

A Graph-Based Approach for Mapping Kernel-Level Telemetry to MITRE ATT&CK

Matteo Lupinacci
matteo.lupinacci@unical.it
University of Calabria
Rende, ITA

Luigi Arena
luigi.arena@unical.it
University of Calabria
Rende, ITA

Francesco Blefari
francesco.blefari@unical.it
University of Calabria
Rende, ITA

Angelo Furfaro
angelo.furfaro@unical.it
University of Calabria
Rende, ITA

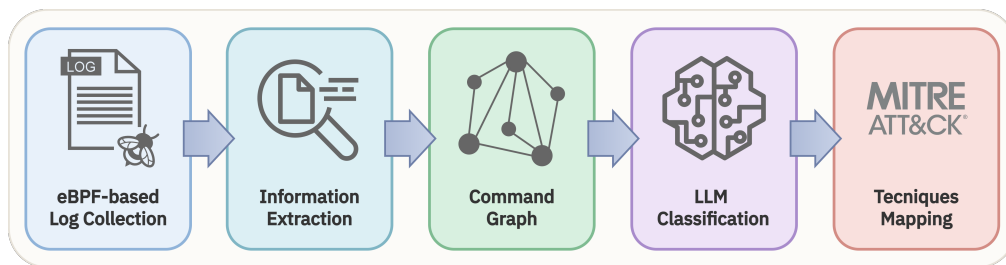


Figure 1: Overview of the proposed methodology for mapping eBPF-based kernel telemetry to MITRE ATT&CK.

Abstract

Mapping observed system behavior to standardized frameworks like MITRE ATT&CK is essential for threat-informed defense, but remains largely manual. Existing automated methods depend on Cyber Threat Intelligence reports, which offer only retrospective accounts of attacks. Low-level telemetry, i.e. kernel-level system calls, instead provides evidence of adversary behavior, yet its volume and complexity have limited its use for automated mapping. We present a methodology that collects kernel-level events via eBPF, correlates attacker commands into a provenance graph, and derives compact graph representations suitable for LLM-based reasoning. These representations are mapped to the MITRE ATT&CK framework using both pure LLM prompting and retrieval-augmented generation (RAG) grounded in the ATT&CK knowledge base, producing ranked technique candidates along with supporting rationales. We implement this methodology as an end-to-end pipeline, named Trace2ATT&CK and evaluate it on 347 Linux Atomic Red Team tests using locally deployed open-weights LLMs. RAG consistently improves ATT&CK mapping performance over pure prompting, while provenance graph substantially outperforms raw telemetry.

These results show that local inference over graph-based behavioral descriptions can make automated ATT&CK mapping from kernel-level telemetry operationally viable, without compromising data confidentiality.

CCS Concepts

• **Security and privacy** → **Vulnerability scanners**; • **Computing methodologies** → *Artificial intelligence*; • **Applied computing** → *Evidence collection, storage and analysis*.

Keywords

Low-level Telemetry, Attack Reconstruction, Provenance-graph, Threat Intelligence, Large Language Models

ACM Reference Format:

Matteo Lupinacci, Luigi Arena, Francesco Blefari, and Angelo Furfaro. 2026. A Graph-Based Approach for Mapping Kernel-Level Telemetry to MITRE ATT&CK. In . ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 Introduction

Modern cyber threats have become increasingly sophisticated, evasive, and rapid in adapting to existing defenses. Attackers employ ever-more-complex tactics, techniques, and procedures (TTPs), making it difficult for security teams to detect and respond to incidents in a timely manner. In this context, the MITRE ATT&CK framework [1] has emerged as a standard for categorizing and understanding adversary behavior, providing a structured knowledge base of tactics and techniques observed in real-world attacks. Applying ATT&CK effectively requires one fundamental step: mapping observed events to the appropriate TTPs. Identifying the specific

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, Washington, DC, USA

© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

techniques used by attackers enables a more consistent approach to threat intelligence, incident alerts, and streamlining incident response. However, due to the vast volume of security events that are typically observed in real systems, this mapping is a labor-intensive and error-prone process that is difficult to scale.

The current landscape of ATT&CK automated mapping has focused primarily on Cyber Threat Intelligence (CTI) reports. Researchers and practitioners have developed methods to extract tactics and techniques from unstructured CTI documentation using natural language processing and machine learning techniques [2, 3, 4]. These approaches have advanced the field by automating the analysis of threat reports, but share a critical limitation: they operate on second-hand information (i.e., the reports). Meanwhile, the actual events occurring on systems (i.e., application logs, kernel-level syscalls, file modifications, etc.) contain evidence of what attackers do. Although these low-level events are where the reality of an attack unfolds, they remain largely untapped for automatic ATT&CK mapping.

To the best of our knowledge, applying MITRE ATT&CK to raw or semi-structured host-level telemetry remains largely manual. Bridging this gap is far from straightforward. Systems' logs are massive in volume and complex in structure. A single system generates thousands of events per second; thus, extracting meaningful knowledge from these data is inherently difficult, as it involves identifying which events are relevant to an attack, understanding how individual operations relate to each other, and reconstructing the attacker's sequence of actions from low-level primitives.

To address these challenges, we present a methodology for the automatic mapping of observed system events to MITRE ATT&CK tactics and techniques that: (1) collects raw logs using eBPF program that captures system calls and kernel events with minimal overhead; (2) reconstructs the attacker's sequence of operations using a graph-based correlation approach that links actions to their consequences on the system, thus creating a comprehensive provenance graph; (3) performs automatic mapping to ATT&CK TTPs leveraging Large Language Models (LLMs)-based systems from the resulting provenance graph.

This methodology offers a different perspective from CTI-based approaches: instead of analyzing what others have reported about attackers, we observe and classify attacker behavior directly from the system itself. The proposed approach is validated using the Linux Atomic Red Team tests [5].

We structured our investigation around the following research questions:

- **RQ1:** Can MITRE ATT&CK techniques be automatically inferred directly from kernel-level system telemetry using LLM-based reasoning over reconstructed provenance graphs?
- **RQ2:** Does the choice of graph representation affect the quality of downstream ATT&CK classification?

The main contributions of this paper are as follows:

- We present an end-to-end methodology for mapping kernel-level host telemetry to MITRE ATT&CK tactics and techniques, grounding the mapping process directly in observed system behavior rather than relying on post-hoc CTI reports or pre-annotated security rules.

- We propose Trace2ATT&CK, an end-to-end pipeline that implements the proposed methodology. It combines eBPF-based telemetry collection, attacker-session anchoring, provenance-aware attack reconstruction, command-event correlation, and the extraction of raw and compressed weighted command-graph representations for downstream LLM-based reasoning.
- We release a [dataset](#) of the kernel-level logs collected while executing Atomic Red Team scenarios to foster future research in this area.

The remainder of this paper is organized as follows. Section 2 reviews the background and prior approaches to MITRE ATT&CK mapping. Section 3 formalizes the problem, states the threat model, and introduces a motivating example that highlights the practical need for real-time ATT&CK mapping. Section 4 presents the overall methodology, while Section 5 describes the evaluation dataset. In Section 6 we present Trace2ATT&CK, an end-to-end pipeline implementing the proposed methodology. Section 7 reports the experimental results and the ablation study. Finally, Section 8 concludes the paper.

2 Background and Related Work

2.1 Low-level Telemetry

Telemetry, in the context of computer security, refers to the automated and continuous collection of operational data from systems under observation, enabling real-time monitoring, anomaly detection, and post-incident forensic analysis [6]. Low-level telemetry is the fine-grained variant of this concept, capturing source-adjacent signals such as packet-level, process-level, kernel-level, and device-level events, rather than only aggregated summaries. When telemetry is captured at the kernel-level, through system-call tracing, audit logs, or provenance recording, it provides a privileged view of system activity that is fundamentally harder for adversaries to evade or manipulate than user-space logs [7, 8]. Kernel-level telemetry captures events such as process executions, file operations, network connections, and inter-process communications. In cybersecurity, this granularity is crucial because it provides the observability needed to detect suspicious behavior early, correlate events across heterogeneous assets, and support forensic reconstruction after compromise [9, 10]. Given this demonstrated effectiveness in reconstructing sophisticated attack chains, kernel-level telemetry constitutes a reliable foundation for security analytics.

2.2 MITRE ATT&CK Mapping Approaches

The automatic identification of MITRE ATT&CK techniques from security data is an area of growing interest for both research and industrial applications. This is due to the fact that it supports activities such as threat hunting, incident response, and the assessment of detection capability coverage.

Various approaches have been developed to address this issue, with the utilization of CTI reports serving as the primary information source. In particular, LLMs have recently been employed for the automatic extraction of entities, relationships, and concepts from unstructured information, facilitating the construction of *knowledge graphs* and the organization of knowledge [11, 12]. These models extend beyond traditional natural language processing, supporting

the representation and interpretation of semantic knowledge in domains characterized by heterogeneous, noisy, and semantically ambiguous information, such as CTI.

