UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Threat Hunting basado en técnicas de Inteligencia Artificial

Departamento de comunicaciones
Universitat Politècnica de València

Tesis presentada para la obtención del grado de
Doctor en Ingeniería de Telecomunicación
por la *Universitat Politècnica de València*

Valencia, Abril de 2024

Autor:
Mario Aragonés Lozano

Director:
Dr. Manuel Esteve Domingo
Dr. Israel Pérez Llopis

A mis padres.

Agradecimientos

Me gustaría empezar estas líneas de agradecimientos con dos frases las cuáles considero que es importante recordar día a día, que son, en primer lugar, “Nada que valga la pena en la vida se consigue de forma fácil” y, en segundo lugar, “Como no estás experimentado en las cosas del mundo, todas las cosas que tienen dificultad te parecen imposibles; confía en el tiempo, que suele dar dulces salidas a muchas amargas dificultades”, esta última de Miguel de Cervantes Saavedra en su libro Don Quijote de la Mancha.

Las primeras personas a las que me gustaría dirigir estos agradecimientos son mis directores de tesis, el catedrático Dr. Manuel Esteve Domingo y el Dr. Israel Pérez Llopis. No tengo forma de agradecer todo lo que habéis hecho por mi desde que os conozco, empezando por lo profesional, por marcarme un objetivo acerca de lo que me gustaría llegar a ser algún día y, en cuanto a lo personal, por el bienestar que generáis en la gente que tenéis a vuestro alrededor. Sin duda alguna, ha sido todo un privilegio el haber coincidido con vosotros y haber podido contar con vosotros durante la realización de este trabajo.

Las segundas personas a las que me gustaría dirigir estos agradecimientos son mis padres, Manuel Aragonés García y Angelina Lozano Fuentes, a los cuáles está dedicado este trabajo. Puedo asegurar, sin miedo a equivocarme, que vosotros sois el motivo por el que este trabajo y, en general, cualquier aspecto de mi vida es como es. Siempre habéis estado en los momentos de alegría, pero más importante, en los momentos difíciles y, sin duda alguna, nunca tendré forma suficiente de agradeceros todo lo que hacéis por mi.

Como es de suponer, este trabajo no se ha conseguido ni de forma fácil ni sin ofrecer (en muchas ocasiones) amargas dificultades, no obstante, gracias a vuestra ayuda y soporte he sido capaz de generar el resultado desarrollado en este documento, por dicho motivo, debéis tener por seguro que este trabajo es tan vuestro como mío.

Resumen

Tanto la cantidad como la tipología de los ciberataques va en aumento día a día y la tendencia es que continúen creciendo de forma exponencial en los próximos años. Estos ciberataques afectan a todos los dispositivos, independientemente de si su propietario es un particular (o ciudadano), una empresa privada, un organismo público o una infraestructura crítica (IC) y los objetivos de estos ataques son muchos, desde la solicitud de una recompensa económica hasta el robo de información clasificada. Dado este hecho, los individuos, las organizaciones y las corporaciones deben tomar medidas para prevenirlos y, en caso de que en algún momento los reciban, analizarlos y reaccionar en caso de que fuese necesario.

Cabe destacar que aquellos ataques que buscan ser más eficientes, son capaces de ocultarse un largo tiempo, incluso después de sus acciones iniciales, por lo que la detección del ataque y el saneamiento del sistema puede llegar a dificultarse a niveles insospechados o, incluso, no tenerse la certeza de que se ha hecho correctamente.

Para prevenir, analizar y reaccionar ante los ataques más complejos, normalmente conocidos como ataques de día cero, las organizaciones deben tener ciberespecialistas conocidos como cazadores de amenazas (TH, del inglés *Threat Hunters*). Éstos son los encargados de monitorizar los dispositivos de la empresa con el objetivo de detectar comportamientos extraños, analizarlos y concluir si se está produciendo un ataque o no con la finalidad de tomar decisiones al respecto.

Estos ciberespecialistas deben analizar grandes cantidades de datos (mayormente benignos, repetitivos y con patrones predecibles) en cortos periodos de tiempo para detectar ciberataques, con la sobrecarga cognitiva asociada. El uso de inteligencia artificial (AI, del inglés *Artificial Intelligence*), específicamente aprendizaje automático (ML, del inglés *Machine Learning*) y aprendizaje profundo (DL, del inglés *Deep Learning*), puede impactar de forma notable en el análisis en tiempo real de dichos datos. Además, si los ciberespecialistas son capaces de visualizar los datos de forma correcta, éstos pueden ser capaces de obtener una mayor consciencia situacional del problema al que se enfrentan.

Este trabajo busca definir una arquitectura que contemple desde la adquisición de datos hasta la visualización de los mismos, pasando por el procesamiento de éstos y la generación de hipótesis acerca de lo que está sucediendo en la infraestructura monitorizada. Además, en la definición de la misma se deberá tener en consideración aspectos tan importantes como la disponibilidad, integridad y confidencialidad de los datos, así como la alta disponibilidad (HA, del inglés *High Availability*) de los distintos componentes que conformen ésta. Una vez definida la arquitectura, este trabajo busca validarla haciendo uso de un prototipo que la implemente en su totalidad. Durante esta fase de evaluación, es importante que quede demostrada la versatilidad de la arquitectura propuesta para trabajar en diferentes casos de uso, así como su capacidad para adaptarse a los cambios que se producen en las distintas técnicas de ML y DL.

Resum

Tant la quantitat com la tipologia dels ciberatacs va en augment dia a dia i la tendència és que continuen creixent de manera exponencial en els pròxims anys. Aquestos ciberatacs afecten a tots els dispositius, independentment de si el seu propietari és un particular (o ciutadà), una empresa privada, un organisme públic o una IC i els objectius d'aquestos atacs són molts, des de la sol · licitud d'una recompensa econòmica fins al robatori d'informació classificada. Donat aquest fet, els individus, les organitzacions i les corporacions deuen prendre mesures per a previndre'ls i, en cas que en algun moment els reben, analitzar-los i reaccionar en cas que fora necessari.

Cal destacar que aquells atacs que busquen ser més eficients, són capaços d'ocultar-se un llarg temps, fins i tot després de les seues accions inicials, per la qual cosa la detecció de l'atac i el sanejament del sistema pot arribar a dificultar-se a nivells insospitats o, fins i tot, no tindre's la certesa que s'ha fet correctament.

Per a previndre, analitzar i reaccionar davant els atacs més complexos, normalment coneguts com a atacs de dia zero, les organitzacions han de tindre ciberespecialistes coneguts com TH. Aquestos són els encarregats de monitoritzar els dispositius de l'empresa amb l'objectiu de detectar comportaments estranys, analitzar-los i concloure si s'està produint un atac o no amb la finalitat de prendre decisions al respecte.

Aquestos ciberespecialistes han d'analitzar grans quantitats de dades (majoritàriament benignes, repetitives i amb patrons predictibles) en curts períodes de temps per a detectar els ciberatacs, amb la sobrecàrrega cognitiva associada. L'ús de AI, específicament ML i DL, pot impactar de manera notable en l'anàlisi en temps real d'aquestes dades. A més, si els ciberespecialistes són capaços de visualitzar les dades de manera correcta, aquestos poden ser capaços d'obtindre una major consciència situacional del problema al qual s'enfronten.

Aquest treball busca definir una arquitectura que contemple des de l'adquisició de dades fins a la visualització d'aquestes, passant pel processament de la informació recol · lectada i la generació d'hipòtesis sobre el que està succeint en la infraestructura monitoritzada. A més, en la definició de la mateixa s'haurà de tindre en consideració aspectes tan importants com la disponibilitat, integritat i confidencialitat de les dades, així com la HA dels diferents components que conformen aquesta. Una volta s'hatja definit l'arquitectura, aquest treball busca validar-la fent ús d'un prototip que la implemente íntegrament. Durant aquesta fase d'avaluació, és important que quede demostrada la versatilitat de l'arquitectura proposada per a treballar en diferents casos d'ús, així com la seua capacitat per a adaptar-se als canvis que es produïxen en les diferents tècniques de ML i DL.

Abstract

Both the number and type of cyber-attacks are increasing day by day and the trend is that they will continue to grow exponentially in the coming years. These cyber-attacks affect all devices, regardless of whether the owner is an individual (or citizen), a private company, a public entity or a IC, and the targets of these attacks are many, ranging from the demand for financial reward to the theft of classified information. Given this fact, individuals, organisations and corporations must take steps to prevent them and, in case they ever receive them, analyse them and react if necessary.

It should be noted that those attacks that seek to be more efficient are able to hide for a long time, even after their initial actions, so that the detection of the attack and the remediation of the system can become difficult to unsuspected levels or even uncertain whether it has been done correctly.

To prevent, analyse and react to the most complex attacks, usually known as zero-day attacks, organisations must have cyber-specialists known as TH. They are responsible for monitoring the company's devices in order to detect strange behaviours, analyse it and conclude whether or not an attack is taking place in order to make decisions about it.

These cyber-specialists must analyse large amounts of data (mostly benign, repetitive and with predictable patterns) in short periods of time to detect cyber-attacks, with the associated cognitive overload. The use of AI, specifically ML and DL, can significantly impact the real-time analysis of such data. Not only that, but if these cyber-specialists are able to visualise the data correctly, they may be able to gain greater situational awareness of the problem they face.

This work seeks to define an architecture that contemplates from data acquisition to data visualisation, including data processing and the generation of hypotheses about what is happening in the monitored infrastructure. In addition, the definition of the architecture must take into consideration important aspects such as the availability, integrity and confidentiality of the data, as well as the HA of the different components that make it up. Once the architecture has been defined, this work seeks to validate it by using a prototype that fully implements it. During this evaluation phase, it is important to demonstrate the versatility of the proposed architecture to work in different use cases, as well as its capacity to adapt to the changes that occur in the different ML and DL techniques.

Índice general

Índice de figuras

Índice de tablas

Índice de códigos

Siglas

AI Inteligencia artificial. 5, 7–10, 21, 24–28, 30, 33–35, 37, 51, 66, 69, 74, 80–82, 84, 91, 108, 109, 121, 122, II, IV, VI

AMQP *Advanced Message Queuing Protocol.* 21, 53

APT Amenaza persistente avanzada. 82, 95, 96, 98

ASR Procesado automático del lenguaje. 37, 103

BERT Representación de codificador bidireccional de transformadores. 34, 85, 105

BiLSTM Red neuronal LSTM bidireccional. 33

C-RNN-GAN Red neuronal recurrente con entrenamiento generativo continuo basado en adversarios. 32, 37, 85

C-TMAC TMAC basado en contexto. 57

CoAP *Constrained Application Protocol.* 19

CR *Coordinated Response.* 111, 112, 114

CSA *Cyber Situational Awareness.* 111–115, 118

CVE Vulnerabilidades y exposiciones comunes. 39

CyberDEM *Cyber Data Exchange Model.* 39

DAC Control de acceso discrecional. 56, 57

DBSCAN Agrupamiento espacial basado en densidad de aplicaciones con ruido. 30, 85

DL Aprendizaje profundo. 7, 9, 24, 66, 69, 73, 75, 80, 85, 107, 121, II, IV, VI

ECS Elastic Common Schema. 23, 66, 83, 84
ESB Bus de servicios empresariales. 14, 16

GDPR Regulación general de protección de datos de la Unión Europea. 50

HA Alta disponibilidad. 12, 13, 15, 48, 53, 75, II, IV, VI
HDD Unidad de disco duro. 80
HIDS Sistema de detección de intrusiones a nivel de máquina. 105
HMI Interfaz de humano a máquina. 40, 65, 75, 87, 88, 90, 94, 96, 100, 112–114
HSA *Hybrid Situational Awareness*. 111, 112, 114
HTTP Protocolo de transferencia de hipertexto. 18–20, 38, 42, 43
HTTPS Protocolo seguro de transferencia de hipertexto. 19

IC Infraestructura crítica. 8, 110–113, 115, 116, I, III, V
IDE Entorno de desarrollo integrado. 88
idP Proveedor de identidad. 55
IDS Sistema de detección de intrusiones. 112
IoC Indicador de compromiso. 8, 9
IoT Internet de las cosas. 8
IPS Sistema de prevención de intrusiones. 112

JSON *JavaScript Object Notation*. 39, 42, 43, 46, 87, 88

KG Grafo de conocimiento. 102, 104

LDA Análisis discriminante lineal. 25
LLM Modelo de lenguaje de gran tamaño. 35–38, 85–87, 95, 100, 102–110, 119, 121, 122
LSTM Memoria a corto plazo largo. 36
LSTM RNN Red neuronal recurrente de memoria a corto plazo largo. 33, 34

M2M Máquina a máquina. 19, 20, 78, 90
ML Aprendizaje automático. 2–4, 7–10, 24, 30, 32, 36, 37, 65–70, 73–75, 80, 81, 83, 85, 87, 91–93, 96, 99, 100, 106–108, 112, 114, 115, 119–122, II, IV, VI

MoE *Mixture of Experts*. 110
MQTT *Message Queuing Telemetry Transport*. 20, 21, 54

NER Reconocimiento de entidades nombradas. 102, 104
NIDS Sistema de detección de intrusiones a nivel de red. 105
NLP Procesado del lenguaje natural. 33–35, 37, 96, 100, 102, 105
NoSQL No sólo SQL. 50, 51

OCR Reconocimiento óptico de caracteres. 86
OSINT Inteligencia de fuentes abiertas. 64
OSS Programa de código abierto. 38, 85, 100, 122

PCA Análisis de componentes principales. 25, 83, 106
PCAP Captura de paquete. 64, 82
PSA *Physical Situational Awareness*. 111, 112, 114, 117
PYME Pequeña y mediana empresa. 1, 111

QoS Calidad de servicio. 20

RAG *Retrieval Augmented Generative AI*. 86, 103, 107
RBAC Control de acceso basado en roles. 56, 57, 90
REA Agencia europea de investigación. 6, 79, 110
RPC Llamada a procedimiento remoto. 42

SCAP *Security Content Automation Protocol*. 39
SDN Red definida por *software*. 8
SIEM Sistema de gestión de información y eventos de seguridad. 64, 82, 112
SOA Arquitectura orientada a servicios. 11, 14–16
SOAP Protocolo simple de acceso a objetos. 14
SOC Centro de operaciones de seguridad. 34
SPA Aplicación de página única. 88
SQL Lenguaje de consulta estructurada. 51
SSD Unidad de estado sólido. 80
SSO Inicio de sesión único. 55

TBAC Control de acceso basado en tareas. 57
TF-IDF *Term Frequency Inverse Document Frequency*. 35
TH Cazador de amenazas. 2–5, 7, 8, 10, 68, 72, 73, 76, 80, 82, 93, 100, 102, 104, 105, 108, 109, 112, 119–121, I, III, V

TMAC Control de acceso basado en equipos. 57
TTS Generación automática de audio a partir de texto. 37

UI Interfaz de usuario. 43, 94
UX Experiencia de usuario. 43, 94

XML *Extensible Markup Language*. 39, 42, 46

Capítulo 1

Introducción y objetivos

En este capítulo se expondrá el problema que se ha detectado junto con los objetivos que se espera alcanzar como resultado de este trabajo. Además, se especificarán las principales aportaciones así como la estructura de la presente memoria.

1.1 Introducción

En el mundo actual donde absolutamente todo está (o busca estar) hiperconectado, se ha generado una dependencia total a las redes de comunicaciones, más específicamente, a Internet. Esta dependencia tiene como resultado que la falta de acceso a dicha red de comunicación deja inservible cualquier servicio, ya esté ofrecido por una pequeña y mediana empresa (PYME) o por una gran corporación, lo que conlleva asociado perdidas económicas y de reputación, aunque el servicio únicamente haya estado inutilizado por horas. No sólo las organizaciones, sino la gran mayoría de personas está rodeada y necesita de dispositivos hiperconectados (como por ejemplo, asistentes de voz) para cualquier actividad. Éstas ni tan siquiera se plantean quién hace uso de sus datos o como se van a procesar los mismos, entre otros. No obstante, lo que es peor todavía, éstas configuran dichos dispositivos sin ni siquiera conocer el impacto que pueden tener dichas modificaciones en la protección de los mismos. Esta falta de conocimiento del grado de exposición (en definitiva, falta de cultura de ciberseguridad) es bien conocida por los ciberdelincuentes, en consecuencia, éstos han puesto a la orden del dia los ciberataques a cualquier objetivo del que puedan sacar rentabilidad.

Para prevenir los ciberataques o, al menos, protegerse ante ellos, las grandes corporaciones están invirtiendo incontables cantidades de dinero en la mejora de los departamentos de ciberseguridad haciéndolos cada vez más grandes con la contratación de profesionales especializados en dicho ámbito [3, 99]. El objetivo que se persigue es el de evitar la pérdida y exfiltración de datos, mantener la reputación y, probablemente, su mayor preocupación, minimizar en todo lo posible el impacto en la continuidad del negocio [9, 171].

Cuando una empresa es muy grande, solamente con la contratación de personal cualificado no es suficiente ya que la gran cantidad de datos a analizar provenientes de todos los dispositivos que se monitorizan y generados en casi tiempo real dejan desbordados a los ciberespecialistas, con lo que es más probable que, por sus peculiaridades, los ataques más complejos consigan ejecutarse satisfactoriamente.

Por otro lado, gran cantidad de los datos que son analizados por dichos ciberespecialistas, ya sea tanto de estaciones de trabajo como de equipos de red, son relacionados con las acciones del día a día de los empleados (como por ejemplo peticiones DNS o navegar por Internet). No solo eso, después de haber realizado cuestionarios a los TH [201] sobre los datos que analizan en su trabajo diario, éstos destacan que muchas de las amenazas que descartan son predecibles y siguen patrones, como por ejemplo, cuando se encuentran con empleados que siempre se olvidan de conectar la VPN para acceder a ciertos recursos compartidos o eventos similares.

Debido al hecho de que se pueden discernir patrones, cabe el pensar de aplicar técnicas para detectarlos automáticamente. Si se plantea dicha hipótesis, probablemente el mejor aliado puede ser el ML ya que es un campo de la ciencia que se caracteriza por proveer de técnicas y procedimientos para la extracción de modelos a partir de información en bruto [148], por tanto, si se generase un escenario de trabajo donde se tuviesen en cuenta algoritmos de ML bien diseñados y correctamente entrenados, se podría llegar a clasificar los distintos eventos a analizar en como de benignos o malignos son.

No solo eso, según diversos estudios [66, 67, 214], la cognición humana tiende a predecir palabras, patrones, etc. y, además, esta predicción está fuertemente influenciada por el contexto [238], todavía más si la toma de decisiones se produce bajo condiciones de estrés [145]. Hay muchas situaciones donde se pueden producir estas situaciones de estrés, siendo una de ellas la que sufren los TH al tener que enfrentarse a grandes cantidades de datos en un tiempo reducido sobre un contexto que varía en cada segundo que pasa y, además, que las decisiones que se van tomando sobre el camino tienen impactos muy grandes, con lo que un mínimo error puede provocar daños incalculables. Por si fuera poco, a todo esto

hay que sumarle que cuando los TH están ante un incidente, muy probablemente, el escenario en el que se encuentren sea completamente desconocido para ellos, ya que el ataque puede no haberse producido con anterioridad y, por tanto, no tengan ni idea de a lo que se enfrentan, por ejemplo, es el caso de los ataques de día cero [26]. Como consecuencia a los hechos anteriormente expuestos y con especial énfasis a la fuerte dependencia en el contexto que tiene la cognición humana a la hora de la predicción, un ataque cuyo comportamiento fuera muy similar al de un usuario del sistema, se podría desechar al considerarse que no es un ataque, en cambio, un sistema de ML sería capaz de discriminar ambos con más precisión, por tanto, si los TH tuvieran la información generada por sistemas de ML (como podrían ser probabilidades, margenes de desviación, etc.), estos ciberespecialistas serían capaces de comprender mejor lo que está sucediendo.

Si lo que se busca finalmente es que los ciberespecialistas comprendan aquello a lo que se enfrentan de la mejor y más rápida manera posible, la generación de reportes textuales, aunque muy orientados, puede llegar a no ser suficiente, ya que como es bien conocido, el cerebro humano procesa patrones visuales mucho más rápido y de forma más precisa que cualquier otro tipo de reporte y, naturalmente, en el caso de la ciberseguridad no es distinto [166, 277], como consecuencia, la adecuada representación y visualización de los datos (ya sea datos en brutos o procesados por sistemas de ML) se considera un factor crítico para los TH para conseguir una consciencia situacional [62, 73] y, por tanto, una pronta detección de cualquier amenaza.

En resumen, teniendo tanto sistemas específicos y bien diseñados de ML (entrenados teniendo como objetivo los problemas que presenta la ciberseguridad), como visualizaciones bien definidas de la información, los TH podrían ser capaces de generar hipótesis de lo que está sucediendo en los sistemas que monitorizan y protegen, pudiendo detectar de forma temprana cualquier amenaza y teniendo suficiente conocimiento para poder enfrentarse a la misma.

La mejor forma de combinar las herramientas de ML y las visualizaciones de la información junto con el resto de elementos auxiliares necesarios para proveer, procesar e interactuar con los datos es mediante una aplicación, o más específicamente, mediante un sistema conformado por un conjunto de aplicaciones. Estos sistemas deben ser capaces de obtener enormes cantidades de datos de todas las fuentes posibles, procesarlos en varias etapas (incluyendo el procesado por herramientas de ML), proveer herramientas para la visualización y, finalmente, permitir la generación de hipótesis. Para conseguirlo, se debe diseñar una arquitectura de aplicación la cual debe ser capaz de tener en cuenta todos los problemas que se deben afrontar. Además, dicha arquitectura debe tener en consideración que las técnicas de ML están en constante evolución y por tanto debe ser fácil

la adición, modificación y borrado de las mismas. No solo eso, gran cantidad de datos van a ser obtenidos, almacenados y procesados en cortos periodos de tiempo, por tanto, se deben considerar técnicas de *big data* para lograr la máxima eficiencia. Teniendo en cuenta lo costoso en tiempo que son algunas técnicas de ML, la paralelización de procesos es un aspecto clave a tener en cuenta. Por otro lado, la arquitectura debe ser capaz de proveer escalado de recursos de forma asimétrica, ya que no todos los recursos definidos van a ser usados por igual, en consecuencia, sería poco eficiente tener que hacer crecer todos los componentes que conforman la arquitectura por falta de recursos en uno de ellos. Finalmente, y no por ello menos importante, los datos con los que se va a trabajar son extremadamente sensibles y su mayor uso se va a producir en situaciones de riesgo, por tanto, todo el sistema debe contar con elementos de seguridad y, además, garantizar la integridad y la confidencialidad de los mismos en todo momento, no solo eso, también es crucial asegurar la disponibilidad para que los usuarios de la aplicación que implemente dicha arquitectura puedan conseguir dar solución a las amenazas detectadas a la mayor brevedad posible.

1.2 Objetivos

El presente trabajo se centra en detectar las necesidades que tienen los TH cuando desempeñan su trabajo diario. Acto seguido, este trabajo plantea definir una arquitectura de aplicación encargada de ayudar a estos especialistas, la cuál espera emplear técnicas de ML para ofrecer la ayuda previamente enunciada. Para proseguir, se busca desarrollar un prototipo que implemente la arquitectura propuesta para validar y verificar la misma.

Con el fin de conseguir el objetivo global anteriormente descrito, se deberán realizar de forma satisfactoria los objetivos parciales que se definen a continuación:

- Realizar un análisis exhaustivo acerca de las necesidades reales de los TH en su trabajo diario. Este trabajo permitirá conocer no solo cuáles son los puntos más destacables del proceso de la caza de amenazas sino también los flujos de trabajo de estos especialistas, lo que permitirá que la arquitectura propuesta se amolde perfectamente a su manera de trabajar.
- Analizar en profundidad el estado del arte actual. En dicho análisis, realizar especial énfasis en las soluciones actuales para realizar la caza de amenazas para detectar si se cumplen las necesidades de los TH. Por otro lado, investigar acerca de los recursos actuales para la definición de una arquitectura de aplicación así como los servicios disponibles a emplear por parte de una aplicación que implemente dicha arquitectura.

- Proponer una arquitectura de aplicación para realizar la caza de amenazas que contemple todas las necesidades de los TH y que, además, permita hacer uso de las bondades de las técnicas de AI para ayudar a los especialistas en su trabajo diario.
- Diseñar y desarrollar un prototipo con el que validar la arquitectura propuesta, el cuál debe implementar todos y cada uno de los componentes de dicha arquitectura.
- Validar la funcionalidad de cada uno de los componentes definidos en la arquitectura mediante el prototipo desarrollado para asegurar que la definición individual de cada uno de los componentes cumpla con los objetivos esperados.
- Verificar la arquitectura en su totalidad haciendo uso del prototipo desarrollado en casos de uso donde se contemplen tanto entornos controlados como entornos no controlados.

1.3 Principales aportaciones

1.3.1 Artículos de revista

- **Mario Aragonés Lozano**, Israel Pérez Llopis y Manuel Esteve Domingo. "Threat Hunting Architecture Using a Machine Learning Approach for Critical Infrastructures Protection". En: *Big data and cognitive computing* 7.2 (2023), pág. 65.
- **Mario Aragonés Lozano**, Israel Pérez Llopis y Manuel Esteve Domingo. "Threat Hunting System for Protecting Critical Infrastructures Using a Machine Learning Approach". En: *Mathematics* 11.16 (2023), pág. 3448.

1.3.2 Publicaciones en congresos

- Jorge Maestre Vidal, Marco Antonio Sotelo Monge, Francisco Antonio Rodríguez López, Mónica Mateos Calle, Juan Manuel Estevez Tapiador, Victor Villagrá González, Daniel Tornero Sánchez, Rebeca Gómez Henche y **Mario Aragonés Lozano**. "Plataforma Europea para adquisición de Consciencia de Situación del Ciberespacio". En: *VII Jornadas Nacionales de investigación en Ciberseguridad* (JNIC 2022). Fundación Tecnalia Research and Innovation. 2022, págs. 117-121.

- **Mario Aragonés Lozano**, Israel Pérez Llopis, Alfonso Climente Alarcón y Manuel Esteve Domingo. "A Machine Learning-Driven Threat Hunting Architecture for Protecting Critical Infrastructures". En: *2023 19th International Conference on the Design of Reliable Communication Networks* (DRCN). IEEE. 2023, págs. 1-5.

1.3.3 Participación en proyectos de investigación

- Proyecto PRAETORIAN
  - H2020-SU-INFRA-2020 Protection of Critical Infrastructures from advanced Combined cyber and physical threats (PRAETORIAN)
  - Proyecto financiado por la llamada SU-INFRA dentro de la convocatoria H2020 de proyectos de I+D de la agencia europea de investigación (REA, del inglés *European Research Executive Agency*) de la Comisión Europea.
  - Convenio de subvención número 101021274.

1.4 Estructura de la memoria

La estructura de la memoria del trabajo realizado es la siguiente:

- Primer capítulo, introducción y objetivos, se expone el problema que se ha detectado, los objetivos que se espera alcanzar y las principales aportaciones de este trabajo.
- Segundo capítulo, estado del arte, se realiza un análisis tanto de las soluciones actuales al problema encontrado como de los posibles recursos y servicios a emplear en las distintas fases del trabajo.
- Tercer capítulo, propuesta de arquitectura, se propone una arquitectura para realizar la caza de amenazas basándose en técnicas de inteligencia artificial.
- Cuarto capítulo, validación y verificación de la arquitectura, se describe como se han realizado el conjunto de pruebas de validación y verificación de la arquitectura.
- Quinto capítulo, conclusiones y trabajos futuros, se analiza el trabajo realizado así como las posibles lineas de investigación originadas a partir de los resultados obtenidos.

Capítulo 2

Estado del arte

En este capítulo se realizará un análisis tanto de las soluciones actuales al problema encontrado como de los posibles recursos y servicios a emplear en las distintas fases del trabajo.

Siendo la ciberseguridad un aspecto con gran relevancia en la actualidad y la AI una herramienta cada vez más empleada en el día a día de los individuos, cabría suponer que ya existiesen estudios acerca del problema detectado, por tanto, el primer paso realizado consistió en investigar acerca de las soluciones actuales. A continuación, se analizaron cuales son las distintas arquitecturas de aplicación existentes. Para proseguir, se investigaron los distintos mecanismos de comunicación empleados en la actualidad. Finalmente, se realizó un estudio tanto de los recursos actuales como de los servicios actuales.

2.1 Soluciones actuales

Es bien conocido que en la actualidad el uso de la AI, más específicamente, el uso de ML y de DL, está aplicándose a todos y cada uno de los aspectos de la vida cotidiana, siendo un claro ejemplo la herramienta ChatGPT [164]. Dado el hecho anteriormente expuesto, cabría pensar que los ciberespecialistas tienen una gran variedad de herramientas capaces de ayudarles en el proceso de toma de decisiones y generación de hipótesis, por ese motivo, se analizaron las soluciones actuales (tanto académicas como comerciales) y se consultó a varios TH cuáles son las herramientas empleadas en la actualidad, buscando así saber si cumplen con todos los requerimientos que éstos necesitan.

En primer lugar, el análisis se centró en conocer cuáles son los distintos posibles ámbitos de aplicación en los que potencialmente estaría interesado un TH y conocer cuáles son las soluciones que se han propuesto en los diferentes estudios y trabajos en la actualidad para hacer frente a cada uno de ellos empleando técnicas de ML. Éstos son muchos y muy variados y, de entre todos, se han destacado estudios que se centran en proponer soluciones basadas en AI para IC, redes definidas por *software* (SDN, del inglés *Software-Defined Networks*) e Internet de las cosas (IoT, del inglés *Internet of Things*). Dentro de este análisis cabría destacar, en primer lugar, el estudio [113] que se centra en IC en cuanto al tiempo de respuesta, por lo tanto, tiene especialmente en cuenta la casuística de situaciones cuyo tiempo de reacción es muy limitado, en consecuencia, busca la máxima efectividad en la detección de *malware* que puede proveer el ML. Para proseguir, los dos estudios [228] y [229] se centran en el desarrollo de aproximaciones para la caza de amenazas basadas en ML, específicamente, para las SDN. Siendo las SDN un aspecto a tener en cuenta en el desarrollo de nuevas redes de comunicaciones por las oportunidades que ofrecen en cuánto a centralización del comportamiento de las redes y por su capacidad para modificar estas mediante *software* [127], toda aplicación que busque ser efectiva y completa para realizar la caza de amenazas deben contemplar aspectos específicos para este tipo de ámbito y, en consecuencia, ambos estudios previamente mencionados aportan un gran valor. También se encuentran estudios como [95] y [209] cuyo objetivo es la protección de los ecosistemas del IoT. Estando el IoT en auge en los últimos años tanto para industria [5] como para uso doméstico [244], toda aplicación de caza de amenazas que se precie debe tener en consideración los aspectos característicos de esta tecnología, donde, el ML puede ser muy relevante, ya que dentro de estos ecosistemas suele haber falta de recursos, ya sea de almacenaje, de procesamiento, etc., y donde la aplicación de técnicas de AI puede ayudar reduciendo la cantidad de información que se almacena y/o intercambia a la hora de la protección. No obstante, el empleo de técnicas de ML en el IoT puede ser muy costoso energéticamente, por tanto, en los distintos estudios y trabajos desarrollados en la actualidad se propone el empleo de aceleradores de ML (como NPUs [144] o TPUs [117]) para conseguir la eficiencia energética [7].

Una vez conocidos una parte considerable de los estudios que emplean ML para la caza de amenazas, el siguiente paso fue investigar acerca de que soluciones han propuesto una arquitectura de aplicación específica y verificada para realizar la caza de amenazas mediante ML. Se pueden encontrar estudios como [81] donde una arquitectura con todas las capas desde la de adquisición hasta la de visualización de los datos es propuesta, aunque su mayor inconveniente es que está centrada en los indicadores de compromiso (IoC, del inglés *Indicators of Compromise*). Otro trabajo interesante que se puede encontrar es [16]. En este trabajo,

a similitud con el anterior, se trata de desarrollar una arquitectura para la generación y enriquecimiento de IoC. Se considera de interés al proporcionar una perspectiva distinta del proceso, aunque no tiene en consideración la visualización de los resultados. Finalmente destacar el trabajo [10] que ofrece una arquitectura muy bien definida para la detección de amenazas teniendo en cuenta desde como recolectar la información hasta como visualizarla. A pesar de ser una opción muy válida, tiene el inconveniente de estar centrada únicamente en una fuente de datos, la red social Twitter (actualmente X). Cabe destacar que en ninguno de los estudios analizados se propone la opción de generación de hipótesis directamente en la arquitectura, ya sea de forma manual o automática. Por tanto, a pesar de todos los esfuerzos realizados hasta la fecha, no existen trabajos específicos sobre una arquitectura que permita la caza de amenazas basada en técnicas de AI, más precisamente, ML o DL, y que cumpla todos los requerimientos desde la adquisición de los datos hasta la visualización, pasando por el procesamiento de los mismos, entre otros.

Por otro lado, no solo es importante ser capaz de procesar los datos mediante AI para conseguir resultados más fiables de una forma menos compleja, también se considera importante el ser capaz de visualizar la información obtenida de forma que el usuario pueda tomar decisiones en situaciones donde hay que actuar muy rápidamente y donde hay mucha información que procesar. Por el motivo anteriormente expuesto, se ha considerado importante realizar un estudio en profundidad acerca de visualizaciones específicas para la ciberconsciencia situacional. En primer lugar se puede encontrar el estudio [135] que expone "Las analíticas visuales se centran en realizar razonamientos analíticos mediante visualizaciones interactivas". Para respaldar la anterior afirmación, en el trabajo [90] se realiza un estudio muy completo sobre los fundamentos de las analíticas visuales. A partir de dicho estudio se pueden conocer una gran variedad de técnicas de visualización. Primeramente, se pueden destacar las técnicas de visualización simples, las cuales podrían contener diagramas de dispersión [19, 63, 134], diagramas de barras [82, 200, 284], diagramas de tarta [284] y diagramas de linea [2, 87, 200]. Otro tipo de visualizaciones simples que se pueden encontrar podrían ser las nubes de palabras [158, 276] y los árboles de decisión [49, 252]. A parte de las visualizaciones simples, también se pueden encontrar las visualizaciones complejas. Las hay de muchos tipos, como por ejemplo, aquellas orientadas a detección de patrones [12, 39, 119, 133, 281]. Asimismo, también existen visualizaciones centradas en geoposicionamiento, ya sea para activos [39, 133, 157], riesgos [36, 170, 199] y amenazas [39, 157]. Finalmente, destacar el desarrollo de visualizaciones en 3D representadas mediante pantallas que envuelven al usuario o mediante gafas de realidad virtual donde éste puede obtener vistas de 360 grados si se interactúa con las mismas [118, 119, 138, 139, 157].

Aparte de estudios realizados por instituciones académicas, también existen estudios y productos de empresas que hacen uso de ML para realizar la caza de amenazas. Algunos de estos podrían ser Splunk [213], los cortafuegos de nueva generación de Palo Alto [152], los sistemas de IBM para ciberseguridad (IBM X-Force Exchange [110, 255]), o productos como Anomali ThreatStream [13].

Tal y como se puede concluir, el proceso de la caza de amenazas ofrece una gran complejidad en cuanto a conocimiento de la situación se refiere debido a la gran cantidad de características necesarias para definir una amenaza, todo esto sumado a que las decisiones se deben tomar en muy poco tiempo dentro de unas condiciones que varían muy rápidamente, además, trabajando sobre amenazas normalmente desconocidas hasta la fecha y, como añadido, con información incompleta en la mayoría de los casos.

La conclusión anteriormente expuesta confirma las deducidas a partir de los cuestionarios realizados a los TH [201], destacando que ninguno de los trabajos analizados tiene en cuenta todos los requerimientos que los ciberespecialistas consideran relevantes para una aplicación de *Threat Hunting* basada en técnicas de AI, como por ejemplo, la generación de hipótesis (ya sea manual o automatizada).

2.2 Arquitecturas de aplicación

Una arquitectura de aplicación ofrece una visión global y de alto nivel acerca de un sistema conformado por un conjunto de programas [204]. Éstas, pasaron a formar parte de la ingeniería de software a partir de mediados de los años 90 hasta a día de hoy, estableciéndose los conceptos más básicos en [234], donde se indica que toda arquitectura de aplicación implica la definición de cuáles son:

- Los componentes y la función que se espera que desempeñen los mismos
- Las tecnologías a emplear en el desarrollo de los componentes
- Las relaciones entre componentes internos del sistema
- Las relaciones con elementos externos

Cabe destacar que en la definición de la arquitectura solamente se describe el comportamiento de los componentes y como éstos se comunican con el resto de componentes y elementos externos [31, 163, 204], tal y como se especifica en gran cantidad de taxonomías en la literatura, como por ejemplo, Darwin [161, 162]. Por este motivo, las arquitecturas de aplicación se definen antes de que cualquier pieza de código sea programada.

A lo largo del desarrollo del *software*, las arquitecturas de aplicación se han ido modificando y adaptándose tanto a las necesidades como a las tecnologías disponibles. En los inicios del uso de aplicaciones en la sociedad, años 1970, las arquitecturas de programa empleadas eran las de tipo monolíticas [174]. Por aquel entonces, los usuarios adaptaban sus procedimientos a las necesidades que tenían los programas para que estos funcionasen siempre de forma óptima, como consecuencia, esta aproximación es la más eficiente, ofreciendo costes reducidos y una baja complejidad del sistema, no solo eso, en este tipo de arquitectura de aplicación, los sistemas generados son muy estables en términos de interacción entre componentes. No obstante, las arquitecturas que emplean este paradigma sufren de complejidad a la hora de añadir funcionalidades y a la hora de ampliar los recursos.

La evolución producida en la tecnología de computación entre los años 1950 y 1990 [184] junto con el avance de las intercomunicaciones entre dispositivos [266] hizo que se produjera un cambio en el paradigma de desarrollo, donde las aplicaciones necesitaron cada vez más módulos específicos dependientes del consumidor, como consecuencia, se buscaron alternativas a este tipo de arquitecturas de aplicación y, en los inicios de los años 1990, se generaron las bases de las conocidas como arquitecturas de tipo distribuidas [140]. Este segundo tipo de arquitectura de programa intenta solucionar los problemas anteriormente detectados y ofrece la posibilidad de añadir nuevos componentes a las aplicaciones que ya han salido de la fase de desarrollo y, por lo tanto, que se encuentran siendo activamente explotadas (es decir, funcionando en producción) sin suponer grandes inconvenientes a los usuarios finales. Además, permiten añadir más recursos (como capacidad de computación, etc.) a ciertos componentes en particular en vez de a todos los componentes a la vez [183], para ello, permiten que los componentes de la arquitectura estén distribuidos en varios nodos y se comuniquen entre ellos empleando una red de comunicaciones [266]. Por contrapartida, las arquitecturas de tipo distribuido introducen una capa adicional de comunicación entre los distintos componentes, lo que las hacen menos eficientes y más complejas de desarrollar, depurar y mantener. Cabe destacar que desde los inicios de las arquitecturas de tipo distribuidas hasta la actualidad se ha producido una evolución muy significativa, por lo que existen subconjuntos dentro de este paradigma de arquitectura dependiendo de como se realiza tanto la definición de los componentes como la forma de interactuar entre ellos. Dentro de esta evolución se puede encontrar, entre otras, las conocidas como arquitecturas orientadas a servicios (SOA, del inglés *Service Oriented Architectures*) y las arquitecturas distribuidas basadas en microservicios [223], ambas detalladas a continuación.

Esquematizando, las arquitecturas de aplicación se podrían clasificar en las de tipo monolíticas y en las de tipo distribuidas.

2.2.1 Monolíticas

Las arquitecturas monolíticas son las primeras que se emplearon en el desarrollo de aplicaciones. En este tipo de arquitecturas, todos los componentes que forman la aplicación se encapsulan en un único programa, consiguiendo que la interacción entre componentes sea directa.

La principal ventaja de este tipo de aplicaciones, por tanto, es la velocidad de interacción entre componentes. Además, al estar todos los componentes juntos, es más difícil generar incoherencias en las estructuras de datos entre los mismos. Asimismo, aquellas aplicaciones que, en primer lugar, tienen la peculiaridad de emplear componentes que no se ven modificados en el tiempo y, además, existe una interacción muy alta entre éstos, son muy eficientes al tener la anteriormente mencionada conexión directa entre componentes y, por tanto, no necesitar toda la información de control y verificación de las arquitecturas distribuidas.

Por otro lado, las arquitecturas de tipo monolíticas son más costosas a la hora de generar HA [88]. Además, únicamente permiten el escalado de tipo vertical [74], es decir, añadiendo más recursos a los nodos existentes (como procesadores, memoria RAM, disco, etc.).

