

**UNIVERSIDAD POLITÉCNICA DE
MADRID**

**ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN**



**MÁSTER UNIVERSITARIO EN
INGENIERÍA DE
TELECOMUNICACIONES**

TRABAJO FIN DE MÁSTER

**DESARROLLO DE UNA HERRAMIENTA
PARA LA SIMULACIÓN DE RUTAS DE
ATAQUE EN ENTORNOS DE
CONCIENCIA CIBERSITUACIONAL
BASADA EN PROCESOS ESTOCÁSTICOS**

SAMUEL GARCÍA SÁNCHEZ

2025

**UNIVERSIDAD POLITÉCNICA DE
MADRID**

**ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN**



**MÁSTER UNIVERSITARIO EN
INGENIERÍA DE
TELECOMUNICACIONES**

TRABAJO FIN DE MÁSTER

**DESARROLLO DE UNA HERRAMIENTA
PARA LA SIMULACIÓN DE RUTAS DE
ATAQUE EN ENTORNOS DE
CONCIENCIA CIBERSITUACIONAL
BASADA EN PROCESOS ESTOCÁSTICOS**

SAMUEL GARCÍA SÁNCHEZ

2025

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIONES

TRABAJO DE FIN DE MÁSTER

Título: Desarrollo de una herramienta para la simulación de rutas de ataque en entornos de conciencia cibersituacional basada en procesos estocásticos

Autor: D. Samuel García Sánchez.

Tutor: Carmen Sánchez Zas.

Departamento: Departamento de Ingeniería de Sistemas Telemáticos (DIT)

MIEMBROS DEL TRIBUNAL

Presidente:

Vocal:

Secretario:

Suplente:

Los miembros del tribunal nombrados acuerdan otorgar la calificación de:

Madrid, a de de 2025

Agradecimientos

A mi familia por apoyarme y a Carmen por guiarme y ayudarme en todo momento.

Gracias.

Resumen

En la actualidad, las organizaciones y gobiernos enfrentan un panorama de ciberamenazas en constante evolución. El crecimiento de la superficie de ataque, impulsado por la adopción de servicios en la nube y dispositivos conectados, ha hecho que las estrategias tradicionales de detección de intrusos sean insuficientes. En este contexto, la ciberdefensa basada en el análisis predictivo se presenta como una solución proactiva para anticipar y mitigar riesgos. Este enfoque no solo busca identificar vulnerabilidades individuales, sino comprender cómo pueden ser explotadas en conjunto para comprometer activos críticos.

El presente proyecto aborda esta problemática mediante el desarrollo de una herramienta de simulación de cadenas de ataque basada en la matriz MITRE ATT&CK. Su objetivo principal es modelar posibles rutas de ataque a través de la generación de secuencias probabilísticas de tácticas empleadas por adversarios y visualizar su impacto sobre escenarios de red previamente definidos y configurados. Para ello, el sistema implementa una cadena de Markov que simula la progresión de un ataque y se integra con la base de datos Neo4j para representar gráficamente las relaciones entre técnicas y activos afectados, facilitando la identificación de vulnerabilidades clave.

Esta aproximación del problema permite a las organizaciones evaluar sus estrategias de defensa con un alto grado de realismo y anticiparse a posibles amenazas antes de que estas se materialicen.

De forma general, los resultados obtenidos con ARGOS respaldan la hipótesis planteada por el presente trabajo, posicionando la herramienta como una solución alternativa viable y mejorada dentro del área de estudio. No obstante, aunque los datos experimentales son alentadores, sería recomendable ampliar el conjunto de pruebas con mayor diversidad y complejidad de casos reales para fortalecer la generalización del modelo.

El código empleado en este trabajo está disponible en el siguiente repositorio de GitHub:

<https://github.com/samugs13/argos>

Abstract

Currently, organizations and governments face an ever-evolving landscape of cyber threats. The expansion of the attack surface, driven by the adoption of cloud services and connected devices, has rendered traditional intrusion detection strategies insufficient. In this context, cyber defense based on predictive analysis emerges as a proactive solution to anticipate and mitigate risks. This approach not only seeks to identify individual vulnerabilities but also to understand how they can be exploited together to compromise critical assets.

This project addresses the issue by developing a simulation tool for attack chains based on the MITRE ATT&CK framework. Its main objective is to model possible attack paths by generating probabilistic sequences of tactics employed by adversaries and visualizing their impact on predefined and configured network scenarios. To achieve this, the system implements a Markov chain to simulate the progression of an attack and integrates with the Neo4j database to graphically represent the relationships between techniques and affected assets, facilitating the identification of key vulnerabilities.

This approach allows organizations to assess their defense strategies with a high degree of realism and anticipate potential threats before they materialize.

In general, the results obtained with ARGOS support the hypothesis proposed in this work, positioning the tool as a viable and improved alternative solution within the field of study. However, although the experimental data are encouraging, it would be advisable to expand the test set with a greater diversity and complexity of real-world cases to strengthen the generalization of the model.

The code developed for this project is available in the following GitHub repository:

<https://github.com/samugs13/argos>

Glosario

- **ETSIT:** Escuela Técnica Superior de Ingenieros de Telecomunicación
- **TFM:** Trabajo de Fin de Máster
- **TTPs:** Tactics, Techniques and Procedures
- **CLI:** Command Line Interface
- **GUI:** Graphical User Interface
- **IT:** Information Technology
- **OT:** Operational Technology
- **CVE:** Common Vulnerabilities and Exposures
- **CWE:** Common Weakness Enumeration
- **CAPEC:** Common Attack Pattern Enumeration and Classification

Índice general

Agradecimientos	I
Resumen	II
Abstract	III
Glosario	IV
Índice de figuras	VIII
Índice de tablas	IX
1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Estructura del documento	2
2. Marco teórico	4
2.1. MITRE ATT&CK	4
2.2. CVEs, CWEs y CAPECs	6
2.3. Procesos estocásticos	7
2.3.1. Cadenas de Markov	7
2.4. Rutas de ataque (<i>Attack Paths</i>)	9
2.5. Bases de datos orientadas a grafos	10
2.6. Tecnologías	11
2.6.1. Neo4j	11
2.6.2. Cypher	12
2.7. Librerías de Python	12
2.7.1. Neo4j	12
2.7.2. Rich	12
3. Diseño y propuesta	13
3.1. Requisitos de diseño	13
3.2. Solución propuesta	14
3.3. Etapas de diseño	15

4. Desarrollo	16
4.1. Obtención de los datos de entrada	17
4.1.1. Información sobre TTPs	17
4.1.2. Modelo de datos y escenarios de red	19
4.2. Preparación del entorno	26
4.2.1. Dependencias	26
4.2.2. Creación y configuración de la base de datos en Neo4j	26
4.3. Explicación de la herramienta	29
4.3.1. Comando <code>prepare</code>	29
4.3.2. Comando <code>attack</code>	30
4.3.3. Comando <code>trace</code>	39
4.3.4. Comando <code>history</code>	40
4.3.5. Comando <code>clean</code>	40
5. Resultados y validación	42
5.1. Carga de escenarios	43
5.1.1. Carga de escenario <i>Oficina Corporativa</i>	43
5.1.2. Carga de escenario <i>Smart home</i>	44
5.1.3. Carga de escenario <i>Planta Industrial</i>	46
5.2. Simulación de ataques sobre escenarios	47
5.2.1. Ataque a escenario <i>Oficina Corporativa</i>	47
5.2.2. Ataque a escenario <i>Smart Home</i>	52
5.2.3. Ataque a escenario <i>Planta Industrial</i>	56
5.3. Borrado de la base de datos	58
5.4. Historial de ataques	59
6. Conclusiones y líneas futuras	60
6.1. Conclusiones	60
6.2. Líneas futuras	61
Bibliografía	62
Anexos	65
A. Aspectos éticos, económicos, sociales y ambientales	66
B. Presupuesto Económico	68
C. Escenarios de red	71
D. Artículo JNIC	76

Índice de figuras

2.1.	Extracto de la matriz empresarial de MITRE ATT&CK [6]	5
2.2.	Comparativa entre los conceptos de CVE, CWE y CAPEC [10] . .	6
2.3.	Ejemplo de cadena de Markov [13]	8
2.4.	Ejemplo de ruta de ataque en <i>Active Directory</i> [17]	9
2.5.	Relaciones en una base de datos orientada a grafos [19]	10
2.6.	Logo de Neo4j [20]	11
3.1.	Diagrama básico de la arquitectura del sistema	14
4.1.	Diagrama detallado de la arquitectura del sistema	16
4.2.	Diagrama del modelo de datos empleado en los escenarios de red	22
4.3.	Escenario de red: Oficina Corporativa	23
4.4.	Escenario de red: <i>Smart Home</i>	24
4.5.	Escenario de red: <i>Planta Industrial</i>	25
4.6.	Botón para crear un nuevo proyecto	27
4.7.	Creación de base de datos	27
4.8.	Iniciación de la base de datos	27
4.9.	Base de datos iniciada correctamente	28
4.10.	Función <code>start_neo4j()</code>	28
4.11.	Consola de Neo4j con la conexión establecida	28
4.12.	Diagrama de ejecución del comando <code>prepare</code>	30
4.13.	Diagrama de ejecución del comando <code>attack</code>	31
4.14.	Diagrama del modelo de datos empleado en los ataques	34
4.15.	Diagrama del modelo de datos empleado en ARGOS	34
4.16.	Condiciones de explotación de una técnica sobre un activo	36
4.17.	Distribución de los paneles en el <i>dashboard</i>	38
4.18.	Diagrama de ejecución del comando <code>trace</code>	39
4.19.	Diagrama de ejecución del comando <code>history</code>	40
4.20.	Diagrama de ejecución del comando <code>clean</code>	40
5.1.	Diagrama de flujo de ARGOS	42
5.2.	Carga exitosa del escenario correspondiente a la oficina corporativa	43
5.3.	Escenario de oficina corporativa en Neo4j	43
5.4.	Carga exitosa del escenario <i>Smart Home</i>	44
5.5.	Escenario <i>Smart Home</i> en Neo4j	45
5.6.	Carga exitosa del escenario correspondiente a la planta industrial	46

5.7.	Escenario <i>Planta Industrial</i> en Neo4j	46
5.8.	Ataque sobre el escenario de oficina corporativa	48
5.9.	Propiedades del nodo MailServer	48
5.10.	Propiedades del nodo T1021	49
5.11.	Resumen del ataque presentado sobre el escenario de oficina . .	50
5.12.	Posibles estados iniciales disponibles en el sistema proporcionados por el comando trace	53
5.13.	Selección de estado inicial erróneo	53
5.14.	Ataque sobre el escenario <i>Smart Home</i>	54
5.15.	Propiedades del nodo T1056	54
5.16.	Propiedades del nodo Router	55
5.17.	Resumen del ataque presentado sobre el escenario <i>Smart Home</i>	55
5.18.	Ataque sobre el escenario <i>Planta Industrial</i>	56
5.19.	Propiedades del nodo PLC	56
5.20.	Propiedades del nodo T1552	57
5.21.	Identificación del CVE afectado en el nodo T1552	57
5.22.	Resumen del ataque presentado sobre el escenario <i>Planta Industrial</i>	58
5.23.	Limpieza exitosa de la base de datos	59
5.24.	Base de datos vacía tras la ejecución del comando clean	59
5.25.	Historial de ataques realizados	59

Índice de tablas

3.1. Requisitos de diseño	13
3.2. Etapas de diseño	15
4.1. Atributos de los nodos Activo	20
4.2. Posibles conexiones entre activos	21
4.3. Atributos de los nodos Técnica	33
4.4. Posibles conexiones de los nodos Técnica	33
B.1. Costes de mano de obra	68
B.2. Costes materiales	69
B.3. Costes indirectos	69
B.4. Costes totales	70

Capítulo 1

Introducción

En este capítulo se va a presentar el proyecto a fin de dar una primera impresión así como proporcionar una idea de lo que se va a ir detallando en capítulos posteriores.

1.1. Motivación

Desde hace tiempo, las organizaciones y los gobiernos se enfrentan constantemente a posibles ataques de seguridad [1]. Sin embargo, la necesidad de una ciberdefensa de próxima generación se ha vuelto aún más urgente en esta era en la que las superficies de ataque que los ciberdelincuentes pueden explotar han crecido a un ritmo alarmante. Este rápido crecimiento deriva del aumento de la complejidad en los sistemas empresariales y la adopción masiva de servicios en la nube y dispositivos conectados [2]. En este contexto, la ciberdefensa basada en el análisis predictivo resulta más proactiva que las tecnologías actuales, que dependen de la detección de intrusiones.

Este nuevo enfoque de ciberdefensa está transformando la manera en que las organizaciones comprenden sus vulnerabilidades y se preparan ante posibles ataques. El *Informe de Situación Ciberseguridad en España 2024* [3] destaca que la adopción de la inteligencia artificial se ha acelerado en el ámbito de la ciberseguridad, proporcionando herramientas de análisis predictivo y detección automatizada de amenazas. Ya no se trata solo de identificar vulnerabilidades aisladas o puntos débiles, ahora se trata de entender cómo los atacantes pueden encadenar estos puntos débiles para crear un camino de ataque crítico hacia sus activos más valiosos, como los administradores de dominio, bases de datos críticas, controladores de dominio de *Active Directory*, etc.

De este contexto surge este proyecto, con el que se pretende ofrecer una solución para la prevención de ciberataques mediante la visualización del camino de ataque o *attack path visualization*, que es una representación gráfica de los posibles caminos de ataque que un adversario podría seguir para comprometer un activo desde cualquier punto de entrada en el sistema objetivo. Por ejemplo,

una vez que un atacante ha logrado un acceso inicial al entorno organizacional, el camino de ataque muestra la posible ruta que el adversario puede tomar entre los activos para llegar a los elementos más valiosos, como los Controladores de Dominio.

1.2. Objetivos

El objetivo principal de este trabajo es el desarrollo de una herramienta que permita la simulación de cadenas de ataque formadas por tácticas de la matriz MITRE ATT&CK, así como la identificación y representación gráfica de su impacto sobre un escenario o infraestructura de red simulada y previamente definida.

Para lograr este propósito, se plantean los siguientes objetivos específicos:

- Análisis y preprocesamiento de los datos relacionados con las tácticas de la matriz MITRE ATT&CK, incluyendo la modelización del impacto en infraestructuras simuladas.
- Estudio y selección de algoritmos para la simulación de cadenas formadas por técnicas de ataque, considerando metodologías actuales de ciberseguridad y análisis de riesgos.
- Desarrollo de un sistema automatizado que permita la simulación de cadenas de ataque, la correlación de tácticas empleadas y la representación visual de su impacto en un entorno definido.
- Validación y evaluación del sistema desarrollado, con pruebas en escenarios controlados que permitan medir su efectividad para identificar, correlacionar y representar gráficamente las tácticas de ataque simuladas.

1.3. Estructura del documento

El presente documento se secciona en capítulos. A fin de tener una visión global de las distintas fases, se muestra a continuación cada uno de ellos junto a una breve descripción:

Capítulo 1. Es la introducción al TFM, donde se proporciona una visión global y se describe la motivación de este, así como los objetivos a conseguir.

Capítulo 2. Se presenta el marco teórico, resumiendo los distintos conceptos y tecnologías claves para el desarrollo del proyecto.

Capítulo 3. Diseño de la solución a implementar, basado en los requisitos identificados. Se presentan también las diferentes etapas a seguir para ello.

Capítulo 4. Se explica la metodología empleada para el desarrollo del proyecto, desde la preparación del entorno hasta alcanzar su funcionalidad completa.

1.3. ESTRUCTURA DEL DOCUMENTO

Capítulo 5. Resumen de los resultados obtenidos tras la fase de desarrollo reforzado con tests que lo prueban.

Capítulo 6. Conclusiones y problemas surgidos a raíz de la realización del proyecto. Líneas futuras sobre las que trabajar.

Capítulo 2

Marco teórico

En este capítulo se van a presentar los distintos conceptos y tecnologías claves aplicados en el proyecto, a fin de facilitar su correcto entendimiento.

2.1. MITRE ATT&CK

MITRE ATT&CK (*Adversarial Tactics, Techniques, and Common Knowledge*) [4] es una base de conocimientos universalmente accesible y continuamente actualizada para modelar, detectar, prevenir y combatir amenazas de ciberseguridad basándose en el comportamiento conocido de los cibercriminales adversarios. Creado por MITRE Corporation, este marco de referencia ofrece una visión estructurada de las tácticas y técnicas empleadas por los atacantes en diferentes fases de una intrusión. Su principal objetivo es proporcionar a organizaciones, analistas de seguridad y profesionales de IT (*Information Technology*) una herramienta para comprender, detectar y mitigar amenazas cibernéticas de manera más efectiva.

El marco se organiza en matrices, que clasifican tácticas y técnicas en función de su propósito y cómo se relacionan con las etapas de un ataque, desde la recopilación inicial de información y los comportamientos de planificación del atacante hasta la ejecución final del ataque. Cada matriz cubre una amplia gama de escenarios, pudiéndose distinguir así tres matrices principales [5]:

- **Matriz empresarial:** La matriz empresarial incluye todas las técnicas de los adversarios utilizadas en los ataques contra la infraestructura empresarial. Esta matriz incluye submatrices para las plataformas Windows, MacOS y Linux, así como para la infraestructura de red, las plataformas en la nube y las tecnologías de contenedores. También incluye una matriz PRE de técnicas preparatorias utilizadas antes de un ataque.
- **Matriz móvil:** Esta matriz engloba aquellas técnicas utilizadas en ataques directos a dispositivos móviles, así como en ataques móviles basados en la red que no requieren acceso a los dispositivos. Esta matriz incluye

2.1. MITRE ATT&CK

submatrices para las plataformas móviles iOS y Android.

- Matriz ICS:** La Matriz ICS incluye técnicas utilizadas en ataques a sistemas de control industrial, en concreto la maquinaria, los dispositivos, los sensores y las redes que se utilizan para controlar o automatizar las operaciones de fábricas, empresas de servicios públicos, sistemas de transporte y otros proveedores de servicios críticos.

Además de las tácticas y técnicas, el marco MITRE ATT&CK incluye otros elementos clave como las subtécnicas, las mitigaciones y las detecciones, que ofrecen un nivel de detalle más granular y recursos prácticos para fortalecer las defensas y responder eficazmente a las amenazas. A continuación, se presenta una breve descripción de los aspectos más relevantes de dichos elementos:

- Tácticas:** Representan las metas o propósitos generales que los atacantes buscan alcanzar durante cada fase del ataque, como obtener acceso inicial, moverse lateralmente por la red o exfiltrar datos.
- Técnicas:** Describen los métodos específicos que los atacantes emplean para lograr las tácticas, por ejemplo, el aprovechamiento de vulnerabilidades o la ejecución de scripts maliciosos. En la Figura 2.1 se muestran las diferentes técnicas de la matriz empresarial, agrupadas en columnas, siendo cada columna la táctica a la que pertenecen.
- Subtécnicas:** Proporcionan un nivel de detalle más granular dentro de las técnicas, explicando variaciones específicas de un método en particular.
- Mitigaciones:** Sugerencias sobre cómo prevenir o reducir el impacto de estas técnicas.
- Detecciones:** Indicadores y patrones que los equipos de seguridad pueden monitorizar para identificar posibles actividades maliciosas.

ATT&CK Matrix for Enterprise																	
Reconnaissance		Resource Development		Initial Access		Execution		Persistence		Privilege Escalation		Defense Evasion		Credential Access		Discovery	
10 techniques	8 techniques	10 techniques	14 techniques	20 techniques	14 techniques	20 techniques	14 techniques	20 techniques	14 techniques	20 techniques	14 techniques	20 techniques	14 techniques	20 techniques	14 techniques	20 techniques	14 techniques
Active Scanning (2)	Acquire Access	Content Injection	Cloud Administration Command	Account Manipulation (7)	Abuse Elevation Mechanism (8)	Abuse Elevation Control Mechanism (2)	Adversary-in-the-Middle (2)	Account Discovery (6)	Exploitation of Services	Adversary-in-the-Middle (2)	Application Protocol (5)	Automated Externalization (1)	Account Access Removal (1)	Automated Externalization (1)	Account Access Removal		
Gather Victim Host Information (4)	Acquire Infrastructure (2)	Compromise by Compromise	Command and Scripting Application	BITS Jobs	Access Token Manipulation (3)	Access Token Manipulation (1)	Adversary-in-the-Middle (2)	Adversary-in-the-Middle (2)	Attack Surface Reduction (1)	Attack Surface Reduction (1)	Attack Transfer Size Limit (1)	Attack Through Alternative Media (1)	Attack Through Alternative Media (1)	Attack Through Alternative Media (1)	Attack Through Alternative Media (1)	Attack Through Alternative Media (1)	
Gather Victim Identity Information (2)	Compromise Infrastructure (2)	Exploit Public-Facing Application	Container Administration	Container Administration	Container Manipulation (2)	BITS Jobs	Build Image on Host	Browser Information Disclosure (1)	Browser Information Disclosure (1)	Browser Information Disclosure (1)	Browser Session Hijacking (2)						
Gather Victim Network Information (3)	Develop Capabilities (2)	Establish Remote Services	Deploy Container	Boot or Logon Autostart Execution (1)	Boot or Logon Autostart Execution (1)	Boot or Logon Autostart Execution (1)	Debugger Evasion	Cloud Infrastructure Discovery	Cloud Service Dashboard	Cloud Service Dashboard	Cloud Storage Object Discovery						
Gather Victim Org Information (2)	Establish Accounts (2)	Exploit Client Execution	Exploit Container	Browser Extensions	Browser Extensions	Browser Extensions	Deobfuscate/Decode Files or Information	Cloud Service Dashboard	Cloud Service Dashboard	Cloud Service Dashboard	Cloud Storage Object Discovery						
Phishing for Information (4)	Phishing (2)	Phishing (2)	Phishing (2)	Compromise Host Software	Compromise Host Software	Create or Modify System Configuration (2)	Direct Volume Access	Cloud Storage Object Discovery									
Search Closed Sources (2)	Replication Through Removable Media (2)	Replication Through Removable Media (2)	Scheduled Task/Job (2)	Native API	Create Account (1)	Domain or Tenant Policy Modification (2)	Domain or Tenant Policy Modification (2)	Container and Resource Discovery									
Search Open Technical Databases (2)	Replication Through Removable Media (2)	Replication Through Removable Media (2)	Serverless Execution	Scheduled Task/Job (2)	Create or Modify System Configuration (2)	Event Triggered Execution (1)	Execution Guardrails (2)	Device Driver Discovery									
Search Open Websites/Domains (2)	Replication Through Removable Media (2)	Replication Through Removable Media (2)	Software Deployment Tools	Event Triggered Execution (1)	Event Triggered Execution (1)	Event Triggered Execution (1)	Exploitation for Defense Evasion	Device Driver Discovery	Domain Trust Discovery	Domain Trust Discovery	Domain Trust Discovery	Domain Trust Discovery	Domain Trust Discovery	Domain Trust Discovery	Domain Trust Discovery	Domain Trust Discovery	
Search Open Websites (2)	Replication Through Removable Media (2)	Replication Through Removable Media (2)	External Services	External Services	External Services	External Services	File and Directory Permissions Modification (2)										
Search Victim-Owned Websites	Replication Through Removable Media (2)	Replication Through Removable Media (2)	System Services (2)	Hijack Execution Flow (18)	Hijack Execution Flow (18)	Hijack Execution Flow (18)	Hijack Execution Flow (18)	Hijack Execution Flow (18)	Hijack Execution Flow (18)	Hijack Execution Flow (18)	Hijack Execution Flow (18)	Hijack Execution Flow (18)	Hijack Execution Flow (18)	Hijack Execution Flow (18)	Hijack Execution Flow (18)	Hijack Execution Flow (18)	
			User Execution (2)	Impersonate Internal Image	Impersonate Internal Image	Impersonate Internal Image	Impersonation										
			Windows Management Instrumentation (2)	Process Injection (17)	Process Injection (17)	Process Injection (17)	Process Injection (17)	DS Credential Indicator Removal (10)									

Figura 2.1: Extracto de la matriz empresarial de MITRE ATT&CK [6]

2.2. CVEs, CWEs y CAPECs

Los *Common Vulnerabilities and Exposures* (CVE) y los *Common Weakness Enumeration* (CWE) son sistemas de clasificación utilizados en el ámbito de la ciberseguridad para identificar y gestionar vulnerabilidades y debilidades en software y hardware.

Los CVE [7] son una lista pública de fallos de seguridad informática conocidos, donde cada vulnerabilidad recibe un identificador único (CVE-ID), una descripción detallada y referencias a soluciones o mitigaciones disponibles. Cada CVE-ID sigue el formato CVE-año-número, por ejemplo, CVE-2024-12345.

Por otro lado, los CWE [8] son una categorización de debilidades comunes en el diseño o implementación del software que podrían derivar en vulnerabilidades. Cada debilidad documentada recibe un identificador CWE único y una descripción que ayuda a los desarrolladores y profesionales de la seguridad a entender y prevenir errores comunes en el desarrollo de software.

Finalmente, los *Common Attack Pattern Enumeration and Classification* (CAPEC) [9] complementan estos sistemas al enfocarse en los patrones de ataque utilizados por los cibercriminales para explotar debilidades y vulnerabilidades. CAPEC proporciona una base de datos estructurada de métodos de ataque conocidos, lo que ayuda a los profesionales de seguridad a anticipar y mitigar amenazas. Este marco se relaciona con las tácticas y técnicas de MITRE ATT&CK, ya que muchos de los patrones de ataque documentados en CAPEC se corresponden con las técnicas específicas que los actores maliciosos emplean en campañas del mundo real.

Tal y como se resume en la Figura 2.2, los CVE se centran en vulnerabilidades específicas ya identificadas en productos o sistemas particulares, los CWE se enfocan en las debilidades subyacentes que pueden conducir a dichas vulnerabilidades, y los CAPEC describen los patrones de ataque utilizados para explotarlas.

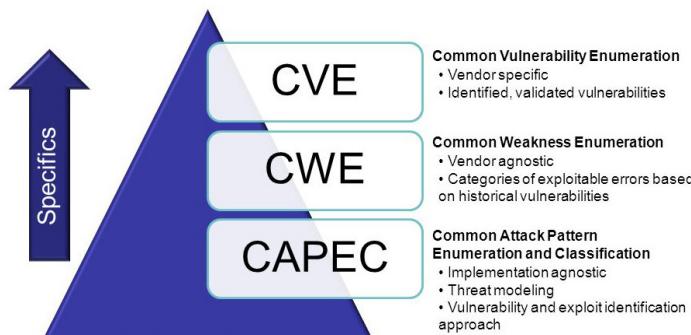


Figura 2.2: Comparativa entre los conceptos de CVE, CWE y CAPEC [10]

2.3. Procesos estocásticos

Un proceso estocástico [11] es un conjunto de variables aleatorias indexadas por el tiempo u otro parámetro, que se utiliza para modelar fenómenos que evolucionan de manera incierta o aleatoria a lo largo del tiempo. A diferencia de un sistema determinista, en el cual el resultado es completamente predecible dado un conjunto de condiciones iniciales, en un proceso estocástico el resultado depende del azar.

De entre los diversos tipos de procesos estocásticos que existen, cabe destacar los siguientes:

- **Proceso de Poisson:** Modela la ocurrencia de eventos aleatorios en el tiempo, como las llamadas en una central telefónica.
- **Proceso de Wiener:** Modela fluctuaciones continuas, como el precio de las acciones.
- **Cadenas de Markov:** Procesos donde el futuro depende únicamente del estado actual, y no del camino seguido para llegar a él.

Una distinción fundamental en los procesos estocásticos es si son discretos o continuos, tanto en el tiempo como en el espacio de estados. Un proceso estocástico discreto se caracteriza por tener una evolución en pasos determinados, como en los instantes $t = 0, 1, 2, \dots$, y usualmente toma valores dentro de un conjunto finito o contable. Un ejemplo típico de este tipo es la cadena de Markov, en la que un sistema cambia de estado en momentos específicos y la probabilidad de transición depende únicamente del estado actual.

Por otro lado, un proceso estocástico continuo permite que el tiempo y, en muchos casos, el espacio de estados, tomen cualquier valor dentro de un intervalo. Esto significa que el proceso puede cambiar en cualquier instante, y las posibles salidas pueden formar un conjunto continuo. Un ejemplo representativo es el movimiento browniano o proceso de Wiener, ampliamente utilizado en física para describir el movimiento aleatorio de partículas, y en finanzas para modelar la evolución de los precios de los activos.

2.3.1. Cadenas de Markov

Una cadena de Markov [12] es un modelo matemático utilizado para describir un sistema en el que los estados cambian de manera secuencial, donde la probabilidad de transición de un estado a otro depende únicamente del estado actual, y no de cómo se llegó a él. Este principio se conoce como la propiedad de Markov.

Las cadenas de Markov son útiles para modelar y analizar procesos estocásticos (procesos que involucran incertidumbre) en los que el resultado futuro depende solo del estado presente. Esto permite hacer predicciones, entender patrones de comportamiento y simular escenarios en una variedad de contextos. De hecho,

estos modelos son ampliamente utilizados en campos como la economía, la biología, la inteligencia artificial y la ciberseguridad, entre otros, a fin de analizar sistemas dinámicos mediante herramientas de probabilidad y estadística.

