Intrusion Modeling with Temporal Graphs: Optimizing Proactive Threat Hunting

Diego Staino[1] and Lucas Sotomayor [2]

[1] Instituto Universitario de la Policia Federal Argentina, Argentina,
[2] Universidad de Buenos Aires, Argentina
diegostaino@gmail.com, sotomayorlucas@pm.me

Abstract. This work proposes a temporal typed knowledge graph model for security telemetry and a hypothesis-driven depth-first search algorithm for proactive Threat Hunting. Unlike traditional SIEM platforms, which store telemetry in tabular form and require iterative manual pivoting to reconstruct attack chains, the proposed approach materializes causal relationships between security events as a directed multigraph with nine entity types and nine relation types. Analysts encode attack hypotheses as typed sequential chains, and the system returns all temporally valid paths through five integrated pruning rules (relation type, entity type, causal monotonicity, time window, and k-simplicity). Empirical evaluation on real-world datasets—including Sysmon telemetry, Azure Sentinel logs, and the Mordor corpus (963K events)—demonstrates speedups from 1,747× to 11,600× over unpruned baselines, with sub-millisecond search times on graphs exceeding 20,000 edges. The operational comparison shows a reduction from 8–20 minutes and multiple manual queries per investigation (SIEM) to 1–2 minutes with a single hypothesis formulation, representing an order-of-magnitude decrease in Mean Time to Investigate (MTTI).

Keywords: Threat Hunting, temporal knowledge graph, security telemetry, hypothesis-driven search, MITRE ATT&CK.

Modelado de Intrusiones con Grafos Temporales: Optimizando el Threat Hunting Proactivo

Resumen. Este trabajo propone un modelo de grafo de conocimiento temporal tipado para telemetría de seguridad y un algoritmo de búsqueda en profundidad dirigido por hipótesis para Threat Hunting proactivo. A diferencia de las plataformas SIEM tradicionales, que almacenan la telemetría en formato tabular y requieren pivoteo iterativo manual para reconstruir cadenas de ataque, el enfoque propuesto materializa las relaciones causales entre eventos de seguridad como un multigrafo dirigido con nueve tipos de entidades y nueve tipos de relaciones. Los analistas codifican hipótesis de ataque como cadenas tipadas secuenciales, y el sistema retorna todos los caminos temporalmente válidos mediante cinco reglas de poda integradas (tipo de relación, tipo de entidad, monotonía causal, ventana temporal y k-simplicidad). La evaluación empírica sobre datasets reales—incluyendo telemetría Sysmon, logs de Azure Sentinel y el corpus Mordor (963K eventos)—demuestra aceleraciones de 1,747× a 11,600× respecto al baseline sin poda, con tiempos de búsqueda sub-milisegundo en grafos con más de 20,000 aristas. La comparativa operativa muestra una reducción de 8–20 minutos y múltiples consultas manuales por investigación (SIEM) a 1–2 minutos con una sola formulación de hipótesis.

Palabras clave: Threat Hunting, grafo de conocimiento temporal, telemetría de seguridad, búsqueda dirigida por hipótesis, MITRE ATT&CK.

1. Introducción

1.1 Threat Hunting: contexto y necesidad

El Threat Hunting es una disciplina proactiva que consiste en la búsqueda iterativa, basada en hipótesis, de amenazas avanzadas que han evadido controles automatizados (Hutchins et al., 2011). El proceso sigue un ciclo: formulación de hipótesis sobre comportamiento adversarial, análisis de registros, identificación de patrones anómalos y retroalimentación hacia controles defensivos. Marcos como MITRE ATT&CK (2026) proporcionan una taxonomía estandarizada de tácticas, técnicas y procedimientos (TTPs) que estructuran este proceso y permiten a los analistas formular hipótesis con base en comportamientos adversariales conocidos.

La efectividad del Threat Hunting depende de dos factores críticos: la calidad de los registros disponibles y la capacidad de las herramientas/personas para revelar relaciones causales entre eventos. En este segundo aspecto, las herramientas predominantes presentan limitaciones estructurales significativas.

1.2 Planteo del problema: divergencia estructural

Existe una diferencia fundamental entre cómo se almacena la telemetría de seguridad y cómo operan los adversarios. Los SIEM (Security Information and Event Management) almacenan registros en formato tabular optimizado para indexación y búsqueda por campos, mientras que los adversarios ejecutan cadenas causales de acciones: compromiso inicial, escalada de privilegios, movimiento lateral y exfiltración.

Las plataformas modernas ofrecen dashboards, alertas estadísticas y capacidades de análisis de comportamiento de usuarios (UEBA), pero estas agregaciones operan a nivel macro y omiten la causalidad a nivel micro entre eventos individuales. Esta divergencia genera barreras operativas concretas para el analista de SOC:

1. **Pivoteo iterativo ineficiente.** Reconstruir una cadena de ataque de L pasos requiere aproximadamente L consultas manuales secuenciales (2–5 minutos cada una), con un tiempo total de 8–20 minutos para L = 4. Cada consulta requiere la extracción manual de identificadores (IPs, nombres de usuario, PIDs) del resultado anterior para formular la siguiente.
2. **Sobrecarga cognitiva.** Al carecer de una representación visual de relaciones, el analista debe mantener mentalmente un "grafo" de dependencias causales entre eventos mientras navega resultados. La capacidad de la memoria de trabajo humana (7 ± 2 ítems según Miller, 1956) nos convierte en un cuello de botella cuando las cadenas superan los 3/4 pasos.
3. **Fragmentación del contexto.** El volumen de telemetría y el formato lineal de los logs provocan pérdida de contexto. Eventos causalmente relacionados quedan dispersos entre miles de registros irrelevantes, dificultando la visión integral del incidente.

1.3 Propuesta y objetivos

Para resolver esta problemática, este trabajo propone la transición del análisis tabular a un modelo de grafo temporal tipado que materializa explícitamente las relaciones causales entre eventos de seguridad. La contribución se materializa sobre un sistema completo—que incluya desde la ingestión de logs hasta la búsqueda de patrones de ataque—y permita al analista plantear hipótesis de caza como consultas de "caminos tipados" y pueda este obtener resultados visuales.

