The Case for LLM-Enhanced Backward Tracking

Jiahui Wang[*]
Zhejiang University
Hangzhou, China
wjh_13@zju.edu.cn

Xiangmin Shen[*]
Hofstra University
Hempstead, NY, USA
xiangmin.shen@hofstra.edu

Zhengkai Wang
Zhejiang University
Hangzhou, China
22451237@zju.edu.cn

Zhenyuan Li[✉]
Zhejiang University
Hangzhou, China
lizhenyuan@zju.edu.cn

***Abstract*—Provenance-based backward tracking is a critical technique for investigating Advanced Persistent Threats (APTs). However, existing approaches utilizing reachability analysis or statistical anomaly detection often suffer from dependency explosion and a significant semantic gap. These methods cannot typically distinguish high-level adversarial intent from benign administrative activities, resulting in a substantial number of false positives.**

In this paper, we introduce TRACKAGENT, a novel system that conceptualizes backward tracking as a knowledge-augmented, context-aware reasoning task. By leveraging Large Language Models (LLMs) enhanced with a knowledge augmentation module, TRACKAGENT aims to bridge the gap between low-level log events and attack intent. Furthermore, we design a context management model to handle the long-term dependencies of APT campaigns within finite context windows.

We report preliminary evaluations on DARPA TC, Aurora, and OpTC datasets to assess the feasibility of this approach. Early results suggest that compared to state-of-the-art baselines, TRACKAGENT can achieve higher fidelity (precision and recall) while generating significantly smaller attack subgraphs. These findings provide early evidence of the LLM-enhanced system's potential to detect critical attack behaviors from massive background noise, while offering analysts concise and interpretable forensic explanations.

I. INTRODUCTION

Recently, sophisticated cyber attacks, particularly Advanced Persistent Threats (APTs), have increasingly targeted critical infrastructures, causing significant damage [1], [2]. As adversaries frequently evade initial defenses to establish persistence, provenance-based backward tracking becomes a critical forensic capability for identifying root causes and assessing impact once an intrusion is detected. The core objective of backward tracking is to reconstruct the attack scenario triggered by a security alert, leveraging data and control flows to locate the attack entry point and evaluate its impact [3].

To support such a comprehensive analysis, provenance graphs have proven instrumental, offering a detailed representation of system activities [4]. Provenance graphs parse system audit logs into a unified event format, representing processes, files, and their behavior as nodes and edges. This representation transforms discrete, fragmented log entries into a temporal graph with inherent causality, enabling analysts to investigate the complex dependencies within system activities effectively.

Prior research based on provenance graphs has largely focused on automating forensic tracking through three primary strategies: reachability analysis, which utilizes tag propagation and graph traversal algorithms to trace the ancestry and impact of suspicious events [5], [6]; subgraph matching, which detects known threats by aligning provenance subgraphs with predefined attack knowledge bases or detection rules [7], [8]; and anomaly detection, which employs statistical or machine learning models to identify system behaviors that deviate from benign execution baselines [9], [10]. While these approaches have achieved significant advancements, they still encounter several challenges in coping with the increasing stealth and complexity of modern APT attacks. First, these approaches lack the capability to capture adversarial intent directly. Whether relying on pattern alignment or statistical deviations, they are fundamentally designed based on predefined assumptions derived from historical attack patterns. Consequently, in the face of increasingly evasive attacks, they struggle to distinguish well-designed actions that mimic benign activity patterns. Second, they suffer from limited interpretability, which is a persistent challenge inherent to learning-based approaches. Although causal paths or risk scores offer structural indicators, they still fail to provide transparent, natural-language explanations, which are essential for analysts to validate forensic findings and efficiently rule out false positives.

To address these challenges, we conceptualize backward tracking as a knowledge-augmented, context-aware reasoning process. This mechanism is inspired by the cognitive workflow of security analysts: rather than mechanically traversing graph edges, analysts leverage domain knowledge and external threat intelligence to interpret the semantics of events such as command-line arguments. More importantly, they maintain a dynamic memory model of the attack activities throughout the tracking, which allows them to infer adversarial intent in contrast to evaluating each event independently.