Tools and frameworks have been developed to assess the efficacy of security measures against the MITRE ATT&CK framework. The DeTT&CT framework [13] facilitates the correlation of data sources and detection capabilities with ATT&CK techniques, while enterprise SIEM (Security Information and Event Management) platforms, including Splunk Enterprise Security [14] and Elastic Security [15], link observed security events to ATT&CK techniques through correlation mechanisms and predefined detection rules. While these solutions are employed in operational contexts to facilitate threat detection and incident response, their effectiveness depends on the availability of relevant information sources and on the coverage provided by predefined detection rules used to associate security events with ATT&CK techniques [16]. Consequently, these solutions impose limitations on the identification of emerging techniques or attack variants that have not yet been modeled.

2.3 Related Work

This work lies at the intersection of provenance-based attack investigation, automated ATT&CK mapping, and LLM-assisted security analysis. Existing studies address these directions, however, fewer works consider ATT&CK mapping from session-level runtime evidence.

MGDA [17] learns structural and contextual representations to identify anomalous provenance entities, then applies rule-based matching to reconstruct TTP-oriented scenarios. CTINexus [3] leverages in-context learning to extract entity-relation triplets from CTI reports, resolve semantically equivalent entities, and infer missing relations. To mitigate the scarcity of labeled data for CTI-to-ATT&CK classification, SynthCTI [18] generates semantically guided synthetic samples that improve the coverage of underrepresented techniques. Likewise, LLM-based threat-hunting frameworks integrate semantic reasoning, threat intelligence, and graph-based correlation to produce investigation outputs aligned with the ATT&CK framework [19]. PROVCON [20] bridges CTI and system-level evidence by extracting attack primitives from reports, reproducing campaigns in a cyber range, and deriving provenance graphs from the resulting telemetry. It therefore relies on *a priori* campaign knowledge to generate representative evidence. Our work instead starts from observed runtime activity and infers ATT&CK candidates without requiring a campaign-specific CTI description to drive execution.

Recent work has also applied LLMs to command-level ATT&CK analysis. LADE [21] analyzes chronologically ordered command and script snippets to detect APT activity, localize malicious snippets, and rank corresponding ATT&CK techniques. It combines rubric-based prompting, ATT&CK knowledge, and summary propagation to preserve context across long sequences. Unlike LADE, which jointly performs detection, localization, and mapping over logged code snippets, Trace2ATT&CK assumes a pre-delimited attack interval and derives its mapping input from a provenance-grounded reconstruction of shell and descendant-process execution. This provides explicit causal relations from kernel-level telemetry rather than relying only on chronological command context.

Table 1: Comparison of Trace2ATT&CK with representative approaches.

Approach	Runtime telemetry	Provenance context	Session-level analysis	LLM reasoning	Ranked techniques
MGDA [17]	✓	✓	✗	✗	✗
CTINexus [3]	✗	✗	✗	✓	✗
SynthCTI [18]	✗	✗	✗	✓	✗
LADE [21]	✓	✗	✓	✓	✓
RAM [2]	✗	✗	✗	✓	✓
Trace2ATT&CK	✓	✓	✓	✓	✓

Runtime telemetry indicates that the approach operates on execution traces rather than CTI reports or analyst-authored rules. **Provenance context** denotes explicit causal dependencies among system entities. **Session-level analysis** refers to reasoning over a bounded attacker execution session. **Ranked techniques** indicates that the approach returns multiple ATT&CK techniques ordered by confidence.

Related approaches also map detection artifacts to ATT&CK. RAM [2] uses an LLM pipeline that combines SIEM-rule descriptions, ATT&CK retrieval, candidate generation, and relevance filtering. SIEM and detection-coverage platforms likewise associate telemetry with ATT&CK through curated rules and correlation logic. However, such rules encode analyst-defined detection hypotheses, whereas runtime events acquire meaning only through their command, process, and temporal context. Consequently, mapping provenance-derived execution flows differs from mapping rule text or detection metadata.

3 Problem Statement

Given a structured log of host-level events related to a pre-selected window of malicious activity, this paper addresses how to automatically enrich event sequences with MITRE ATT&CK techniques and transparent rationales. By correlating these events, we synthesize a unified, human-readable narrative that maps tactics, techniques, and context across the entire attack lifecycle. Three design goals guide the system:

- **Context sensitivity.** The system should recognize that many meaningful MITRE ATT&CK inferences depend on event sequences, causality, and process relationships rather than isolated activities.
- **Local execution.** The full pipeline should run with local models, local embeddings, and a local vector store, supporting on-premise and privacy-sensitive use cases.
- **Interpretability and Explainability.** The system should not only predict techniques but also explain why the evidence supports those techniques.

We focus on a predefined temporal window in which the attack is known to have occurred, so the logs already contain the relevant event sequences. Accordingly, our objective is not anomaly detection, but contextual explainability, enrichment, and reconstruction of attack behavior within that known attack interval. While anomaly detection may be used as a preceding step to identify such a window, it is outside the scope of this work. In practice, any anomaly detector or incident triage mechanism could serve this purpose, but our method assumes that the attack interval has already been delimited and therefore concentrates on fine-grained semantic enrichment.

Motivating Example. Consider the SolarWinds supply-chain compromise [22], attributed to APT29 and disclosed in December 2020 after attackers had remained inside victim networks for roughly 14 months. The campaign chained over 70 MITRE ATT&CK techniques. Crucially, the ATT&CK mapping was produced only months after the initial compromise through manual correlation of reports from FireEye, Microsoft, and CrowdStrike; CISA released its technique table only after public disclosure. In the meantime, victim organizations already had raw telemetry in their logs, such as DNS beaconing, suspicious rundll32.exe invocations, and SAML token anomalies, but lacked the contextual mapping needed to elevate these signals to technique-level alerts.

This case highlights a key limitation of CTI-driven mapping pipelines: classification latency. Mandiant and IBM X-Force reports [23] show that the median dwell time for advanced threats can range from 100 to over 200 days. Our approach addresses this gap by shifting from post-hoc intelligence to direct behavior classification from host-level telemetry. Beyond reducing latency, this shift could support fast structured alerting, the generation of detection rules, and adversary emulation. Moreover, by attaching a rationale to the proposed mapping, the output can be consumed by SOAR platforms, translated into Sigma rules, and leveraged in purple teaming exercises.

3.1 Threat Model

Similar to prior researches [24, 25], our work considers attackers having access to the target system (locally or remotely) and attempting to maintain a persistent presence by exploiting software vulnerabilities and chaining privilege escalation operations. Our trusted computing base includes the underlying system kernel and the audit framework, along with the offline analysis pipeline code, which is also standard among existing works.

Any kernel-level attacks that deliberately compromise security auditing systems are beyond the scope of this work. As such, we assume the use of existing system hardening techniques to mitigate any potential audit framework compromise. Moreover, we do not consider hardware-level or side-channel attacks, since their behavior is typically not explicitly captured by kernel-level audit systems.

4 Methodology

4.1 Data Collection

We leverage the extended Berkeley Packet Filter (eBPF) [26] to implement kernel-level telemetry. eBPF is a technology that enables the safe execution of sandboxed programs directly within the Linux kernel without requiring kernel modifications or module loading [27]. It has become a critical component in modern Linux systems and is widely adopted by cloud providers to enhance container security, network management, and system observability [28]. Although eBPF was initially developed for Linux systems, the advantages it has introduced have led to a current project that aims to introduce eBPF even for Windows systems [29].

Compared to traditional kernel audit frameworks (e.g., `auditd`), eBPF offers three key advantages for security telemetry. First, programs are statically verified before execution, ensuring they cannot crash or compromise the kernel, making eBPF a safer alternative to

loadable kernel modules [27, 30]. Second, eBPF programs can be attached to a wide range of kernel hook points – including system call tracepoints, kprobes, and perf events – at runtime without recompilation or reboot, providing unmatched flexibility in selecting which events to monitor [28]. Third, because eBPF programs execute in kernel space with Just-In-Time compilation to native instructions, they can filter and aggregate telemetry data before it reaches user space, significantly reducing the performance overhead associated with high-volume event collection [27]. These properties make eBPF particularly well-suited as a low-level telemetry mechanism for cybersecurity, combining the depth of kernel-level observation with the safety, flexibility, and efficiency [28, 30].

4.2 Provenance Graph

While raw log entries may not portray the intrinsic characteristics of a complex cyber attack, Provenance Graphs (PG) preserve the causal relationships between system entities, enabling analysts to trace attack entry points, identify multi-step attack paths, and reconstruct attack scenarios with high-level semantics [31, 10]. Provenance-based representation offers a holistic, attack-vector-agnostic view of system execution, making it ideal for intrusion detection because it captures interactions between processes, files, and network sockets as a unified directed graph rather than isolated events [31]. This graph-based approach has become *the de facto standard* for detecting and investigating Advanced Persistent Threats (APTs) [32]. Graph-based correlation is particularly effective for attack reconstruction because it exposes multi-level hidden relationships – causal, contextual, and indirect – among system behaviors, allowing detection systems to distill compact summary graphs that accurately describe malicious activity from large streams of audit logs [10]. For these reasons, we use PG as a formal model for the attack reconstruction phase.