Las arquitecturas de tipo monolíticas, por tanto, son muy útiles en aquellos casos donde, o bien las comunicaciones entre componentes son excesivamente complejas, o bien la cantidad de componentes que conforman la aplicación son demasiados (pudiendo llegar a situaciones de inestabilidad en las aplicaciones), o bien la eficiencia se encuentra afectada debido a la capa de intercomunicación entre componentes, entre otros casos [245]. Es claro el ejemplo de Amazon [112, 131] donde al tener una estructura de componentes muy compleja en su servicio de Amazon Prime Video, redujeron los costes del servicio en hasta un 90% al pasar de arquitecturas de tipo distribuidas a monolíticas.

2.2.2 Distribuidas

El paradigma que más se ha desarrollado en los últimos años y el que más se ha usado en la mayoría de nuevas aplicaciones (incluso consiguiendo que muchas compañías migren sus aplicaciones monolíticas) ha sido el de las arquitecturas distribuidas [183]. Este paradigma se ha podido desarrollar, principalmente, por la evolución de los sistemas de comunicaciones, más concretamente, por las capacidades de interconexión que ofrecen las redes globales de IP, es decir, Internet.

Este paradigma de arquitectura de aplicación se basa en que los componentes de una aplicación no se encapsulan en un único programa, sino en que una determinada aplicación consta de una serie de programas. Asimismo, cada uno de los distintos programas que conforman la aplicación se encarga de servir uno o más componentes del total que conforman el sistema. Además, los distintos programas se pueden encontrar en distintas unidades de cómputo, los cuáles, se comunican entre ellos haciendo uso de una red de comunicaciones (normalmente Internet).

Dentro de la aplicaciones distribuidas, existen ciertos programas que son específicos para que los usuarios de la aplicación puedan interactuar con la misma, mientras que hay otros que nunca interactuaran directamente con el usuario, únicamente con otros programas.

Asimismo, y debido a la interacción entre componentes, deben existir servicios y/o protocolos dentro de estos componentes que permitan la intercomunicación entre ellos.

La principal ventaja de los sistemas distribuidos es la capacidad que ofrecen a la hora de incorporar nuevas funcionalidades a la aplicación, ya que no hay que modificar la aplicación entera, de hecho, si la arquitectura está bien planteada, no hay que modificar la aplicación, únicamente basta con añadir un nuevo componente con la funcionalidad requerida.

Además, las arquitecturas de tipo distribuidas permiten realizar escalados de tipo horizontal [74], es decir, para añadir más recursos no hay que tener equipos (nodos) más potentes, sino más equipos. No solo eso, esta característica también permite que se puedan añadir recursos únicamente para los componentes que lo necesiten en vez de para toda la aplicación, consiguiendo ser más eficiente.

Otra ventaja que tienen las arquitecturas de tipo distribuidas es la de permitir la HA de forma sencilla, al únicamente tener que replicar aquellos componentes que requieran de dicha característica. Por otro lado, esta virtud también les sirve en aquellas aplicaciones que son accedidas desde distintos puntos geográficos, ya que se pueden distribuir los nodos que contienen los programas en granjas de

servidores alrededor del mundo y ofrecer latencias bajas a todos los usuarios, independientemente de cual sea el punto geográfico desde el que se realiza la petición.

En contrapartida y tal y como se ha mencionado anteriormente, los sistemas distribuidos necesitan una capa adicional que no necesitan los sistemas monolíticos, la capa de comunicaciones. Ésta se puede implementar o bien con procedimientos que deben cumplir todos aquellos componentes que quieran interactuar con otros de éstos o mediante un componente (o servicio) específico que se encargue de realizar la función de agente de comunicaciones.

Los sistemas distribuidos han evolucionado con el tiempo, por lo que existe una gran cantidad de definiciones de arquitecturas de sistemas distribuidos, aunque la vertiente probablemente más empleada empezó con las SOA [275] dando lugar finalmente a las arquitecturas basadas en microservicios [223], ambas analizadas a continuación.

SOA

Las SOA [275] son un tipo de arquitecturas distribuidas dentro del modelo cliente-servidor (en el que un cliente solicita una serie de recursos a un servidor) [193].

Las arquitecturas orientadas a servicios se basan en dividir una aplicación existente en múltiples servicios, los cuales, se encargan de dar solución a una necesidad específica de la aplicación general. Estos servicios continúan siendo específicos para la aplicación global y, además, no están desacoplados entre ellos. Para que los componentes se comuniquen entre ellos, este tipo de arquitecturas propone un componente de programa específico llamado bus de servicios empresariales (ESB, del inglés *Enterprise Service Bus*). Este componente ofrece mecanismos para que los servicios se registren y, una vez registrados, puedan enviar y recibir contenido por el mismo. Asimismo, se definen mensajes para la interacción entre los servicios [223, 275]. El protocolo de comunicaciones más empleado por este bus es el protocolo simple de acceso a objetos (SOAP, del inglés *Simple Object Access Protocol*).

Al implementar esta arquitectura se consigue que, sin perder la cohesión de los componentes entre si, se pueda desarrollar la lógica de la aplicación de forma independiente. Además, permite aplicar cambios en ciertos servicios de la aplicación sin tener en cuenta otra lógica de la misma, consiguiendo una mayor agilidad.

Microservicios

Las arquitecturas basadas en microservicios es el siguiente tipo de arquitecturas distribuidas a analizar. En muchos trabajos y estudios realizados en la actualidad se considera este tipo de arquitecturas como una forma especifica de implementación de las SOA [223]. En este tipo de arquitecturas, al igual que en las SOA, se busca dividir las aplicaciones en múltiples servicios, en este caso, denominados microservicios aunque, a diferencia de las SOA, éstos están desacoplados de la aplicación y se encargan de resolver tareas muy simples y muy especificas. Además, éstos se despliegan de forma independiente, con lo que cada una de estas partes de la división puede ser desarrollada y verificada de forma aislada y por diferentes equipos si fuese necesario. No solo eso, cada microservicio puede estar desarrollado con un lenguaje de programación distinto al de los otros microservicios si así fuese requerido (por ejemplo, por temas de eficiencia) [15, 223].

Otra característica considerada relevante, específica de los microservicios y diferenciable de los servicios de las SOA, es que los microservicios deben ser de pequeño tamaño, por ese motivo, las tareas que realizan son muy simples y muy especificas [59]. Asimismo, al tener dicho tamaño tan reducido, son más fáciles de mantener [15, 268] algo considerado muy importante al ser una de las limitaciones del desarrollo de programas desde sus orígenes [249].

Además, al forzar la distribución de la aplicación en múltiples servicios independientes, los cuáles pueden ser replicados, se consigue tanto una mayor tolerancia a fallos como la posibilidad de conseguir la HA en la aplicación de forma sencilla [253].

Los sistemas basados en microservicios también ofrecen la posibilidad de editar, añadir o eliminar funcionalidades a la aplicación (o a un servicio en específico) sin tener que modificar todos los servicios que la conforman, por tanto, es muy fácil y económico [35] mejorar y ampliar la aplicación, haciendo esta arquitectura perfecta para situaciones donde se producen muchos cambios a lo largo de la vida útil de la misma.

Asimismo, se conoce que, por regla general, no todos los componentes de una aplicación tienen la misma carga de trabajo, sino que dicha carga sigue la distribución de Pareto, de la cuál se infiere que el 80% de los resultados obtenidos como consecuencia del empleo de una cierta aplicación se obtienen mediante, únicamente, el 20% de los componentes de la misma [29]. Además, también es importante destacar que no todos los procesos necesitan el mismo número de recursos para funcionar de forma correcta. En este tipo de sistemas, al desplegar cada uno de los servicios de forma independiente, se pueden asignar los recursos acorde a las

necesidades especificas y reales de cada uno de los servicios en cada momento. Con esto se consigue que si hubiese un pico de uso para un servicio específico, solamente sería necesario escalar ese servicio en particular y no el resto de servicios, o lo que es lo mismo, la escalada es horizontal [74] y no es necesariamente uniforme [59, 223].

En contrapartida, y al igual que en las SOA, se necesita de un mecanismo de comunicación entre los microservicios. En este caso, para realizar dicho intercambio de mensajes no se emplea un ESB como en las SOA, sino que se emplea un componente conocido como el agente de comunicaciones. Éste, se encarga de recibir los mensajes de los distintos servicios y enviárselos al servicio al que van destinado (o a los servicios a los que van destinados).

2.3 Mecanismos de comunicación

En el mundo actual dónde todo, absolutamente todo, está interconectado, no existe la posibilidad de pensar que aquellos nuevos desarrollos, ya sea *hardware* o *software*, no contemplen en algún punto de su desarrollo ofrecer interoperatividad con otros sistemas.

Para conseguir el objetivo de realizar un intercambio de datos entre dos pares (es decir, entre un emisor y un receptor) o entre múltiples agentes, en la actualidad existe una gran cantidad tanto de paradigmas de comunicación como de protocolos de intercambio de información, los cuales, se analizarán a continuación.

2.3.1 Paradigmas de comunicación

Dentro de los distintos paradigmas de comunicación existentes en la actualidad cabría destacar los dos más factibles para las posibles fuentes de datos de las que pueda obtener información una aplicación que implemente la arquitectura definida en este trabajo, que serían [27, 216]:

- Paradigma de petición y respuesta
- Paradigma de publicación y suscripción

Paradigma de petición y respuesta

El paradigma de comunicación de petición y respuesta existe desde los inicios de las comunicaciones entre pares [38, 43]. Éste, es el protocolo más básico de una comunicación bidireccional.

En este paradigma, uno de los pares debe encontrarse en todo momento esperando una petición de comunicación. Cuando éste recibe una de éstas, procesa el contenido y devuelve una respuesta al remitente para finalizar la comunicación.

Con este paradigma de comunicación se consigue un intercambio fácil de información entre dos pares. En contrapartida, hay que ir realizando peticiones cada cierto tiempo para poder tener los datos actualizados. A la técnica de la realización de peticiones cada cierto tiempo para obtener la información actualizada se le llama sondeo (o *polling* en inglés).

Paradigma de publicación y suscripción

Por otro lado se encuentra el paradigma de comunicación de publicación y suscripción. Este paradigma surgió años después del paradigma de petición y respuesta, encontrando trabajos analizándolo a partir de los años 2000 [64, 156].

En este paradigma de comunicación no se busca una comunicación entre pares, sino que por el contrario se piensa una comunicación entre muchos a muchos.

En este paradigma se suele necesitar hacer uso de un servicio (o componente) específico que se encargue de recibir el contenido generado por las fuentes de información y enviarlo a los consumidores de información que lo necesiten. A este servicio se le llama agente de comunicaciones.

En primer lugar se encuentra la publicación. Cuando un generador de información tiene contenido que compartir con el resto de los participantes de la comunicación, esta fuente de información envía los datos al agente de comunicaciones. Estos datos están etiquetados con información adicional, como por ejemplo, un parámetro que le permita al agente de comunicaciones conocer cuales son los consumidores de información que pueden estar interesados en obtener dicho contenido.

En segundo lugar se encuentra la suscripción. Cuando un consumidor de información quiere obtener cierto tipo de contenido, le indica al agente de comunicaciones sus intereses y se queda escuchando al agente de comunicaciones esperando que le llegue información.

Este paradigma de comunicaciones es más complejo que el anterior, pero permite recibir información actualizada sin tener que estar realizando consultas de forma periódica. Asimismo, permite el desacoplo entre generadores y consumidores de la información.

2.3.2 Protocolos de intercambio de información

Una vez conocidos los distintos paradigmas de comunicación que son susceptibles de ser empleados, el siguiente paso es conocer cuales son los protocolos de intercambio de información actuales, así como el análisis de los mismos.

En el estado del arte actual se pueden encontrar gran variedad de protocolos a emplear, de los que se destacarían los siguientes [114, 197]:

- HTTP
- CoAP
- WebSocket
- MQTT
- AMQP

HTTP

El primer protocolo a analizar es el protocolo de transferencia de hipertexto (HTTP, del inglés *HyperText Transfer Protocol*), el cual fue definido en su versión 1.1 en la RFC 2616 [191] en el año 1999.

HTTP es probablemente el protocolo más usado en Internet al ser el empleado por navegadores web, servidores y aplicaciones web en todo el mundo [85].

HTTP se basa en el paradigma de comunicación petición y respuesta, habiendo un cliente que realiza una petición HTTP y recibiendo una respuesta de un servidor. Al realizar una petición HTTP, se le solicita un recurso específico al servidor. Un mismo recurso puede devolver y/o realizar distintas operaciones en el servidor dependiendo del método HTTP empleado, siendo los más comunes (pero no los únicos) `GET`, `POST`, `PUT` y `DELETE`.

En su versión 2 [20] se evoluciona de forma que una vez se ha establecido el intercambio de información entre el cliente y el servidor, se permita comunicación bidireccional simultánea, es decir, al mismo tiempo se pueden encontrar intercambios de mensajes originados tanto por el cliente como por el servidor.

El protocolo HTTP tiene una versión segura, el protocolo seguro de transferencia de hipertexto (HTTPS, del inglés *HyperText Transfer Protocol Secure*), en la que se emplea una capa de protección SSL/TLS por debajo. Además, el protocolo HTTP permite el uso de *proxies*, balanceadores de carga, etc.

CoAP

El protocolo de intercambio de información CoAP (del inglés, *Constrained Application Protocol*) se definió en la RFC 7252 [235] en el año 2014.

El protocolo CoAP se creó pensando en comunicaciones máquina a máquina (M2M, del inglés *Machine To Machine*), sobretodo para aquellos casos donde los recursos de los dispositivos son limitados [30, 137].

La definición está basada en el protocolo HTTP, aunque, para hacerlo más eficiente, se realizó una simplificación del mismo.

Junto con el protocolo HTTP, se basa en el paradigma de comunicación petición y respuesta. Además, comparte la petición por recursos y métodos. Cabe destacar que únicamente implementa los métodos HTTP más comunes, es decir, `GET`, `POST`, `PUT` y `DELETE`.

Por contrapartida, al realizar la simplificación anteriormente mencionada, no se pueden emplear ciertos servicios como los *proxies*, como si se puede con HTTP.

WebSocket

El protocolo *WebSocket* se definió en la RFC 6455 [173] en el año 2011, entre la definición de las versiones 1.1 y 2 de HTTP.

HTTP v1.1 no permite el intercambio bidireccional de mensajes, en cambio, debido al avance del desarrollo de soluciones empleando el entorno web, surgió la necesidad de que los servidores inicializaran comunicaciones hacía los clientes. Debido al problema anteriormente mencionado, se desarrolló el protocolo de comunicaciones *WebSocket* [159].

El protocolo *WebSocket* está pensado para trabajar conjuntamente con HTTP. Cuando una cierta aplicación desea establecer una comunicación bidireccional empleando el protocolo *WebSocket*, en primer lugar establece una conexión HTTP especifica para negociar el intercambio de información mediante *WebSocket* y, una vez se ha realizado esta negociación, ya se deja de usar HTTP.

Al ser un protocolo pensado para trabajar conjuntamente con HTTP, toda la infraestructura de HTTP (incluyendo los *proxies*) son compatibles con *WebSocket*, además, también comparte los puertos estándar con HTTP.

Además, al igual que HTTP, *WebSocket* también tiene una versión segura en la que se emplea una capa de protección SSL/TLS por debajo.

MQTT

El protocolo MQTT (del inglés, *Message Queuing Telemetry Transport*) fue definido en el año 1999 y originalmente fue creado para comunicar oleoductos via satélite haciendo uso del mínimo ancho de banda posible y optimizando la duración de las baterías de los emisores y receptores [34].

MQTT es un protocolo de intercambio de información muy ligero, lo que lo hace perfecto para aplicaciones M2M. A diferencia de los tres protocolos anteriores, se basa en el paradigma de publicación y suscripción, por tanto, lo hace perfecto para la comunicación de datos en tiempo real.

Este protocolo emplea temas (o en inglés, *topics*) a los que se suscriben los consumidores de información y donde publican las fuentes de información. Estos temas tienen una estructura jerárquica, además, permiten el uso de caracteres comodín para la suscripción a múltiples temas que cumplan un cierto patrón.

Además, a pesar de estar pensado para ser simple, también busca asegurar que los intercambios de mensajes sean satisfactorios. Para ello, emplea un parámetro al que llaman calidad de servicio (QoS, del inglés *Quality of Service*) el cual define cuanto tiempo una cierta fuente de datos debe almacenar un dato desde que lo envía.

AMQP

El protocolo AMQP (del inglés, *Advanced Message Queuing Protocol*) se creó en el año 2006 buscando ser un bus de datos de alto rendimiento, abierto y sin derechos [192].

Al igual que pasa con MQTT, el protocolo AMQP también se basa en el paradigma de publicación y suscripción.

A diferencia del protocolo MQTT, AMQP no busca ser ligero sino, por el contrario, busca ser tan completo como sea posible para cubrir todas las posibles necesidades que le puedan surgir a un sistema que haga uso del mismo como su protocolo para intercambiar la información.

Este protocolo tiene un sistema complejo para hacer llegar los mensajes de los publicadores a los consumidores. En primer lugar, define colas, las cuáles, son los destinos finales de los mensajes. Para enviar mensajes a esas colas (o en inglés *queues*), los generadores de información publican su contenido mediante intercambiadores (o en inglés *exchanges*), los cuáles, emplean vínculos (o en inglés *bindings*) para conocer cuál es la cola de destino.

2.4 Recursos actuales

Una vez se conocen tanto las distintas arquitecturas de aplicación como los distintos mecanismos de comunicación empleados en la actualidad, el siguiente paso consiste en analizar cuales son los recursos actuales que se pueden emplear para los distintos componentes que puede tener una arquitectura de aplicación para un sistema de caza de amenazas basado en AI. Tras estudiar los estudios expuestos en la sección anterior [10, 16, 81, 95, 105, 113, 189, 209, 228, 229], se podrían clasificar dichos componentes en:

1. Adquisición de los datos
2. Modelado de los datos
3. Preparación de los datos para Inteligencia Artificial
4. Procesado de los datos mediante Inteligencia Artificial
5. Intercambio de los datos
6. Interacción con los datos

2.4.1 Adquisición de los datos

Toda aplicación que vaya a interactuar con datos necesita de un componente específico encargado de realizar la adquisición de los mismos, es por ello, que el primer tipo de componentes a desarrollar y analizar sea el de adquisición de datos [50].

Para realizar la adquisición de los datos, en la literatura se encuentran los elementos conocidos como *wrappers* [57, 218], éstos, tienen dos funciones principales [55], las cuáles, se estudian con detalle a continuación.

En primer lugar, estos elementos, los *wrappers*, entendidos dentro del contexto de un componente de adquisición de datos, se encargan de comunicarse con las distintas fuentes de datos, para ello, son capaces tanto de realizar peticiones a dichas fuentes de datos en el formato que éstas las esperan, como de recibir la respuesta proporcionada por dichas fuentes y tener capacidades de procesado de las mismas. Hay que tener en cuenta que estos componentes deben ser capaces de emplear cualquiera de los paradigmas de comunicación y protocolos de intercambios de información detallados en secciones anteriores, ya que, potencialmente, cada fuente de datos empleará un paradigma de comunicación y un protocolo de intercambio de información distinto.

En segundo lugar, estos componentes también deben ser capaces de aplicar las transformaciones necesarias a las respuestas recibidas, con ello, aportan aspectos tan importantes como flexibilidad y generalidad en cuanto a la estructura de datos se refiere, algo imprescindible en cualquier aplicación que busque poder ser empleada en un ámbito de aplicación como es la caza de amenazas. Por tanto, y dado que estos componentes son capaces de procesar las respuestas del las peticiones realizadas a las fuentes de datos, las cuáles, potencialmente, son distintas para cada una de las fuentes de las que se realice la adquisición, estos componentes también tienen la tarea de homogeneizar la información recibida a una estructura de datos conocida por el resto de componentes de la aplicación, la cuál se conoce como modelo de datos, consiguiendo la flexibilidad y generalidad buscada.

2.4.2 Modelado de los datos

Para poder definir el modelo de datos que se va a emplear por una aplicación específica, es decir, la colección de atributos y tipos de atributos con los que describir los datos con los que interactúa y trabaja una aplicación o sistema [71, 198], es importante realizar un análisis de las ontologías existentes, es decir, de la definición genérica de los distintos elementos que conforman una cierta aplicación [241].

Aunque la definición del modelo de datos no sea la parte del desarrollo que más tiempo cueste, si que es una de las partes más críticas, ya que si éste está bien definido tendrá un gran impacto positivo en el producto final, mientras que si está mal definido tendrá un gran impacto negativo [181], por ese motivo, toda aplicación que busque ser eficiente, que busque ser flexible a la hora de recibir modificaciones en un futuro y que busque perdurar en el tiempo ofreciendo servicios de gran calidad, debe tener como base un buen modelo de datos.

Debido a este motivo, en la literatura se pueden encontrar ontologías de prácticamente cualquier ámbito, por tanto, es muy importante analizar cuáles de las existentes, con sus beneficios y sus perjuicios, son las mejores para ser empleadas en una aplicación específica, ya que si existe una definición muy bien desarrollada para un cierto elemento de la aplicación, no hace falta invertir tiempo en volver a realizar dicho trabajo, en el cuál, además, se podría omitir aspectos relevantes de dicha definición.

En el ámbito de la ciberseguridad se puede encontrar gran variedad de ontologías ya definidas. Algunos ejemplos de éstas podrían ser CRUSOE [132], SEPSES [128], STUCCO [109], VERIS [32, 182], la propuesta de ontología especificada en [97], la propuesta de ontología especificada en [92] y CAPEC [165], entre otras.

Por otro lado, se pueden encontrar ontologías más generalistas, como por ejemplo, GFO [100] y ECS (Elastic Common Schema) [77], entre otras.

2.4.3 Preparación de los datos para Inteligencia Artificial

Como bien es sabido, las acciones que realiza toda aplicación que interactúa con datos son, por ejemplo:

- Almacenar el estado de la propia aplicación
- Generar inteligencia a partir de ellos
- Representarlos

Ahora bien, incluso en el caso de estar homogeneizados a un modelo, los datos en crudo (o en *raw*, como se dice inglés) pueden necesitar de una serie de procesos que los preparen para que sean útiles y así se pueda extraer información de los mismos.

Los procesos de preparación de los datos existen desde que hay datos [203], aunque, para este trabajo, se analizarán aquellos que son específicos para darles valor y para que puedan ser empleados por técnicas de AI [33, 195], más específicamente, aquellos que son también útiles para realizar el *Threat Hunting*.

Limpieza de datos

La limpieza de datos es uno de los procesos más importantes (sino el que más) a realizar en la preparación de datos para AI.

No todos los datos obtenidos de las fuentes aportan información a las técnicas de AI, más concretamente, a las técnicas de ML y DL, no solo eso, incluso pueden llevar a impactar de forma negativa en el modelo de predicción, en consecuencia, es primordial emplear únicamente aquella información que realmente es útil.

Para el proceso de la limpieza de datos no se requiere tanto el tener conocimientos de programación avanzados (con los que realizar métodos que sean eficientes a la hora de realizar dicha limpieza) sino conocer el área de estudio y la tipología de los datos a tratar [122].

Dentro de este proceso se podrían definir una serie de tareas a realizar, aunque no son las únicas.

Una de las tareas debería identificar y eliminar aquellos atributos de los datos que no aportan información por tener el mismo valor (o valores similares) en la totalidad de la serie.

También se podrían realizar estudios de correlación cruzada de los atributos de los datos, ya que, aquellos que se encuentran altamente correlacionados, aportan información muy similar entre ellos, por tanto, eliminarlos mejora el rendimiento.

Otra tarea importante a realizar sería identificar aquellos datos del conjunto que están repetidos y eliminarlos, al no aportar información nueva.

Transformación de datos

El siguiente de los procesos a realizar consiste en la transformación de los datos.

Ciertas técnicas de AI únicamente son capaces de recibir como entradas valores numéricos, pero se puede dar el caso en el que el contenido obtenido de la fuente de datos no sean valores de tipo numéricos, sino otro tipo de datos. En estos casos, hay que realizar una preparación de los datos que sea capaz de generar relaciones entre el valor original y uno o varios valores numéricos y, si fuese requerido, el proceso inverso, de los valores numéricos al valor original.

Una de las técnicas que se emplean en la actualidad para realizar dicha transformación de los datos se llama codificar. Existen muchos codificadores distintos dependiendo de la codificación que empleen, como por ejemplo, aquellos que empelan la codificación de etiquetas (o en inglés, *label encoding*) [96] en la que se pasa de hacer uso de palabras a emplear un valor numérico natural. Cabe destacar que ésta no es la única existente, ya que se pueden encontrar otras alternativas como, por ejemplo, los *One-Hot encoders* [96] y los *Leave-one-out encoders* [96].

A pesar de que las soluciones anteriormente mencionadas son simples y fáciles de integrar, algunas de éstas pueden llevar asociado problemas de sobredimensionamiento. Como solución, en la literatura se encuentran alternativas basadas en emplear un aprendizaje de los datos para realizar dicha transformación sin necesidad de aumentar el dimensionamiento. Algunas de las técnicas propuestas serían el análisis de componentes principales (PCA, del inglés *Principal Component Analysis*) o el análisis discriminante lineal (LDA, del inglés *Linear Discriminant Analysis*) [22, 282], entre otras.

Normalización numérica

A continuación se encontraría la preparación de los datos que son de tipo numérico.

Ciertos tipos de técnicas de AI pueden requerir que los datos numéricos lleguen siguiendo unos requerimientos para que puedan funcionar de forma satisfactoria.

El caso más común que se busca es que la técnica de AI trate a todos los atributos por igual, para ello, se pueden emplear técnicas como la normalización de los valores numéricos (a media 0 y varianza 1) o como el escalado de los valores máximos y mínimos, para que todos se encuentren en el mismo rango.

Otro caso que se puede encontrar es el contrario, si lo que se busca es que las técnicas de AI se centren más en unos atributos que en otros, lo que se hará es hacer más difícil el aprendizaje de los atributos en los que se quiere que la técnica de AI se centre.

Normalización de textos

Otro proceso de preparación de datos importante es el que se lleva a cabo con los textos.

Es sistemas que trabajan con *Threat Hunting* es bastante común el análisis de los *logs* mediante AI, por tanto, hay que optimizar estos procesos con una correcta preparación de la información adquirida de las fuentes de datos.

Dentro de los distintos procesos que existen para la normalización de textos, se puede encontrar uno consistente en la segmentación de sentencias haciendo uso de los límites comunes. En este proceso se suelen emplear los puntos, aunque siempre teniendo en cuenta que estos caracteres también se usan para denotar abreviaciones [89].

Otro de los procesos que se encuentran dentro de la normalización de textos es el consistente en la separación de las sentencias en los distintos componentes que la conforman [220, 240], como palabras, artículos, pronombres, etc.

También se puede encontrar el proceso de lematización (en inglés *Stemming* o *Lemmatization*) [116, 160] que consiste en la extracción de la raíz de las palabras mediante la separación de los prefijos y sufijos. Por ejemplo, si se aplicase este proceso a las palabras visual, visualmente y visualizado, para todas ellas devolvería la raíz, es decir, visual.

Asimismo, hay veces que interesa procesar únicamente la información que se considera que puede aportar valor y no analizar los enlaces entre las distintas sentencias, por tanto, existe un proceso que consiste en eliminar lo que se denomina como palabras de parada (en inglés *stop words*) [124], como podrían ser las palabras:

- a
- actualmente
- adelante
- además
- y
- ya

2.4.4 Procesado de los datos mediante Inteligencia Artificial

Una vez los datos han sido preparados para ser empleados con técnicas de AI, el siguiente paso es emplear dichas técnicas para procesarlos y generar información e, incluso, inteligencia, para así poder realizar la caza de amenazas de los sistemas que están siendo monitorizados.

Bien es sabido que las técnicas de AI no son algo nuevo y, a lo largo de su larga evolución, su desarrollo se ha ido centrando en todos y cada unos de los ámbitos imaginables [42], por tanto, no todas las técnicas existentes sirven para todos los casos de uso. Debido a este hecho, el estudio de las técnicas de procesado de datos mediante AI se ha centrado en aquellas que se pueden emplear para realizar la caza de amenazas. De estas, cabe destacar las técnicas incluidas dentro de los siguientes tipos de las mismas, aunque no son las únicas que se pueden emplear:

- Agrupadores
- Redes neuronales
- Procesado del Lenguaje Natural
- Modelos de lenguaje de gran tamaño

Agrupadores

El primero de los tipos de técnicas de AI que pueden ser útiles para la caza de amenas son los agrupadores. Estas técnicas buscan generar grupos de datos que contienen atributos similares entre si.

Algunas técnicas que pueden ser útiles (pero no todas) serían:

- K-means
- Affinity Propagation
- Mean Shift
- Spectral Clustering
- Hierarchical Clustering
- DBSCAN

K-means. K-means es una técnica de agrupación que se basa en la generación de un número k de agrupaciones, por eso del nombre K-means.

Cada una de las agrupaciones consta de un centroide y la técnica busca localizar dicho centroide en un espacio N-dimensional teniendo como objetivo conseguir el mínimo cuadrado de la distancia euclidiana [106].

El principal inconveniente que tiene esta técnica es que sufre de problemas de inicializaciones aleatorias y de convergencias a mínimos locales en su implementación original.

Para solucionar dichos problemas, existen alternativas a dicha implementación original como *Ellipsoidal K-means*, *Bisecting K-means* [6, 280], entre otras.

Affinity Propagation. *Affinity Propagation* es una técnica que busca encontrar cuales son los datos que mejor ejemplifican el conjunto de datos que ha sido procesado.

Dependiendo de como se configuren los distintos parámetros de la técnica, se pueden encontrar oscilaciones que no consigan el resultado esperado.

Se han propuesto alternativas a la implementación original para tratar de solucionar dichos problemas como, por ejemplo, *Hierarchical Affinity Propagation Clustering* [78] y *Adaptive Affinity Propagation Clustering* [271], entre otras.

Mean Shift. *Mean Shift* es una técnica de agrupaciones que funciona de forma muy eficiente para espacios con pocas dimensiones.

La principal ventaja de esta técnica es que no necesita parámetros de configuración [52].

En contrapartida, no es muy buena trabajando con espacios con muchas dimensiones. Para hacer que trabaje en estas condiciones, existen alternativas a la técnica original que si que lo permiten. Un ejemplo de modificación consiste en combinar técnicas de aproximaciones jerárquicas con *mean shift* [51].

Spectral Clustering. La técnica *Spectral Clustering* se ha convertido en una de las más populares en la actualidad, llegando incluso a sustituir técnicas más tradicionales como *k-means* en ciertos casos de uso.

La principal ventaja que presenta esta técnica de agrupaciones es que es fácil de implementar y, además, para generar las agrupaciones, emplea funciones de optimización cuya solución se puede obtener empleando una serie de cálculos matemáticos, los cuáles, son fáciles de resolver mediante programas de álgebra lineal.

No obstante, las técnicas, aunque buenas, siempre pueden mejorarse. De ésta en particular existen distintas variaciones como podrían ser la *Unnormalized Spectral Clustering* y la *Normalized Spectral Clustering* [270].

Hierarchical Clustering. Se denominan técnicas de agrupación jerárquica (o en inglés, *Hierarchical Clustering*) a aquellas que consisten en la obtención de la raíz de un árbol cuyas hojas son los elementos del conjunto de datos [190].

De este tipo de técnicas, se considera importante destacar el parámetro "función distancia base", que es el empleado para calcular la distancia en el espacio entre los distintos elementos del conjunto de datos, es decir, su desemejanza.

Un ejemplo de técnica de agrupación jerárquica podría ser *m-ADIC clustering* [185].

DBSCAN. La técnica de agrupamiento espacial basado en densidad de aplicaciones con ruido (DBSCAN, del inglés *Density-Based Spatial Clustering of Applications with Noise*) tiene como característica principal su capacidad de generación de grupos de cualquier tipo de forma y tamaño sin importar si hay ruido o si hay elementos atípicos.

Si se desea emplear esta técnica, hay que tener en cuenta que hay una serie de parámetros a configurar para su funcionamiento correcto, así como que en su ejecución se requiere de una serie de elementos computacionales complejos [125].

Para solucionar los inconvenientes anteriormente enunciados, existen alternativas como VDBSCAN (DBSCAN variada) [154] y FDBSCAN (DBSCAN rápida) [283], entre otras.

Redes neuronales

Dentro de todo aquello que engloba la AI, existe un subconjunto denominado ML [219] que busca no solo ser capaz de reproducir aquello para lo que ha sido entrenado, sino además tener la capacidad de generar modelos de los conjuntos de datos empleados para su aprendizaje.

De alguno forma, el ML busca ser capaz de imitar al cerebro humano y, a partir de una entrada, poder deducir una salida, empleando como referencia, la información con la que ha sido entrenado con anterioridad, por este motivo, una de las técnicas que se emplean dentro del ML son las conocidas como redes neuronales.

Las redes neuronales consisten en un elemento inicial con un conjunto de entradas (o neuronas), conocido como capa de entrada; un elemento final con un conjunto de salidas (o neuronas), conocido como capa de salida; y un conjunto de elementos entre la capa inicial y la final dónde, cada uno de éstos contiene sus propias neuronas, conocidos como capas intermedias. Éstas, las capas intermedias, son las que se encargan de aprender de la información procesada durante la fase de entrenamiento. En la figura 2.1 se encuentra un ejemplo gráfico de los elementos anteriormente descritos.

Cabe destacar que los parámetros de la capa de entrada, los parámetros de la capa de salida y tanto el número de capas intermedias como el número de neuronas por capa intermedia, deben ser definidos por el especialista en ML.

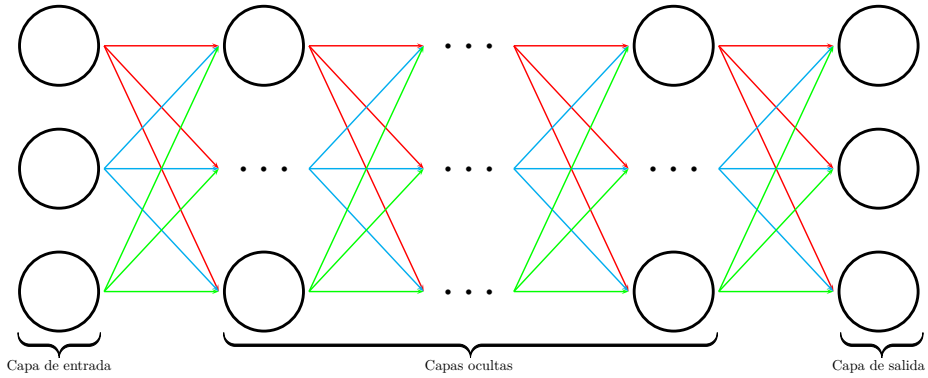

Figura 2.1: Red Neuronal: Definición

Redes neuronales hay muchas y de muchos tipos dependiendo del de contenido que se espera que aprendan. Dentro de este gran conjunto, para una aplicación de caza de amenazas, se pueden destacar los siguientes tipos de redes neuronales, aunque no son lo únicos que se pueden emplear:

- Graph Neural Networks
- C-RNN-GAN
- LSTM RNN
- BiLSTM

Graph Neural Networks. Cuando se trata de análisis de datos complejos, como podrían ser imágenes o vídeos, es bastante común emplear redes neuronales basadas en capas dónde un núcleo con una forma (multidimensional) específica recorre el conjunto de los datos (a través de todas las dimensiones) y le aporta la información obtenida a la siguiente capa. Este tipo de redes es posible gracias a propiedades específicas de este tipo de datos, aunque no es aplicable para todos los datos complejos.

Cuando se trata de otro tipo de datos complejos (como podrían ser los datos obtenidos de fuentes de redes sociales, los datos de tipo financiero, los datos de comunicaciones IP, etc.) que siguen, normalmente, relaciones de tipo nodo-arista, es decir, relaciones de tipo grafo, las redes neuronales más indicadas son las llamadas redes neuronales de tipo grafo.

Debido a la gran cantidad de datos que existen de tipo grafo, se han hecho grandes esfuerzos para diseñar redes eficientes para trabajar con este tipo de datos. Algunos de estos ejemplos podrían ser *GNNExplainer*, *GNN-LRP* y *PGExplainer* [278].

C-RNN-GAN. La técnica de ML de redes neuronales recurrentes con entrenamiento generativo continuo basado en adversarios (C-RNN-GAN, del inglés *Continuous Recurrent Neural Networks with Generative Adversarial Training*) [179], tal y como se puede deducir de su nombre, se basa en establecer una competición entre redes neuronales, para ello, se emplean dos de éstas, las cuales, a medida que avanzan los entrenamientos, ofrecen resultados cada vez más precisos.

La primera de las dos redes se denomina como red discriminadora. Ésta tiene como objetivo calcular cuál es el grado de similitud existente entre los dos elementos que ha recibido a su entrada. Cabe destacar que cuanto más parecidos hayan sido los elementos empleados para generar el modelo de la red neuronal, más potente será la capacidad discriminadora de ésta.

Por otro lado existe una red neuronal la cuál busca devolver como resultado un dato los más parecido al que ha recibido a su entrada. Ésta se conoce como la red generadora. Para generar el modelo de la red neuronal, se emplea el grado de similitud generado por la red discriminadora (a veces conocido como error de discriminación).

Al estar ambas redes enlazadas entre si, a medida que progrese el entrenamiento de las redes neuronales, se generarán datos cada vez más parecidos a los reales, con la consecuencia de que los datos serán cada vez más parecidos entre si y, por tanto, el grado de similitud será cada vez más preciso.

Un caso de uso de este tipo de redes es, por ejemplo, el de generar un modelo del comportamiento normal de un sistema. De un lado, con la red generadora se consigue un modelo muy aproximado de dicho comportamiento del sistema, con lo que se pueden generar datos sintéticos muy parecidos a los reales. Por otro lado, con la red discriminadora se puede generar un clasificador entre datos normales del sistema y datos anómalos [143].

LSTM RNN. Las redes neuronales recurrentes de memoria a corto plazo largo (LSTM RNN, del inglés *Long Short-Term Memory Recurrent Neural Networks*) son uno de los clasificadores dinámicos más potentes que existen.

Dependiendo de la complejidad de la red neuronal generada, este tipo de redes pueden contar con más de 1000 capas ocultas si fuese necesario [237, 243].

BiLSTM. A las redes neuronales de tipo LSTM RNN se les detectó un problema el cual consiste en que cuando la secuencia que se está analizando es muy larga, la información introducida al inicio de la secuencia se perdía, por este motivo, se generaron las redes neuronales LSTM bidireccionales (BiLSTM, del inglés *Bidirectional LSTM RNNs*) [236, 237].

Este tipo de redes neuronales consiste en emplear dos LSTM RNN de forma simultánea, con la peculiaridad de que a una de ellas los datos introducidos van en orden ascendente (es decir, empezando por el primer dato y finalizando en el último dato) y, en la otra, los datos vas en sentido descendente (es decir, empezando por el último dato y finalizando en el primer dato).

Con este tipo de redes se mejoraron las LSTM RNN para aquellos conjuntos de datos muy grandes.

Procesado del Lenguaje Natural

Otro de los tipos de técnicas de AI que se pueden emplear para realizar la caza de amenazas es el de procesado del lenguaje natural (NLP, del inglés *Natural Language Processing*).

Como anteriormente se ha indicado, las fuentes de datos que alimentan una aplicación de caza de amenazas son muchas y muy variadas. Dentro este conjunto tan amplio, se considera importante la obtención de información de registros (ya sea de aplicaciones o equipos de red, etc.), la obtención de reportes (de inteligencia, etc.), así como cualquier otro elemento que pudiera ser generado por un especialista de ciberseguridad. Además, dado que toda esta información (registros, reportes, etc.) se puede interpretar como datos textuales [150], es posible emplear técnicas de NLP para generar información (o incluso inteligencia) de dichos datos mediante AI [76, 169, 261].

Algunas de las técnicas, aunque no las únicas, de NLP que se pueden emplear para realizar la caza de amenazas son:

- BERT
- CyBERT
- MalBERT
- TF-IDF

BERT. La técnica de representación de codificador bidireccional de transformadores (BERT, del inglés *Bidirectional Encoder Representations from Transformers*) es una de las técnicas de NLP que se podrían emplear en caza de amenazas [80, 237].

Esta técnica se caracteriza por ser capaz de realizar clasificación de textos sin necesidad de requerir supervisión humana.

CyBERT. A partir de BERT, al igual que ha pasado con otras técnicas de AI (como por ejemplo, las técnicas K-means o LSTM RNN anteriormente descritas) se han realizado evoluciones que la han convertido en específica para buscar solución a un problema determinado o a un área de trabajo concreta. En este caso, la técnica CyBERT busca aportar un modelo de BERT específico para la ciberseguridad.