En una cadena de Markov, se distinguen los principales elementos:

- **Estados:** Representan las posibles condiciones o configuraciones del sistema. Por ejemplo, en un modelo del clima, los estados podrían ser soleado, lluvioso o nublado. Un estado se dice absorbente si es imposible abandonar dicho estado. Un ejemplo sería la muerte en un modelo de esperanza de vida.
- **Transiciones:** Son los movimientos entre estados, descritos por probabilidades. En la Figura 2.3 se pueden apreciar 4 estados junto a las posibilidades de transición entre ellos, siendo el estado 3 un estado absorbente.
- **Matriz de transición:** Indica la probabilidad de pasar de un estado a otro. Cada fila de la matriz corresponde a un estado actual, y cada columna a un estado futuro.
- **Estado inicial:** Estado que posee el sistema en el instante considerado como origen del tiempo.

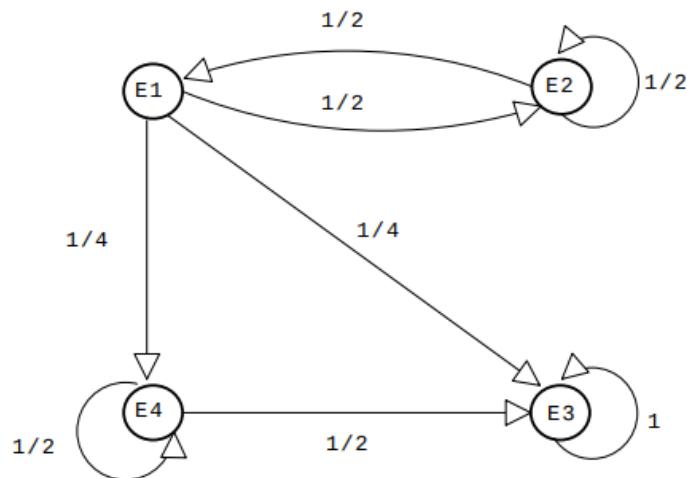


Figura 2.3: Ejemplo de cadena de Markov [13]

2.4. Rutas de ataque (*Attack Paths*)

Una ruta de ataque [14] es una representación visual del camino que un atacante podría seguir para explotar debilidades en un sistema y comprometer activos críticos, como bases de datos sensibles o cuentas de administrador de dominio.

Este concepto, clave en la ciberseguridad, no solamente permite analizar vulnerabilidades aisladas, sino también cómo estas se interconectan para formar un trayecto explotable. Su objetivo principal es ayudar a las organizaciones a identificar, priorizar y mitigar riesgos, abordando las amenazas desde la perspectiva del atacante. En lugar de centrarse únicamente en solucionar vulnerabilidades individuales, el análisis de rutas de ataque proporciona un enfoque estratégico para visualizar el contexto completo de los riesgos, lo que resulta fundamental para proteger activos esenciales.

Para comprender mejor esta idea, es importante distinguirla de otros términos que, pese a estar muy relacionados, no son exactamente iguales, como por ejemplo un vector de ataque [15] (el método específico que un atacante utiliza para explotar una vulnerabilidad) o una superficie de ataque [16] (el conjunto total de puntos vulnerables en una infraestructura). A diferencia de estos, una ruta de ataque sería más bien la cadena de vulnerabilidades interconectadas que un atacante podría explotar para alcanzar un objetivo.

El análisis de rutas de ataque es crucial para identificar y proteger los puntos más críticos en un entorno. Este enfoque permite a las organizaciones anticiparse a los movimientos de los atacantes, modelando cómo podrían explorar la red, explotar credenciales robadas o configuraciones incorrectas, y aprovechar vulnerabilidades interconectadas. De esta forma, en lugar de reaccionar ante incidentes de seguridad una vez que ocurren, este análisis facilita la proactividad, ya que permite rastrear posibles trayectorias antes de que sean utilizadas por adversarios.



Figura 2.4: Ejemplo de ruta de ataque en *Active Directory* [17]

2.5. Bases de datos orientadas a grafos

Una ruta de ataque pierde gran parte de su potencial si no se visualiza correctamente. La visualización de rutas de ataque [18] es una herramienta muy poderosa ya que, mediante el uso de grafos, es posible mapear los caminos que un atacante podría seguir dentro de una red, destacando conexiones entre activos, configuraciones erróneas y vulnerabilidades.

Esta representación visual ayuda a priorizar recursos al identificar los trayectos que conducen a activos críticos, como controladores de dominio o bases de datos. De esta forma, se transforma la información sobre los riesgos en una estrategia clara de mitigación.

Para mapear estas rutas de manera efectiva, se utilizan bases de datos de grafos y algoritmos avanzados. Estas bases de datos permiten representar de manera visual y estructurada la interconexión entre activos, vulnerabilidades y posibles movimientos de un atacante dentro de una red. Los elementos de estas bases de datos que hacen posible el mapeo son:

- **Nodos o vértices:** Representan los elementos individuales de una red o sistema. En el contexto de ciberseguridad, un nodo puede ser un servidor, un equipo de usuario, una base de datos...
- **Aristas o relaciones:** Representan las relaciones o conexiones entre los nodos. En un modelo de ciberataque, las aristas pueden describir por ejemplo vectores de ataque o movimientos laterales.

A diferencia de las bases de datos tradicionales, que organizan la información en tablas y requieren consultas complejas para analizar relaciones entre datos, las bases de datos de grafos están diseñadas para manejar interconexiones de manera eficiente [19]. Este aspecto las hace especialmente útiles para modelar escenarios de ataque, ya que permiten identificar caminos críticos con mayor rapidez y precisión.

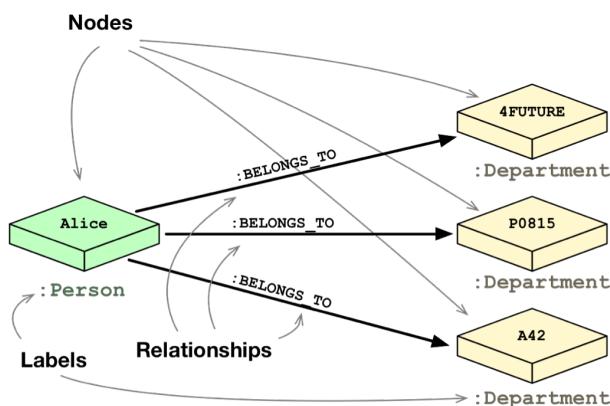


Figura 2.5: Relaciones en una base de datos orientada a grafos [19]

2.6. Tecnologías

Las diferentes herramientas y *frameworks* utilizados en el proyecto se describen en esta sección.

2.6.1. Neo4j

Neo4j [20] es un software *open source* de base de datos orientadas a grafos, desarrollado por *Neo Technology Inc* e implementado en Java.

Tal y como se ha comentado en la sección 2.5, estas bases de datos permiten el almacenamiento datos estructurados como grafos en lugar de tablas, lo que implica que la información se organiza como un grafo dirigido, compuesto por nodos y relaciones que los conectan.

Neo4j es compatible con los principales sistemas operativos, como Windows, Linux y macOS. Está disponible en dos versiones: una gratuita y otra de pago. La versión de pago incluye características avanzadas, como replicación, monitorización y alta disponibilidad.

Dos de las funcionalidades más atractivas de Neo4j, que de hecho constituyen la base de este trabajo, son el lenguaje *Cypher* y el etiquetado de los nodos.

Cypher es el lenguaje de consulta diseñado específicamente para trabajar con grafos en Neo4j. Utiliza una sintaxis inspirada en SQL pero optimizada para manejar nodos y relaciones. De esta forma, es posible la realización de consultas y operaciones complejas sobre los grafos.

Por otro lado, Neo4j permite el etiquetado de los nodos, que son uno de los elementos principales de una base de datos orientada a grafos, tal y como se mencionó en la sección anterior. Estas etiquetas son una forma de clasificarlos, actuando como categorías o tipos que agrupan nodos similares para facilitar su identificación y consulta.

El uso combinado de estas capacidades permite, por tanto, una organización clara de los datos con flexibilidad tanto para modelar escenarios complejos como para la realización de consultas específicas y escalables.



Figura 2.6: Logo de Neo4j [20]

2.6.2. *Cypher*

Cypher es el lenguaje de consulta utilizado en Neo4j, diseñado para trabajar con bases de datos orientadas a grafos. Su sintaxis declarativa facilita la representación y análisis de relaciones complejas, lo que lo hace ideal para aplicaciones de ciberseguridad.

En este contexto, *Cypher* permite modelar redes de dispositivos, conexiones y eventos sospechosos, facilitando la detección de patrones de ataque, el análisis de amenazas y la identificación de actores maliciosos. Algunos ejemplos de consultas básicas son los siguientes:

- Crear nodos con etiqueta y propiedades:

```
CREATE (nodo:Etiqueta {propiedad1:"prop1", propiedad2:"prop2"})
```

- Crear relaciones entre dos nodos existentes:

```
MATCH (nodo1), (nodo2)
CREATE (nodo1)-[:REL_TYPE]->(nodo2)
```

- Buscar relaciones entre nodos filtrando por etiquetas y propiedades:

```
MATCH (nodo1:Etiqueta1)-[r:REL_TYPE]->(nodo2:Etiqueta2)
WHERE nodo1.propiedad1 = "valor1" AND nodo2.propiedad2 = "valor2"
RETURN nodo1, r, nodo2
```

2.7. Librerías de Python

El lenguaje de programación Python cuenta con una serie de módulos o librerías que permiten trabajar con bases de datos orientadas a grafos, así como con la información extraída de estas.

2.7.1. Neo4j

La librería neo4j para Python es el conector oficial que permite interactuar con una base de datos Neo4j desde aplicaciones Python. Este módulo facilita la ejecución de consultas *Cypher*, la manipulación de datos en la base de grafos, y la integración con otras herramientas.

2.7.2. Rich

Rich [21] es una librería de Python para la creación de interfaces de consola interactivas y visuales. Permite la creación de salidas de texto enriquecido, como tablas, gráficos y representaciones coloridas, lo que es útil para mostrar la información de manera atractiva y fácil de interpretar.

Capítulo 3

Diseño y propuesta

En este capítulo se van a detallar las tecnologías a utilizar, seleccionadas a partir de un análisis de los requisitos a cumplir por el sistema. Posteriormente, se definirán las etapas en las que se va a desarrollar el sistema.

3.1. Requisitos de diseño

En primer lugar, se han identificado los siguientes requisitos a cumplir por el sistema:

Id	Requisito
RD1	El sistema debe garantizar un entorno seguro, de forma que la simulación de ataques no genere riesgos reales en infraestructuras o redes externas.
RD2	Ha de ser escalable, permitiendo la incorporación de nuevos escenarios de red y técnicas de ataque sin comprometer el rendimiento o la estabilidad del sistema.
RD3	La parametrización de escenarios y ataques debe ser flexible, permitiendo la actualización de datos y la adaptación a nuevas amenazas sin necesidad de modificar la estructura del sistema.
RD4	Debe contar con tiempos de ejecución optimizados y un uso eficiente de los recursos, asegurando un procesamiento fluido incluso en simulaciones de gran escala.
RD5	La interfaz de usuario debe ser clara e interactiva, proporcionando una experiencia intuitiva y sencilla para la visualización y análisis de las simulaciones.

Tabla 3.1: Requisitos de diseño

3.2. Solución propuesta

Una vez definidos los requisitos a satisfacer, ya se puede realizar una propuesta de sistema que se ajuste a ellos, especificando las tecnologías y conceptos encargados de cada tarea. Toda esta información es la que se va a proporcionar en este apartado.

Para la implementación del sistema, se ha optado por desarrollar una interfaz de usuario basada en línea de comandos (CLI) en Python, la cual permitirá una interacción fluida con el usuario sin necesidad de una interfaz gráfica compleja.

La información relativa a Tácticas, Técnicas y Procedimientos (TTPs) se obtendrá de la matriz empresarial de MITRE ATT&CK, y se gestionará junto a los escenarios de red mediante diferentes ficheros (`.json`, `.csv` y `.txt`), proporcionando un formato flexible y fácilmente modificable para la introducción y actualización de datos.

Además, se utilizará una base de datos de grafos Neo4j para representar tanto los escenarios como la propagación de los *attack paths* simulados sobre ellos, a la que se accederá a través de su librería oficial para Python.

Todo esto permitirá al sistema simular rutas de ataque mediante cadenas de Márkov y realizar las operaciones correspondientes al mapeo entre las TTPs y los activos de los escenarios sobre la base de datos.

Finalmente, para la representación visual de los resultados en la terminal, se empleará la librería Rich, que permitirá mostrar de manera clara y estructurada métricas en formato *dashboard*, además de un historial de ataques simulados.

A continuación se muestra un diagrama que resume brevemente la arquitectura del sistema:

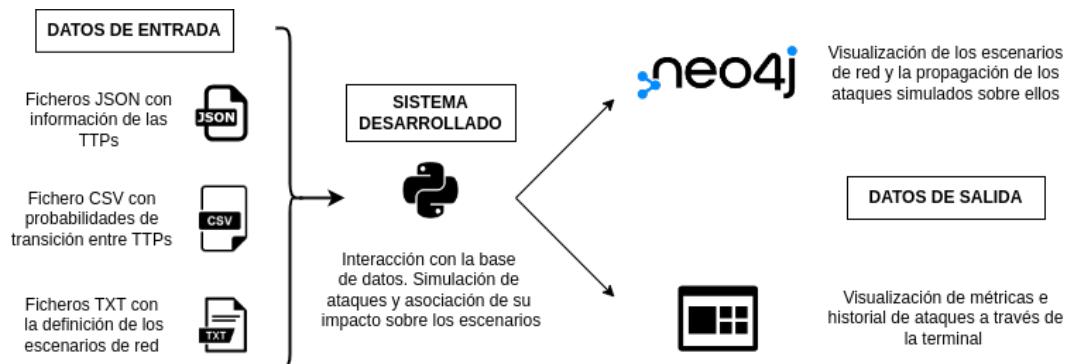


Figura 3.1: Diagrama básico de la arquitectura del sistema

3.3. Etapas de diseño

Una vez propuesta la solución, se establecen una serie de etapas de diseño en las que se divide el proceso, siguiendo un orden secuencial que se muestra a continuación:

Id	Etapa de diseño
ED1	Análisis, extracción y preprocesado de la información relativa a las TTPs de MITRE ATT&CK
ED2	Definición de un breve catálogo de escenarios de red
ED3	Implementación de una cadena de Markov en Python para la generación de secuencias de ataque
ED4	Automatización de operaciones con la base de datos, incluyendo la carga de los escenarios y la asociación entre las TTPs de los ataques y los activos a los que afectan.
ED5	Creación de dos interfaces gráficas: un <i>dashboard</i> que muestre las estadísticas principales de un ataque tras su ejecución y un historial de todos los ataques simulados

Tabla 3.2: Etapas de diseño

Como resultado de este proceso, se obtendrá una herramienta que permitirá la simulación de ciberataques basados en la matriz MITRE ATT&CK y la visualización de su impacto sobre escenarios de red previamente definidos.

Capítulo 4

Desarrollo

En este capítulo se va a detallar el desarrollo del sistema, siguiendo las etapas especificadas en el capítulo anterior. La herramienta desarrollada ha recibido el nombre **ARGOS** (*Attack Route Graph Observation System*).

Antes de comenzar, y para tener una visión global de los apartados que se van a ir cubriendo, se adjunta la arquitectura final de ARGOS. La arquitectura descrita en la Figura 4.1 cuenta con un mayor nivel de detalle en comparación a la que se presentó previamente en la sección 3.2, cuyo objetivo era la justificación de las tecnologías utilizadas y no tanto el funcionamiento a más bajo nivel.

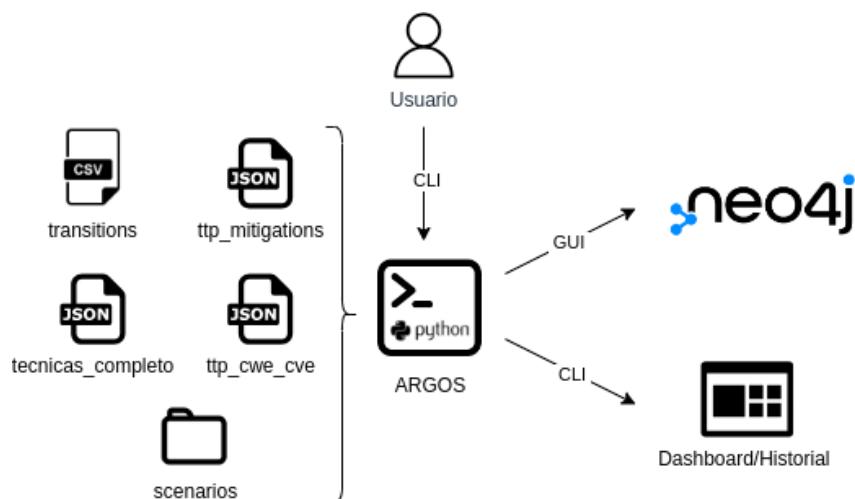


Figura 4.1: Diagrama detallado de la arquitectura del sistema

Recapitulando su funcionamiento, al sistema se le proporcionará la información necesaria acerca de TTPs y escenarios mediante ficheros externos. Esta información permitirá que, mediante la ejecución de comandos, el usuario cargue escenarios de red y simule ataques sobre ellos.

4.1. OBTENCIÓN DE LOS DATOS DE ENTRADA

Finalmente, se podrán visualizar por un lado los *attack paths* a través la interfaz gráfica de Neo4j y por otro lado un panel informativo y un historial de ataques a través de la terminal, ejecutando los comandos correspondientes.

En base a este funcionamiento, se ha estructurado este capítulo en tres secciones principales. En la primera de ellas se explicarán los ficheros externos que contienen la información necesaria para el funcionamiento de la herramienta. En la segunda se especificarán los requisitos necesarios para el funcionamiento del sistema, así como la preparación del entorno necesario para su ejecución. Por último, en la tercera sección se detallará la lógica de la herramienta, así como los *outputs* que genera.

4.1. Obtención de los datos de entrada

En esta primera sección se van a describir los procesos de generación y obtención de la información necesaria para el funcionamiento de la herramienta. Esta información se puede dividir a su vez en dos fuentes, por un lado está la información relativa a las TTPs de la matriz MITRE ATT&CK y por otro los escenarios de red. En los siguientes apartados se describen ambas fuentes.

4.1.1. Información sobre TTPs

ARGOS se alimenta de una serie de ficheros que le permiten obtener información y contexto de las TTPs. Estos ficheros se pueden generar y actualizar periódicamente ya que están construidos a partir de diferentes fuentes de información estructurada, como archivos XML, JSON y hojas de cálculo en Excel. Además, para garantizar que el sistema disponga de información actualizada y precisa sobre las técnicas de ataque, los datos a partir de los cuales son construidos dichos ficheros son extraídos de fuentes públicas especializadas, como MITRE ATT&CK. Estos ficheros se describen a continuación, detallando tanto la información que contienen como su proceso de obtención:

- **cwe_cve.json:** contiene la asociación entre cada técnica y los CVE y CWES que compromete. Para su generación, en primer lugar se descarga el archivo XML `cwec_v4.15.xml` de la base de datos de MITRE [22]. Acto seguido, el script `cwe.py` (previamente desarrollado) procesa este XML, recorriendo cada CWE y extrayendo los CVEs y CAPEC asociados. La información se estructura en un formato adecuado y se almacena en el fichero JSON.
- **ttp_mitigations.json:** proporciona información sobre las mitigaciones existentes para las distintas técnicas de ataque. Su generación se basa en el archivo `enterprise-attack-v15.1-techniques.xlsx`, que contiene un dataset que representa la matriz empresarial de MITRE ATT&CK en formato Excel y se descarga también de la base de datos de MITRE [23]. Este archivo es procesado por el script `mit_ttp.py`, que extrae tres columnas clave (ID de la mitigación, nombre de la mitigación y técnicas

sobre las que actúa) para posteriormente eliminar registros duplicados y se agrupan los datos, de manera que cada mitigación tenga una única entrada con la lista de técnicas afectadas.

- ***tecnicas_completo.json*:** contiene una descripción detallada de cada técnica de ataque. Se genera a partir del mismo archivo Excel utilizado en el caso anterior (*enterprise-attack-v15.1-techniques.xlsx*), procesado esta vez con el script *tech_assets.py*. Este script extrae en formato JSON diversas características de cada técnica, incluyendo: identificador y nombre, descripción, tácticas a las que está asociada, métodos de detección, plataformas afectadas, fuentes de datos relevantes, defensas que debe superar, permisos y requisitos del sistema.
- ***transitions.csv*:** proporciona las probabilidades de transición entre pares de técnicas. Este proceso es el más complejo, ya que implica la ejecución de varios scripts antes de llegar al fichero final. Se desarrolla en los siguientes pasos:
 1. Obtención de la información de los grupos: Se ejecuta *get_groups.py*, que realiza peticiones a las URLs de todos los grupos disponibles en la página de MITRE. La información extraída de cada grupo se almacena en el fichero *grupos.json*.
 2. Extracción de técnicas utilizadas por los grupos: Se ejecuta el script *extract_techniques.py*, que carga el fichero *grupos.json* y extrae el identificador, el nombre de cada grupo y la lista de técnicas que utiliza. El resultado se almacena en *grupos_tecnicas.json*.
 3. Cálculo de relaciones entre técnicas: El script *calculos.py* toma *grupos_tecnicas.json* y analiza la secuencia de técnicas utilizadas por cada grupo. Se extraen las técnicas en pares consecutivos (por ejemplo: TTP0 → TTP1, TTP1 → TTP2, TTP2 → TTP3, etc.). Se construye un diccionario donde se cuenta cuántas veces aparece cada par de técnicas consecutivas, así como la frecuencia individual de cada técnica. A partir de estos datos, se calculan porcentajes, es decir, la frecuencia con la que una técnica es seguida por otra en comparación con el número total de apariciones de la primera técnica en la secuencia. Finalmente, utilizando estos cálculos se construye un fichero CSV con tres columnas: técnica origen, técnica destino y probabilidad de transición.

4.1.2. Modelo de datos y escenarios de red

El segundo de los *inputs* que recibe el sistema es la definición de los escenarios de red sobre los que se van a realizar las simulaciones. La definición de cada escenario se especifica mediante un fichero de texto que contiene las instrucciones *Cypher* necesarias para la creación de los nodos y relaciones que lo componen en la base de datos. Estos ficheros se han incluido en el Apéndice C.

En primer lugar, se va a desarrollar el modelo de datos utilizado para definir los componentes de los escenarios. Una vez explicado el modelo, se van a describir tres escenarios básicos, que son los que se emplearán para las pruebas y validación del sistema.

Como ya se detalló en la sección 2.5, las bases de datos orientadas a grafos permiten representar la información mediante vértices o nodos y aristas o relaciones, lo que resulta especialmente útil para modelar redes informáticas y sus relaciones. De esta forma, cada componente del sistema, como usuarios, servidores o dispositivos de red, se representa como un nodo, mientras que las interacciones entre ellos, como conexiones, autenticaciones o dependencias, se modelan a través de relaciones dirigidas.

El modelo de datos desarrollado en este trabajo está diseñado para ser escalable y flexible, permitiendo representar distintos escenarios de red bajo una lógica común. Para ello, se ha definido una estructura basada en dos entidades principales: **Usuario** y **Activo**. La entidad **Activo** agrupa además todos los elementos de la red, y diferentes subtipos de activos, permitiendo clasificar cada componente según su función dentro del sistema. Se han definido también las diversas relaciones que modelan las conexiones y dependencias entre entidades, facilitando así el análisis de rutas de ataque y la evaluación de riesgos. A continuación se profundiza en cada uno de estos elementos.

Nodos

Se distinguen dos tipos principales de nodos, el primero de ellos son los nodos con etiqueta **Usuario**, que hacen referencia a las cuentas personales o de administración que se autentican en los elementos de la red. El resto de elementos de la red, es decir, los elementos informáticos, se modelan bajo la etiqueta **Activo**, a la que se añade una de las siguientes etiquetas para asignarle una subcategoría según su función en la infraestructura:

- **Servidor**: Dispositivos que ejecutan servicios accesibles.
- **DispositivoRed**: Elementos de red como equipos de usuario y otros sistemas conectados.
- **DispositivoSeguridad**: Elementos de seguridad como *firewalls*, VPN, IDS, EDR, etc.
- **Aplicación**: Software instalado en servidores o estaciones de trabajo.

Neo4j permite, además, agregar atributos a los nodos. Los atributos permiten describir las características de cada nodo. Se almacenan en formato clave-valor, proporcionando información detallada y particularizada para cada activo.

Hay una serie de atributos que son comunes a todos los activos, independientemente de su subcategoría. La información que proporcionan estos atributos sobre los activos, se corresponde con la información que explota cada técnica (obtenida en el apartado anterior). De esta forma, es posible realizar una asociación entre las técnicas y los activos a los que afectan de forma sencilla.

En la Tabla 4.1 se muestra una descripción de estos parámetros:

Atributo	Tipo	Descripción
<code>name</code>	String	Nombre del activo
<code>ip</code>	String	Dirección IP asignada
<code>platform</code>	String	Sistema Operativo
<code>permissions</code>	List	Nivel de permisos necesario para su acceso
<code>capabilities</code>	List	Capacidades del activo
<code>cve</code>	List	Identificadores de vulnerabilidades CVE asociadas
<code>cwe</code>	List	Categorías de debilidades de seguridad CWE aplicables

Tabla 4.1: Atributos de los nodos **Activo**

Los valores de los atributos `name` e `ip` son de libre elección. Para los atributos `cve` y `cwe` sería conveniente elegir los que más se ajusten de la lista disponible en el fichero `ttp_cwe_cve.json`. Finalmente, se muestran los posibles valores asignables para el resto de parámetros:

- **platform:** *Azure AD, Containers, Google Workspace, IaaS, Linux, Network, Office 365, PRE, SaaS, Windows, macOS*
- **permissions:** *Administrator, User, SYSTEM, root*
- **capabilities:**
 - *Access to shared folders and content with write permissions*
 - *Active remote service accepting connections and valid credentials*
 - *Autorun enabled or vulnerability present that allows for code execution*
 - *Kerberos authentication enabled*
 - *Microsoft Core XML Services (MSXML) or access to wmic.exe*
 - *Network interface access and packet capture driver*

4.1. OBTENCIÓN DE LOS DATOS DE ENTRADA

- *Presence of physical medium or device*
- *Privileges to access certain files and directories*
- *Privileges to access network shared drive*
- *Privileges to access removable media drive and files*
- *Remote exploitation for execution requires a remotely accessible service reachable over the network or other vector of access such as spearphishing or drive-by compromise*
- *Removable media allowed*
- *Unpatched software or otherwise vulnerable target*
- *Depending on the target and goal, the system and exploitable service may need to be remotely accessible from the internal network*
- *Permissions to access directories, files, and API endpoints that store information of interest*

Aristas

Las relaciones describen cómo los distintos activos interactúan entre sí, permitiendo modelar conexiones de red, autenticaciones, protecciones de seguridad y dependencias entre servicios. Cada relación tiene atributos que proporcionan información adicional, como el protocolo de comunicación, los puertos utilizados o las reglas de filtrado aplicadas. En la Tabla 4.2 se detallan las relaciones definidas en el modelo, explicando su significado, los nodos que conectan y los atributos asociados a cada una de ellas.

Relación	Descripción	Atributos
CONEXIÓN	Un activo se comunica con otro a través de un puerto	puerto, protocolo
PROTECCIÓN	Un activo está resguardado por un dispositivo de seguridad	tipo
AUTENTICACIÓN	Un usuario necesita autenticarse en un activo	usuario, contraseña
ALOJAMIENTO	Una aplicación o servicio está instalado en un servidor	versión

Tabla 4.2: Posibles conexiones entre activos

Con esta información, queda definido un modelo de datos en el que apoyarse para la definición de los escenarios de red y la futura evaluación del impacto de técnicas de ataque sobre ellos. En la Figura 4.2 se muestra el modelo definido para los escenarios de red en forma de diagrama, donde se muestran los tipos de nodos y las posibles relaciones entre ellos:

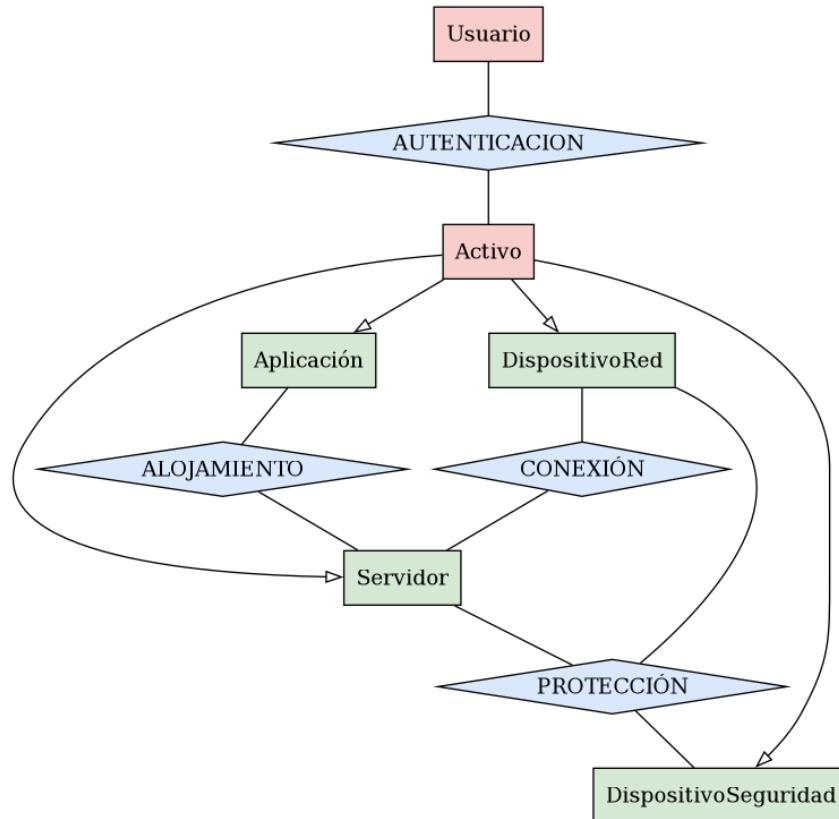


Figura 4.2: Diagrama del modelo de datos empleado en los escenarios de red

Escenarios de red

Una vez definido el modelo de datos, es posible aplicarlo a distintos escenarios para demostrar su utilidad en la representación y análisis de infraestructuras de red, permitiendo la identificación de posibles vulnerabilidades, la evaluación de riesgos y la optimización de estrategias de ciberseguridad. La flexibilidad del modelo facilita además la incorporación de nuevos elementos y la simulación de distintos entornos.