In this phase, the collected kernel-level logs are uniformly transformed into a PG, where entities represent dynamic or persistent objects in the system, such as processes, files, sockets. They can perform actions on another entity or on the entity itself, such as process creation, file read/write, or network communication. Based on these interactions, our methodology constructs a directed PG, in which nodes represent system entities and edges represent their interactions.

Let us consider the following sets of system entities:

- $\mathcal{P}$ be the set of *processes*, where each process is an execution context identified by a unique entity identifier and characterized by attributes such as a process name and a user identifier.
- $\mathcal{C}$ be the set of *commands*, where each command represents an operation correlated with a process execution.
- $\mathcal{F}$ be the set of *files*, where each file is a file-system object identified by a path.

Furthermore, let $T = \{s, x, r, w, d, c, a\}$ be the set of event types, representing the following actions: spawn (*s*), execute (*x*), read (*r*), write (*w*), delete (*d*), connect (*c*), and accept (*a*). For each event type $t \in T$, we define a causal relationship R_t as follows:

- $R_s \subseteq \mathcal{P} \times \mathcal{P}$: a process creates another process (establishing a parent-child relationship).

- $R_c, R_a \subseteq \mathcal{P} \times \mathcal{P}$: a process establishes or accepts a connection with another process to exchange data.
- $R_x \subseteq \mathcal{P} \times \mathcal{C}$: a process executes a command.
- $R_r, R_w, R_d \subseteq \mathcal{P} \times \mathcal{F}$: a process accesses a file to read, modify, or delete it, respectively.

Definition 4.1 (Provenance Graph). A provenance graph is defined as a directed graph $G = (V, E)$ where $V \subseteq \mathcal{P} \cup \mathcal{C} \cup \mathcal{F}$ is the set of nodes, and

$$E = \bigcup_{t \in T} R_t \cap (V \times V)$$

is the set of edges.

By definition, each edge corresponds to a specific causal relationship between two system entities.

Definition 4.2 (Process-Centric Community). A *process-centric community* rooted at a master process $p \in \mathcal{P}$ is the induced subgraph $G_p = (V_p, E_p)$, where V_p contains p and all entities reachable from p via causal edges in E , and E_p contains all edges among entities in $\mathcal{P}$.

We introduce the notion of a *process-centric community* as the analytical unit for attack reconstruction, according to previous research [24]. This concept captures the observation that system activity during an attack is not uniformly distributed across the provenance graph, but rather concentrates around a small number of *master processes* that drive the malicious execution. A process-centric community is rooted at a master process and encompasses all entities causally reachable from it – child processes, their commands, and the associated events – forming a self-contained subgraph that represents a coherent unit of adversarial behavior.

4.3 Mapping Approach

Once the PG has been reconstructed from kernel-level eBPF telemetry, the final step of our methodology-consists of mapping the observed attack flow to MITRE ATT&CK techniques and sub-techniques. More in detail, the reconstructed PG is serialized into a structured textual representation and fed to LLM systems that classify the activity.

We adopt LLM-based approaches because these models have demonstrated state-of-the-art performance in processing and reasoning over structured and semi-structured textual data, including security-related graph representations [33]. We employ two complementary LLM paradigms: pure LLM prompting and RAG to evaluate which configuration best leverages the ATT&CK knowledge base for accurate classification. RAG operates by dynamically retrieving relevant ATT&CK technique descriptions and providing them as factual context to the LLM, which has been shown to reduce hallucination without requiring fine-tuning.

For each input graph, the system outputs a ranked list of top- k candidate classifications, where each candidate includes the ATT&CK technique identifier, the associated tactic, a confidence score in the range $[0, 1]$, and a natural-language explanation justifying the mapping. The inclusion of explanations serves a dual purpose: it increases the explainability of the classification and enables human analysts to validate the LLM's reasoning, which is critical for trustworthy deployment in security operations.

The decision to output a ranked list of top- k candidate techniques, rather than a single best-match prediction, is motivated by the inherent ambiguity of ATT&CK technique attribution. As previously observed in [21], distinct techniques can exhibit highly similar behavioral characteristics at the system level; for instance, T1105 (Ingress Tool Transfer) and T1048 (Exfiltration Over Alternative Protocol) frequently rely on comparable network utilities, making a single-prediction approach prone to false negatives when the model's confidence is distributed among semantically related candidates. A ranked top- k output mitigates this risk by preserving the most plausible alternatives, enabling downstream evaluation metrics and providing security analysts with a broader set of hypotheses to validate.

5 Dataset

For evaluation, we rely on Atomic Red Team [5], an open-source library of adversary emulation tests directly mapped to the MITRE ATT&CK framework [1]. The dataset is organized as a structured repository where each test – referred to as an "atomic test" – resides in a dedicated directory named after its corresponding ATT&CK technique identifier (e.g., T1055 for Process Injection). Each technique directory contains a canonical YAML definition file specifying the test metadata, supported platforms (Linux, macOS, Windows), execution commands, optional prerequisites, input arguments, and cleanup commands. To execute the atomic tests in a controlled and reproducible manner, they provide Invoke-AtomicRedTeam [34], a PowerShell-based execution framework that automates the invocation of tests defined in the atomics folder. Invoke-AtomicRedTeam allows selecting specific tests by technique ID and test number (e.g., Invoke-AtomicTest T1055 -TestNumbers 1,2), running all tests for a given technique, or executing the entire repository via Invoke-AtomicTest All. The framework also handles prerequisite downloads (-GetPrereqs), post-test cleanup (-Cleanup), and logging, ensuring that each execution produces a clean, isolated, and reproducible telemetry trace. Each atomic test corresponds to a specific ATT&CK technique, providing a pre-defined labeling scheme that eliminates the need for manual annotation and ensures that the ground truth is aligned with a taxonomy widely recognized in both academia and industry [35].

Leveraging the Invoke-AtomicRedTeam module, the command used to execute a test may explicitly embed the ground-truth ATT&CK identifier. If left unchanged, this would make the evaluation unfair, as the LLM could recover the correct label directly from a single command rather than from the reconstructed behavior. Therefore, we sanitize command lines before inference by obfuscating all ATT&CK technique and sub-technique identifiers with placeholder tokens (e.g., T1059 $\rightarrow$ TXXXX).

The resulting dataset, obtained by executing all attacks in the evaluation set and collecting logs via Tracee as the eBPF program, consists of 21,950,327 lines of raw logs, amounting to 29,8 GB.

6 Trace2ATT&CK Design and Implementation

This section details the architecture and implementation of our end-to-end pipeline for automatic mapping of kernel-level telemetry to MITRE ATT&CK techniques, named Trace2ATT&CK. It was developed according to the presented methodology.

We assume a threat scenario in which the attacker has remote access to the target system, for instance through an interactive SSH session, and issues commands from a remote shell. These commands may correspond either to the actual attack sequence or to a higher-level script, wrapper, or utility that internally triggers the malicious actions. The pipeline processes a single attack scenario through four stages: (1) kernel-level telemetry collection via eBPF (2) PG construction in a graph database build with neo4j (3) command-sequence extraction and graph serialization using custom Cypher query (4) LLM-based ATT&CK classification using either pure LLM prompting or RAG.

6.1 Telemetry Collection and Preprocessing

Trace2ATT&CK relies on Tracee [36], an eBPF-based runtime monitoring tool, to collect kernel-level telemetry from the target system. Tracee captures system calls and kernel events with minimal performance overhead, writing them as structured JSONL records enriched with process metadata (entity identifiers, user IDs, timestamps, and argument arrays).

The raw Tracee output is processed by a Rust-based preprocessor that performs three operations: (1) *event filtering*, removing noise entries (e.g., date probes) and projecting each event onto a fixed schema of canonical fields (e.g., timestamp, userId, processId, parentProcessId, syscall, processName, args); (2) *chunking*, splitting the event stream into ordered files of 10,000 records each to bound I/O cost; (3) optional *TOON (Token-Oriented Object Notation) serialization* [37]. TOON is a format built for LLM-driven workflows, where verbosity equals cost. It aims to make structured data token-efficient, reducing the cost of processing data within language models. The preprocessor is distributed as a cross-platform binary (Linux x86_64/ARM64, macOS x86_64/ARM64) and requires no kernel module loading.

It should be noted that, although we rely on Tracee for kernel-level log collection, this methodological step can be performed using any eBPF program. This is achieved by extracting the set of features described above from the raw log.

After our preprocessing, the final lines of logs analyzed are 21.823.051 amounting to 9,7 GB. On average, each attack scenario in our dataset is described by more than 62K lines of preprocessed kernel-level telemetry.

6.2 Provenance Graph Construction

Trace2ATT&CK utilizes a graph database (specifically, Neo4j) as its primary data structure to store a structured representation of the attack's provenance. The module responsible for database creation orchestrates the ingestion pipeline in four phases.

1. Attacker shell identification. The module scans the preprocessed logs in chronological order to locate the `execve` of a recognized shell (e.g., `bash`, `sh`) that is not a probe artifact. Following the assumptions given in Section 3, within the predefined temporal window, we are sure to find the attacker shell and deterministically determine the community boundary by traversing the process lineage tree rooted at the identified shell.