Los objetivos específicos son:

1. Definir un modelo de datos basado en un multigrafo dirigido, etiquetado y temporal, capaz de representar telemetría heterogénea (Sysmon, Azure Sentinel, logs genéricos, CSV) bajo una estructura unificada alineada con MITRE ATT&CK.
2. Diseñar un algoritmo de búsqueda en profundidad con reglas de poda que explota las restricciones del dominio de ciberseguridad (tipado y temporalidad) para lograr el tiempo esperado lineal.
3. Validar el método sobre datasets realistas (> 900K eventos) y una reducción del Mean Time to Investigate (MTTI) respecto al flujo de trabajo SIEM tradicional.
4. Implementar el método sobre una herramienta funcional que integre ingestión, búsqueda, scoring de anomalías y visualización de grafos para uso en SOCs.
5. Integrar clasificación de amenazas basada en redes neuronales de grafos (GNN) que opere sobre el mismo grafo de conocimiento, cerrando el ciclo entre detección automática e investigación dirigida por hipótesis, con aceleración opcional por NPU/GPU mediante DirectML.
6. Integración de LLM mediante MCP: Implementar un servidor Model Context Protocol (MCP) para que los asistentes de IA utilicen las operaciones del motor de búsqueda y análisis de grafos como "herramientas".

2. Estado del arte

2.1 Herramientas SIEM y sus limitaciones para Threat Hunting

La piedra angular de los Centros de Operaciones de Seguridad (SOC) han sido los SIEM—Splunk, Microsoft Sentinel, Elastic Security—que almacenan telemetría en formato tabular optimizado para búsqueda full-text y agregaciones estadísticas. Si bien estas plataformas ofrecen capacidades de búsqueda, dashboards y alertas, su diseño no materializa relaciones causales entre eventos.

Para mitigar la fatiga de alertas, la industria incorporó análisis estadísticos y heurísticos. Las soluciones UEBA (User and Entity Behavior Analytics) detectan anomalías en patrones de comportamiento, proporcionando conciencia situacional a nivel macro. Sin embargo, no resuelven el problema de la reconstrucción causal micro: determinar qué proceso específico generó qué conexión de red hacia qué servidor, y qué sucedió después.

Las capacidades de grafos en plataformas comerciales—como los Investigation Graphs de Microsoft Sentinel o los Causality Views de Palo Alto Cortex XDR—representan un avance, pero operan como capas de visualización sobre datos tabulares subyacentes. No explotan formalmente la estructura del grafo para búsqueda de patrones, y las travesías con restricciones temporales típicamente superan los 100 ms para más de 3 saltos.

2.2 Enfoques de grafos en ciberseguridad

BloodHound (Robbins et al., 2017) demostró el poder de la modelización relacional en Active Directory, permitiendo descubrir rutas de escalada de privilegios mediante travesía de grafos de permisos. Sin embargo, opera sobre relaciones estáticas—permisos y membresías de grupos—sin componente temporal, lo que lo

limita a análisis de configuración y no a detección de actividad maliciosa en tiempo real.

SLEUTH (Hossain et al., 2017) y HOLMES (Milajerdi et al., 2019a) construyen grafos de proveniencia a partir de llamadas de sistema y aplican reglas de detección basadas en flujos de información. Poirot (Milajerdi et al., 2019b) alinea grafos de ataque provenientes de inteligencia de amenazas contra logs de auditoría mediante alineación aproximada nodo-a-nodo. ProvDetector (Wang et al., 2020) utiliza embeddings de caminos para detección de anomalías en grafos de proveniencia. Unicorn (Han et al., 2020) emplea histosketches para detección de APT a nivel de grafo completo.

Estos sistemas comparten tres características que los distinguen del enfoque propuesto: (1) operan exclusivamente sobre logs de llamadas de sistema (auditoría del kernel), no sobre telemetría heterogénea de múltiples fuentes; (2) emplean detección automática basada en reglas predefinidas o modelos aprendidos, sin permitir al analista formular hipótesis ad-hoc; y (3) no proporcionan una interfaz de búsqueda interactiva para exploración dirigida por el analista.

El trabajo más cercano en espíritu es Poirot, que también realiza alineación de grafos de ataque. Sin embargo, Poirot usa alineación aproximada contra un grafo de consulta fijo derivado de reportes CTI, mientras que nuestro enfoque permite matching exacto de caminos temporales con hipótesis formuladas dinámicamente por el analista, priorizando interpretabilidad y control humano.

2.3 Subgrafo temporal

En el ámbito de algoritmos de grafos, Ti&To (Kim et al., 2022) intercala filtrado topológico y temporal para isomorfismo de subgrafos temporal, análogo a nuestra poda integrada tipo y temporal. Sin embargo, la propuesta de este trabajo especializa el problema a consultas de caminos tipados secuenciales, permitiendo una estructura DFS (búsqueda en profundidad) lineal sin la sobrecarga del isomorfismo general. El método edge-driven cronológico (Cai et al., 2023) procesa aristas en orden temporal, y SAL/SAM (Lai et al., 2022) ofrece ejecución distribuida streaming. Estos enfoques están diseñados para grafos de propósito general y no explotan las restricciones específicas del dominio de ciberseguridad (tipos de entidades y relaciones de seguridad, distribución temporal de eventos) que permiten las optimizaciones de nuestro enfoque.

2.4 Detección basada en aprendizaje sobre grafos

Trabajos recientes aplican Graph Neural Networks (GNN) a grafos de proveniencia para detección automática de APT. ThreaTrace (Wang et al., 2022) entrena modelos GNN para clasificar nodos en grafos de proveniencia como benignos o maliciosos. MAGIC (Zengy et al., 2023) utiliza aprendizaje de representaciones enmascarado sobre grafos para detección de APT a nivel de sistema. Sin embargo, estos sistemas operan como detectores de caja negra sin capacidad de investigación interactiva: generan alertas pero no permiten al analista formular hipótesis ad-hoc ni explorar el contexto causal de la alerta. Nuestra propuesta cierra esta brecha integrando la clasificación GNN como un componente del scoring de anomalía, de modo que la detección automática alimenta la priorización de caminos en la búsqueda dirigida por hipótesis, y viceversa.