A continuación, se presentan varios ejemplos de escenarios modelados con esta estructura. Este breve catálogo de escenarios servirá como base para la realización de pruebas en el sistema. En el Apéndice C se adjuntan las instrucciones necesarias para la implementación del escenario en Neo4j, donde se pueden apreciar los atributos y relaciones asignadas a los activos:

1. Oficina corporativa

Este escenario representa una infraestructura típica de oficina, donde los usuarios necesitan acceder a servicios internos como almacenamiento de archivos y correo electrónico. La red cuenta con un firewall que protege los servidores, y una aplicación web alojada en uno de ellos. Este modelo permite analizar la seguridad de las autenticaciones, las conexiones entre dispositivos y las posibles vulnerabilidades presentes en los sistemas.

4.1. OBTENCIÓN DE LOS DATOS DE ENTRADA

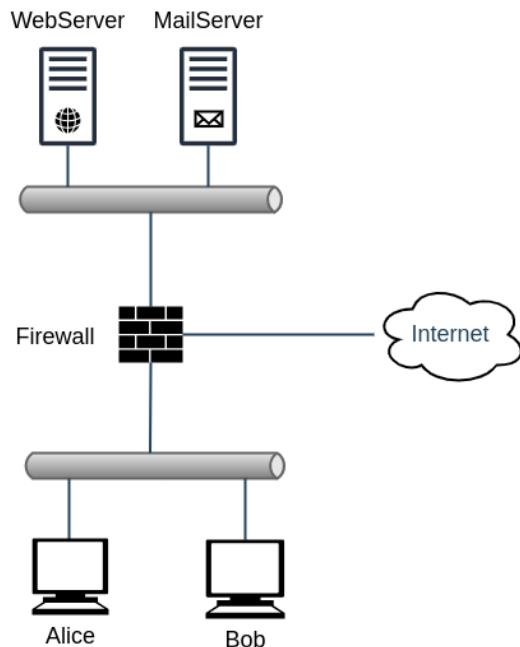


Figura 4.3: Escenario de red: Oficina Corporativa

El ordenador personal de Alice utiliza Windows como sistema operativo, y contiene la vulnerabilidad CVE-2004-2227, que afecta al navegador web *Mozilla Firefox*. El usuario Bob cuenta con Linux en su equipo, donde dispone de permisos de administrador. Tiene instalado el software *Digitaldesign CMS*. El CVE-2009-3597 afecta a *Digitaldesign CMS* en su versión v0.1, que guarda información sensible en el directorio web raíz con insuficiente control de acceso, lo que permite a los atacantes remotos descargar el fichero de la base de datos a través de una petición directa a `autoconfig.dd`.

En cuanto a los servidores, ambos cuentan con permisos de administración. El servidor de correo utiliza Linux, mientras que el servidor web corre un sistema operativo Windows. Este último, además, aloja un Apache.

2. *Smart home*

Este escenario refleja un entorno doméstico inteligente, en el que múltiples dispositivos conectados interactúan a través de una red centralizada.

El router actúa como un dispositivo de seguridad que filtra el tráfico y gestiona la conectividad. Es un router WiFi Zyxel, y cuenta con una vulnerabilidad (CVE-2021-35033) que en versiones específicas del firmware en las que se disponga de administración de contraseñas preconfigurada podría permitir a un atacante obtener acceso root del dispositivo.

Uno de los dispositivos conectados es una impresora WiFi, que recibe documentos desde computadoras y smartphones utilizando protocolos

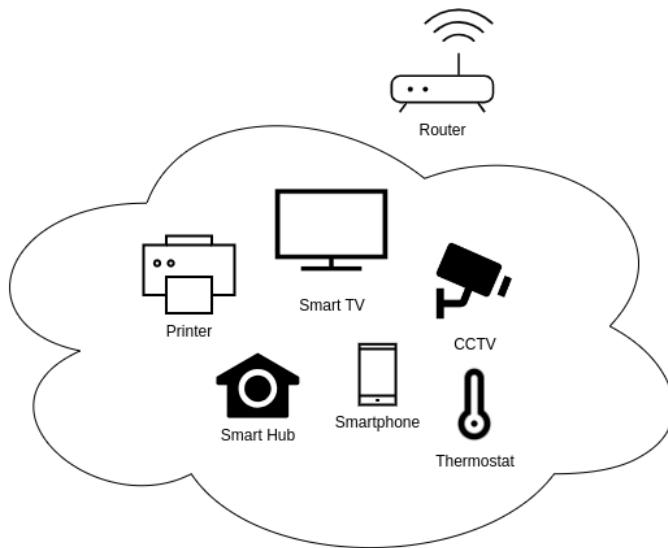


Figura 4.4: Escenario de red: *Smart Home*

como LPD. Otro elemento clave es la Smart TV, que se conecta al router para recibir contenido en streaming. Además, puede recibir contenido desde un smartphone conectado a la misma red, permitiendo compartir videos y fotos de forma rápida y sencilla.

El hub domótico es otro componente esencial, ya que se encarga de gestionar los dispositivos inteligentes del hogar, como luces, sensores y el termostato.

El smartphone del usuario también juega un papel importante, ya que permite gestionar la mayoría de los dispositivos inteligentes mediante aplicaciones específicas como el hub domótico, la impresora y el Smart TV.

Para garantizar la seguridad del hogar, el sistema de CCTV se conecta al router, permitiendo la transmisión en tiempo real de las cámaras a una aplicación móvil o un grabador de vídeo en red.

Por último, el termostato inteligente se comunica tanto con el router como con el hub domótico para recibir comandos y enviar datos sobre la temperatura del hogar. Cuando el usuario cambia la temperatura desde su smartphone, la solicitud viaja a través del router hasta el hub domótico, que la retransmite al termostato, asegurando un control preciso del clima interior.

3. Planta industrial

Este escenario representa una línea de producción de ensamblaje de piezas automatizadas, donde se utilizan varios dispositivos para monitorizar y controlar el proceso. Los dispositivos incluyen sensores, un PLC (Controlador Lógico Programable), y una interfaz HMI (Interfaz Hombre-Máqui-

4.1. OBTENCIÓN DE LOS DATOS DE ENTRADA

na) que se comunica con una base de datos centralizada en la nube.

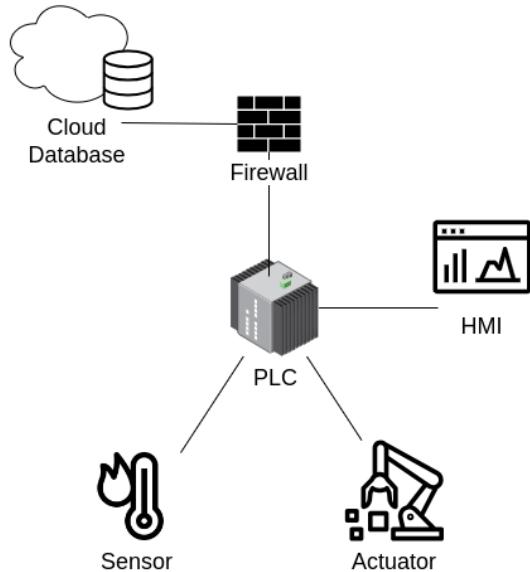


Figura 4.5: Escenario de red: *Planta Industrial*

En la zona de control existen unos sensores de temperatura y humedad distribuidos a lo largo de la línea de producción para monitorear las condiciones ambientales.

Los datos recogidos por estos sensores se envían al dispositivo central de la red, que es el controlador lógico programable o PLC. Este elemento controla la lógica de la producción, como el control de las máquinas y las actuadores (motores, válvulas, etc.). El controlador PLC presenta, además, la vulnerabilidad CVE-2022-29519. Esta vulnerabilidad, presente en STARDOM FCN Controller y FCJ Controller versiones R1.01 a R4.31, podría permitir que un atacante obtuviese credenciales en texto claro, otorgándole la posibilidad de leer/cambiar la configuración o actualizar el firmware alterado del controlador [24].

El PLC controla los actuadores (motores, válvulas) que son los encargados de realizar las acciones físicas de la línea de producción, como mover las piezas.

En la zona de monitorización se encuentra un dispositivo HMI que permite a los operadores supervisar el estado de la línea de producción y ajustar parámetros en tiempo real.

Finalmente, en la red IT se ubica un servidor en la nube que recopila y almacena los datos históricos de producción para análisis y reportes. Las comunicaciones entre la red IT y la red OT se producen a través de un firewall que mantiene ambos entornos correctamente separados.

4.2. Preparación del entorno

En esta sección se van a detallar los requisitos previos a satisfacer antes de la puesta en marcha de la herramienta. En primer lugar, se van a especificar las dependencias necesarias para su correcto funcionamiento, continuando con la preparación del entorno necesario para su ejecución.

4.2.1. Dependencias

Como ya se ha mencionado, ARGOS está escrito en Python, por lo que contar con este lenguaje de programación instalado es crucial. Python [25] es un lenguaje de programación interpretado de alto nivel. Entre sus principales ventajas cabe destacar su fácil sintaxis, su versatilidad y su rapidez. Además, es un lenguaje multiplataforma, lo que significa que puede ejecutarse en varios sistemas operativos, como Windows, macOS y Linux.

Al instalarlo, se pueden encontrar dos grandes ramas de versiones: **Python 2.x**, la cual llegó al final de su vida útil en 2020, y **Python 3.x** la más actual y por tanto la versión que se recomienda instalar. Esta primera versión de ARGOS se ha desarrollado utilizando la versión 3.13.1.

Otro de los puntos fuertes de Python es que tiene un amplio ecosistema de bibliotecas y paquetes que, además de cubrir una amplia gama de funcionalidades, pueden importarse fácilmente, agilizando el proceso de desarrollo. ARGOS hace uso de dos bibliotecas: **neo4j** para automatizar operaciones con la base de datos y **rich** para mostrar texto enriquecido a través de la terminal. Ambas pueden instalarse fácilmente ejecutando dentro de Python los comandos `pip install neo4j` y `pip install rich`, respectivamente.

Además de Python, la otra dependencia imprescindible para el correcto funcionamiento de ARGOS es Neo4j Desktop, un entorno de desarrollo local para trabajar con Neo4j de forma sencilla a través de una interfaz gráfica. Este software se puede obtener desde su sitio oficial [26].

4.2.2. Creación y configuración de la base de datos en Neo4j

Una vez se dispone de los requisitos de software necesarios, lo primero es crear una base de datos en Neo4j Desktop y configurar ARGOS para que haga uso de ella.

Para configurar la base de datos, basta con seguir los pasos que se indican a continuación:

1. Abrir Neo4j Desktop. La primera vez que se ejecute, es posible que se requiera iniciar sesión o crear una cuenta gratuita en su defecto.
2. Dentro de Neo4j Desktop, es posible organizar las bases de datos en proyectos. En el panel **Projects**, accesible clicando en el ícono de la carpeta

4.2. PREPARACIÓN DEL ENTORNO

de la parte superior izquierda, hacer clic en **New > Create Project** para crear un nuevo proyecto donde se alojará la base de datos.

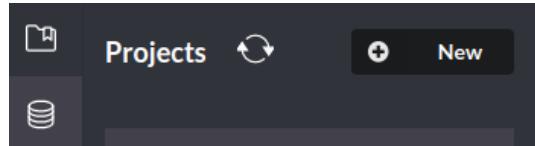


Figura 4.6: Botón para crear un nuevo proyecto

3. Dentro del proyecto, hacer clic en **Add** en la parte superior derecha y luego seleccionar **Local DBMS** para crear una nueva base de datos local. Será necesario establecer un nombre y una contraseña para poder utilizarla. Una vez definidas las credenciales, seleccionar **Create** para hacer efectiva la creación.

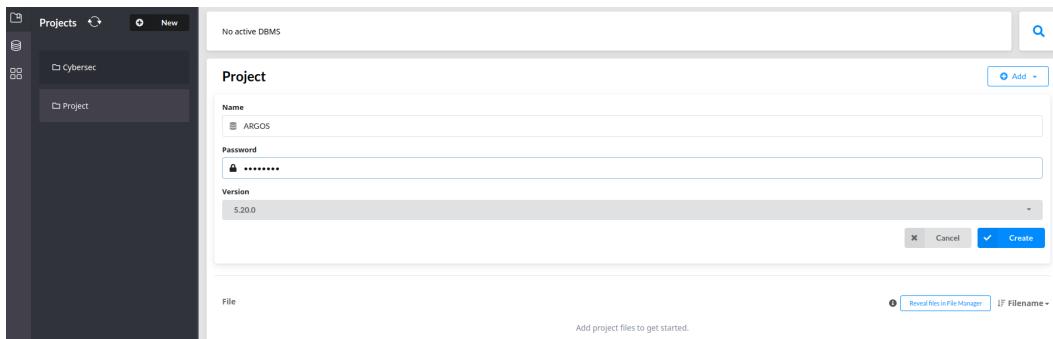


Figura 4.7: Creación de base de datos

4. Una vez creada, la base de datos aparecerá dentro del proyecto. Será necesario ahora pulsar **Start** para iniciarla.

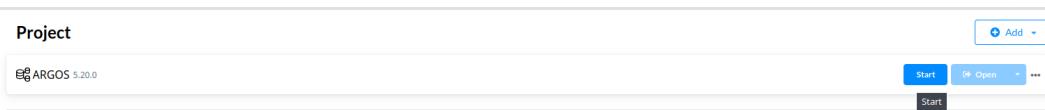


Figura 4.8: Iniciación de la base de datos

5. Cuando la base de datos se inicie correctamente y por tanto se encuentre en estado **Active**, pulsar el botón **Open** abrirá la consola de Neo4j en el navegador, haciendo posible el establecimiento de una conexión con ella.
6. En la parte inferior de la consola aparecerá la información asociada a la conexión (usuario y dirección URL). Es necesario introducir esta información (junto a la contraseña establecida en el paso 3) dentro de la función `start_neo4j()` en el fichero `argos.py`. Esto permitirá que ARGOS interaccione con la base de datos.
7. Finalmente, se recomienda configurar en la sección **Favorites** de la consola la `query MATCH (n) RETURN n`, que no hace más que devolver todos

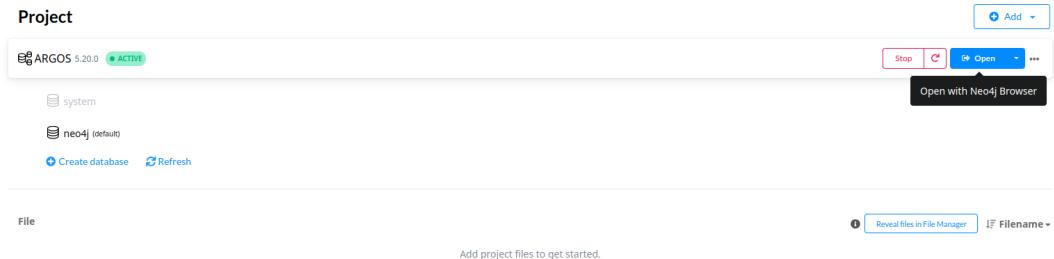


Figura 4.9: Base de datos iniciada correctamente

```

572 def start_neo4j():
573     # Conectar a la base de datos de Neo4j
574     uri = "bolt://localhost:7687"
575     user = "neo4j"
576     password = "argos123"
577     driver = connect_neo4j(uri, user, password)
578     return driver

```

Figura 4.10: Función `start_neo4j()`

los nodos y aristas presentes en la base de datos. Este ajuste permitirá ejecutar esta instrucción rápidamente, ya que dicha ejecución será necesaria cada vez que se realice una operación sobre la base de datos. En la Figura 4.11 se puede ver la consola, incluyendo la información de la conexión mencionada en el paso anterior y la configuración de la *query*.

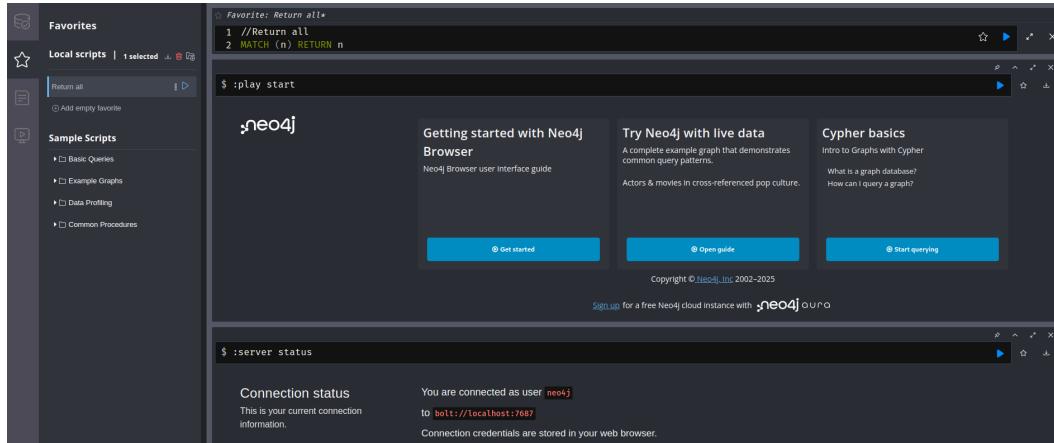


Figura 4.11: Consola de Neo4j con la conexión establecida

4.3. Explicación de la herramienta

En esta sección, se profundizará en la lógica interna de la herramienta, describiendo su funcionamiento a nivel detallado.

ARGOS es un programa de interfaz de línea de comandos (CLI), desarrollado en Python, que permite visualizar el impacto de secuencias de TTPs sobre escenarios de red. Para ello, interactúa de manera automatizada con la base de datos definida en la sección 4.2, utilizando los inputs descritos previamente en la sección 4.1.

A continuación, se describe la sintaxis de la herramienta, donde se pueden apreciar los comandos disponibles, los cuales se explicarán de manera detallada en los subapartados siguientes.

```
USO: python3 argos.py [comando] [opciones]

COMANDOS:

    prepare:      Cargar el escenario de red pasado
                   como argumento en Neo4j.

    attack:       Generar cadena de ataque y
                   dirigirla al escenario creado.

    trace:        Generar cadena de ataque a partir
                   de un estado inicial seleccionado
                   manualmente por el usuario.

    history:      Mostrar historial de ataques.

    clean:        Vaciar la base de datos.

OPCIONES:

    --help|-h:    Mostrar ayuda y salir.
```

4.3.1. Comando prepare

El comando `prepare` permite cargar en la base de datos Neo4j uno de los escenarios de red disponibles en el directorio *scenarios*. El fichero que contiene el escenario de red elegido ha de pasarse como argumento a la hora de ejecutar el comando (`$ python3 argos.py prepare <fichero-a-cargar>`)

La lógica detrás de esta instrucción es muy sencilla, ya que los ficheros *Cypher* contienen las instrucciones necesarias para crear directamente los escenarios como grafos. De esta forma, lo que hace ARGOS es, tras iniciar una conexión con la base de datos, leer y ejecutar el contenido del archivo de escenario como una *query*. Una vez insertado el escenario, se cierra la conexión con la base de datos.

En la Figura 4.12 se muestra un diagrama de alto nivel con las partes de la arquitectura involucradas.

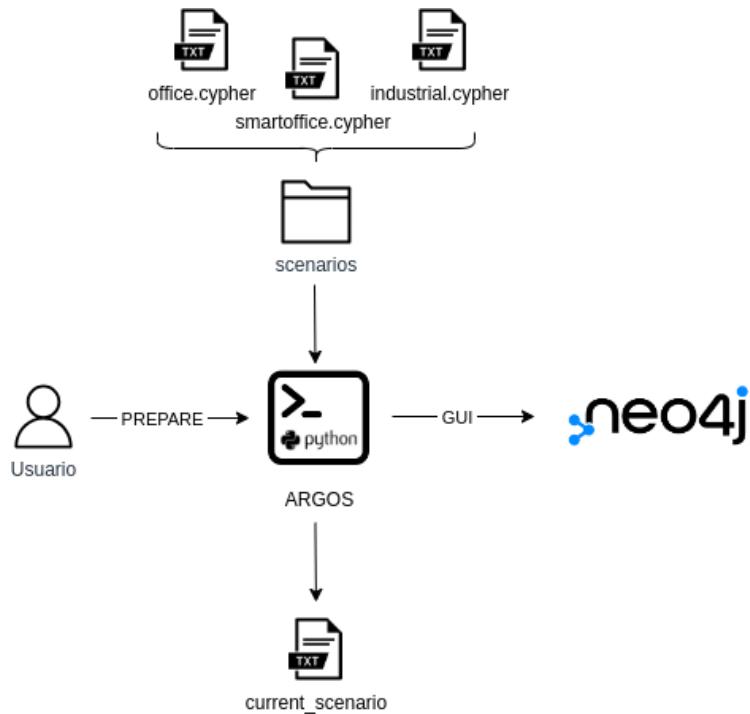


Figura 4.12: Diagrama de ejecución del comando `prepare`

En el fichero `current_scenario.txt` se almacena el nombre del escenario que se ha cargado. Esta información es utilizada por el comando `history` para llevar una trazabilidad de en qué escenario se han ido ejecutando los ataques. Su contenido se sobreescribe con cada ejecución del comando `prepare`.

4.3.2. Comando attack

Partiendo de una técnica seleccionada de manera aleatoria en cada ejecución, el comando `attack` genera de forma probabilística una secuencia de técnicas de ataque utilizando un modelo de Cadenas de Markov y la ejecuta sobre el escenario cargado, asociando cada técnica de la secuencia con los activos del escenario a los que afecta. Tras el ataque, se utiliza la librería Rich para mostrar por la terminal un *dashboard* con una serie de métricas sobre su impacto en el escenario (activos afectados, técnicas utilizadas, criticidad del ataque, etc.). Parte de estas métricas se almacenan junto al escenario en el que se ha ejecutado el ataque en el archivo `attack_history.csv`, que contiene el historial de todos los ataques ejecutados.

El funcionamiento de este comando es el más complejo, por lo que se va a subdividir la explicación de su ejecución en módulos. El primer módulo se corresponde con la simulación de las cadenas de TTPs. Estas TTPs se representan en la bases de datos con sus correspondientes nodos y relaciones, al igual que los escenarios, permitiendo realizar una asociación entre cada TTP

4.3. EXPLICACIÓN DE LA HERRAMIENTA

y los activos a los que afecta. La explicación del modelo de datos empleado en Neo4j para las secuencias sería el segundo módulo.

Finalmente, una vez simulada la secuencia y su impacto sobre el escenario, en el tercer módulo se calculan algunas métricas y estadísticas del ataque. En la Figura 4.13 se puede observar un diagrama con todas las partes implicadas en el funcionamiento de este comando.

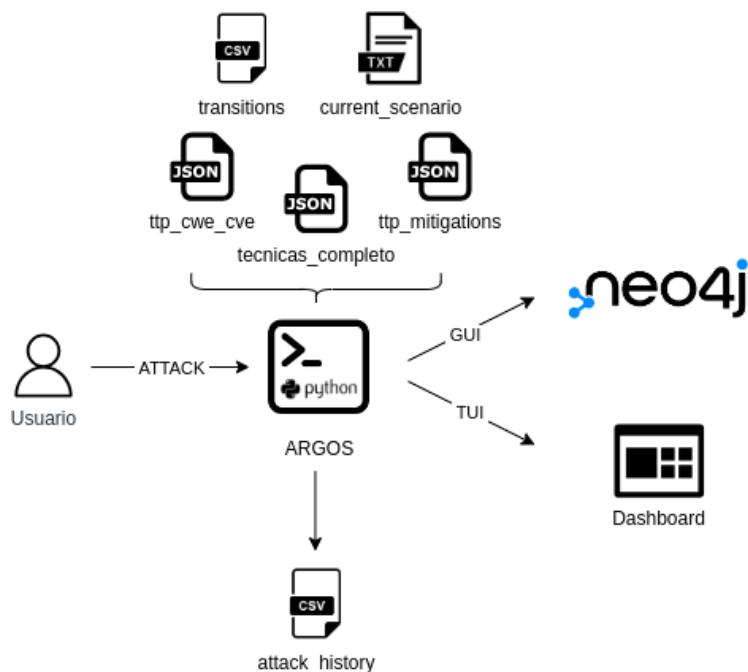


Figura 4.13: Diagrama de ejecución del comando `attack`

Simulación de secuencias de TTPs

Lo primero que hace el comando `attack` una vez ejecutado es simular una secuencia de técnicas de la matriz empresarial de MITRE ATT&CK. Se utilizan para ello Cadenas de Markov, que permiten modelar las transiciones entre diferentes estados o técnicas de un ataque.

La Cadena de Markov implementada permite calcular el siguiente estado únicamente a partir del estado actual utilizando las probabilidades de transición entre estados, definidas en el fichero `transitions.csv`. La técnica inicial de la secuencia se selecciona de manera aleatoria en cada ejecución de entre las disponibles en dicho fichero.

Esto significa que cada técnica utilizada en el ataque tiene una probabilidad de llevar a la siguiente, asegurando que las transiciones entre estados sean coherentes con patrones de ataque reales. De esta forma, ARGOS puede modelar la progresión de un ciberataque desde su punto de inicio hasta su posible finalización, considerando factores como la probabilidad de éxito de cada técnica y las condiciones del escenario.

La función `generate_markov_sequence` genera la secuencia de estados. Esta función carga en primer lugar las posibles transiciones entre estados desde el archivo `transitions.csv`. Acto seguido, se verifica que la suma de las transiciones de un estado a otros sume 1, pues una de las características principales de las Cadenas de Markov consiste en que la probabilidad de un estado futuro depende única y exclusivamente del estado actual y no de estados anteriores.

Tras verificar las probabilidades, se selecciona de forma aleatoria un estado inicial de entre aquellos disponibles en el fichero de transiciones. A continuación, se ejecuta la función `simulate_chain`, encargada de simular la cadena partiendo del estado inicial seleccionado. En cada iteración, esta función calcula el siguiente estado utilizando el estado actual y las probabilidades definidas en `transitions.csv`, previamente extraídas. La longitud de la secuencia generada viene determinada por varios factores, los cuales se describen a continuación:

Antes de calcular el siguiente estado, se determina si el estado actual es absorbente, es decir, un estado final que indica que la secuencia debe detenerse. Para ello, se utiliza la lista `absorbent_probabilities`, que contiene los estados candidatos a ser absorbentes junto con la probabilidad de que así lo sean. La determinación de los posibles estados absorbentes se basó en la frecuencia con la que ciertas técnicas aparecen en la última posición de las secuencias de ataque simuladas. Se analizaron las técnicas más recurrentes y se calculó el porcentaje de veces que cada una ocupaba la posición final. Las cuatro técnicas seleccionadas fueron T1204, T1047, T1078 y T1102, ya que son las más frecuentes y presentan patrones de finalización significativos. Por ejemplo, la técnica T1047 aparece en la última posición el 97 % de las veces, lo que indica una alta probabilidad de ser un estado final. Las técnicas T1102, T1204 y T1078 aparecen en la última posición con porcentajes del 59 %, 42 % y 40,5 % respectivamente. Estas últimas, pese a ser notablemente inferiores al caso anterior, también implican la finalización del ataque en una gran cantidad de casos. Estos porcentajes se tradujeron directamente en las probabilidades con las que cada estado de la lista `absorbent_probabilities` será elegido como absorbente en el modelo de cadena de Markov, provocando la finalización de la secuencia al ser alcanzado.

Otra comprobación que realiza esta función antes de seleccionar el siguiente estado es la detección de la repetición de estados recientes. Esto previene que la secuencia quede atrapada en ciclos repetitivos y garantiza una evolución más realista de los estados. Para ello, se utiliza una estructura `deque` con una longitud máxima predefinida para almacenar los últimos cinco estados visitados. Antes de avanzar, se comprueba si el estado actual ya se encuentra en esta lista. Si es así, se considera que se ha detectado un bucle, se muestra un mensaje de advertencia y la simulación se detiene. Si no, el estado se agrega a la lista y la simulación continúa.

Resumiendo, en cada paso se verifica si el estado actual es absorbente o si se ha detectado un bucle reciente. Si se cumple alguna de estas condiciones, la simulación termina. De lo contrario, se calcula el siguiente estado y se repite

4.3. EXPLICACIÓN DE LA HERRAMIENTA

el proceso hasta alcanzar un número máximo de pasos (definido para evitar bucles infinitos) o no encontrar más transiciones. El resultado de la ejecución de esta función es una cadena formada por una secuencia de estados que se corresponden con técnicas de la matriz empresarial de MITRE ATT&CK.