Los ámbitos de aplicación de CyBERT son muchos y muy variados, entre otros, algunos ejemplos de estos serían ayudar en las tareas que se llevan a cabo en los centros de operaciones de seguridad (SOC, del inglés *Security Operations Centers*) y mejorar la ciberseguridad en los sistemas de control industrial [11, 212].

MalBERT. Al igual que con CyBERT, MalBERT [206, 207] es una evolución de BERT, en este caso, aplicado a la detección de código malicioso.

Gracias al empleo de técnicas como MalBERT, se reduce la sobrecarga que se produce en los sistemas de detección de código malicioso tradicionales, con la consecuente mejora en el tiempo de detección de este tipo de ejecutables no benignos.

TF-IDF. Otra técnica útil de NLP podría ser TF-IDF (del inglés, *Term Frequency Inverse Document Frequency*).

Esta técnica se caracteriza por ser capaz de calcular, de forma estadística, cuan significante es una palabra específica dentro de un texto [47].

La información generada por esta técnica pueden ser muy útil, por ejemplo, para la detección de URLs maliciosas, entre otras aplicaciones [141].

Modelos de lenguaje de gran tamaño

El último de los tipos de técnicas de AI que ha sido desarrollado y que, además, ha sido uno de los que más repercusión mediática ha generado por emplearse en herramientas como ChatGPT [164] de OpenAI, Anthropic Claude [153], Google Gemini [222] y Perplexity AI [126] así como en las desarrolladas por Microsoft en dicha área, como pueden ser Bing Chat AI [123] y Copilot [4] (tanto en su versión generalista como en la versión específica aplicada al desarrollo de software en la plataforma GitHub, también propiedad de Microsoft) han sido los modelos de lenguaje de gran tamaño (LLM, del inglés *Large Language Models*).

Se podría decir que este cambio de paradigma se ha producido debido a una nueva perspectiva dentro del procesado del NLP (aunque no limitado en exclusiva a este área) contribuida, fundamentalmente, por un trabajo de 2017 que revolucionó el enfoque para el tratamiento del procesado de secuencias (y, dentro de este procesado, la predicción del siguiente elemento en una secuencia). Este trabajo es el conocido "Attention is all you need" [262] que proponía la arquitectura conocida como "Transformers" y que ha acabado siendo completamente disruptiva en muchos aspectos, incluyendo la descomunal explosión de los LLM en los últimos años dentro de la AI, y siendo uno de los temas más destacados debido a la relevancia de las soluciones previamente citadas.

En la figura 2.2 se muestra una representación de dicha arquitectura propuesta en la que se considera importante destacar la modularidad de la misma y como, gracias a ésta, los sistemas de AI desarrollados a partir de ella pueden centrarse tanto en la codificación de conjuntos de datos de un modelo (*encoders*) como en la generación de información y/o inferencia sobre los datos de un modelo (*decoders*).

Esta arquitectura, la cuál ha permitido la proliferación de aplicaciones disruptivas y de uso mundial (como la anteriormente citada ChatGPT), no fué desarrollada en 2017, si no que es el fruto de muchos trabajos previos que, además, constituyen piezas relevantes de la misma, como el uso de los *word embeddings* [61], el *Fast Weight Controller* [227], el desarrollo de las técnicas (también empleadas en los

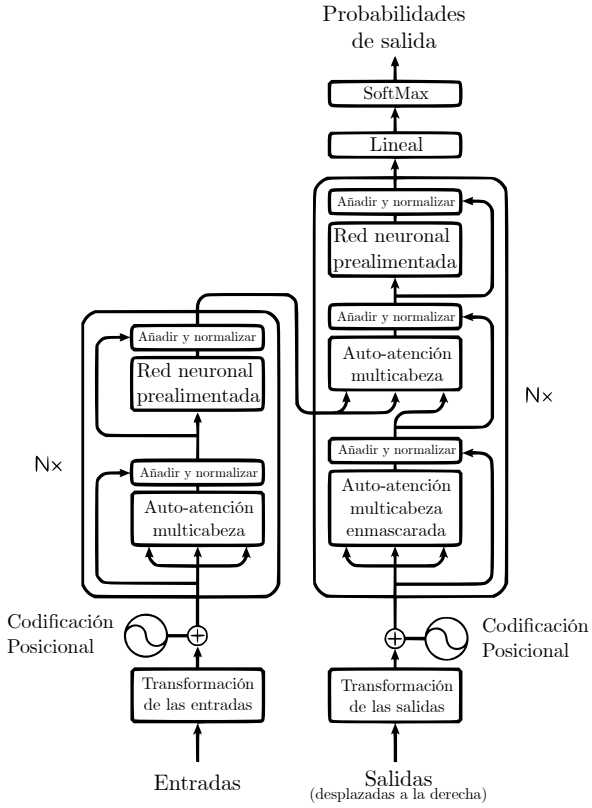

Figura 2.2: Modelos de lenguaje de gran tamaño: Arquitectura Transformers

tipos de técnicas de ML de redes neuronales) de memoria a corto plazo largo (LSTM, del inglés *Long Short-Term Memory*) [103] y los avances en modelos secuencia a secuencia [248], entre otros, y siendo [262] el que dió con la clave para convertirla en la base de los sistemas anteriormente destacados, principalmente, con la incorporación de los mecanismos de auto-atención.

Otra característica muy destacada de esta arquitectura es que los modelos desarrollados con la misma, en la gran mayoría de los casos, son entrenados con conjuntos de datos de tamaños descomunales (del orden de miles de millones o centenares de miles de millones de datos de entrenamiento), y, de ahí, que se les llame comúnmente LLM.

Como se puede observar en la figura 2.2, la arquitectura se divide principalmente en dos partes, a la izquierda, un *encoder* y, a la derecha, un *decoder*. En ambas, el primer paso consiste en realizar una transformación del contenido recibido mediante *word embeddings* para generar una representación vectorial de las información recibida, consiguiendo que aquella información con sentido semántico similar se encuentre cerca en dicho espacio vectorial. A continuación, y debido a como funciona la arquitectura "Transformer", para no perder la posición que ocupa cada una de las palabras en la frase, se añade un metadato con dicha información a las mismas en el componente de codificación posicional. A continuación se encuentra el componente de auto-atención, el cuál es capaz de permitir al modelo conocer con que otra palabra de la secuencia está relacionada la palabra que está siendo procesada. Para proseguir, se procesa la información mediante las redes neuronales prealimentadas (estas fueron las primeras redes neuronales generadas y las más simples), en las que la información únicamente avanza hacía adelante. Como se puede observar, la salida del *encoder* es también la entrada de uno de los componentes de auto-atención del *decoder*. En el *decoder*, su salida finaliza en una capa de tipo lineal y una función de activación *SoftMax* para obtener la probabilidad de la siguiente palabra y, una vez se ha obtenido la probabilidad de todas las palabras, devolver aquella palabra con la probabilidad más alta como la siguiente palabra.

El espectro de posibilidades y aplicaciones desarrollables con esta aproximación no está limitado al NLP, hay toda una serie de áreas de ML que se han beneficiado de la misma, como pueden ser el procesado automático del lenguaje (ASR, del inglés *Automatic Speech Recognition*), la generación automática de audio a partir de texto (TTS, del inglés *Text To Speech*) y la detección de elementos y patrones en imágenes para conseguir responder a peticiones textuales de información sobre las mismas como si lo hiciese un humano.

Además, otro aspecto relevante es el de la AI generativa que, empezando desde las bien conocidas arquitecturas de C-RNN-GAN [179], ha experimentado un impulso muy relevante debido a las arquitecturas "Transformers" y los LLM, por ejemplo, en el área del NLP, modelos basados en las mismas se han convertido en herramientas muy relevantes a la hora de generar, tanto de manera automática como bajo el guiado textual del usuario, texto de alta calidad en cualquier área, código eficiente y orientado a la disminución de vulnerabilidades (para cualquier contexto y lenguaje de programación) y generación automática de música.

Cabe destacar que los LLM no se han limitado al NLP, también se ha empleado para la generación automática de imágenes (específicamente en tareas de texto a imagen) así como tareas de texto a vídeo, donde cabe destacar la muy reciente aparición del revolucionario sistema de OpenAI, DALL-E [263].

Es importante remarcar que, en paralelo con la aparición de sistemas y plataformas en línea de los grandes actores tecnológicos (como por ejemplo, Google, Microsoft, Facebook y OpenAI), también han proliferado gran número de esfuerzos por parte de la comunidad de programas de código abierto (OSS, del inglés *Open Source Softwares*) para poder hacer llegar al gran público y desarrolladores LLM de libre uso, como pueden ser LangChain [259] y LlamaIndex [155].

2.4.5 Intercambio de los datos

Desde los inicios de la ciberseguridad, el intercambio de conocimiento entre agencias ha sido un elemento clave para garantizar el mejor de los resultados cuando se trata de prevenir y reaccionar frente a un ataque [46]. Hoy en día, cuando éstos (los ataques) no solo no se han reducido en número, sino que al revés, han aumentado tanto en número como en peligrosidad, se considera importante que toda aplicación de ciberseguridad (y, en su defecto, toda aplicación de caza de amenazas) debe ser capaz de intercambiar información (tanto proveer como suministrarse) con otras aplicaciones de ámbitos similares. En consecuencia, al realizar un correcto intercambio de datos, se logra proveer al especialista de ciberseguridad de una ciberconsciencia situacional completa y, por tanto, se consigue acelerar el rendimiento y la velocidad de toma de decisiones, pieza crítica cuando se trata de resolver incidentes de ciberseguridad.

Cuando se trata de enviar y recibir información desde y hasta otras agencias, potencialmente, cada una de estas emplee una aplicación distinta para realizar la caza de amenazas (o incluso para conducir cada uno de los distintos pasos necesarios para protegerse frente a amenazas cibernéticas), por tanto, tienen que ser conocidos por ambos extremos de la comunicación tanto los mecanismos de comunicación a emplear, como las representaciones de los datos, como la estructura del contenido a intercambiar, es decir, hay que emplear estándares en todos los niveles que puedan intervenir en dicho intercambio de información.

Analizando las distintas opciones en cuanto a mecanismos de comunicación se refiere, en la actualidad, el más extendido para enviar y recibir información sería aquel que emplease el paradigma de comunicación de petición y respuesta y trabajase sobre el protocolo de intercambio de información HTTP, además, sobre este mecanismo de comunicación, se emplearía la interfaz de comunicación API REST [168].

Aunque el mecanismo de intercambio de información es bastante claro, tanto en las representaciones de los datos como en la estructura del contenido a intercambiar, se puede encontrar una mayor variedad.

Empezando con las representaciones de los datos, las dos más frecuentes en la actualidad son JSON (del inglés, *JavaScript Object Notation*) y XML (del inglés, *Extensible Markup Language*) [86].

La representación de los datos XML es la más antigua de las dos más empleadas en la actualidad. XML se continua manteniendo en muchos sistemas a día de hoy, entre otros motivos, debido a:

- La capacidad de adición de metadatos
- Los sistemas de verificación de la estructura de datos.

La representación de los datos JSON, a pesar de surgir más tarde que XML, es la más extendida en la actualidad. De los distintos motivos por los que se prefiere JSON a XML, se podría destacar:

- La facilidad que tienen los programas para analizar el contenido
- El alto rendimiento
- La rapidez y ligereza
- La capacidad de soporte a tipos de datos y conjuntos

Prosiguiendo con la estructura del contenido, actualmente se pueden encontrar varios estándares como STIX [17, 232], SCAP (del inglés, *Security Content Automation Protocol*) [205, 232] y CyberDEM (del inglés, *Cyber Data Exchange Model*) [272].

En primer lugar se encontraría STIX. Esta estructura de contenidos se caracteriza por describir información acerca de una amenaza cibernética. Ha sido desarrollado por MITRE, la organización encargada de mantener la base de datos de vulnerabilidades y exposiciones comunes (CVE, del inglés *Common Vulnerabilities and Exposures*) [172, 186], entre otros. Además, se ha convertido en el estándar *de facto* de hoy en día para inteligencia de ciberamenazas [215].

A continuación se podría encontrar el conjunto de estructuras SCAP, la cuál se caracteriza por establecer un estándar para indicar tanto el formato como las nomenclaturas acerca de identificación de programas, defectos en los programas, configuraciones de seguridad y contenidos relacionados.

Otra estructura de contenido interesante podría ser CyberDEM. Este estándar está pensado para representar objetos y eventos en el mundo cibernético que pueden considerarse de interés. En consecuencia, uno de los principales casos de uso es el modelado y la simulación de elementos cibernéticos.

2.4.6 Interacción con los datos

Finalmente, pero no menos importante, los usuarios finales deberán tener un punto de entrada para interactuar con la aplicación. Al respecto, existe una gran variedad de formas de interactuar con las aplicaciones [167], las cuáles, serán analizadas a continuación.

Acorde a los estudios realizados y las soluciones actuales, en la actualidad, la forma más simple de interacción con los datos es el empleo de ficheros donde el usuario introduce las configuraciones oportunas y el servicio funciona acorde a dicho contenido.

Por otro lado, una forma más simple de interacción con las aplicaciones por parte de los usuarios, es aquella donde éstas exponen una consola con la que, mediante la introducción de comandos específicos, navegan a través de los distintos menús de la aplicación, realizan las configuraciones pertinentes e, incluso, se realizan ciertas representaciones de la información. A este tipo de aproximación se la conoce como aplicaciones de consola.

Finalmente, existe lo que se conoce como aplicaciones con interfaz gráfica de usuario. Éstas, tienen lo que se conoce como interfaz de humano a máquina (HMI, del inglés *Human Machine Interface*). Mediante este HMI, este tipo de aplicaciones ofrecen al usuario experiencias de uso de la aplicación fácil e intuitivas, así como un gran abanico de representaciones de los distintos datos con los que trabaja la misma.

Toda aplicación, independientemente de si la arquitectura que implementa es de tipo monolíticas o distribuidas, se le podrían diferenciar tres partes principales en el componente de interacción con los datos,

- Servidor
- Cliente
- Visualizaciones

Servidor

La primera de la partes de los elementos de interacción con los datos es el servidor. Esta pieza del componente es la encargada de preparar la información a visualizar y entregársela al cliente una vez éste la solicite. Aunque la finalidad de esta pieza es siempre la misma, hay gran cantidad de formas de interacción del cliente con el servidor.

En primer lugar se encontrarían aquellas aplicaciones donde tanto el servidor como el cliente del componente de interacción se encuentran en el mismo ejecutable. En este caso en particular, la transferencia de datos entre ambos componentes es directa.

Otra posible forma de interacción es aquella donde ambas piezas del componente de interacción se ejecutan en la misma máquina, aunque desde distintos procesos (o ejecutables). En este caso en particular, se pueden emplear distintas soluciones, entre otras, lo que se denomina como canalizaciones (o en inglés, *pipes*) del sistema operativo o memoria compartida (o en inglés, *shared memory*) para realizar el intercambio de información [178, 264, 279].

Por otro lado, se encuentran aquellos casos donde el servidor y el cliente no tienen porque estar ofreciendo sus servicios desde la misma máquina. En este tipo de casos, se emplea la red para realizar los intercambios de información. Para realizar dicho intercambio, se puede encontrar una gran variedad tanto en las distintas aproximaciones acerca de como generar el contenido a devolver al cliente como en las interfaces de comunicación empleadas entre el servidor y el cliente.

Aproximaciones acerca de como generar el contenido. De la gran variedad de aproximaciones acerca de como generar el contenido, las dos más frecuentes serían:

- Generación de las representaciones en el lado del servidor
- Generación del contenido a representar en el lado del servidor

Generación de las representaciones en el lado del servidor. Cuando el servidor es el encargado de generar las representaciones se considera que es de tipo *server-side rendering* [246].

Con esta aproximación, el servidor se encarga tanto de solicitar la información a la base de datos y procesarla como de generar las representaciones que se mostrarán al usuario por parte del cliente (*i.e.*, cuando un cliente es de tipo web, el servidor devuelve el código HTML listo para representar).

La principal característica de esta aproximación es que todo el procesamiento se hace por parte del servidor, por tanto, el cliente únicamente emplea recursos para representar el contenido generado por el servidor.

Generación del contenido a representar en el lado del servidor. Otra aproximación consiste en aquella donde el servidor únicamente se encarga de generar el contenido a representar, también llamada *API First* [18].

En ésta, cuando se recibe una petición, el servidor se encarga de solicitar la información a la base de datos y procesarla y, una vez ha realizado estas tareas, le devuelve al cliente los datos con el contenido a representar, normalmente, empleando alguna de las representaciones de los datos más frecuentes en la actualidad, es decir, JSON o XML [86], aunque también se pueden emplear aproximaciones más eficientes como los *protocol buffers* [44] desarrollados por Google.

A continuación, es tarea del cliente tanto el generar las representaciones como el mostrarlas.

La principal característica de esta aproximación es que sirve para cualquier tecnología que emplee el cliente para representar el contenido.

Interfaces de comunicación. Respecto a las distintas interfaces de comunicación existentes, las más destacadas serían [233]:

- API REST
- RPC

API REST. La interfaz de comunicación se considera API REST [168] cuando se emplea el paradigma de comunicación de petición y respuesta y se trabaja empleando el protocolo de intercambio de información HTTP. Además, se consideran comunicaciones sin estado, por tanto, en la solicitudes no viaja información del cliente, es decir, cada una de las peticiones realizadas es completamente independiente al resto de peticiones (previas y futuras).

RPC. La interfaz de comunicación de llamada a procedimiento remoto (RPC, del inglés *Remote Procedure Call*) fue inicialmente especificada en la RFC 1831 [242] en el año 1995 y se encarga de definir como se produce la comunicación entre un cliente y un servidor de forma bilateral. Esta interfaz de comunicación ha ido evolucionando y, a día de hoy, se pueden encontrar encontrar soluciones basadas en RPC como gRPC y JSON-RPC entre otras.

En primer lugar, gRPC, la llamada a procedimiento remoto desarrollada por Google, es una evolución de RPC multilenguaje la cuál hace uso del protocolo de intercambio de información HTTP en su versión 2, lo que le permite emplear el paradigma de comunicación de petición y respuesta de forma mejorada, permitiendo comunicación bidireccional simultanea de cliente a servidor y de servidor

a cliente. No solo eso, para conseguir que el intercambio de información sea lo más eficiente posible, se emplean *protocol buffers* como representación de los datos, una tecnología desarrollada también por Google que emplea la representación binaria de la información para conseguir la máxima eficiencia posible.

Una alternativa a gRPC es JSON-RPC. Ésta, a diferencia de gRPC, se caracteriza por priorizar la sencillez frente a la eficiencia. Las comunicaciones entre cliente y servidor son unidireccionales si se emplea el protocolo de intercambio de información HTTP, pudiendo ser bidireccionales si se emplea el protocolo de intercambio de información *WebSocket*. Como ventaja respecto a gRPC, el contenido intercambiado emplea la representación de datos JSON, la cuál, es la más extendida en la actualidad [86].

Cliente

El cliente es la parte del elemento de interacción con los datos dónde acceden los usuarios finales.

Éste se encarga de establecer comunicaciones con el servidor para solicitarle la información necesaria. Con ésta, el cliente es capaz de mostrar representaciones específicas al usuario dependiendo del recurso que haya solicitado, permitiéndole realizar todas las interacciones con los datos necesarios, como por ejemplo, visualizar datos, crear datos, editar datos, eliminar datos, entre otras.

El cliente debe ser atractivo a la vista y muy fácil de utilizar, o lo que es lo mismo, debe tener una buena interfaz de usuario (UI, del inglés *User Interface*) y una agradable experiencia de usuario (UX, del inglés *User Experience*), con esto se conseguirá que se extraiga el mayor rendimiento posible de la aplicación [8, 104, 210], ya que no hay que olvidar que este elemento va a ser el punto de entrada de los personas que vayan a hacer uso de la aplicación.

No obstante, conseguir una buena UI y UX no es tarea sencilla, ya que, hoy en día, es bien conocido que los usuarios emplean gran cantidad de dispositivos distintos para interactuar con las aplicaciones, desde ordenadores hasta teléfonos móviles, tabletas, e incluso relojes inteligentes. Por tanto, hay que tener en cuenta cuáles van a ser los dispositivos desde los que el usuario será capaz de acceder y, además, para cada uno de ellos, desarrollar los flujos de interacción óptimos (ya que no es lo mismo estar trabajando en un ordenador con pantalla horizontal y con un teclado y un ratón que en un teléfono móvil con pantalla táctil vertical).

Debido a este motivo, los tipos de clientes se podrían dividir en cuatro grandes grupos:

- De escritorio
- Móviles
- Híbridos
- De realidad virtual

De escritorio. Los clientes de escritorio serían aquellos pensados para trabajar con ordenadores, ya sea de sobremesa o portátiles.

Estos clientes suelen trabajar con pantallas horizontales e interactúan con la aplicación empleando, de forma general, un teclado y un ratón.

Existe gran cantidad de marcos de trabajo para desarrollar aplicaciones de este tipo, como por ejemplo, Qt [45], WinForms [267], WPF [267], Electron [231] y Tauri [58], entre otros.

Móviles. Los clientes para dispositivos móviles son aquellos pensados para trabajar con teléfonos, tabletas, relojes y bandas inteligentes.

Estos dispositivos, en la actualidad, suelen emplear pantallas táctiles y, dependiendo del dispositivo, pueden trabajar en posición vertical u horizontal.

Para el desarrollo de este tipo de clientes (usualmente conocidos como aplicaciones) existe gran variedad de marcos de trabajo, habiéndolos tanto específicos para el sistema operativo sobre el que se va a ejecutar como multiplataforma. De estos, se podrían destacar React-Native [28] y Xamarin [257], entre otros.

Híbridos. También existen opciones donde se desarrolla el cliente una vez y sirve tanto para aplicaciones de escritorio como para aplicaciones móviles.

En este aspecto, la solución predominante es el empleo de aplicaciones basadas en páginas web, aunque también existen alternativas como Flutter [254], entre otras.

De realidad virtual. Otro tipo de cliente es el que está pensando para trabajar con realidad virtual.

Para estos clientes, normalmente se emplean gafas especiales que permiten al usuario navegar por la aplicación empleando los 360 grados del espacio disponible.

Para desarrollar estas aplicaciones se suele emplear software específico, como por ejemplo, Unity [221] o React-Native [28], entre otros.

Visualizaciones

Finalmente, dentro del componente de interacción de datos, se encuentra el conocer tanto cuál es la información que le interesa visualizar al usuario final así como cuáles son las técnicas de visualización que permiten a dicho usuario, de la mejor y más rápida forma posible, generar inteligencia a partir información.

Cuando se trata de visualizaciones, hay que tener en cuenta que la aplicación va a ser, potencialmente, empleada por varios tipos de usuarios, dónde cada uno de ellos va a requerir de distintos tipos de visualizaciones.

Tipos de usuario. Dentro de los distintos tipos de usuario, se podrían diferenciar tres roles principales, los cuales serían:

- Estratégico
- Operacional
- Táctico

Estratégico. Dentro del rol estratégico se encontrarían aquellas personas encargadas de definir la estrategia, es decir, cuál es el objetivo final que se quiere conseguir a largo plazo.

Operacional. El rol operacional depende de las decisiones que se han tomado por parte de los componentes del rol estratégico.

Los integrantes del rol operacional se encargan de planificar las distintas operaciones necesarias a la largo del tiempo que permitan cumplir los objetivos que busca alcanzar el rol estratégico.

Táctico. Finalmente, los integrantes del rol táctico se encargan de ejecutar las distintas operaciones diseñadas por el rol operacional.

Tipos de visualizaciones. Dependiendo de los datos que se quieran representar y del tipo de usuario que está interactuando con dichos datos, las visualizaciones deberán adaptarse.

Existe una gran variedad de tipos de visualizaciones distintas, aunque de todas éstas, se podrían destacar dos grandes grupos, los cuáles serían:

- Visualizaciones basadas en textos
- Visualizaciones basadas en gráficos

Visualizaciones basadas en textos. Ciertos roles necesitan acceder a la información en crudo para realizar análisis muy precisos y exhaustivos.

Este tipo de visualizaciones, normalmente, se suelen representar tal y como han sido recolectadas o generadas (empleando JSON o XML [86]), en forma de tablas o en forma de reportes textuales.

Dentro de esta información en crudo se puede encontrar, entre otra:

- Adquisiciones de las fuentes de datos
- Información acerca del estado de los distintos componentes que conforman la aplicación
- Información generada por los elementos de preprocesamiento y procesamiento de los datos

Visualizaciones basadas en gráficos. La información basada en textos, aunque útil para cierto tipo específico de casos, puede resultar difícil de analizar cuando hay gran cantidad de información por unidad de tiempo. Una solución a dicho problema es mostrar dicha información empleando componentes gráficos. Con éstos se consigue conocer lo que está sucediendo más rápidamente y mejor y, en consecuencia, obtener mejor resultados [37, 40].

Visualizaciones gráficas hay de muchos tipos, algunas que destacan podrían ser:

- Gráficos de fuerza
- Agrupaciones circulares
- Árboles de datos

2.5 Servicios actuales

Tras analizar los distintos estudios y trabajos académicos que tratan sobre arquitecturas de aplicación, se observa que hay ciertos elementos de dichas arquitecturas los cuáles se repiten, por tanto, dada esta casuística, cabría la posibilidad de que existieran soluciones específicas encargadas de proporcionar los servicios para los que están pensados dichos componentes y, si así fuera, implicaría que se podría hacer uso de dichas soluciones, las cuáles, de forma ideal, ya se habrían sometido a un proceso de verificación exhaustivo por aquellos usuarios que las explotan y, por tanto, asegurarían su correcto funcionamiento en los ámbitos para los que han sido pensados.

Existen soluciones para muchos de los posibles componentes que podrían conformar una arquitectura de aplicación, de entre éstos (aunque no los únicos), actualmente existen servicios para:

- Almacenamiento de los datos
- Interacción entre componentes
- Gestión de la autenticación y la autorización

2.5.1 Almacenamiento de los datos

Toda aplicación que vaya a trabajar con información (es decir, que vaya a recolectar datos, los vaya a procesar y finalmente vaya a inteactuar con los mismos) necesita de un sitio donde almacenarlos y necesita de técnicas para interactuar con los mismos.

El tener los mecanismos adecuados para consultar, crear, editar y eliminar información puede marcar la diferencia y hacer que una cierta aplicación funcione correctamente o, que por el contrario, no cumpla con la finalidad para la que ha sido desarrollada, es por ello que se considera crítico el tener un buen sistema que sea capaz de realizar la interacción con los datos de la forma más óptima posible [130].

Para satisfacer esta necesidad existen desde soluciones simples (como podría ser el uso de ficheros de texto) hasta soluciones avanzadas, donde se emplearía el uso de bases de datos. Las soluciones simples podrían servir para aquellos sistemas auxiliares donde se necesita almacenar información que no se modifica con el tiempo o donde no es crítica la interacción con los mismos. Por otra parte, para aquellos sistemas que son extremadamente dependientes de los datos, es muy importante escoger una base de datos adecuada al tipo de los mismos. Dado que en la actualidad existe una gran variedad de sistemas de gestión de bases de datos, a continuación se va a realizar un estudio de los mismos.

Bases de datos

Una base de datos, más específicamente, los sistemas de gestión de bases de datos, son servicios que se encargan, no solo de almacenar la información, sino también, de proveer los mecanismos necesarios para realizar las tareas anteriormente indicadas de consulta, creación, actualización y borrado de la información de forma óptima. Además, dependiendo del tipo de base de datos, son capaces de filtrar la información, relacionarla entre sí, etc.

Tal y como hay gran variedad de estructuras de datos [274] existen gran variedad de bases de datos desde el punto de vista de para que tipo de datos están optimizadas. Asimismo, las bases de datos también se clasifican por otros factores, como podría ser la HA, capacidad de lecturas y escrituras simultaneas, etc.

Algunos ejemplos de las bases de datos más empleadas actualmente son:

- MySQL
- Cockroach
- MongoDB
- Elastic Search
- Neo4J
- SurrealDB

MySQL. MySQL es una base de datos bastante popular y de código abierto cuyo mantenimiento está respaldado por la compañía Oracle [41, 115, 196].

MySQL es un motor de base de datos de tipo relacional. Este tipo de bases de datos se caracterizan por tener una estructura de tablas fuertemente definida, lo que significa que la forma en la que se deben realizar las consultas a la base de datos es bien conocida. Además, las bases de datos de tipo relacional permiten hacer referencias de filas entre tablas. Otra característica de este tipo de bases de datos son los mecanismos que ofrecen para eliminar datos que mantienen dependencias entre ellos.

Por contrapartida, las bases de datos de tipo MySQL son poco escalables y la capacidad de datos que pueden procesar por unidades de tiempo es inferior al de otras bases de datos.

Cockroach. La base de datos Cockroach se empieza a desarrollar a partir de 2014 como solución a uno de los problemas de MySQL [121, 251], tal y como se detalla a continuación.

Actualmente, los usuarios acceden a las aplicaciones desde cualquier parte del mundo y, por regla general, las redes de comunicaciones son capaces de conectar dos puntos (aunque éstos se encuentren en extremos opuestos) sin que suponga un tiempo de latencia elevado. No obstante, ni todas las aplicaciones son iguales, ni la normativa actual permite cierta casuística, es aquí dónde Cockroach soluciona los inconvenientes de MySQL en cuanto a escalado horizontal se refiere.

En primer lugar se encuentran aquellas aplicaciones que tienen un tiempo de latencia específico que no pueden superar independientemente de desde donde se origina la petición. Para ello, la solución más lógica es tener réplicas de la base de datos distribuidas por el mundo para que las aplicaciones accedan a la réplica más cercana.

Por otro lado se encuentra la normativa. Que una aplicación sea accesible desde cualquier parte del mundo no significa que los datos se puedan almacenar en cualquier parte del mundo. Por poner un ejemplo, según la regulación general de protección de datos de la Unión Europea (GDPR, del inglés *General Data Protection Regulation*), los datos de los ciudadanos europeos deben almacenarse en bases de datos que se encuentren en territorio europeo (para así estar sujetas a las leyes de privacidad europeas) o en lugares donde la jurisdicción tenga niveles similares de protección [65]. Asimismo, en caso de no aplicarse el GDPR (por ejemplo, al no ofrecer los servicios en la unión europea), las bases de datos deben tener en consideración otros estándares de seguridad como el ISO 27001 o el NIST 800-53. El ISO 27001 ofrece un catálogo de puntos de control y buenas prácticas para asegurar la seguridad tanto en elementos físicos y de equipamiento de comunicaciones como en elementos cibernéticos. Por otro lado, el NIST 800-53 ofrece una guía detallada acerca de como implementar y gestionar controles de seguridad desde una perspectiva basada en los riesgos de la seguridad de la información.

MongoDB. MongoDB es una base de datos que fue fundada el año 2007. Mientras las bases de datos anteriormente analizadas eran de tipo relacional, MongoDB es una base de datos de tipo no sólo SQL (NoSQL, del inglés *Not only SQL*) [60, 115, 196].

Ante el avance de las nuevas tecnologías, también se produjeron cambios en los requisitos que se esperaban de las bases de datos. Las bases de datos de tipo relacional no eran capaces de cumplir con las necesidades del momento y es por ese motivo que se empezaron a desarrollar las bases de datos de tipo NoSQL.

Principalmente, MongoDB es capaz de solucionar las necesidades relacionadas con el *Big Data*, como podrían ser la alta velocidad de lectura y escritura así como el escalado horizontal.

Además, las bases de datos de tipo NoSQL y, en consecuencia, MongoDB, tienen otra diferencia destacable respecto a las bases de datos de tipo relacional, que es la ausencia de definición de tipos de datos, lo que permite consultar y almacenar los datos sin estar forzado a un modelo de datos específico.

Elastic Search. Elastic Search es un tipo de base de datos NoSQL la cuál se empezó a desarrollar a partir del año 2010. Se basa en Apache Lucene, una librería muy potente desarrollada por la fundación Apache para el indexado de datos y la consulta eficiente de contenidos [1, 84, 149].

Elastic Search es una base de datos pensada para trabajar con grandes cantidad de datos, por este motivo, desde que se empezó a desarrollar, se tuvo en mente como requisito la capacidad de escalabilidad.

Elastic Search se ha desarrollado para realizar consultas de lectura muy complejas (incorporando un motor de búsqueda de textos muy potente), en tiempo real y muy eficientes. Estas características la hacen óptima para trabajar con *Big Data* y AI.

Con respecto a otras bases de datos NoSQL, Elastic Search es menos eficiente a la hora de realizar operaciones de escritura.

Neo4J. Neo4J es una base de datos pensada para trabajar con estructuras de datos de tipo grafo. Su lanzamiento oficial fue en el año 2007 [1, 69, 149].

Neo4J es una base de datos de código abierto que ofrece tanto versiones de uso gratuito como versiones de uso con licencia.

Las bases de datos de este tipo estructuran sus datos en nodos y relaciones (como en los grafos) en vez de en tablas.

SurrealDB. SurrealDB es una base de datos que ha sido lanzada oficialmente en septiembre del año 2023 y que busca combinar todas las virtudes de los distintos tipos de base de datos [146, 247].

SurrealDB es una base de datos que permite tanto trabajar con tablas con esquemas definidos como con esquemas libres. Asimismo, también permite trabajar con estructuras de tipo grafo. Además, la forma de realizar consultas a la base de datos se basa en el lenguaje de consulta estructurada (SQL, del inglés *Structured Query Language*) aunque con ampliaciones para poder ejecutar todas las funcionalidades que ofrece.

SurrealDB es una base de datos que está pensada para ser escalable. También ofrece un motor de búsqueda de textos muy potente e incorpora funcionalidades para realizar procesamiento de datos con AI en la misma base de datos.

Otra característica destacable, es la capacidad de generar consultas en tiempo real donde, cada vez que hay modificaciones, es la propia base de datos la que notifica a las aplicaciones de dichos cambios.

Finalmente, y no por ello menos importante, ofrece una gran capacidad de granularidad en la autorización de los usuarios.

2.5.2 Interacción entre componentes

Independientemente del tipo de arquitectura empleada, los componentes deben ser capaces de realizar intercambios de información entre ellos. No solo eso, elementos externos a la aplicación, ya sea, por ejemplo, otras aplicaciones o usuarios, deben tener herramientas para interactuar con ésta.

Anteriormente en este trabajo se ha avanzado cuáles son las soluciones actuales al respecto para abordar este problema, no obstante, a continuación se analizará con más detalle dichas soluciones.

La primera solución que se puede encontrar en estudios y trabajos en la actualidad consiste en el empleo del protocolo de comunicaciones IP. Al emplear esta solución, mediante procedimiento, se puede definir de que manera se debe realizar dicho intercambio de información. En este caso, cabe destacar que en el protocolo IP se definen tanto direcciones para las comunicaciones de tipo punto a punto (o comunicaciones *unicast*) como direcciones para comunicaciones de tipo punto a multipunto (o comunicaciones *multicast*). La principal ventaja de esta solución, al trabajar directamente sobre el protocolo de comunicaciones IP, reside en su eficiencia, ya que el método de intercambio de información, *per se*, no añade ningún tipo de sobrecargas de procesamiento, cabeceras, etc.

Por otro lado, existen soluciones en gran cantidad de estudios y trabajos académicos que consisten en el empleo de un servicio (o componente) que trabaje como agente de comunicaciones [35], siendo esta alternativa analizada en detalle a continuación.

Agente de comunicaciones

El servicio de agente de comunicaciones tiene como finalidad la de recibir paquetes de los remitentes y enviarlos al destinatario (o a los destinatarios). Dado que es un único servicio (o grupo de servicios) el encargado de procesar todas las comunicaciones, presenta la ventaja de la centralización de éstas un único punto, con lo que también se abre la posibilidad de realizar la gestión de autenticación y

autorización de las comunicaciones en este elemento, si fuese necesario. Por otro lado, como desventaja, es un elemento cuyo funcionamiento debe estar asegurado para evitar interrupciones en el servicio de la aplicación.

Aunque dicho servicio se podría desarrollar dentro de la aplicación, en la actualidad existen gran cantidad de soluciones que realizan dicha función y están pensadas y verificadas para realizar su cometido de forma eficiente. Algunos ejemplos de las mismas podrían ser:

- RabbitMQ
- Kafka
- HiveMQ

RabbitMQ. RabbitMQ es uno de los servicios de mensajería más populares en la actualidad y fué lanzado en el año 2007 [56, 217].

RabbitMQ se caracteriza por implementar el protocolo de intercambio de información AMQP. El protocolo AMQP, como se ha detallado anteriormente, consiste en colas, intercambiadores y vínculos para hacer llegar un paquete desde un productor hasta un consumidor. Esta granularidad permite tanto la definición de rutas especificas entre productores y consumidores como la definición de políticas de autorización muy precisas, si fuese necesario.

Otra gran ventaja que tiene RabbitMQ es la capacidad de generar agrupaciones de agentes de comunicaciones para así tener la capacidad de ofrecer HA [88].

Kafka. Kafka es un servicio de mensajería que, en sus orígenes, fue desarrollado internamente dentro de la red social LinkedIN y cuyo lanzamiento oficial se produjo el año 2021 [56, 83].

Kafka fue desarrollado teniendo en mente el poder procesar ingentes cantidades de mensajes para poder ser usado en sistemas con gran cantidad de intercambios de información por unidad de tiempo, como podría ser aplicaciones que trabajen con *big data*. Kafka trabaja directamente sobre TCP sin emplear ningún protocolo de intercambio de información estandarizado.

Para poder llegar a dichos niveles de mensajes por unidad de tiempo, Kafka tiene una lógica de enrutamiento muy básica basada en temas (o en inglés, *topics*). Asimismo, debido a la gran granularidad que tiene AMQP, se puede llegar a realizar una configuración que siga el paradigma de configuración de tipo petición y respuesta mientras que con kafka sería más complejo.

HiveMQ. HiveMQ es una plataforma MQTT de código abierto que implementa todas la funcionalidades del protocolo de intercambio de información MQTT [21, 53, 176, 177].

HiveMQ fue desarrollado para ser escalable desde sus origenes y prioriza evitar la pérdida de mensajes frente a otras características.

HiveMQ, respecto a otras plataformas MQTT más simples, necesita una mayor cantidad de recursos de CPU y memoria RAM para funcionar de forma óptima.

2.5.3 Gestión de la autenticación y la autorización

Desde los comienzos de las aplicaciones multiusuario, inicio de los años 1970 [225], se empezó a tener en cuenta el concepto de la gestión de la autenticación y la autorización. Inicialmente se desarrolló este concepto para los usuarios (internos y externos) que interactúan con las aplicaciones y, más adelante, este concepto se extendió a los componentes que conforman una cierta arquitectura de aplicación. Aunque parecidos, los requisitos de cada uno de estos elementos es distinto, por tanto, a continuación se analizarán las diferencias entre ambos.

En primer lugar, en aquellas arquitecturas donde los componentes son susceptibles a ser distribuidos, éstos también son susceptibles a ser suplantados, con consecuencias tan fatales como usurpación de identidades, exfiltración de datos confidenciales e indisponibilidad de los servicios ofrecidos, entre otros [79]. Para evitar que se lleven a cabo este tipo de ataques, los componentes que conforman este tipo de arquitecturas deben implementar mecanismos de autenticación.

Con respecto a los usuarios, éstos, al igual que los componentes, deben realizar una primera fase de autenticación [224], la cuál, tradicionalmente se ha realizado mediante algo que conocen (como por ejemplo, una contraseña), algo que tienen (como por ejemplo, un número de teléfono) o algo que son (como por ejemplo, elementos biométricos como huellas dactilares) [129], aunque en la actualidad se están desarrollando sistemas de autenticación de acceso anónimo en los que en lugar de emplear contraseñas, el verificador emplea una serie de desafíos para realizar o no la autenticación del solicitador [142]. Además, para mejorar la autenticación de los usuarios, existe lo que se conoce como doble factor de autenticación [23], dónde se fuerza al usuario a que emplee dos de las formas tradicionales de autenticación.

Una vez se ha realizado la autenticación del usuario, el siguiente paso consiste en conocer cuáles son los permisos de interacción que tiene dicho usuario para cada uno de los recursos que ofrece la aplicación. Para llevar a cabo dicho proceso, se realiza la que se conoce como autorización (o control de accesos), de la cuál, existen gran cantidad de modelos dependiendo de como se realice, los cuáles, serán analizados a continuación [258].

Finalmente, en caso de que suceda algún incidente de ciberseguridad, se debe tener evidencias que permitan realizar un análisis exhaustivo del mismo para conocer como se ha podido llevar a cabo, poder atribuirlo y generar lecciones aprendidas para que no vuelva a suceder. Para poder realizar esta tarea, tanto los sistemas de autenticación como de autorización deben ser capaces de poder ser auditados por un servicio externo [226].