This step operationalizes the concept of process-centric community introduced in Section 4.2: the identified shell serves as the master process p , then the subsequent phases of the pipeline –

command–event correlation and graph ingestion – build the induced subgraph G_p by collecting all entities causally reachable from p .

2. Command–event correlation. A key step is to correlate the command issued by the attacker with the consequences it produces in the system. For the attack reconstruction, this is important because a single command generates several events, all of which have an impact on the system. To retrieve the commands issued by the attacker, we rely on a command-collection mechanism based on shell logging. This information can be obtained through shell history files such as `bash_history` for Bash and `zsh_history` for Zsh, through accounting tools such as `psacct` on Red Hat-based systems and `acct` on Debian-based systems, or through a custom trap-based logger.

For each directly issued command, the module identifies the corresponding `execve` event by matching the command timestamp to the nearest `execve` occurrence within a bounded lookahead window and its arguments. To preserve command ordering and prevent many-to-one matches, the search position is advanced monotonically after each successful association. If shell logging is unavailable or fails to capture a command, the module falls back to directly reconstructing the command line from any remaining, unmatched `execve` events, ensuring that no observed process execution is silently dropped from the correlation even in the absence of shell-level logging.

Then, Trace2ATT&CK reconstructs the full descendant process tree through a breadth-first traversal (BFS) over parent-child `execve` relations, and attaches to each discovered subprocess both its reconstructed command line and its complete local event stream. This yields a causally structured execution slice that captures not only the directly issued command, but also the derived activity it triggers on the host.

3. Graph Reduction. A key design decision concerns the granularity of the provenance graph. We apply a Provenance Graph Reduction strategy tailored for the task of MITRE ATT&CK mapping by restricting the node set to Process and Command entities, deliberately excluding the database ingestion of the file-level and network-level nodes.

This choice is motivated by the nature of the mapping task itself: ATT&CK techniques are defined at the level of adversary *procedures* (i.e., the concrete actions performed on the system) and these actions are primarily captured by process executions and their associated command-line arguments. File paths, network endpoints, and other system objects are then embedded within command arguments and can therefore be inferred contextually by the LLM without explicit graph nodes.

This selective ingestion optimizes query performance and maintains a tractable input size for LLM reasoning while preserving the complete provenance information at the construction stage for potential use in other analysis tasks. Moreover, restricting the node set directly addresses the well-documented *dependency explosion* problem in provenance analysis: large, fully expanded provenance graphs are impractical to process within the finite context window of large language models, as injecting comprehensive event histories incurs prohibitive token overhead [38]. Other research in the field further supports this design: PROV-LLM [39] demonstrates

that decomposing large provenance graphs into process-centered subgraphs – enriched with process names, execution paths, and command-line arguments – yields superior detection accuracy and precision compared to full-graph approaches. More evidences about this choice are illustrated in Section 7.2.

4. Graph ingestion. The graph ingestion module builds Process Node identified by `entityID`, `processName`, `userId`, and Command Node identified by `cmdID`, `timestamp`, and `cmdline`. The relations between nodes are defined as follows:

- SPAWNED: parent-to-child process relationships;
- EXECUTES: direct command execution by a process;
- INDIRECT_EXECUTES: sub-process command execution;
- TRIGGERS: parent-command-to-child-command causal links.

All operations use batched Cypher queries (default batch size 1,000) with MERGE for idempotent upserts and uniqueness constraints on `entityId` and `commandId`.

6.3 Command Graph Extraction

Another key design choice of Trace2ATT&CK is the extraction of *weighted command graphs* in which repeated executions of the same command pattern are aggregated and represented through a weight proportional to their execution frequency. Weight highlights a useful property for ATT&CK mapping: it is not just which commands appear that matters, but also how often they are repeated. Frequency helps distinguish isolated actions from stronger procedural patterns. Trace2ATT&CK derives two weighted representations.

Full weighted command graph. The first representation preserves the original command instances observed during the attack session. Each EXECUTES edge yields a node containing the executing process (entity ID, name, user ID), the command (full command line, timestamp), its privilege-escalation status, and the list of triggered sub-commands retrieved through TRIGGERS edges. In addition, each command node is annotated with the number of times the same command line has been executed in the session, while repeated triggered sub-commands are aggregated and labeled with their corresponding occurrence count.

Compressed weighted command graph. The second representation reduces redundancy by normalizing commands through (i.e., extracting the base binary and option flags), and merging identical command+option patterns into a single normalized view. As in the raw version, the resulting graph is weighted. This compressed weighted view is especially useful for attacks characterized by repetitive behavior, such as iterative reconnaissance or repeated payload invocation.

Both representations are serialized as DOT graph files (via pydot). When the Full DOT graph exceeds 15,000 characters, the system automatically falls back to the compressed representation. A representative example of the extracted command graph is shown in Figure 2.

6.4 LLM Prompting

Trace2ATT&CK supports both pure LLM prompting and RAG as classification paradigms, allowing empirical comparison of their effectiveness for ATT&CK mapping from provenance graphs.

Prompt design. The system prompt instructs the LLM to act as a cyber threat intelligence analyst and to output a JSON array of the top-*K* candidate techniques ranked by confidence (see appendix C). Each candidate includes the tactic name, the technique with sub-technique (e.g., T1003.007 if any), a confidence score in (0, 1), and a natural-language rationale for the proposed mapping. The prompt enforces two critical constraints: (1) sub-techniques must be used whenever the attack evidence warrants them; (2) rationales must reference the *entire command flow*, not isolated commands;

Retrieval-augmented generation. The RAG pipeline ingests the MITRE ATT&CK Enterprise knowledge base [40] into a local Chroma vector store. Technique descriptions are split into 800-word chunks and indexed using the `mx-bai-embed-large-v1` [41, 42] embedding model, widely used in AI for semantic search, clustering, classification, and RAG. At inference time, the DOT graph is embedded, and the top-5 most relevant technique chunks are retrieved via Maximal Marginal Relevance (MMR) search ($\lambda_{\text{MMR}} = 0.5$, `fetch_k` = 20). These chunks are prepended as factual context to the user prompt, grounding the LLM's reasoning in the official ATT&CK taxonomy without requiring the full knowledge base to fit in the context window.

6.5 Experimental Setup

The experimental evaluation has been conducted using an infrastructure composed of two virtual machines hosted on a Proxmox Virtual Environment (Proxmox VE) 9.1.7 server, running Linux kernel 6.17.13-2-pve on an x86_64 architecture. The physical server is equipped with an AMD EPYC™ Genoa 9554P processor (64 cores/128 threads, 3.1 GHz base frequency) and 192 GB of RAM.

The first virtual machine, hosting the entire pipeline execution environment, runs Ubuntu 24.04.4 LTS (Linux kernel 6.17.0-35-generic) and is configured with 16 vCPUs (1 socket, 16 cores), 16 GB of RAM, Docker 29.5.3, and LXC 5.21.5 LTS. The test environment is composed of four main components: (1) the *target* environment, representing the system under analysis (based on Ubuntu 24.04), which can be deployed either as a Docker container or as an LXC virtual machine. This environment is configured with the tools required to execute Atomic Red Team tests, including OpenSSH and PowerShell; (2) the *tracee* container, running *tracee* v0.24.1, which leverages eBPF technology to monitor the activities performed within the *target* environment, collecting logs and forwarding them in JSON format to *fluentd*; (3) the *fluentd* container, running *fluentd* v1.19.3, responsible for managing the logs generated by *tracee*, applying buffering techniques and log rotation mechanisms; (4) the *neo4j* container, running *neo4j* v2026.05.0, used for storing and querying the generated provenance graph.

The second virtual machine is dedicated to the local execution of the LLMs employed during the analysis phases. It runs Ubuntu 24.04.4 LTS (Linux kernel 6.17.0-35-generic) and is configured with 16 vCPUs (1 socket, 16 cores), 32 GB of RAM, LM Studio 0.4.16 (Build 2), NVIDIA driver 595.71.05, and CUDA 13.2. Model inference is performed entirely locally through LM Studio, leveraging two NVIDIA L40S GPUs with 48 GB of memory each.

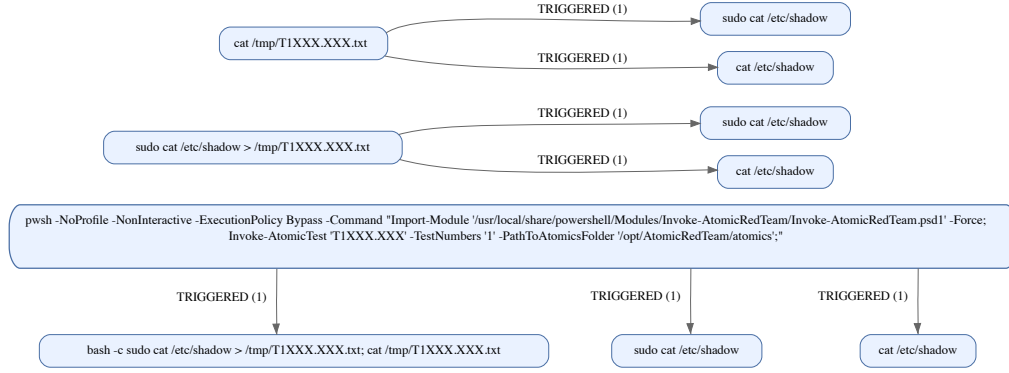


Figure 2: Weighted Command Graph Extracted for T1003.008 (OS Credential Dumping: /etc/passwd and /etc/shadow)

Table 2: LLMs evaluated in our experiments.