Modelo de datos

Una vez se dispone de la secuencia de ataque, el siguiente paso es su inserción en la base de datos y la identificación de los activos del escenario de red cargado que son vulnerables a las técnicas de la secuencia simulada.

Para ello, de igual forma que se hizo con los escenarios de red, lo primero es definir un modelo de datos. En este caso, el modelo es muy sencillo, pues para modelar los ataques tan solo es necesaria una entidad, la entidad **Técnica**. Esta entidad permite representar cada una de las técnicas de la secuencia generada. Todas ellas cuentan con una serie de atributos que se utilizan para su asociación con los activos vulnerables, descritos en la Tabla 4.3.

Atributo	Tipo	Descripción
<code>id</code>	String	ID de la técnica
<code>name</code>	String	Nombre de la técnica
<code>platforms</code>	List	Plataformas comprometidas por la técnica
<code>permissions_required</code>	List	Nivel de permisos necesario en el activo comprometido
<code>system_requirements</code>	List	Capacidades necesarias para comprometer el activo
<code>cve</code>	List	Identificadores de vulnerabilidades CVE explotadas por la técnica
<code>cwe</code>	List	Categorías de debilidades de seguridad CWE explotables por la técnica

Tabla 4.3: Atributos de los nodos **Técnica**

Los valores de estos atributos para cada técnica de la secuencia se extraen de los ficheros `tecnicas_completo.json` y `ttp_cwe_cve.json`, que contienen las características principales y los CVEs y CWEs a los que afecta cada una, respectivamente. Respecto a las relaciones, se distinguen principalmente dos de ellas, descritas en la siguiente tabla:

Relación	Descripción
TRANSICIÓN	Una técnica sucede a otra en la secuencia de ataque
EXPLOTACIÓN	Una técnica explota o compromete un activo vulnerable

Tabla 4.4: Posibles conexiones de los nodos **Técnica**

De esta forma, el diagrama del modelo de datos relativo a las rutas de ataque quedaría como se indica en la Figura 4.14. Este modelo permite modelar las secuencias de ataque y su impacto sobre los escenarios de red.

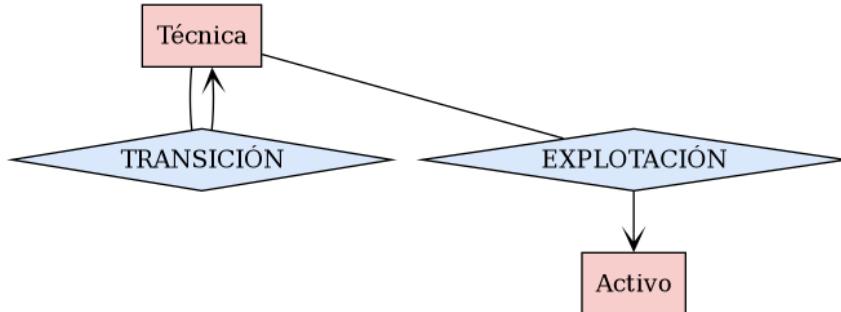


Figura 4.14: Diagrama del modelo de datos empleado en los ataques

Si se unifican los dos modelos de datos ya presentados, quedaría definido el modelo de datos global utilizado en ARGOS, presentado en la Figura 4.15.

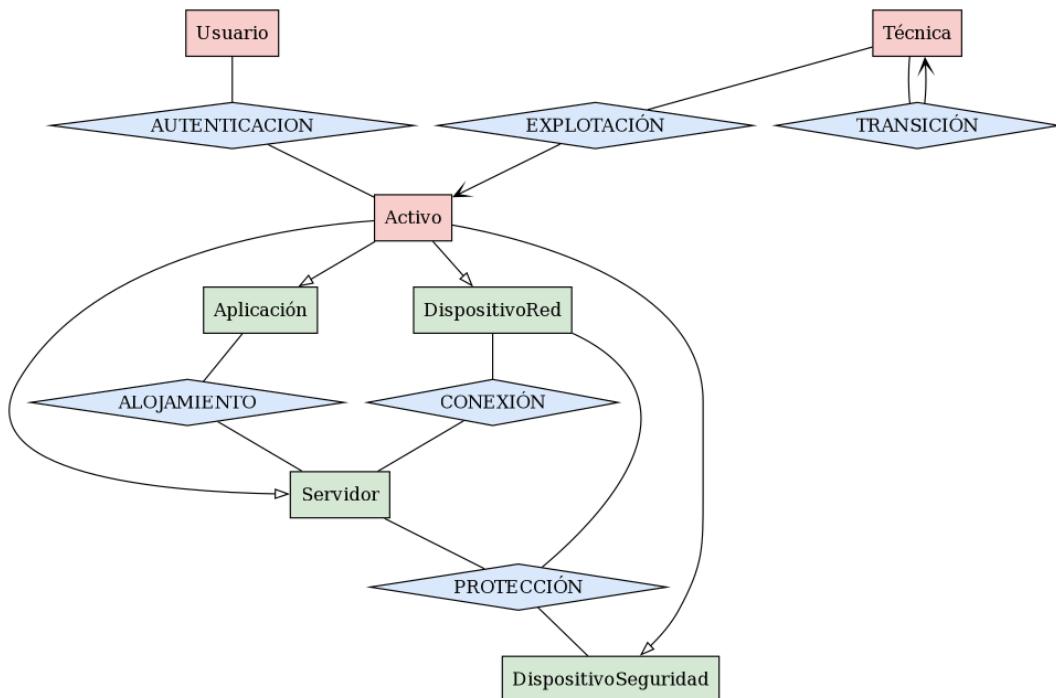


Figura 4.15: Diagrama del modelo de datos empleado en ARGOS

Inserción de la secuencia de ataque en la base de datos

Una vez generada la secuencia de ataque, esta se inserta en la base de datos siguiendo el modelo definido. Posteriormente, ARGOS asocia cada técnica con los activos a los que afecta dentro del escenario cargado. Para ello, analiza

4.3. EXPLICACIÓN DE LA HERRAMIENTA

características como los sistemas operativos presentes, los permisos disponibles y las vulnerabilidades conocidas, que habían sido definidas como atributos en los nodos. Si un activo cumple con los criterios necesarios, se establece una relación entre la técnica de ataque y el activo comprometido, reflejando el impacto del ataque en la red simulada.

En primer lugar, para la inserción de la secuencia de técnicas en el escenario, se cargan tres conjuntos de datos desde archivos JSON: las técnicas de ataque desde `tecnicas_completo.json`, y las vulnerabilidades y debilidades asociadas (CVEs y CWEs) desde `ttp_cwe_cve.json`

La función `load_techniques_json` es la encargada de leer el archivo JSON que contiene las técnicas de ataque y sus atributos. Para cada técnica, extrae los datos asociados a los atributos, como el identificador, el nombre, las tácticas asociadas, las plataformas afectadas, los permisos requeridos y los requisitos del sistema. Toda esta información se organiza en un diccionario para un acceso eficiente durante la construcción del grafo. Este paso es crucial para contextualizar cada técnica dentro del escenario de ataque simulado.

La creación del grafo que representa la secuencia de ataque se lleva a cabo mediante la función `create_attack_graph`. Esta función recorre la cadena de técnicas generada por la función `generate_markov_sequence` (la cual se le pasa como parámetro) y establece conexiones entre técnicas consecutivas, representando las transiciones entre ellas. Para cada técnica de origen y destino, o estado n y $n+1$, se extraen del diccionario anterior los atributos como las plataformas afectadas, los permisos necesarios, los requisitos del sistema y las vulnerabilidades asociadas (CVEs y CWEs). Con esta información, se crean nodos en la base de datos Neo4j que representan las técnicas y sus características, y se establecen relaciones de transición entre ellas. Esto permite visualizar cómo se desarrolla el ataque a lo largo de diferentes etapas y técnicas.

Finalmente, la función `link_attack_to_scenario` vincula las técnicas de ataque con los activos que pueden verse afectados. La lógica utilizada para vincular las técnicas de ataque con los activos afectados se basa en la ejecución de una consulta *Cypher* sobre la base de datos Neo4j. Esta asociación se puede dar mediante dos condiciones, descritas de forma esquemática en la Figura 4.16:

Primero, la consulta realiza un emparejamiento con los nodos de tipo **Activo**. Acto seguido, se aplica un conjunto de condiciones para filtrar solo aquellos activos que sean relevantes para la técnica en cuestión. El filtro principal, representado de color naranja en la Figura 4.16, exige que el activo se ejecute en alguna de las plataformas afectadas por la técnica y, además, que cumpla con dos condiciones simultáneas: que posea al menos un permiso que coincida con los permisos requeridos por la técnica y que cuente con alguna capacidad del sistema que corresponda con los requisitos de la técnica.

Esta condición compuesta refleja la necesidad de que el activo no solo opere en la plataforma objetivo, sino que también cumpla con las condiciones de acceso y recursos que la técnica requiere para ser aplicada. Cuando en el modelo de

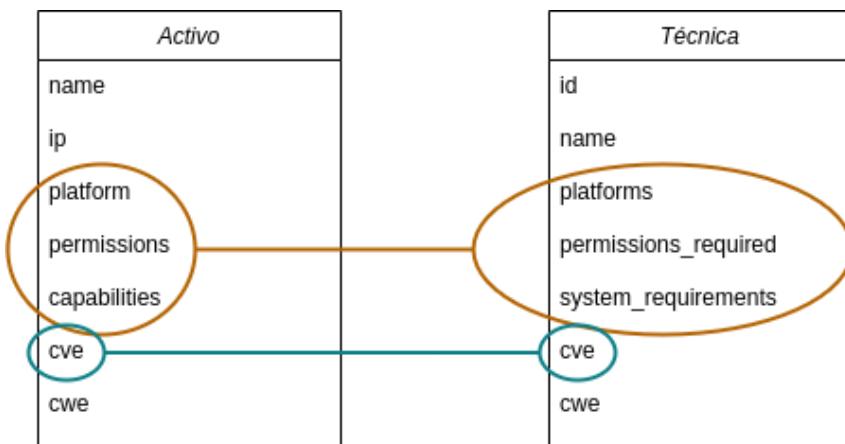


Figura 4.16: Condiciones de explotación de una técnica sobre un activo

datos de los escenarios de red (detallado en la sección 4.1.2) se especificaron los valores concretos que podía tomar cada parámetro, era porque estos valores debían de ser los mismos que se utilizaban en las técnicas para que se pudiera realizar una asociación.

Adicionalmente, como alternativa a este conjunto de condiciones, la consulta también considera un segundo filtro (representado en color azul) para los activos que tengan vulnerabilidades identificadas por los CVEs relacionados con la técnica, permitiendo así que la técnica explote directamente vulnerabilidades conocidas.

Cuando se encuentra una coincidencia que cumpla una de estas dos condiciones, se establece una relación de explotación entre la técnica y el activo.

Métricas

A la hora de asociar la secuencia de ataque generada con los activos del escenario, el comando `attack` extrae simultáneamente una serie de métricas acerca de los resultados de la ejecución. Estos datos permiten comprender tanto el alcance del ataque como la relación entre las técnicas empleadas y los activos comprometidos, lo que resulta fundamental para la visualización y el análisis del escenario de ataque simulado.

En el flujo del código, esta extracción se lleva a cabo durante la ejecución de la función `link_attack_to_scenario`. Con cada iteración de la consulta *Cypher* en la que se evalúan las condiciones anteriores y crean las relaciones de explotación, también se devuelve y se almacena en *arrays* información clave sobre dicha relación. Concretamente, la información que se almacena es el nombre del activo afectado, el identificador y el nombre de la técnica que lo compromete.

Esta información se envía como argumento a la función `create_dashboard`, la cual genera un tablero visual e interactivo que muestra un resumen detallado

4.3. EXPLICACIÓN DE LA HERRAMIENTA

del escenario de ataque simulado. Para ello, se apoya en la biblioteca *Rich* de Python, que facilita la renderización de paneles y visualizaciones en la terminal.

Con la información recibida, dicha función calcula una serie de métricas que facilitan la comprensión del impacto del ataque, mostrando información clave sobre los activos comprometidos, las técnicas utilizadas, la gravedad del ataque y las recomendaciones de mitigación. Estas estadísticas se presentan en paneles organizados de forma jerárquica y visualmente clara, y permiten a los usuarios interpretar de manera rápida y eficaz los resultados del ataque simulado.

A continuación, se detallan los paneles que componen el *dashboard* junto a las métricas que muestra cada uno:

- **Panel ATAQUE:** muestra la cadena de técnicas empleadas en el ataque. Las técnicas aparecen en orden y conectadas con flechas, lo que permite visualizar claramente cómo se ha desarrollado el ataque desde su inicio hasta su última fase.
- **Panel DEFENSA:** muestra una lista detallada en la que se indica, para cada activo comprometido, la técnica específica que lo ha vulnerado. Este panel proporciona información valiosa para identificar puntos débiles en la infraestructura y planificar medidas correctivas.
- **Panel ACTIVOS:** este panel proporciona una serie de estadísticas sobre los activos del escenario, permitiendo tener una visión clara del impacto del ataque sobre la infraestructura tecnológica. En concreto, se muestran las siguientes métricas:
 - **Total de activos:** número total de activos presentes en el escenario cargado.
 - **Activos afectados:** número y porcentaje de activos que han sido comprometidos por, al menos, una técnica.
 - **Activos seguros:** número y porcentaje de activos que no han sido vulnerados.
 - **Activo(s) más afectado(s):** de entre los activos afectados, activo o activos comprometidos por el mayor número de técnicas.
- **Panel TÉCNICAS:** este panel resume datos sobre las técnicas empleadas, lo que permite enfocar las medidas defensivas en los métodos de ataque más dañinos.
 - **Técnicas en la secuencia:** número total de técnicas en la secuencia.
 - **Técnicas distintas empleadas:** de entre las técnicas totales de la secuencia, número de técnicas que aparecen una única vez en ella.
 - **Técnicas exitosas:** número de técnicas que han logrado comprometer, al menos, un activo.

- **Técnica más dañina:** identificador de la técnica que ha comprometido el mayor número de activos distintos.
- **Mitigación recomendada:** estrategia de defensa recomendada para paliar el impacto de la técnica más dañina, obtenida del fichero `ttp_mitigations.json`
- **Panel CRITICIDAD:** utilizando los datos anteriores, se le asigna una criticidad (C) de entre 0 y 100 al ataque, calculada como:

$$C = \left(\left\lfloor \frac{\text{Activos afectados}}{\text{Activos totales}} \times 50 \right\rfloor + \left\lfloor \frac{\text{Técnicas exitosas}}{\text{Total explotaciones}} \times 50 \right\rfloor \right)$$

Esta criticidad se visualiza mediante una barra de progreso coloreada de la siguiente forma:

- Verde (baja) si $C \leq 33$
- Amarillo (media) si $33 < C \leq 66$
- Rojo (alta) si $66 < C \leq 100$

Antes de finalizar, la función genera un ID único para el ataque y guarda un registro de las estadísticas más relevantes (activo más afectado, técnica más exitosa, escenario y fecha de ejecución) en un archivo CSV denominado `attack_history.csv`. Esto permite llevar un histórico de las simulaciones para futuros análisis.

Finalmente, el tablero se muestra en pantalla de forma interactiva hasta que el usuario presiona `Enter`, lo que facilita revisar con calma toda la información antes de cerrar la visualización. En la Figura 4.17 se presenta un *mockup* con la distribución de los paneles en el tablero.



Figura 4.17: Distribución de los paneles en el *dashboard*

4.3.3. Comando trace

El comando `trace` permite simular una secuencia de ataque partiendo de una técnica inicial seleccionada manualmente por el usuario. Esto resulta especialmente útil en situaciones en las que se ha detectado una técnica en la red y se desea analizar los posibles próximos movimientos del atacante. Al igual que `attack`, este comando utiliza un modelo basado en Cadenas de Markov para generar probabilísticamente la evolución del ataque, asociando cada técnica simulada a los activos relevantes del escenario cargado. Tras la simulación, se presenta un *dashboard* en la terminal (apoyado en la librería Rich) con métricas detalladas del impacto del ataque, y los resultados se registran en el archivo `attack_history.csv` para su posterior análisis.

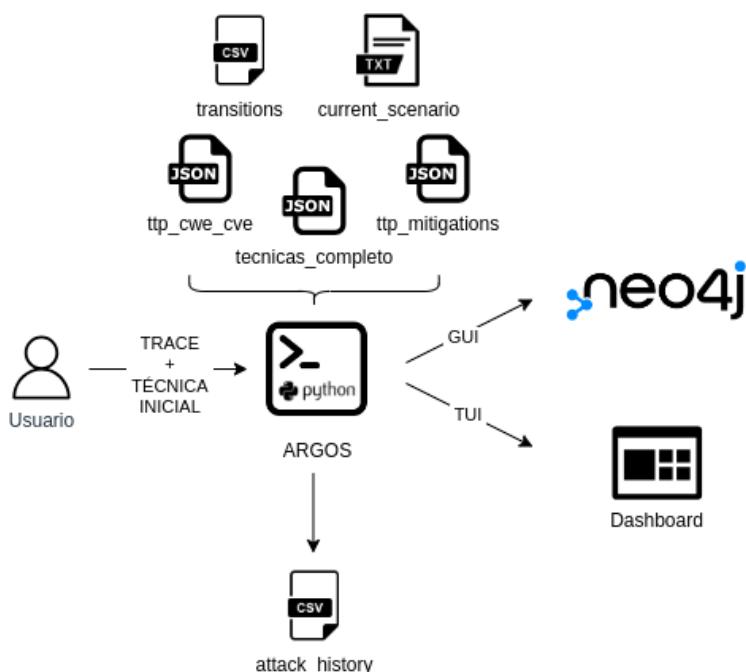


Figura 4.18: Diagrama de ejecución del comando `trace`

El flujo de ejecución del comando `trace` es exactamente el mismo que el del comando `attack`, con la diferencia de que el estado inicial es seleccionado manualmente por el usuario en vez de aleatoriamente por la herramienta.

La lógica encargada de la selección del estado inicial se implementa en la función `generate_markov_sequence`, en la cual se genera la secuencia de estados. Dicha función recibe como parámetro el comando seleccionado por el usuario.

Como ya se ha mencionado, si el comando es `attack`, se selecciona un estado aleatoriamente. Si el comando elegido es `trace`, se muestran por la terminal los estados disponibles en el fichero `transitions.csv` y se le pide al usuario que introduzca aquel que desea como inicial. Si el estado seleccionado no se encuentra entre los disponibles en el fichero, se mostrará un error y finalizará la ejecución, mientras que si el estado es correcto, la ejecución continúa de manera idéntica a la descrita en la sección anterior para el comando `attack`.

4.3.4. Comando history

El comando `history` se encarga de mostrar el historial de ataques registrados en el sistema. Su ejecución activa la función `display_attack_history_csv()`, la cual lee y presenta los datos almacenados en el archivo `attack_history.csv`.

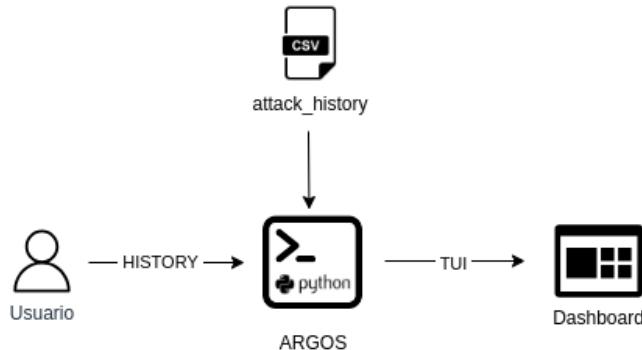


Figura 4.19: Diagrama de ejecución del comando `history`

Cuando se ejecuta el comando, el programa primero verifica si el archivo `attack_history.csv` existe. Si el archivo no se encuentra, se muestra un mensaje de error indicando que aún no se han registrado ataques previos y que es necesario ejecutar al menos un ataque para generar el historial.

Si el archivo sí existe, se utiliza Pandas para leer su contenido y estructurarlo en una tabla. Cada fila del archivo representa un ataque previo y contiene datos clave como el ID del ataque, el escenario en el que ocurrió, el activo más afectado, la técnica más exitosa y la fecha de ejecución.

Finalmente, la información se organiza en un formato visualmente claro mediante la biblioteca *Rich*, que permite mostrar la tabla con colores y alineación mejorada. Cada columna del archivo se convierte en un encabezado dentro de la tabla, y las filas con los datos de los ataques previos se agregan con un estilo distintivo para facilitar su lectura.

4.3.5. Comando clean

El comando `clean` se encarga de borrar completamente todos los datos almacenados en la base de datos de Neo4j. Su ejecución elimina todos los nodos y relaciones, dejando la base de datos en un estado vacío.



Figura 4.20: Diagrama de ejecución del comando `clean`

Tras ejecutar el comando, primero se inicia una conexión con Neo4j mediante la función `start_neo4j()`. Luego, se llama a la función `clean_database()`,

4.3. EXPLICACIÓN DE LA HERRAMIENTA

que establece una sesión con la base de datos y ejecuta la consulta `MATCH (n) DETACH DELETE n`. Esta consulta selecciona todos los nodos y los elimina junto con sus relaciones.

Si la operación se realiza con éxito, se muestra un mensaje en color verde indicando que la base de datos ha sido limpiada correctamente. En caso de que ocurra un error al conectar con Neo4j, se captura la excepción `Neo4jError` y se muestra un mensaje en rojo informando del fallo en la conexión.

Finalmente, se cierra la conexión con la base de datos y el programa termina su ejecución.

Capítulo 5

Resultados y validación

En este capítulo se presenta la validación de la herramienta ARGOS y los resultados obtenidos a partir de su ejecución. El proceso de validación se divide en varios pasos, que incluyen la carga de escenarios, la ejecución de simulaciones de ataque y el análisis de los resultados obtenidos, tal y como se muestra en el diagrama de ejecución de la Figura 5.1.

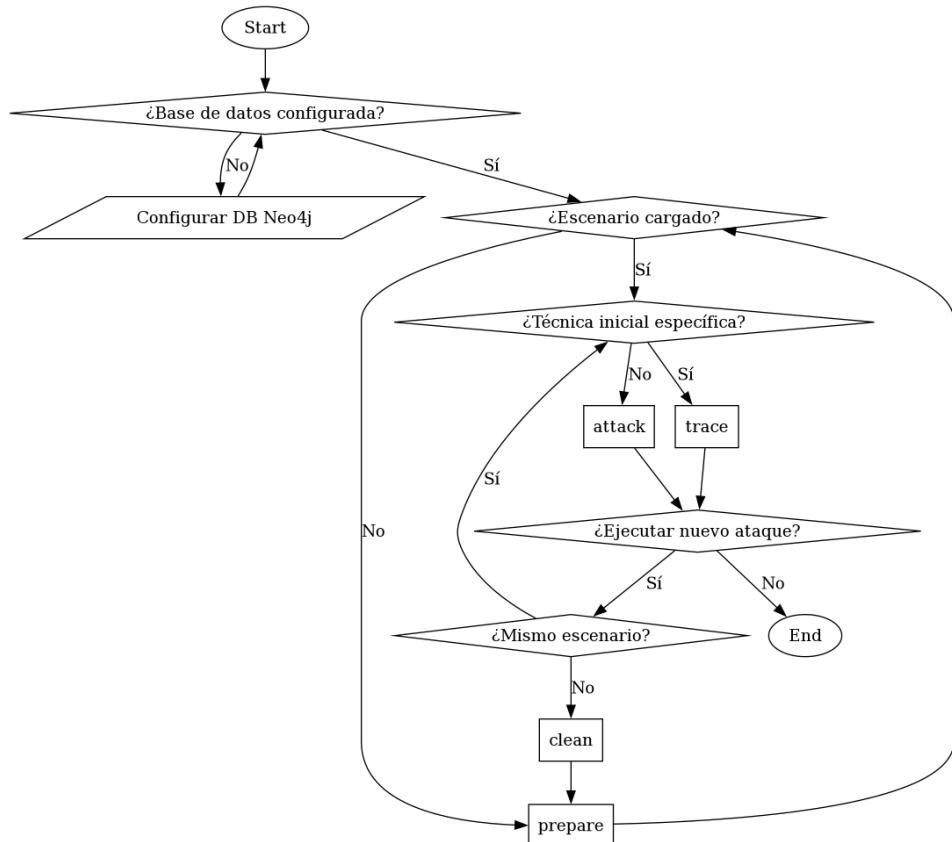


Figura 5.1: Diagrama de flujo de ARGOS

5.1. CARGA DE ESCENARIOS

5.1. Carga de escenarios

En esta sección se va a proceder a cargar en Neo4j cada uno de los tres escenarios de red definidos en la sección 4.1.2. De esta forma, se podrá observar si la definición de los ficheros *Cypher* es correcta. En caso de serlo, se apreciará la traducción del escenario de red teórico presentado como un grafo.

5.1.1. Carga de escenario *Oficina Corporativa*

El primer paso antes de utilizar ARGOS es iniciar la base de datos y abrir la consola, tal y como se explicó en la sección 4.2. El segundo paso imprescindible es desplazarse en la terminal hasta el directorio *argos*, donde se encuentra el fichero *argos.py*. Una vez hecho esto, el comando a ejecutar en la terminal para cargar el escenario correspondiente a la oficina corporativa sería el siguiente:

```
$ python3 argos.py prepare scenarios/oficina.cypher
```

Tras ejecutarlo, el programa indica que dicho escenario se ha cargado sin problemas en la base de datos.

```
~/TFM/argos >>> python3 argos.py prepare scenarios/oficina.cypher
[+] Escenario 'oficina' insertado en Neo4j desde el fichero 'scenarios/oficina.cypher'.
~/TFM/argos >>> |
```

Figura 5.2: Carga exitosa del escenario correspondiente a la oficina corporativa

Tras este mensaje, habría que desplazarse a la consola de Neo4j y ejecutar la consulta *Return all* que se recomendó agregar a favoritos en la sección 4.2. Esta query devolverá todos los nodos y relaciones presentes en la base de datos, en este caso el escenario de red que se presentó en la Figura 4.3.

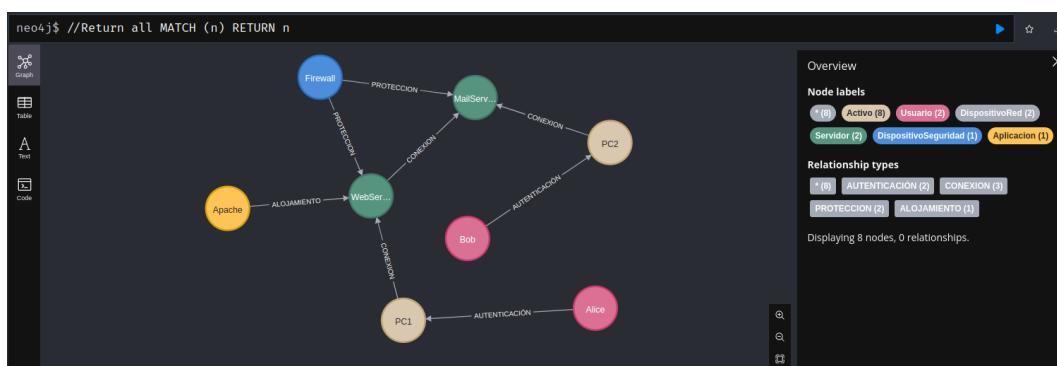


Figura 5.3: Escenario de oficina corporativa en Neo4j

La oficina queda representada por 8 nodos: 2 de tipo **Usuario** (representados en la Figura 5.3 en color rojo), 2 de tipo **DispositivoRed** (en color arena), 2 de tipo **Servidor** (en color verde), 1 **DispositivoSeguridad** (en color azul) y 1 **Aplicacion** (en color amarillo).

Los usuarios están representados por los nodos **Alice** y **Bob**. **Alice** se autentica en el nodo **PC1**, un equipo Windows con permisos estándar y diversas capacidades que le permiten interactuar con otros activos en la red. **Bob**, en cambio, opera en el nodo **PC2**, el cual representa un equipo Linux y cuenta con permisos de administrador, lo que le otorga un mayor nivel de acceso y control.

Los nodos de tipo **Servidor** son **WebServer** y **MailServer**. **WebServer** corre en Windows con privilegios administrativos y capacidades relacionadas con la ejecución remota y la gestión de archivos y directorios críticos. **MailServer**, basado en Linux, presenta características similares en cuanto a acceso remoto y almacenamiento de información.

El dispositivo de seguridad en la red es el nodo **Firewall**, que cumple una función de protección. Cuenta con permisos administrativos y la capacidad de capturar tráfico de red y analizar paquetes, lo que permite monitorear y restringir accesos no autorizados.

También se ha modelado una aplicación, **Apache**, que opera en Linux con permisos administrativos. Esta aplicación tiene acceso a archivos y directorios específicos y presenta vulnerabilidades potenciales debido a software no actualizado.

En cuanto a las relaciones, **Alice** se conecta a **WebServer** mediante el protocolo LDAP, mientras que **Bob** establece comunicación con **MailServer** utilizando el protocolo SMTP. Además, **WebServer** y **MailServer** están enlazados mediante una conexión en el puerto 25 con protocolo SMTP, lo que indica la comunicación entre estos servicios.

El nodo **Firewall** protege tanto a **WebServer** como a **MailServer**, reforzando la seguridad de la red. Finalmente, **Apache** está alojado en **WebServer**, lo que indica su dependencia de este servidor para operar dentro del entorno web.