3. Modelo de grafo para telemetría de seguridad

3.1 Definición de grafo de conocimiento temporal de amenazas

Grafo de conocimiento temporal de amenazas: Una tupla G = (V, E, τ, σ_V, σ_E, μ_V, μ_E) donde: V es un conjunto finito de vértices (entidades de seguridad); E es una bolsa (multiconjunto) sobre V × V con identificadores únicos; τ: E → ℤ asigna marcas temporales Unix; σ_v: V → Σ_v y σ_v: E → Σ_E asignan tipos; μ_V y μ_E asignan diccionarios de metadatos.

Alfabetos de tipos de seguridad: Σ_V = {IP, Host, User, Process, File, Domain, Registry, Email, URL}; Σ_E = {Auth, Connect, Execute, Read, Write, DNS, Spawn, Modify, Alert}. Con |Σ_V| = |Σ_E| = 9.

La elección de E como multiconjunto es fundamental para el dominio: un mismo usuario puede autenticarse al mismo host repetidas veces, cada evento con marca temporal y contexto distintos. Esto distingue el modelo de enfoques estáticos como BloodHound, donde una arista «tiene-permiso» es binaria.

Estos alfabetos de tipos básicos fueron diseñados para cubrir las entidades y relaciones más frecuentes en telemetría de seguridad según la taxonomía MITRE ATT&CK. Cada tipo de relación mapea naturalmente a una o más técnicas ATT&CK: Auth cubre técnicas de acceso con credenciales válidas (T1078), Execute y Spawn cubren ejecución (TA0002), Connect y DNS cubren comunicaciones C2 (TA0011), y Read/Write cubren acceso a datos y exfiltración (TA0009, TA0010).

3.2 Estructuras y mapeo de logs a grafos

Definición 3.3 (Tripleta de seguridad). Una tripleta (s, r, d) codifica entidad fuente s ∈ V, tipo de relación r ∈ Σ_E con marca temporal τ, y entidad destino d ∈ V. Un parser P: Log → P(Tripleta) transforma registros de log en conjuntos de tripletas.

El sistema implementa cuatro parsers para fuentes de telemetría comunes:

- **Sysmon:** Procesa eventos de Windows Sysmon (creación de procesos, conexiones de red, acceso a archivos, consultas DNS, modificaciones de registro). Cada tipo de evento (EventID 1, 3, 7, 11, 13, 22) se mapea a tripletas específicas. Por ejemplo, un EventID 1 (Process Create) genera una tripleta (ParentProcess, Spawn, ChildProcess).
- **Azure Sentinel:** Procesa logs del SIEM cloud de Microsoft, incluyendo SecurityEvent (eventos de Windows reenviados), SigninLogs (autenticaciones Azure AD), AzureActivity (operaciones en recursos cloud) y CommonSecurityLog (datos CEF). Los EventIDs 4624/4625 (logon exitoso/fallido) generan tripletas (User, Auth, Host).
- **Genérico JSON:** Acepta registros JSON con campos configurables, permitiendo la ingestión de fuentes no estandarizadas.
- **CSV:** Procesa datos tabulares exportados de cualquier herramienta, transformando columnas en entidades y relaciones según mapeo configurable.

La política de ingesta emplea upsert para vértices (deduplicación por identificador, fusión de metadatos) e inserción incondicional para aristas, preservando la granularidad temporal completa. La resolución de entidades emplea identificadores compuestos—por ejemplo, user@domain para usuarios o host\process:pid:timestamp para procesos—para desambiguar entidades homónimas como un PID reutilizado en diferentes sesiones.

3.3 Definiciones sobre hipótesis de caza como consultas de caminos

El concepto central del sistema es la hipótesis de caza: una secuencia tipada que describe un patrón de ataque esperado.

Hipótesis de caza: Una secuencia $H = [(v_0, r_1, v_1), (v_1, r_2, v_2), \ldots, (v\_\{L-1\}, r\_L, v\_L)]$ donde $v\_i \in \Sigma\_V \cup \{*\}$ y $r\_j \in \Sigma\_E \cup \{*\}$, con * denotando comodín. La hipótesis es consistente si los tipos destino-fuente de pasos consecutivos coinciden.

Camino válido: Un camino $p = [e_1, e_2, \ldots, e\_L]$ sobre aristas del grafo es válido para H si: (1) los tipos de relación y entidad coinciden con H; (2) se cumple monotonía causal $\tau(e\_i) \leq \tau(e\_\{i+1\})$; (3) opcionalmente, una ventana temporal [t_min, t_max]; y (4) k-simplicidad: ningún vértice aparece más de k veces.

Esta formalización permite al analista expresar directamente hipótesis derivadas de MITRE ATT&CK. Por ejemplo, la hipótesis de movimiento lateral «un usuario se autentica en un host, desde donde se ejecuta un proceso que se conecta a un dominio externo» se codifica como:

H = [(User, Auth, Host), (Host, Execute, Process), (Process, Connect, Domain)]

La monotonía causal ($\tau(e\_i) \leq \tau(e\_\{i+1\})$) es una restricción natural del dominio de seguridad: un proceso no puede conectarse a un servidor C2 antes de ser ejecutado. Esta restricción, aparentemente simple, resulta ser un factor de poda extremadamente efectivo, como se demuestra en la sección de evaluación.

4. Arquitectura del sistema

4.1 Visión general

El sistema se implementa como una aplicación de escritorio con tres capas: (1) un núcleo en Rust (The Rust Team, 2026) que contiene toda la lógica de dominio (grafo en memoria, parsers, algoritmos de búsqueda, scoring endógeno de anomalía y bridge GNN para extracción de features de subgrafos); (2) un backend Tauri v2 que expone el núcleo como comandos invocables y una API HTTP REST con autenticación Bearer token para integración con herramientas externas; y (3) un frontend React con TypeScript (Microsoft, 2026) y Cytoscape.js (Franz et al., 2016). para visualización interactiva de grafos. Adicionalmente, un servidor MCP (Model Context Protocol) expone las operaciones del motor como herramientas para asistentes de IA, permitiendo que agentes inteligentes ejecuten hipótesis, expandan nodos y analicen resultados programáticamente.

