. [Citado en pág. 34.]

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. [Citado en págs. 12 y 32.]

[14] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. [Citado en págs. 45 y 46.]

[15] Chiao-Cheng Huang, Pei-Yu Huang, Ying-Ren Kuo, Guo-Wei Wong, Yi-Ting Huang, Yeali S. Sun, and Meng Chang Chen. Building cybersecurity ontology for understanding and reasoning adversary tactics and techniques. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 4266–4274, Dec 2022. [Citado en pág. 11.]

[16] Peter E. Kaloroumakis and Michael J. Smith. Toward a knowledge graph of cybersecurity countermeasures. `https://d3fend.mitre.org/resources/D3FEND.pdf`, 2023. [Citado en pág. 18.]

[17] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition, 2016. [Citado en pág. 33.]

[18] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020. [Citado en pág. 50.]

[19] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6, 01 2014. [Citado en pág. 18.]

[20] Lingzi Li, Cheng Huang, and Junren Chen. Automated discovery and mapping attck tactics and techniques for unstructured cyber threat intelligence. *Computers Security*, 140:103815, 2024. [Citado en págs. 11 y 13.]

[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. [Citado en pág. 35.]

[22] Edward Loper and Steven Bird. Nltk: The natural language toolkit, 2002. [Citado en pág. 35.]

[23] Simone Magnolini, Valerio Piccioni, Vevake Balaraman, Marco Guerini, and Bernardo Magnini. How to use gazetteers for entity recognition with neural models. In Luis Espinosa-Anke, Thierry Declerck, Dagmar Gromann, Jose Camacho-Collados, and Mohammad Taher Pilehvar, editors, *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 40–49, Macau, China, August 2019. Association for Computational Linguistics. [Citado en pág. 36.]

[24] Fabian Neuhaus. What is an ontology? *CoRR*, abs/1810.09171, 2018. [Citado en pág. 15.]

[25] Matthew J. Page, Joanne E. McKenzie, Patrick M. Bossuyt, Isabelle Boutron, Tammy C. Hoffmann, Cynthia D. Mulrow, Larissa Shamseer, Jennifer M. Tetzlaff, Elie A. Akl, Sue E. Brennan, Roger Chou, Julie Glanville, Jeremy M. Grimshaw, Asbjørn Hróbjartsson, Manoj M. Lalu, Tianjing Li, Elizabeth W. Loder, Evan Mayo-Wilson, Steve McDonald, Luke A. McGuinness, Lesley A. Stewart, James Thomas, Andrea C. Tricco, Vivian A. Welch, Penny Whiting, David Moher, Juan José Yepes-Nuñez, Gerard Urrútia, Marta Romero-García, and Sergio Alonso-Fernández. Declaración prisma 2020: una guía actualizada para la publicación de revisiones sistemáticas. *Revista Española de Cardiología*, 74(9):790–799, 2021. [Citado en pág. 8.]

[26] María Poveda-Villalón, Alba Fernández-Izquierdo, Mariano Fernández-López, and Raúl García-Castro. Lot: An industrial oriented ontology engineering framework. *Engineering Applications of Artificial Intelligence*, 111:104755, 2022. [Citado en págs. 18 y 27.]

[27] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A python natural language processing toolkit for many human languages, 2020. [Citado en pág. 35.]

[28] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018. [Citado en págs. 33 y 34.]

[29] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. OpenAI, San Francisco, California, United States. [Citado en pág. 51.]

[30] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. [Citado en pág. 54.]

[31] Lance A. Ramshaw and Mitchell P. Marcus. Text chunking using transformation-based learning, 1995. [Citado en pág. 36.]

[32] Nanda Rani, Bikash Saha, Vikas Maurya, and Sandeep Kumar Shukla. Ttphunter: Automated extraction of actionable intelligence as ttps from narrative threat reports. In *Proceedings of the 2023 Australasian Computer Science Week*,

ACSW '23, page 126–134, New York, NY, USA, 2023. Association for Computing Machinery. [Citado en pág. 13.]

[33] Yitong Ren, Yanjun Xiao, Yinghai Zhou, Zhiyong Zhang, and Zhihong Tian. Cskg4apt: A cybersecurity knowledge graph for advanced persistent threat organization attribution. *IEEE Transactions on Knowledge and Data Engineering*, 35(6):5695–5709, June 2023. [Citado en págs. 12 y 13.]

[34] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks, 2020. [Citado en pág. 56.]

[35] Wenli Shang, Bowen Wang, Pengcheng Zhu, Lei Ding, and Shuang Wang. A span-based multivariate information-aware embedding network for joint relational triplet extraction of threat intelligence. *Knowledge-Based Systems*, 295:111829, 2024. [Citado en pág. 12.]

[36] Ray Smith. An overview of the tesseract ocr engine. Google Inc., 2007. Accessed: 02/07/2024. [Citado en pág. 30.]

[37] Blake E. Strom, Andy Applebaum, Doug P. Miller, Kathryn C. Nickels, Adam G. Pennington, and Cody B. Thomas. Mitre att&ck: Design and philosophy. Technical Report MP180360R1, The MITRE Corporation, McLean, VA, 2020. Approved for Public Release. Distribution Unlimited 19-01075-28. [Citado en pág. 21.]

[38] Zareen Syed, Ankur Padia, Tim Finin, Lisa Mathews, and Anupam Joshi. Uco: A unified cybersecurity ontology. 02 2016. [Citado en pág. 17.]

[39] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003. [Citado en pág. 36.]

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. [Citado en pág. 32.]

[41] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019. [Citado en pág. 56.]

[42] Gaosheng Wang, Peipei Liu, Jintao Huang, Haoyu Bin, Xi Wang, and Hongsong Zhu. Knowcti: Knowledge-based cyber threat intelligence entity and relation extraction. *Computers Security*, 141:103824, 2024. [Citado en pág. 13.]

[43] Gang Zhao, Yanbin Gao, and Robert Meersman. An ontology-based approach to business modeling. *Proceedings of the International Conference of Knowledge Engineering and Decision Support*, 01 2004. [Citado en pág. 16.]

[44] Xiaojuan Zhao, Rong Jiang, Yue Han, Aiping Li, and Zhichao Peng. A survey on cybersecurity knowledge graph construction. *Computers Security*, 136:103524, 2024. [Citado en págs. 11 y 13.]