5.1.2. Carga de escenario *Smart home*

Para cargar el escenario de tipo *smart home* sería necesario ejecutar el siguiente comando:

```
$ python3 argos.py prepare scenarios/smarthome.cypher
```

```
~/TFM/argos >>> python3 argos.py prepare scenarios/smarthome.cypher
[+] Escenario 'smarthome' insertado en Neo4j desde el fichero 'scenarios/smarthome.cypher'.
```

Figura 5.4: Carga exitosa del escenario *Smart Home*

Tras la carga exitosa, ejecutando la query *Return all* en la consola de Neo4j se visualiza el escenario descrito en la Figura 4.4, el cual modela un entorno de hogar inteligente donde diversos dispositivos se interconectan a través de un router principal, permitiendo la comunicación y automatización del hogar.

5.1. CARGA DE ESCENARIOS

Está compuesto por 6 nodos de tipo **DispositivoRed** (de color arena en la Figura 5.5) y un nodo de tipo **DispositivoSeguridad** (representado en color azul).

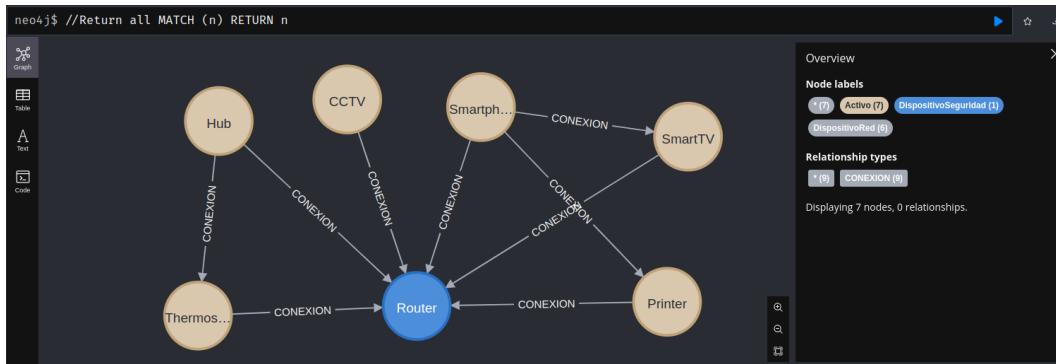


Figura 5.5: Escenario *Smart Home* en Neo4j

El nodo central de la red es el **Router**, que opera en la plataforma de red con permisos administrativos. Este dispositivo tiene capacidades de captura de tráfico y explotación remota mediante servicios accesibles en la red, desempeñando un rol fundamental en la seguridad y conectividad del entorno.

Los dispositivos conectados a la red se han modelado de la siguiente forma:

- **Printer**: impresora conectada a la red con permisos de usuario. Cuenta con un servicio activo que acepta conexiones remotas y credenciales válidas, además de acceso a unidades compartidas de la red.
- **SmartTV**: televisor inteligente basado en una plataforma SaaS, con permisos de usuario y capacidad para acceder a contenido multimedia compartido en la red.
- **CCTV**: sistema de videovigilancia con permisos administrativos. Posee servicios remotos accesibles mediante credenciales válidas y capacidades avanzadas de captura de tráfico.
- **Hub**: dispositivo IoT en una plataforma IaaS, con permisos administrativos. Puede acceder a directorios, archivos y puntos de acceso API para gestionar información de interés.
- **Smartphone**: dispositivo basado en macOS con permisos de usuario. Puede interactuar con medios extraíbles y cuenta con software potencialmente vulnerable.
- **Thermostat**: termostato inteligente basado en IaaS, con permisos a nivel de SYSTEM. Cuenta con acceso a archivos, directorios y puntos API clave para la gestión de la climatización.

En cuanto a las relaciones, todos los dispositivos principales (**Hub**, **Thermostat**, **CCTV**, **Printer**, **Smartphone** y **SmartTV**) se conectan al **Router** mediante el protocolo WPA2, garantizando la seguridad de la red inalámbrica. Pese a que

la red sea centralizada, existen además una serie de conexiones dedicadas a servicios específicos:

- El **Hub** se comunica con el nodo **Thermostat** mediante el protocolo MQTT en el puerto 1883, permitiendo la automatización doméstica.
- El **Smartphone** establece conexión con la **SmartTV** a través de DLNA en el puerto 5000 para la transmisión de contenido multimedia.
- La **Printer** es accesible desde el **Smartphone** a través del protocolo IPP en el puerto 631, permitiendo la impresión remota.

5.1.3. Carga de escenario *Planta Industrial*

El tercer y último escenario se carga ejecutando el siguiente comando:

```
$ python3 argos.py prepare scenarios/industrial.cypher
```

```
~/TFM/argos >>> python3 argos.py prepare scenarios/industrial.cypher
[+] Escenario 'industrial' insertado en Neo4j desde el fichero 'scenarios/industrial.cypher'.
```

Figura 5.6: Carga exitosa del escenario correspondiente a la planta industrial

En la consola de Neo4j, el escenario (presentado inicialmente en la Figura 4.5) se traduce en el grafo visible en la Figura 5.7. Este grafo lo componen 5 nodos de tipo **DispositivoRed** (en color arena) y 1 **DispositivoSeguridad** (en color azul).

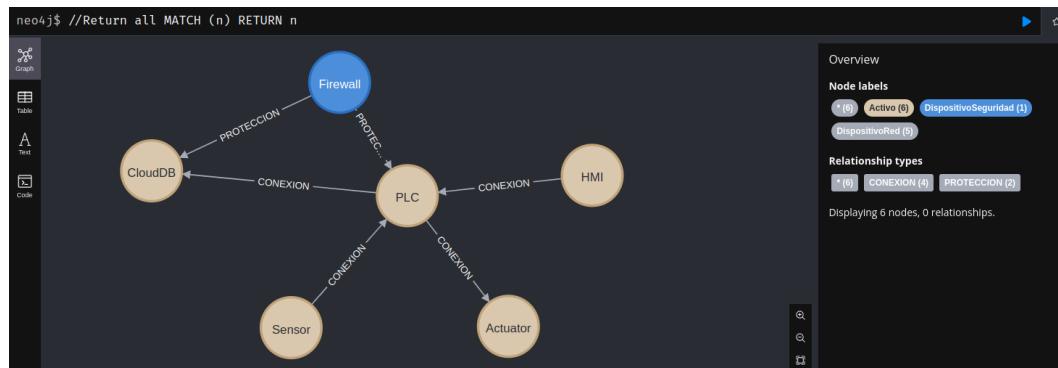


Figura 5.7: Escenario *Planta Industrial* en Neo4j

Entre ellos se encuentran el **Sensor** y el **Actuator**, ambos ejecutándose en una plataforma de tipo PRE (presumiblemente un entorno embebido o propietario de sistemas industriales). Estos dispositivos cuentan con permisos de usuario y capacidades específicas como la presencia física en el entorno y privilegios para acceder a determinados archivos y directorios. Estas características reflejan su función en la interacción directa con el entorno físico de la planta o sistema automatizado.

5.2. SIMULACIÓN DE ATAQUES SOBRE ESCENARIOS

El nodo **HMI** (Interfaz Hombre-Máquina) actúa como un punto de control para operadores humanos. Corre sobre una plataforma Windows y dispone de permisos administrativos. Sus capacidades incluyen el acceso a carpetas compartidas con permisos de escritura, así como el uso de servicios como Microsoft Core XML Services y acceso a la utilidad `wmic.exe`, lo que le permite monitorear y controlar otros dispositivos de la red.

Otro nodo relevante es el **PLC** (Controlador Lógico Programable), que también opera sobre una plataforma **PRE** y con privilegios de administrador. Posee acceso a archivos y directorios compartidos, reflejando su rol central en la lógica de control del sistema industrial.

La base de datos en la nube está representada por el nodo **CloudDB**, el cual corre en una plataforma tipo **SaaS** y tiene permisos administrativos. Se caracteriza por ofrecer endpoints de API que almacenan información de interés y capacidades para acceder a directorios y archivos que contienen datos críticos.

En términos de seguridad, el nodo **Firewall** actúa como un dispositivo de protección dentro de la red. Posee privilegios administrativos y capacidades relacionadas con la captura de paquetes y el monitoreo del tráfico en las interfaces de red.

En lo que respecta a las relaciones entre los nodos, el **Sensor** se comunica con el **PLC** utilizando el protocolo **Modbus**, al igual que el **Actuator**, que recibe instrucciones procesadas por el **PLC**. Además, la **HMI** también se conecta al **PLC** mediante el mismo protocolo. El **PLC**, a su vez, establece una conexión con la **CloudDB** mediante el protocolo **MQTT** sobre el puerto 1883, facilitando así el envío de datos a la nube para su almacenamiento o procesamiento posterior.

Finalmente, el **Firewall** filtra las conexiones a través de una relación de **PROTECCIÓN** tanto con el **PLC** como con la **CloudDB**, asegurando una defensa perimetral en esta arquitectura de red industrial extendida hacia la nube.

5.2. Simulación de ataques sobre escenarios

En esta sección se va a simular la ejecución de secuencias de técnicas de ataque sobre los escenarios cargados en la sección anterior. Esto permitirá analizar y evaluar el impacto de las técnicas sobre el escenario, dando lugar a la identificación de los puntos fuertes y débiles del mismo.

5.2.1. Ataque a escenario *Oficina Corporativa*

Si ahora se ejecuta el comando `$ python3 argos.py attack`, el sistema generará una secuencia aleatoria de técnicas y evaluará su impacto sobre el escenario cargado. Al ejecutar de nuevo la `query MATCH (n) RETURN (n)`, se puede apreciar que la secuencia de ataque generada ha sido insertada y que los nodos y relaciones correspondientes han sido creados sobre el escenario (Figura 5.8).

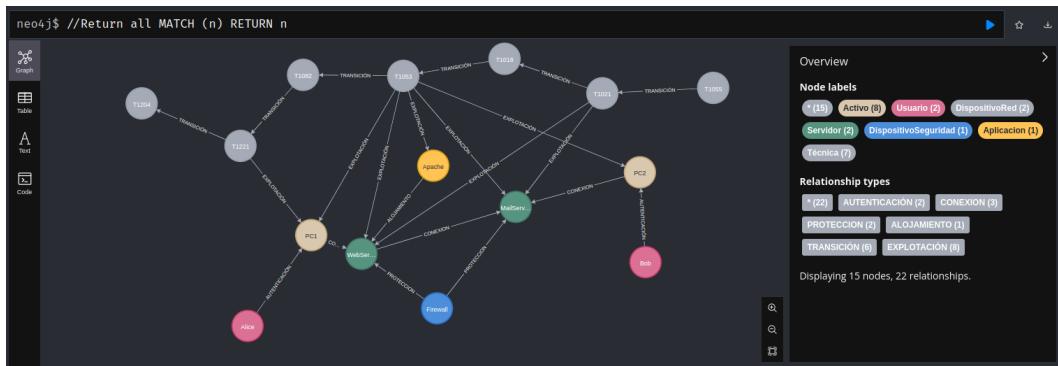


Figura 5.8: Ataque sobre el escenario de oficina corporativa

Se va a analizar una de las relaciones de explotación creadas, en concreto la que asocia a la técnica T1021 (uno de los nodos de color gris en la Figura 5.8) con el activo **WebServer** (representado en verde). Lo primero es verificar que los nodos se han creado correctamente, conteniendo los atributos que los caracterizan. El nodo **MailServer** representa el servidor de correo dentro de la infraestructura de red, es un nodo de tipo **Activo** y subtipo **Servidor**, sus atributos se muestran en la Figura 5.9.

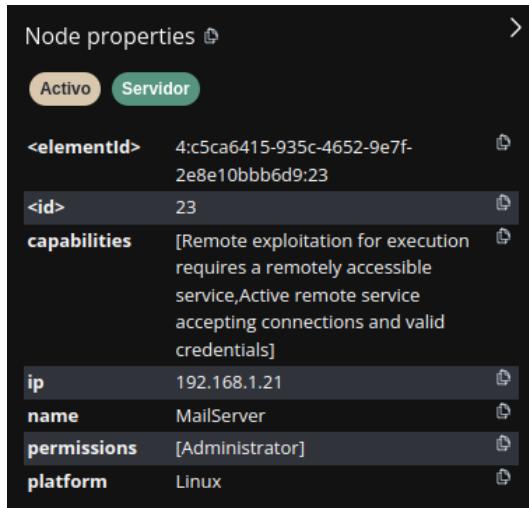


Figura 5.9: Propiedades del nodo **MailServer**

Por otro lado, el nodo T1021, de tipo **Técnica**, representa la técnica *Remote services*, que describe cómo los adversarios pueden usar cuentas válidas para acceder remotamente a sistemas dentro de una red empresarial que cuenta con un sistema de inicio de sesión centralizado. Si logran credenciales legítimas, los atacantes pueden iniciar sesión en múltiples dispositivos mediante protocolos como SSH o RDP, o incluso en servicios en la nube vinculados al dominio. Sus atributos se pueden apreciar en la Figura 5.10.

Al no haber ninguna vulnerabilidad detallada en el atributo **cve** de **MailServer**, el origen de la relación de **EXPLOTACIÓN** han de ser los parámetros indicados

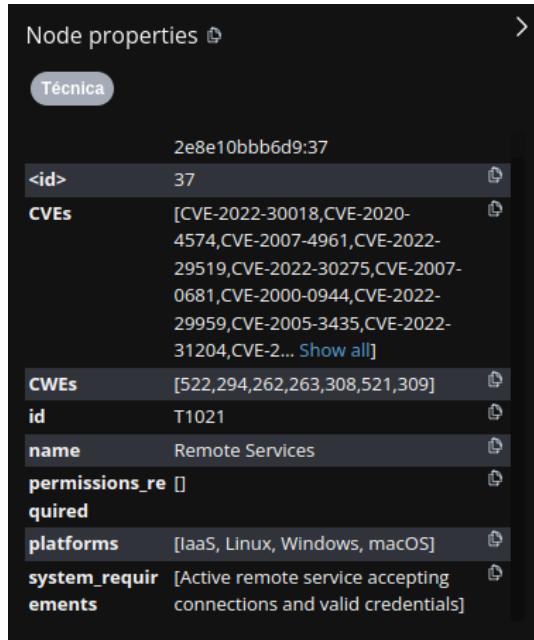


Figura 5.10: Propiedades del nodo T1021

en naranja en la Figura 4.16. Si se revisan, se puede apreciar que la plataforma que corre el servidor, Linux, es una de las plataformas afectadas por el sistema. Además, el sistema cuenta con la capacidad *Active remote service accepting connections and valid credentials*, uno de los requisitos necesarios por esta técnica para ser explotada. Finalmente, la técnica no requiere de ningún permiso en concreto para ser explotada con éxito. De hecho, esta técnica se clasifica en la matriz empresarial dentro de la táctica *Lateral Movement*, pues al obtener credenciales válidas de dominio, un atacante puede desplazarse dentro de la red accediendo a múltiples dispositivos y servicios sin necesidad de explotar vulnerabilidades adicionales. Se cumplen, por tanto, las tres condiciones para que exista una relación de tipo EXPLORACIÓN entre estos dos nodos.

A la hora de asociar la secuencia de ataque generada con los activos del escenario, el comando `attack` extrae simultáneamente una serie de métricas acerca del ataque que ha tenido lugar. Estas se presentan al terminar la ejecución a través de la terminal, organizadas en paneles gracias a la librería Rich. Este *dashboard* permite a los usuarios interpretar de manera rápida y eficaz los resultados del ataque simulado, y se puede observar en la Figura 5.11.

En dicha figura, se puede apreciar que el ataque ha sido puntuado con un nivel 50 de criticidad. La fórmula detrás de este cálculo se detalla en la Sección 4.3.2, que con los datos del ataque resultaría en lo siguiente:

$$C = \left[\left(\frac{5 \text{ activos afectados}}{8 \text{ activos totales}} \times 50 \right) + \left(\frac{3 \text{ técnicas exitosas}}{8 \text{ explotaciones}} \times 50 \right) \right] = 50$$

El número de activos vulnerados es ligeramente superior a la mitad de activos

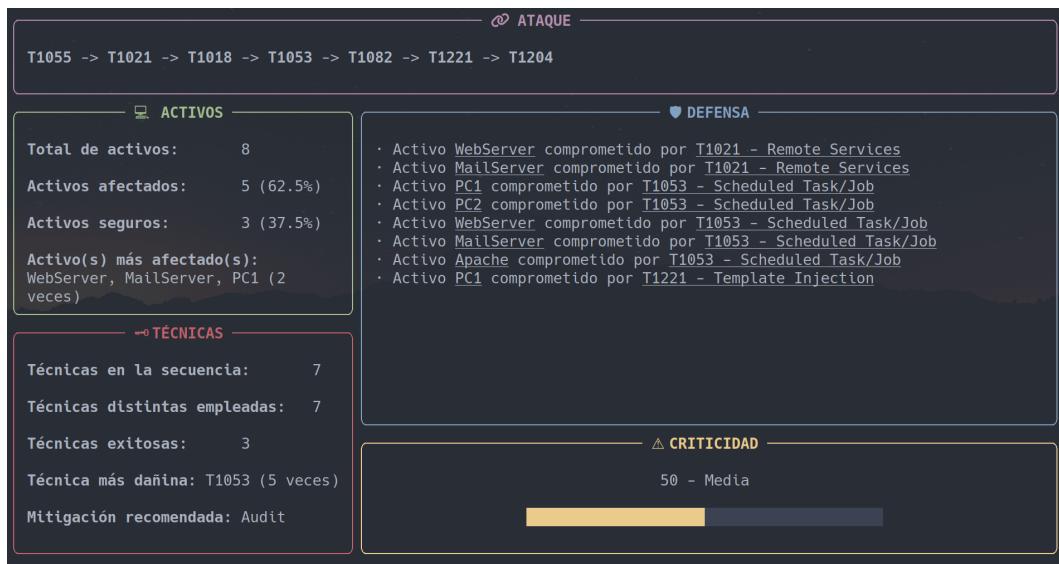


Figura 5.11: Resumen del ataque presentado sobre el escenario de oficina

totales, mientras que el número de técnicas distintas es ligeramente inferior a la mitad del total de explotaciones, lo cual explica que la criticidad asociada sea media.

Por último, se va a analizar la secuencia de técnicas que componen la cadena para analizar la coherencia del ataque. Las técnicas que componen la cadena se muestran a continuación en orden de ocurrencia:

1. *T1055 - Process Injection*: Los atacantes pueden injectar código en procesos en ejecución para evadir defensas basadas en procesos y, en algunos casos, escalar privilegios. Esta técnica permite ejecutar código arbitrario dentro del espacio de memoria de otro proceso activo.
2. *T1021 - Remote Services*: Los atacantes pueden usar cuentas válidas para iniciar sesión en servicios que aceptan conexiones remotas, como Telnet, SSH o VNC. Una vez conectados, pueden realizar acciones con los permisos del usuario autenticado.
3. *T1018 - Remote System Discovery*: Los atacantes pueden intentar identificar otros sistemas en la red mediante direcciones IP, nombres de host u otros identificadores, con el fin de preparar movimientos laterales desde el sistema actual.
4. *T1053 - Scheduled Task/Job*: Los atacantes pueden abusar de la funcionalidad de programación de tareas para ejecutar código malicioso, ya sea de forma inicial o recurrente. Todos los sistemas operativos principales permiten programar la ejecución de programas o scripts, incluso en sistemas remotos si se cuenta con la autenticación adecuada. Esta técnica es, además, la más letal del ataque, y para ella se sugiere como mitigación *Audit*. El propósito principal de la auditoría es detectar anomalías

5.2. SIMULACIÓN DE ATAQUES SOBRE ESCENARIOS

e identificar posibles amenazas o debilidades en el entorno mediante la revisión de las actividades y configuraciones del sistema.

5. T1082 - *System Information Discovery*: El atacante puede buscar información detallada sobre el sistema operativo y el hardware, como versión, parches, actualizaciones y arquitectura. Esta información ayuda a decidir los siguientes pasos del ataque, como la infección completa del objetivo o la ejecución de acciones específicas.
6. T1221 - *Template Injection*: Los atacantes pueden crear o modificar referencias en plantillas de documentos de usuario para ocultar código malicioso o provocar intentos de autenticación, aprovechando funciones legítimas del software.
7. T1204 - *User Execution*: El atacante depende de que el usuario realice una acción específica, como abrir un archivo o enlace malicioso, generalmente tras una campaña de ingeniería social o phishing, para lograr la ejecución de su código malicioso. Cabe recordar que esta técnica es una de las que se había definido como posibles técnicas absorbentes, lo cual explica su posición como técnica de cierre. Esta posición es completamente coherente dado que se trata de una técnica perteneciente a la táctica *Execution*. La ejecución consiste en técnicas que permiten que código controlado por el adversario se ejecute en un sistema local o remoto, lo cual suele ser uno de los últimos movimientos en un ciberataque.

El ataque comienza con una inyección de procesos (T1055), lo cual sugiere una preocupación temprana por la evasión de defensas. Al inyectar código en procesos legítimos, el atacante logra esconder su actividad y posiblemente adquirir mayores privilegios sin ser detectado por soluciones de seguridad convencionales como antivirus o EDR.

Una vez consolidado el acceso inicial, el adversario procede a explotar servicios remotos (T1021), utilizando credenciales válidas para conectarse a otros sistemas dentro de la red. Este movimiento lateral es perfectamente viable en redes de oficina, donde el acceso remoto a estaciones de trabajo y servidores está habilitado para tareas administrativas o de soporte. En este punto, el atacante probablemente ya tiene acceso a sistemas clave o cuentas con privilegios suficientes para avanzar.

Para maximizar su efectividad, realiza un descubrimiento de sistemas remotos (T1018), buscando identificar otros dispositivos disponibles en la red, lo cual es crítico para decidir a qué activos moverse a continuación. Esta fase de reconocimiento refuerza la intención de expandir su presencia dentro del entorno corporativo.

Luego, el uso de *Scheduled Task/Job* (T1053) indica un interés en establecer persistencia o automatizar la ejecución de tareas maliciosas, ya sea para mantener el acceso tras reinicios o para lanzar comandos de manera programada sin intervención manual. La combinación de persistencia y automatización en

este punto es clave para mantener el control sin levantar sospechas.

A continuación, se realiza un descubrimiento de información del sistema (T1082), lo que sugiere que el atacante está evaluando las características de los sistemas comprometidos. Tras esto, el ataque incorpora la técnica *Template Injection* (T1221), lo cual indica un intento de explotar documentos comunes en el entorno de oficina, como archivos de Word o Excel, que utilizan plantillas. Al inyectar código en una plantilla, el atacante puede propagar su carga útil de manera sutil entre documentos compartidos, muy común en entornos colaborativos.

Finalmente, *User Execution* (T1204) cierra la cadena, lo que indica que parte del éxito del ataque depende de la interacción del usuario, probablemente como resultado de ingeniería social. Este paso sugiere que se buscó engañar a los empleados para que abrieran un archivo o hicieran clic en un enlace, activando así el componente final del ataque.

La secuencia es coherente y refleja un enfoque metódico: evasión → movimiento lateral → reconocimiento → persistencia → evaluación → propagación → ejecución final. El uso de técnicas como la inyección de plantillas y ejecución por el usuario revela un entendimiento del entorno de oficina y de los comportamientos típicos del personal, lo cual es coherente con el escenario sobre el que se está realizando el ataque.

5.2.2. Ataque a escenario *Smart Home*

Para el escenario *Smart Home*, el ataque se llevará a cabo mediante el comando **trace**, de forma que será necesario especificar una técnica inicial para la cadena.

Tras ejecutar el comando `$ python3 argos.py trace`, se mostrará por pantalla una lista con las técnicas iniciales disponibles en el sistema, ordenadas según su identificador, y se solicitará la introducción de una de ellas, tal y como refleja la Figura 5.12. Si se selecciona una técnica inexistente, se detendrá la ejecución del programa, indicando el mensaje de error visible en la Figura 5.13.

Si por el contrario se selecciona un estado válido, es decir, uno de los mostrados en la lista, la ejecución transcurrirá con normalidad, de la misma manera que en el caso anterior con el comando **attack**. En este caso se ha seleccionado como técnica inicial la técnica **T1484 - Domain or Tenant Policy Modification**. Utilizando esta técnica, los adversarios pueden modificar la configuración de un dominio o inquilino de identidad para evadir defensas y/o escalar privilegios en entornos gestionados de forma centralizada. Esto afecta a aquellos servicios que ofrecen una forma centralizada de administrar recursos de identidad como dispositivos y cuentas, y que suelen incluir configuraciones que pueden aplicarse entre dominios o inquilinos, como relaciones de confianza, sincronización de identidades o federación de identidades.

5.2. SIMULACIÓN DE ATAQUES SOBRE ESCENARIOS

```
~/TFM/argos >>> python3 argos.py trace

[+] Técnicas disponibles:

T1001 - Data Obfuscation
T1005 - Direct Volume Access
T1010 - Application Windows Discovery
T1016 - Application Network Configuration Discovery
T1021 - Remote Services
T1029 - Scheduled Transfer
T1036 - Masquerading
T1040 - Network Sniffing
T1047 - Windows Management Instrumentation
T1052 - Exfiltration Over Physical Medium
T1056 - Input Capture
T1068 - Exploitation for Privilege Escalation
T1072 - Application Layer Protocol
T1078 - Valid Accounts
T1083 - File and Directory Discovery
T1091 - Replication Through Removable Media
T1099 - Account Manipulation
T1105 - Ingress Tool Transfer
T1111 - Multi-Factor Authentication Interception
T1114 - Email Collection
T1120 - Peripheral Device Discovery
T1130 - Video Capture
T1134 - Account Manipulation
T1137 - Office Application Startup
T1187 - Forced Authentication
T1195 - Supply Chain Compromise
T1200 - Hardware Additions
T1203 - Exploitation for Client Execution
T1210 - Exploitation of Remote Services
T1216 - System Script Proxy Execution
T1219 - Remote Access Software
T1223 - Persistence Credentials
T1405 - Data Destruction
T1490 - Inhibit System Recovery
T1497 - Virtualization/Sandbox Evasion
T1505 - Server Software Component
T1529 - System Shutdown/Reboot
T1534 - Internal Spearphishing
T1542 - Pre-OS Boot
T1547 - Boot or Logon Autostart Execution
T1552 - Unsecured Credentials
T1553 - Credentials from Password Stores
T1558 - Steel or Forge Kerberos Tickets
T1561 - Disk Wipe
T1564 - Hide Artifacts
T1567 - Exfiltration Over Web Service
T1570 - Lateral Tool Transfer
T1573 - Encrypted Channel
T1580 - Cloud Infrastructure Discovery
T1585 - Establish Accounts
T1587 - Obtain Capabilities
T1591 - Search Victim-Defined Information
T1594 - Search Victim-Defined Websites
T1597 - Search Closed Sources
T1608 - Stage Capabilities
T1611 - Escape to Host
T1615 - Group Policy Discovery
T1622 - Debugger Evasion
T1654 - Log Enumeration
T1659 - Content Injection

| T1003 - OS Credential Dumping
| T1007 - System Service Discovery
| T1012 - Query Registry
| T1013 - Application Network Discovery
| T1025 - Data from Removable Media
| T1030 - Data Transfer Size Limits
| T1037 - Boot or Logon Initialization Scripts
| T1041 - Exfiltration Over C2 Channel
| T1048 - Exfiltration Over Alternative Protocol
| T1053 - Scheduled Task/Job
| T1057 - Process Discovery
| T1060 - Permission Groups Discovery
| T1063 - Software Configuration Tools
| T1080 - Saint Shared Content
| T1087 - Account Discovery
| T1092 - Communication Through Removable Media
| T1102 - Web Service
| T1106 - Native API
| T1112 - Modify Registry
| T1115 - Clipboard Data
| T1123 - Audio Capture
| T1132 - Data Encoding
| T1135 - Network Share Discovery
| T1140 - Decompress/Decode Files or Information
| T1180 - Drive-by Compromise
| T1197 - BITS Jobs
| T1201 - Password Policy Discovery
| T1204 - User Execution
| T1211 - Exploitation for Defense Evasion
| T1217 - Browser Information Discovery
| T1223 - Template Injection
| T1482 - Domains Trust Discovery
| T1486 - Exploit for Impact
| T1491 - Defacement
| T1498 - Network Denial of Service
| T1518 - Software Discovery
| T1530 - Data from Cloud Storage
| T1538 - Cloud Service Dashboard
| T1543 - Create or Modify System Process
| T1548 - Abuse Elevation Control Mechanism
| T1553 - Subvert Trust Controls
| T1556 - Modify Application Process
| T1559 - Impair Process Communication
| T1562 - Impair Defenses
| T1565 - Data Manipulation
| T1568 - Dynamic Resolution
| T1571 - Non-Standard Port
| T1574 - Hijack Execution Flow
| T1583 - Acquire Infrastructure
| T1586 - Compromise Accounts
| T1589 - Gather Victim Identity Information
| T1592 - Extract Host Information
| T1595 - Active Scanning
| T1598 - Phishing for Information
| T1600 - Container Administration Command
| T1613 - Container and Resource Discovery
| T1620 - Reflective Code Loading
| T1649 - Steel or Forge Authentication Certificates
| T1655 - Impersonation
| T1665 - Hide Infrastructure

| T1005 - Data from Local System
| T1008 - Fallback Channels
| T1014 - Rootkit
| T1019 - Advanced Efiltration
| T1027 - Obfuscated Files or Information
| T1033 - System Owner/User Discovery
| T1039 - Data from Network Shared Drive
| T1046 - Network Service Discovery
| T1049 - System Network Connections Discovery
| T1055 - Process Injection
| T1059 - Command and Scripting Interpreter
| T1070 - Indicator Removal
| T1074 - Multi-Stage
| T1082 - System Information Discovery
| T1090 - Proxy
| T1095 - Non-Application Layer Protocol
| T1104 - Multi-Stage Channels
| T1110 - Brute Force
| T1113 - Screen Capture
| T1119 - Automated Collection
| T1124 - System Time Discovery
| T1130 - Exploit Remote Services
| T1136 - Create Account
| T1176 - Browser Extensions
| T1190 - Exploit Public-Facing Application
| T1199 - Trusted Relationship
| T1202 - Indirect Command Execution
| T1205 - Traffic Signaling
| T1213 - Data from Information Repositories
| T1218 - System Binary Proxy Execution
| T1221 - File and Directory Permissions Modification
| T1284 - Create or Modify Tenant Policy Modification
| T1489 - Service Stop
| T1496 - Resource Hijacking
| T1499 - Endpoint Denial of Service
| T1528 - Steal Application Access Token
| T1531 - Account Access Removal
| T1539 - Steal Web Session Cookie
| T1546 - Event Triggered Execution
| T1550 - Use Alternate Authentication Material
| T1554 - Compromise Host Software Binary
| T1562 - Impair Application-in-the-Middle
| T1568 - Archive Collected Data
| T1563 - Remote Service Session Hijacking
| T1566 - Phishing
| T1569 - System Services
| T1572 - Protocol Tunneling
| T1578 - Modify Cloud Compute Infrastructure
| T1584 - Compromise Infrastructure
| T1587 - Develop Capabilities
| T1593 - Gather Victim Network Information
| T1596 - Search Open Technical Domains
| T1598 - Search Open Technical Databases
| T1606 - Forge Web Credentials
| T1610 - Deploy Container
| T1614 - System Location Discovery
| T1621 - Multi-Factor Authentication Request Generation
| T1651 - Cloud Administration Command
| T1657 - Financial Theft

Ingrese el estado inicial: |
```

Figura 5.12: Posibles estados iniciales disponibles en el sistema proporcionados por el comando `trace`

```
Ingrese el estado inicial: T999

[+] El estado inicial no es válido. Por favor, elija una técnica válida.

~/TFM/argos >>> |
```

Figura 5.13: Selección de estado inicial erróneo

Una vez seleccionada una técnica válida, se generará una cadena de ataque a partir de ella. Esta cadena se visualizará en la consola de Neo4j tras al ejecutar la query `Return all`, resultando en la secuencia de la Figura 5.14.