La elección de un grafo en memoria con estructuras nativas (HashMaps, vectores) en lugar de una base de datos de grafos externa, Neo4j (Neo4j, Inc., 2026) y ArangoDB (ArangoDB GmbH., 2026) responde al patrón de acceso del Threat Hunting: se ingieren datos de sesión, se ejecutan búsquedas interactivas intensivas, y se descartan al finalizar. La latencia de una travesía en Neo4j con restricciones temporales típicamente supera los 100 ms para más de 3 saltos con filtros temporales, mientras que el enfoque en memoria logra tiempos sub-milisegundo.

Además, el control explícito sobre el algoritmo permite optimizaciones específicas del dominio difíciles de expresar en lenguajes de consulta declarativos. La monotonía causal requiere mantener estado sobre la última marca temporal durante la travesía, operación que en Cypher requeriría procedimientos almacenados.

Para fomentar la transparencia y el uso en entornos de SOC, el código fuente, incluyendo los parsers y el motor de búsqueda en Rust, se ha publicado bajo una licencia GNU GPLv3 en https://github.com/Base4Security/GraphHunter

4.2 Estructuras de datos

El motor mantiene cinco estructuras de datos en memoria:

1. **Entidades** (HashMap<Id, Entity>): índice principal de vértices por identificador único.
2. **Adyacencia saliente** (HashMap<Id, Vec<Relation>>): lista de adyacencia para travesía directa.
3. **Índice por tipo de entidad** (HashMap<Σ_V, HashSet<Id>>): localiza todos los vértices de un tipo dado en O(1), fundamental para la fase de semillas del algoritmo.
4. **Adyacencia inversa:** para travesía en dirección opuesta (BFS bidireccional).
5. **Índice secundario de relaciones** (HashMap<(Id, Σ_E), Vec<Relation>>): recupera en O(1) las aristas salientes de un vértice con un tipo de relación específico, evitando el escaneo O(deg) de la lista de adyacencia completa. Impacto especialmente alto en hubs de alto grado.

4.3 Algoritmo de búsqueda: DFS con poda por dominio

El algoritmo central, Temporal-Hypothesis-Search, opera en dos fases:

Fase 1 (Semillas): Obtiene del índice de tipos todos los vértices cuyo tipo coincide con v_0 de la hipótesis.

Fase 2 (DFS con backtracking): Para cada semilla, lanza una búsqueda en profundidad aplicando cinco reglas de poda en cada paso:

1. **Poda de tipo de relación:** Descarta aristas cuyo tipo no coincide con r_i del paso actual. Cuando el índice secundario de relaciones está disponible, esta poda se convierte en un pre-filtro O(1).
2. **Poda de tipo de entidad destino:** Descarta vértices destino cuyo tipo no coincide con v_i del paso actual.
3. **Poda de monotonía causal:** Exige $\tau(e\_i) \leq \tau(e\_\{i+1\})$, garantizando que la cadena de eventos respeta el orden temporal.
4. **Poda de ventana temporal:** Restringe a $\tau(e\_i) \in [t\_min, t\_max]$ cuando el analista especifica un período de interés.
5. **Poda de k-simplicidad:** Impide que un vértice aparezca más de k veces en el camino (k=1 por defecto). Permite detectar patrones cíclicos (callbacks C2, movimiento lateral con retorno) cuando k > 1.

En el peor caso teórico (un solo tipo de entidad, un solo tipo de relación, sin restricción temporal), el problema se reduce a búsqueda de caminos en grafos dirigidos, que es NP-completo por reducción desde Camino Hamiltoniano Dirigido (Karp, 1972). Sin embargo, las restricciones del dominio de ciberseguridad hacen el problema tratable en la práctica:

Tratabilidad práctica: Con $|\Sigma\_V| = |\Sigma\_E| = 9$ tipos, la poda por tipo de relación retiene una fracción esperada $1/9 \approx 11\%$ de aristas, y la poda por tipo de entidad retiene otra fracción 1/9. La monotonía causal retiene en promedio ~50% de las aristas restantes. El factor de ramificación efectivo resulta $b\_eff = \bar{d} \cdot (1/9) \cdot (1/9) \cdot (1/2) \approx \bar{d}/162$, que cae por debajo de 1 cuando el grado medio $\bar{d} < 162$—condición satisfecha en todos los grafos de telemetría prácticos. Cuando $b\_eff < 1$, el árbol de

búsqueda converge y el tiempo esperado es $O(|S_0| \cdot c)$ donde S_0 son las semillas y c es una constante, resultando lineal en el tamaño del grafo.

4.4 Scoring y priorización de amenazas

Para guiar al analista en la formulación de hipótesis y la priorización de resultados, el sistema calcula tres métricas de centralidad:

Centralidad de grado normalizada: $score\_deg(v) = (deg\_in(v) + deg\_out(v)) / (2 \cdot |V|)$. Identifica entidades altamente conectadas (hubs potencialmente comprometidos).

PageRank temporal: Incorpora decaimiento exponencial por antigüedad de arista: $w(e) = \exp(-\lambda(\tau\_max - \tau(e)))$. Prioriza entidades involucradas en actividad reciente. Implementación: iteración de potencia con damping $\alpha = 0.85$ (Page et al., 1999).

Betweenness centrality: Algoritmo de Brandes (2001), exacto en $O(|V| \cdot |E|)$ o aproximado con k fuentes muestreadas. Identifica entidades que actúan como intermediarios en múltiples caminos—nodos de pivote potenciales en cadenas de movimiento lateral.

El score compuesto $S = w_1 \cdot deg + w_2 \cdot PR + w_3 \cdot BC$ (con pesos por defecto 0.3, 0.4, 0.3) permite al analista identificar rápidamente las entidades más relevantes para iniciar una investigación, complementando el enfoque hipótesis-dirigido con señales automáticas de anomalía.