Recent advances in Large Language Models (LLMs) offer a promising pathway to automate this cognitive process. Benefiting from pre-training on massive corpora, LLMs have not only internalized vast general knowledge but also acquired logical reasoning capabilities through the phenomenon of emergence [11]. These attributes provide a promising technical pathway to simulate the workflows of security experts, thereby

[*] Two authors contribute equally to this work.
[✉] Corresponding author: Zhenyuan Li (lizhenyuan@zju.edu.cn)

Workshop on Attack Provenance, Reasoning, and Investigation for Security in the Monitored Environment (PRISM) 2026
23 February 2026, San Diego, CA, USA
ISBN 978-1-970672-00-8
https://dx.doi.org/10.14722/prism.2026.23022
www.ndss-symposium.org

achieving more effective backward tracking.

However, applying LLMs to this task faces two significant challenges. The first is the domain knowledge gap and knowledge cutoff [12]: general LLMs lack the specialized understanding of modern attacks and heterogeneous log structures, leading to incorrect judgments or hallucinations when interpreting events. The second is context maintenance. Provenance graphs for APT attacks typically span long time windows and involve massive sequences. Constrained by token window limits, it is impractical to simply inject all tracked attack events into the prompt as the context, which means LLMs cannot process the entire history of an APT attack in a single interaction without effective context design.

To overcome these limitations, we propose TRACKAGENT, a novel system that transforms backward tracking into a knowledge-augmented, context-aware reasoning process. TRACKAGENT achieves this through two core components. First, to bridge the domain knowledge gap, we design a knowledge augmentation module that provides targeted information from CTI reports, MITRE ATT&CK TTPs [13], and log schemas, enabling the LLMs to use professional and effective security knowledge to distinguish malicious events and mitigate hallucinations. Second, to resolve the context limitation, we design a context management module acting as a stateful memory. By maintaining a summarized narrative of critical events alongside their corresponding tactical chain, this module enables the system to preserve a global perception of complex attack scenarios within a limited token window.

We report early results on DARPA TC, OpTC, and Aurora scenarios. Compared to NODOZE and DEPIMPACT, TRACKAGENT often retains a smaller investigation subgraph while recovering more attack-relevant events, suggesting that knowledge plus stateful context can reduce false positives in ambiguous regions of provenance.

This paper presents early evidence that knowledge-augmented, context-aware reasoning can make backward tracking both more precise and more usable for analysts. As a work in progress, our current prototype and evaluation focus on representative scenarios and a small set of baselines. We welcome feedback from the community on the problem framing, system design choices, and evaluation methodology, as we extend TRACKAGENT to broader datasets, stronger baselines, and more diverse attack behaviors.

II. BACKGROUND AND MOTIVATION

A. Provenance Graph and Backward Tracking

Provenance Graph. Following standard definitions, we model the system execution history as a provenance graph, a directed acyclic graph $G = (V, E)$ [4]. Nodes V represent system entities such as processes, files, and network sockets, where each node $v \in V$ contains a set of attributes. Edges E represent causal interactions between entities, denoted as $e = (u, r, v, t)$, indicating that subject u operates r on object v at time t.

Backward Tracking. Backward tracking is a fundamental process in forensic analysis [3]. In standard security operations, Intrusion Detection Systems (IDS) continuously monitor system activities and identify suspicious behaviors to generate alerts, referred to as Points-of-Interest (POIs). A POI event, such as a malicious shell command or an anomalous network connection, serves as the starting point for investigation. The primary objective is to traverse the provenance graph backward to trace the causal dependencies of the target POI, aiming to reconstruct the full attack scenario and correlate the detected anomaly with the initial intrusion (root cause).

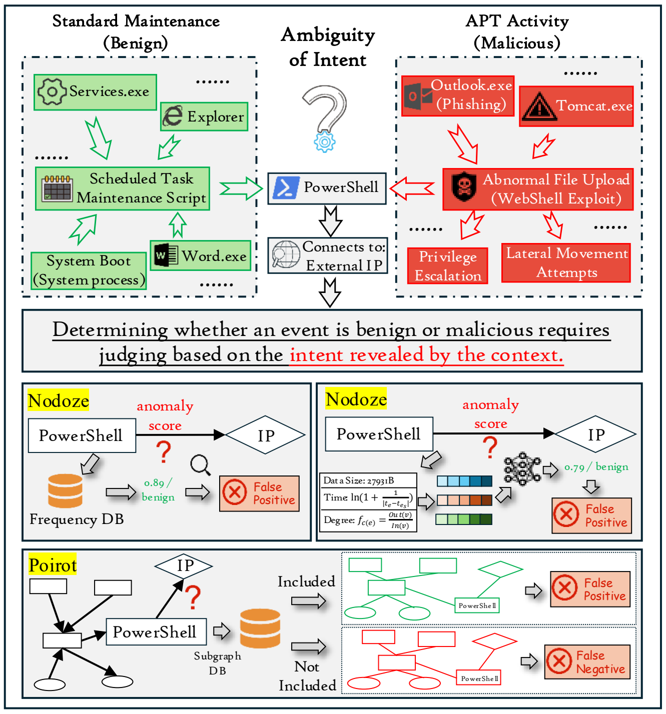


Fig. 1: Motivational example: the ambiguity of local topology. While the target event (PowerShell connection) is structurally identical in both graphs, its malicious nature can only be determined by reasoning over the preceding causal path (history) and the semantic attributes of the command (knowledge).