Se puede apreciar que la secuencia generada es larga, compuesta por un total de 22 técnicas, de las cuales 6 han resultado exitosas. De entre las técnicas exitosas, la más dañina ha resultado ser la técnica T1056, afectando a los activos `Printer`, `CCTV`, `Smartphone` y `Router`. Mediante el uso de la técnica T1056, llamada *Input Capture*, los adversarios pueden emplear métodos de captura de entrada del usuario para obtener credenciales o recopilar información. Durante el uso normal del sistema, los usuarios suelen introducir credenciales en diversos lugares, como páginas de inicio de sesión, portales o cuadros de diálogo del sistema. Los mecanismos de captura de entrada pueden ser invisibles para el usuario (por ejemplo, enganche de API de credenciales) o basarse en engañar

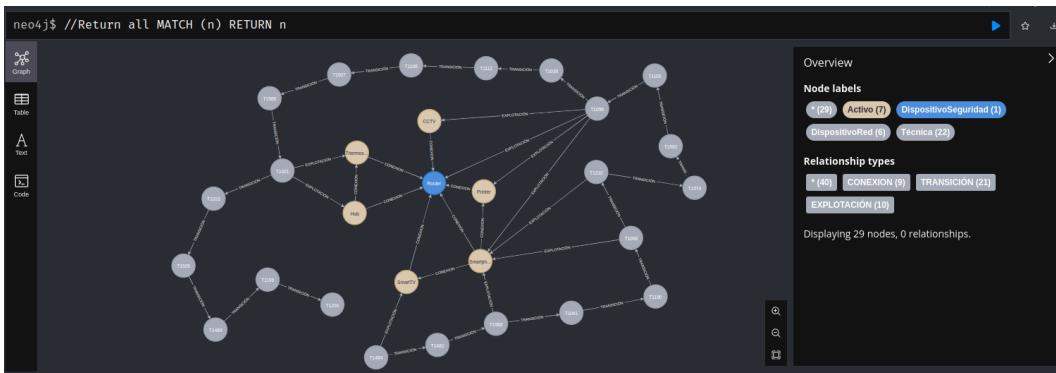


Figura 5.14: Ataque sobre el escenario *Smart Home*

al usuario para que introduzca información en lo que cree que es un servicio legítimo (por ejemplo, captura a través de un portal web falso). Los atributos correspondientes a esta técnica se presentan en la Figura 5.15.

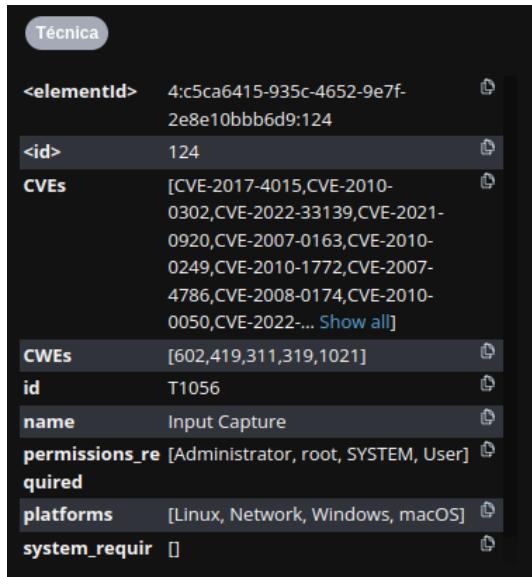


Figura 5.15: Propiedades del nodo T1056

Por ser el elemento central de la red, y por tanto el más crítico, se va a analizar la relación de EXPLOTACIÓN con el activo Router, cuyos atributos se detallan en la Figura 5.16. Al igual que en el caso anterior, este nodo no cuenta con ningún CVE asociado. Observando el resto de parámetros, se aprecia que la plataforma sobre la que opera este dispositivo es *Network*, una de las plataformas vulneradas por la técnica T1056. Además, este activo cuenta con permisos de administrador, otro de los requerimientos de la técnica para ser efectiva.

Finalmente, el activo cuenta con diversas capacidades de red, aunque se puede observar que la lista **system_requirements** del nodo T1056 no especifica ninguna capacidad en concreto, por lo que cualquier capacidad disponible será considerada como válida.

5.2. SIMULACIÓN DE ATAQUES SOBRE ESCENARIOS

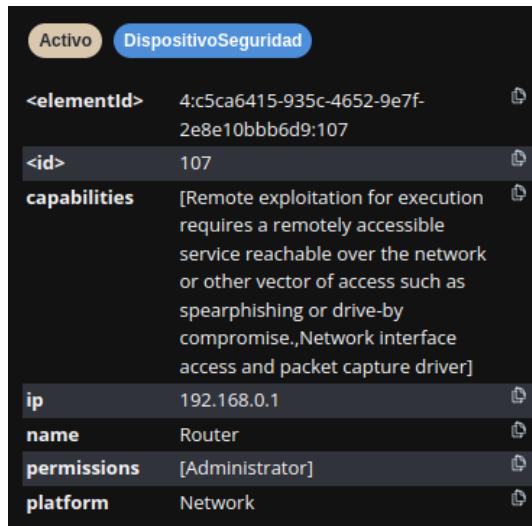


Figura 5.16: Propiedades del nodo Router

Todo esto conlleva que, en efecto, se produzca una relación de EXPLORACIÓN entre ambos nodos. En la Figura 5.17 se presenta el *dashboard* que resume la ejecución de este ataque.

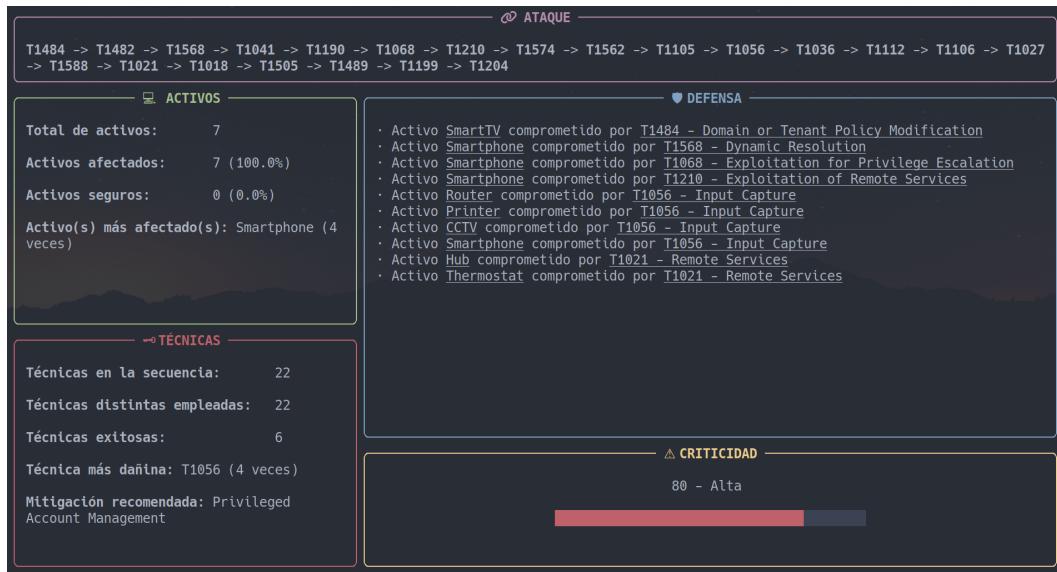


Figura 5.17: Resumen del ataque presentado sobre el escenario *Smart Home*

La criticidad de este ataque ha sido evaluada como alta con una puntuación de 80. Este valor se corresponde con la gravedad del ataque, ya que la totalidad de los activos han sido vulnerados mediante el empleo de hasta 6 técnicas distintas. El cálculo correspondiente sería el siguiente:

$$C = \left[\left(\frac{7 \text{ activos afectados}}{7 \text{ activos totales}} \times 50 \right) + \left(\frac{6 \text{ técnicas exitosas}}{10 \text{ explotaciones}} \times 50 \right) \right] = 80$$

5.2.3. Ataque a escenario *Planta Industrial*

La secuencia obtenida como resultado de ejecutar el comando `attack` sobre el escenario *Planta Industrial* se muestra en la Figura 5.18.

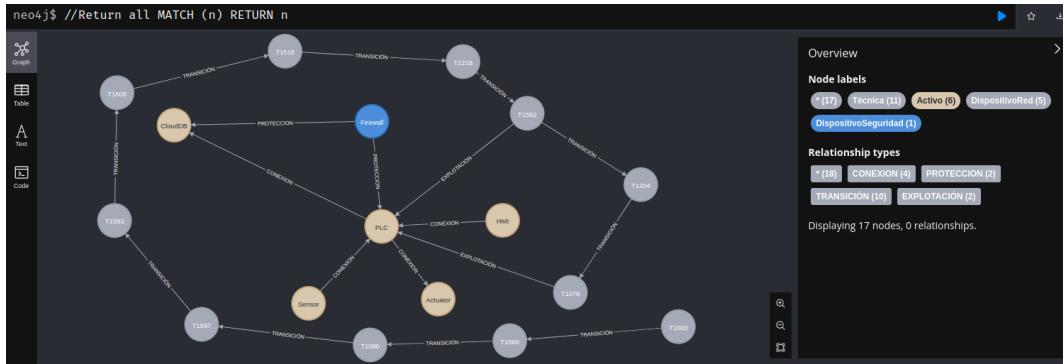


Figura 5.18: Ataque sobre el escenario *Planta Industrial*

Se puede apreciar que hay un total de 2 técnicas exitosas distintas, que además comprometen el mismo activo. Se va a analizar una de estas relaciones de explotación, en concreto la provocada por la técnica T1552.

El activo comprometido es el PLC, que juega un papel fundamental en la lógica de control del sistema industrial. Este nodo, de tipo *DispositivoRed*, presenta los atributos de la Figura 5.19.

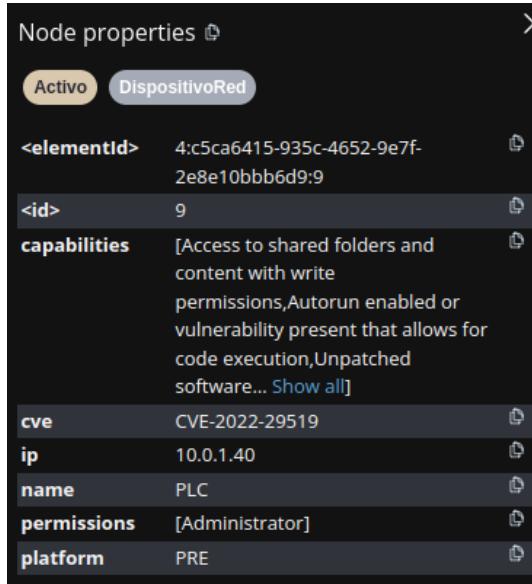


Figura 5.19: Propiedades del nodo PLC

La técnica que compromete este dispositivo se denomina *Unsecured Credentials*. Los atacantes que emplean esta técnica pueden buscar en sistemas comprometidos para encontrar y obtener credenciales almacenadas de manera insegura. Estas credenciales pueden estar almacenadas y/o mal ubicadas en muchos

5.2. SIMULACIÓN DE ATAQUES SOBRE ESCENARIOS

lugares de un sistema, incluidos archivos en texto claro, repositorios específicos del sistema operativo o la aplicación, o en otros archivos/artifazos especializados.

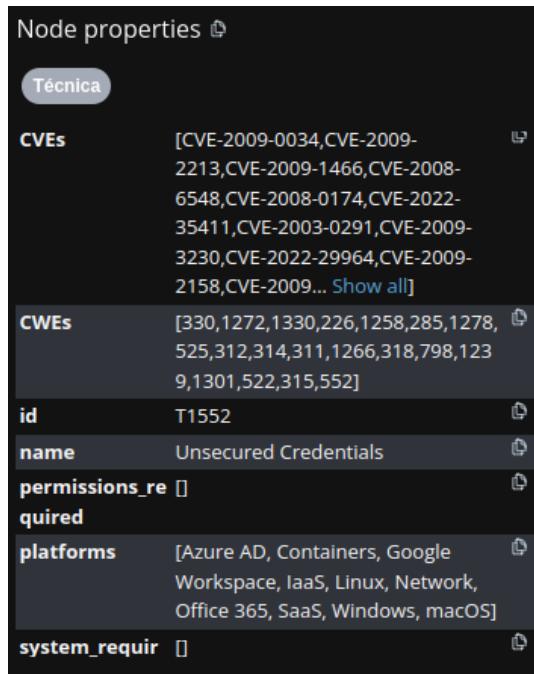


Figura 5.20: Propiedades del nodo T1552

A simple vista se observa que, en este caso, el criterio que determinaba la relación de explotación en los ejemplos anteriores no aplica, ya que la plataforma PRE en la que opera el PLC no es una de las explotables por la técnica. Al no cumplirse esta condición, la única forma de que exista una relación de EXPLORACIÓN entre estos nodos es la existencia de una vulnerabilidad concreta en el activo que además sea explotable por la técnica. De hecho, como ya se mencionó en la Sección 4.1.2 a la hora de presentar el escenario, el controlador PLC presenta la vulnerabilidad CVE-2022-29519, que en determinadas versiones de los controladores STARDOM FCN Controller y FCJ Controller podría permitir que un atacante obtuviese credenciales en texto claro.

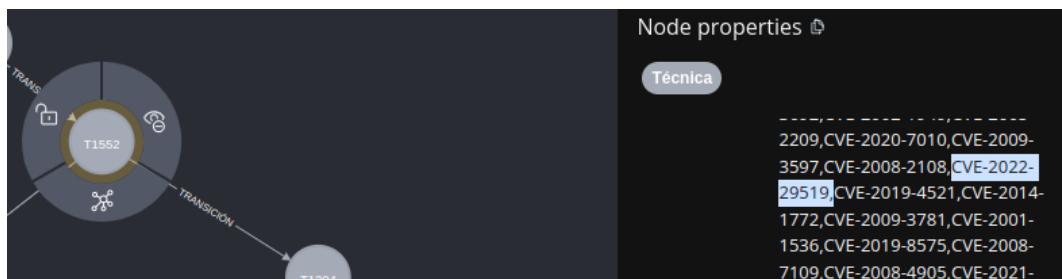


Figura 5.21: Identificación del CVE afectado en el nodo T1552

En la Figura 5.21 se puede ver como, seleccionando este nodo y expandiendo la

lista de CVEs que explota, se encuentra el valor CVE-2022-29519, que coincide con el valor que toma el parámetro `cve` para el nodo PLC.

Por último, el resumen del ataque ejecutado se presenta en la Figura 5.22.

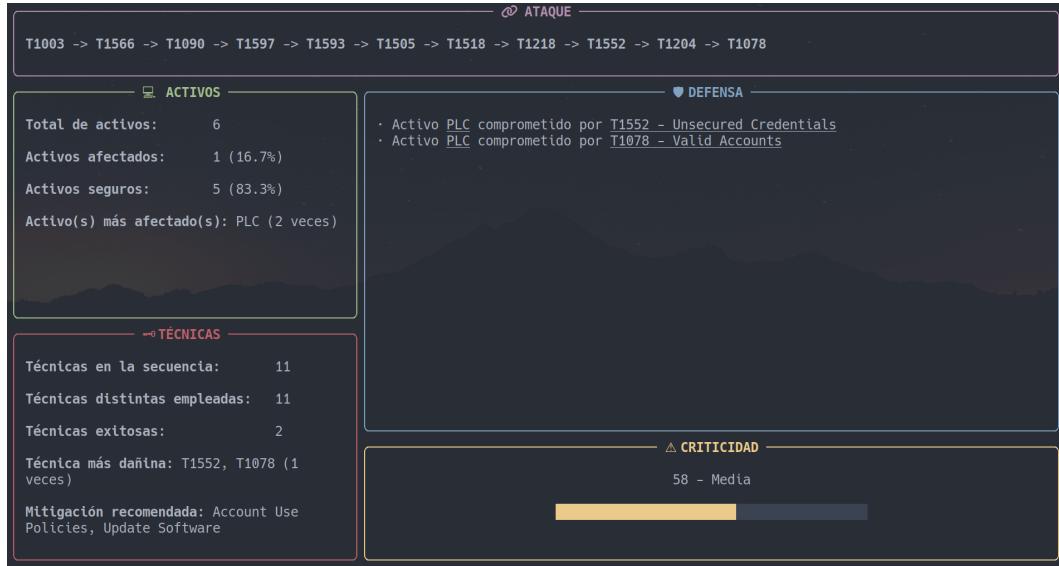


Figura 5.22: Resumen del ataque presentado sobre el escenario *Planta Industrial*

Las mitigaciones recomendadas son *Account Use Policies* para la técnica *Valid Accounts* (esta técnica permite a los atacantes obtener y abusar de las credenciales de cuentas existentes como medio para lograr acceso inicial, persistencia, escalada de privilegios o evasión de defensas), pues el uso de políticas de acceso condicional permite bloquear inicios de sesión desde dispositivos no compatibles o desde direcciones IP que estén fuera de los rangos definidos; y *Update Software* para la técnica *Unsecured Credentials*, ya que la aplicación de parches de seguridad impide que las credenciales se almacenen o transporten de manera insegura.

Finalmente, la criticidad asignada al ataque vuelve a ser media, con una puntuación de 58, provocada por 1 activo afectado (de los 6 que componen el escenario) por 2 técnicas exitosas.

$$C = \left[\left(\frac{1 \text{ activo afectado}}{6 \text{ activos totales}} \times 50 \right) + \left(\frac{2 \text{ técnicas exitosas}}{2 \text{ explotaciones}} \times 50 \right) \right] = 58$$

5.3. Borrado de la base de datos

Una vez se ha ejecutado un ataque, es deseable vaciar la base de datos antes de realizar una nueva ejecución. Para ello se ejecuta el siguiente comando:

```
$ python3 argos.py clean
```

5.4. HISTORIAL DE ATAQUES

Una vez ejecutado, el programa indica que la base de datos ha sido vaciada.

```
~/TFM/argos >>> python3 argos.py clean
[+] Database cleared
```

Figura 5.23: Limpieza exitosa de la base de datos

Tras este mensaje, al igual que en los ejemplos anteriores, habría que desplazarse a la consola de Neo4j y ejecutar la consulta *Return all*. Esta query devolverá todos los nodos y relaciones presentes en la base de datos, que en este caso no existen.

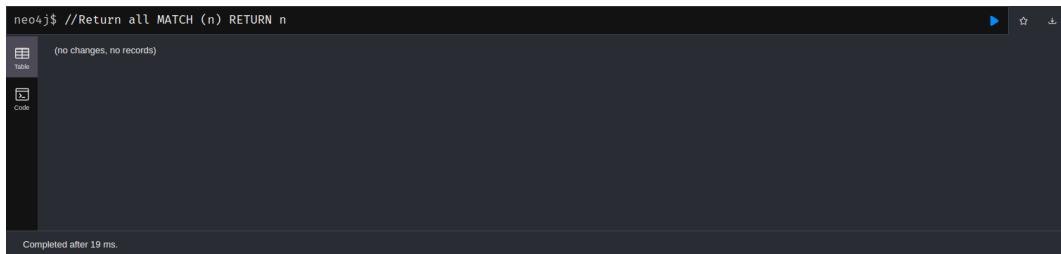


Figura 5.24: Base de datos vacía tras la ejecución del comando `clean`

5.4. Historial de ataques

Con cada ejecución de un ataque, una selección de las estadísticas mostradas en el *dashboard* se almacenan de forma permanente. El histórico de ataques ejecutados se puede consultar a través del siguiente comando:

```
$ python3 argos.py history
```

Acto seguido, se muestra en la terminal una tabla resumen de todos los ataques que se han llevado a cabo, tal y como muestra la Figura 5.25.

Historial de ataques				
ID Ataque	Escenario	Activo Más Afectado	Técnica más exitosa	Fecha de ejecución
Ataque-3730	industrial	PLC	T1021, T1078	2025-04-10 20:12:20
Ataque-7863	oficina	PC1	T1052	2025-04-10 20:12:30
Ataque-3218	smarthome	Hub, Thermostat	T1021	2025-04-10 20:15:41
Ataque-6304	oficina	PC1, WebServer, MailServer	T1021	2025-04-10 20:16:04
Ataque-5094	oficina	PC1	T1080	2025-04-10 20:16:06
Ataque-8807	oficina	PC1, PC2, WebServer, MailServer	T1119, T1021	2025-04-10 20:16:08
Ataque-9939	oficina	WebServer, MailServer	T1021	2025-04-10 20:16:14
Ataque-4159	industrial	PLC	T1078	2025-04-10 20:16:32
Ataque-1567	industrial	PLC	T1005, T1021	2025-04-10 20:16:34
Ataque-4828	industrial	PLC	T1552	2025-04-10 20:16:36

Figura 5.25: Historial de ataques realizados

Capítulo 6

Conclusiones y líneas futuras

Este capítulo presenta las conclusiones extraídas a raíz de la realización del proyecto y las líneas futuras en las que se puede seguir trabajando.

6.1. Conclusiones

El análisis de las rutas de ataque es uno de los puntos clave en la investigación de ciberseguridad actualmente. Para ello, la visualización del camino que este ataque puede recorrer dentro de una red resulta una herramienta eficaz para el estudio de los ciberataques con el objetivo de prevenirlos o mitigar su efecto dentro de entornos organizacionales. Al proporcionar una representación gráfica de los posibles recorridos de un atacante dentro de una red o sistema, se contribuye a la identificación de vulnerabilidades y la toma de decisiones estratégicas en seguridad, objetivos clave de los entornos de conciencia cibersituacional.

Los resultados obtenidos con ARGOS reflejan una mejora significativa en la eficacia del sistema propuesto frente a métodos tradicionales. A través de las pruebas experimentales realizadas, se evidencia un gran desempeño tanto en precisión como en eficiencia operativa, lo que demuestra el potencial del enfoque para su aplicación práctica. Además, se observa una consistencia en los resultados a lo largo de diferentes escenarios evaluados, lo que valida la robustez de la propuesta.

Estos resultados se complementan con el desarrollo de una estrategia de seguridad que permita abordar la respuesta frente a las amenazas a raíz de la caracterización de las técnicas utilizadas. El análisis de las rutas de ataque desde el punto de vista planteado en ARGOS permite identificar activos vulnerables, el impacto de los ataques y las contramedidas más eficaces; mientras que el estudio de los ataques para su caracterización, en este contexto, permite definir modelos más adaptados al comportamiento de los atacantes y que generen simulaciones más realistas.

6.2. Líneas futuras

Con el objetivo de mejorar la funcionalidad y el alcance de ARGOS, se presentan a continuación diversas líneas de desarrollo que podrían implementarse en versiones futuras de la herramienta. La implementación de estas mejoras permitiría que ARGOS se convierta en una herramienta más robusta y adaptable a las necesidades de análisis de seguridad en entornos de red.

Actualmente, ARGOS permite la carga de escenarios de red definidos mediante ficheros Cypher, pero la complejidad de estos escenarios podría ampliarse. Se podría considerar la integración de configuraciones más detalladas, incluyendo topologías de red dinámicas, segmentación de redes, o la representación de defensas activas como sistemas de detección de intrusiones (IDS) y *firewalls*.

La simulación de ataques mediante Cadenas de Markov podría beneficiarse de una mayor personalización en la generación de secuencias de TTPs. Por ejemplo, podría incorporarse un sistema de pesos dinámicos basado en inteligencia de amenazas en tiempo real, permitiendo que las probabilidades de transición entre técnicas cambien en función de tendencias recientes en ataques reales. Otra posible mejora sería la incorporación de modelos probabilísticos más avanzados, como redes bayesianas, para mejorar larealismo de las transiciones entre estados.

En el estado actual, la herramienta almacena métricas sobre los ataques ejecutados, pero se podría ampliar la capacidad de análisis mediante la generación de informes automáticos con visualizaciones interactivas. El uso de librerías de análisis de datos como Pandas o herramientas de visualización como Dash podría permitir una exploración más intuitiva del impacto de los ataques en los activos del sistema.

Por último, el almacenamiento del historial de ataques podría enriquecerse con un sistema de versionado que permitiera comparar la evolución de escenarios a lo largo del tiempo. Además, se podría incluir un módulo de aprendizaje automático para identificar patrones en ataques previos y predecir posibles rutas de ataque futuras.

Bibliografía

- [1] Check Point. *Cyberattacks targetting governments*. Último acceso: 17 de febrero de 2025. URL: <https://www.checkpoint.com/es/cyber-hub/cyber-security/what-is-cybersecurity-for-governments/cyberattacks-targeting-governments/>.
- [2] Telefónica Tech. *Tendencias en Ciberseguridad para 2025*. Último acceso: 6 de febrero de 2025. URL: <https://telefonicatech.com/blog/tendencias-en-ciberseguridad-para-2025>.
- [3] Plataforma Tecnológica Española de Tecnologías Disruptivas. *Informe de situación 2024*. Último acceso: 6 de febrero de 2025. URL: https://ptedisruptive.es/wp-content/uploads/2024/12/INFORME-DE-SITUACION_CIBERSEGURIDAD_2024.pdf.
- [4] The MITRE Corporation. *MITRE ATT&CK*. Último acceso: 10 de enero de 2025. URL: <https://attack.mitre.org/>.
- [5] IBM. *¿Qué es el marco MITRE ATT&CK?* Último acceso: 10 de enero de 2025. URL: <https://www.ibm.com/es-es/topics/mitre-attack>.
- [6] The MITRE Corporation. *Enterprise Matrix*. Último acceso: 6 de febrero de 2025. URL: <https://attack.mitre.org/matrices/enterprise/>.
- [7] Red Hat. *El concepto de CVE*. Último acceso: 21 de marzo de 2025. URL: <https://www.redhat.com/es/topics/security/what-is-cve>.
- [8] Parasoft. *Todo sobre CWE: enumeración de debilidades comunes*. Último acceso: 21 de marzo de 2025. URL: <https://es.parasoft.com/blog/what-is-cwe/>.
- [9] The MITRE Corporation. *CAPEC - Common Attack Pattern Enumerations and Classifications*. Último acceso: 21 de marzo de 2025. URL: <https://capec.mitre.org/>.
- [10] Knowledge Consulting Group. *Agenda Introductions Common Attack Pattern Enumeration and Classification (CAPEC) Common Weakness Enumeration (CWE) Penetration Testing: Real-world Examples*. Último acceso: 23 de abril de 2025. URL: <https://slideplayer.com/slide/4316286/>.
- [11] S. M. Ross. *Introduction to Probability Models*. Último acceso: 23 de abril de 2025. URL: <https://www-elec.inaoep.mx/~rogerio/IntrodProbabModels.pdf>.
- [12] Wikipedia. *Cadena de Markov*. Último acceso: 6 de febrero de 2025. URL: https://es.wikipedia.org/wiki/Cadena_de_M%C3%A1rkov.