4.5 Scoring endógeno de anomalía por camino

Complementariamente a la centralidad por nodo, el sistema implementa un scoring de anomalía que opera sobre caminos completos retornados por la búsqueda. El objetivo es priorizar automáticamente los caminos que involucran entidades y relaciones estadísticamente inusuales, sin requerir inteligencia de amenazas externas. El score compuesto $S(p) \in [0, 1]$ combina cinco componentes con pesos configurables: (1) Entity Rarity (ER): entidades observadas pocas veces reciben score alto, calculado como $ER(v) = 1 - \ln(freq(v)) / \ln(f\_max)$; (2) Edge Rarity (EdgeR): conexiones únicas entre pares de entidades—una conexión nunca vista entre un usuario y un host remoto recibe score mayor que una autenticación rutinaria; (3) Neighborhood Concentration (NC): basado en la entropía de Shannon normalizada del vecindario—un nodo que solo se conecta con entidades de un solo tipo sugiere infraestructura dedicada como servidores C2; (4) Temporal Novelty (TN): entidades que aparecieron recientemente en el dataset; (5) GNN Threat: clasificación de subgrafos k-hop mediante redes neuronales de grafos. En nuestro caso específicamente se ha utilizado un modelo de GCN (Graph Convolutional Network) pero el sistema no está limitado específicamente al uso de ella sino de otros tipos de GNN.

Un modelo ONNX clasifica el vecindario de cada entidad en 5 categorías: Benigno, Exfiltración, C2, Movimiento Lateral, Escalación de Privilegios. El puntaje de amenaza es $1 - P(\text{Benigno})$ tras softmax. La retroalimentación es bidireccional: los scores endógenos (ER, TN, NC) alimentan el vector de features de la GNN como dimensiones de entrada, y la salida de la GNN alimenta el puntaje compuesto. Esto crea un ciclo cerrado donde la detección estadística y la clasificación aprendida se potencian mutuamente. El puntaje también guía la poda durante la búsqueda DFS: nodos con anomalías score bajo pueden ser descartados tempranamente, reduciendo el espacio de búsqueda.

Entrenamiento del modelo GCN. El modelo se entrena sobre el dataset StreamSpot (Manzoor et al., 2016), que contiene 600 grafos de procedencia de sistema: 500 benignos y 100 correspondientes a ataques “drive-by download”. El pipeline de entrenamiento está definido en tres etapas:

1. Se parsean las trazas de auditoría a un grafo dirigido (~5 millones de nodos, ~7,6 millones de aristas).

2. Para cada nodo centro se extrae un subgrafo k-hop (k = 2, máximo K_max = 32 nodos) y se featuriza en un vector de 1.536 floats: 32 × 16 features de nodo (one-hot del tipo de entidad, grado normalizado, flag de centro) concatenados con la matriz de adyacencia 32 × 32 aplanada — idéntico al formato que produce gnn_bridge.rs en tiempo de ejecución.

3. Se entrena una red convolucional de grafos (GCN) de dos capas siguiendo la formulación de Kipf y Welling (2017): $H^{(l+1)} = \sigma(\hat{D}^{-1}\hat{A}\, H^{(l)} W^{(l)})$, donde $\hat{A} = A + I$ incorpora self-loops y $\hat{D}$ es la matriz diagonal de grados. Tras el message passing, se aplica mean-pooling sobre los K_max nodos y un clasificador lineal produce 5 logits (Benigno, Exfiltración, C2, Movimiento Lateral, Escalación de Privilegios).

El entrenamiento utiliza CrossEntropyLoss con pesos inversamente proporcionales a la frecuencia de cada clase, optimizador Adam (lr = 10^{-3}, weight decay = 10^{-4}) y early stopping con paciencia de 10 épocas. El modelo entrenado (~5.000 parámetros) se exporta a ONNX (opset 17) y es cargado en tiempo de ejecución por npu_scorer.rs mediante ONNX Runtime, con soporte opcional de aceleración por DirectML.

4.6 Integración con agentes de IA vía MCP

El sistema expone una API HTTP REST en localhost con autenticación Bearer token, permitiendo que herramientas externas consulten el grafo sin usar la interfaz gráfica. Sobre esta API, un servidor MCP (Model Context Protocol) traduce las operaciones del motor en herramientas invocables por asistentes de IA. Las primitivas expuestas están orientadas al flujo de trabajo de investigación: búsqueda de entidades, expansión de vecindarios k-hop, ejecución de hipótesis con ranking por anomalía, obtención de detalles de nodos con scores GNN, y creación de notas de análisis. Esta arquitectura permite que un analista delegue tareas exploratorias a un agente de IA que opera directamente sobre el grafo de la sesión activa, manteniendo el humano en el loop para decisiones de alto nivel.

5. Evaluación

5.1 Entorno experimental

Todas las mediciones utilizan compilaciones en modo release de Rust sobre un solo core (AMD Ryzen 5 PRO 5650U, 16 GB RAM). No se emplea paralelismo en la fase de búsqueda para obtener mediciones reproducibles.

Datasets reales. Tres datasets de escala creciente:

APT simulation (30 eventos Sysmon): Cadena APT completa desde acceso inicial (phishing) hasta exfiltración, con procesos encadenados y conexiones de red. Simula técnicas T1566, T1059, T1071, T1041 de MITRE ATT&CK.

Sentinel simulation (32 eventos Azure Sentinel): Escenario de compromiso cloud-to-on-premises con autenticaciones Azure AD, actividad en recursos cloud y movimiento lateral hacia infraestructura local.

Mordor combined (963,000 eventos): Telemetría real del proyecto Mordor (Rodriguez, 2020), con sesgo extremo de tipos (85% nodos Registry) y distribución de grado power-law con d_max = 9,411. Este dataset estresa particularmente las podas de tipo y la gestión de hubs de alto grado.

5.2 Caso de estudio 1: Detección de movimiento lateral

Consideremos un escenario de movimiento lateral mediante Pass-the-Hash, una técnica frecuente en APTs donde un atacante compromete un endpoint vía phishing, extrae credenciales con herramientas como Mimikatz, y las utiliza para autenticarse en servidores internos donde ejecuta comandos de exfiltración.