Cabe destacar que, de forma habitual, las organizaciones trabajan con múltiples aplicaciones distintas, cada una de ellas encargada de dar solución a una o más necesidades de los empleados de las mismas. En caso de que la gestión de la autenticación y la autorización se realizase por separado para cada una de ellas, supondría un inconveniente, tanto para los usuarios de las mismas (por tener tanto un usuario como una contraseña para cada una de ellas), como para los administradores de los sistemas (ya que cada vez que tuvieran que dar de alta o de baja a un usuario tendrían que ir aplicación por aplicación), además, con el añadido de que en el proceso de registro y suspensión de los usuarios, un cierto administrador de un sistema podría cometer un error y generar problemas de seguridad para la compañía [269]. Para evitar que sucedan estas situaciones, existen los mecanismos de inicio de sesión único (SSO, del inglés *Single Sign-On*), los cuáles, centralizan estas acciones en los conocidos como proveedores de identidad (idP, del inglés *identity Providers*) [93, 269]. Para realizar estas acciones, existen protocolos con la funcionalidad de realizar el proceso de autenticación, así como para realizar el proceso de la autorización, éstos, serán analizados a continuación. Asimismo, existen soluciones que implementan dichos protocolos y se encargan de realizar los procesos de autenticación y autorización acorde a los parámetros establecidos por el administrador del sistema, éstas, también serán analizadas a continuación.

Protocolos para realizar la autenticación

En cuanto a protocolos para realizar la autenticación se refiere, se pueden encontrar gran cantidad de opciones, de entre las que se podrían destacar, entre otras, Kerberos [75], Radius [101], SAML 2 [111, 273] y OpenID Connect [136, 273].

Protocolos para realizar la autorización

Con respecto a los protocolos para realizar la autorización, algunos de los más clásicos han sido RADIUS [175] y TACACS+ [175], aunque en la actualidad, el protocolo que se emplea como referencia para realizar la autenticación es OAuth 2.0 [25].

Modelos de control de acceso

Dependiendo de las características de la infraestructura de servicios que se administre en una cierta empresa, se puede necesitar mayor o menor granularidad con respecto a la gestión del control de accesos, por ese motivo, a día de hoy existen gran cantidad de soluciones en cuanto a modelos de control de acceso se refiere, de las cuáles, a continuación se analizarán algunas de ellas.

RBAC El control de acceso basado en roles (RBAC, del inglés *Role-based Access Control*) fue de los primeros en aparecer y existe desde inicios de los años 1970 cuando aparecieron las aplicaciones multiusuario [225].

Este sistema de control de accesos se caracteriza por asociar los permisos a roles de usuario y, una vez estos roles han sido definidos y caracterizados con los permisos correspondientes, a cada uno de los usuarios se le asigna un rol específico, asignándosele a éste los permisos asociados a dicho rol.

DAC El control de acceso discrecional (DAC, del inglés *Discretionary Access Control*) es uno de los más específicos. Este control de accesos se basa en una matriz mediante la que se relacionan las acciones que puede realizar un sujeto (es decir, un usuario específico o una aplicación especifica) a un cierto objeto (como por ejemplo, un fichero o un directorio) [120, 258]. A esta matriz se la conoce como la matriz de permisos [120].

TMAC El control de acceso basado en equipos (TMAC, del inglés *Team-Based Access Control*) es similar a RBAC aunque, en este caso, en vez de emplear roles genéricos, se emplean roles de equipos, para ello, en primer lugar se generan unos equipos y, sobre estos equipos, se generan roles a los cuales se les asignan los permisos, finalmente, se asigna cada uno de los usuarios finales a los equipos correspondientes y, dentro de éstos, a los roles necesarios acorde a las necesidades [256, 258].

Cabe destacar que existen modificaciones de TMAC, como TMAC basado en contexto (C-TMAC, del inglés *Context-based TMAC*) [258].

TBAC El control de acceso basado en tareas (TBAC, del inglés *Task-Based Access Control*) es una ampliación de DAC en la que, continuando manteniendo una matriz de permisos, esta no es estática, sino que va evolucionando junto con el progreso de las tareas [258]. Para conseguirlo, las tareas se dividen en etapas y, a partir del contexto de las tareas, TBAC es capaz de tener conocimiento de cuál es la etapa en la que se encuentra la tarea y, en consecuencia, hacer uso de la matriz de permisos correspondiente.

Soluciones para gestionar la autenticación y la autorización

Dado que desde pequeños grupos de trabajo, hasta grandes compañías, tienen la necesidad de emplear distintos servicios para dar solución a sus necesidades, hoy en día existen gran cantidad de soluciones para realizar las tareas de autenticación y autorización empleando los protocolos anteriormente analizados. Dentro de estas soluciones, se pueden encontrar desde librerías (en gran cantidad de lenguajes de programación) que se encargan de abstraer al desarrollador de los elementos específicos de cada uno de los protocolos, con la finalidad, de que la única tarea de éste sea la implementación de los puntos de entrada de las conexiones y la lógica de autenticación y autorización; hasta servicios ya implementados y listos para funcionar, dónde mediante configuraciones específicas de éstos, se genera la lógica de autenticación y autorización.

Librerías Con respecto a las librerías actuales, se podría destacar, entre otras, para Radius, aaa4j-radius [250]; para SAML 2, Python3-SAML [230] y ZXID [14]; para OpenID Connect, openidconnect-rs [211] y IdentityServer4 [102]; para OAuth 2.0, oxide-auth [180] y DNOA [70].

Servicios Realizando un estudio en profundidad de los distintos servicios existentes para realizar la autenticación y la autorización, se pueden encontrar un gran número de soluciones, de éstas, se podrían destacar, aunque no son las únicas, el directorio activo de Windows [72], FreeRADIUS [107], Keycloack [54] y Authentik [98].

Capítulo 3

Propuesta de arquitectura

En este capítulo se propondrá una arquitectura para realizar la caza de amenazas basándose en técnicas de inteligencia artificial.

Para poder recolectar datos de distintas fuentes, almacenarlos, procesarlos e interactuar tanto con los mismos como con la distinta información e inteligencia generada a partir de éstos, es muy importante que el sistema a desarrollar este basado en una arquitectura bien definida, la cuál, debe poder contemplar todos los aspectos necesarios para que sea óptima, principalmente, estos elementos son:

- Seguridad
- Escalabilidad
- Adaptabilidad
- Alta disponibilidad
- Eficiencia

Por dicho motivo, en este capítulo se definirá en detalle la arquitectura propuesta para realizar la caza de amenazas basándose en técnicas de inteligencia artificial.

En la figura 3.1 se encuentra representada gráficamente la arquitectura propuesta en el presente trabajo.

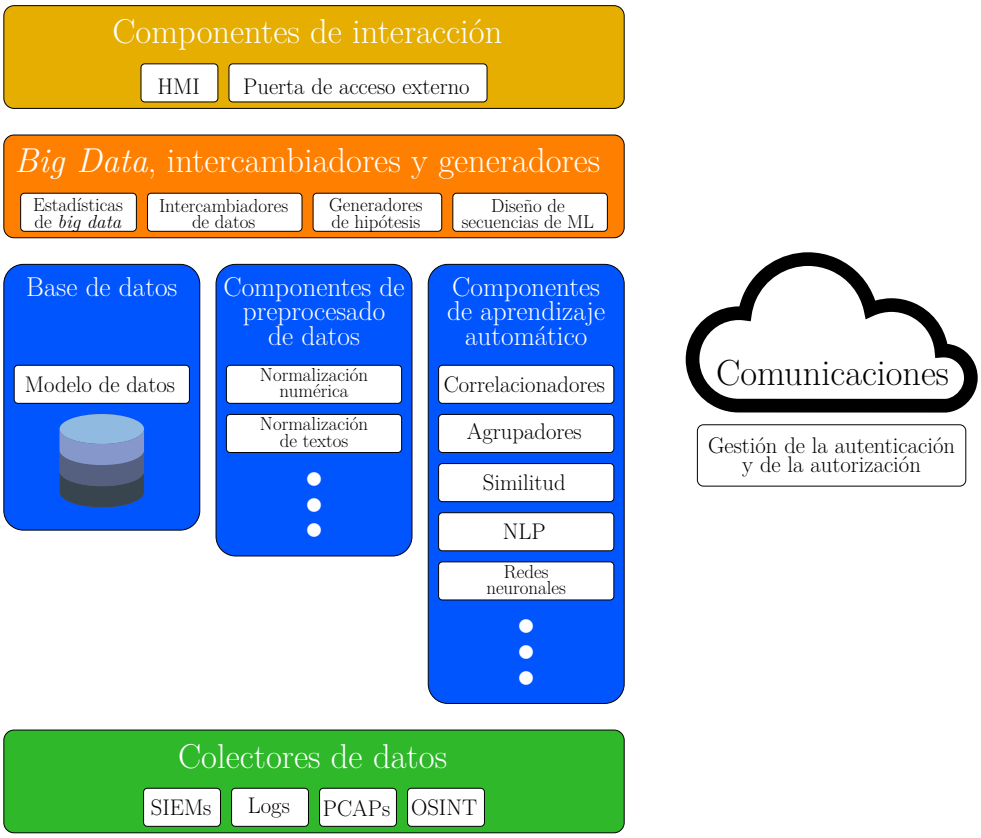

Figura 3.1: Arquitectura propuesta

La caza de amenazas es el conjunto de procesos ejecutados para detectar, analizar y responder ante ciberamenazas y, en el peor de los casos, ciberataques. Éstos se realizan en contextos muchas veces desconocidos, empleando técnicas en constante evolución y por especialistas de un gran abanico de ámbitos muy distintos. La casuística anteriormente enunciada es el principal motivo por el que toda aplicación que busque ayudar a los ciberespecialistas en su trabajo debe contar con una arquitectura muy bien definida desde su origen, en caso contrario, dicha solución tendría o una vida útil muy corta, o no ofrecería toda la ayuda que esperan los ciberespecialistas.

De entre las distintas soluciones que existen respecto a las arquitecturas de aplicación, para un programa que busque ser empleado para realizar la caza de amenazas, con las características anteriormente enunciadas (contextos desconocidos y empleando herramientas en constante evolución), la aproximación más ideal de entre las existentes (monolíticas o distribuidas) sería el empleo de arquitecturas de tipo distribuidas, ya que éstas ofrecen la ventaja de que para la incorporación de nuevas funcionalidades no hay que modificar la aplicación entera, sino únicamente añadir el componente nuevo al conjunto de los existentes y, no sólo eso, gracias a su capacidad de escalado horizontal y asimétrico, si existieran tipos de componentes saturados (en cuanto a carga de trabajo se refiere), se podrían añadir más de éstos sin tener que escalar la aplicación entera, consiguiendo en todo momento la eficiencia.

Desde sus orígenes hasta la actualidad, las arquitecturas de tipo distribuidas han sufrido una gran evolución, creándose definiciones específicas dependiendo, entre otros factores, de como se definen los componentes que conforman dicha arquitectura y de como es la interacción entre los mismos. De entre todas éstas, por la granularidad que ofrecen y por su fácil mantenimiento, se propone emplear arquitecturas distribuidas basadas en microservicios.

Tal y como se ha analizado anteriormente, las soluciones basadas microservicios llevan asociadas un elemento encargado de realizar el intercambio de mensajes entre componentes, conocido como el agente de comunicaciones, el cuál será analizado con detenimiento a continuación.

La arquitectura propuesta, principalmente, distribuye sus componentes en dos grandes conjuntos, uno encargado de interactuar propiamente con los datos y otro encargado de ofrecer los recursos necesarios para gestionar tanto las comunicaciones internas como la gestión de la autenticación y la autorización, ambos analizados a continuación.

3.1 Conjunto de componentes de los datos

Dentro del conjunto de funciones que se deben realizar en una arquitectura para realizar la caza de amenazas, se puede encontrar desde la recolección de los datos hasta la visualización de la inteligencia generada a partir de éstos, pasando por el almacenamiento y procesado de los mismos, entre otras. La parte de la arquitectura encargada de realizar las funciones anteriormente enunciadas es la del conjunto de componentes de los datos y será descrita a continuación.

Esta parte de la arquitectura se ha realizado por medio de una aproximación basada en capas lógicas y, al emplear esta filosofía, se generan niveles donde, en cada uno de ellos, se encuentran componentes, los cuáles, se clasifican dependiendo de la forma en la que interactúan con los datos. Esta arquitectura en particular propone el empleo de cuatro capas lógicas, las cuáles, evolucionan junto con el grado de interacción con los datos. Estos niveles, que serán analizados con detenimiento en esta sección, consisten en, (i) el de recolección, encargado de obtener los datos de las fuentes; (ii) el de procesamiento y almacenamiento, encargado tanto de generar información e inteligencia de los datos como de la persistencia del contenido obtenido y generado; (iii) el de preparación y configuración, encargado de filtrar la información relevante y de generar estructuras de datos estándar; y (iv) el de presentación, encargado de generar las visualizaciones así como de procesar los intercambios de información con elementos externos.

Cabe destacar que en una filosofía basada en capas, los componentes de una capa lógica únicamente pueden interactuar con los componentes de sus capas lógicas adyacentes, siendo una manera de hacer cumplir este requerimiento mediante el uso de un agente de comunicaciones.

3.1.1 Recolección

La primera de las capas, la de recolección, corresponde al nivel mínimo de procesamiento de los datos. En esta capa, la única transformación que reciben los datos es la consistente con la estandarización de la información obtenida de las distintas fuentes a un modelo de datos conocido por el conjunto de los componentes que conforman la arquitectura.

En esta capa, únicamente se pueden encontrar elementos de tipo colectores de datos, los cuáles, se definirán a continuación.

Colectores de datos

Los colectores de datos son los componentes encargados de realizar la adquisición de los datos de las distintas fuentes con las que cuente la aplicación que implemente la arquitectura propuesta.

Filosofías de recolección de datos. Tal y como se ha analizado con anterioridad, existe una gran cantidad de mecanismos de comunicación, pudiendo ser cualquiera de ellos empleado por las distintas fuentes de datos de las que se desee recolectar el contenido con el que trabajar en un futuro. Debido a este motivo, los componentes encargados de la recolección de los datos deben ser capaces de trabajar con diversos paradigmas de obtención de la información, destacando, (i) petición y respuesta bajo demanda, (ii) petición y respuesta recurrente y (iii) publicación y suscripción.

Petición y respuesta bajo demanda. En este modo de recolección, los componentes solicitan bajo demanda la recolección de nueva información.

Petición y respuesta recurrente. En este modo de recolección, los colectores de datos, bajo los parámetros de configuración necesarios, realizan la recolección de nueva información de acuerdo con la frecuencia establecida y de manera recurrente en el tiempo.

Publicación y suscripción. En este modo de recolección, los colectores de datos realizan suscripciones a las distintas fuentes y, en el momento que éstas tienen información nueva, automáticamente se la envían a los colectores de datos para que éstos la almacenen.

Modos de recolección de datos. A parte de la filosofía de recolección de datos, hay que remarcar que no todas las fuentes trabajan con el mismo tipo de contenido ni tampoco lo almacenan igual, en consecuencia, también es importante destacar los distintos modos de recolección que se pueden requerir a la hora de realizar la recolección. En este aspecto, los modos disponibles que deben soportar los componentes de recolección de datos son (i) espejo y (ii) actualización.

Espejo. En el modo espejo, en todas las peticiones, el recolector de datos solicita a la fuente de datos toda la información con respecto al contenido que desea obtener.

Actualización. En el modo actualización, el recolector de datos únicamente le solicita a la fuente de datos aquella información que se ha generado a partir de una cierta marca temporal, considerando que la información anterior ha sido recolectada con anterioridad.

Modos de almacenamiento de datos. El tipo de contenido a recolectar puede implicar desde información completamente nueva, pasando por actualizaciones de la información recolectada, hasta aquellas fuentes de datos que, por motivos internos, entre actualizaciones de los datos, realicen cambios que provocan que la información recolectada con anterioridad no sea válida y, por tanto, que haya que eliminar la información de dicha fuente y volver a almacenarla, en consecuencia, los distintos modos de almacenamiento que debe soportar una elemento de recolección de datos son (i) inyección, (ii) actualización y (iii) reemplazo.

Inyección. Al trabajar en el modo de inyección, el colector únicamente se encarga de introducir los datos nuevos sin tener en consideración la información existente.

Actualización. Al trabajar en modo actualización, el colector analiza el contenido existente en la base de datos y realiza las operaciones de creación, actualización y borrado necesarias.

Reemplazo. Al trabajar en modo reemplazo, el colector realiza una primera fase de borrado del contenido actual de la base de datos en relación con la fuente de datos y, a continuación, realiza la introducción de la nueva información.

Fuentes de datos. Finalmente, aunque no por ello menos importante, los tipos de fuentes de datos que se han considerado necesarios para realizar la caza de amenazas serían los que recolectan información de los sistemas de gestión de información y eventos de seguridad (SIEM, del inglés *Security Information and Event Management*), de los sistemas de agregación de registros, de los sistemas de agregación de captura de paquetes (PCAP, del inglés *Packet capture*) y de la inteligencia de fuentes abiertas (OSINT, del inglés *Open-Source Intelligence*). No obstante, cabe la posibilidad de que en un futuro se requiera de más tipos distintos de fuentes de datos, por dicho motivo el diseño de la arquitectura contempla la incorporación de forma sencilla de nuevas fuentes de datos.

3.1.2 Procesamiento y almacenamiento

La segunda capa, la de procesamiento y almacenamiento, es donde se encuentran tanto aquellos componentes que interactúan con los datos para generar información e inteligencia a partir de los mismos, como los componentes encargados de realizar la persistencia del contenido generado.

Aquí se pueden encontrar tres tipos distintos de componentes, los cuáles son, (i) la base de datos, (ii) los componentes de preprocesado de datos y (iii) los componentes de aprendizaje automático. Todos ellos serán analizados con detenimiento a continuación.

Base de datos

El componente de la base de datos permite que se puede generar persistencia del contenido generado por el conjunto de los elementos que forman la arquitectura, es decir, es el encargado de realizar el almacenamiento de los datos.

Tal y como se ha analizado en la sección correspondiente (página 48), hoy en día existe una gran variedad de servicios para el almacenamiento de información, todos ellos con sus ventajas e inconvenientes. La incorrecta elección de éste puede suponer que las peticiones realizadas para interactuar con la información almacenada (ya sea la consulta, inserción, actualización o el borrado de información) se realicen de forma ineficiente. Éstas, son las más empleadas tanto por los componentes de procesado de ML como por los componentes encargados de generar el contenido con el que interactúan los usuarios finales (ya sea tanto por medio del HMI como por medio de elementos externos mediante la compartición de inteligencia), por tanto, si no son eficientes puede suponer pérdidas de rendimiento notables para el conjunto de los elementos de la infraestructura, por este motivo, es crucial que la base de datos escogida sea la óptima para el tipo de datos con el que se va a trabajar.

Para escoger de forma correcta la mejor base de datos para esta arquitectura, se ha considerado relevante analizar tanto el tipo de contenido a recolectar como los distintos tipos de componentes que van generar información y van a requerir de su almacenamiento para generar persistencia.

Con respecto al contenido a recolectar, tanto para poder realizar un análisis exhaustivo de un incidente de seguridad como para alimentar el conjunto de los sistemas de ML, es necesario tener la mayor cantidad de datos posible, en consecuencia, es importante que la base de datos ofrezca soluciones de escalamiento.

Cabe destacar que, potencialmente, la información que se va a recolectar será siempre nueva, es decir, se realizarán operaciones de inserción de contenido nuevo, no de actualización ni de eliminación, por tanto, las operaciones de escritura serán simples.

Por otro lado, si se analizan los requerimientos de los componentes de ML, éstos tienen una seria de peculiaridades, las cuáles son, entre otras:

- Estructura de datos dinámica
- Peticiones de lectura compleja que permitan eliminar aquellas partes de la información que no aportan valor
- Capacidades para recibir automáticamente nuevos datos o mecanismos propios de la base de datos que se encarguen de gestionar consultas en las que la información evoluciona a lo largo del tiempo

Acorde a los requerimientos anteriormente expuestos, se propone emplear Elastic Search como base de datos por sus capacidades de escalado, consultas de lectura complejas y recursos para realizar consultas específicas en las que, entre peticiones, la base de datos es capaz de devolver únicamente aquella información que se ha generado nueva (llamados *data streams*).

Modelo de datos Con respecto al modelado de los datos, de entre las distintas alternativas que se pueden encontrar hoy en día, analizadas en la sección correspondiente de este trabajo (página 23), se ha decidido emplear una solución generalista, ya que no se conocen exactamente todos los tipos de datos que se va a necesitar almacenar y, de esta manera, cualquier necesidad futura puede ser cubierta.

De entras las distintas alternativas de este tipo de soluciones de modelado de datos, se ha decidido emplear ECS. Esta solución, en primer lugar, ofrece generalidad a la hora de definir los distintos elementos, además, propone definiciones estándar aceptadas de forma general para gran cantidad de elementos (como por ejemplo, los relacionados con eventos, registros y paquetes de red), y finalmente, está fuertemente relacionada con Elastic Search, lo que asegura que el desarrollo de la base de datos va asociado con el del modelo de datos.

De entre los elementos que se definen en ECS cabe destacar los eventos, al ser las secuencias de éstos un elemento de gran interés para las posibles técnicas de AI a desarrollar por los expertos en ML y DL.

Componentes de preprocesado de datos

Los componentes de preprocesado de datos son los encargados de realizar las tareas relacionadas con la preparación de los datos para Inteligencia Artificial, las cuáles, por su naturaleza, van a ser completamente dinámicas, aspecto muy importante a tener en cuenta.

La finalidad de este tipo de componentes es la de realizar todas las modificaciones de los datos necesarias (como por ejemplo, agregación de información o filtrado) para que los componentes de aprendizaje automático puedan desempeñar sus trabajos de forma óptima. Aunque cada tipo de técnica de ML es potencialmente distinta, existen ciertas tareas de preprocesado que pueden ser requeridas por múltiples de éstas, en consecuencia, si en cada una de ellas se tuviera que implementar este preprocesamiento, implicaría, no solo una carga de trabajo extra (y evitable) para el experto en ML, sino también una ineficacia con respecto a los recursos con los que cuenta el sistema en general. Además, iría en contra de la filosofía de la arquitectura escogida (arquitectura basada en microservicios), la cuál define que cada uno de los componentes de la arquitectura (o microservicios) debe desarrollar tareas muy específicas. No solo eso, como estos componentes van a ser empleados por múltiples servicios, éstos deben ser agnósticos con respecto a la información que están procesando, únicamente deben conocer como realizar su tarea, sin importar la procedencia de los datos a los que están realizando las transformaciones.

Tal y como se ha mencionado con anterioridad, estos elementos tienen la peculiaridad de evolucionar con el tiempo, en consecuencia, deben existir procedimientos dentro de la arquitectura para permitir la modificación fácil y rápida de los componentes de preprocesado cuando sea necesario. Para tener en cuenta esta casuística, se propone que los componentes de preprocesado, como tal, no tengan lógica alguna, sino que la función de éstos sea la de recibir el programa a ejecutar de algún servicio externo, realizar las tareas necesarias de dicho programa y devolver el resultado. Para conseguir este objetivo, se proponen una serie de requisitos que se analizarán a continuación.

En primer lugar, los componentes deben ser capaces de recibir la información que van a procesar, para ello, estos obtendrán la misma por parámetros de llamada (es decir, información recibida cuando empiezan a trabajar), la cuál puede provenir de otros componentes o del resultado de consultas a la base de datos. Como esta información puede variar dependiendo del destino de los datos, es necesario que sea configurable por el usuario.

En segundo lugar, tal y como se ha mencionado anteriormente, estos componentes no conocen cual es el procesamiento que deben realizar y, por tanto, deben existir repositorios de código accesibles desde estos microservicios a los que solicitar la lógica del procesamiento que se desea realizar. No sólo se le debe indicar al componente cuál es el código en particular que debe procesar en cada momento, sino que además, el usuario debe tener recursos, ya sea mediante el HMI o mediante elementos externos, para realizar las modificaciones que desee oportunas a dicha lógica.

Finalmente, el contenido generado por este preprocesamiento puede tener dos finalidades (no excluyentes entre si), (i) su almacenamiento en la base de datos para generar persistencia de éste y (ii) su empleo como entrada de otro componente de preprocesado o de un componente de ML.

Al emplear la lógica anteriormente descrita en los componentes de preprocesado, se consigue generar una agrupación de componentes que se encuentran a la espera de recibir solicitudes de cualquiera del resto de éstos, con lo que se logra la máxima eficiencia. Por otro lado, el resto de servicios pueden conocer tanto si hay un algún componente de preprocesado libre como el tamaño de las colas de éstos, con lo que cuando necesiten solicitar una tarea de éstas, se la pueda requerir a aquel componente que tenga la menor carga de trabajo, consiguiendo en todo momento ser eficiente y, por tanto, permitiendo mayor fluidez y rapidez en los flujos de trabajo.

Componentes de aprendizaje automático

El conjunto de los componentes de aprendizaje automático se corresponde con aquellos encargados de realizar el procesado de los datos mediante Inteligencia Artificial y, por tanto, serán los que generen la información e inteligencia para ayudar a los ciberespecialistas en la caza de amenazas.

Al igual que sucede con los componentes de preprocesado, las técnicas de aprendizaje automático también pueden ser empleadas para el procesamiento del contenido con el que trabajan los TH, no solo eso, tal y como se ha analizado en la sección correspondiente (página 27), existen técnicas complejas que se conforman a partir de la combinación de varias técnicas más simples, por dicho motivo, se debe permitir la posibilidad de que se realicen dichas configuraciones por parte de los especialistas en ML si lo requiriesen.

Dicho lo cual, los componentes de ML deben cumplir los siguientes requisitos para asegurar que se integran con el resto del ecosistema propuesto por esta arquitectura.

En primer lugar, al igual que con los componentes de preprocesado, los datos a tratar se recibirán siempre como parámetros, pero en este aso, serán procedentes únicamente de la salida de un componente anterior, ya sea un componente de preprocesado o de ML. De esta forma, se limita el acceso a únicamente información producida por otros componentes para así reducir la complejidad y, por tanto, facilitar el trabajo a los especialistas en ML.

A continuación, se encuentra el procesamiento de los datos recibidos, el cuál será dinámico como ocurre con los componentes de preprocesado. Tal y como sucede anteriormente, deben existir herramientas sencillas y de fácil acceso para los usuarios que les permitan realizar todas las modificaciones que consideren oportunas, así como realizar la depuración de la lógica desarrollada en caso de que fuese necesario. En este punto, existe una peculiaridad de los componentes de procesado de ML con respecto a los otros, y es que pueden requerir de repositorios donde se almacene información auxiliar necesaria para llevar a cabo su tarea, como por ejemplo, los distintos modelos ya entrenados de las redes neuronales. Por tanto, dentro de la configuración de las ejecuciones de estas tareas se les debe poder indicar de donde deben obtener dicha información.

Finalmente, al igual que sucede con los elementos de preprocesado, el contenido generado podría llegar a dos destinos (no excluyentes entre ellos), (i) la base de datos, para conseguir generar persistencia de la información, y (ii) otro componente, para emplear el resultado como entrada de otro procesamiento.

La definición de los componentes de ML realizada anteriormente ofrece la posibilidad de añadir, editar o eliminar las técnicas de AI necesarias en todo momento y de forma sencilla por parte de los especialistas en ML y DL, lo que les permitirá centrarse en el desarrollo de soluciones óptimas para los problemas que tienen que afrontar sin necesidad de preocuparse ni por conocer la infraestructura subyacente ni tampoco por tener que aprender como interactúan los componentes entre si, ya que existen componentes auxiliares encargados de realizar las interconexiones oportunas. No solo eso, se garantiza la adhesión a uno de los principios del paradigma de las arquitecturas basadas en microservicios que señala que cada uno de los componentes de la arquitectura (o microservicios) debe desarrollar tareas muy específicas, sin perjuicio de los requerimientos auxiliares de las técnicas de ML complejas, como por ejemplo, los modelos de redes neuronales entrenados con anterioridad.

3.1.3 Preparación y configuración

La tercera capa, la de preparación y configuración, contiene aquellos componentes que, o bien interactúan con los datos ya procesados dentro del modelo de datos de la arquitectura, o bien se encargan de realizar configuraciones sobre los componentes de la arquitectura.

En ésta, se encuentran componentes especializados en trabajos con grandes cantidades de datos, que son los componentes de estadísticas de *big data*; además, existen componentes capaces de estandarizar el modelo de datos de la arquitectura a modelos de datos específicos dependiendo del ámbito, los componentes conocidos como intercambiadores de datos; a continuación, se pueden encontrar componentes encargados de generar hipótesis a partir de la información procesada, que son los generadores de hipótesis; y, finalmente, se encuentran los componentes empleados por los especialistas en ML para crear, editar y eliminar las configuraciones de las técnicas de ML, los cuáles son los componentes de diseño de secuencias de aprendizaje automático.

Estadísticas de big data

El primero de los componentes que se encuentra en esta capa es el de estadísticas de *big data*. Un sistema que vaya a trabajar con la tipología de datos que se ha descrito anteriormente (es decir, gran cantidad de datos por unidad de tiempo, provenientes de distintas fuentes de datos y procesados por gran variedad de componentes distintos) debe considerar las peculiaridades de dicha tipología para conseguir tanto ser eficiente como, al mismo tiempo, ofrecer una forma fácil de interacción con los datos. Por este motivo, dentro de la arquitectura se define éste componente, el cuál, es accesible tanto desde la capa de procesamiento y almacenamiento (para los componentes de preprocesado de datos y los componentes de aprendizaje automático) como desde la capa de presentación (para el HMI y la puerta de acceso externo).

Cuando se necesita realizar una agregación de datos para generar información, incluso cuando se trabaja con cantidades de datos grandes, las soluciones que existen en la actualidad son realmente capaces de realizarlas en un tiempo muy razonable, sin embargo, cuando se desean procesar cantidades de datos como aquellas con las que se trabaja con *big data*, este procesamiento puede ser más costoso y, en consecuencia, durar un periodo de tiempo prolongado en el cual los recursos se encuentran bloqueados realizando dicha tarea. Además, es importante destacar que existen ciertas consultas que se repiten por parte de más de un

componente, no solo eso, incluso dentro de un mismo componente se puede dar el caso que varios procesos estén realizando peticiones a la misma consulta.

Dada la casuística anteriormente expuesta, se plantea el componente de estadísticas de *big data* que será el encargado de realizar este tipo de consultas y ofrecer el resultado a aquellos componentes que lo requieran. Es importante tener en cuenta que los datos empleados para obtener dichas consultas pueden variar a lo largo del tiempo y, en consecuencia, este componente deberá ser capaz de reaccionar ante cambios en los datos de sus consultas, de forma que cuando la información de entrada se actualice, este regenerará los resultados. En ciertos contextos, los componentes que hacen uso de los resultados del procesamiento de este componente pueden requerir de trabajar siempre con la versión más actualizada de esta información, para ello, los consumidores de los resultados tendrán la posibilidad de suscribirse a las actualizaciones de los resultados.

Cabe destacar que existe cierta casuística donde no sirve únicamente con almacenar la última versión de los resultados generados. En tal caso, este componente también deberá tener la posibilidad de emplear la base de datos para almacenar un histórico de los datos generados.

Intercambiadores de datos

Los componentes de la capa de presentación no tienen acceso directo a la información almacenada en la base de datos, aunque eso no significa que no puedan acceder a ellos, puesto que para permitir dicho acceso se incluyen los intercambiadores de datos en la arquitectura.

Tal y como se ha mencionado anteriormente, en la base de datos se almacena mucha información, entre ella, alguna que es interna y que no debería poder ser accesible por el resto de elementos de la arquitectura, en consecuencia, para forzar que se cumpla el requisito anteriormente expuesto, los componentes de la capa de presentación, aquella que emplean tanto los usuarios como las aplicaciones externas para interactuar con la aplicación, solicitan a los intercambiadores de datos la información que desean consultar, actualizar o eliminar y éste se encarga de realizar dicha tarea.

Ahora bien, cada uno de los componentes de la capa de presentación, como se verá más adelante, realiza unas tareas específicas, implicando esto que la forma en la que esperan recibir los datos puede variar. Por este motivo, los intercambiadores de datos deberán tener una función adicional, que es, realizar transformaciones en los datos para estandarizarlos.

Tal y como se ha analizado en el apartado específico del estado del arte (página 38), cabe la posibilidad de que se solicite presentar la información siguiendo alguno de los estándares existentes en la actualidad, en consecuencia, estos componentes también deberán tener la capacidad de transformar la información del modelo de datos de la arquitectura al estándar solicitado.

Generadores de hipótesis

Los componentes generadores de hipótesis son los encargados de ofrecer al TH suposiciones acerca de lo que está sucediendo en la infraestructura monitorizada, por tanto, se considera uno de los componentes más importantes de la arquitectura propuesta que, además, no se define en las soluciones existentes en la actualidad.

Cuando un TH está analizando un evento de seguridad, éste al final está analizando las posibles causas que han podido llevar a que suceda dicho evento (o incluso, grupo de eventos), es decir, está realizando hipótesis sobre lo que está ocurriendo en el teatro de operaciones. Estas hipótesis, al fin y al cabo, se generan a partir de, por un lado, los datos obtenidos de las distintas fuentes con las que se obtiene una ciberconsciencia situacional y, por otro lado, de las experiencias previas del analista resolviendo incidentes. Dado que en la arquitectura propuesta ya se ha detallado el como se va a realizar la recolección de las distintas fuentes de datos, este componente busca ser capaz de ayudar al TH en la parte que se refiere a experiencias previas.

Para conseguir ayudar al ciberespecialista, los generadores de hipótesis tienen dos aproximaciones no excluyentes entre si, las cuáles se analizarán con detenimiento a continuación.

Durante el ejercicio de sus funciones, un TH suele analizar gran cantidad de falsos positivos por comportamientos erróneos de los usuarios de los sistemas monitorizados, de tal forma que al final pueden llegar incluso a automatizarlos. Este hecho, junto con otros similares, son aprovechados por los ciberdelincuentes como vector de entrada de sus ataques. Todo sea dicho, cuando se llega a automatizar una cierta función por parte de un ciberespecialista, también se podría llegar a realizar un correlacionador de eventos, el cuál, bajos los parámetros del especialista, realizase estas tareas automáticas, siendo ésta la primera función de los generadores de hipótesis. Al realizar estas tareas empleando un sistema informático, en vez de ser realizadas por personas, se consigue que frente a una mínima desviación de los parámetros, probablemente imperceptible por un humano (ya que así lo preparan los ciberatacantes), se dispare una alerta, cuando, de forma contraria, hubiera pasado desapercibida. Estos sistemas se pueden implementar empleando tanto

reglas estáticas (como la reglas Sigma [194, 265] o las reglas YARA [187, 188] ya existentes), como empleando técnicas de ML, ya que estos componentes tienen acceso tanto a los componentes de preprocesado de datos como a los componentes de aprendizaje automático.

Por otro lado, existen ciertas técnicas de DL especializadas en la generación de contenido a partir de la información empleada para realizar el entrenamiento de las mismas. Con un conjunto de datos suficientemente grande y específico (que incluyese técnicas empleadas para realizar ataques, hipótesis sugeridas por especialistas en ciberseguridad y de más información relacionada), se podrían generar modelos, ya sea tanto generalistas como con sesgos específicos, que mientras analizan el contenido que va siendo procesado por el sistema, vayan generando una lista de posibles hipótesis sobre lo que está sucediendo en la infraestructura monitorizada, las cuáles, vayan siendo mostradas a los TH para su evaluación. Con esto se conseguiría que en caso de que éstos, bajo su punto de vista, considerasen que se está produciendo un incidente de seguridad, realizasen la investigación necesaria y tomasen las medidas que considerasen oportunas.

Diseño de secuencias de aprendizaje automático

El componente de diseño de secuencias de aprendizaje automático es el encargado de permitir a los usuarios interactuar con los programas que ejecutan tanto los componentes de preprocesado de datos como los componentes de aprendizaje automático, de generar flujos donde se enlacen los distintos componentes necesarios para llevar a cabo el procesado de los datos, de gestionar las ejecuciones de dichos flujos y de llevar un control de la evolución de cada una de las mismas. Tal y como se puede observar, es un componente que realiza funciones muy importantes dentro de la arquitectura, las cuáles, serán analizadas a continuación.

En primer lugar, este componente permite a los usuarios interactuar con los programas que se ejecutan tanto en los componentes de preprocesado de datos como en los componentes de aprendizaje automático, más precisamente, este componente tiene la lógica necesaria para solicitar la información requerida a los repositorios de códigos para poder ofrecérselos al HMI y que éste, finalmente, se la presente al usuario para que pueda realizar tantos las visualizaciones como las modificaciones que considere oportunas. Además, tal y como se ha mencionado con anterioridad, también debe de existir un elemento externo donde se almacene toda aquella información auxiliar que necesitan los componentes de aprendizaje automático para poder realizar sus funciones, este componente también deberá poder listar, añadir y eliminar información de dicho repositorio y ofrecerle al HMI las herramientas para notificarle de estos cambios.

Prosiguiendo, cuando se desea realizar un procesamiento empleando técnicas de ML, hay que realizar unas tareas previas de preparación y, a continuación, unas tareas de ejecución. En esta arquitectura se ha propuesto que los programas encargados de realizar cada uno de esos procesos se encuentren en componentes separados, en consecuencia, debe existir algún tipo de orquestador dónde los usuarios puedan realizar la configuraciones necesarias. En este caso en particular, se propone la generación de flujos de información entre componentes, donde los usuarios puedan, de forma visual, enlazar la salida de uno o más componentes a la entrada de otro u otros, de esta forma, se pueden generar sistemas de AI complejos de una manera sencilla.

Cuando los distintos flujos de ejecución se han generado, el siguiendo paso es iniciarlos para que empiecen a realizar sus trabajos. Para iniciar la ejecución de un flujo de ML, se proponen distintas opciones. En primer lugar, deberían existir aquellas secuencias que son ejecutadas una única vez, a continuación, debería existir un tipo de ejecución recurrente basada en temporizadores y, finalmente, se le debería poder definir a las secuencias un conjunto de dependencias para que la ejecución de dicha secuencia se produjese ante un cambio en alguna de las dependencias. Una vez se ha iniciado un flujo, tanto los componentes de preprocesado de datos como los componentes de aprendizaje automático tienen toda la información necesaria para enviar el contenido generado y los metadatos de la ejecución al elemento siguiente, de forma que el componente de diseño de secuencias de aprendizaje automático no se encarga de iniciar el trabajo de los procesos intermedios de la secuencia, únicamente del inicial.

Finalmente, este componente también debe poder ser capaz de realizar una traza de las ejecuciones, para ello, ante el inicio y el fin de un cierto proceso, el componente encargado de llevarlo a cabo debe notificarle la realización de dicha acción a éste componente y, además, si el experto en ML decide introducir puntos de control en los programas, toda esa información también debe llegar a este componente para que el susodicho experto pueda analizarla y verificar que la ejecución está funcionando acorde a lo esperado.

3.1.4 Presentación

La última capa, la de presentación, es la encargada de trabajar con los datos que han sido procesados dentro del modelo de datos de la arquitectura, que han sido filtrados para no proporcionar información interna de los componentes de la misma y, en caso de que fuese necesario, que han sido estandarizados.

La misión de esta capa es la de ofrecer los servicios necesarios para que puedan interactuar con los datos tanto los usuarios finales del sistema como aplicaciones externas, para ello, se pueden encontrar los componentes de HMI y de puerta de acceso externo, los cuáles, serán analizados a continuación.

HMI

El HMI va a ser el componente empleado por los usuarios del sistema para interactuar con el resto de componentes de la arquitectura, por tanto, es uno de los componentes más importantes de todo el conjunto, debido a ello, éste debería permitir la HA con el fin de asegurar el acceso al mismo en todo momento. Asimismo, este componente debe ser amigable e intuitivo para al usuario aunque sin perder ninguna de las funcionalidades requeridas para una aplicación de este tipo, por este motivo, a continuación se analizará con detenimiento.

Mediante el HMI los usuarios de la aplicación deben ser capaces de interactuar con el resto de componentes del sistema, sin olvidar en ningún momento que la aplicación va a ser empleada por usuarios con distintos roles y, en consecuencia, con distintas necesidades. Cabe destacar que no todos roles tienen porque poder acceder a todas y cada una de las funcionalidades de este componente, por dicho motivo es muy importante que éste también sea capaz de permitir la configuración de acceso a cada uno de los recursos ofrecidos. Finalmente, pero no por ello menos importante, debe ser fácil el diseño de la distribución de las distintas visualizaciones en las vistas con la que interactúa el usuario y, en caso de que fuese necesario, el clonado de éste a distintos usuarios.