Formally, given a provenance graph $G = (V, E)$ and a POI event denoted as e_{poi}, backward tracking aims to identify a critical attack subgraph $G^* \subseteq G$ that captures the attack scenario preceding e_{poi}. This optimization problem is formalized as selecting a subset of causally relevant edges:

$$G^* = \{e \in E \mid \mathcal{R}_{attack}(e, e_{poi})\} \tag{1}$$

Here, $\mathcal{R}_{attack}(e, e_{poi})$ denotes a causal relevance condition. This binary relation holds if and only if event e is structurally reachable from the POI event e_{poi} (i.e., they belong to the same causal dependency chain) and semantically contributes to the adversary's intent. The optimization goal is to minimize $|G^*|$ to reduce noise while maximizing the coverage of $\mathcal{R}_{attack}$ to ensure completeness.

B. Motivation: The Challenge of Contextual Ambiguity

Prior research has primarily focused on mitigating the dependency explosion problem [14]. Due to the inherent complexity of system activities, the number of benign events exceeds the malicious ones. If the attack subgraph generated by tracking is saturated with benign events, effective forensic analysis becomes infeasible.

Existing approaches typically employ statistical anomaly scoring or predefined topological rules to prune irrelevant events. Although partially effective, these methods struggle to combat modern stealth attacks because they fail to capture the attacker's intent.

It is important to capture the attacker's intent because raw system events often show neutral characters when viewed in isolation. For instance, a connection initiated by `powershell.exe` is functionally consistent and topologically indistinguishable, regardless of whether it represents a routine administrative task or a lateral movement attempt by an APT adversary. As illustrated in Figure 1, in Scenario A (Benign), a `powershell.exe` process is spawned by `services.exe` during a scheduled maintenance window. While in Scenario B (Malicious), an identical `powershell.exe` is spawned by `tomcat.exe` (a vulnerable web server) following a suspicious file upload. Distinguishing between these scenarios requires comprehensive reasoning based on the intent revealed by the context.

As depicted in the lower part of Figure 1, prior approaches are limited in this scenario. Frequency-based method—NoDoze [9] is prone to false negatives. Since PowerShell network activity is ubiquitous in normal administrative workflows, such events are often statistically indistinguishable from benign baselines. Similarly, DepImpact [5] struggles to differentiate malicious events from background noise, as its empirically selected statistical features are insufficient to characterize high-level adversarial intent. Finally, subgraph matching methods like Poirot [7] are constrained by their adherence to topological signatures. These methods are vulnerable to evasion when attack activities coincidentally mimic benign patterns or when adversaries employ mutations to deviate from known rules.

These limitations demonstrate that relying solely on topological structure or statistical metrics is insufficient. Consequently, we look to the methodology of human security experts, conceptualizing the backward tracking task as a knowledge-augmented, context-aware reasoning task.

C. The Knowledge-Augmented, Context-Aware Reasoning

We conceptualize backward tracking as a knowledge-augmented, context-aware reasoning task. Let e_t be the current event under backward tracking. In our proposed framework, the decision function is modeled as a conditional probability dependent on two variables:

$$f_{reasoning}(e_t) = P(\mathit{IsMalicious}(e_t) \mid \mathcal{K}_s, \mathcal{C}_c) \tag{2}$$

$\mathcal{K}_s$ represents the static, domain-specific knowledge required to interpret event semantics. Raw log attributes are typically low-level and semantically sparse, such as commands, hash values, and binary flags. To perform effective reasoning, the system must contextualize these raw artifacts by leveraging external security knowledge.

$\mathcal{C}_c$ represents the dynamic, evolving history of the backward tracking. It captures the causal dependencies and consistency of the attack scenario. If the tracking has traversed a path $p = \{e_0, e_1, ..., e_{t-1}\}$, the relevance of e_t is conditional on this accumulated state.