Flujo en SIEM tradicional. El analista detecta un pico de autenticaciones fallidas seguido de una exitosa desde una IP inusual. Para reconstruir la cadena causal debe: (1) buscar la IP origen en los logs de firewall; (2) extraer el nombre de usuario y buscar eventos de proceso asociados; (3) identificar el proceso sospechoso y buscar sus conexiones de red; (4) rastrear conexiones posteriores a otros hosts; (5) buscar autenticaciones en los hosts destino; (6) buscar procesos ejecutados en el destino. Cada paso requiere una consulta independiente con extracción manual de identificadores del resultado anterior.

Flujo con el enfoque propuesto. La investigación se codifica como una sola hipótesis:

$$H = [(\text{IP}, \text{Connect}, \text{Host}), (\text{Host}, \text{Auth}, \text{User}), (\text{User}, \text{Execute}, \text{Process}), (\text{Process}, \text{Write}, \text{File})]$$

El algoritmo retorna todos los caminos válidos con monotonía causal en milisegundos. El analista visualiza la cadena completa: IP atacante → Host comprometido → Usuario admin → Mimikatz.exe → archivo exfiltrado. La ventana temporal acota la búsqueda al período sospechoso, y el scoring de anomalía identifica automáticamente los nodos hub.

5.3 Caso de estudio 2: Ransomware fileless

En un escenario de ransomware fileless, un documento de Office ejecuta una macro que invoca PowerShell (T1059.001). PowerShell descarga un payload en memoria (T1105), inyecta código en svchost.exe (T1055), realiza consultas DNS a un dominio C2 (T1071.004), y cifra archivos (T1486).

La hipótesis correspondiente:

$$H = [(\text{Process}, \text{Spawn}, \text{Process}), (\text{Process}, \text{DNS}, \text{Domain}), (\text{Process}, \text{Write}, \text{File})]$$

La ventaja del grafo se manifiesta especialmente en la detección de inyección de código. En logs lineales, la relación entre PowerShell y svchost.exe se pierde entre miles de eventos de spawn de procesos. En el grafo, la arista Spawn con su marca temporal conecta directamente los procesos, y la BFS bidireccional desde el dominio C2 revela inmediatamente los procesos contactantes y sus procesos padres.

5.4 Caso de estudio 3: Compromiso cloud-to-on-premises

Un escenario emergente en entornos híbridos: un atacante obtiene credenciales de Azure AD mediante phishing, accede a recursos cloud, y desde allí pivota hacia infraestructura on-premises. La hipótesis:

$$H = [(\text{User}, \text{Auth}, \text{Host}), (\text{Host}, \text{Connect}, \text{IP}), (\text{IP}, \text{Auth}, \text{Host})]$$

El parser de Sentinel normaliza tanto los SigninLogs (Azure AD) como los SecurityEvent (Windows on-premises) al mismo modelo de grafo, permitiendo trazar

la cadena completa a través de ambos entornos en una sola consulta. Esta capacidad de unificar telemetría heterogénea bajo un modelo común es una ventaja diferencial frente a herramientas que operan sobre una sola fuente de datos.

5.5 Resultados de rendimiento

Tabla 1. Rendimiento de la búsqueda sobre datasets reales

Dataset	|V|	|E|	L	Tiempo	Speedup	Resultados
APT	24	30	2	0.005 ms	7.9×	0
APT	24	30	5	0.010 ms	4.6×	7
Sentinel	28	32	3	0.006 ms	2.1×	3
Mordor	772	21,963	2	4.3 ms	1,747×	438
Mordor	772	21,963	3	4.4 ms	5,121×	42
Mordor	772	21,963	5	4.4 ms	11,600×	0

El dataset Mordor, con 963K eventos originales comprimidos a 772 nodos y 21,963 aristas, presenta el caso más exigente por su distribución power-law (d_max = 9,411). Aun así, la búsqueda con L=2 completa en 4.3ms visitando solo 13 nodos frente a 22,735 del baseline sin poda.

5.6 Efectividad de las reglas de poda

Tabla 2. Porcentaje de candidatos eliminados por cada regla de poda

Regla de poda	Teórico	Mordor
Tipo de relación (pre-filtro por índice)	88.9%	~0% *
Tipo de entidad destino	88.9%	92.3%
Monotonía causal	50.0%	38.1%
k-simplicidad	<1%	0%

* Pre-filtrada por índice secundario antes del bucle de podas.

La poda de tipo de relación muestra 0% en el bucle porque el índice secundario la resuelve antes de la iteración—las aristas con tipo incorrecto nunca entran al bucle de podas. La poda de tipo de entidad es el filtro dominante en ejecución, eliminando >90% de candidatos en datos reales. La monotonía causal aporta un 38% adicional en Mordor, mayor que en grafos sintéticos, lo que refleja la distribución temporal no uniforme de eventos reales de seguridad.

5.7 Análisis comparativo: SIEM vs. Grafo

Tabla 3. Comparativa operativa para investigación de movimiento lateral (L=4)

Métrica	SIEM tradicional	Enfoque propuesto
Consultas/operaciones	4 secuenciales	1 hipótesis
Tiempo de formulación	—	1–2 min
Tiempo de ejecución	8–20 min total	< 5 ms
Extracción manual de IDs	Sí, en cada paso	No
Descubrimiento exhaustivo	No (3–5 caminos)	Sí (todos)
Escalamiento con L	Lineal (~L × 2–5 min)	Constante
Visualización causal	No nativa	Grafo interactivo

La diferencia más significativa es el escalamiento: en SIEM, el tiempo de investigación crece linealmente con la longitud de la cadena de ataque, mientras que en el enfoque propuesto el tiempo de ejecución es esencialmente constante respecto a L (dado que $b_{eff} < 1$). Para investigaciones complejas con $L \geq 4$, la reducción supera un orden de magnitud.

6. Discusión

6.1 Resolución de las barreras operativas

A continuación se detalla cómo el sistema propuesto resuelve cada una de las tres barreras identificadas en la Sección 1.2.