Model	Params	Quantization	Context Window
gpt-oss-20b [43]	20B	GGUF MXFP4	32K
gpt-oss-120b [43]	120B	GGUF MXFP4	32K
gemma-4-31b-it	31B	GGUF Q4_K_S	32K
llama-3.3-70b-instruct [44]	70B	GGUF Q4_K_S	32K
deepseek-r1-distill-qwen-32b	32B	GGUF Q8_0	32K
foundation-sec-8b-reasoning [45]	8B	GGUF Q8_0	32K
qwen3.5-9b [46]	9B	GGUF Q8_0	32K

7 Results and Ablation Study

We test Trace2ATT&CK using seven local LLMs, listed in Table 2, on the Linux-specific subset of test from the Atomic Red Team dataset. All metrics reported in this section are computed over the N valid mappings remaining after excluding cases in which the LLM output was not a well-formed JSON object, which we treat as unparseable predictions. More details of invalid output per model are shown in Table 5.

Moreover, since a subset of tests fails at execution time due to missing dependencies in the Atomic Red Team invoker repository or due to missing permissions of the user executing the attack, we report metrics separately for the test successfully concluded (i.e., the test returned status code is 0, $N = 166$) and for test failed ($N = 181$). Along with results on technique identification across the entire dataset, we report results separately for the subset of tests whose ground truth specifies a sub-technique ($N = 214$) by looking for the exact match (e.g., T1003.007). We report multiple ranking-based metrics [21] in order to capture both correctness and placement quality. The metrics are detailed in appendix A.

7.1 Quantitative Results

A key factor influencing ATT&CK mapping quality is the amount of discriminative behavioral evidence contained in the input sequence. At the technique level, longer multi-step command traces often provide a richer causal context, making it easier to distinguish among broad ATT&CK behaviors. By contrast, sub-technique classification is inherently more ambiguous, since sub-techniques capture finer-grained variations of a broader behavior and therefore require more specific contextual cues to disambiguate between command sequences that may be compatible with multiple related ATT&CK classes [47]. Recent work, directly from MITRE organization and

the Center for Threat-Informed Defense [48] also highlights that many ATT&CK techniques are ambiguous when considered in isolation and that surrounding context is necessary to infer the correct adversarial intent.

Label-identifier inconsistency. During evaluation, we observed a recurring failure mode across all tested models: the model correctly identifies a technique by name but associates it with an incorrect technique ID, or conversely, produces the correct ID paired with a mismatched technique name. This inconsistency arises because the model relies on its parametric knowledge to map the observed behavior to a technique name and its corresponding identifier, but this internal mapping is not always reliable. To mitigate this issue, we introduce a third classification paradigm, termed *taxonomy-grounded prompting*. In this configuration, the system prompt is augmented with the complete mapping of technique IDs to their canonical names, extracted directly from the MITRE ATT&CK Enterprise knowledge base [40]. The injected mapping acts as a grounding signal that anchors the model’s predictions to the official taxonomy.

Tables 3, 4, summarize the evaluation results across the three configurations considered in this study. For paired comparisons, Δ indicates the absolute improvement in HR@ k of taxonomy-grounded prompting over base LLM prompting. HR@ k measures the percentage of test cases where the correct label is within the top- k predictions returned by the LLM (Appendix A). We set $K = 5$.

LLM results. Under base LLM prompting, performance varies substantially across models. Notably, foundation-sec-8b-reasoning – an 8B model specialized on security-related pretraining data – outperforms substantially larger general-purpose models such as llama-3.3-70b-instruct (Tech HR@5 = 49, 2% vs. 35, 26%, a +13, 94 point gap despite an almost 9× parameter disadvantage) and deepseek-r1-distill-qwen-32b (49, 2% vs. 29, 97%), and is competitive with gpt-oss-20b. This suggests that domain-specialized pretraining can be an effective driver of ATT&CK mapping quality. gpt-oss-120b remains the strongest model overall (Tech HR@5 = 60, 58%, Sub-tech HR@5 = 33, 33%), while deepseek-r1-distill-qwen-32b is the weakest. Taxonomy-grounded prompting yields consistently higher scores across all models and settings, with again gpt-oss-120b outperforms other models. On the exact sub-technique match level, it reaches HR@5 = 43, 66%, the best result among all models. In the

technique setting, gpt-oss120b achieves the highest HR@5 (64, 64%), closely followed by gemma-4-31-b (63, 69%), while deepseek-r1-distill-qwen-32b again records the lowest scores.

RAG results. By injecting relevant ATT&CK knowledge at inference time, the model can compare the observed behavior against a narrower and better grounded set of candidate techniques, reducing semantic ambiguity and improving fine-grained sub-technique attribution. This effect is reflected in our results: on the exact sub-technique match level, gpt-oss-20b comes close to doubling its HR@5 relative to base prompting (16, 11% $\rightarrow$ 31.55%, +95, 85% relative gain). Overall, gpt-oss-120b reaches the highest HR@5 at sub-techniques level (46, 26%), while in the technique setting it achieves the strongest overall result across all configurations (HR@5 = 66, 57%) followed by gemma-4-31-b (HR@5 = 65, 42), while deepseek-r1-distill-qwen-32b remains the weakest performer.

Comparison. Comparing base LLM prompting directly against RAG, HR@5 improves consistently across all model-criteria combinations (14 out of 14), a stronger and more uniform trend than what is observed for taxonomy-grounded prompting alone. At the technique level, HR@5 gains range from +1, 78 (foundation-sec-8b-reasoning) to +10, 66 (gemma4-31b-it), with an average improvement of $\sim 5, 52$ points. At the exact sub-technique level, gains are larger and more variable, ranging from +0.93 (gemma4-31b-it) to +15, 44 (gpt-oss-20b), averaging $\sim 8, 03$ points, confirming that retrieval-grounded context is especially beneficial for disambiguating fine-grained behavioral variants. Notably, the largest technique-level gain and the largest sub-technique-level gain occur for different, mid-sized models, suggesting that RAG's benefit is not strictly tied to model scale but rather to how well a given model's parametric knowledge already covers the ATT&CK taxonomy.

Successful vs. Failed Attacks. An important practical question is whether Trace2ATT&CK remains useful when the underlying attack does not fully execute – a common scenario in real deployments, where adversarial actions are frequently interrupted by defenses, misconfigurations, or missing prerequisites before completion.

At the technique level, performance on failed attacks remains close to, and in some cases exceeds, performance on successful ones. For instance, under base prompting foundation-sec-8b reaches HR@5 = 49, 67% on successful attacks versus 48, 75% on failed ones. In general, results indicate that the coarse-grained adversarial intent underlying an ATT&CK technique is often already legible from the partial command flow generated before an attack is interrupted, without requiring the full attack chain to unfold.

At the exact sub-technique level, the picture changes: the gap between successful and failed attacks widens substantially and is consistently in favor of successful attacks, and reaching over 21 points for gemma4-31b-it and, under RAG, over 29 points for gpt-oss120b (HR@5 = 60, 36% on successful attacks vs. 31.07% on failed ones). This is consistent with the finer-grained nature of sub-technique classification discussed above: disambiguating between closely related behavioral variants requires more complete contextual evidence, which an interrupted execution, by definition, cannot fully provide.

Taken together, these results suggest that Trace2ATT&CK retains most of its value for technique-level triage and alerting, which is arguably the more actionable output for a SOC analyst reacting to an in-progress or blocked intrusion attempt, while its finer-grained sub-technique attribution capability is, as expected, more dependent on observing the complete adversarial procedure.

Discussion. Although our primary objective is to support effective local inference with smaller models, we also include experiments with the frontier model Claude-Sonnet-5 to validate the methodology itself. The underlying rationale is that strong performance with a highly capable model indicates that the proposed pipeline is sound; any performance degradation observed with smaller local models can therefore be attributed mainly to limited model capacity rather than to flaws in the method.

The results support this interpretation. Sonnet-5 achieves high performance across all settings, reaching an HR@5 of 90, 91% at the technique level and 74, 76% at the exact sub-technique level, compared with the best local-model results (gpt-oss-120b) it scores +24, 34% and +28, 5%. This confirms that the task is well handled by the proposed pipeline when sufficient model capability is available.

Moreover, adding RAG does not yield meaningful improvements for Sonnet-5, suggesting that the model already encodes enough relevant ATT&CK knowledge internally. This behavior contrasts with smaller models, where RAG produces clear gains, indicating that retrieval mainly compensates for weaker parametric knowledge. Overall, these results suggest that the observed performance gap is driven primarily by model strength while also validating the methodological soundness of the proposed approach.