La primera de las funciones con las que va a tener que contar el componente del HMI es con las configuraciones. Un sistema tan complejo como el propuesto tiene muchas piezas que deben poder ser gestionadas desde un punto de central, en este caso, el HMI. Dentro de la gestión de los componentes también se encuentra el propio HMI, ya que a éste se le debe poder configurar tanto los usuarios que tienen acceso al mismo como los permisos para cada uno de ellos.

Otro elemento a destacar es que desde este componente se deben poder cubrir todas las necesidades que llevan asociadas los componentes de preprocesado de datos y los componentes de aprendizaje automático, como podría ser la consulta, definición, edición y borrado tanto de códigos como de archivos auxiliares, todo ello, sin olvidar que debe ser fácil e intuitivo para el experto en ML y DL. Asimismo, este componente debe ofrecer un sistema para la ejecución y monitorización de los distintos flujos de ejecución de las secuencias de ML.

Por otro lado, la gestión de las hipótesis también debe ser accesible desde este componente para el experto en TH junto con todos los requerimientos asociados. Con esta función se consigue tanto poder evaluar las distintas hipótesis que se van generando como poder tomar decisiones de forma rápida y clara en aquellas situaciones donde hay muchos estrés y un mínimo error puede ser catastrófico.

Y finalmente, aunque no por ello menos importante, todos los datos generados por el sistema deben poder ser visualizados mediante mandos de control para poder generar una consciencia situacional de lo que está sucediendo en el teatro de operaciones y, así, tener la capacidad de tomar decisiones de alto nivel en caso de que fuesen necesarias.

Puerta de acceso externo

Sin duda alguna, para conseguir la mayor protección de un sistema, uno de los elementos claves es la compartición de información entre entidades amigas, en consecuencia, toda aplicación de ciberseguridad que busque ser efectiva debe implementar mecanismos para permitir de forma rápida y parametrizable la compartición de información, tanto el envío como la recepción. En la arquitectura propuesta, el componente encargado de realizar dicha tarea es la puerta de acceso externo.

La principal función de este componente será la de solicitar información a elementos externos y enviar información a aquellos sistemas que la soliciten. No obstante, es importante destacar dos aspectos muy relevantes, en primer lugar, que datos se van a enviar y, en segundo lugar, como se van a enviar dichos datos. Empezando con los datos que se van a enviar, hay que tener en cuenta que puede haber múltiples servicios solicitando información y todos ellos no tienen porque acceder a todos los datos del sistema, en consecuencia, se debe poder filtrar el contenido que se envía dependiendo de cuál es el origen de la petición. En segundo lugar, los datos se deben enviar de forma que ambos extremos de la comunicación sean capaces de entender el mensaje intercambio, por ese motivo, existen estándares para realizar estos intercambios. En un caso ideal, la interpretación de dichos estándares sería igual por todos los intervinientes en la comunicación, no obstante, algunos de éstos son tan generalistas que pueden llevar a que cada interviniente los interprete de forma distinta, por este motivo, es importante que este componente tenga la capacidad de permitir la incorporación de programas de preprocesado de los datos (tanto al recibir como enviar) para que así se pueda realizar el intercambio de mensajes de forma satisfactoria.

La puerta de acceso externo se encuentra en la capa de presentación, la cuál no tiene acceso directo a los datos, por tanto, este componente le solicita a los componentes de intercambio de los datos la información requerida en el estándar correspondiente, de forma que éste únicamente deba realizar un preprocesado de la información obtenida en casos específicos y no de forma general.

3.2 Conjunto de componentes de los recursos

Al igual que los usuarios necesitan de aplicaciones (como la que implementa esta arquitectura) para realizar sus tareas, los componentes internos de una cierta aplicación también pueden requerir de servicios específicos para llevar a cabo las tareas para las que han sido asignados. La arquitectura definida en este trabajo también ha tenido en consideración esta necesidad y, para ello, se propone un conjunto específico de componentes, el de los recursos, donde se definirán todos aquellos componentes a los que deben acceder el resto de los componentes para poder funcionar de manera satisfactoria.

En este grupo se pueden encontrar dos tipos distintos de componentes, los encargados de las comunicaciones y los encargado de la gestión de la autenticación y de la autorización, ambos, descritos con detenimiento a continuación.

3.2.1 Comunicaciones

Tal y como se ha descrito anteriormente, se propone una arquitectura distribuida basada en microservicios en la que, si se analizan sus necesidades, se puede destacar que existe un componente específico llamado agente de comunicaciones el cuál se encarga de recibir paquetes de los remitentes y enviarlos al destinatario (o a los destinatarios), en consecuencia, en esta arquitectura también existe dicho agente de comunicaciones.

La función básica de dicho componente se ha descrito anteriormente, recepción de mensajes y retransmisión de los mismos a sus destinatarios, no obstante, cabe destacar que en ciertas ocasiones dicha retransmisión debe poder ser asegurada y, en caso de que no se cumpla, se debe poder notificar al emisor de que ha habido algún tipo de problema en la retransmisión de los mensajes.

Además, debido a la aproximación basada en capas lógicas empleada en la estructura del conjunto de componentes de los datos dónde los componentes de una determinada capa solo pueden interactuar con otros de su misma capa o de capas adyacentes, el componente empleado para las comunicaciones también debería limitar la retransmisión de mensajes a únicamente aquellos destinos que cumplan dicho requisito.

3.2.2 Gestión de la autenticación y de la autorización

En una arquitectura distribuida, al poder estar cada uno de los componentes en un nodo de computación distinto, es importante asegurar que intervienen en la comunicación únicamente aquellos componentes que tienen permisos para hacerlo, por tanto, es imprescindible que exista un componente encargado tanto de realizar la autenticación cuando un componente accede a un sistema como de aportar recursos al resto de componentes para que realicen la validación de que los datos recibidos tienen como origen un componente autorizado en el sistema.

No solo los componentes, los usuarios que interactúan con los distintos componentes de la arquitectura (ya sea mediante el HMI o la puerta de acceso externo) también deben ser autenticados y autorizados para asegurarse que tienen los permisos necesarios para ejecutar las acciones que soliciten realizar.

Las tareas de autenticación y autorización las realizará este componente, el cuál, se sugiere que emplee el estándar OAuth 2.0 para realizar dichas tareas, al ser el protocolo de referencia en la industria en la actualidad para realizar este tipo de cometidos.

Cabe destacar que en OAuth 2.0 se definen tanto comunicaciones M2M, es decir, entre componentes (donde en la jerga de OAuth 2.0, los componentes serían conocidos como clientes), como comunicaciones entre usuarios y los distintos recursos con los que cuenta la aplicación (donde en la jerga de OAuth 2.0, los usuarios serían conocidos como propietarios de recursos).

Capítulo 4

Validación y verificación de la arquitectura

En este capítulo se describirá como se han realizado el conjunto de pruebas de validación y verificación de la arquitectura.

En este capítulo se va a detallar como ha sido el proceso empleado para realizar las pruebas relacionadas tanto con la validación como con la verificación de la arquitectura propuesta en el capítulo anterior. En primer lugar se detallará como se ha desarrollado el prototipo para realizar dicho conjunto de pruebas, a continuación se analizará como se ha realizado el proceso de validación de cada uno de los componentes de la arquitectura en el prototipo desarrollado, para proseguir se analizará el proceso empleado para realizar una verificación de la arquitectura en un entorno controlado (en un laboratorio) y finalmente se detallará como se han realizado las pruebas de verificación de la arquitectura en un entorno real no controlado (mediante el proyecto PRAETORIAN de la llamada SU-INFRA dentro de la convocatoria H2020 de proyectos de I+D financiados por la REA de la Comisión Europea).

4.1 Prototipo

La propuesta de arquitectura analizada anteriormente ha tenido en consideración tanto las necesidades manifestadas por los TH como las herramientas y servicios que se ha considerado que mejores resultados iban a aportar en cada uno de los distintos ámbitos de la misma, no obstante, la teoría no siempre es consecuente con la práctica, por tanto, se considera imprescindible ejecutar tantas pruebas de validación y verificación de dicha arquitectura como sean necesarias. Sobre una arquitectura como tal no se pueden ejecutar pruebas prácticas, por tanto, para poder llevar a cabo dichas pruebas se ha desarrollado un prototipo que ha implementado la arquitectura definida en su totalidad. En esta sección se analizará dicho prototipo destacando los aspectos más relevantes del mismo.

Al igual que en la definición de la arquitectura, los componentes del prototipo también serán diferenciados entre los encargados de interactuar con los datos y los encargados de ofrecer recursos internos dentro del mismo.

4.1.1 Infraestructura

El primer aspecto a tener en cuenta es que un prototipo de este tipo necesita de gran cantidad de recursos para poder ejecutarse de forma correcta, por tanto, el primer paso consistió en reservar recursos de cómputo dónde posteriormente desplegar los distintos componentes que conforman el prototipo.

El prototipo desarrollado fue desplegado en un servidor con las siguientes características:

- 16 CPU x 2.10 GHz
- 140 GB RAM
- 5 TB SSD
- GPU RTX 4070

Tal y como se puede observar, el servidor dispone de una gran cantidad de memoria RAM para poder emplear modelos de DL así como una GPU para poder hacer uso de las capacidades de cómputo específicas que ofrecen éstas y permiten un procesamiento más rápido de las distintas técnicas de AI, a costa de un mayor consumo energético. Además, la persistencia de la información se ha realizado empleando unidades de estado sólido (SSD, del inglés *Solid State Drives*) en vez de unidades de disco duro (HDD, del inglés *Hard Disk Drives*), ya que la diferencia entre ambas tecnologías tiene una gran repercusión en tareas de ML.

4.1.2 Lenguaje de programación

Hoy en día se puede encontrar una gran variedad de lenguajes de programación, empleándose cada uno ellos, con sus ventajas e inconvenientes, en ámbitos de uso específicos. Realizando una valoración de las más empleados a día de hoy (como podría ser C++, C#, Rust, Kotlin, Go o Python, entre otros), se ha decidido emplear Python para el desarrollo del prototipo por los motivos expuestos a continuación.

Python es un lenguaje de programación interpretado (no compilado) y no tipado. Al ser un lenguaje interpretado, ofrece mecanismos para recargar programas previamente cargados, por lo que lo hace perfecto para así poder permitir a los usuarios realizar las modificaciones oportunas a los programas desde elementos externos de forma transparente. En contrapartida, los lenguajes interpretados tienen tiempos de ejecución más largos, no obstante, muchos paquetes de Python realmente no ejecutan código escrito en Python, sino que internamente hacen llamadas a librerías escritas en C++ o Rust, cuyos tiempos de ejecución son muy cortos, con lo que se combina la capacidad de realizar modificaciones de forma sencilla aportada por Python con la velocidad de las librerías. Asimismo, al ser un lenguaje no tipado, lo hace muy sencillo para aquellas personas que no son expertas programadoras, sino que emplean los lenguajes de programación como herramienta para que les ayude en su trabajo, con la contrapartida de errores en tiempo de ejecución. Para mejorar con este problema, Python tienen un sistema al que llaman sugerencias de tipado (en inglés, *type hint*) que permite a los programadores más avanzados describir cuales son lo tipos empleados por variables y funciones aunque, como bien indica su nombre, solo son sugerencias, es tarea del programador asegurarse de que en todo momento está describiendo estos tipos como corresponde.

Por las peculiaridades mencionadas anteriormente, Python es el lenguaje de programación que se emplea de forma generalizada tanto para análisis de datos como para AI.

4.1.3 Conjunto de componentes de los datos

Colectores de datos

Para el prototipo responsable de de realizar la evaluación de la arquitectura se han desarrollado colectores de datos encargados de obtener todo aquel contenido necesario para realizar los análisis de ML en las distintas pruebas a realizar posteriormente.

En cuanto a la filosofía de recolección de datos de estos colectores, se han desarrollado tanto soluciones de tipo petición y respuesta recurrente (debido al hecho de que no todas las fuentes tienen la capacidad de la notificación de actualizaciones) como soluciones de tipo publicación y suscripción, empleándose esta última siempre que ha sido posible.

Por otro lado, el modo de recolección de datos implementado ha sido el de actualización, ya que tal y como se ha definido el modelo de datos, no se ha requerido en ninguno de los distintos tipos de datos la necesidad de emplear la solución de tipo espejo.

Se han desarrollado colectores de datos para fuentes de tipo SIEM, más precisamente, Arcsight, Ossim, Qradar y Gloria; por otro lado, también se han desarrollado integraciones para plataformas de respuesta a incidentes de seguridad, TheHive y Lucia; otro tipo de recolectores de datos que han sido implementados han sido para plataformas de inteligencia de amenazas, MISP y OpenCTI; también se ha realizado la integración de la herramienta de detección de amenazas persistentes avanzadas (APT, del inglés *Advanced Persistent Threats*) Carmen; y finalmente, aunque no por ello menos importante, se han desarrollado colectores encargados de recibir registros, PCAP e informes de TH.

Base de datos

Con respecto a la base de datos encargada de realizar la persistencia del contenido recolectado de las distintas fuentes así como de la información e inteligencia generada durante los procesos de AI, se ha decidido emplear Elastic Search, tal y como se propone en la definición de la arquitectura, al estar pensada para trabajar con grandes cantidades de datos y permitir realizar tanto consultas de lectura muy complejas como consultas continuadas en el tiempo mediante los *data streams*.

Para el prototipo generado se ha decidido emplear un único nodo de Elastic Search ya que, al ser un servicio externo, no se ha considerado relevante para el trabajo realizar tareas de verificación y validación de las capacidades de escalado de la base de datos.

Modelo de datos

Con respecto al modelo de datos, se ha decidido emplear ECS, tal y como se propone en la definición de la arquitectura. La definición de este modelo es pública, con lo que cualquiera puede hacer uso de la misma, además, permite que sea más fácil la compartición de información entre distintas aplicaciones que lo empleen, siempre y cuando se ciñan a la descripción de cada uno de los campos.

ECS es un modelo de datos jerárquico, donde los distintos niveles de la jerarquía se delimitan empleando el signo de puntuación punto (.). De las distintas definiciones dentro de ECS, se ha decidido trabajar con los campos relacionados con eventos y registros y, a partir de éstos, obtener la información relacionada con equipos, redes de comunicaciones u otras consideradas de interés.

En la tabla 4.1 se encuentra una muestra de los campos ECS más destacados empleados en el prototipo.

Componentes de preprocesado de datos

Para las distintas técnicas de ML a emplear en el prototipo (descritas a continuación), se ha requerido de la definición de componentes de preprocesado de datos específicos para las mismas. De entre los definidos, cabe destacar, entre otros:

- **Los encargados de realizar conversiones de tipo Sigma**: Estos componentes son capaces de convertir reglas Sigma [194, 265] a consultas de base de datos
- **Los encargados de realizar limpieza de los datos**
- **Los encargados de realizar transformaciones de los datos**: Entre los que se puede incluir tanto *One-Hot encoders* y *Leave-one-out encoders* como soluciones basadas en PCA
- **Los encargados de realizar la normalización numérica**: Tanto a valores normales como otros configurables para forzar que ciertos atributos tengan más error, buscando así que se las técnicas de ML entrenen más dichos parámetros
- **Los capaces de dividir textos en porciones**
- **Los encargados de recibir ficheros de audio y convertirlos a texto**
- **Los encargados de realizar la normalización de textos**

Tabla 4.1: Campos ECS más destacados del modelo de datos.

Campo ECS	Definición
event.dataset	Nombre del conjunto de datos al que pertenece el evento
event.id	Identificador que describe el evento
event.ingested	Marca temporal cuando el evento fue recibido
event.created	La fecha y hora cuando el evento fue leído por primera vez
event.starts	La fecha y hora cuando el evento empezó
event.end	La fecha y hora cuando el evento finalizó
event.action	La acción capturada por el evento
event.original	Texto en crudo del evento en su totalidad
source.ip	Dirección IP (IPv4 o IPv6) de la fuente
source.mac	Dirección MAC de la fuente
source.port	Puerto de la fuente
source.hostname	Nombre de la máquina de la fuente
destination.ip	Dirección IP (IPv4 o IPv6) del destino
destination.mac	Dirección MAC del destino
destination.port	Puerto del destino
destination.hostname	Nombre de la máquina del destino

Componentes de aprendizaje automático

Tal y como se ha introducido en secciones anteriores, la cantidad de técnicas de AI existentes en la actualidad es muy diversa y la evolución y creación de las mismas se produce con gran frecuencia, por tanto, es crucial estar siempre al día de las últimas novedades en cuanto a técnicas se refiere y la posible aplicación de las mismas.

En este aspecto, para realizar el *Threat Hunting* mediante este prototipo, se ha considerado relevante implementar los siguientes tipos de técnicas de ML y DL:

- **Agrupadores**: Componentes de *Spectral Clustering* (así como de sus variaciones, *Unnormalized Spectral Clustering* y *Normalized Spectral Clustering*) y de DBSCAN
- **Redes neuronales**: Componentes de tipo C-RNN-GAN (como una posible solución encargada de realizar tanto clasificación como generación de datos sintéticos) y redes neuronales basadas en grafos
- **Procesado del lenguaje natural**: Técnicas basadas en BERT, más específicamente, soluciones similares a CyBERT y MalBERT
- **Modelos de lenguaje de gran tamaño**: LLM para generar inferencias acerca de un cierto contexto basado en las entradas que reciben los mismos

Con respecto a los LLM, éstos son procesos extremadamente exigentes en recursos y fuera de las capacidades de un grupo de investigación, sólo al alcance de grandes corporaciones. Un investigador o un entusiasta puede entrenar desde cero un modelo simple, pero únicamente le servirá para generar, por ejemplo, texto que tiene todo el estilo de los clásicos del siglo de Oro Español, si lo ha entrenado con conjuntos de datos provenientes de obras de este periodo, o para realizar tareas de análisis (como por ejemplo, predicción del siguiente token) pero que no permitirá ir más allá de eso.

Sin embargo, para modelos como ChatGPT (en cualquiera de sus versiones) [164] o los provistos gratuitamente a la comunidad OSS por parte de Meta, las costes son muchos ordenes de magnitud mayores. Por ejemplo, el modelo Llama2 [260] de Meta, según se detalla en su carta de modelo, tuvo un coste de entrenamiento aproximado de 4.6 millones de dólares y hubiera consumido 233 años de computación en una GPU de alta gama, teniendo además una huella de carbono muy relevante. Por el motivo anteriormente expuesto, para poder entrenar este tipo de sistemas o particularizarlos a los conjuntos de datos que necesite el prototipo, otro tipo de aproximaciones se deben de llevar a cabo.

Una primera aproximación es la de hacer *fine-tuning* de un LLM previamente entrenado. De esta manera, se parte de un LLM completo, como Llama2 [260] y éste se entrena con conjuntos de datos propios, con lo que al finalizar dicho entrenamiento se consiguen todas las ventajas del (inasumible) LLM pero particularizado al contexto de operación deseado.

Otra aproximación son lo que se conoce como RAG (del inglés, *Retrieval Augmented Generative AIs*) [147] que consiste en introducir grandes conjuntos de datos de interés en un LLM. Este proceso debe ser debidamente segmentado, atendiendo al tamaño máximo de tokens que acepta cada modelo (parámetro definido en la carta de modelo de cada uno de estos), permitiendo inyectarle al dicho modelo preentrenado todo tipo de informaciones, desde registros, hasta ficheros, pasando por imágenes incluso, si, por ejemplo, el modelo tiene capacidades de segmentación, análisis y reconocimiento óptico de caracteres (OCR, del inglés *Optical Character Recognition*) multimodales. Normalmente, este tipo de acciones de RAG se llevan a cabo haciendo uso de plataformas a tal efecto como LangChain [259] y LlamaIndex [155] y, los datos, independientemente de su tipología original, suelen vectorizarse y almacenarse en bases de datos que soporten extensiones vectoriales, como por ejemplo, PostgreSQL, Neo4J o SurrealDB.

Estadísticas de Big Data

El módulo de estadísticas de *Big Data* desarrollado es capaz de almacenar tantas generaciones de información como el usuario requiera. Para ello, el analista es capaz de configurar los parámetros de las mismas (ya sea, entre otros, las consultas a la base de datos, como se debe actualizar dicha información, si hay que mantener un histórico o no).

De este módulo, algunas de las estadísticas más empleadas por parte de los analistas han sido:

- **Cuáles son los tipos de ataques con mayor frecuencia de ocurrencia**
- **Cuáles son los ataques con mayor impacto**
- **Cuáles son los dispositivos más atacados junto con su nivel de riesgo**
- **Cuáles han sido los dispositivos que normalmente no son atacados pero si que lo han sido recientemente**

Intercambiadores de datos

Con respecto a los componentes encargados de proveer información filtrada tanto al HMI como a la puerta de acceso externo, para este prototipo, éstos se han desarrollado de forma que reciben una petición en formato GraphQL y devuelven la información empleando las representación de los datos JSON mediante dos posibles estructuras de contenido:

- Siguiendo la estructura del modelo de datos, siendo cada uno de los niveles del mismo un identificador dentro del objeto JSON hasta, finalmente, llegar al valor.
- Empleando el estándar STIX

Generadores de hipótesis

Los componentes generadores de hipótesis implementados han sido capaces de emplear tanto reglas básicas de generación de hipótesis basadas en firmas generadas por los analistas como de emplear los componentes de aprendizaje automático para ofrecer al analista de seguridad hipótesis sobre lo que está sucediendo.

De entre los componentes de aprendizaje automático empleados, cabe destacar los LLM por su capacidad de generación de varias salidas alternativas a partir de los mismos datos de entrada. Ponderándose las probabilidades de suceso para cada una de las mismas, se consigue ofrecer al analista puntos de vista que pudiera no haber considerado asociados con su probabilidad de ocurrencia.

Diseño de secuencias de aprendizaje automático

Con respecto a los componentes de diseño de secuencias de aprendizaje automático en el prototipo implementado, éstos han usado la base de datos para el almacenamiento del estado de las distintas secuencias de ML para así conseguir que se pudieran desplegar tantos de éstos como fuese necesario. Además, éstos también han empleado un repositorio de tipo GIT interno para la gestión de los distintos versionados de secuencias generadas por los especialistas en ML.

Se considera importante destacar que se ha empleado un repositorio de versionado para que los analistas sean capaces tanto de emplear secuencias de ML previas a la última versión (en caso de que los resultados generados por las mismas no sean los esperados) como para que se pueda realizar un seguimiento de la evolución de las mismas y, en caso de que se necesitase, realizar una auditoria.

HMI

Analizando las distintas alternativas con respecto al componente HMI en cuanto a generación del contenido se refiere, se ha optado por desarrollar una solución donde el servidor genera el contenido a representar y el cliente es el encargado de realizar las representaciones, conocida como *API First*. Se ha realizado esta elección para permitir que, en caso de que se quiera realizar automatizaciones de los procesos, se puedan realizar todas las consultas que va a realizar el usuario (mediante el cliente) haciendo uso de peticiones API REST. Además, el servidor responde a las peticiones empleando la representación de los datos JSON, que, aunque no es tan eficiente como otras alternativas, es la más versátil, liviana y fácil de implementar.

Con respecto al cliente del HMI, éste ha sido desarrollado como una aplicación de página única (SPA, del inglés *Single Page Application*), haciendo uso de React (una biblioteca para interfaces de usuario web y nativas) y empleando el lenguaje TypeScript (desarrollado por Microsoft como extensión del lenguaje JavaScript).

El HMI desarrollado es capaz de:

- **Realizar configuraciones**: Los distintos componentes con los que cuenta la arquitectura tienen parámetros de configuración cuya modificación está centralizada en el HMI
- **Interactuar con componentes auxiliares**: La interacción con los distintos componentes auxiliares (como almacenes de modelos de datos y repositorios de códigos) está centralizada en el componente HMI
- **Permitir interactuar con líneas de código**: Para permitir a los analistas interactuar de forma fácil e intuitiva con los distintos códigos generados para ser ejecutados por los distintos componentes con los que cuenta el prototipo, se ha hecho uso de un entorno de desarrollo integrado (IDE, del inglés *Integrated Development Environment*) basado en web que ofrece una experiencia similar a la de trabajar con IDE basados en programas de escritorio.
- **Mostrar visualizaciones**: Toda la información generada por los distintos componentes se debe poder visualizar de forma adecuada para cada uno de los distintos tipos de usuario, en consecuencia, debe haber desde visualizaciones de muy alto nivel para los usuarios de tipo estratégico basadas en gráficas mostrando agregación de la información más relevante, hasta visualizaciones muy detalladas para los usuarios de tipo táctico basadas en tablas.

Puerta de acceso externo

Para permitir a aplicaciones externas interactuar con la información del prototipo, ya sea, consultar, añadir o editar la misma, se ha desarrollado la puerta de acceso externo. En cualquier sistema que trabaja con información relacionada con la seguridad de los distintos componentes y recursos de una organización, este elemento es crucial para estar al día de las posibles amenazas que están afectando a otras organizaciones consideradas amigas, aunque, por otro lado, también es un punto de ataque al proveer información sensible de la organización que busca proteger, por este motivo, este elemento tiene asignadas las tareas de:

- **Realización de los procesos necesarios de autenticación y autorización de las peticiones**
- **Procesamiento de las peticiones**
- **Peticiones de la información a los componentes intercambiadores de datos**
- **Respuesta a las peticiones recibidas**

4.1.4 Conjunto de componentes de los recursos

Comunicaciones

Las comunicaciones realizadas entre los distintos componentes del conjunto de los datos se ha realizado mediante el uso del agente de comunicaciones RabbitMQ. Se ha escogido este agente de comunicaciones para el prototipo por su capacidad de enrutamiento compleja de los mensajes, lo que permite realizar tanto peticiones *unicast* como *multicast* de forma nativa.

Asimismo, mediante las capacidades de RabbitMQ y siguiendo una definición específica de la nomenclatura de la clave de enrutado de los vínculos, se ha garantizado que las comunicaciones del conjunto de componentes de los datos únicamente se produzca entre elementos de capas adyacentes o de la misma capa, requerimiento de la filosofía basada en capas lógicas que sigue dicho grupo de componentes.

Gestión de la autenticación y de la autorización

Para gestionar la autenticación y la autorización tanto de los componentes del prototipo como de los usuarios (internos y externos) del mismo, se ha decido emplear el servicio Authentik.

Authentik ofrece un servicio tipo API REST para la configuración del mismo mediante interacciones M2M que ha sido empleado por los componentes del HMI para realizar las consultas y modificaciones necesarias.

En cuanto a protocolos se refiere, se ha hecho uso del protocolo OpenID Connect para realizar la autenticación y el protocolo OAuth 2.0 para realizar la autorización. Cabe destacar que en Authentik ambos protocolos se aglutinan en un único proveedor.

En cuanto a la autorización de los componentes, Authentik es el encargado de configurar cuales son los permisos de RabbitMQ para asegurar el enrutado solo entre componentes de capas adyacentes y, en cuanto a la autorización de los usuarios, Authentik ha sido el encargado de establecer que recursos son accesibles para cada uno de los distintos roles con lo que se ha configurado el prototipo.

Finalmente, en cuanto a modelo de control de accesos, se ha empelado RBAC.

4.2 Validación

Las primeras pruebas realizadas a la arquitectura propuesta mediante el prototipo desarrollado ha sido la validación de que la definición de cada uno de los componentes de la misma es funcional, es decir, que al recibir los datos de entrada esperados se obtiene como salida un resultado acorde a las especificaciones. Para ello, se han llevado a cabo este tipo de pruebas en cada uno éstos y, en esta sección del trabajo, se analizarán los resultados obtenidos tras realizar la ejecución de dichas pruebas.

Cabe destacar que estas pruebas únicamente se han realizado a aquellos componentes desarrollados para el prototipo y no para los servicios, al considerarse que dichos servicios ya han sido sometidos a éstas en el proceso de desarrollo de los mismos.

4.2.1 Conjunto de componentes de los datos

Colectores de datos

Los primeros componentes a los que se les ha realizado las pruebas de validación ha sido a los colectores de datos, ya que sin éstos funcionando correctamente el sistema no puede obtener el contenido con el que trabajar.

En las pruebas de validación de éstos se ha asegurado que, para cada una de las fuentes de datos, estos componentes realizan la adquisición del contenido y envían las peticiones correctas para almacenar la información recolectada.

Componentes de preprocesado de datos

El siguiente elemento fundamental para el funcionamiento del prototipo es la capacidad de preprocesado de la información almacenada en la base de datos para las distintas técnicas de ML, ya que si éstos no funcionan bien las técnicas de AI empleadas tampoco devolverán los resultados esperados.

En las pruebas de validación realizadas a estos componentes se ha asegurado que para cada uno de los conjuntos de datos recibidos por éstos a su entrada se obtenía como salida los datos esperados.

Un ejemplo de las pruebas realizadas a éstos serían las consistentes en la validación de los elementos encargados de realizar las transformaciones de los datos, más específicamente, aquellas realizadas a los *One-Hot encoders*, donde tras introducir datos con clasificaciones basadas en elementos textuales (como “Benigno” y “Maligno”), en la salida se sustituían dichos valores por dos nuevos campos donde uno de ellos tenía el valor uno (1) y el otro tenía el valor cero (0).

Componentes de aprendizaje automático

Con respecto a los componentes de aprendizaje automático se han llevado a cabo dos tipos distintos de pruebas de validación.

Las primeras de ellas han consistido en (i) asegurarse de que los distintos componentes eran capaces de recibir la información tal y como se espera que la recibieran, (ii) verificar que las distintas técnicas se ejecutasen sin problemas y, que tras realizar dicho proceso, (iii) se realizaba la petición para hacer llegar la información al siguiente elemento de la cadena y (iv) se notificaba al componente de diseño de secuencias de ML que la ejecución había resultado satisfactoria.

Las segundas han consistido en tener la certeza de que cuando sucede cualquier tipo de inconveniente estos componentes son capaces de notificar al componente de diseño de secuencias de ML del mismo. Para ello, se han preparado sistemas con fallos y se ha validado que, tras un error, toda la información de dicho error se hacía llegar al componente correspondiente para que los expertos tengan conocimiento de lo que ha sucedido y puedan realizar las correcciones necesarias.

Estadísticas de Big Data

Los componentes de estadísticas *Big Data* han sido sometidos a un proceso de validación muy exhaustivo para tener la certeza de que realizan sus operaciones de consultas y procesado de forma eficiente debido a que la finalidad de su existencia en la arquitectura es la de asegurar la eficacia de las interacciones con enormes cantidades de datos mediante la liberación de carga de trabajo y reutilización del procesado de la información entre distintos componentes con necesidades muy parecidas.

En este aspecto, estos componentes han sido validados realizando tanto consultas simples como complejas y con distintos niveles de llenado de la base de datos, es decir, haciéndoles procesar más o menos información. Asimismo, también se ha asegurado que frente a un cambio en el contenido de dichas consultas, estos componentes son capaces de volver a realizar los procesados de la información necesarios así como de informar a aquellos componentes suscritos al contenido generado por éstos que se ha realizado dicha actualización.

Intercambiadores de datos

Para la validación de los componentes intercambiadores de datos se ha certificado que estos componentes han sido capaces de recibir una estructura tipo GraphQL, han realizado las consultas y procesamientos necesarios y, finalmente, han devuelto la información siguiendo la estructura de contenido solicitada.

Se considera importante destacar que se ha realizado una validación minuciosa cuando se ha solicitado que la estructura de las respuesta de los datos siguiese el formato STIX debido a que para conseguir la correcta interoperabilidad entre distintas aplicaciones es primordial que todas ellas implementen de forma correcta los distintos estándares.

Generadores de hipótesis

Tal y como se ha analizado con anterioridad, los componentes generadores de hipótesis tienen dos vertientes, una en la que únicamente realizan sus suposiciones basándose en firmas o consultas simples y otra donde se emplean técnicas avanzadas de ML para generar las mismas, habiendo realizado las pruebas de validación para ambas vertientes tal y como se detalla a continuación.

Con respecto a la primera vertiente, en las pruebas de validación se ha comprobado que estos componentes pueden realizar de forma satisfactoria las consultas necesarias a la base de datos basándose en la configuración del usuario así como procesar las respuestas de ésta para realizar la generación de hipótesis.

Para la segunda vertiente, se ha asegurado que los elementos generadores de hipótesis son capaces de realizar las peticiones necesarias al componente de diseño de secuencias de aprendizaje automático con el objetivo de crear (en caso de que no existieran) las secuencias de ML, ejecutar y, finalmente, almacenar el resultado obtenido en la base de datos. Finalmente, se ha verificado que el componente era capaz de recibir la información de la base de datos y realizar cualquier procesamiento extra configurado por el TH.

Diseño de secuencias de aprendizaje automático

Los componentes de diseño de secuencias de aprendizaje automático, tal y como se ha visto con anterioridad, son los encargados de las funciones del diseño de los flujos de ML, de la ejecución de los distintos flujos de ML y de la monitorización del estado de ejecución de los flujos, en consecuencia, el correcto funcionamiento de los mismos es primordial para que el conjunto de los elementos de la arquitectura funcionen de forma satisfactoria, por este motivo, la validación de todas y cada una de las funciones de este componente se ha considerada de gran relevancia.

Las primeras pruebas de validación de este componente han consistido en asegurar que éstos son capaces de recibir peticiones para la creación, edición y borrado de secuencias así como la ejecución satisfactoria de las mismas.

A continuación se ha comprobado que los componentes son capaces de ejecutar los flujos acorde a las configuraciones de los usuarios, analizando tanto cuando se realizaban las ejecuciones como el contenido que se envía al primero de los elementos del flujo.

Para proseguir se ha constatado que estos componentes tienen la capacidad de recepción de actualizaciones del estado de ejecución de los distintos flujos.

Finalmente, se ha comprobado que éstos ejecutan las peticiones para realizar la persistencia del estado de los flujos así como que son capaces de proveer dicha información al ser solicitada.

HMI

Las pruebas de validación del componente HMI han sido las más laboriosas debido a toda la casuística que este componente lleva asociado. Éstas se han divido en dos grupos, uno para el servidor y otro para el cliente, analizados a continuación.

Con respecto a las pruebas del servidor, se han analizado todos y cada uno de los recursos que expone el mismo para asegurar que éstos cumplen con sus funciones correctamente. Acto seguido, se han repetido dichas pruebas con distintos usuarios (cada uno con distintos roles) para evaluar tanto que los mecanismos de autenticación y autorización funcionan correctamente como que los roles han sido asignados correctamente a los recursos.

Con respecto a las pruebas del cliente, en primer lugar se ha comprobado que cuando el usuario emplea un dispositivo inválido, este cliente le notifique a dicho usuario que el dispositivo que está usando no se puede emplear para interactuar con la aplicación. A continuación, se han realizado las pruebas pertinentes para asegurar que la aplicación se adapta a los distintos dispositivos para los que ha sido pensada. Para proseguir, se ha verificado que la interfaz con la que interactúa el usuario tiene una buena UI así como que ofrece una buena UX. A continuación, se ha realizado un análisis de cada uno de los distintos menús de configuración, de los distintos mecanismos de edición y de las distintas visualizaciones para certificar que el funcionamiento y la interoperatividad es la esperada. Finalmente, se han repetido dichas pruebas con distintos roles para asegurar que únicamente se muestran los recursos específicos para cada rol y que en caso de se intente acceder a un recurso no permitido se muestre al usuario el mensaje de error correspondiente.

Puerta de acceso externo

El último de los componentes al que se le han realizado pruebas de validación ha sido a la puerta de acceso externo. El funcionamiento de este componente es crítico para asegurar la interoperatividad entre el prototipo y aplicaciones de terceros. Cabe hacer especial énfasis en que se debe asegurar en todo momento la integridad y confidencialidad de la información intercambiada por su sensibilidad.

Las pruebas de validación de este componente han consistido en la certificación de que no hay modos de evitar la autenticación al solicitarle información al mismo así como que no se pueden ejecutar ataques en el proceso de autenticación de las peticiones. A continuación, también se ha validado que este componente es capaz de realizar las solicitudes pertinentes a los componentes intercambiadores de datos, filtrar el contenido recibido de éstos acorde a los permisos del solicitante y enviarle la información al mismo siguiendo el estándar correspondiente.

4.3 Verificación en entorno controlado

Una vez concluidas las pruebas de validación de la arquitectura, el siguiente paso consistió en la realización de las pruebas de verificación de ésta en un entorno controlado. Para llevarlas a cabo, se hizo uso de los recursos de un laboratorio del departamento de comunicaciones de la Universitat Politècnica de València.

Mientras que en las pruebas de validación se buscaba asegurar el correcto funcionamiento de cada uno de los componentes por separado, en las pruebas de verificación se busca evaluar la funcionalidad de la arquitectura en su totalidad (es decir, empleando todos los componentes que conforman la misma) para así confirmar que es válida para dar solución al problema por el que ha sido desarrollado.

Para realizar estas pruebas de verificación en entorno controlado se ha sometido el prototipo desarrollado con anterioridad a seis casos de uso reales. Éstos se enumeran a continuación y cada uno de ellos será analizado con detenimiento en esta sección:

1. Generación de agrupaciones de registros relacionadas con las técnicas de MITRE Attack empleadas por APT
2. Generación de hipótesis mediante redes neuronales basadas en grafos
3. Detección y mapeo de amenazas a MITRE Attack con LLM
4. Generación de grafos de conocimiento con LLM
5. Detección de amenazas a partir de registros con LLM
6. Extensión de la capacidad de los LLM para el análisis de registros. Uso de la ruptura de la simetría

4.3.1 Primer caso de uso: Generación de agrupaciones de registros relacionadas con las técnicas de MITRE Attack empleadas por APTs

La primera de la pruebas de verificación en entorno controlado consistió en alimentar la base de datos con registros relacionados con técnicas empleadas por APT buscando generar un sistema que fuese capaz de clasificar dichos registros por técnica empleada así como indicar también las posibles APT que hacen uso de la misma.

En esta prueba de verificación se asume que los datos ya han sido colectados y almacenados, por dicho motivo, en esta prueba no se ha realizado una verificación de los colectores de datos.

Para la ejecución de ésta, el procedimiento realizado por el experto en ML consistió, en primer lugar, en la generación de una secuencia de ML, la cuál se encuentra representada gráficamente en la figura 4.1. Ésta consiste, primeramente, en una fase dónde se le indica al primer componente que debe obtener la información de la base de datos, a continuación, se ejecutan una serie de normalizadores de textos (para eliminar palabras de parada y separar frases, entre otros), para proseguir, se realiza un primer procesado mediante técnicas de NLP con el objetivo de obtener distribuciones y frecuencias de palabras y, finalmente, se realizan agrupaciones de dichas palabras y se relacionan con técnicas de MITRE Attack empleadas por APT.

Figura 4.1: Primer caso de uso: Generación de agrupaciones de registros relacionadas con las técnicas de MITRE Attack empleadas por APTs: Secuencia de aprendizaje automático

Una vez procesado el contenido almacenado en la base de datos por la secuencia de ML diseñada por el expero en ML, el siguiente paso consistió en generar una estadística de *big data* encargada de transformar la información generada por dicha secuencia en contenido representable para así poder visualizar desde el HMI dicha información en un diagrama de dos dimensiones, tal y como se puede observar en la figura 4.2 y en la figura 4.3.