Pivoteo iterativo ineficiente. El grafo temporal materializa todas las relaciones entre entidades en el momento de la ingestión. Lo que en un SIEM requiere L consultas manuales secuenciales —extrayendo identificadores del resultado anterior para formular la siguiente— se reemplaza por una única formulación de hipótesis tipada (por ejemplo, User →Auth→ Host →Execute→ Process →Connect→ IP). El algoritmo DFS recorre automáticamente el grafo siguiendo las restricciones de tipo y temporalidad, retornando todos los caminos válidos en tiempo sub-milisegundo. El analista ya no necesita extraer manualmente IPs, PIDs o nombres de usuario entre consultas: el grafo los conecta estructuralmente. El resultado es una reducción del MTTI de 8–20 minutos a 1–2 minutos para cadenas de 4 pasos.

Sobrecarga cognitiva. El sistema externaliza el modelo mental que el analista debía mantener en memoria de trabajo mediante tres mecanismos concretos: (a) el grafo de conocimiento almacena las relaciones causales como aristas tipadas y temporales, eliminando la necesidad de que el analista las recuerde; (b) la visualización interactiva con Cytoscape renderiza el grafo causal como un diagrama navegable donde el analista puede expandir vecindarios con un clic, en contraste con las tablas de resultados lineales del SIEM; y (c) los scores de anomalía (rareza de entidad, concentración de vecindario, novedad temporal, y opcionalmente la clasificación GNN) priorizan automáticamente los nodos de interés, reduciendo la carga de decisión y mitigando el cuello de botella de los 7 ± 2 ítems de Miller (1956).

Fragmentación del contexto. Tres mecanismos abordan la dispersión de eventos relevantes: (a) los parsers multi-formato (Sysmon, Azure Sentinel, JSON genérico, CSV) ingestan telemetría heterogénea bajo una ontología unificada de nueve tipos de entidades y nueve tipos de relaciones, eliminando los silos de datos entre fuentes; (b) la deduplicación por identidad de entidad garantiza que una misma IP, host o usuario que aparece en logs de distintas fuentes se representa como un único nodo con todas sus relaciones preservadas, reconstruyendo el contexto integral; y (c) la búsqueda por hipótesis filtra los caminos relevantes del volumen total de telemetría, retornando únicamente los subgrafos que coinciden con el patrón de ataque especificado. Eventos causalmente relacionados que en el formato lineal del SIEM quedarían separados por miles de registros irrelevantes aparecen como nodos adyacentes en el grafo.

Adicional: Descubrimiento exhaustivo de relaciones latentes. La enumeración exhaustiva (THS-Enum) revela rutas de ataque que el pivoteo manual no descubriría. En el dataset Mordor, la hipótesis con L=2 retorna 438 caminos válidos.

Un analista típicamente explorará 3–5 antes de considerar completa la investigación, perdiendo potencialmente cientos de indicadores.

Ciclo cerrado detección-investigación. La integración de clasificación GNN como componente del scoring de anomalía unifica dos capacidades que tradicionalmente requieren herramientas separadas: la detección automática de comportamiento malicioso y la investigación interactiva dirigida por hipótesis. Cuando la GNN detecta un subgrafo anómalo, el analista puede inmediatamente formular una hipótesis sobre el mismo grafo para investigar el contexto causal, sin exportar datos ni reconstruir el grafo en otra herramienta. Adicionalmente, un servidor MCP (Model Context Protocol) expone las operaciones del motor como herramientas para asistentes de IA, permitiendo que agentes inteligentes ejecuten hipótesis, expandan nodos y analicen resultados programáticamente.

6.2 Limitaciones

Escalabilidad en memoria. El enfoque en memoria limita el tamaño del dataset al RAM disponible. Con 963K eventos de Mordor, el grafo consume ~200 MB. Para entornos empresariales con volúmenes mayores se requiere compactación temporal (ya implementada, con ~40% de reducción en aristas) o ventanas deslizantes.

Visualización a escala. Cuando el grafo supera ~500 nodos, la visualización directa pierde utilidad (problema «hairball»). Se mitiga con la búsqueda por hipótesis (que retorna subgrafos específicos), BFS acotada, y los scores de anomalía para filtrar.

Dependencia de la hipótesis. El enfoque es tan efectivo como las hipótesis formuladas. Un analista sin experiencia o sin conocimiento del framework ATT&CK podría no formular las hipótesis correctas. Se complementa con los scores de anomalía que proveen señales automáticas, y se plantea como trabajo futuro la generación automática de hipótesis mediante aprendizaje automático sobre grafos.

Betweenness sobre proyección estática. La betweenness centrality opera sobre la proyección estática del multigrafo, ignorando el orden temporal. Esto puede sobreestimar la importancia de nodos cuyas rutas no son temporalmente factibles. Computar betweenness temporal bajo semántica de camino más temprano es #P-hard (Buss et al., 2022), por lo que la aproximación estática es un compromiso deliberado de tratabilidad.

Dependencia de modelos GNN pre-entrenados. El componente GNN Threat requiere un modelo ONNX entrenado sobre grafos de proveniencia etiquetados. La calidad de la clasificación depende de la representatividad del dataset de entrenamiento respecto al entorno del SOC. Modelos entrenados sobre telemetría de endpoints Linux pueden no generalizar a entornos Windows o cloud. Se mitiga con el peso configurable W5 = 0 (desactivado por defecto) y la posibilidad de fine-tuning sobre datos propios del SOC.

7. Conclusiones y trabajo futuro

Este trabajo demuestra que la modelización de telemetría de seguridad como grafo temporal tipado, combinada con búsqueda de hipótesis por DFS con cinco reglas de poda, reduce significativamente el tiempo y la complejidad cognitiva de las investigaciones de Threat Hunting. Las restricciones inherentes al dominio de ciberseguridad—tipado de entidades y relaciones, causalidad temporal—convierten un problema teóricamente intratable en uno con tiempo esperado lineal en la práctica.