- [13] UC3M. *Cadenas de Markov*. Último acceso: 30 de enero de 2025. URL: <https://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/PEst/tema4pe.pdf>.
- [14] Becca Gomby. *What Is An Attack Path & How Does It Help Identify Risks?* Último acceso: 30 de enero de 2025. URL: <https://www.panoptica.app/blog/what-is-an-attack-path-how-does-it-help-identify-risks>.
- [15] Cloudflare. *¿Qué es un vector de ataque?* Último acceso: 6 de febrero de 2025. URL: <https://www.cloudflare.com/es-es/learning/security/glossary/attack-vector/>.
- [16] Cloudflare. *¿Qué es una superficie de ataque?* Último acceso: 6 de febrero de 2025. URL: <https://www.cloudflare.com/es-es/learning/security/what-is-an-attack-surface/>.
- [17] Fidelis Security. *Proactive AD Security: Leveraging Risk Assessment and Attack Path Analysis*. Último acceso: 30 de enero de 2025. URL: <https://fidelissecurity.com/threatgeek/active-directory-security/active-directory-risk-assessment-and-attack-path-analysis/>.
- [18] Picus Labs. *What Is Attack Path Visualization?* Último acceso: 30 de enero de 2025. URL: <https://www.picussecurity.com/resource/glossary/what-is-attack-path-visualization>.
- [19] Ryan Boyd Michael Hunger y William Lyon (Neo4j blog). *RDBMS & Graphs: Relational vs. Graph Data Modeling*. Último acceso: 21 de marzo de 2025. URL: <https://neo4j.com/blog/developer/rdbms-vs-graph-data-modeling/>.
- [20] Neo4j. *Neo4j Graph Database & Analytics — Graph Database Management System*. Último acceso: 30 de enero de 2025. URL: <https://neo4j.com/>.
- [21] Textualize. *Rich (official GitHub repository)*. Último acceso: 30 de enero de 2025. URL: <https://github.com/Textualize/rich>.
- [22] The MITRE Corporation. *Common Weakness Enumeration*. Último acceso: 21 de marzo de 2025. URL: <https://cwe.mitre.org/data/downloads.html>.
- [23] The MITRE Corporation. *Excel Spreadsheets representing the ATT&CK dataset*. Último acceso: 21 de marzo de 2025. URL: https://attack.mitre.org/resources/attack-data-and-tools/?trk=public_post_comment-text#excel-attack.
- [24] INCIBE. *Múltiples vulnerabilidades en Yokogawa STARDOM*. Último acceso: 13 de febrero de 2025. URL: <https://www.incibe.es/incibe-cert/alerta-temprana/avisos-sci/multiples-vulnerabilidades-en-yokogawa-stardom>.
- [25] Python.org. *Welcome to Python.org*. Último acceso: 3 de marzo de 2025. URL: <https://www.python.org/>.
- [26] Neo4j. *Neo4j Desktop Download — Free Graph Database Download*. Último acceso: 3 de marzo de 2025. URL: <https://neo4j.com/download>.
- [27] Glasdoor. *Sueldos para Ingeniero Junior De Telecomunicaciones en Madrid, España*. Último acceso: 16 de mayo de 2025. URL: <https://glasdoor.com/salario/ingeniero-junior-de-telecomunicaciones-madrid-espana>

www.glassdoor.com.ar/Sueldos/madrid-ingenero-junior-de-telecomunicaciones-sueldo-SRCH_IL.0%2C6_IM1030_K07%2C45.htm.

Anexos

Apéndice A

Aspectos éticos, económicos, sociales y ambientales

En este anexo se va a realizar una descripción de los aspectos éticos, económicos, sociales y ambientales surgidos de la realización del trabajo.

Introducción

El desarrollo de tecnologías inteligentes aplicadas a la ciberseguridad, como ARGOS, tiene un impacto significativo en diversos ámbitos más allá del técnico. Este tipo de herramientas, basadas en aprendizaje automático, buscan aumentar la eficacia en la detección de amenazas, pero también implican consideraciones éticas, económicas, sociales y medioambientales. El uso de grandes volúmenes de datos, la automatización de secuencias de ataque y el acceso a escenarios de red realistas exigen una reflexión sobre cómo equilibrar la innovación tecnológica con la responsabilidad social.

Aspectos Sociales y Legales

El script desarrollado para interactuar con la base de datos Neo4j puede analizar información crítica relacionada con la actividad en una red. Si esta información incluye datos personales o confidenciales, es fundamental aplicar buenas prácticas en cuanto a la gestión de la privacidad: acceso controlado, anonimización y cumplimiento normativo. La trazabilidad y la transparencia del uso de los datos también son elementos clave para mantener la confianza de los usuarios y las partes interesadas.

Además, la necesidad de contar con escenarios de red realistas para la detección de puntos débiles conlleva desafíos en materia de privacidad. Para mitigar estos riesgos, se recomienda utilizar escenarios genéricos o anonimizados, como los empleados en este proyecto.

Aspectos Económicos

El costo de desarrollo e implementación de esta herramienta es reducido, especialmente comparado con los sistemas comerciales de detección de amenazas. Sin embargo, su valor económico radica en la prevención: evitar un incidente de seguridad puede suponer un ahorro considerable para una organización, tanto en términos financieros como reputacionales. De este modo, herramientas simples pero efectivas, como la aquí presentada, pueden ser una solución rentable para pequeñas y medianas empresas.

El desarrollo e implementación de ARGOS debe entenderse como una inversión estratégica. Dado que la mayoría de las pequeñas y medianas empresas no logra recuperarse tras un ciberataque, sistemas como ARGOS pueden representar una barrera crítica para evitar el colapso financiero de las organizaciones.

Aspectos Ambientales

A diferencia de los sistemas basados en cómputo intensivo, el impacto ambiental del presente trabajo es limitado. No obstante, su ejecución constante y el almacenamiento de grandes volúmenes de datos en bases como Neo4j pueden implicar un consumo energético relevante en entornos reales. Se recomienda, por tanto, optimizar las consultas, gestionar eficientemente los recursos del servidor y considerar opciones de infraestructura energéticamente responsable.

Apéndice B

Presupuesto Económico

Este anexo presenta una estimación de los costes asociados a la realización de este trabajo de fin de máster y que han sido necesarios para la consecución de los objetivos. Se van a tener en cuenta para ello tanto los materiales utilizados como la mano de obra, a partir de los cuales se realizará una aproximación de los costes directos e indirectos.

Costes de mano de obra

Para estimar el coste asociado a la mano de obra, en primer lugar se ha realizado una aproximación del número de horas empleadas en la realización del mismo. Se ha considerado un periodo de 7 meses en los que se han empleado una media de 30 horas por semana, lo que hace un total de 840 horas.

Una vez obtenida una aproximación de las horas empleadas en todo el proceso de diseño y desarrollo del sistema, se van a considerar 27.000€ como el salario anual de un Ingeniero de Telecomunicación *junior*[27] para estimar el coste de mano de obra. Con un contrato de 40h semanales y estimando 48 semanas laborables, este salario resultaría en un importe de 14€/hora, lo que permite calcular el montante total:

Costes de mano de obra		
Horas	€/Hora	Total
840	14	11.760€

Tabla B.1: Costes de mano de obra

Costes materiales

Para los costes materiales se tendrá en cuenta el ordenador personal con el que se ha realizado todo el desarrollo. Dicho equipo es un HP Pavilion 15-cs0012ns, que cuenta con un procesador Intel Core i7, una pantalla de 15.6 pulgadas y 16GB de memoria RAM como algunas de sus características principales. Este ordenador tiene un coste aproximado de 1.100€. Además, se ha calculado la amortización mensual en base a los 6 años de antigüedad que tiene y esta se ha multiplicado por el número de meses en los que se ha estado trabajando en el proyecto (Septiembre-Mayo) para obtener la amortización total.

Costes materiales					
Material	Tiempo de vida (años)	Precio	Amortización (€/mes)	Uso (meses)	Total
PC	6	1.100€	15,28	8	122,22€

Tabla B.2: Costes materiales

Costes indirectos

Se han estimado los costes indirectos (como pueden ser luz, agua o internet) en un 20 % sobre la suma de los costes de mano de obra y material:

Costes indirectos		
Coste directo	Estimación	Total
11.882,22€	20 %	2376,44€

Tabla B.3: Costes indirectos

Costes totales

Para terminar, se incluyen el IVA y el beneficio industrial para obtener los costes totales asociados al proyecto:

Costes totales	
Concepto	Coste
Costes directos	Mano de obra
	Material
	Total
Costes indirectos	
Total antes de beneficios e impuestos	
Beneficio industrial (10 %)	
Total antes de impuestos	
IVA (21 %)	
Total	

Tabla B.4: Costes totales

Apéndice C

Escenarios de red

Oficina corporativa

```
1 CREATE
2   (u1:Activo:Usuario {
3     name: 'Alice'
4   },
5
6   (u2:Activo:Usuario {
7     name: 'Bob'
8   },
9
10  (pc1:Activo:DispositivoRed {
11    name: 'PC1',
12    ip: '192.168.1.10',
13    platform: 'Windows',
14    permissions: ['User'],
15    capabilities: ['Permissions to access directories', ,
16      'Access to shared folders and content with write permissions',
17      'Presence of physical medium or device', 'Privileges to
18      access certain files and directories', 'Privileges to access
19      network shared drive', 'Privileges to access removable
20      media drive and files'],
21    cve: ['CVE-2004-2227']
22  },
23
24  (pc2:Activo:DispositivoRed {
25    name: 'PC2',
26    ip: '192.168.1.11',
27    platform: 'Linux',
28    permissions: ['Administrator'],
29    capabilities: ['Permissions to access directories', ,
30      'Privileges to access network shared drive', 'Privileges to
31      access removable media drive and files']
32  },
33
34  (s1:Activo:Servidor {
35    name: 'WebServer',
```

```

29     ip: '192.168.1.20',
30     platform: 'Windows',
31     permissions: ['Administrator'],
32     capabilities: ['Remote exploitation for execution
33     requires a remotely accessible service', 'Active remote
34     service accepting connections and valid credentials']
35   },
36
37   (s2:Activo:Servidor {
38     name: 'MailServer',
39     ip: '192.168.1.21',
40     platform: 'Linux',
41     permissions: ['Administrator'],
42     capabilities: ['Remote exploitation for execution
43     requires a remotely accessible service', 'Active remote
44     service accepting connections and valid credentials']
45   },
46
47   (fw:Activo:DispositivoSeguridad {
48     name: 'Firewall',
49     ip: '192.168.1.1',
50     platform: 'Network',
51     permissions: ['Administrator'],
52     capabilities: ['Network interface access and packet
53     capture driver']
54   },
55
56   (app1:Activo:Aplicacion {
57     name: 'Apache',
58     ip: '192.168.1.30',
59     platform: 'Linux',
60     permissions: ['Administrator'],
61     capabilities: ['Privileges to access certain files and
62     directories']
63   })
64
65 CREATE
66   (u1)-[:AUTENTICACION]->(pc1),
67   (u2)-[:AUTENTICACION]->(pc2),
68   (pc1)-[:CONEXION {protocolo: 'LDAP'}]->(s1),
69   (pc2)-[:CONEXION {protocolo: 'SMTP'}]->(s2),
70   (s1)-[:CONEXION {puerto: 25, protocolo: 'SMTP'}]->(s2),
71   (fw)-[:PROTECCION {tipo: 'Firewall'}]->(s1),
72   (fw)-[:PROTECCION {tipo: 'Firewall'}]->(s2),
73   (app1)-[:ALOJAMIENTO {entorno: 'WebServer'}]->(s1);

```

Smart Home

```

1 CREATE
2   (r:Activo:DispositivoSeguridad {
3     name: 'Router',
4     ip: '192.168.0.1',
5     platform: 'Network',
6     permissions: ['Administrator'],

```

```
7     capabilities: ['Remote exploitation for execution
8      requires a remotely accessible service reachable over the
9      network or other vector of access such as spearphishing or
10     drive-by compromise.', 'Network interface access and packet
11     capture driver']
12   },
13
14   (p:Activo:DispositivoRed {
15     name: 'Printer',
16     ip: '192.168.0.100',
17     platform: 'Network',
18     permissions: ['User'],
19     capabilities: ['Active remote service accepting
20      connections and valid credentials', 'Privileges to access
21      network shared drive', 'Presence of physical medium or
22      device']
23   }),
24
25   (st:Activo:DispositivoRed {
26     name: 'SmartTV',
27     ip: '192.168.0.200',
28     platform: 'SaaS',
29     permissions: ['User'],
30     capabilities: ['Active remote service accepting
31      connections and valid credentials', 'Privileges to access
32      certain files and directories', 'Unpatched software or
33      otherwise vulnerable target. Depending on the target and
34      goal, the system and exploitable service may need to be
35      remotely accessible from the internal network.' ]
36   }),
37
38   (c:Activo:DispositivoRed {
39     name: 'CCTV',
40     ip: '192.168.0.300',
41     platform: 'Network',
42     permissions: ['Administrator'],
43     capabilities: ['Remote exploitation for execution
44      requires a remotely accessible service reachable over the
45      network or other vector of access such as spearphishing or
46      drive-by compromise.', 'Active remote service accepting
47      connections and valid credentials', 'Network interface
48      access and packet capture driver']
49   }),
50
51   (h:Activo:DispositivoRed {
52     name: 'Hub',
53     ip: '192.168.0.400',
54     platform: 'IaaS',
55     permissions: ['Administrator'],
56     capabilities: ['Active remote service accepting
57      connections and valid credentials', 'Network interface
58      access and packet capture driver', 'Privileges to access
59      directories, files, and API endpoints that store information
60      of interest']
61   }),
```

```

41
42     (s:Activo:DispositivoRed {
43         name: 'Smartphone',
44         ip: '192.168.0.500',
45         platform: 'macOS',
46         permissions: ['User'],
47         capabilities: ['Removable media allowed', 'Privileges to
48             access removable media drive and files', 'Unpatched software
49             or otherwise vulnerable target.']
50     }),
51
52     (t:Activo:DispositivoRed {
53         name: 'Thermostat',
54         ip: '192.168.0.600',
55         platform: 'IaaS',
56         permissions: ['SYSTEM'],
57         capabilities: ['Active remote service accepting
58             connections and valid credentials', 'Privileges to access
59             directories, files, and API endpoints that store information
60             of interest', 'Remote exploitation for execution requires a
61             remotely accessible service reachable over the network or
62             other vector of access such as spearphishing or drive-by
63             compromise.']
64     })
65
66 CREATE
67     (h)-[:CONEXION {protocolo: 'WPA2'}]->(r),
68     (t)-[:CONEXION {protocolo: 'WPA2'}]->(r),
69     (c)-[:CONEXION {protocolo: 'WPA2'}]->(r),
70     (p)-[:CONEXION {protocolo: 'WPA2'}]->(r),
71     (s)-[:CONEXION {protocolo: 'WPA2'}]->(r),
72     (st)-[:CONEXION {protocolo: 'WPA2'}]->(r),
73     (h)-[:CONEXION {puerto: 1883, protocolo: 'MQTT'}]->(t),
74     (s)-[:CONEXION {puerto: 5000, protocolo: 'DLNA'}]->(st),
75     (s)-[:CONEXION {puerto: 631, protocolo: 'IPP'}]->(p);

```

Fábrica industrial

```

1 CREATE
2     (s:Activo:DispositivoRed {
3         name: 'Sensor',
4         ip: '10.0.1.10',
5         platform: 'PRE',
6         permissions: ['User'],
7         capabilities: ['Presence of physical medium or device', 'Privileges to access certain files and directories']
8     }),
9
10    (a:Activo:DispositivoRed {
11        name: 'Actuator',
12        ip: '10.0.1.20',
13        platform: 'PRE',
14        permissions: ['User'],
15        capabilities: ['Presence of physical medium or device', 'Privileges to access certain files and directories']
16    });

```

```

    Privileges to access certain files and directories ']
16   }),

17
18 (hmi:Activo:DispositivoRed {
19     name: 'HMI',
20     ip: '10.0.1.30',
21     platform: 'Windows',
22     permissions: ['Administrator'],
23     capabilities: ['Access to shared folders and content with
24       write permissions', 'Microsoft Core XML Services (MSXML) or
25       access to wmic.exe']
26   }),

27
28 (plc:Activo:DispositivoRed {
29     name: 'PLC',
30     ip: '10.0.1.40',
31     platform: 'PRE',
32     permissions: ['Administrator'],
33     capabilities: ['Access to shared folders and content with
34       write permissions', 'Autorun enabled or vulnerability
35       present that allows for code execution', 'Unpatched software
36       or otherwise vulnerable target. Depending on the target and
37       goal, the system and exploitable service may need to be
38       remotely accessible from the internal network.'],
39     cve: 'CVE-2022-29519'
40   }),

41
42 (cdb:Activo:DispositivoRed {
43     name: 'CloudDB',
44     ip: '10.0.1.50',
45     platform: 'SaaS',
46     permissions: ['Administrator'],
47     capabilities: ['API endpoints that store information of
48       interest', 'Permissions to access directories, files, and
49       API endpoints that store information of interest']
50   }),

51
52 (fw:Activo:DispositivoSeguridad {
53     name: 'Firewall',
54     ip: '10.0.1.1',
55     platform: 'Network',
56     permissions: ['Administrator'],
57     capabilities: ['Network interface access and packet
       capture driver']
58   })

59 CREATE
60   (plc)-[:CONEXION {protocolo: 'Modbus'}]->(a),
61   (s)-[:CONEXION {protocolo: 'Modbus'}]->(plc),
62   (hmi)-[:CONEXION {protocolo: 'Modbus'}]->(plc),
63   (plc)-[:CONEXION {puerto: '1883', protocolo: 'MQTT'}]->(cdb)
64   ),
65   (fw)-[:PROTECCION {tipo: 'Firewall'}]->(plc),
66   (fw)-[:PROTECCION {tipo: 'Firewall'}]->(cdb);

```

Apéndice D

Artículo JNIC

Como resultado del desarrollo de este Trabajo de Fin de Master, se ha escrito un artículo sobre ARGOS para su presentación en las Jornadas Nacionales de Investigación en Ciberseguridad (JNIC), cuya edición de 2025 tendrá lugar el mes de junio en Zaragoza.

El artículo, cuya publicación ya ha sido aprobada, se adjunta en la siguiente página.

ARGOS: Herramienta para la simulación de rutas de ataque en entornos de conciencia cibersituacional

Carmen Sánchez-Zas, Samuel García Sánchez, Xavier Larriva-Novo, Sonia Solera-Cotanilla,
Oscar Jover-Walsh, Víctor A. Villagrá

Universidad Politécnica de Madrid (UPM). DIT, ETSI Telecomunicaciones. Avda. Complutense 30, 28040 Madrid
carmen.szas@upm.es, samuel.garcia.sanchez@alumnos.upm.es, xavier.larriva.novo@upm.es,
sonia.solera@upm.es, oscar.jwalsh@upm.es, victor.villagra@upm.es

Resumen—En un contexto donde las amenazas de ciberseguridad son cada vez más sofisticadas, este artículo propone una solución para la prevención de ciberataques mediante la visualización del camino de ataque. Esto facilita la identificación de vulnerabilidades y la evaluación del impacto potencial de una intrusión, permitiendo a los equipos de seguridad anticiparse a posibles amenazas y fortalecer la protección de los sistemas. En esta propuesta se simula una ruta de ataque partiendo de técnicas definidas en MITRE ATT&CK y, a continuación, se representa gráficamente el posible recorrido que un adversario seguiría para comprometer un activo vulnerable desde cualquier punto de entrada al sistema. Al obtener acceso inicial a una infraestructura organizacional, un atacante puede moverse lateralmente entre activos hasta alcanzar los elementos más críticos. Para tratar de evitarlo se ha desarrollado una herramienta denominada ARGOS, presentada como un enfoque proactivo para mejorar la ciberseguridad en entornos de conciencia cibersituacional.

Index Terms—Ruta de ataque, Ciberseguridad, TTPs, Cadenas de Markov, MITRE ATT&CK.

Tipo de contribución: Investigación original (límite 8 páginas)

I. INTRODUCCIÓN

Desde hace tiempo, las organizaciones y los gobiernos se enfrentan constantemente a posibles ataques de seguridad [1]. Sin embargo, la necesidad de una ciberdefensa de próxima generación se ha vuelto aún más urgente en esta era en la que las superficies de ataque que los ciberdelincuentes pueden explotar han crecido a un ritmo alarmante. Este rápido crecimiento deriva del aumento de la complejidad en los sistemas empresariales y la adopción masiva de servicios en la nube y dispositivos conectados [2]. En este contexto, la ciberdefensa basada en el análisis predictivo resulta más proactiva que las tecnologías actuales, que dependen de la detección de intrusiones.

Este nuevo enfoque de ciberdefensa está transformando la manera en que las organizaciones comprenden sus vulnerabilidades y se preparan ante posibles ataques. El *Informe de Situación Ciberseguridad en España 2024* [3] destaca que la adopción de la inteligencia artificial se ha acelerado en el ámbito de la ciberseguridad, proporcionando herramientas de análisis predictivo y detección automatizada de amenazas dentro de los entornos de conciencia cibersituacional. Ya no se trata solo de identificar vulnerabilidades aisladas o puntos débiles, ahora se trata de entender cómo los atacantes pueden encadenar estos puntos débiles para crear un camino de ataque crítico hacia sus activos más valiosos, como los administradores de dominio, bases de datos críticas, controladores de dominio de *Active Directory*, etc.

De este contexto surge esta propuesta, con el que se pretende ofrecer una solución para la prevención de ciberataques mediante la visualización del camino de ataque o *attack path visualization*, que es una representación gráfica de los posibles caminos de ataque que un adversario podría seguir para comprometer un activo desde cualquier punto de entrada en el sistema objetivo. Por ejemplo, una vez que un atacante ha logrado un acceso inicial al entorno organizacional, el camino de ataque muestra la posible ruta que el adversario puede tomar entre los activos para llegar a los elementos más valiosos, como los controladores de dominio.

El enfoque de la propuesta permite identificar los activos más vulnerables y evaluar la severidad de los ataques según rutas de ataque obtenidas de la simulación e integrar estos resultados con estrategias de mitigación, proporcionando recomendaciones adaptadas a los ataques identificados. De esta forma, se ha desarrollado un prototipo de esta propuesta en la herramienta ARGOS que contribuye a la mejora en la toma de decisiones en relación con la gestión de riesgos, un concepto clave en los entornos de conciencia cibersituacional.

A lo largo del documento presentaremos el entorno de la propuesta y los conceptos principales en los que se basa el desarrollo. A continuación, detallaremos la arquitectura de la herramienta, el desarrollo y los resultados obtenidos. En la sección I se presenta el entorno del trabajo. A continuación, en la sección II, se presentan trabajos anteriores que tratan la misma problemática. En la sección III se definen brevemente los conceptos teóricos relacionados con el proyecto. La arquitectura de ARGOS se define en la sección IV y se desarrolla en la sección V. Finalmente, la validación y los resultados obtenidos se presentan en la sección VI. Finalmente, las conclusiones y líneas futuras se presentan en la sección VII.

II. TRABAJOS RELACIONADOS

La simulación de rutas de ataque en ciberseguridad mediante modelos probabilísticos ha sido un objetivo recurrente en la investigación de seguridad. Ha sido ampliamente estudiado para mejorar la gestión de riesgos y la identificación de activos críticos. En particular, el uso de cadenas de Markov ha demostrado ser una herramienta eficaz para estimar rutas de ataque y apoyar a la toma de decisiones en seguridad informática. En [4], los autores plantean el uso de HMM siguiendo una mezcla de distribuciones gaussianas (*Gaussian Mixture Hidden Markov Model* o GM-HMM) para conseguir detectar ataques multipaso, es decir, una secuencia de ataques.

Holgado et al. presentan en [5] un sistema de predicción de ataques multietapa basado en modelos ocultos de Markov

(HMM). Este enfoque modela la progresión de ataques partiendo de alertas que provienen de sistemas de detección de intrusiones (IDS), con el objetivo de predecir las próximas acciones de los atacantes y mejorar las estrategias de defensa. Sin embargo, se centra en la predicción, sobre todo centrada en ataques DDoS, pero no se integra en una herramienta enfocada en la gestión de riesgos y la protección específica de activos.

Siguiendo con la aplicación de HMM en este ámbito, Amin et al. combinan en [6] esta herramienta con técnicas de cibergaño para frustrar el movimiento lateral de atacantes dentro de la red. Con un enfoque a la vez reactivo y proactivo, en primer lugar predicen la ruta de ataque más probable partiendo de alertas de IDS y determinan el despliegue óptimo de nodos señuelo a lo largo de esa ruta. Mediante la planificación parcialmente observable de Monte-Carlo (POMCP) definen acciones defensivas para bloquear al atacante en caso de que no siga la ruta calculada. Su investigación, partiendo de una base muy similar a la que se va a presentar en este documento, propone un enfoque hacia la defensa activa en lugar de la evaluación del impacto de los ataques en activos y la gestión de riesgos. Además, en [7] los autores tratan de predecir ataques y la familia a la que pertenecen mediante HMM. El objetivo es generar una secuencia de estados que correspondan con las etapas del ataque *Action Spoofing*, aunque sería generalizable a cualquier tipo de ataque.

Los autores de [8], por su parte, proponen un sistema basado en grafos de ataque y cadenas de Markov para modelar escenarios de ataque. Su metodología permite identificar los posibles caminos entre un origen y un destino dentro de una red y calcula la probabilidad de ataque. Finalmente, calcula el riesgo total del ataque, lo que le permite evaluar el nivel de peligro frente al grado de penetración del ataque. En este caso, los activos más vulnerables no se identifican y por tanto no se llega a evaluar la efectividad de diferentes estrategias de mitigación.

Otro enfoque en el cálculo de patrones de ataque consiste en considerar la variable temporal, como en [9], donde los autores presentan una herramienta capaz de computar todas las rutas de ataque iterando sobre las posibles duraciones de los posibles ataques. Los autores plantean añadir sobre los árboles de ataque utilizados para calcular las rutas, sellos temporales mediante *Boolean logic Driven Markov Processes* (BDMP).

El estudio sobre los trabajos relacionados han permitido demostrar que las cadenas de Markov son una herramienta clave para determinar qué activos son críticos, evaluar la severidad de los ataques, definiendo las respuestas más adecuadas para mitigar de forma proactiva las amenazas.

Los trabajos anteriormente analizados han contribuido notablemente al desarrollo de modelos predictivos en seguridad, pero su integración con enfoques de gestión de riesgos sigue siendo un desafío. A diferencia de estos, la herramienta que se propone en esta investigación contiene un modelo para la simulación de rutas de ataque utilizando cadenas de Markov con el objetivo de estimar qué movimientos son los más probables dentro de la red y utilizar esta información como complemento a la gestión de riesgos, lo que supone una ventaja con respecto a otros enfoques tradicionales.

III. MARCO TEÓRICO

III-A. MITRE ATT&CK

MITRE ATT&CK (*Adversarial Tactics, Techniques, and Common Knowledge*) [10] es una base de conocimientos universalmente accesible y continuamente actualizada para modelar, detectar, prevenir y combatir amenazas de ciberseguridad basándose en el comportamiento conocido de los cibercriminales adversarios. Creado por MITRE Corporation, este marco de referencia ofrece una visión estructurada de las tácticas y técnicas empleadas por los atacantes en diferentes fases de una intrusión. Su principal objetivo es proporcionar a organizaciones, analistas de seguridad y profesionales de IT (*Information Technology*) una herramienta para comprender, detectar y mitigar amenazas cibernéticas de manera más efectiva.

El marco se organiza en matrices, que clasifican tácticas y técnicas en función de su propósito y cómo se relacionan con las etapas de un ataque. Las tácticas representan las metas o propósitos generales que los atacantes buscan alcanzar durante cada fase del ataque, como obtener acceso inicial, moverse lateralmente por la red o exfiltrar datos. Por otro lado, las técnicas describen los métodos específicos que los atacantes emplean para lograr las tácticas, por ejemplo, el aprovechamiento de vulnerabilidades o la ejecución de scripts maliciosos.

III-B. Cadenas de Markov

Una cadena de Markov [11] es un modelo matemático utilizado para describir sistemas que evolucionan de manera secuencial, donde la probabilidad de transición entre estados depende únicamente del estado actual y no del historial previo. Este principio se conoce como la propiedad de Markov.

Las cadenas de Markov son particularmente útiles para modelar y analizar procesos estocásticos, es decir, aquellos que involucran incertidumbre. Su capacidad para representar dinámicas donde el futuro depende solo del presente las hace valiosas en áreas como la predicción, la identificación de patrones de comportamiento y la simulación de escenarios en distintos contextos.

Una cadena de Markov consta de tres elementos principales: los estados, que representan las posibles condiciones del sistema (por ejemplo, “soleado”, “lluvioso” o “nublado” en un modelo climático); las transiciones, que describen los cambios entre estados con ciertas probabilidades; y la matriz de transición, que indica la probabilidad de pasar de un estado a otro, donde cada fila representa un estado actual y cada columna un estado futuro. Un estado es absorbente si, una vez alcanzado, no se puede abandonar, mientras que el estado inicial es aquel en el que se encuentra el sistema al inicio del análisis.

III-C. Rutas de ataque

Una ruta de ataque [12] es una representación visual del camino que un atacante podría seguir para explotar debilidades en un sistema y comprometer activos críticos. Este concepto, clave en la ciberseguridad, no solamente permite analizar vulnerabilidades aisladas, sino también cómo estas se interconectan para formar un trayecto explotable.