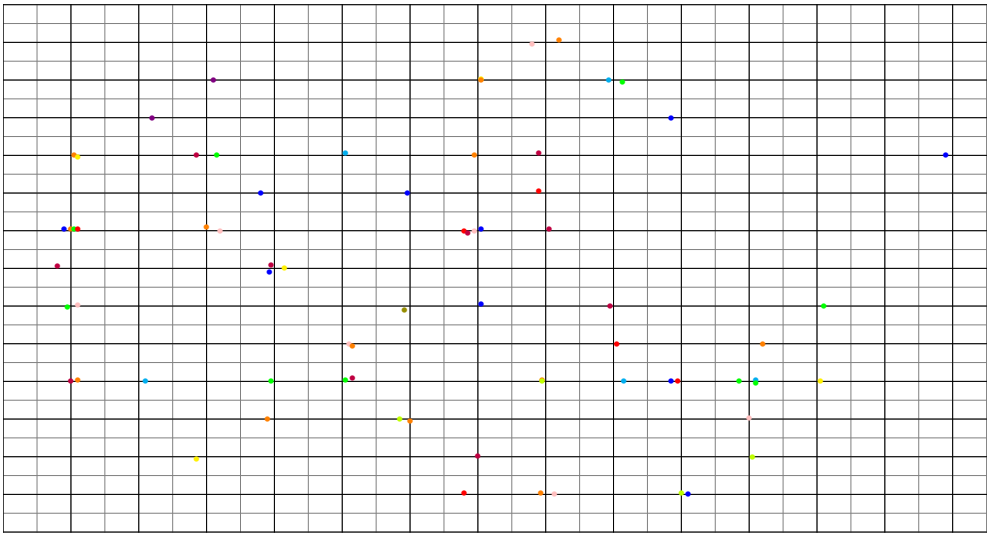

Figura 4.2: Primer caso de uso: Generación de agrupaciones de registros relacionadas con las técnicas de MITRE Attack empleadas por APTs: Diagrama

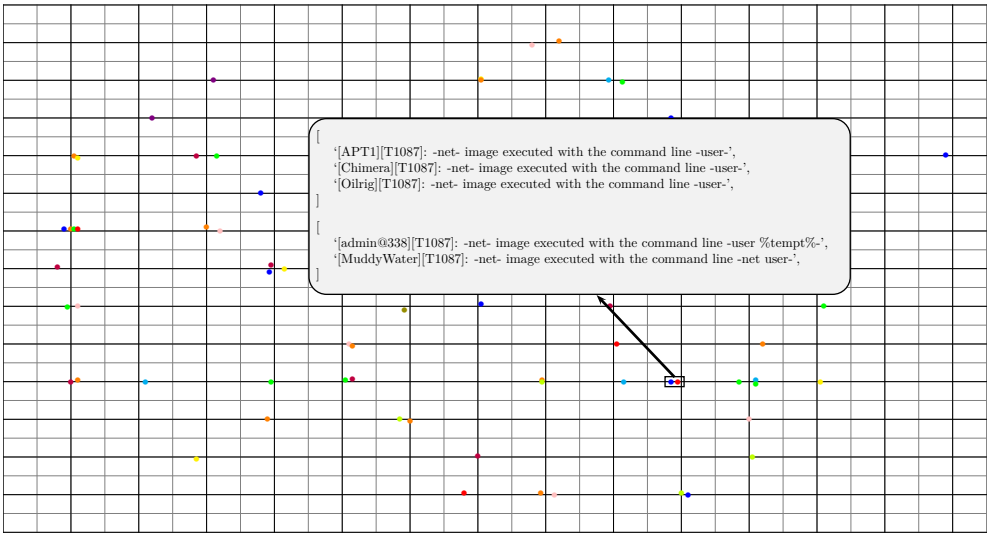

Figura 4.3: Primer caso de uso: Generación de agrupaciones de registros relacionadas con las técnicas de MITRE Attack empleadas por APTs: Diagrama con leyenda

4.3.2 Segundo caso de uso: Generación de hipótesis mediante redes neuronales basadas en grafos

En este caso de uso se ha realizado un supuesto donde se ha simulado el ataque de una APT y se ha empleado el prototipo desarrollado para la generación de las hipótesis acerca de lo que estaba sucediendo en la infraestructura monitorizada.

Para simular este caso de uso, se ha generado un escenario simulado donde se ha desplegado el prototipo desarrollado junto con dos redes simulando los recursos de una organización (una de ellas con recursos de los empleados y otra de ellas simulando una red desmilitarizado con servicios expuestos al exterior) y una red simulando internet (desde la que se han realizado los ataques simulando el *modus operandi* de una APT). Una representación gráfica del escenario simulado se puede encontrar en la figura 4.4.

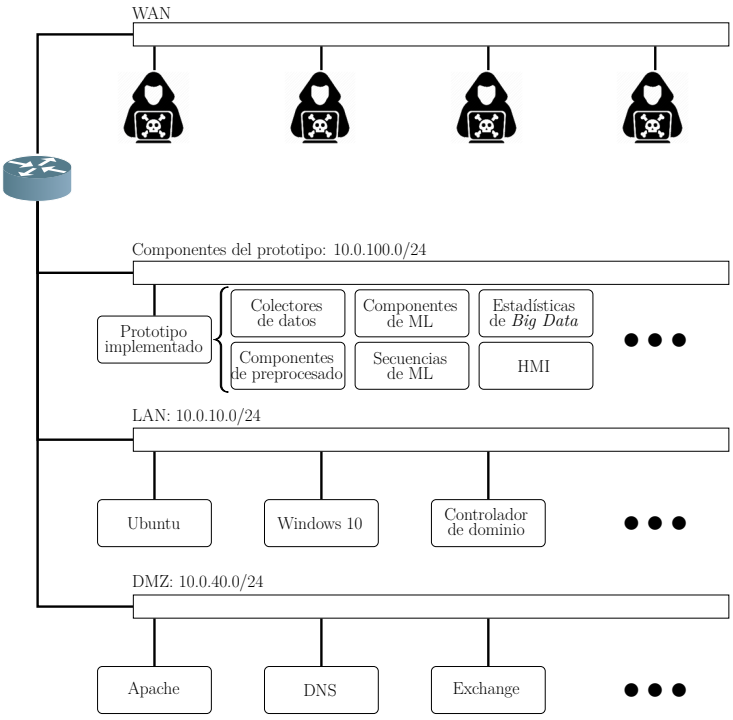

Figura 4.4: Segundo caso de uso: Generación de hipótesis mediante redes neuronales basadas en grafos: Escenario

Una vez desplegado el escenario simulado, el siguiente paso consistió en la instalación de sondas tanto en los equipos como en los dispositivos de red. Una vez instaladas, se configuraron los colectores de datos para que hicieran uso de dichas sondas para alimentar el sistema verificando así que recolectaban la información y realizaban la transformación al modelo de datos.

Para modelar la información recolectada y sobre la que se realizó el proceso de generación de hipótesis, se decidió emplear la propuesta realizada en [151] donde se sigue una aproximación de relaciones basadas en grafos donde se tienen en cuenta tanto los procesos internos de los equipos como los comunicaciones realizadas entre equipos. Dicha aproximación de relaciones basadas en grafos se puede encontrar representada gráficamente en la figura 4.5.

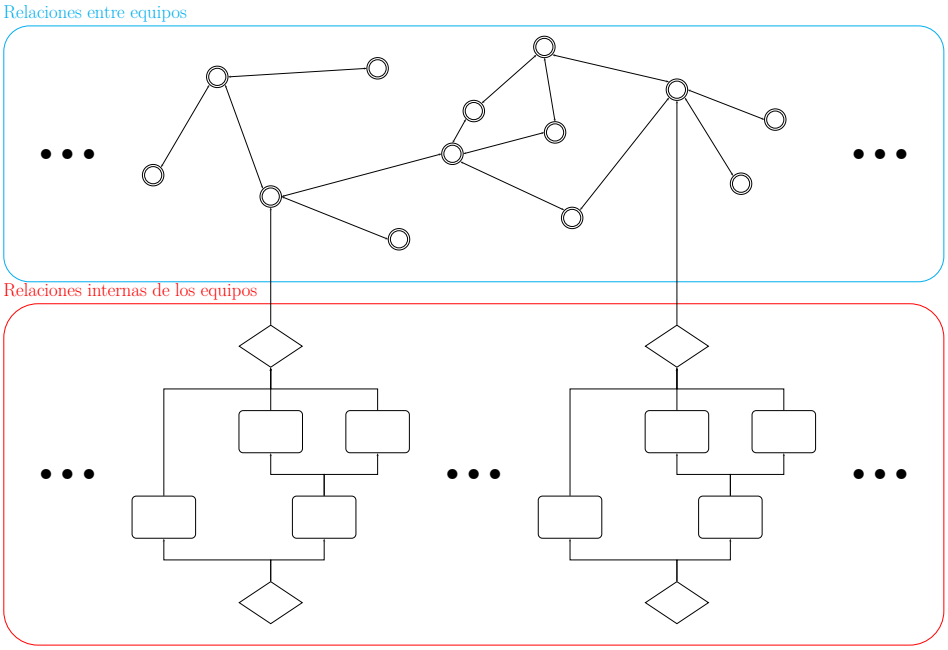

Figura 4.5: Segundo caso de uso: Generación de hipótesis mediante redes neuronales basadas en grafos: Modelo de las relaciones del contenido

Teniendo datos recolectados y habiendo escogido como modelar dicha información recolectada, el siguiente paso consistió en la generación de las secuencias de ML necesarias por parte del expero en ML siguiendo la propuesta realizada en [151].

Finalmente, se configuraron estadísticas de *big data* para procesar la información generada por la distintas técnicas de ML y poder ser visualizada en el HMI, obteniendo resultados como los representados en la figura 4.6.

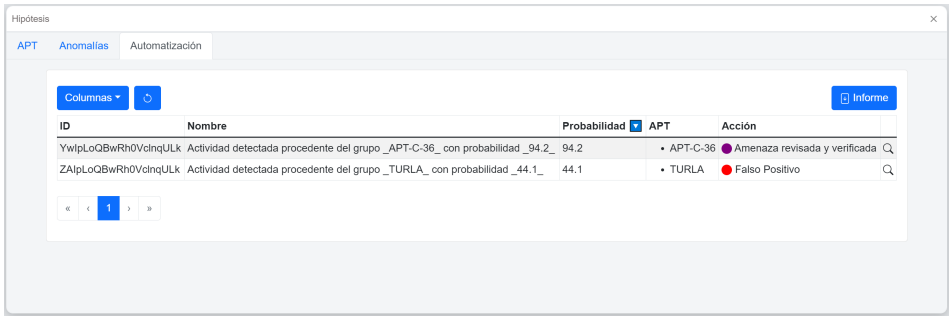

Figura 4.6: Segundo caso de uso: Generación de hipótesis mediante redes neuronales basadas en grafos: Visualización de la información

4.3.3 Tercer caso de uso: Detección y mapeo de amenazas a MITRE Attack con LLMs

Este caso de uso consistió en la detección y el mapeo de reportes de ciberinteligencia a MITRE Attack [208] mediante LLM, un caso de uso basado en la aproximación realizada por MITRE TRAM [91]. Los flujos de información de éste, que serán analizados con detenimiento a continuación, se encuentran representados gráficamente en la figura 4.7.

Los reportes de inteligencia a partir de los que se ha alimentado el sistema provienen de fuentes abiertas. Éstos son por naturaleza dispares, por este motivo, se decidió emplear LLM al ofrecer capacidades de NLP muy dispares y, en consecuencia, adaptables a este tipo de fuentes heterogéneas.

Algunos ejemplos de las fuentes empleadas para este caso de uso y, además, comúnmente empleadas por TH son:

- Welivesecurity
- Securelist
- AlienVault OTX
- Datos obtenidos y procesados haciendo uso de la herramienta OSS OpenCTI

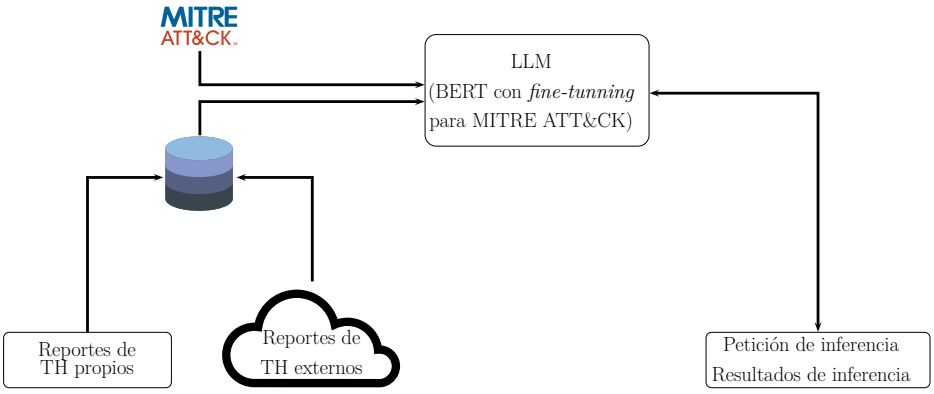

Figura 4.7: Tercer caso de uso: Detección y mapeo de amenazas a MITRE Attack con LLMs: Flujos de información

Para este caso de uso se ha empleado *fine-tuning*, en consecuencia, se pueden procesar los reportes de inteligencia o bien mediante un modelo entrenado con reportes propios de inteligencia, o bien mediante un modelo entrenado por la comunidad (como por ejemplo, los que se encuentran en la web de MITRE TRAM).

Para este caso de uso se ha decidido realizar el *fine-tuning* a BERT con conjuntos de datos cuya estructura sigue la que se puede encontrar en el código 4.1.

```
[
    {
        "text": "replacing an early call instruction to their malicious code, or by overwriting the entry point in the PE header",
        "label": "T1055",
        "doc_title": "Tailoring Sandbox Techniques to Hidden Threats"
    },
    {
        "text": "unpack any additional payloads,",
        "label": "T1027",
        "doc_title": "Tailoring Sandbox Techniques to Hidden Threats"
    },
    {
        "text": "HTTPS and other protocols built on SSL have",
        "label": "T1071.001",
        "doc_title": "Tailoring Sandbox Techniques to Hidden Threats"
    },
    {
        "text": "service using HTTPS",
        "label": "T1071.001",
        "doc_title": "Tailoring Sandbox Techniques to Hidden Threats"
    }
]
```

Código 4.1: Tercer caso de uso: Detección y mapeo de amenazas a MITRE Attack con LLMs: Estructura de los datos de entrenamiento

Tras el procesado de los datos, la información devuelta al TH podía indicar tanto que un cierto dato de entrada no había obtenido coincidencia con ningún ataque de la base de datos de MITRE ATT&CK como que si que la había obtenido, dando como resultado una respuesta como la que se puede encontrar en el código 4.2.


```
{
    "input_data": "Fake SecureVPN v1.0.9 (SecureVPN_109) with malicious code included into OpenVPN as its parent application even
       though the hardcoded VERSION_NAME (1.0.0) 'wasnt changed between versions Besides the split in these two branches, where the
       same malicious code is implanted into two different VPN apps, other fake SecureVPN version updates contained only minor code
       changes or fixes, with nothing significant considering its overall functionality.",
    "analysis_result": "T1140 - Deobfuscate/Decode Files or Information",
    "likeliness": ".992"
}
```


Código 4.2: Tercer caso de uso: Detección y mapeo de amenazas a MITRE Attack con LLMs: Ejemplo de resultado obtenido

Tras completar este caso de uso, los TH consideraron una muy buena aproximación el empleo de LLM haciendo uso de la arquitectura propuesta al considerar que gracias a la interoperatividad de los distintos componentes se consigue una reducción importante de esfuerzo (y por tanto, de tiempo) a la hora de analizar las ingentes cantidades de información textual que se ven obligados a procesar.

4.3.4 Cuarto caso de uso: Generación de grafos de conocimiento con LLMs

En la presente tesis se vislumbró el uso de grafos de conocimiento (KG, del inglés *Knowledge Graphs*) como una extensión de la detección y mapeo de amenazas a MITRE Attack con LLM en la que hacer uso de las capacidades de los LLM para NLP, más específicamente el reconocimiento de entidades nombradas (NER, del inglés *Named Entity Recognition*), para encontrar las entidades más destacadas así como sus relaciones, de esta forma se conseguiría ayudar a los TH mediante una herramienta capaz de analizar los reportes de inteligencia de manera automática generando como salida un conjunto de entidades destacadas (los conceptos clave) y sus relaciones, en resumen, esta herramienta estaría generando un KG (cuya definición se encuentra a continuación).

Un KG es tanto un concepto del análisis de información como un tipo de representación de los datos. En un KG cada nodo representa un concepto y cada enlace representa la relación entre un par de dichos conceptos, pudiéndose considerar también este tipo de representación como una representación de "grafo de conceptos". La utilidad de los KG es evidente, permite generar una representación visual de un dominio descriptivo (por ejemplo, un texto) donde no solo se destacan los conceptos más relevantes sino que también se visualizan en un grafo las relaciones entre los mismos.

Para este caso de uso, se decidió seguir una aproximación donde toda la información se almacenaba de forma local, por tanto se decidió emplear los LLM mistral 7B y Zephyr que no tienen la necesidad de acceder a servidores de terceros (como es el caso de ChatGPT). Asimismo, se empleó RAG (técnica descrita previamente) como técnica para realizar la vectorización de los contenidos de los documentos (como por ejemplo, un reporte de ciberinteligencia en PDF) y asi poder emplear dicha vectorización como contexto de un LLM para, acto seguido, permitirle al usuario realizarle preguntas a dicho modelo sobre los datos de su contexto. Cabe destacar que en este caso de uso no se ha realizado un entrenamiento con los datos del usuario (como se haría al emplear *fine-tuning*), únicamente se le proporciona al modelo la información mediante su contexto.

Hay que destacar que al usar RAG hay que tener siempre en cuenta el tamaño máximo de la entrada del LLM, es decir, cuántos tokens se le pueden pasar como contexto al modelo. Se considera relevante remarcar que la longitud de la entrada del modelo va a ser un aspecto muy relevante siempre que se trabaje con LLM (ya sea al solicitar tareas de traducción, de resumen, o incluso al realizar un ASR sobre una grabación de audio) lo que obliga casi siempre a realizar una división de la entrada, siendo recomendable dejar además un margen para tener una cierta holgura (en nuestras pruebas de un 5%).

El esquema general de RAG empleado se puede encontrar representando gráficamente en la figura 4.8.

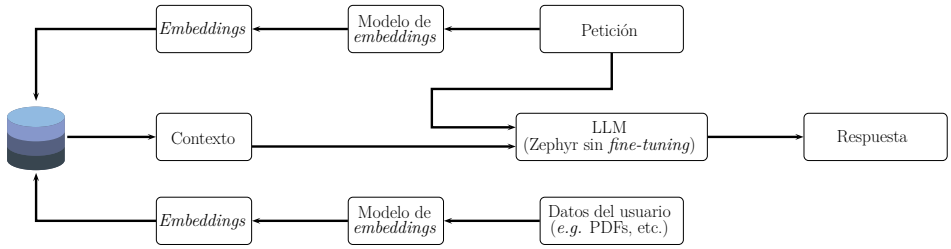

Figura 4.8: Cuarto caso de uso: Generación de grafos de conocimiento con LLMs: Esquema RAG

De esta forma, de manera iterativa se alimenta el modelo con subconjuntos de los datos del usuario y realizando un NER se van obteniendo las relaciones más relevantes entre los distintos activos que el modelo es capaz de detectar. Una vez concluido el proceso, a partir del vector de conceptos, sus pesos relativos y las relaciones entre los mismos se puede realizar un grafo como el que se ve en la figura 4.9.

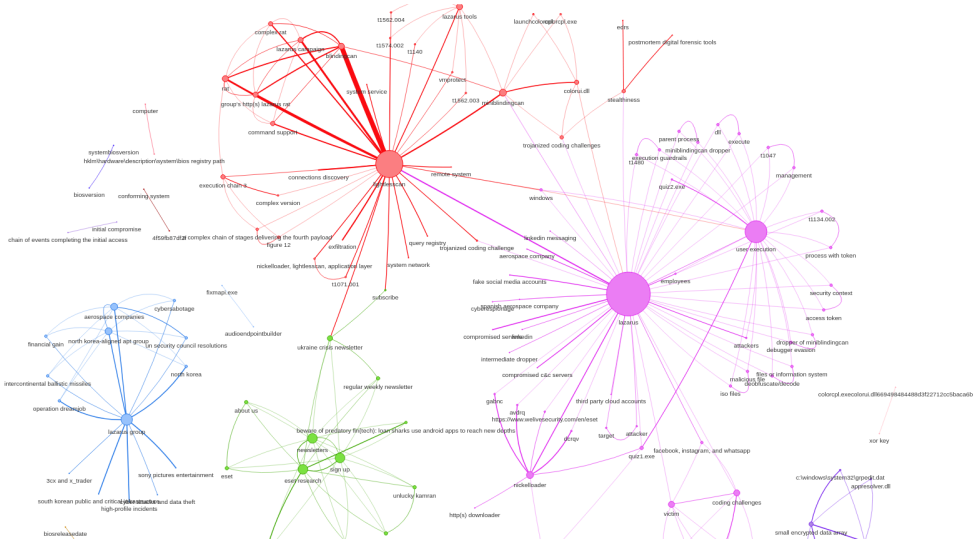

Figura 4.9: Cuarto caso de uso: Generación de grafos de conocimiento con LLMs: Grafo de conocimiento

Tal y como se ha visto, el caso de uso de los KG es una extensión del trabajo realizado con relación a la inferencia, detección y clasificación con LLM, aunque en este caso en particular, existe un progreso al obtener también los elementos (o conceptos) más destacados mediante un NER donde además se consigue determinar las relaciones entre estos elementos de forma que se le puede entregar al TH información adicional más allá del saber que está (probablemente) sucediendo.

4.3.5 Quinto caso de uso: Detección de amenazas a partir de registros con LLMs

Tras el trabajo realizado previamente relacionado con el uso de LLM para la detección y el mapeo de reportes de ciberinteligencia a MITRE Attack se decidió continuar con dicha linea de investigación con un planteamiento similar pero en vez de emplear reportes de ciberinteligencia como entradas hacer uso tanto de registros de máquinas Windows y Linux obtenidos de sistemas de detección de intrusiones a nivel de máquina (HIDS, del inglés *Host-based Intrusion Detection Systems*) como de tráfico de red proveniente de sistemas de detección de intrusiones a nivel de red (NIDS, del inglés *Network-based Intrusion Detection Systems*).

Siendo el procesamiento de secuencias de caracteres (y por extensión el NLP) un proceso propio de muchas tareas de ciberseguridad (como por ejemplo, el análisis de registros, el análisis de ficheros de configuración, el análisis de programas maliciosos y el análisis del grado de seguridad del código) es lógico pensar que un LLM ampliamente utilizado para el NLP también pueda ser de utilidad en muchas de estas tareas. A partir de esta suposición, en este caso de uso se decidió tomar como punto de partida los esfuerzos existentes basados en BERT (el LLM desarrollado por Google y liberado a la comunidad que ha sido ampliamente utilizado en tareas de NLP) y albergados en plataformas libres como "Hugging Face" de entre los que destacar CySecBERT (entrenado con información de ciberseguridad en general), SecureBERT (entrenado para Threat Intelligence) y CyBERT (orientado a información proveniente de dispositivos sensores como IoT).

Para este caso de uso en particular se decidió obtener información tanto de fuentes abiertas de Internet (conjuntos de datos de registros preexistentes) como de sistemas empleados por los TH y usados para realizar las pruebas de validación y verificación de la arquitectura de este trabajo, consiguiendo así un conjunto de datos etiquetados con los que entrenar y personalizar un BERT.

Un ejemplo de los datos empleados para entrenar el BERT serían los registros provenientes de un servidor web Apache cuya estructura tiene un aspecto como el representado en el código 4.3.

```
{
    "log27347": "193.106.31.130 - - [01/Sep/2019:03:28:00 +0200] \"POST /administrator/index.php HTTP/1.0\" 200 4481 \"-\"
       \"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)\" \"-\""
}
```

Código 4.3: Quinto caso de uso: Detección de amenazas a partir de registros con LLMs: Log de Apache

Se considera importante destacar que en la literatura se pueden encontrar muchas aproximaciones donde se realizan agrupaciones de los registros para categorizarlos en normales y anómalos, habiendo desde las más tradicionales basadas en el uso de palabras clave y/o expresiones regulares hasta las más recientes donde se propone el empleo de técnicas de ML mediante el uso de clasificadores binarios o algoritmos no supervisados como PCA.

A pesar de los esfuerzos realizados hasta la fecha, se considera que el uso de LLM aporta muchas ventajas con respecto a las aproximaciones previas (incluso de ML) por la gran capacidad de las arquitecturas "Transformer" en el modelado de datos secuenciales y, por tanto, la predicción acerca de cuál es el siguiente token más verosímil en una secuencia, de esta forma, el patrón de secuencias de registros normales es aprendido por el modelo en la fase de entrenamiento, detectando con una mayor precisión (y sobretodo, descubriendo patrones ocultos en los datos) cuando un registro se corresponde a una situación anómala y cuando no. Por tanto, lo que se entrenará en este caso son secuencias de registros normales y secuencias de registros anómalas para, haciendo uso de la extraordinaria capacidad de los LLM en el modelado de secuencias, poder categorizar una cierta secuencia en una categoría u otra. Finalmente, cabe destacar que la aproximación seguida es similar a LogBERT [94].

Tras realizar el caso de uso descrito con anterioridad, al realizar una tarea de inferencia sobre un conjunto de datos de entrada se obtuvieron las detecciones representadas en el código 4.4.


```
{
    "log_event_sequence": [
        "log27347",
        "log27452",
        "log27455",
        "log27458",
        "log27468",
        "log27593"
    ],
    "result": "anomaly: brute force attack",
    "likeliness": ".678"
}
```


Código 4.4: Quinto caso de uso: Detección de amenazas a partir de registros con LLMs: Resultado

4.3.6 Sexto caso de uso: Extensión de la capacidad de los LLMs para el análisis de registros. Uso de la ruptura de la simetría

Un factor que tradicionalmente ha sido muy limitante en los procesos de ML es el "viciado" de los modelos a conjuntos específicos de datos, o más concretamente, el impacto que los mínimos locales pueden tener en la búsqueda de un mínimo global. A pesar de existir soluciones como la técnica *gradient descent* para "perceptron" y para "perceptron multicapa", en este caso de uso se propone beneficiarse de las características específicas de los LLM, más precisamente del hecho de que los LLM son un tipo de técnicas de DL que devuelven resultados no deterministas, es decir, para una misma entrada sobre un modelo que ha sido entrenado y *fine-tuned* empleando unos mismos conjuntos de datos específicos (para este caso de uso, datos relacionados con la caza de amenazas), la salida obtenida no siempre es la misma (cabe destacar que dichas salidas si que serán similares entre ellas, aunque no exactamente iguales).

Además, se considera relevante destacar que factores como la entrada del usuario, el contexto que se le pasa al LLM, el uso de técnicas de RAG y la interacción con otros modelos (algo cada vez más usual), entre otros, pueden contribuir de manera drástica a amplificar dicha intrínseca naturaleza estadística de un LLM y conducir a los modelos a entregar resultados diferentes dadas las mismas entradas.

De hecho, como se señala en [24] y en [239], los sistemas LLM pueden ser formalmente descritos, en la perspectiva de la teoría de control de procesos, como una clase de sistemas dinámicos, discretos y estocásticos. Además, en el primero de los trabajos citados también se puede encontrar un modelado matemático de la capacidad de control de los LLM y su característica fundamental del mecanismo de auto-atención, donde se señala formalmente el inherente no determinismo de los mismos.

Esta característica de los LLM puede verse como un elemento negativo o problemático, sin embargo, y es el punto clave de toda la línea de trabajo descrita en este punto, ésta se puede ver como un elemento positivo para el caso de uso adecuado. Uno de esos casos de uso donde se podría ver esta característica como un elemento positivo sería al considerar los LLM actuando como máquinas de Turing [48] complejas y no deterministas, con lo que se pueden enriquecer mucho sus salidas al proveer de respuestas de inferencia no lineales y no previstas, consiguiendo así enriquecer la calidad de los resultados asumiendo un coste tolerable de no determinismo.

Esta ruptura de simetría, aplicada a la AI, puede llegar a ser una característica muy positiva si se controla adecuadamente, puesto que habilita la capacidad de permitir a este tipo de procesos la inferencia desde diferentes perspectivas para el mismo conjunto de datos de entrada. De esta forma, los bien conocidos problemas de ML (como los anteriormente señalados de tener un sistema que se queda encajado en mínimos locales, pese al uso de aproximaciones que intentan solventarlos como las previamente destacadas) pueden ser parcialmente solventados.

Uno de los ámbitos donde se podría explotar dicho no determinismo es el de la caza de amenazas al que se le intenta aportar soluciones en el transcurso de este trabajo. Cuando un analista se encuentra realizando sus tareas se puede ver confrontado con la detección de un problema de ciberseguridad al que debe dar una respuesta en muy poco tiempo y del que tiene pocos datos, un conocimiento difuso y cuya solución se obtiene tras recorrer ingentes cantidades de registros para así poder intentar hacerse una idea de lo que puede estar sucediendo. Como se ha visto previamente, las técnicas de ML pueden ser muy útiles a la hora de ayudar a este ciberespecialista a la hora de encontrar patrones y sugerencias que le conduzcan a obtener una conciencia situacional adecuada, sin embargo, en este proceso las técnicas de ML pueden introducir sesgo o dar siempre los mismos resultados para una misma situación. Idealmente, el TH busca que el sistema le sugiera diferentes perspectivas acerca de lo que puede estar sucediendo en la infraestructura monitorizada y sobre esas sugerencias ponderar y elegir que puede estar sucediendo en dicha infraestructura. En consecuencia, un LLM con el grado de no determinismo bien ajustado puede ser muy útil para aportar un conjunto de propuestas distintas (aunque similares entre ellas) sobre unos mismos datos de entrada, permitiendole así al TH ampliar el abanico de posibilidades acerca de lo que está sucediendo y evitar que se quede atascado en una de las ramas del árbol de posibilidades por las limitaciones del procedimiento, siendo una característica muy necesaria y apreciada por los mismos acorde a nuestra experiencia trabajando con TH.

Otra ventaja de esta aproximación no determinista es la aplicación de la misma a las capacidades de razonamiento de los LLM ya que, cuando se entrenan con grandes cantidades de información de ciberseguridad (por ejemplo registros) pueden ampliar de forma muy considerable el espectro de preguntas que se le pueden hacer al afrontar un ciberincidente, entregando perspectivas y sugerencias no previstas por el analista de ciberseguridad y, muy importante, que no pueden ser provistas por procesos de inferencia complemente deterministas.

Por tanto, además de desarrollar una arquitectura de caza de amenazas basada en AI como se ha visto en puntos previos, se propone extender la arquitectura con la perspectiva de que los LLM pueden ser muy útiles a la hora de proveer pistas y sugerencias, no sólo por su bien conocida capacidad de detección de patrones ocultos, sino también por su capacidad para generar respuestas y razonamientos no deterministas gracias a su inherente arquitectura estadística y a esta "ruptura de simetría" que proveen en la topología de sus máquinas de estado en términos de entradas y salidas, siendo este comportamiento un producto de la naturaleza de las redes neuronales complejas en las que se fundamentan, de sus mecanismos de entrenamiento estocásticos y del vasto conjunto de datos que se usan para entrenar a estos modelos.

Hay que destacar que si bien esta característica puede conducir a la diversidad de respuestas y creatividad en la exploración del espacio de soluciones, también puede provocar inconsistencias, aspecto que se deberá tener muy bien controlado y acotado. En cualquier caso, como se ha comentado previamente, en el coste beneficio dentro del proceso de la caza de amenazas, el TH prefiere que se le den opciones innovadoras a un problema en el que está atascado antes que una consistencia completa, por lo que perder un pequeño porcentaje de la misma es muy aceptable si se consideran los beneficios obtenidos, ya que muchas veces, esa chispa de creatividad que conduce a desatascar una caza de una amenaza es lo que más valor tiene en el conjunto global del proceso.

Un factor importante en todo este punto es que el grado de determinismo del modelo puede ser controlado mediante el ajuste de ciertos parámetros del mismo. Todos los LLM tienen una descripción de sus características en la carta de modelo donde se detallan los parámetros ajustables del mismo, tanto de entrenamiento como de inferencia. Los parámetros de entrenamiento se refieren a los conocidos como hiperparámetros y permiten personalizar en el proceso de *fine-tuning* el entrenamiento del modelo, mientras que los parámetros de inferencia permiten ajustar las características de las respuestas del modelo en ejecución. En consecuencia, el grado de determinismo se puede configurar con algunos de estos parámetros para cada una de las ejecuciones de inferencia se lancen. Cabe destacar que casi todo los modelos emplean para tal efecto los parámetros (i) *temperature*, (ii) *max output tokens*, (iii) *topK*, (iv) *topP* y (v) *stop sequences*.

Se considera relevante remarcar el parámetro *temperature* para controlar el grado de determinismo de los resultados que entrega un determinado LLM ya que éste influye en el grado de aleatoriedad de la selección de tokens. Típicamente, temperaturas bajas son buenas para entradas que requieren respuestas más deterministas (o menos abiertas en la exploración del árbol de soluciones), mientras que temperaturas altas se prefieren para respuestas menos deterministas (ya que conducen a una mayor diversidad y creatividad en resultados). Los valores de este parámetro suelen ir de cero (0) a uno (1). Cabe destacar que incluso cuando la temperatura está fijada a cero no existe una garantía estricta de obtener salidas completamente deterministas, puesto que en algunos casos los dos tokens con mayor puntuación en la etapa de decodificación pueden estar tan próximos entre si que incluso una discrepancia muy pequeña en valores de coma flotante (inherentes a los sistemas de cómputo) puede hacer que el LLM escoja el segundo token más probable.

Finalmente, como ampliación de la aproximación descrita en este caso de uso se propone el empleo de arquitecturas MoE (del inglés, *Mixture of Experts*) donde se emplean LLM que en su diseño incluyen a su vez varios LLM y hacen uso de algoritmos de votación entre los resultados obtenidos de cada uno de los LLM para proporcionar una respuesta ponderada las salidas. Cabe destacar que esta respuesta ponderada obtenida a la salida agrega de forma no lineal el ruido de cada uno de los LLM incluidos, pudiendo conseguir así una mayor variedad en cuanto a las respuestas generadas por dichos modelos se refiere [68, 108, 202].

4.4 Verificación en entorno no controlado: PRAETORIAN

PRAETORIAN (*Protection of Critical Infrastructures from Advanced Combined Cyber and Physical Threats*) es un proyecto de la llamada SU-INFRA dentro de la convocatoria H2020 de proyectos de I+D financiados por la REA de la Comisión Europea. El objetivo del proyecto era el de desarrollar un sistema que permitiera la protección, tanto en el dominio cibernético como en el dominio físico, de las IC ante posibles ataques o problemas que impidieran la continuidad de operación de estos elementos fundamentales para el funcionamiento de un país, bien fueran hospitales, puertos, aeropuertos, centrales hidroeléctricas o empresas de distribución de energía, existiendo usuarios finales para todos estos dominios de aplicación.

En dicho proyecto, recientemente finalizado (noviembre 2023), además de la UPV, han participado empresas líderes en el sector tecnológico como THALES Group, Electricité de France, IDEMIA Germany y ETRA I+D, junto con institutos de investigación como Deutsches Zentrum für Luft und Raumfahrt (DLR), ICCS y Austrian Institute of Technology (AIT). Además, un nutrido grupo de PYME de toda Europa han sido socios del proyecto como Rinigard Zagreb y SubCmarine Francia. Un elemento muy destacado del conjunto de socios del proyecto es la amplia y muy variada representación de destacados usuarios finales de diversos países y sectores de las IC y, de entre estas, cabe destacar a la entidad que gestiona los aeropuertos de España (AENA), a la empresa responsable de la explotación y mantenimiento de las centrales hidroeléctricas Croatas (KONCAR), al puerto de Valencia, al puerto de Burdeos, al aeropuerto de Zagreb, a la hidroeléctrica nacional Croata (HEP), a bomberos y servicios de rescate de España, Croacia y Francia y, en la parte sanitaria, al hospital la Fé de Valencia y al consorcio de hospitales austríacos KABEG.

Además, un elemento muy innovador del proyecto fue el de ir un paso más allá de la protección del dominio cibernético y físico e investigar y proveer soluciones al respecto acerca de como proteger el dominio híbrido resultante de la interacción de la parte cibernética y la física. Así, el sistema permitía responder a preguntas del tipo ¿qué pasaría a los activos físicos de esta área del puerto si se produce un ciberataque en dicha IC? Esto es, el objetivo del proyecto, el cuál fue ampliamente alcanzado, era el de permitir el análisis de consecuencias de un evento adverso (o un ataque) en el dominio físico, en el dominio cibernético, o la propagación no lineal entre dominios así como los posibles efectos en cascada que se pudieran producir entre estos.

Para poder implementar las capacidades propuestas, la plataforma desarrollada en el proyecto tenía la arquitectura representada en la figura 4.10.

Como se puede observar en el diagrama de la figura 4.10, los sistemas fundamentales dentro de la plataforma son la PSA (del inglés, *Physical Situational Awareness*), CSA (del inglés, *Cyber Situational Awareness*), HSA (del inglés, *Hybrid Situational Awareness*) y CR (del inglés, *Coordinated Response*).

La PSA se integraba con los sistemas físicos de la IC, principalmente sensores (como por ejemplo, vídeo, temperatura, humos, controles de acceso y parámetros en las turbinas de la central hidroeléctrica), así como con los sensores de los drones y otros equipos provistos por socios del proyecto como RINIGARD para fusionar toda esa información y entregarle al oficial al mando una consciencia situacional de la parte física.

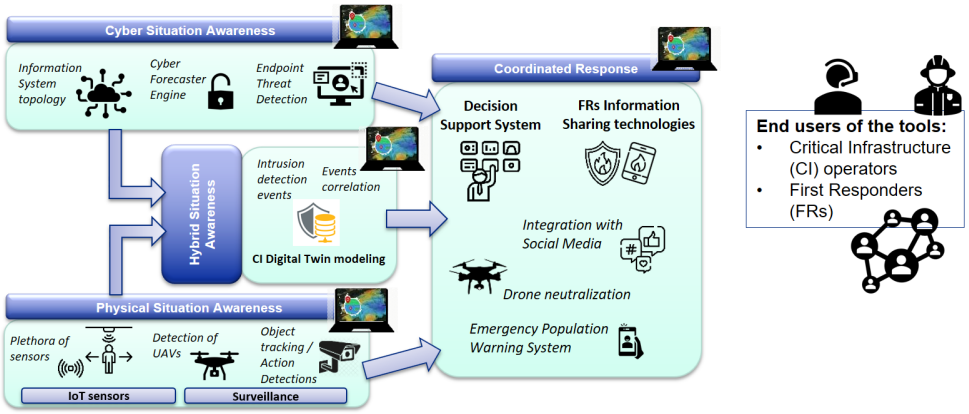

Figura 4.10: Verificación en entorno no controlado: PRAETORIAN: Arquitectura

La CSA se integraba con todos los sistemas cibernéticos de la IC, como por ejemplo, SIEM, sistemas de detección de intrusiones (IDS, del inglés *Intrusion Detection Systems*), sistemas de prevención de intrusiones (IPS, del inglés *Intrusion Prevention Systems*), sondas y fuentes de registros presentes en los sistemas cibernéticos de las IC. Tras recibir toda la información de dichos sistemas, se encargaba procesarla para detectar y anticiparse a posibles amenazas ayudando a los TH con las herramientas de ML analizadas en la descripción de la arquitectura propuesta en la presente tesis y, finalmente, mostrar toda la información generada en en un HMI personalizado.

La HSA recibía información de la PSA y la CSA para poder alimentar a un gemelo digital híbrido y, mediante avanzadas técnicas de modelado y simulación, poder generar análisis de consecuencias y efectos en cascada en el dominio híbrido. Es de destacar, como contribución del proyecto, que no sólo se circunscribía una única IC si no que respondía a preguntas también del tipo, ¿qué efectos podría tener un ataque en el dominio físico en el puerto de Valencia sobre el dominio cibernético de la IC Hospital la Fé?

El CR se encargaba de facilitar tanto herramientas para la toma de decisiones como también herramientas para actuar de interfaz de toda la plataforma PRAETORIAN con los sistemas de las autoridades y protección civil, así como de informar a la población ante un evento o amenaza por diversos canales.

La UPV participó activamente tanto en la parte de ciberseguridad dentro de la herramienta de CSA como en la plataforma, permitiendo así realizar *in situ* pruebas de verificación de la arquitectura de caza de amenazas objeto de esta tesis en los pilotos de las IC de Croacia, Francia y España, tal y como se describirá a continuación.

Dentro de la CSA, la arquitectura de caza de amenazas fue desplegada en las pruebas de laboratorio y, a continuación, usada e integrada con las fuentes de datos de los usuarios finales en las pruebas realizadas en los pilotos y demostraciones.

Además, para la parte del HMI, se realizaron personalizaciones y adaptaciones del mismo para las necesidades específicas del proyecto. En particular, en la figura 4.11 se puede observar una captura de la visión general del HMI resultante. En dicha figura se pueden diferenciar distintas técnicas avanzadas de visualización como "Hebbian dynamics" o "Hilbert maps", así como visualizaciones 3D de la información. Un elemento relevante del HMI era que reflejaba la diferencia entre los activos de soporte (como por ejemplo, un servidor de base datos) y los activos primarios (como por ejemplo, el sistema de control de las turbinas) y como las degradaciones o impactos que sufrían los primeros afectaban a los segundos, que era los más importante.