However, directly applying general-purpose LLMs poses significant challenges. First, regarding $\mathcal{K}_s$, general-purpose models lack the specialized knowledge required for security analysis. Trained on public datasets, LLMs cannot act as domain experts directly, and their frozen knowledge base fails to keep pace with rapidly evolving APT techniques. Second, regarding $\mathcal{C}_c$, maintaining a complete causal context is hindered by the finite context window. In real-world scenarios, backward tracking often involves dependency chains extending to thousands of events. Directly injecting this comprehensive history into the prompt incurs prohibitive overhead and exceeds the model's token limit [15].

III. System Design

As illustrated in Figure 2, TrackAgent operates as an iterative reasoning pipeline initiated by a Point-of-Interest (POI) event. The system reconstructs the attack scenario through three sequential phases: knowledge retrieval, which bridges the domain knowledge gap by retrieving relevant intelligence via a hybrid search mechanism; context management, which maintains a stateful memory to record the backtracking's progress; and reasoning and state update, where the LLM performs inference using a structured prompt and dynamically updates the results and context. The detailed procedure is formalized in Algorithm 1 in Appendix B.

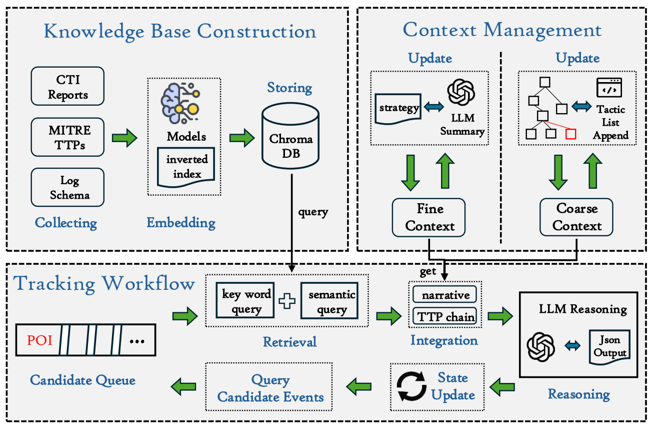


Fig. 2: The workflow of TrackAgent.

A. Knowledge Base Construction and Retrieval

To address the domain knowledge gap and the knowledge cutoff of LLMs, TrackAgent incorporates a specialized knowledge augmentation module. Rather than simply feeding raw documents, we design a structured pipeline for knowledge construction and a retrieval mechanism to ensure precise context alignment.

Knowledge Base Construction. We aggregate three distinct types of information to form a comprehensive view of the threat landscape: CTI reports for macro-level descriptions of the attack lifecycle, also enabling LLMs to keep pace with evolving adversarial techniques; MITRE ATT&CK Tactics, Techniques, and Procedures (TTPs) for standardizing tactical behaviors, and log schemas for understanding heterogeneous system events.

To make these unstructured text sources like CTI and MITRE TTPs retrievable, we employ a semantic-aware chunking strategy during the construction phase. Instead of strictly adhering to fixed-character sliding windows, which often sever sentences and destroy logic, we parse documents based on natural language boundaries while enforcing a maximum character constraint. In forensic reasoning, a single sentence often encapsulates a complete causal logic. Preserving this semantic integrity is crucial to preventing the LLM from hallucinating relationships based on fragmented text. These processed chunks are then embedded into high-dimensional vectors using an embedding model and stored in a vector database. Distinctly, the log schema is maintained as a lightweight key-value table to directly map log fields to their natural language definitions. We also maintain an inverted index of the raw text to support keyword-based queries.

Hybrid Retrieval Strategy. Based on this infrastructure, we implement a hybrid retrieval mechanism to align unstructured security knowledge. This approach is necessitated by the dual nature of security analysis, which requires retrieving both high-level semantic intent and precise low-level indicators like specific registry keys. Relying solely on vector embeddings often dilutes unique technical artifacts like IOCs, whereas pure keyword search fails to bridge terminological gaps. Consequently, we combine the BM25 algorithm for exact keyword matching with dense vector search for semantic understanding. For each candidate event, we construct a query using its key attributes, such as process command lines and file paths, and corresponding context described in Section II-B, retrieving the top-3 knowledge fragments based on a combined score where both retrieval methods are weighted equally. After that, definitions of specific log fields are directly queried from the log schema table to ensure precise semantic interpretation.