All these results and considerations allow us to answer positively to the first RQ: MITRE ATT&CK techniques can be reliably inferred from kernel-level telemetry via LLM-based reasoning over reconstructed provenance graphs. The frontier-model validation further confirms the soundness of the overall pipeline, indicating that the observed gap for local models stems from model capacity rather than methodological limitations.

Comparison with LADE. Among prior work, LADE [21] is the closest to our approach, as it similarly applies LLM-based reasoning directly to command-level evidence rather than to CTI reports. However, LADE is evaluated on a substantially smaller and less realistic scale: only 35 attack sequences, each averaging approximately 990 code snippets with just 4.8 non-empty lines per snippet ($\sim 4, 7K$ lines of logs each). In contrast, our evaluation spans 347 Linux Atomic Red Team test cases, each derived from a full kernel-level telemetry trace averaging $\sim 62K$ lines of raw eBPF events per scenario – more than three orders of magnitude larger per scenario than LADE's snippet sequences. This makes our evaluation setting closer to a realistic operational deployment.

7.2 Ablation Study

We performed two ablations to quantify the contribution of the graph-construction stage in Trace2ATT&CK. The ablation study was conducted only on gpt-oss-120b under taxonomy-grounded prompt, the best-performing LLM identified in the main experiments.

Table 3: Mapping performance across LLMs: base vs. taxonomy-grounded prompt.

Model	Level	Base Prompt								Taxonomy-Grounded Prompt								Δ HR@5 (%)
		All HR@5	Successful			Failed			All HR@5	Successful			Failed					
		HR@5	HR@5	MRR@5	NDCG@5	HR@5	MRR@5	NDCG@5	HR@5	HR@5	MRR@5	NDCG@5	HR@5	MRR@5	NDCG@5			
foundation-sec-8b-reasoning	Tech	49.20	49.67	0.342	0.381	48.75	0.332	0.371	56.27	55.63	0.368	0.416	56.88	0.395	0.438	+7.07		
	Subtech	27.78	30.10	0.203	0.227	25.26	0.176	0.195	34.34	36.89	0.253	0.282	31.58	0.242	0.260	+6.56		
qwen3.5-9b	Tech	43.01	48.09	0.350	0.382	38.71	0.298	0.321	51.40	55.73	0.400	0.439	47.74	0.348	0.380	+8.39		
	Subtech	13.97	18.09	0.136	0.147	9.41	0.075	0.080	22.91	30.85	0.222	0.244	14.12	0.108	0.116	+8.94		
gpt-oss-20b	Tech	43.79	42.59	0.305	0.335	44.89	0.338	0.366	46.75	44.44	0.318	0.350	48.86	0.382	0.409	+2.96		
	Subtech	16.11	19.27	0.134	0.149	12.75	0.102	0.108	25.59	26.61	0.197	0.214	24.51	0.201	0.212	+9.48		
gemma4-31b-it	Tech	54.76	62.05	0.510	0.538	48.07	0.390	0.412	63.69	68.07	0.562	0.592	59.67	0.462	0.495	+8.93		
	Subtech	31.31	41.44	0.335	0.355	20.39	0.183	0.188	36.45	46.85	0.382	0.404	25.24	0.225	0.232	+5.14		
deepseek-r1-distill-qwen-32b	Tech	29.97	35.54	0.250	0.277	24.86	0.191	0.206	38.04	40.96	0.292	0.321	35.36	0.253	0.278	+8.07		
	Subtech	4.21	7.21	0.053	0.058	0.97	0.005	0.006	9.35	10.81	0.081	0.088	7.77	0.063	0.067	+5.14		
llama-3.3-70b-instruct	Tech	35.26	41.21	0.286	0.317	29.83	0.175	0.205	40.46	44.85	0.327	0.357	36.46	0.222	0.257	+5.20		
	Subtech	13.55	18.92	0.129	0.144	7.77	0.053	0.059	21.96	24.32	0.151	0.174	19.42	0.121	0.139	+8.41		
gpt-oss-120b	Tech	60.58	59.64	0.486	0.513	61.45	0.482	0.515	64.64	63.25	0.504	0.536	65.92	0.509	0.546	+4.06		
	Subtech	33.33	40.54	0.344	0.360	25.49	0.195	0.210	43.66	54.05	0.445	0.470	32.35	0.265	0.280	+10.33		

Table 4: Mapping performance across LLMs with RAG.

Model	Level	All Attacks HR@5 (%)	Successful Attacks			Failed Attacks		
		HR@5 (%)	HR@5 (%)	MRR@5	NDCG@5	HR@5 (%)	MRR@5	NDCG@5
foundation-sec-8b-reasoning	Tech	50.98	52.38	0.370	0.408	49.69	0.333	0.374
	Subtech	34.72	36.63	0.290	0.309	32.61	0.246	0.266
qwen3.5-9b	Tech	50.00	55.86	0.434	0.465	44.97	0.322	0.353
	Subtech	22.68	28.28	0.210	0.228	16.84	0.120	0.132
gpt-oss-20b	Tech	46.61	46.58	0.367	0.392	46.63	0.367	0.392
	Subtech	31.55	36.79	0.299	0.316	26.00	0.218	0.229
gemma4-31b-it	Tech	65.42	68.67	0.550	0.585	62.43	0.488	0.522
	Subtech	32.24	40.54	0.312	0.336	23.30	0.179	0.193
deepseek-r1-distill-qwen-32b	Tech	37.18	37.95	0.269	0.297	36.46	0.269	0.293
	Subtech	12.15	16.22	0.129	0.137	7.77	0.045	0.054
llama-3.3-70b-instruct	Tech	38.44	42.42	0.314	0.341	34.81	0.193	0.231
	Subtech	16.90	20.00	0.137	0.152	13.59	0.073	0.088
gpt-oss-120b	Tech	66.57	64.46	0.505	0.540	68.51	0.515	0.558
	Subtech	46.26	60.36	0.489	0.518	31.07	0.267	0.278

Raw logs vs. graph-based representation. We assess whether the graph construction step introduced in Section 4 is actually necessary, or whether an LLM can perform comparably well when given the raw, chronologically ordered kernel-level event stream. Directly prompting the LLM with raw logs led to a clear degradation in ranking quality with respect to the graph-based representation. Raw eBPF output is significantly more verbose per unit of information, since it repeats process/user metadata and low-level syscall arguments across many redundant events; this inflates the prompt and, for longer attack sequences, risks exceeding the model’s effective context window or forcing truncation, which directly harms recall of relevant behavior. Second, and more importantly, raw logs do not make causal relationships between events explicit – the LLM must itself infer which process spawned which command, and how a sequence of syscalls composes into a coherent adversarial procedure.

Raw logs achieve 19, 57% HR@5, 0.214 MRR@5, and 0.227 NDCG@5, with empty outputs in 53, 1% cases at the sub-technique level (exact

match), while at the technique level raw logs achieve 37, 88% HR@5, 0.298 MRR@5, and 0.385 NDCG@5, with empty outputs for token-limit exceeding, in 56, 73% of cases. This confirms that unstructured telemetry is difficult to map reliably to ATT&CK techniques and, more importantly, that the raw-log baseline can fail when the input approaches the context limit of the local inference settings.

Full Provenance Graph vs. Command Graph. We compared the full PG against the reduced Command Graph used in the main experiments under a fixed context budget of 32K tokens. The proposed full PG correlation achieves 46, 81% HR@5 at sub-techniques level with exact match, while at technique level 69, 27% HR@5 when the mapping occurs without reaching context-window limit. Although the full PG achieves higher raw HR@5 whenever it fits within the context window, this advantage is undermined by its practical unreliability: with nearly one in two inputs producing an empty output (up to 49, 76% at the technique level and 45, 93% at the sub-technique level), the full PG effectively fails to return *any* mapping for roughly half of the test cases. In an operational setting, a technique-level prediction that is only slightly less accurate but consistently available is far more valuable than a marginally better prediction that is missing half of the time. The compact command-graph representation, by contrast, never exceeds the context budget and therefore always yields a usable candidate list, making it the more practical choice for deployment under bounded local inference resources, even at the cost of a modest drop in per-case accuracy.

We can now answer RQ2: the choice of command-graph representation has a substantial effect on classification quality. Raw, unstructured telemetry performs markedly worse (HR@5 = 19, 57% sub-technique / 37, 88% technique) and frequently produces empty outputs due to context-window truncation. Full provenance graphs improve accuracy over raw logs but remain token-heavy, causing empty outputs in up to 49, 76% of cases; the compact, weighted command-graph representation used in our main experiments avoids this failure mode entirely (0, 0% empty outputs).

8 Conclusion

This paper presented a methodology and end-to-end pipeline for automatically mapping kernel-level telemetry to MITRE ATT&CK techniques and sub-techniques. By collecting system calls via eBPF, reconstructing attacker-associated process trees into a provenance graph, and deriving compact weighted command-graph representations, our approach bridges low-level host evidence and high-level threat-informed classification. We evaluated the methodology on Atomic Red Team scenarios across seven locally deployed LLMs under three classification paradigms: base prompting, taxonomy-grounded prompting, and RAG.