El objetivo principal es ayudar a las organizaciones a identificar, priorizar y mitigar riesgos, abordando las amenazas

desde la perspectiva del atacante. En lugar de centrarse únicamente en solucionar vulnerabilidades individuales, el análisis de rutas de ataque proporciona un enfoque estratégico para visualizar el contexto completo de los riesgos, lo que resulta fundamental para proteger activos esenciales.

III-D. Bases de datos orientadas a grafos

Una ruta de ataque pierde gran parte de su potencial si no se visualiza correctamente. La visualización de rutas de ataque [13] es una herramienta poderosa que, mediante el uso de grafos, permite mapear los caminos que un atacante podría seguir dentro de una red, identificando conexiones entre activos, configuraciones erróneas y vulnerabilidades.

Esta representación visual facilita la priorización de recursos, ya que permite identificar los trayectos que conducen a activos críticos, como controladores de dominio o bases de datos sensibles. Así, la información sobre riesgos se transforma en una estrategia clara de mitigación.

Para mapear estas rutas de manera efectiva, se utilizan bases de datos de grafos, que representan de forma estructurada y visual la interconexión entre activos, vulnerabilidades y posibles movimientos de un atacante dentro de una red. Estas bases de datos se componen de nodos o vértices, que representan elementos individuales del sistema, y aristas o relaciones, que reflejan las conexiones entre ellos.

A diferencia de las bases de datos tradicionales, que organizan la información en tablas y requieren consultas complejas para analizar relaciones entre datos, las bases de datos de grafos están diseñadas para manejar interconexiones de manera eficiente. Este aspecto las hace especialmente útiles para modelar escenarios de ataque, ya que permiten identificar caminos críticos con mayor rapidez y precisión.

IV. ARQUITECTURA DE ARGOS

El diseño de la propuesta para la simulación de rutas de ataque se muestra en la Figura 1, cuya arquitectura se implementa en la herramienta ARGOS.

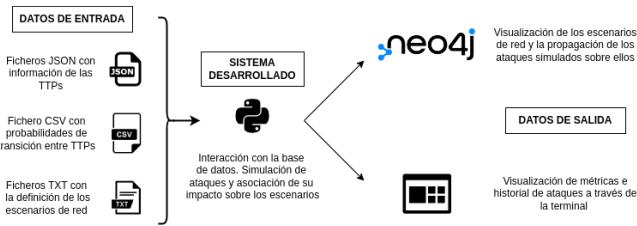


Figura 1. Diagrama de la arquitectura

La información sobre Tácticas, Técnicas y Procedimientos (TTPs) se extrae de la matriz empresarial de MITRE ATT&CK y se gestiona mediante archivos en formatos .json, y .csv, lo que facilita la actualización y la adaptación a nuevas amenazas sin necesidad de modificar la estructura del sistema.

Los escenarios de red se definen y cargan desde ficheros de texto que contienen las relaciones y configuraciones de los activos en un formato interpretable por Neo4j. Esto permite modelar la infraestructura de la red de forma escalable, pudiendo adaptarse a diferentes situaciones. Para la validación

del sistema, se ha desarrollado un breve catálogo de tres simples escenarios: una pequeña oficina corporativa, un escenario de tipo *smart home* y una pequeña planta industrial.

Con toda esta información, el sistema simula rutas de ataque utilizando cadenas de Markov, generando secuencias de técnicas en función de probabilidades definidas. Posteriormente, estas secuencias se integran en el grafo del escenario de red que se haya cargado, permitiendo visualizar la propagación de amenazas y su impacto en los activos comprometidos. Esta asociación se lleva a cabo mediante un mapeo entre las TTPs y los activos afectados, considerando factores como la plataforma y capacidades del activo, los permisos de los que dispone y vulnerabilidades asociadas (CWE/CVE).

Para la visualización en la terminal, se utiliza la librería Rich, que proporciona un dashboard estructurado donde se muestra un resumen del ataque junto a métricas clave acerca de los activos y técnicas involucrados en el mismo, en base a las cuales se calcula una criticidad. Además, el sistema cuenta con un historial de ataques simulados.

V. DESARROLLO

V-A. Modelo de datos

Como ya se ha mencionado, las bases de datos orientadas a grafos permiten representar la información mediante vértices o nodos y aristas o relaciones, lo que resulta especialmente útil para modelar redes informáticas y sus relaciones. Por ello, se ha desarrollado un modelo de datos escalable y flexible, permitiendo la representación tanto de distintos escenarios de red como secuencias de ataque bajo una lógica común.

Para el modelado de los escenarios de red, se han definido dos entidades principales: **Usuario** y **Activo**. La entidad **Usuario** representa las cuentas personales o administrativas que requieren autenticación en distintos elementos de la red. Por otro lado, la entidad **Activo** agrupa todos los dispositivos y sistemas de la infraestructura, dividiéndose en cuatro subcategorías según su función. Un **Servidor** representa dispositivos que ejecutan servicios accesibles, mientras que un **DispositivoRed** incluye equipos de usuario y otros sistemas conectados. Los elementos de seguridad, como *firewalls*, VPNs o sistemas IDS, se agrupan bajo la categoría **DispositivoSeguridad**. Finalmente, el software instalado en servidores o estaciones de trabajo se modela como **Aplicación**.

Cada nodo en el grafo cuenta con atributos en formato clave-valor, lo que permite almacenar información detallada sobre sus características. Estos atributos son fundamentales para asociar los activos con posibles vulnerabilidades y técnicas de ataque, facilitando el análisis de seguridad.

Las relaciones entre nodos son esenciales para modelar la conectividad y las dependencias dentro de la red. Se han definido cuatro tipos principales de relaciones entre los elementos presentes en los escenarios de red: **CONEXIÓN**, que indica la comunicación entre dos activos a través de un puerto; **PROTECCIÓN**, que señala cuando un activo está resguardado por un dispositivo de seguridad; **AUTENTICACIÓN**, que representa la necesidad de un usuario de autenticarse en un activo; y **ALOJAMIENTO**, que vincula una aplicación o servicio con el servidor donde está instalado.

Por otro lado, para modelar los ataques dentro del sistema, se ha definido la entidad **Técnica**, que representa cada una

de las técnicas de la matriz empresarial de MITRE ATT&CK utilizadas en una secuencia de ataque. Esta entidad se asocia con los activos vulnerables a través de atributos específicos. Además, se han definido dos relaciones clave para representar la progresión de los ataques: TRANSICIÓN, que indica la sucesión entre técnicas dentro de una secuencia de ataque, y EXPLOTACIÓN, que muestra la afectación de un activo por una técnica específica. En la Figura 2 se muestran todos los nodos y relaciones descritos.

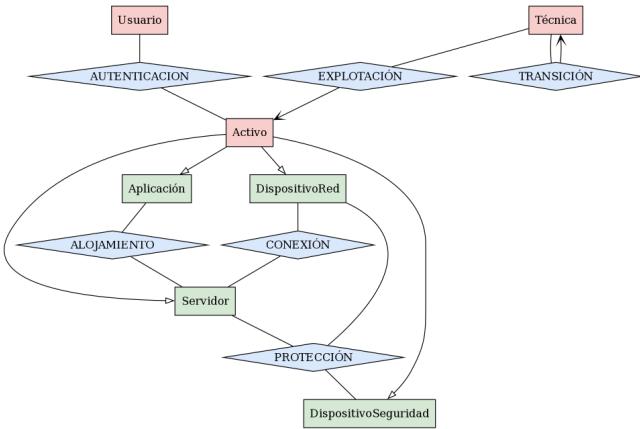


Figura 2. Modelo de datos empleado en ARGOS

V-B. Funcionamiento general

ARGOS es una herramienta de línea de comandos desarrollada en Python que permite analizar cómo una serie de TTPs pueden afectar un entorno de red simulado. Su principal objetivo es proporcionar una manera estructurada y automatizada de modelar ciberataques y visualizar su impacto en infraestructuras digitales.

ARGOS trabaja sobre una base de datos en Neo4j, que ha de ser configurada previamente. Para operar, ARGOS cuenta con cuatro comandos principales. El comando `prepare` permite cargar un escenario de red en la base de datos, transformándolo en un grafo dentro de Neo4j que representa activos y sus conexiones. Los escenarios se leen desde un archivo de texto ubicado en la carpeta `scenarios`. Estos archivos están escritos en un lenguaje denominado *Cypher*.

Cypher es el lenguaje de consulta diseñado específicamente para trabajar con grafos en Neo4j. Utiliza una sintaxis inspirada en SQL pero optimizada para manejar nodos y relaciones. De esta forma, es posible la realización de consultas y operaciones complejas sobre los grafos.

Cada escenario contiene información sobre los activos que lo componen (servidores, estaciones de trabajo, dispositivos, etc.) y sus características clave, como los sistemas operativos que utilizan, los permisos que poseen y las vulnerabilidades que podrían explotarse, utilizando el modelo de datos descrito en la sección V-A.

Antes de ejecutar cualquier simulación, ARGOS requiere que se cargue un escenario de red en la base de datos. Luego, el comando `attack` genera una secuencia de ataque basada en un modelo de Cadenas de Markov, que permite generar secuencias de técnicas de manera probabilística, y la ejecuta sobre el escenario cargado. Partiendo de una técnica seleccionada de manera aleatoria, la Cadena de Markov implementada

permite calcular el siguiente estado únicamente a partir del estado actual utilizando las probabilidades de transición entre estados, definidas en el fichero `transitions.csv`. Esto significa que cada técnica utilizada en el ataque tiene una probabilidad de llevar a la siguiente, asegurando que las transiciones entre estados sean coherentes con patrones de ataque reales. De esta forma, ARGOS puede modelar la progresión de un ciberataque desde su punto de inicio hasta su posible finalización, considerando factores como la probabilidad de éxito de cada técnica y las condiciones del escenario.

Una vez generada la secuencia de ataque, ARGOS asocia cada técnica con los activos a los que afecta dentro del escenario cargado. Para ello, analiza características como los sistemas operativos presentes, los permisos disponibles y las vulnerabilidades conocidas, que habían sido definidas como atributos en los nodos. Si un activo cumple con los criterios necesarios, se establece una relación entre la técnica de ataque y el activo comprometido, reflejando el impacto del ataque en la red simulada. En la Figura 3, se presentan los atributos que caracterizan cada entidad, así como las condiciones para que se produzca una relación de tipo EXPLOTACIÓN entre ellos.

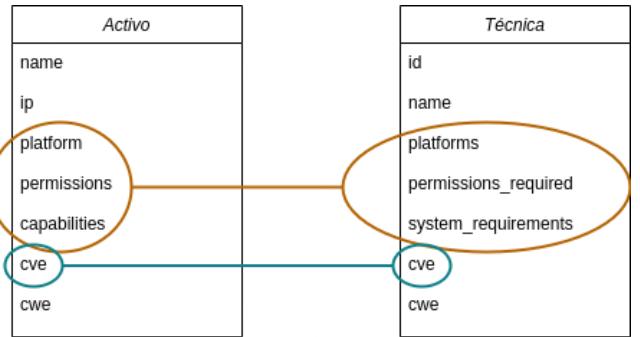


Figura 3. Condiciones de explotación de una técnica sobre un activo

La lógica utilizada para vincular las técnicas de ataque con los activos afectados se basa en la ejecución de una consulta Cypher sobre la base de datos Neo4j. Esta asociación se puede dar mediante dos condiciones. El filtro principal, representado de color naranja en la Figura 3, exige que el activo se ejecute en alguna de las plataformas afectadas por la técnica y, además, que cumpla con dos condiciones simultáneas: que posea al menos un permiso que coincida con los permisos requeridos por la técnica y que cuente con alguna capacidad del sistema que corresponda con los requisitos de la técnica. Esta condición compuesta refleja la necesidad de que el activo no solo opere en la plataforma objetivo, sino que también cumpla con las condiciones de acceso y recursos que la técnica requiere para ser aplicada. Adicionalmente, como alternativa a este conjunto de condiciones, la consulta también considera un segundo filtro (representado en color azul) para los activos que tengan vulnerabilidades identificadas por los CVEs relacionados con la técnica, permitiendo así que la técnica explote directamente vulnerabilidades conocidas.

Finalmente, el comando `history` permite consultar los ataques realizados previamente, mientras que el comando `clean` limpia la base de datos, eliminando cualquier información previa para realizar nuevas simulaciones desde cero.

VI. VALIDACIÓN

En esta sección, se presenta la validación de la herramienta ARGOS y los resultados obtenidos a partir de su ejecución. El proceso de validación se divide en varios pasos, que incluyen la carga de escenarios, la ejecución de simulaciones de ataque y el análisis de los resultados obtenidos, tal y como se muestra en el diagrama de ejecución de la Figura 4.

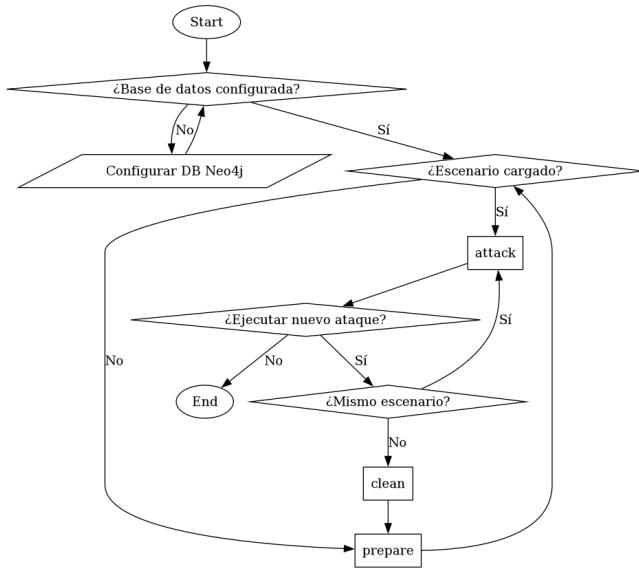


Figura 4. Diagrama de flujo de ARGOS

El primer paso antes de utilizar ARGOS es configurar e iniciar la base de datos. Una vez configurada, es necesario abrir la consola de Neo4j, ya que es donde se mostrarán los grafos correspondientes a la ejecución de ARGOS. Una vez hecho esto, ya se puede interactuar con la base de datos desde la terminal, mediante la interfaz de línea de comandos. El comando a ejecutar para cargar el escenario correspondiente a la oficina corporativa sería el siguiente: `$ python3 argos.py prepare scenarios/oficina.cypher`. Tras ejecutarlo, el programa indica que dicho escenario se ha cargado sin problemas en la base de datos.

Tras este mensaje, en la consola de Neo4j se puede visualizar el escenario cargado en forma de grafo, tal y como se muestra en la Figura 5.

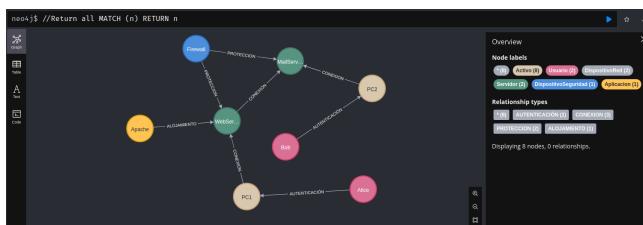


Figura 5. Escenario de oficina corporativa cargado en Neo4j

En este escenario, los usuarios están representados por los nodos Alice y Bob. Alice utiliza un PC Windows con permisos estándar y diversas capacidades que le permiten interactuar con otros activos en la red. El usuario Bob cuenta con Linux en su equipo, donde dispone de permisos de administrador. Tiene instalado el software *Digitaldesign CMS*.

El CVE-2009-3597 afecta a *Digitaldesign CMS* en su versión v0.1, que guarda información sensible en el directorio web raíz con insuficiente control de acceso, lo que permite a los atacantes remotos descargar el fichero de la base de datos a través de una petición directa a `autoconfig.dd`.

Los nodos de tipo servidor son WebServer y MailServer. WebServer corre en Windows con privilegios administrativos y capacidades relacionadas con la ejecución remota y la gestión de archivos y directorios críticos. MailServer, basado en Linux, presenta características similares en cuanto a acceso remoto y almacenamiento de información.

El dispositivo de seguridad en la red es un *firewall*, que cumple una función de protección. Cuenta con permisos administrativos y la capacidad de capturar tráfico de red y analizar paquetes, lo que permite monitorear y restringir accesos no autorizados.

También se ha modelado una aplicación, Apache, que opera en Linux con permisos administrativos. Esta aplicación tiene acceso a archivos y directorios específicos y presenta vulnerabilidades potenciales debido a software no actualizado.

En cuanto a las relaciones, Alice se autentica en su equipo, el cual accede a WebServer mediante el protocolo LDAP, mientras que el equipo utilizado por Bob establece comunicación con MailServer utilizando el protocolo SMTP. Además, WebServer y MailServer están enlazados mediante una conexión en el puerto 25 con protocolo SMTP, lo que indica la comunicación entre estos servicios.

El nodo Firewall protege tanto a WebServer como a MailServer, reforzando la seguridad de la red. Finalmente, Apache está alojado en WebServer, lo que indica su dependencia de este servidor para operar dentro del entorno web.

Si ahora se ejecuta el comando `$ python3 argos.py attack`, el sistema generará una secuencia aleatoria de técnicas y evaluará su impacto sobre el escenario cargado. Al ejecutar de nuevo la `query MATCH (n) RETURN (n)`, se puede apreciar que la secuencia de ataque generada ha sido insertada y que los nodos y relaciones correspondientes han sido creados sobre el escenario (Figura 6).

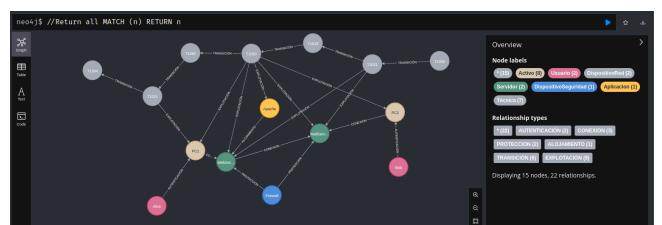


Figura 6. Ataque sobre el escenario de oficina corporativa

Se va a analizar una de las relaciones de explotación creadas, en concreto la que asocia a la técnica T1021 (uno de los nodos de color gris en la Figura 6) con el activo WebServer (representado en verde).

Lo primero es verificar que los nodos se han creado correctamente, conteniendo los atributos que los caracterizan. El nodo MailServer representa el servidor de correo dentro de la infraestructura de red, es un nodo de tipo Activo y subtipo Servidor, sus atributos se muestran en la Figura 7.

Figura 7. Propiedades del nodo MailServer

Por otro lado, el nodo T1021, de tipo Técnica, representa la técnica *Remote services* de la matriz empresarial de MITRE ATT&CK. Esta técnica describe cómo los adversarios pueden usar cuentas válidas para acceder remotamente a sistemas dentro de una red empresarial que cuenta con un sistema de inicio de sesión centralizado. Si logran credenciales legítimas, los atacantes pueden iniciar sesión en múltiples dispositivos mediante protocolos como SSH o RDP, o incluso en servicios en la nube vinculados al dominio. Sus atributos se pueden apreciar en la Figura 8.

Figura 8. Propiedades del nodo T1021

Al no haber ninguna vulnerabilidad configurada en el atributo cve de MailServer, el origen de la relación de EXPLORACIÓN han de ser los parámetros indicados en naranja en la Figura 3. Si se revisan, se puede apreciar que la plataforma que corre el servidor, Linux, es una de las plataformas afectadas por el sistema. Además, el sistema cuenta con la capacidad *Active remote service accepting connections and valid credentials*, uno de los requisitos necesarios por esta técnica para ser explotada. Finalmente, la técnica no requiere de ningún permiso en concreto para ser explotada con éxito. De hecho, esta técnica se clasifica en la matriz empresarial dentro de la táctica *Lateral Movement*, pues al

obtener credenciales válidas de dominio, un atacante puede desplazarse dentro de la red accediendo a múltiples dispositivos y servicios sin necesidad de explotar vulnerabilidades adicionales.

Se cumplen, por tanto, las tres condiciones establecidas para que exista una relación de tipo EXPLORACIÓN entre estos dos nodos.

A la hora de asociar la secuencia de ataque generada con los activos del escenario, el comando attack extrae simultáneamente una serie de métricas acerca del ataque que ha tenido lugar. Estas estadísticas se presentan al terminar la ejecución a través de la terminal, organizadas en paneles de forma clara y jerárquica mediante la librería Rich. Este dashboard permite a los usuarios interpretar de manera rápida y eficaz los resultados del ataque simulado. La información mostrada es la siguiente, y se puede observar en la Figura 9.

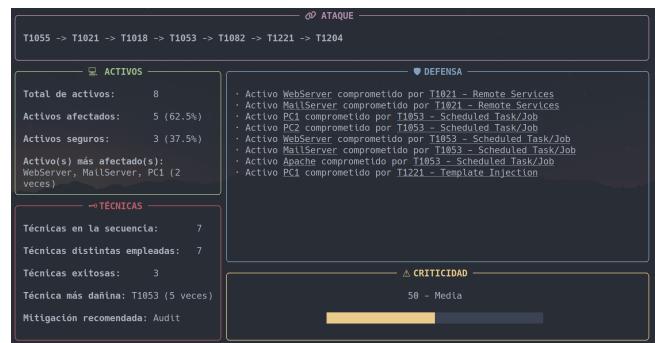


Figura 9. Resumen del ataque presentado sobre el escenario de oficina

- Panel ATAQUE: muestra la cadena de técnicas empleadas en el ataque. Las técnicas aparecen en orden y conectadas con flechas, lo que permite visualizar claramente cómo se ha desarrollado el ataque desde su inicio hasta su última fase.
- Panel DEFENSA: muestra una lista detallada en la que se indica, para cada activo comprometido, la técnica específica que lo ha vulnerado. Este panel proporciona información valiosa para identificar puntos débiles en la infraestructura y planificar medidas correctivas.
- Panel ACTIVOS: este panel proporciona una serie de estadísticas sobre los activos del escenario, permitiendo tener una visión clara del impacto del ataque sobre la infraestructura tecnológica. En concreto, se muestran las siguientes métricas:
 - Total de activos: número total de activos presentes en el escenario cargado.
 - Activos afectados: número y porcentaje de activos que han sido comprometidos por, al menos, una técnica.
 - Activos seguros: número y porcentaje de activos que no han sido vulnerados.
 - Activo(s) más afectado(s): de entre los activos afectados, activo o activos comprometidos por el mayor número de técnicas.
- Panel TÉCNICAS: este panel resume datos sobre las técnicas empleadas, lo que permite enfocar las medidas defensivas en los métodos de ataque más dañinos.

- Técnicas en la secuencia: número total de técnicas en la secuencia.
 - Técnicas distintas empleadas: de entre las técnicas totales de la secuencia, número de técnicas que aparecen una única vez en ella.
 - Técnicas exitosas: número de técnicas que han logrado comprometer, al menos, un activo.
 - Técnica más dañina: identificador de la técnica que ha comprometido el mayor número de activos distintos.
 - Mitigación recomendada: estrategia de defensa recomendada para paliar el impacto de la técnica más dañina.
- Panel CRITICIDAD: utilizando los datos anteriores, se le asigna una criticidad (C) de entre 0 y 100 al ataque, calculada como:

$$C = \left(\left\lfloor \frac{\text{Total activos afectados}}{\text{Total activos}} \times 50 \right\rfloor + \left\lfloor \frac{\text{Total técnicas exitosas}}{\text{Total técnicas empleadas}} \times 50 \right\rfloor \right)$$

Esta criticidad se visualiza mediante una barra de progreso coloreada de la siguiente forma:

- Verde (baja) si $C \leq 33$
- Amarillo (media) si $33 < C \leq 66$
- Rojo (alta) si $66 < C \leq 100$

VII. CONCLUSIONES Y LÍNEAS FUTURAS

El análisis de las rutas de ataque es uno de los puntos clave en la investigación de ciberseguridad actualmente. Para ello, la visualización del camino que este ataque puede recorrer dentro de una red resulta una herramienta eficaz para el estudio de los ciberataques con el objetivo de prevenirlos o mitigar su efecto dentro de entornos organizacionales. Al proporcionar una representación gráfica de los posibles recorridos de un atacante dentro de una red o sistema, se contribuye a la identificación de vulnerabilidades y la toma de decisiones estratégicas en seguridad, objetivos clave de los entornos de conciencia cibersituacional.

Los resultados obtenidos con ARGOS se complementan con el desarrollo de una estrategia de seguridad que permita abordar la respuesta frente a las amenazas a raíz de la caracterización de las técnicas utilizadas. El análisis de las rutas de ataque desde el punto de vista planteado en ARGOS permite identificar activos vulnerables, el impacto de los ataques y las contramedidas más eficaces; mientras que el estudio de los ataques para su caracterización, en este contexto, permite definir modelos más adaptados al comportamiento de los atacantes y que generen simulaciones más realistas.

Con el objetivo de mejorar la funcionalidad y el alcance de ARGOS, se presentan a continuación diversas líneas de desarrollo que podrían implementarse en versiones futuras de la herramienta. La implementación de estas mejoras permitiría que ARGOS se convierta en una herramienta más robusta y adaptable a las necesidades de análisis de seguridad en entornos de red.

Actualmente, ARGOS permite la carga de escenarios de red definidos mediante ficheros Cypher, pero la complejidad de estos escenarios podría ampliarse. Se podría considerar la integración de configuraciones más detalladas, incluyendo topologías de red dinámicas, segmentación de redes, o la

representación de defensas activas como sistemas de detección de intrusiones (IDS) y firewalls.

La simulación de ataques mediante Cadenas de Markov podría beneficiarse de una mayor personalización en la generación de secuencias de TTPs. Por ejemplo, podría incorporarse un sistema de pesos dinámicos basado en inteligencia de amenazas en tiempo real, permitiendo que las probabilidades de transición entre técnicas cambien en función de tendencias recientes en ataques reales. Otra posible mejora sería la incorporación de modelos probabilísticos más avanzados, como redes bayesianas, para mejorar la realismo de las transiciones entre estados.

En el estado actual, la herramienta almacena métricas sobre los ataques ejecutados, pero se podría ampliar la capacidad de análisis mediante la generación de informes automáticos con visualizaciones interactivas. El uso de librerías de análisis de datos como Pandas o herramientas de visualización como Dash podría permitir una exploración más intuitiva del impacto de los ataques en los activos del sistema.

Por último, el almacenamiento del historial de ataques podría enriquecerse con un sistema de versionado que permitiera comparar la evolución de escenarios a lo largo del tiempo. Además, se podría incluir un módulo de aprendizaje automático para identificar patrones en ataques previos y predecir posibles rutas de ataque futuras.

REFERENCIAS

- [1] Check Point: Cyberattacks targeting governments <https://www.checkpoint.com/es/cyber-hub/cyber-security/what-is-cybersecurity-for-governments/cyberattacks-targeting-governments/>. Último acceso: 17 de febrero de 2025.
- [2] Telefónica Tech: Tendencias en Ciberseguridad para 2025 <https://telefonicatech.com/blog/tendencias-en-ciberseguridad-para-2025>. Último acceso: 6 de febrero de 2025.
- [3] Plataforma Tecnológica Española de Tecnologías Disruptivas https://ptedisruptive.es/wp-content/uploads/2024/12/INFORME-DE-SITUACION_CIBERSEGURIDAD_2024.pdf. Último acceso: 6 de febrero de 2025.
- [4] Q. Wang, W. Wang, Y. Wang, J. Ren, B. Zhang: "Multi-Stage Network Attack Detection Algorithm Based on Gaussian Mixture Hidden Markov Model and Transfer Learning" en *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 3470-3484, 2025.
- [5] Pilar Holgado, Víctor A. Villagrá, Luis Vázquez: "Real-Time Multistep Attack Prediction Based on Hidden Markov Model", en *IEEE Transactions on Dependable and Secure Computing*, vol. 17, n. 1, pp. 134-147, 2017.
- [6] M. A. R. A. Amin, S. Shetty, L. Njilla, D. K. Tosh, C. Kamhoua: "Hidden Markov Model and Cyber Deception for the Prevention of Adversarial Lateral Movement", en *IEEE Access*, vol. 9, pp. 49662-49682, 2021.
- [7] S. Dass, P. Datta, A. S. Namin: "Attack Prediction using Hidden Markov Model", en *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1695-1702, 2021.
- [8] Fuxiong Sun, Juntao Pi, Jin Lv, Tian Cao: "Network Security Risk Assessment System Based on Attack Graph and Markov Chain", en *Journal of Physics: Conference Series*, 910, 2017.
- [9] Ricardo M. Czekster, Charles Morisset: "BDMPATHfinder: a tool for exploring attack paths in models defined by Boolean logic Driven Markov Processes", en *2021 17th European Dependable Computing Conference (EDCC)*, pp. 83-86, 2021.
- [10] The Mitre Corporation: MITRE ATT&CK <https://attack.mitre.org/>. Último acceso: 10 de enero de 2025.
- [11] Universidad Politécnica de Madrid: Cadenas de Markov https://dcain.etsin.upm.es/~carlos/bookAA/04.02_CadenasMarkovResultados.html. Último acceso: 10 de marzo de 2025.
- [12] Becca Gomby: What Is An Attack Path & How Does It Help Identify Risks? <https://www.panoptica.app/blog/what-is-an-attack-path-how-does-it-help-identify-risks>. Último acceso: 30 de enero de 2025.

- [13] Picus Labs: What Is Attack Path Visualization? <https://www.picussecurity.com/resource/glossary/what-is-attack-path-visualization>. Ultimo acceso: 30 de enero de 2025.