B. Context Management

After interpreting the semantics of individual events via the knowledge augmentation module, TRACKAGENT must evaluate their malice within the historical context of the ongoing backtracking. However, provenance graphs for APT attacks typically span long time windows, and injecting the entire history into a prompt is impractical due to token limits. To address this, we design a stateful context management module that maintains a two-level memory model, fine context and coarse context, to balance information density with global perception.

Fine context is designed as a dynamic, natural language narrative that functions as the system's short-term memory. Structurally, it is not a raw list of events but a cohesive summary paragraph describing the reconstructed attack story, such as stating that the attacker compromised the host via RDP, then executed a PowerShell script to download a payload... To maintain this context within token constraints, we employ a memory decay strategy inspired by human cognition. When a new event is identified as attack-related, the system triggers an LLM-based update process whose prompt is shown in Figure 4. This process rewrites the fine context, keeping the most recent events described with granular details while compressing older events into high-level semantic summaries. This design choice ensures that the LLM always has access to the immediate causal predecessors required for local reasoning.

Conversely, we also design a coarse context to serve as the long-term tactical constraint. It acts as an ordered sequence of identified MITRE ATT&CK tactics such as *[Initial Access, Execution, Persistence]*. This sequence is constructed by mapping the identified attack-related events to MITRE TTPs. The primary purpose of this component is to solve the dependency explosion caused by the sycophancy problem [16], which means that LLMs tend to over-interpret benign events as malicious in backward tracking. Coarse context prevents this by monitoring the logical completeness of the attack chain. For instance, if the coarse context indicates that the investigation has successfully traced back to an *Initial Access* tactic, the system recognizes this as a logical boundary of the attack campaign. Consequently, it guides the LLM to correctly distinguish subsequent benign events, thereby aggressively and accurately pruning the search space.

C. Reasoning and State Update

Finally, TRACKAGENT integrates the retrieved knowledge and maintained context into a structured prompt to execute the decision-making process. As visualized in Figure 5 in Appendix C, the prompt is constructed using a specialized template. Instead of merely concatenating inputs, we employ specific prompt engineering techniques, including role-play to establish expert personas and Chain-of-Thought (CoT) instructions to enforce step-by-step logic. This design ensures that the LLM performs inference by synthesizing local event semantics with global attack progression effectively.

To ensure TRACKAGENT's decisions are parsable for automated algorithm expansion, we enforce a strict JSON output schema, as defined in Figure 6 in Appendix C. By constraining the LLM to return a structured object containing critical fields—such as confidence score, and natural language reasoning—the system can mechanically filter low-confidence predictions while retaining rich interpretability for human analysts without parsing complex free text.

After detecting an attack-related event, the system executes a state update to reflect the latest tracking status. The new event is added to the result graph G_{attack}. Crucially, the outputs directly drive the context management update described in Section III-B: the extracted techniques update the coarse context, while the event details trigger the fine context's update. This update process utilizes the prompt illustrated in Figure 4 in Appendix C, which explicitly instructs the model to apply a memory decay strategy to preserve a global perception within finite token limits. With the state updated, the tracking process then recursively expands to the event's predecessors until the rate of change in the graph size falls below a defined threshold.

TABLE I: Results of attack scenarios. Rec: Recall, Prec: Precision, $|G|$: Final graph size (edges), $|E|$: Input edge count.

Scenario	Input	NoDoze			DepImpact			Ours		
	$\|E\|$	|G|	Rec	Prec	|G|	Rec	Prec	|G|	Rec	Prec
Dataset: OpTC										
Case 1	829K	126	68%	14%	76	68%	23%	**51**	**85%**	**43%**
Case 2	1.4M	221	71%	16%	130	57%	22%	**73**	**78%**	**53%**
Case 3	1.8M	252	65%	9%	112	64%	20%	**52**	**77%**	**52%**
Dataset: Aurora										
Case 1	86K	83	88%	18%	46	82%	30%	**26**	**94%**	**62%**
Case 2	74K	136	71%	12%	85	74%	20%	**37**	**87%**	**54%**
Case 3	90K	162	72%	12%	42	70%	45%	**35**	**85%**	**66%**
Dataset: DARPA										
CADETS-1	3.8M	50	71%	24%	55	65%	20%	**28**	**94%**	**57%**
CADETS-2	3.8M	80	50%	5%	32	50%	13%	**12**	**63%**	**42%**
CADETS-3	3.6M	135	61%	20%	92	73%	35%	**72**	**91%**	**56%**
CADETS-4	3.7M	114	76%	14%	42	57%	38%	**34**	**90%**	**56%**

IV. Evaluation

In this section, we present a preliminary evaluation of TrackAgent to validate its feasibility as a knowledge-augmented, context-aware reasoning system. As a work-in-progress paper, our evaluation focuses on two key objectives: (1) verifying whether our approach can effectively reconstruct high-fidelity attack graphs with significantly reduced benign noise compared to state-of-the-art baselines; and (2) demonstrating the system's interpretability and reasoning logic through a detailed case study.