RAG consistently improves mapping accuracy over base prompting (up to +15,44 HR@5 points), especially for smaller models, while a security-specialized 8B model rivals much larger general-purpose ones, suggesting that domain pretraining can matter more than scale. The system also degrades gracefully on incomplete attacks, preserving technique-level accuracy even when execution is interrupted. Our ablation confirms that compact command graphs are key to reliable local inference: full provenance graphs are more accurate but return no output in nearly half of cases due to context limits. Finally, validation with Claude-Sonnet-5 (HR@5 = 90, 91%) confirms the pipeline's soundness, indicating that the local-model gap stems from capacity rather than methodology.

Acknowledgments

This work has been partially supported within the technology transfer activities of the Second-Level Master's Program in Electric Mobility and Circular Economy framework, funded by the Italian MUR under the "Territorial Pacts for Higher Education for Enterprises" (CUP H52C23000090001). The work of M. Lupinacci was supported by Agenzia per la cybersicurezza nazionale under the programme for promotion of XLI cycle PhD research in cybersecurity – (CUP H23C25000360005). The views expressed are those of the authors and do not represent the funding institution.

References

- [1] MITRE. 2026. Mitre att&ck. <https://attack.mitre.org/>. Accessed: 2026-07-02. (2026).
- [2] Prasanna N Wudali, Moshe Kravchik, Ehud Malul, Parth A Gandhi, Yuval Elovici, and Asaf Shabtai. 2025. Rule-att&ck mapper (ram): mapping siem rules to ttps using llms. *arXiv preprint arXiv:2502.02337*.
- [3] Yutong Cheng, Osama Bajaber, Saimon Amanuel Tsegai, Dawn Song, and Peng Gao. 2025. Ctinexus: automatic cyber threat intelligence knowledge graph construction using large language models. In *2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P)*, 923–938. doi:10.1109/EuroSP63326.2025.00057.
- [4] Álvaro Ruiz-Ródenas, Jaime Pujante Sáez, Daniel García-Algora, Mario Rodríguez Béjar, Jorge Blasco, and José Luis Hernández-Ramos. 2026. Syntheti: llm-driven synthetic cti generation to enhance mitre technique mapping. *Future Generation Computer Systems*, 177, 108232. doi:https://doi.org/10.1016/j.future.2025.108232.
- [5] Red Canary. 2024. Atomic red team. <https://atomicredteam.io>. Accessed: 2026-07-02. (2024).
- [6] Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji. 1998. Intrusion detection using sequences of system calls. *J. Comput. Secur.*, 6, 3, (Aug. 1998), 151–180.
- [7] Thomas Pasquier, Xueyuan Han, Thomas Moyer, Adam Bates, Olivier Hermant, David Evers, Jean Bacon, and Margo Seltzer. 2018. Runtime analysis of whole-system provenance. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Association for Computing Machinery, Toronto, Canada, 1601–1616. ISBN: 9781450356930. doi:10.1145/3243734.3243776.
- [8] Adam Bates, Dave Tian, Kevin R. B. Butler, and Thomas Moyer. 2015. Trustworthy whole-system provenance for the linux kernel. In *Proceedings of the 24th USENIX Conference on Security Symposium (SEC'15)*. USENIX Association, Washington, D.C., 319–334. ISBN: 9781931971232.
- [9] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. 2020. Unicorn: runtime provenance-based detector for advanced persistent threats. *arXiv preprint arXiv:2001.01525*.
- [10] Zijun Cheng, Qiujuan Lv, Jinyuan Liang, Yan Wang, Degang Sun, Thomas Pasquier, and Xueyuan Han. 2024. Kairos: practical intrusion detection and investigation using whole-system provenance. In *2024 IEEE Symposium on Security and Privacy (SP)*, 3533–3551. doi:10.1109/SP54263.2024.00005.
- [11] Yuelin Hu, Futai Zou, Jiajia Han, Xin Sun, and Yilei Wang. 2024. Llm-tikg: threat intelligence knowledge graph construction utilizing large language model. *Computers & Security*, 145, 103999. doi:https://doi.org/10.1016/j.cose.2024.103999.
- [12] Xiuzhang Yang et al. 2026. Cti-thinker: an llm-driven system for cti knowledge graph construction and attack reasoning. *Cybersecurity*, 9, (Jan. 2026). doi:10.1186/s42400-025-00505-y.
- [13] [n. d.] GitHub - rabobank-cdc/DeTTECT: Detect Tactics, Techniques & Combat Threats — github.com. <https://github.com/rabobank-cdc/DeTTECT>. [Accessed 10-07-2026]. ().
- [14] [n. d.] Splunk Enterprise Security | Splunk — splunk.com. https://www.splunk.com/en_us/products/enterprise-security.html. [Accessed 10-07-2026]. ().
- [15] [n. d.] Agentic security operations from Elastic Security — elastic.co. <https://www.elastic.co/security>. [Accessed 10-07-2026]. ().
- [16] CardinalOps. 2025. State of SIEM Detection Risk: 5th Annual Report. Tech. rep. CardinalOps. <https://cardinalops.com/wp-content/uploads/2025/06/25-CardinalOps-2025-State-of-SIEM-Report.pdf>.
- [17] Mengjiao Cui, Zhengwei Jiang, Shuai Li, Chunyan Ma, Kai Zhang, Peian Yang, and Huamin Feng. 2026. Mgda: a provenance graph-based framework for threat detection and attack scenario reconstruction. *Computer Networks*, 274, 111806. doi:https://doi.org/10.1016/j.comnet.2025.111806.
- [18] Álvaro Ruiz-Ródenas, Jaime Pujante Sáez, Daniel García-Algora, Mario Rodríguez Béjar, Jorge Blasco, and José Luis Hernández-Ramos. 2026. Syntheti: llm-driven synthetic cti generation to enhance mitre technique mapping. *Future Gener. Comput. Syst.*, 177, C, (June 2026), 13 pages. doi:10.1016/j.future.2025.108232.
- [19] Manikandan R, Royal Bhavitha, Shaik Banuheer, Sasmal Aniket Arun, and Dinesh K. 2026. Agentic llm-driven cyber threat hunting using mitre att&ck knowledge mapping and graph-based evidence correlation. In *2026 Third International Conference on Networking and Communications (ICNWC)*, 1–9. doi:10.1109/ICNWC68145.2026.11518378.
- [20] Anis Yusof, Shaofei Li, Arshdeep Singh Kawatra, Ding Li, Ee-Chien Chang, and Zhenkai Liang. 2025. From observations to insights: constructing effective cyberattack provenance with provcon. In *Proceedings of the Workshop on SOC Operations and Construction (WOSOC)*.
- [21] Joon-Young Gwak, Aubrey Strier, Zhaoan Xi, Guanhua Yan, Xiaokui Shu, Scott D Stoller, and Ping Yang. [n. d.] Lade: llm-assisted advanced persistent threat detection and explanation.
- [22] [n. d.] SolarWinds Compromise, Campaign C0024 | MITRE ATT&CK®. <https://attack.mitre.org/campaigns/C0024/>. [Accessed 07-07-2026]. ().
- [23] [n. d.] The Cost of Detection Delays: Lessons from Assumed Breach Simulations — packetlabs.net. <https://www.packetlabs.net/posts/cost-of-detection-delay/>. [Accessed 07-07-2026]. ().
- [24] Zhiqiang Xu, Pengcheng Fang, Changlin Liu, Xusheng Xiao, Yu Wen, and Dan Meng. 2022. Depcomm: graph summarization on system audit logs for attack investigation. In *2022 IEEE Symposium on Security and Privacy (SP)*, 540–557. doi:10.1109/SP46214.2022.9833632.
- [25] Ahmed Aly, Essam Mansour, and Amr Youssef. 2025. Ocr-apt: reconstructing apt stories from audit logs using subgraph anomaly detection and llms. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25)*. Association for Computing Machinery, Taipei, Taiwan, 261–275. ISBN: 9798400715259. doi:10.1145/3719027.3765219.
- [26] eBPF. 2024. What is eBPF? an introduction and deep dive into the eBPF technology. <https://ebpf.io/what-is-ebpf/>. Accessed: 2026-07-02. (2024).
- [27] Peihua Zhang et al. 2024. Hive: a hardware-assisted isolated execution environment for ebpf on aarch64. In *Proceedings of the 33rd USENIX Conference on Security Symposium (SEC '24)* Article 10. USENIX Association, Philadelphia, PA, USA, 18 pages. ISBN: 978-1-939133-44-1.
- [28] Yi He, Roland Guo, Yunlong Xing, Xijia Che, Kun Sun, Zhuotao Liu, Ke Xu, and Qi Li. 2023. Cross container attacks: the bewildered ebpf on clouds. In *Proceedings of the 32nd USENIX Conference on Security Symposium (SEC '23)* Article 334. USENIX Association, Anaheim, CA, USA, 18 pages. ISBN: 978-1-939133-37-3.
- [29] [n. d.] GitHub - microsoft/ebpf-for-windows: eBPF implementation that runs on top of Windows — github.com. <https://github.com/microsoft/ebpf-for-windows>. [Accessed 16-07-2026]. ().
- [30] Hongyi Lu, Shuai Wang, Yechang Wu, Wanning He, and Fengwei Zhang. 2024. Moat: towards safe bpf kernel extension. In *Proceedings of the 33rd USENIX*