Los resultados principales son: (1) aceleraciones de 1,747× a 11,600× sobre baseline sin poda en datasets reales; (2) tiempos de búsqueda sub-milisegundo en grafos con más de 20,000 aristas; (3) reducción de un orden de magnitud en el MTTI (de 8–20 minutos a 1–2 minutos para investigaciones de 4 pasos); (4) descubrimiento exhaustivo de caminos de ataque versus la exploración parcial del pivoteo manual; y (5) integración de clasificación GNN sobre subgrafos k-hop como quinto componente del scoring de anomalía, habilitando un ciclo cerrado de detección-investigación con retroalimentación bidireccional entre scoring endógeno y clasificación aprendida.

Se identifican las siguientes extensiones: Búsqueda en streaming: Extensión a ventanas deslizantes con manejo de eventos fuera de orden, siguiendo el enfoque de Lai et al. (2022). Generación automática de hipótesis: Explotación de los scores GNN y embeddings del grafo para sugerir hipótesis de caza sin intervención humana, complementando el catálogo ATT&CK existente. Integración con feeds de inteligencia de amenazas: Enriquecimiento automático de entidades con indicadores de compromiso (IoCs) provenientes de fuentes STIX/TAXII. Evaluación con analistas: Estudio de usabilidad con analistas de SOC para medir la reducción real de carga cognitiva mediante métricas como NASA-TLX. Entrenamiento de modelos GNN específicos por SOC: Fine-tuning de modelos ONNX sobre la telemetría propia de cada organización para capturar patrones organizacionales. Arquitectura distribuida: Separación entre fast-path en endpoint (inferencia NPU en tiempo real) y servidor de investigación (grafo temporal + DFS) para entornos enterprise.

Referencias

ArangoDB GmbH. (2026). ArangoDB: The Multi-Model NoSQL Database (Versión X.X) [Software de computación]. https://www.arangodb.com/

Brandes, U. (2001). A faster algorithm for betweenness centrality. Journal of Mathematical Sociology, 25(2), 163–177.

Buss, S., Molter, H., Niedermeier, R. y Renken, M. (2022). Computing optimal temporal walks under waiting-time constraints. Algorithmica, 84, 2754–2802.

Cai, T., Li, C. y Pei, J. (2023). Exact subgraph matching via edge-ordered temporal graphs. En Proceedings of IEEE ICDE (pp. 733–744). IEEE.

Franz, M., Lopes, C. T., Huck, G., Dong, Y., Sumer, O., y Bader, G. D. (2016). Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics*, *32*(2), 309–311. https://doi.org/10.1093/bioinformatics/btv557

Han, X., Pasquier, T., Bates, A., Mickens, J. y Seltzer, M. (2020). Unicorn: Runtime provenance-based detector for advanced persistent threats. En Proceedings of NDSS. Internet Society.

Hossain, M. N., Milajerdi, S. M., Wang, J., Sadeghian, B., Gjomemo, R., Sekar, R., Stoller, S. y Venkatakrishnan, V. N. (2017). SLEUTH: Real-time attack scenario reconstruction from COTS audit data. En Proceedings of USENIX Security Symposium (pp. 487–504). USENIX.

Hutchins, E. M., Cloppert, M. J. y Amin, R. M. (2011). Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. En Proceedings of the 6th International Conference on Information Warfare and Security. Academic Publishing International.

Karp, R. M. (1972). Reducibility among combinatorial problems. En R. E. Miller y J. W. Thatcher (Eds.), Complexity of Computer Computations (pp. 85–103). Plenum Press.

Kim, H., Lee, Y. y Shin, K. (2022). Temporal subgraph isomorphism. En Proceedings of IEEE ICDE (pp. 532–543). IEEE.

Lai, S., Li, X., Lu, J., Luo, C. y Zhao, D. (2022). Scalable temporal subgraph pattern matching. Proceedings of the VLDB Endowment, 15(11), 2647–2660.

Meta Open Source. (2026). React: A JavaScript library for building user interfaces. https://react.dev/

Microsoft. (2026). TypeScript: JavaScript with syntax for types.https://www.typescriptlang.org/

Milajerdi, S. M., Gjomemo, R., Eshete, B., Sekar, R. y Venkatakrishnan, V. N. (2019a). HOLMES: Real-time APT detection through correlation of suspicious information flows. En Proceedings of IEEE Symposium on Security and Privacy (pp. 1137–1152). IEEE.

Milajerdi, S. M., Eshete, B., Gjomemo, R. y Venkatakrishnan, V. N. (2019b). Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting. En Proceedings of ACM CCS (pp. 1795–1812). ACM.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review, 63(2), 81–97.

MITRE ATT&CK®. (2026). *MITRE ATT&CK*. https://attack.mitre.org/

Neo4j, Inc. (2026). *Neo4j Graph Database Management System*. https://neo4j.com/

Oettershagen, L. y Mutzel, P. (2022). Efficient top-k temporal closeness centrality computation. En Proceedings of The Web Conference (pp. 2836–2845). ACM.

Page, L., Brin, S., Motwani, R. y Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web (Technical Report 1999-66). Stanford InfoLab.

Robbins, A., Schroeder, W. y Foss, L. (2017). An ACL-based attack path analysis for Active Directory. BloodHound: Six Degrees of Domain Admin. SpecterOps.

Rodriguez, R. (2020). Mordor: Pre-recorded security events from simulated adversarial techniques. Open Threat Research. https://github.com/OTRF/mordor

The Rust Team. (2026). The Rust Programming Language (Versión X.X) [Software de computación]. https://www.rust-lang.org/

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. y Bengio, Y. (2018). Graph attention networks. En Proceedings of ICLR. OpenReview.

Wang, Q., Hassan, W. U., Li, D., Jee, K. y Yu, X. (2020). You are what you do: Hunting stealthy malware via data provenance analysis. En Proceedings of NDSS. Internet Society.

Wang, Q., Hassan, W. U., Li, D., Jee, K. y Yu, X. (2022). ThreaTrace: Detecting and tracing host-based threats in node level through provenance graph learning. IEEE Transactions on Dependable and Secure Computing, 19(6), 3792–3808.

Zengy, J., Wang, X., Liu, J., Chen, Y., Liang, Z., Chua, T.-S. y Cai, Z. (2023). MAGIC: Detecting advanced persistent threats via masked graph representation learning. En Proceedings of USENIX Security Symposium. USENIX.