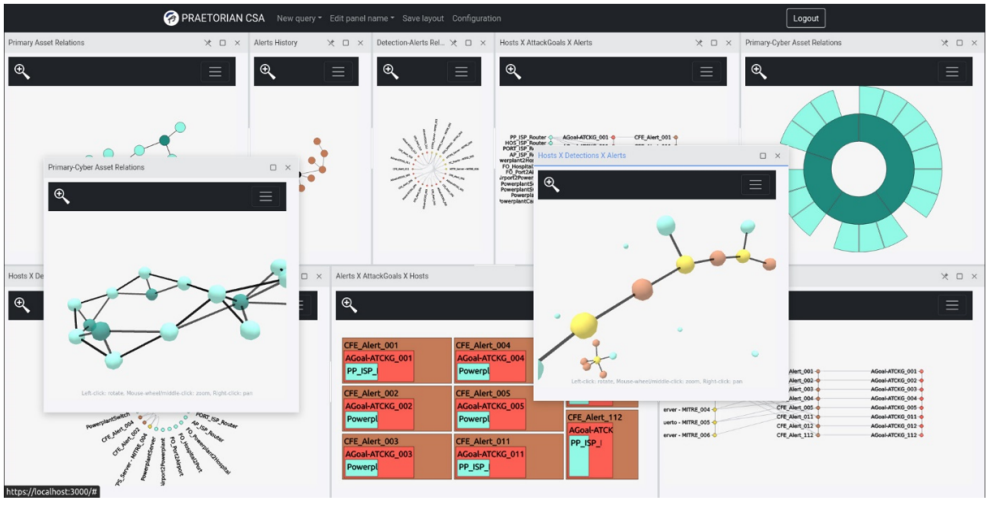

Figura 4.11: Verificación en entorno no controlado: PRAETORIAN: HMI: Visión General

Por otra parte, como se observa en la figura 4.12, el HMI también reflejaba toda la información de la plataforma de caza de amenazas. En ésta, se muestra un ejemplo en el que cuando los distintos microservicios de ML detectaban que se había producido una alerta, informaban al operador de la misma, mapeándola a MITRE Attack y, como ocurre en todo los sistemas de ayuda a la decisión, solicitando la validación de la alerta detectada al operador o analista humano.

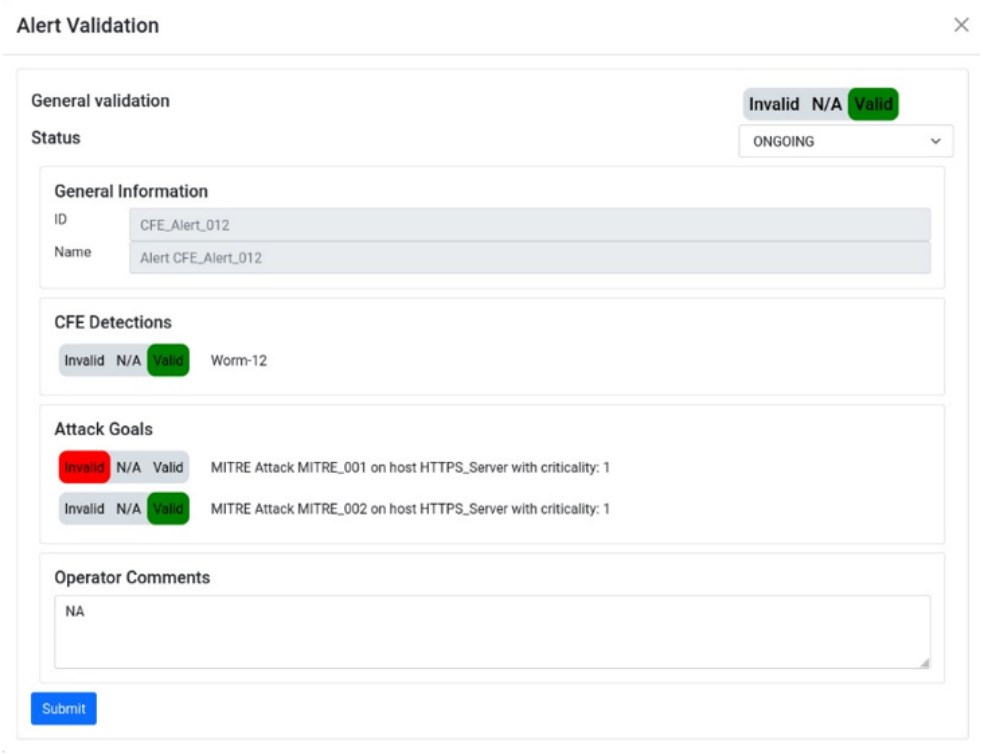

Figura 4.12: Verificación en entorno no controlado: PRAETORIAN: HMI: Validación de las Alertas

Una vez desarrollados los distintos sistemas del proyecto, se procedió a la validación en laboratorio del mismo. Para ello, se llevaron a cabo diversas pruebas en remoto entre los distintos sistemas (PSA, CSA, HSA y CR) y también conectados a gemelos digitales *ad-hoc* utilizados como plataformas de pruebas.

Superada esta fase, se procedió a llevar a cabo los distintos pilotos. El primero se realizó en marzo de 2023 en el aeropuerto de Zagreb, en el que se conectaron los distintos sistemas físicos y cibernéticos de dicha IC a la plataforma PRAETORIAN. Además, uno de los casos de uso que se probó en dicho piloto fue el de la interconexión de los sistemas físicos y cibernéticos de los hospitales austríacos del consorcio KABEG, aportando información relevante acerca de como podían afectar los problemas de una IC en las otras, teniendo en cuenta los tres dominios, físico, cibernético e híbrido.

El siguiente piloto se llevó a cabo en mayo de 2023 en el puerto de Burdeos, Francia, del que se puede observar una imagen de dicho piloto en la figura 4.13. Aquí, el sistema PRAETORIAN se conectó a los sistemas físicos y cibernéticos de dicho puerto así como a los de EDF (el principal proveedor de energía eléctrica del país). De especial relevancia para las pruebas de la arquitectura de esta tesis fue la conexión de la CSA a los sistemas cibernéticos de EDF, que fueron sometidos a una serie de ciberataques y se pudo ver como éstos no se detectaban claramente con las herramientas convencionales aunque sí que lo hacían con los microservicios de ML de esta arquitectura, siendo de gran relevancia ya que este ciberataque acababa afectando a las compuertas que regulan los accesos al rio Garona, donde se encuentra el puerto de Burdeos.

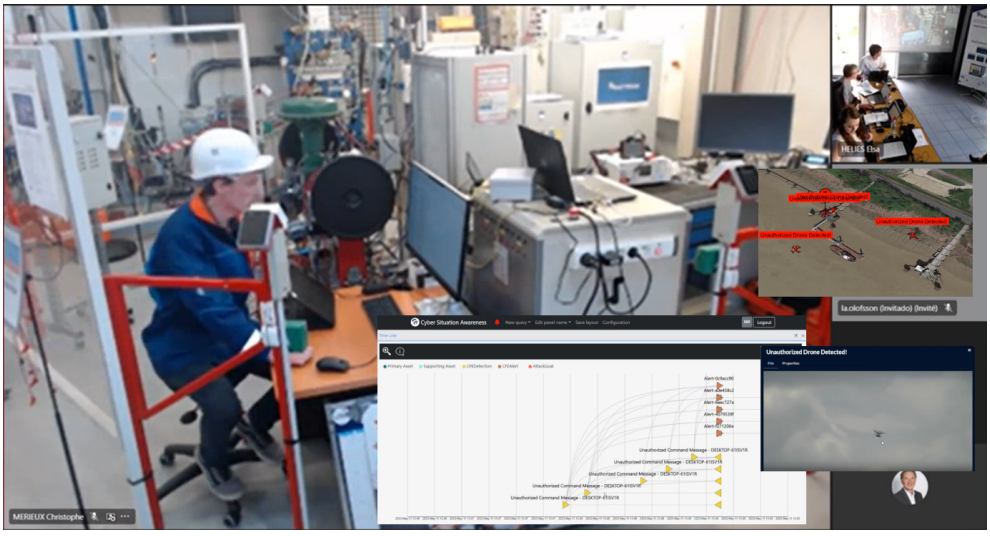

Figura 4.13: Verificación en entorno no controlado: PRAETORIAN: Piloto Burdeos

El tercer piloto se desarrolló en la localidad de Sinj, Croacia. Próxima a ella está la presa de Perucka, donde se volvió a probar satisfactoriamente la utilidad del sistem PRAETORIAN ante ataques combinados en el dominio físico y en el dominio cibernético. En la figura 4.14 se puede observar una imagen de dicho piloto.

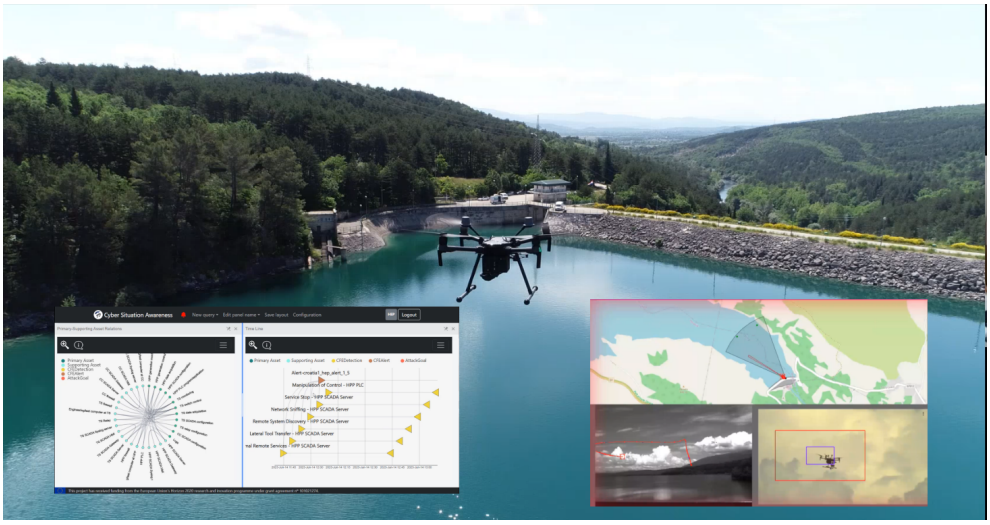

Figura 4.14: Verificación en entorno no controlado: PRAETORIAN: Piloto Croacia

El último piloto, y el más completo de ellos, tuvo lugar en el mes de julio de 2023 en Valencia, en concreto, en el puerto de Valencia. En el mismo, se probó tanto la capacidad del sistema desarrollado en la protección del propio puerto como un ataque combinado sobre varias IC, las cuáles fueron el aeropuerto de Valencia y el hospital la Fé. En este piloto, el sistema desarrollado en esta tesis sirvió para anticipar ciberataques en las tres infraestructuras y para propagar información al dominio híbrido donde se pudieran evaluar las consecuencias de dichos ciberataques en los otros dominios.

A continuación se podrá visualizar algunas imágenes correspondientes a dicho último piloto realizado en el proyecto PRAETORIAN dónde se verificaron dichas capacidades del sistema.

En primer lugar, en la figura 4.15 se puede observar la intervención de los bomberos ante una explosión así como el triaje de los heridos.

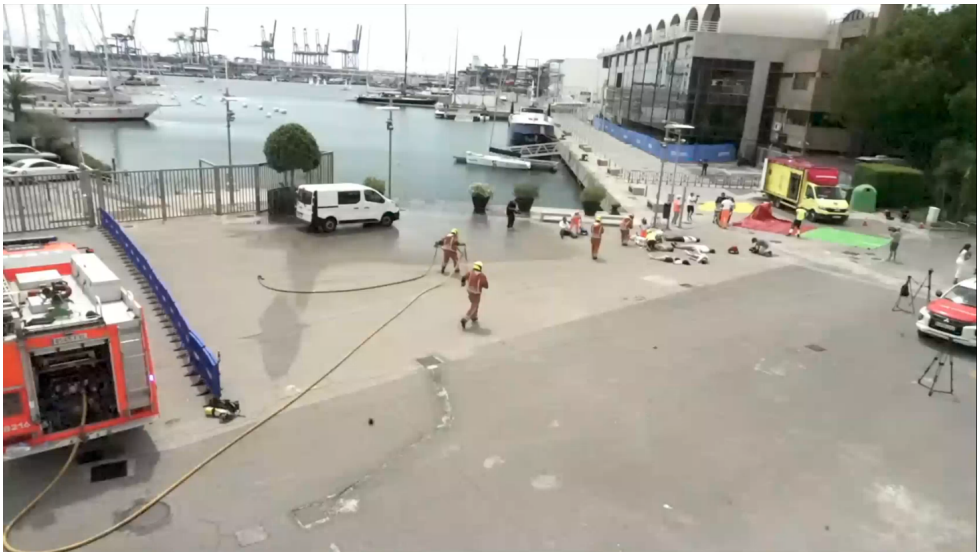

Figura 4.15: Verificación en entorno no controlado: PRAETORIAN: Piloto Valencia (I)

A continuación, en la figura 4.16 se puede observar la PSA así como la detección automática de eventos de violencia.

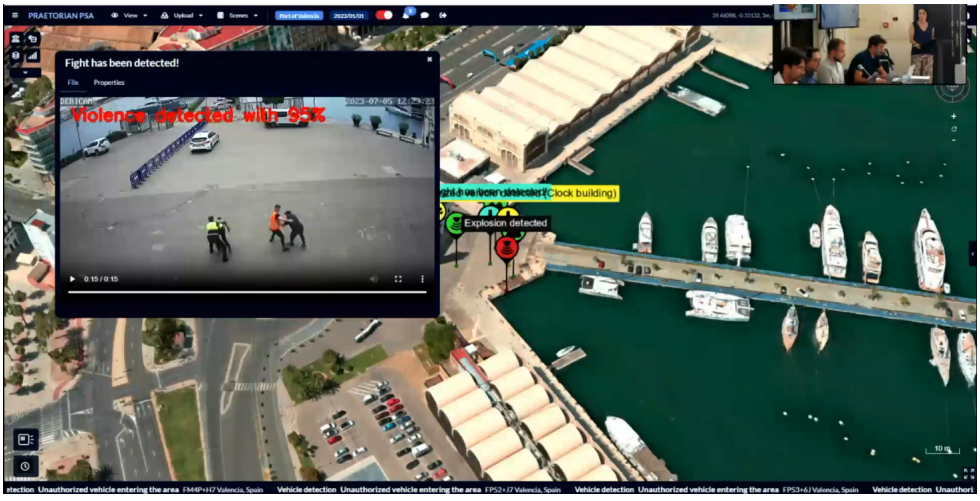

Figura 4.16: Verificación en entorno no controlado: PRAETORIAN: Piloto Valencia (II)

Finalmente, en la figura 4.17 se pueden observar las capacidades cibernéticas de la CSA.

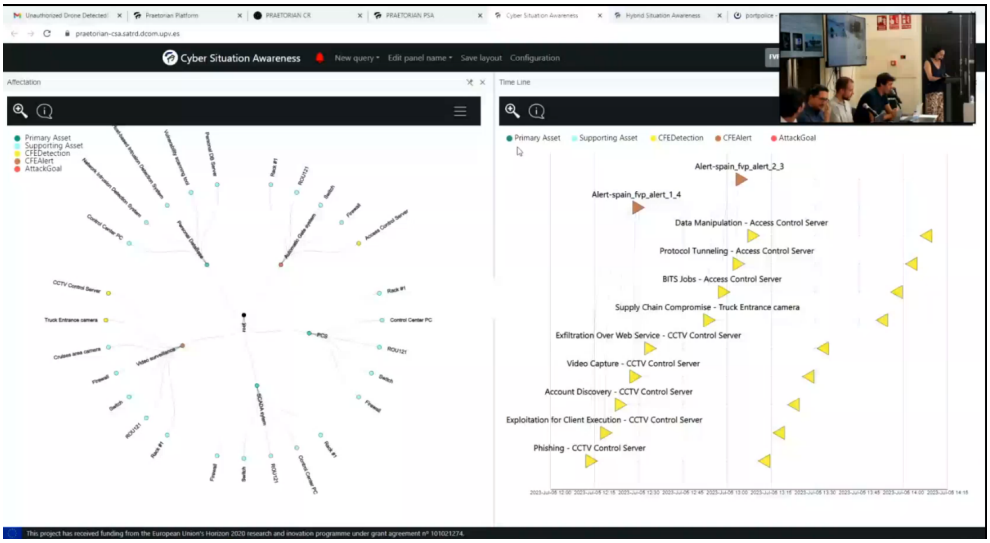

Figura 4.17: Verificación en entorno no controlado: PRAETORIAN: Piloto Valencia (III)

Capítulo 5

Conclusiones y trabajos futuros

En este capítulo se analizará el trabajo realizado así como las posibles lineas de investigación originadas a partir de los resultados obtenidos.

5.1 Conclusiones

En este trabajo se ha realizado un estudio acerca del trabajo desempeñado por los TH en su dia a dia donde se han detectado las necesidades asociadas al mismo y se ha descubierto que estos especialistas todavía necesitan de más líneas de investigación que se centren en la generación de herramientas para permitir que puedan desarrollar su trabajo de manera más eficiente. Acto seguido, se ha propuesto una arquitectura capaz de ayudar a éstos empleando técnicas de ML. A continuación, se ha desarrollado un prototipo que ha implementado la arquitectura propuesta en su totalidad. Para proseguir, se ha validado la funcionalidad de los componentes de la arquitectura. Finalmente, se ha verificado la arquitectura propuesta en entornos controlados y no controlados en los que se han realizado casos de uso reales donde emplear la arquitectura. Cabe destacar que también se ha propuesto una aplicación novedosa acerca de como emplear los LLM para la generación de sugerencias basadas en un contexto.

El objetivo global anteriormente expuesto se ha conseguido habiendo completado de forma satisfactoria los distintos objetivos esperados de este trabajo, que han sido:

- Se ha realizado un análisis exhaustivo acerca de las necesidades reales de los TH en su trabajo diario donde se han detectado cuáles son los principales requerimientos reportados por estos especialistas así como la posible aplicación de técnicas de ML para dar solución a los mismos por su casuística.
- Se ha analizado en profundidad el estado del arte actual con respecto a las necesidades de este trabajo. En primer lugar, se ha investigado acerca de las distintas soluciones empleadas por los TH para realizar sus trabajos diarios. A continuación, se ha estudiado cuales son los recursos actuales para realizar una arquitectura de aplicación. Finalmente, se han estudiado los servicios actuales a emplear por parte de una aplicación que implementase cualquier arquitectura.
- Se ha propuesto una arquitectura de aplicación para realizar la caza de amenazas, la cuál contempla todas las necesidades de los TH. Esta arquitectura propuesta tiene en cuenta aspectos tan importantes como la seguridad, escalabilidad, alta disponibilidad y eficiencia. Además, es una arquitectura preparada para trabajar con técnicas de ML, con lo que permite la adición, modificación y borrado de componentes de forma fácil y sencilla. No solo eso, también está preparada para trabajar con *big data*, teniendo módulos específicos que se encargan de realizar dicho tipo de procesados y visualizaciones de forma eficiente.
- Se ha diseñado y desarrollado un prototipo que ha implementado la arquitectura propuesta. Éste ha implementado todos los componentes definidos en la arquitectura. Además, en su diseño y desarrollo se han contemplado todos los aspectos de mayor relevancia con respecto a los datos con los que va a trabajar así como sus usuarios finales para que contemplase todos los casos de uso posibles, fuese eficiente y fuese tanto fácil de usar como intuitivo para los usuarios finales.
- Se han validado los componentes de la arquitectura haciendo uso del prototipo anteriormente desarrollado asegurando así que cumplen con las funciones esperadas en la fase de diseño de la arquitectura propuesta.
- Se ha verificado la arquitectura para certificar que cumple con las necesidades de los TH en entornos controlados y no controlados. Estas pruebas de verificación se han realizado empleando como base casos reales de uso y de aplicación de la arquitectura propuesta.

- Se ha propuesto una aplicación novedosa acerca de como emplear los LLM para la generación de sugerencias basadas en contextos.

Cabe destacar que este trabajo ha mejorado el campo de la caza de amenazas al realizar las siguientes aportaciones:

- Una arquitectura que contempla todas las necesidades de un TH, empezando por la recolección de los datos, prosiguiendo con el almacenamiento y el procesado de los mismos y finalizando en la visualización de la información generada para todos los posibles usuarios de la aplicación (desde el más táctico al más estratégico) sin obviar aspectos tan importantes como la eficiencia a la hora de interactuar con los datos o la compartición de los mismos con aplicaciones de terceros.
- La propuesta de un componente específico centrado en la generación de hipótesis que contempla tanto configuraciones sencillas como complejas siendo suficientemente flexible para poder ser empleado en todos los posibles casos de uso que necesitan los TH.
- El análisis de las técnicas de ML y DL que pueden ser empleadas para realizar la caza de amenazas con una aproximación basada en AI.
- Una aplicación novedosa acerca de como emplear los LLM para la generación de sugerencias basadas en contextos.

5.2 Trabajos futuros

Aunque el desarrollo realizado ha sido muy completo y satisfactorio, siempre existen puntos de mejora detectados durante el desarrollo del mismo. Para este trabajo en particular se podrían destacar los siguientes puntos que pueden tener una potencial mejora:

- Uso de SurrealDB en vez de Elastic Search en el prototipo. Para el desarrollo del prototipo se ha empleado Elastic Search tanto por las razones especificadas en el apartado correspondiente como porque es una base de datos consolidada y empleada por gran cantidad de soluciones similares, lo que aporta la seguridad de estar muy bien probada y transmite la confianza de que uso no originará problemas. No obstante, tal y como se ha analizado en el estado del arte, recientemente ha surgido SurrealDB, una nueva base de datos que ofrece características similares a Elastic Search, sino mejores. En consecuencia, uno de los trabajos futuros a realizar es la sustitución de Elastic Search por SurrealDB.

- Dentro de las técnicas de AI empleadas, como se ha podido verificar en el desarrollo del trabajo, existe una gran cantidad de opciones a la hora de escoger la mejor de éstas para una aplicación en particular. Para el desarrollo de este trabajo se han empleado las técnicas que se ha considerado como mejores para los distintos casos de uso, no obstante, cabe la posibilidad de que, tanto en las técnicas existentes en su estado actual como en la evolución de dichas técnicas o como en la creación de nuevas técnicas, existan mejores alternativas para los distintos casos de uso de la caza de amenazas mediante técnicas de AI, por ese motivo, como trabajo futuro se encuentra el análisis constante de éstas para buscar alternativas que puedan ofrecer mejores resultados para los problemas detectados.

- Los repositorios OSS de dónde se han obtenido gran cantidad de LLM se alimentan de investigaciones realizadas tanto por grandes organizaciones como por entusiastas del ML que suben sus aportaciones para que el resto de miembros de la comunidad puedan hacer uso de ellas. Un posible trabajo futuro podría ser la compartición de las distintas redes y LLM entrenados con el resto de miembros de la comunidad para que puedan hacer uso del esfuerzo realizado en este trabajo. Uno de estos portales donde se propone subir este contenido es "Hugging Face".

- Actualmente, los distintos componentes que conforman el conjunto de los datos de la arquitectura deben establecer dos conexiones con servicios del conjunto de los recursos, una para el agente de comunicaciones y otra para el servicio de gestión de la autenticación y la autorización. Un posible trabajo futuro podría consistir en la transmisión de los mensajes de autenticación y autorización empleando los protocolos OpenID Connect y OAuth 2.0 pero mediante AMQP en vez de HTTP. Con ésto se conseguiría que los componentes únicamente tuviesen que generar una única conexión contra un único servicio del conjunto de los recursos, el agente de comunicaciones, y el resto de las comunicaciones se gestionarían a partir de éste. Para poder conseguirlo, habría que desarrollar un componente específico que implementase los protocolos anteriormente enunciados pero que emplease AMQP para el envío y la recepción de los mensajes.

Para poder desarrollar estos trabajos futuros, actualmente el grupo de investigación se encuentra en el proceso de dar continuidad a la linea de investigación en la que se incluye este trabajo mediante la esperada futura adjudicación de nuevos proyectos donde, a parte de desarrollar e implementar los trabajos futuros anteriormente expuestos, se profundicen y amplíen los resultados obtenidos al emplear la arquitectura propuesta en nuevos ámbitos de aplicación.

Bibliografía

[1] Manar Abourezq y Abdellah Idrissi. "Database-as-a-service for big data: An overview". En: *International Journal of Advanced Computer Science and Applications* 7.1 (2016) (vid. pág. 51).

[2] Subil Abraham y Suku Nair. "Comparative analysis and patch optimization using the cyber security analytics framework". En: *The Journal of Defense Modeling and Simulation* 15.2 (2018), págs. 161-180 (vid. pág. 9).

[3] Babatunde Adetoye y Rose Cheuk-wai Fong. "Building a resilient cybersecurity workforce: a multidisciplinary solution to the problem of high turnover of cybersecurity analysts". En: *Cybersecurity in the Age of Smart Societies: Proceedings of the 14th International Conference on Global Security, Safety and Sustainability, London, September 2022*. Springer. 2023, págs. 61-87 (vid. pág. 2).

[4] Vibhor Agarwal y col. ""Which LLM should I use?": Evaluating LLMs for tasks performed by Undergraduate Computer Science Students in India". En: *arXiv preprint arXiv:2402.01687* (2024) (vid. pág. 35).

[5] Shohin Aheleroff y col. "IoT-enabled smart appliances under industry 4.0: A case study". En: *Advanced engineering informatics* 43 (2020), pág. 101043 (vid. pág. 8).

[6] Mohiuddin Ahmed, Raihan Seraj y Syed Mohammed Shamsul Islam. "The k-means algorithm: A comprehensive survey and performance evaluation". En: *Electronics* 9.8 (2020), pág. 1295 (vid. pág. 28).

[7] Berkin Akin y col. "Searching for efficient neural architectures for on-device ML on edge TPUs". En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, págs. 2667-2676 (vid. pág. 8).

[8] Hakam W Alomari y col. "A User Interface (UI) and User eXperience (UX) evaluation framework for cyberlearning environments in computer science and software engineering education". En: *Heliyon* 6.5 (2020) (vid. pág. 43).

[9] Maher Alsharif, Shailendra Mishra y Mohammed AlShehri. "Impact of Human Vulnerabilities on Cybersecurity." En: *Computer Systems Science & Engineering* 40.3 (2022) (vid. pág. 2).

[10] Fernando Alves, Pedro M Ferreira y Alysson Bessani. "OSINT-based Data-driven Cybersecurity Discovery". En: *12th Eurosys Doctoral Conference*. 2012, págs. 1-5 (vid. págs. 9, 21).

[11] Kimia Ameri y col. "Cybert: Cybersecurity claim classification by fine-tuning the bert language model". En: *Journal of Cybersecurity and Privacy* 1.4 (2021), págs. 615-637 (vid. pág. 34).

[12] Marco Angelini y col. "MAD: A visual analytics solution for Multi-step cyber Attacks Detection". En: *Journal of Computer Languages* 52 (2019), págs. 10-24 (vid. pág. 9).

[13] *Anomali ThreatStream: Automated threat intelligence management at scale*. `https://www.anomali.com/products/threatstream`. Last accessed 03 March 2023 (vid. pág. 10).

[14] Patricia Arias Cabarcos y col. "Enabling saml for dynamic identity federation management". En: *Joint IFIP Wireless and Mobile Networking Conference*. Springer. 2009, págs. 173-184 (vid. pág. 58).

[15] Florian Auer y col. "From monolithic systems to Microservices: An assessment framework". En: *Information and Software Technology* 137 (2021), pág. 106600 (vid. pág. 15).

[16] Rui Azevedo, Ibéria Medeiros y Alysson Bessani. "PURE: Generating quality threat intelligence by clustering and correlating OSINT". En: *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE. 2019, págs. 483-490 (vid. págs. 8, 21).

[17] Sean Barnum. "Standardizing cyber threat intelligence information with the structured threat information expression (stix)". En: *Mitre Corporation* 11 (2012), págs. 1-22 (vid. pág. 39).

[18] Nicole Beaulieu, Sergiu M Dascalu y Emily Hand. "API-First Design: A Survey of the State of Academia and Industry". En: *ITNG 2022 19th International Conference on Information Technology-New Generations*. Springer. 2022, págs. 73-79 (vid. pág. 42).

[19] Justin M Beaver y col. "Visualization techniques for computer network defense". En: *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense X*. Vol. 8019. SPIE. 2011, págs. 18-26 (vid. pág. 9).

[20] Mike Belshe, Roberto Peon y Martin Thomson. *Hypertext Transfer Protocol Version 2 (HTTP/2)*. RFC 7540. Mayo de 2015. DOI: 10.17487/RFC7540 (vid. pág. 19).

[21] Melvin Bender y col. "Open-source mqtt evaluation". En: *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. 2021, págs. 1-4 (vid. pág. 54).

[22] Yoshua Bengio, Aaron Courville y Pascal Vincent. "Representation learning: A review and new perspectives". En: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), págs. 1798-1828 (vid. pág. 25).

[23] Dhruv Bhanderi y col. "Impact of Two-Factor Authentication on User Convenience and Security". En: *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE. 2023, págs. 617-622 (vid. pág. 54).

[24] Aman Bhargava y col. "What's the Magic Word? A Control Theory of LLM Prompting". En: *arXiv preprint arXiv:2310.04444* (2023) (vid. pág. 107).

[25] Charles Bihis. *Mastering OAuth 2.0*. Packt Publishing Ltd, 2015 (vid. pág. 56).

[26] Leyla Bilge y Tudor Dumitraş. "Before we knew it: an empirical study of zero-day attacks in the real world". En: *Proceedings of the 2012 ACM conference on Computer and communications security*. 2012, págs. 833-844 (vid. pág. 3).

[27] Trude H Bloebaum y Frank T Johnsen. "Evaluating publish/subscribe approaches for use in tactical broadband networks". En: *MILCOM 2015-2015 IEEE Military Communications Conference*. IEEE. 2015, págs. 605-610 (vid. pág. 16).

[28] Adam Boduch y Roy Derks. *React and React Native: A complete hands-on guide to modern web and mobile development with React. js*. Packt Publishing Ltd, 2020 (vid. págs. 44, 45).

[29] Barry W Boehm. "Value-based software engineering: Overview and agenda". En: *Value-based software engineering* (2006), págs. 3-14 (vid. pág. 15).

[30] Carsten Bormann, Angelo P Castellani y Zach Shelby. "Coap: An application protocol for billions of tiny internet nodes". En: *IEEE Internet Computing* 16.2 (2012), págs. 62-67 (vid. pág. 19).

[31] Hongyu Pei Breivold, Ivica Crnkovic y Magnus Larsson. "A systematic review of software architecture evolution research". En: *Information and Software Technology* 54.1 (2012), págs. 16-40 (vid. pág. 10).

[32] Siri Bromander y col. "Investigating sharing of cyber threat intelligence and proposing a new data model for enabling automation in knowledge representation and exchange". En: *Digital Threats: Research and Practice (DTRAP)* 3.1 (2021), págs. 1-22 (vid. pág. 23).

[33] Jason Brownlee. *Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python*. Machine Learning Mastery, 2020 (vid. pág. 24).

[34] Robert Bryce, Thomas Shaw y Gautam Srivastava. "Mqtt-g: A publish/-subscribe protocol with geolocation". En: *2018 41st international conference on telecommunications and signal processing (TSP)*. IEEE. 2018, págs. 1-4 (vid. pág. 20).

[35] Antonio Bucchiarone y col. "From monolithic to microservices: An experience report from the banking domain". En: *Ieee Software* 35.3 (2018), págs. 50-55 (vid. págs. 15, 52).

[36] Vasco Samuel Carvalho, Maria Joao Polidoro y Joao Paulo Magalhaes. "Owlsight: Platform for real-time detection and visualization of cyber threats". En: *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*. IEEE. 2016, págs. 61-66 (vid. pág. 9).

[37] Siming Chen y col. "Oceans: Online collaborative explorative analysis on network security". En: *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*. 2014, págs. 1-8 (vid. pág. 47).

[38] David R Cheriton y Carey L Williamson. "Network measurement of the VMTP request-response protocol in the V distributed system". En: *Proceedings of the 1987 ACM SIGMETRICS conference on Measurement and modeling of computer systems*. 1987, págs. 216-225 (vid. pág. 17).

[39] Sungyoung Cho y col. "Cyber kill chain based threat taxonomy and its application on cyber common operational picture". En: *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*. IEEE. 2018, págs. 1-8 (vid. pág. 9).

[40] Hyunsang Choi y Heejo Lee. "PCAV: Internet attack visualization on parallel coordinates". En: *Information and Communications Security: 7th International Conference, ICICS 2005, Beijing, China, December 10-13, 2005. Proceedings 7*. Springer. 2005, págs. 454-466 (vid. pág. 47).

[41] Binildas Christudas y Binildas Christudas. *MySQL*. Springer, 2019 (vid. pág. 49).

[42] Stanley Cohen. "The evolution of machine learning: Past, present, and future". En: *Artificial intelligence and deep learning in pathology*. Elsevier, 2021, págs. 1-12 (vid. pág. 27).

[43] Patrick Crowley. "Statement: Transmit/Receive and Request/Response Models for Communication". En: *Proceedings of the 10th ACM Conference on Information-Centric Networking*. 2023, págs. 109-111 (vid. pág. 17).

[44] Chris Currier. "Protocol buffers". En: *Mobile Forensics–The File Format Handbook: Common File Formats and File Systems Used in Mobile Devices*. Springer, 2022, págs. 223-260 (vid. pág. 42).

[45] Csongor Miklós Cziráki y col. "Changing your mindset: a desktop application supporting the changeover from ICD-10 to ICD-11". En: *Pannonian Conference on Advances in Information Technology (PCIT 2020)*. 2020, pág. 70 (vid. pág. 44).

[46] Luc Dandurand y Oscar Serrano Serrano. "Towards improved cyber security information sharing". En: *2013 5th International Conference on Cyber Conflict (CYCON 2013)*. IEEE. 2013, págs. 1-16 (vid. pág. 38).

[47] Mamata Das, Selvakumar Kamalanathan y PJA Alphonse. "A Comparative Study on TF-IDF Feature Weighting Method and Its Analysis Using Unstructured Dataset." En: *COLINS*. 2021, págs. 98-107 (vid. pág. 35).

[48] Liesbeth De Mol. "Turing machines". En: (2018) (vid. pág. 107).

[49] Barry De Ville. "Decision trees". En: *Wiley Interdisciplinary Reviews: Computational Statistics* 5.6 (2013), págs. 448-455 (vid. pág. 9).

[50] Yuri Demchenko, Cees De Laat y Peter Membrey. "Defining architecture components of the Big Data Ecosystem". En: *2014 International conference on collaboration technologies and systems (CTS)*. IEEE. 2014, págs. 104-112 (vid. pág. 22).

[51] Daniel DeMenthon y Remi Megret. *Spatio-temporal segmentation of video by hierarchical mean shift analysis*. Computer Vision Laboratory, Center for Automation Research, University of …, 2002 (vid. pág. 29).

[52] Konstantinos G Derpanis. "Mean shift clustering". En: *Lecture Notes* 32 (2005), págs. 1-4 (vid. pág. 29).

[53] Andrea Detti, Ludovico Funari y Nicola Blefari-Melazzi. "Sub-linear scalability of MQTT clusters in topic-based publish-subscribe applications". En: *IEEE Transactions on Network and Service Management* 17.3 (2020), págs. 1954-1968 (vid. pág. 54).

[54] DN Divyabharathi y Nagaraj G Cholli. "A review on identity and access management server (keycloak)". En: *International Journal of Security and Privacy in Pervasive Computing (IJSPPC)* 12.3 (2020), págs. 46-53 (vid. pág. 58).

[55] AnHai Doan, Alon Halevy y Zachary Ives. *Principles of data integration*. Elsevier, 2012 (vid. pág. 22).

[56] Philippe Dobbelaere y Kyumars Sheykh Esmaili. "Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry Paper". En: *Proceedings of the 11th ACM international conference on distributed and event-based systems*. 2017, págs. 227-238 (vid. pág. 53).

[57] Juraj Dončević y col. "Mask–Mediator–Wrapper: A Revised Mediator–Wrapper Architecture for Heterogeneous Data Source Integration". En: *Applied Sciences* 13.4 (2023), pág. 2471 (vid. pág. 22).

[58] Rob Donnelly y Josh Geden. "Modern Hardware/Software Interface (HSI) Documentation". En: *Proceedings of the International Research Journal of Modernization in Engineering Technology and Science*. 2023, págs. 1-10 (vid. pág. 44).

[59] Nicola Dragoni y col. "Microservices: How to make your application scale". En: *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*. Springer. 2017, págs. 95-104 (vid. págs. 15, 16).

[60] Shakuntala Gupta Edward y col. "Introducing MongoDB". En: *Practical MongoDB: Architecting, Developing, and Administering MongoDB* (2015), págs. 25-28 (vid. pág. 50).

[61] Jeffrey L Elman. "Finding structure in time". En: *Cognitive science* 14.2 (1990), págs. 179-211 (vid. pág. 35).

[62] Mica R Endsley. "Measurement of situation awareness in dynamic systems". En: *Human factors* 37.1 (1995), págs. 65-84 (vid. pág. 3).

[63] Mohammed Eslami y col. "Deriving cyber use cases from graph projections of cyber data represented as bipartite graphs". En: *2017 IEEE International Conference on Big Data (Big Data)*. IEEE. 2017, págs. 4658-4663 (vid. pág. 9).

[64] Patrick Th Eugster y col. "The many faces of publish/subscribe". En: *ACM computing surveys (CSUR)* 35.2 (2003), págs. 114-131 (vid. pág. 17).

[65] European Commission. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance)*. 2016 (vid. pág. 50).

[66] J Benjamin Falandays, Benjamin Nguyen y Michael J Spivey. "Is prediction nothing more than multi-scale pattern completion of the future?" En: *Brain Research* 1768 (2021), pág. 147578 (vid. pág. 2).

[67] Kara D Federmeier. "Thinking ahead: The role and roots of prediction in language comprehension". En: *Psychophysiology* 44.4 (2007), págs. 491-505 (vid. pág. 2).

[68] William Fedus, Barret Zoph y Noam Shazeer. "Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity.(2021)". En: *arXiv preprint cs.LG/2101.03961* (2021) (vid. pág. 110).

[69] Diogo Fernandes, Jorge Bernardino y col. "Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB." En: *Data* 10 (2018), pág. 0006910203730380 (vid. pág. 51).

[70] Eugene Ferry, John O Raw y Kevin Curran. "Security evaluation of the OAuth 2.0 framework". En: *Information & Computer Security* 23.1 (2015), págs. 73-101 (vid. pág. 58).

[71] Luca Forlizzi y col. "A data model and data structures for moving objects databases". En: *ACM SIGMOD Record* 29.2 (2000), págs. 319-330 (vid. pág. 23).

[72] Dishan Francis. *Mastering Active Directory: Design, Deploy, and Protect Active Directory Domain Services for Windows Server 2022*. Packt Publishing Ltd, 2021 (vid. pág. 58).

[73] Ulrik Franke y Joel Brynielsson. "Cyber situational awareness–a systematic review of the literature". En: *Computers & security* 46 (2014), págs. 18-31 (vid. pág. 3).

[74] Daniel F Garcia y col. "Experimental evaluation of horizontal and vertical scalability of cluster-based application servers for transactional workloads". En: *8th International Conference on Applied Informatics and Communications (AIC'08)*. Citeseer. 2008, págs. 29-34 (vid. págs. 12, 13, 16).

[75] Jason Garman. *Kerberos: The Definitive Guide: The Definitive Guide*. " O'Reilly Media, Inc.", 2003 (vid. pág. 56).

[76] Tiberiu-Marian Georgescu. "Natural language processing model for automatic analysis of cybersecurity-related documents". En: *Symmetry* 12.3 (2020), pág. 354 (vid. pág. 33).

[77] Steven Gianvecchio y col. "Closing the gap with APTs through semantic clusters and automated cybergames". En: *Security and Privacy in Communication Networks: 15th EAI International Conference, SecureComm 2019, Orlando, FL, USA, October 23-25, 2019, Proceedings, Part I 15*. Springer. 2019, págs. 235-254 (vid. pág. 23).

[78] Inmar Givoni, Clement Chung y Brendan J Frey. "Hierarchical affinity propagation". En: *arXiv preprint arXiv:1202.3722* (2012) (vid. pág. 28).

[79] Akarsh Goel y B Thangaraju. "Authenticating distributed systems using SPIRE over kubernetes cluster". En: *2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE. 2022, págs. 1-6 (vid. pág. 54).

[80] Santiago González-Carvajal y Eduardo C Garrido-Merchán. "Comparing BERT against traditional machine learning text classification". En: *arXiv preprint arXiv:2005.13012* (2020) (vid. pág. 34).

[81] Gustavo Gonzalez-Granadillo y col. "ETIP: An Enriched Threat Intelligence Platform for improving OSINT correlation, analysis, visualization and sharing capabilities". En: *Journal of Information Security and Applications* 58 (2021), pág. 102715 (vid. págs. 8, 21).

[82] John R Goodall y col. "Situ: Identifying and explaining suspicious behavior in networks". En: *IEEE transactions on visualization and computer graphics* 25.1 (2018), págs. 204-214 (vid. pág. 9).