- Conference on Security Symposium (SEC '24) Article 65. USENIX Association, Philadelphia, PA, USA, 18 pages. ISBN: 978-1-939133-44-1.
- [31] Xueyuan Han, Thomas Pasquier, and Margo Seltzer. 2018. Provenance-based intrusion detection: opportunities and challenges. In *Proceedings of the 10th USENIX Conference on Theory and Practice of Provenance (TaPP'18)*. USENIX Association, London, UK, 3.
- [32] Fan Yang, Jiachen Xu, Chunlin Xiong, Zhou Li, and Kehuan Zhang. 2023. Prographer: an anomaly detection system based on provenance graph embedding. In *Proceedings of the 32nd USENIX Conference on Security Symposium (SEC '23)* Article 244. USENIX Association, Anaheim, CA, USA, 18 pages. ISBN: 978-1-939133-37-3.
- [33] Ahmed Lekkays, Hamza Mouhcine, Khang Tran, Ting Yu, and Issa Khalil. 2025. Llmxcpg: context-aware vulnerability detection through code property graph-guided large language models. In *Proceedings of the 34th USENIX Conference on Security Symposium (SEC '25)* Article 26. USENIX Association, Seattle, WA, USA, 19 pages. ISBN: 978-1-939133-52-6.
- [34] Red Canary. 2020. Invoke-AtomicRedTeam: a PowerShell-based execution framework for Atomic Red Team. <https://github.com/redcanaryco/invoke-atomicroteam>. Accessed: 2026-07-02. (2020).
- [35] Bader Al-Sada, Alireza Sadighian, and Gabriele Oliveri. 2024. Mitre att&ck: state of the art and way forward. *ACM Comput. Surv.*, 57, 1, Article 12, (Oct. 2024), 37 pages. doi:10.1145/3687300.
- [36] [n. d.] GitHub - aquasecurity/tracee: Linux Runtime Security and Forensics using eBPF — github.com. <https://github.com/aquasecurity/tracee>. [Accessed 06-07-2026]. ().
- [37] [n. d.] GitHub - toon-format/toon: Token-Oriented Object Notation (TOON) — Compact, human-readable, schema-aware JSON for LLM prompts. Spec, benchmarks, TypeScript SDK. — github.com. <https://github.com/toon-format/toon>. [Accessed 06-07-2026]. ().
- [38] Jiahui Wang, Xiangmin Shen, Zhengkai Wang, and Zhenyuan Li. [n. d.] The case for llm-enhanced backward tracking.
- [39] Canhua Chen, Yang Song, Yunpeng Li, Yun Feng, Xiaoyu Wang, Yuling Liu, and Qixu Liu. 2025. Prov-llm: advanced persistent threat detection based on process behavior subgraphs using large language models. In *2025 IEEE 24th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 978–985. doi:10.1109/TrustCom66490.2025.00113.
- [40] [n. d.] Attack-stix-data/enterprise-attack/enterprise-attack.json. <https://github.com/mitre-attack/attack-stix-data/blob/master/enterprise-attack/enterprise-attack.json>. [Accessed 06-07-2026]. ().
- [41] Sean Lee, Aamir Shakir, Darius Koenig, and Julius Lipp. 2024. Open source strikes bread - new fluffy embeddings model. <https://www.mixedbread.ai/blog/mxbai-embed-large-v1>.
- [42] Xianming Li and Jing Li. 2023. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*.
- [43] Sandhini Agarwal et al. 2025. Gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.
- [44] David Patterson et al. 2022. The carbon footprint of machine learning training will plateau, then shrink. *Computer*, 55, 7, 18–28.
- [45] Paul Kassianik et al. 2025. Llama-3.1-foundationai-securityllm-base-8b technical report. *arXiv preprint arXiv:2504.21039*.
- [46] Qwen Team. 2026. Qwen3.5: towards native multimodal agents. (Feb. 2026). <https://qwen.ai/blog?id=qwen3.5>.
- [47] 2021. Mitre att&ck: design and philosophy. <https://www.mitre.org/sites/default/files/2021-11/prs-19-01075-28-mitre-attack-design-and-philosophy.pdf>. [Accessed 09-07-2026]. (2021).
- [48] Center for Threat-Informed Defense. 2026. Ambiguous Techniques — ctid.mitre.org. <https://ctid.mitre.org/projects/ambiguous-techniques/>. [Accessed 09-07-2026]. (2026).

A Evaluation Metrics

Let N denote the total number of valid test cases, and let $\text{rank}_i \in \{1, \dots, K\}$ denote the position of the correct label in the ranked list of predictions for test case i (with $\text{rank}_i = \infty$ if the correct label does not appear within the top- K candidates).

Hit Rate (HR@K). HR@K measures the percentage of test cases for which the correct label appears anywhere within the top- K ranked predictions, regardless of its exact position:

$$\text{HR@K} = \frac{100}{N} \sum_{i=1}^N I(\text{rank}_i \leq K) \quad (1)$$

where $I(\text{rank}_i \leq K)$ is an indicator function that evaluates to 1 if the ground-truth label appears in the top- K predictions and 0 otherwise.

Mean Reciprocal Rank (MRR@K). MRR@K rewards predictions in which the correct label is ranked closer to the top, assigning a reciprocal-rank score of zero whenever the correct label falls outside the top- K :

$$\text{MRR@K} = \frac{1}{N} \sum_{i=1}^N \begin{cases} \frac{1}{\text{rank}_i} & \text{if } \text{rank}_i \leq K \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Normalized Discounted Cumulative Gain (NDCG@K). Since each test case has exactly one relevant (ground-truth) label, the ideal DCG@K is always 1 (achieved when the correct label is ranked first), and NDCG@K reduces to:

$$\text{NDCG@K} = \frac{1}{N} \sum_{i=1}^N \begin{cases} \frac{1}{\log_2(\text{rank}_i + 1)} & \text{if } \text{rank}_i \leq K \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

NDCG@K provides a softer rank-sensitive score, rewarding correct predictions more when they appear near the top of the ranked list while penalizing lower-ranked matches less aggressively than MRR .

B Invalid JSON output during evaluation tests

Table 5: Number of invalid JSON outputs for each model, with and without RAG.

	Pure LLM	RAG
deepseek-r1-distill-qwen-32b	0	0
foundation-sec-8b-reasoning	16	20
gemma4-31b-it	0	0
gpt-oss120b	2	0
gpt-oss-20b	8	6
llama-3.3-70b-instruct	1	1
qwen3.5-9b	0	1

C System Prompt

LLM System Prompt

You are a cyber threat intelligence analyst analyzing an attack sequence received as a DOT file. The DOT file represents the FULL attack sequence as a complete command graph with interconnected nodes and edges.

Your Task

1. Read and analyze the ENTIRE graph comprehensively - examine ALL nodes, ALL edges, and the COMPLETE flow from start to finish
2. Map the WHOLE attacker sequence (not isolated commands) to MITRE ATT&CK techniques
3. For each mapping, consider how commands relate to each other across the entire workflow
4. Select the top 5 most appropriate techniques.

Critical Requirements for Each Technique:

- MUST identify the complete TACTIC
- MUST identify the complete TECHNIQUE with sub-technique in format: 'Technique Name (ID)' where ID includes sub-technique (e.g., "T1003.007" not just "T1003")
- sub-techniqueS are REQUIRED: If a technique has sub-techniques and the attack shows sub-technique-level behavior, you MUST map to the sub-technique (e.g., T1003.007 "Proc Filesystem" not just T1003 "OS Credential Dumping")
- confidence: a (0,1) value reflecting how well the ENTIRE flow supports this mapping
- rationale: Explain WHY this mapping fits based on the HOLISTIC attack pattern across the entire graph flow, NOT based on a single isolated command

Rationale Writing Guidelines:

- BAD: "Mapping chosen due to presence of 'sh' command"
- GOOD: "Mapping chosen because the entire flow shows process memory dumping through sequential commands: target process launch → sh-based memory extraction → credential file creation, indicating a complete OS credential dumping workflow"
- BAD: "Found T1003 because of dump command"
- GOOD: "The complete attack sequence demonstrates credential dumping through the entire workflow: initial process targeting, memory extraction via shell, and subsequent credential file manipulation, matching T1003.007 Proc Filesystem sub-technique"

Output Rules:

Each element of the output is a JSON entry containing:

- "tactic": the ATT&CK tactic name
- "technique": the ATT&CK technique with sub-technique name and ID in format 'Technique Name (ID)' (e.g., "OS Credential Dumping: Proc Filesystem (T1003.007)")
- "confidence": a (0,1) value
- "rationale": explanation based on the COMPLETE attack flow, not isolated commands Return ONLY a JSON array containing the 5 candidates ordered by confidence descending. NO text, explanations, or markdown outside the JSON array.