[83] Ken Goodhope y col. "Building LinkedIn's Real-time Activity Data Pipeline." En: *IEEE Data Eng. Bull.* 35.2 (2012), págs. 33-45 (vid. pág. 53).

[84] Clinton Gormley y Zachary Tong. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine.* " O'Reilly Media, Inc.", 2015 (vid. pág. 51).

[85] David Gourley y Brian Totty. *HTTP: the definitive guide.* " O'Reilly Media, Inc.", 2002 (vid. pág. 18).

[86] Gaurav Goyal, Karanjit Singh y KR Ramkumar. "A detailed analysis of data consistency concepts in data exchange formats (JSON & XML)". En: *2017 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE. 2017, págs. 72-77 (vid. págs. 39, 42, 43, 46).

[87] Roman Graf y col. "An Expert System for Facilitating an Institutional Risk Profile Definition for Cyber Situational Awareness." En: *ICISSP*. 2016, págs. 347-354 (vid. pág. 9).

[88] Jim Gray y Daniel P. Siewiorek. "High-availability computer systems". En: *Computer* 24.9 (1991), págs. 39-48 (vid. págs. 12, 53).

[89] Gregory Grefenstette y Pasi Tapanainen. "What is a word, what is a sentence?: problems of Tokenisation". En: (1994) (vid. pág. 26).

[90] Frank L Greitzer, Christine F Noonan y Lyndsey Franklin. *Cognitive foundations for visual analytics*. Inf. téc. Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2011 (vid. pág. 9).

[91] Octavian Grigorescu y col. "Cve2att&ck: Bert-based mapping of cves to mitre att&ck techniques". En: *Algorithms* 15.9 (2022), pág. 314 (vid. pág. 100).

[92] Christos Grigoriadis y col. "A cybersecurity ontology to support risk information gathering in cyber-physical systems". En: *European Symposium on Research in Computer Security*. Springer. 2021, págs. 23-39 (vid. pág. 23).

[93] Kumar Gunjan, G Sahoo y RK Tiwari. "Identity management in cloud computing–a review". En: *International Journal of Engineering Research & Technology* 1.4 (2012), págs. 1-5 (vid. pág. 55).

[94] Haixuan Guo, Shuhan Yuan y Xintao Wu. "Logbert: Log anomaly detection via bert". En: *2021 international joint conference on neural networks (IJCNN)*. IEEE. 2021, págs. 1-8 (vid. pág. 106).

[95] Hamed HaddadPajouh y col. "A deep recurrent neural network based approach for internet of things malware threat hunting". En: *Future Generation Computer Systems* 85 (2018), págs. 88-96 (vid. págs. 8, 21).

[96] John T Hancock y Taghi M Khoshgoftaar. "Survey on categorical data for neural networks". En: *Journal of Big Data* 7.1 (2020), págs. 1-41 (vid. pág. 25).

[97] Simon Hansman y Ray Hunt. "A taxonomy of network and computer attacks". En: *Computers & Security* 24.1 (2005), págs. 31-43 (vid. pág. 23).

[98] William Hatcher y col. "CyberExpert: Towards an Automated Framework for Cybersecurity Expertise Acquisition and Mastery". En: *2023 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2023, págs. 1-7 (vid. pág. 58).

[99] Ahmad Mtair AL-Hawamleh. "Predictions of cybersecurity experts on future cyber-attacks and related cybersecurity measures". En: *International Journal of Advanced Computer Science and Applications* 14.2 (2023) (vid. pág. 2).

[100] Heinrich Herre. "General Formal Ontology (GFO): A foundational ontology for conceptual modelling". En: *Theory and applications of ontology: computer applications*. Springer, 2010, págs. 297-345 (vid. pág. 23).

[101] Joshua Hill. *An analysis of the RADIUS authentication protocol*. 2001 (vid. pág. 56).

[102] Peter Himschoot. *Microsoft Blazor: Building Web Applications in. NET 6 and Beyond*. Springer, 2021 (vid. pág. 58).

[103] Sepp Hochreiter y Jürgen Schmidhuber. "Long short-term memory". En: *Neural computation* 9.8 (1997), págs. 1735-1780 (vid. pág. 36).

[104] Laura Hokkanen, Kati Kuusinen y Kaisa Väänänen. "Early product design in startups: towards a UX strategy". En: *Product-Focused Software Process Improvement: 16th International Conference, PROFES 2015, Bolzano, Italy, December 2-4, 2015, Proceedings 16*. Springer. 2015, págs. 217-224 (vid. pág. 43).

[105] Sajad Homayoun y col. "Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence". En: *IEEE transactions on emerging topics in computing* 8.2 (2017), págs. 341-351 (vid. pág. 21).

[106] Mohammad Hossain y Sameer Abufardeh. "A New Method of Calculating Squared Euclidean Distance (SED) Using pTree Technology and Its Performance Analysis." En: *CATA*. 2019, págs. 45-54 (vid. pág. 28).

[107] Sergey Hruschev y Svetlana Cherenkova. "Building a RADIUS Protocol Based Authentication Authorization and Accounting System". En: () (vid. pág. 58).

[108] Yen-Chang Hsu y col. "A closer look at knowledge distillation with features, logits, and gradients". En: *arXiv preprint arXiv:2203.10163* (2022) (vid. pág. 110).

[109] Michael Iannacone y col. "Developing an ontology for cyber security knowledge graphs". En: *Proceedings of the 10th Annual Cyber and Information Security Research Conference*. 2015, págs. 1-4 (vid. pág. 23).

[110] *IBM X-Force Exchange*. `https://exchange.xforce.ibmcloud.com/`. Last accessed 03 March 2023 (vid. pág. 10).

[111] I Indu, Rubesh Anand PM y Vidhyacharan Bhaskar. "Encrypted token based authentication with adapted SAML technology for cloud web services". En: *Journal of Network and Computer Applications* 99 (2017), págs. 131-145 (vid. pág. 56).

[112] Joab Jackson. *Return of the Monolith: Amazon Dumps Microservices for Video Monitoring*. Mayo de 2023 (vid. pág. 12).

[113] Amir Namavar Jahromi y col. "An enhanced stacked LSTM method with no random initialization for malware threat hunting in safety and time-critical systems". En: *IEEE Transactions on Emerging Topics in Computational Intelligence* 4.5 (2020), págs. 630-640. DOI: `10.1109/TETCI.2019.2910243` (vid. págs. 8, 21).

[114] Norman Jansen y col. "NATO Core Services profiling for Hybrid Tactical Networks—Results and Recommendations". En: *2021 International Conference on Military Communication and Information Systems (ICMCIS)*. IEEE. 2021, págs. 1-8 (vid. pág. 18).

[115] Nishtha Jatana y col. "A survey and comparison of relational and non-relational database". En: *International Journal of Engineering Research & Technology* 1.6 (2012), págs. 1-5 (vid. págs. 49, 50).

[116] Anjali Ganesh Jivani y col. "A comparative study of stemming algorithms". En: *Int. J. Comp. Tech. Appl* 2.6 (2011), págs. 1930-1938 (vid. pág. 26).

[117] Norm Jouppi y col. "Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings". En: *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 2023, págs. 1-14 (vid. pág. 8).

[118] Alexandre Kabil, Thierry Duval y Nora Cuppens. "Alert characterization by non-expert users in a cybersecurity virtual environment: a usability study". En: *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*. Springer. 2020, págs. 82-101 (vid. pág. 9).

[119] Alexandre Kabil y col. "From cyber security activities to collaborative virtual environments practices through the 3D cybercop platform". En: *International Conference on Information Systems Security*. Springer. 2018, págs. 272-287 (vid. pág. 9).

[120] Anas Abou El Kalam y col. "Organization based access control". En: *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*. IEEE. 2003, págs. 120-131 (vid. pág. 56).

[121] Karambir Kaur y Monika Sachdeva. "Performance evaluation of NewSQL databases". En: *2017 International Conference on Inventive Systems and Control (ICISC)*. IEEE. 2017, págs. 1-5 (vid. pág. 49).

[122] Jacqueline Kazil y Katharine Jarmul. *Data wrangling with python: tips and tools to make your life easier*. " O'Reilly Media, Inc.", 2016 (vid. pág. 24).

[123] Dominique Kelly y col. "Bing Chat: The Future of Search Engines?" En: *Proceedings of the Association for Information Science and Technology* 60.1 (2023), págs. 1007-1009 (vid. pág. 35).

[124] Alam Sher Khan y col. "Personality classification from online text using machine learning approach". En: *International journal of advanced computer science and applications* 11.3 (2020), págs. 460-476 (vid. pág. 27).

[125] Kamran Khan y col. "DBSCAN: Past, present and future". En: *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*. IEEE. 2014, págs. 232-238 (vid. pág. 30).

[126] Rahat Khan y col. "Impact of Conversational and Generative AI Systems on Libraries: A Use Case Large Language Model (LLM)". En: *Science & Technology Libraries* (2023), págs. 1-15 (vid. pág. 35).

[127] Sajad Khorsandroo y col. "Hybrid SDN evolution: A comprehensive survey of the state-of-the-art". En: *Computer Networks* 192 (2021), pág. 107981 (vid. pág. 8).

[128] Elmar Kiesling y col. "The SEPSES knowledge graph: An integrated resource for cybersecurity". En: *International Semantic Web Conference*. Springer. 2019, págs. 198-214 (vid. pág. 23).

[129] Joseph Migga Kizza. "Authentication". En: *Guide to Computer Network Security*. Springer, 2024, págs. 215-238 (vid. pág. 54).

[130] Donald E. Knuth. *The Art of Computer Programming, Volume 3: (2nd Ed.) Sorting and Searching*. USA: Addison Wesley Longman Publishing Co., Inc., 1998. ISBN: 0201896850 (vid. pág. 48).

[131] Marcin Kolny. *Scaling up the Prime Video audio/video monitoring service and reducing costs by 90%*. Mar. de 2023 (vid. pág. 12).

[132] Jana Komárková y col. "Crusoe: Data model for cyber situational awareness". En: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. 2018, págs. 1-10 (vid. pág. 23).

[133] Jason Kopylec, Anita D'Amico y John Goodall. "Visualizing cascading failures in critical cyber infrastructures". En: *International Conference on Critical Infrastructure Protection*. Springer. 2007, págs. 351-364 (vid. pág. 9).

[134] Igor Kotenko y Evgenia Novikova. "Visualization of security metrics for cyber situation awareness". En: *2014 Ninth International Conference on Availability, Reliability and Security*. IEEE. 2014, págs. 506-513 (vid. pág. 9).

[135] Alexander Kott, Cliff Wang y Robert F Erbacher. *Cyber defense and situational awareness*. Vol. 62. Springer, 2015 (vid. pág. 9).

[136] Maximilian Kroschewski y Anja Lehmann. "Save The Implicit Flow? Enabling Privacy-Preserving RP Authentication in OpenID Connect". En: *Proceedings on Privacy Enhancing Technologies* 4 (2023), págs. 96-116 (vid. pág. 56).

[137] Koojana Kuladinithi y col. "Implementation of coap and its application in transport logistics". En: *Proc. IP+ SN, Chicago, IL, USA* (2011) (vid. pág. 19).

[138] Kaur Kullman, Noam Ben Asher y Char Sample. "Operator impressions of 3D visualizations for cybersecurity analysts". En: *ECCWS 2019 18th European Conference on Cyber Warfare and Security*. Academic Conferences y publishing limited. 2019, pág. 257 (vid. pág. 9).

[139] Kaur Kullman, Jennifer Cowley y Noam Ben-Asher. "Enhancing cyber defense situational awareness using 3D visualizations". En: *Proceedings of the 13th International Conference on Cyber Warfare and Security ICCWS 2018: National Defense University, Washington DC, USA 8*. 2018, págs. 369-378 (vid. pág. 9).

[140] Mary C Lacity, Leslie P Willcocks y Ashok Subramanian. "A strategic client/server implementation: new technology, lessons from history". En: *The Journal of Strategic Information Systems* 6.2 (1997), págs. 95-128 (vid. pág. 11).

[141] A Lakshmanarao, M Raja Babu y MM Bala Krishna. "Malicious URL Detection using NLP, Machine Learning and FLASK". En: *2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*. IEEE. 2021, págs. 1-4 (vid. pág. 35).

[142] Antonio M Larriba y Damián López. "How to Grant Anonymous Access". En: *IEEE Transactions on Information Forensics and Security* 18 (2022), págs. 613-625 (vid. pág. 54).

[143] JooHwa Lee y KeeHyun Park. "GAN-based imbalanced data intrusion detection system". En: *Personal and Ubiquitous Computing* 25 (2021), págs. 121-128 (vid. pág. 32).

[144] Sunho Lee y col. "Tnpu: Supporting trusted execution with tree-less integrity protection for neural processing unit". En: *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE. 2022, págs. 229-243 (vid. pág. 8).

[145] Paul Lehner y col. "Cognitive biases and time stress in team decision making". En: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 27.5 (1997), págs. 698-703 (vid. pág. 2).

[146] David Lengweiler, Marco Vogt y Heiko Schuldt. "MMSBench-Net: Scenario-Based Evaluation of Multi-Model Database Systems". En: (2023) (vid. pág. 51).

[147] Patrick Lewis y col. "Retrieval-augmented generation for knowledge-intensive nlp tasks". En: *Advances in Neural Information Processing Systems* 33 (2020), págs. 9459-9474 (vid. pág. 86).

[148] Jian-hua Li. "Cyber security meets artificial intelligence: a survey". En: *Frontiers of Information Technology & Electronic Engineering* 19.12 (2018), págs. 1462-1474 (vid. pág. 2).

[149] Jingbo Li y Rongli Gai. "Survey of performance comparison based on non relational database". En: *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE. 2021, págs. 2278-2282 (vid. pág. 51).

[150] Weixi Li. *Automatic log analysis using machine learning: awesome automatic log analysis version 2.0*. 2013 (vid. pág. 33).

[151] Zitong Li y col. "A hierarchical approach for advanced persistent threat detection with attention-based graph neural networks". En: *Security and Communication Networks* 2021 (2021), págs. 1-14 (vid. pág. 99).

[152] Junyan Liang y Yoohwan Kim. "Evolution of Firewalls: Toward Securer Network Using Next Generation Firewall". En: *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE. 2022, págs. 0752-0759 (vid. pág. 10).

[153] Yen-Ting Lin y Yun-Nung Chen. "LLM-Eval: Unified Multi-Dimensional Automatic Evaluation for Open-Domain Conversations with Large Language Models". En: *arXiv preprint arXiv:2305.13711* (2023) (vid. pág. 35).

[154] Peng Liu, Dong Zhou y Naijun Wu. "VDBSCAN: varied density based spatial clustering of applications with noise". En: *2007 International conference on service systems and service management*. IEEE. 2007, págs. 1-4 (vid. pág. 30).

[155] Tong Liu y col. "Demystifying rce vulnerabilities in llm-integrated apps". En: *arXiv preprint arXiv:2309.02926* (2023) (vid. págs. 38, 86).

[156] Ying Liu, Beth Plale y col. "Survey of publish subscribe event systems". En: *Computer Science Dept, Indian University* 16 (2003) (vid. pág. 17).

[157] Salvador Llopis y col. "A comparative analysis of visualisation techniques to achieve cyber situational awareness in the military". En: *2018 Interna-*

tional Conference on Military Communications and Information Systems (ICMCIS). IEEE. 2018, págs. 1-7 (vid. pág. 9).

[158] Steffen Lohmann y col. "Concentri cloud: Word cloud visualization for multiple text documents". En: *2015 19th International Conference on Information Visualisation*. IEEE. 2015, págs. 114-120 (vid. pág. 9).

[159] Andrew Lombardi. *WebSocket: lightweight client-server communications*. " O'Reilly Media, Inc.", 2015 (vid. pág. 19).

[160] Julie Beth Lovins. "Development of a stemming algorithm." En: *Mech. Transl. Comput. Linguistics* 11.1-2 (1968), págs. 22-31 (vid. pág. 26).

[161] Jeff Magee y Jeff Kramer. "Dynamic structure in software architectures". En: *ACM SIGSOFT Software Engineering Notes* 21.6 (1996), págs. 3-14 (vid. pág. 10).

[162] Jeff Magee y col. "Specifying distributed software architectures". En: *Software Engineering—ESEC'95: 5th European Software Engineering Conference Sitges, Spain, September 25–28, 1995 Proceedings 5*. Springer. 1995, págs. 137-153 (vid. pág. 10).

[163] Mark W Maier, David Emery y Rich Hilliard. "Software architecture: Introducing IEEE standard 1471". En: *Computer* 34.4 (2001), págs. 107-109 (vid. pág. 10).

[164] Rui Mao y col. "GPTEval: A survey on assessments of ChatGPT and GPT-4". En: *arXiv preprint arXiv:2308.12488* (2023) (vid. págs. 7, 35, 85).

[165] Francesco Mariotti y col. "Extending a security ontology framework to model CAPEC attack paths and TAL adversary profiles". En: *2022 18th European Dependable Computing Conference (EDCC)*. IEEE. 2022, págs. 25-32 (vid. pág. 23).

[166] G Markowsky y L Markowsky. "Visualizing cybersecurity events". En: *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer … 2013, pág. 1 (vid. pág. 3).

[167] Wendy L Martinez. "Graphical user interfaces". En: *Wiley Interdisciplinary Reviews: Computational Statistics* 3.2 (2011), págs. 119-133 (vid. pág. 40).

[168] Mark Masse. *REST API design rulebook: designing consistent RESTful web service interfaces.* " O'Reilly Media, Inc.", 2011 (vid. págs. 38, 42).

[169] Sherin Mary Mathews. "Explainable artificial intelligence applications in NLP, biomedical, and malware classification: a literature review". En: *Intelligent computing-proceedings of the computing conference*. Springer. 2019, págs. 1269-1292 (vid. pág. 33).

[170] William J Matuszak, Lisa DiPippo y Yan Lindsay Sun. "Cybersave: situational awareness visualization for cyber security of smart grid systems". En: *Proceedings of the Tenth Workshop on Visualization for Cyber Security*. 2013, págs. 25-32 (vid. pág. 9).

[171] Henock Mulugeta Melaku. "A dynamic and adaptive cybersecurity governance framework". En: *Journal of Cybersecurity and Privacy* 3.3 (2023), págs. 327-350 (vid. pág. 2).

[172] Peter Mell y Tim Grance. *Use of the common vulnerabilities and exposures (cve) vulnerability naming scheme*. Inf. téc. National Inst Of Standards y Technology Gaithersburg Md Computer Security Div, 2002 (vid. pág. 39).

[173] Alexey Melnikov y Ian Fette. *The WebSocket Protocol*. RFC 6455. Dic. de 2011. DOI: 10.17487/RFC6455 (vid. pág. 19).

[174] Tom Mens y col. "Software architecture evolution". En: *Software Evolution* (2008), págs. 233-262 (vid. pág. 11).

[175] Christopher Metz. "AAA protocols: authentication, authorization, and accounting for the Internet". En: *IEEE Internet Computing* 3.6 (1999), págs. 75-79 (vid. pág. 56).

[176] Biswajeeban Mishra. "Performance evaluation of MQTT broker servers". En: *International Conference on Computational Science and Its Applications*. Springer. 2018, págs. 599-609 (vid. pág. 54).

[177] Biswajeeban Mishra, Biswaranjan Mishra y Attila Kertesz. "Stress-testing MQTT brokers: A comparative analysis of performance measurements". En: *Energies* 14.18 (2021), pág. 5817 (vid. pág. 54).

[178] Rohitshankar Mishra, Ishfaq Ahmad y Akshaya Sharma. "A Dynamic Multi-Threaded Queuing Mechanism for Reducing the Inter-Process Communication Latency on Multi-Core Chips". En: *2020 3rd International Conference on Data Intelligence and Security (ICDIS)*. IEEE. 2020, págs. 12-19 (vid. pág. 41).

[179] Olof Mogren. "C-RNN-GAN: Continuous recurrent neural networks with adversarial training". En: *arXiv preprint arXiv:1611.09904* (2016) (vid. págs. 32, 37).

[180] Andreas Molzer. *oxide-auth*. Ver. 0.5.4. Sep. de 2023 (vid. pág. 58).

[181] Daniel L Moody y Graeme G Shanks. "What makes a good data model? Evaluating the quality of entity relationship models". En: *International Conference on Conceptual Modeling*. Springer. 1994, págs. 94-111 (vid. pág. 23).

[182] Guilherme Baesso Moreira y col. "Extending the VERIS framework to an incident handling ontology". En: *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE. 2018, págs. 440-445 (vid. pág. 23).

[183] Mohsen Mosleh, Kia Dalili y Babak Heydari. "Distributed or monolithic? a computational architecture decision framework". En: *IEEE Systems journal* 12.1 (2016), págs. 125-136 (vid. págs. 11, 13).

[184] Sape J Mullender. "Introduction to distributed systems". En: *15th CERN School of Computing, CSC 1992*. CERN. 1992, págs. 29-46 (vid. pág. 11).

[185] Fionn Murtagh y Pedro Contreras. "Algorithms for hierarchical clustering: an overview". En: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.1 (2012), págs. 86-97 (vid. pág. 29).

[186] Sarang Na, Taeeun Kim y Hwankuk Kim. "A study on the classification of common vulnerabilities and exposures using naïve bayes". En: *International*

Conference on Broadband and Wireless Computing, Communication and Applications. Springer. 2016, págs. 657-662 (vid. pág. 39).

[187] Nitin Naik y col. "Cyberthreat Hunting-Part 1: triaging ransomware using fuzzy hashing, import hashing and YARA rules". En: *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE. 2019, págs. 1-6 (vid. pág. 73).

[188] Nitin Naik y col. "Embedded YARA rules: strengthening YARA rules utilising fuzzy hashing and fuzzy rules for malware analysis". En: *Complex & Intelligent Systems* 7.2 (2021), págs. 687-702 (vid. pág. 73).

[189] Antonio José Horta Neto y Anderson Fernandes Pereira dos Santos. "Cyber threat hunting through automated hypothesis and multi-criteria decision making". En: *2020 IEEE International Conference on Big Data (Big Data)*. IEEE. 2020, págs. 1823-1830 (vid. pág. 21).

[190] Frank Nielsen y Frank Nielsen. "Hierarchical clustering". En: *Introduction to HPC with MPI for Data Science* (2016), págs. 195-211 (vid. pág. 29).

[191] Henrik Nielsen y col. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. Jun. de 1999. DOI: 10.17487/RFC2616 (vid. pág. 18).

[192] John O'Hara. "Toward a commodity enterprise middleware: Can AMQP enable a new era in messaging middleware? A look inside standards-based messaging with AMQP". En: *Queue* 5.4 (2007), págs. 48-55 (vid. pág. 21).

[193] Haroon Shakirat Oluwatosin. "Client-server model". En: *IOSRJ Comput. Eng* 16.1 (2014), págs. 2278-8727 (vid. pág. 14).

[194] Ömer. *What is sigma? threat hunting in Siem products with sigma rules - example sigma rules*. Mar. de 2021 (vid. págs. 73, 83).

[195] Ashwin Pajankar y Aditya Joshi. "Preparing data for machine learning". En: *Hands-on Machine Learning with Python: Implement Neural Network Solutions with Scikit-learn and PyTorch*. Springer, 2022, págs. 79-97 (vid. pág. 24).

[196] Sowndarya Palanisamy y P SuvithaVani. "A survey on RDBMS and NoSQL Databases MySQL vs MongoDB". En: *2020 International Conference on*

Computer Communication and Informatics (ICCCI). IEEE. 2020, págs. 1-7 (vid. págs. 49, 50).

[197] Suman Patro, Manish Potey y Amit Golhani. "Comparative study of middleware solutions for control and monitoring systems". En: *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE. 2017, págs. 1-10 (vid. pág. 18).

[198] Joan Peckham y Fred Maryanski. "Semantic data models". En: *ACM Computing Surveys (CSUR)* 20.3 (1988), págs. 153-189 (vid. pág. 23).

[199] Stan Pietrowicz y col. "Web-Based Smart Grid Network Analytics Framework". En: *2015 IEEE International Conference on Information Reuse and Integration*. IEEE. 2015, págs. 496-501 (vid. pág. 9).

[200] William A Pike, Chad Scherrer y Sean Zabriskie. "Putting security in context: Visual correlation of network activity with real-world information". En: *VizSEC 2007*. Springer, 2008, págs. 203-220 (vid. pág. 9).

[201] PRAETORIAN. *D3.1 Transitioning Risk Management*. 2021 (vid. págs. 2, 10).

[202] Joan Puigcerver y col. "From sparse to soft mixtures of experts". En: *arXiv preprint arXiv:2308.00951* (2023) (vid. pág. 110).

[203] Dorian Pyle. *Data preparation for data mining*. morgan kaufmann, 1999 (vid. pág. 24).

[204] Zheng Qin, Xiang Zheng y Jiankuan Xing. *Introduction to software architecture*. Springer, 2008 (vid. pág. 10).

[205] Shirley Radack y Rick Kuhn. "Managing security: The security content automation protocol". En: *IT professional* 13.1 (2011), págs. 9-11 (vid. pág. 39).

[206] Abir Rahali y Moulay A Akhloufi. "MalBERT: Using transformers for cybersecurity and malicious software detection". En: *arXiv preprint arXiv:2103.03806* (2021) (vid. pág. 34).

[207] Abir Rahali y Moulay A Akhloufi. "MalBERTv2: Code Aware BERT-Based Model for Malware Identification". En: *Big Data and Cognitive Computing* 7.2 (2023), pág. 60 (vid. pág. 34).

[208] P Rajesh y col. "Analysis of cyber threat detection and emulation using mitre attack framework". En: *2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*. IEEE. 2022, págs. 4-12 (vid. pág. 100).

[209] Anandharaju Durai Raju y col. "A survey on cross-architectural IoT malware threat hunting". En: *IEEE Access* 9 (2021), págs. 91686-91709 (vid. págs. 8, 21).

[210] Natasya Titania Ramadhanti, Cucuk Wawan Budiyanto y Rosihan Ari Yuana. "The use of heuristic evaluation on UI/UX design: A review to anticipate web app's usability". En: *AIP Conference Proceedings*. Vol. 2540. 1. AIP Publishing. 2023 (vid. pág. 43).

[211] David Ramos. *openidconnect-rs*. Ver. 3.4.0. Oct. de 2023 (vid. pág. 58).

[212] Priyanka Ranade y col. "Cybert: Contextualized embeddings for the cybersecurity domain". En: *2021 IEEE International Conference on Big Data (Big Data)*. IEEE. 2021, págs. 3334-3342 (vid. pág. 34).

[213] John Reed. *Threat hunting with ML: Another reason to SMLE*. Feb. de 2021 (vid. pág. 10).

[214] Alexander Riegler. "The role of anticipation in cognition". En: *AIP Conference Proceedings*. Vol. 573. American Institute of Physics. 2001, págs. 534-541 (vid. pág. 2).

[215] Raúl Riesco y Víctor A Villagrá. "Leveraging cyber threat intelligence for a dynamic risk framework". En: *International Journal of Information Security* 18.6 (2019), págs. 715-739 (vid. pág. 39).

[216] Carlos Rodríguez-Domínguez y col. "A communication model to integrate the request-response and the publish-subscribe paradigms into ubiquitous systems". En: *Sensors* 12.6 (2012), págs. 7648-7668 (vid. pág. 16).

[217] Maciej Rostanski, Krzysztof Grochla y Aleksander Seman. "Evaluation of highly available and fault-tolerant middleware clustered architectures using RabbitMQ". En: *2014 federated conference on computer science and information systems*. IEEE. 2014, págs. 879-884 (vid. pág. 53).

[218] Mary Tork Roth y Peter M Schwarz. "Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources." En: *VLDB*. Vol. 97. 1997, págs. 25-29 (vid. pág. 22).

[219] Stuart J Russell, Peter Norvig y Peter Norvig. "Artificial intelligence: Prentice Hall series in artificial intelligence". En: (2003) (vid. pág. 30).

[220] T Rustamov y col. "Classification symbols of words". En: *Asian Journal of Research in Social Sciences and Humanities* 12.2 (2022), págs. 213-219 (vid. pág. 26).

[221] Dhia Elhaq Rzig y col. "Virtual reality (vr) automated testing in the wild: A case study on unity-based vr applications". En: *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2023, págs. 1269-1281 (vid. pág. 45).

[222] Hamid Reza Saeidnia. "Welcome to the Gemini era: Google DeepMind and the information industry". En: *Library Hi Tech News* ahead-of-print (2023) (vid. pág. 35).

[223] Tasneem Salah y col. "The evolution of distributed systems towards microservices architecture". En: *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE. 2016, págs. 318-325 (vid. págs. 11, 14-16).

[224] Ravi Sandhu y Pierangela Samarati. "Authentication, access control, and audit". En: *ACM Computing Surveys (CSUR)* 28.1 (1996), págs. 241-243 (vid. pág. 54).

[225] Ravi S Sandhu. "Role-based access control". En: *Advances in computers*. Vol. 46. Elsevier, 1998, págs. 237-286 (vid. págs. 54, 56).

[226] Ravi S Sandhu y Pierangela Samarati. "Access control: principle and practice". En: *IEEE communications magazine* 32.9 (1994), págs. 40-48 (vid. pág. 55).

[227] Imanol Schlag, Kazuki Irie y Jürgen Schmidhuber. "Linear transformers are secretly fast weight programmers". En: *International Conference on Machine Learning*. PMLR. 2021, págs. 9355-9366 (vid. pág. 35).

[228] Steven Schmitt. *Advanced threat hunting over software-defined networks in smart cities*. 2018 (vid. págs. 8, 21).

[229] Steven Schmitt, Farah I Kandah y Dylan Brownell. "Intelligent threat hunting in software-defined networking". En: *2019 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE. 2019, págs. 1-5 (vid. págs. 8, 21).

[230] Michael Schwartz y col. "SAML". En: *Securing the Perimeter: Deploying Identity and Access Management with Free Open Source Software* (2018), págs. 59-103 (vid. pág. 58).

[231] Gian Luca Scoccia y Marco Autili. "Web frameworks for desktop apps: an exploratory study". En: *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2020, págs. 1-6 (vid. pág. 44).

[232] Oscar Serrano, Luc Dandurand y Sarah Brown. "On the design of a cyber security data sharing system". En: *proceedings of the 2014 ACM workshop on information sharing & collaborative security*. 2014, págs. 61-69 (vid. pág. 39).

[233] Sourabh Sharma. *Modern API Development with Spring and Spring Boot: Design highly scalable and maintainable APIs with REST, gRPC, GraphQL, and the reactive paradigm*. Packt Publishing Ltd, 2021 (vid. pág. 42).

[234] Mary Shaw y David Garlan. *Software architecture: perspectives on an emerging discipline*. Prentice-Hall, Inc., 1996 (vid. pág. 10).

[235] Zach Shelby, Klaus Hartke y Carsten Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. Jun. de 2014. DOI: 10.17487/RFC7252 (vid. pág. 19).

[236] Sima Siami-Namini, Neda Tavakoli y Akbar Siami Namin. "The performance of LSTM and BiLSTM in forecasting time series". En: *2019 IEEE*

International Conference on Big Data (Big Data). IEEE. 2019, págs. 3285-3292 (vid. pág. 33).

[237] Kanchan Singh, Sakshi S Grover y Ranjini Kishen Kumar. "Cyber Security Vulnerability Detection Using Natural Language Processing". En: *2022 IEEE World AI IoT Congress (AIIoT)*. IEEE. 2022, págs. 174-178 (vid. págs. 33, 34).

[238] Timothy J Slattery y Mark Yates. "Word skipping: Effects of word length, predictability, spelling and reading skill". En: *Quarterly Journal of Experimental Psychology* 71.1 (2018), págs. 250-259 (vid. pág. 2).

[239] Stefano Soatto y col. "Taming AI Bots: Controllability of Neural States in Large Language Models". En: *arXiv preprint arXiv:2305.18449* (2023) (vid. pág. 107).

[240] Xinying Song y col. "Fast wordpiece tokenization". En: *arXiv preprint arXiv:2012.15524* (2020) (vid. pág. 26).

[241] Peter Spyns, Robert Meersman y Mustafa Jarrar. "Data modelling versus ontology engineering". En: *ACM SIGMod Record* 31.4 (2002), págs. 12-17 (vid. pág. 23).

[242] Raj Srinivasan. *RPC: Remote Procedure Call Protocol Specification Version 2*. RFC 1831. Ago. de 1995. DOI: 10.17487/RFC1831 (vid. pág. 42).

[243] Ralf C Staudemeyer y Eric Rothstein Morris. "Understanding LSTM–a tutorial into long short-term memory recurrent neural networks". En: *arXiv preprint arXiv:1909.09586* (2019) (vid. pág. 33).

[244] Cristina Stolojescu-Crisan, Calin Crisan y Bogdan-Petru Butunoi. "An IoT-based smart home automation system". En: *Sensors* 21.11 (2021), pág. 3784 (vid. pág. 8).

[245] Ruoyu Su, Xiaozhou Li y Davide Taibi. "Back to the Future: From Microservice to Monolith". En: *arXiv preprint arXiv:2308.15281* (2023) (vid. pág. 12).

[246] Yiyi Sun y Yiyi Sun. "Server-Side Rendering". En: *Practical Application Development with AppRun: Building Reliable, High-Performance Web Apps*

Using Elm-Inspired Architecture, Event Pub-Sub, and Components (2019), págs. 191-217 (vid. pág. 41).

[247] *SurrealDB*. `https://surrealdb.com/`. 2023 (vid. pág. 51).

[248] Ilya Sutskever, Oriol Vinyals y Quoc V Le. "Sequence to sequence learning with neural networks". En: *Advances in neural information processing systems* 27 (2014) (vid. pág. 36).

[249] E Burton Swanson. "The dimensions of maintenance". En: *Proceedings of the 2nd international conference on Software engineering*. 1976, págs. 492-497 (vid. pág. 15).

[250] Thomas Sydorowski. *aaa4j-radius*. Ver. 0.2.4. Nov. de 2023 (vid. pág. 58).

[251] Rebecca Taft y col. "Cockroachdb: The resilient geo-distributed sql database". En: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2020, págs. 1493-1509 (vid. pág. 49).

[252] Susanne Tak y Andy Cockburn. "Enhanced spatial stability with hilbert and moore treemaps". En: *IEEE Transactions on Visualization and Computer Graphics* 19.1 (2012), págs. 141-148 (vid. pág. 9).

[253] Freddy Tapia y col. "From monolithic systems to microservices: A comparative study of performance". En: *Applied sciences* 10.17 (2020), pág. 5797 (vid. pág. 15).

[254] Aakanksha Tashildar y col. "Application development using flutter". En: *International Research Journal of Modernization in Engineering Technology and Science* 2.8 (2020), págs. 1262-1266 (vid. pág. 44).

[255] *The security immune system: An integrated approach to protecting your organization*. `https://www.midlandinfosys.com/pdf/qradar-siem-cybersecurity-ai-products.pdf`. Last accessed 03 March 2023 (vid. pág. 10).

[256] Roshan K Thomas. "Team-based access control (TMAC) a primitive for applying role-based access controls in collaborative environments". En: *Proceedings of the second ACM workshop on Role-based access control*. 1997, págs. 13-19 (vid. pág. 57).

[257] Renato M Toasa y col. "Mobile Development with Xamarin: Brief Literature, Visualizations and Important Issues". En: *International Conference on Information Technology & Systems*. Springer. 2023, págs. 299-307 (vid. pág. 44).

[258] William Tolone y col. "Access control in collaborative systems". En: *ACM Computing Surveys (CSUR)* 37.1 (2005), págs. 29-41 (vid. págs. 55-57).

[259] Oguzhan Topsakal y Tahir Cetin Akinci. "Creating large language model applications utilizing langchain: A primer on developing llm apps fast". En: *International Conference on Applied Engineering and Natural Sciences*. Vol. 1. 1. 2023, págs. 1050-1056 (vid. págs. 38, 86).

[260] Hugo Touvron y col. "Llama 2: Open foundation and fine-tuned chat models". En: *arXiv preprint arXiv:2307.09288* (2023) (vid. pág. 85).

[261] David Okore Ukwen y Murat Karabatak. "Review of NLP-based Systems in Digital Forensics and Cybersecurity". En: *2021 9th International Symposium on Digital Forensics and Security (ISDFS)*. IEEE. 2021, págs. 1-9 (vid. pág. 33).

[262] Ashish Vaswani y col. "Attention is all you need". En: *Advances in neural information processing systems* 30 (2017) (vid. págs. 35, 36).

[263] Kuldeep Vayadande y col. "AI-Based Image Generator Web Application using OpenAI's DALL-E System". En: *2023 International Conference on Recent Advances in Science and Engineering Technology (ICRASET)*. IEEE. 2023, págs. 1-5 (vid. pág. 37).

[264] Aditya Venkataraman y Kishore Kumar Jagadeesha. "Evaluation of interprocess communication mechanisms". En: *Architecture* 86 (2015), pág. 64 (vid. pág. 41).

[265] Remco Verhoef. *Sigma rules! the generic signature format for SIEM systems.* (Vid. págs. 73, 83).

[266] Paulo Verissimo y Luis Rodrigues. *Distributed systems for system architects*. Vol. 1. Springer Science & Business Media, 2001 (vid. pág. 11).

[267] Nico Vermeir. "Runtimes and Desktop Packs". En: *Introducing. NET 6: Getting Started with Blazor, MAUI, Windows App SDK, Desktop Development, and Containers*. Springer, 2022, págs. 21-29 (vid. pág. 44).

[268] Matthias Vianden, Horst Lichter y Andreas Steffens. "Experience on a microservice-based reference architecture for measurement systems". En: *2014 21st Asia-Pacific Software Engineering Conference*. Vol. 1. IEEE. 2014, págs. 183-190 (vid. pág. 15).

[269] Andrej Volchkov. "Revisiting single sign-on: a pragmatic approach in a new context". En: *IT Professional* 3.1 (2001), págs. 39-45 (vid. pág. 55).

[270] Ulrike Von Luxburg. "A tutorial on spectral clustering". En: *Statistics and computing* 17 (2007), págs. 395-416 (vid. pág. 29).

[271] Kaijun Wang y col. "Adaptive affinity propagation clustering". En: *arXiv preprint arXiv:0805.1096* (2008) (vid. pág. 28).

[272] David Wells y Derek Bryan. *Cyber operational architecture training system cyber for all*. Inf. téc. US Pacific Command J81 Camp HM Smith United States, 2015 (vid. pág. 39).

[273] Yvonne Wilson y Abhishek Hingnikar. "SAML 2". En: *Solving Identity Management in Modern Applications: Demystifying OAuth 2, OpenID Connect, and SAML 2*. Springer, 2022, págs. 127-141 (vid. pág. 56).

[274] Niklaus Wirth. *Algorithms & data structures*. Prentice-Hall, Inc., 1985 (vid. pág. 48).

[275] Zhongxiang Xiao, Inji Wijegunaratne y Xinjian Qiang. "Reflections on SOA and Microservices". En: *2016 4th International Conference on Enterprise Systems (ES)*. IEEE. 2016, págs. 60-67 (vid. pág. 14).

[276] Jin Xu, Yubo Tao y Hai Lin. "Semantic word cloud generation based on word embeddings". En: *2016 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE. 2016, págs. 239-243 (vid. pág. 9).

[277] Carl S Young. "Representing Cybersecurity Risk". En: *Cybercomplexity*. Springer, 2022, págs. 19-24 (vid. pág. 3).

[278] Hao Yuan y col. "Explainability in graph neural networks: A taxonomic survey". En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022) (vid. pág. 32).

[279] Hong Yuan y Ping-ping Gu. "Application of windows inter-process communication in software system integration". En: *2010 International Conference on Intelligent System Design and Engineering Application*. Vol. 1. IEEE. 2010, págs. 375-378 (vid. pág. 41).

[280] Wan-Lei Zhao, Cheng-Hao Deng y Chong-Wah Ngo. "k-means: A revisit". En: *Neurocomputing* 291 (2018), págs. 195-206 (vid. pág. 28).

[281] Chen Zhong y col. "AOH-map: A mind mapping system for supporting collaborative cyber security analysis". En: *2019 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*. IEEE. 2019, págs. 74-80 (vid. pág. 9).

[282] Guoqiang Zhong y col. "An overview on data representation learning: From traditional feature learning to recent deep learning". En: *The Journal of Finance and Data Science* 2.4 (2016), págs. 265-278 (vid. pág. 25).

[283] Shuigeng Zhou y col. "FDBSCAN: a fast DBSCAN algorithm". En: *Journal of Software* 11.6 (2000), págs. 735-744 (vid. pág. 30).

[284] Ying Zhuo, Qiang Zhang y Zhenghu Gong. "Cyberspace situation representation based on niche theory". En: *2008 International Conference on Information and Automation*. IEEE. 2008, págs. 1400-1405 (vid. pág. 9).