A. Experimental Setup

We implemented TrackAgent in Python 3.10, utilizing Neo4j (v5.11) as the graph database backend for efficient provenance storage and retrieval. The system is hosted on a server running Ubuntu 24.04.1 LTS, equipped with an Intel Xeon Gold 6330 CPU @ 2.00GHz and 1.0 TB of RAM. For the core reasoning engine, all experiments were conducted using the DeepSeek-R1 model via its API [17], unless otherwise specified.

We utilize the DARPA Transparent Computing (TC) datasets [18] (including four CADETS scenarios), the Aurora dataset [19], which is created by an automated attack construction engine, and the OpTC dataset [20], which represents real-world APT scenarios. We compare TrackAgent against two representative baselines: NoDoze [9] (statistical anomaly scoring) and DepImpact [5] (learning-based propagation).

To quantify performance, we measure: (1) graph reduction, which is represented by the number of edges in the final reconstructed graph ($|G|$), and (2) detection quality, which is measured by recall and precision based on recognized attack events.

B. Backward tracking Efficacy

We evaluate TrackAgent against NoDoze and DepImpact in terms of their efficacy in attack reconstruction and their capability to prune noise. Table I shows the performance across various datasets. Generally, TrackAgent consistently achieves superior precision and recall while maintaining significantly smaller attack graphs than the baselines.

In complex scenarios involving massive background system activities, such as the OpTC dataset, existing systems struggle significantly with dependency explosion. DepImpact and NoDoze exhibit low precision, often retaining hundreds of irrelevant edges due to their inability to distinguish between structurally reachable benign events and adversarial events. In contrast, TrackAgent can filter these semantically irrelevant activities, maintaining high precision (up to 85%) and reducing the graph size to fewer edges. This demonstrates its robust ability to identify adversarial intent within noisy scenarios where other systems frequently introduce false positives.

In datasets with more distinctive attack patterns, such as Aurora, all three systems achieve relatively high recall, reflecting that the compact attack sequences align well with general tracking assumptions. However, the baselines still generate higher false positives since they typically misclassify benign administrative anomalies as malicious events. TrackAgent performs better by producing the smallest attack graphs with the highest fidelity. This performance is attributed to the fact that the attack sequences in these datasets strictly follow the tactical order of the MITRE ATT&CK framework [13], [21], enabling TrackAgent to terminate unnecessary paths efficiently and accurately.

On the DARPA CADETS dataset, TrackAgent also shows superior noise pruning capabilities. A notable exception is observed in the CADETS-2, where the recall dropped to 63% (with precision at 42%). Our analysis indicates that this exception is caused by a specific TTP that belongs to multiple MITRE tactics. During the coarse context update, the LLM selected an incorrect tactic for this TTP. Consequently, the system misjudged the attack progress and terminated the backward tracking process prematurely. Addressing such TTPs that correspond to multiple tactics requires a specific fallback mechanism, which we leave as future work.

C. Ablation Study

To evaluate the individual contributions of the proposed components in TrackAgent, we conducted an ablation study on the DARPA CADETS dataset. We measured the average performance across all four scenarios by selectively removing specific modules from TrackAgent. The results are summarized in Table II.

We first evaluate the impact of external knowledge sources. Removing MITRE TTPs leaves recall stable (83%) but significantly drops precision to 35%, indicating that without the assistance of MITRE-related knowledge, the system struggles

TABLE II: Ablation study on the CADETS dataset (Average of 4 scenarios). We evaluate the contribution of each component: each knowledge source, fine context (C_{fine}), and coarse context (C_{coarse}).

Configuration	Size (↓)	Rec. (↑)	Prec. (↑)
TRACKAGENT (Full)	**37**	**85%**	**53%**
w/o Knowledge Components:			
w/o CTI Reports	39	84%	51%
w/o MITRE TTPs	54	83%	35%
w/o Log Schema	62	65%	25%
w/o Context Components:			
w/o Fine Context	21	52%	58%
w/o Coarse Context	125	88%	17%

to accurately complete the tactical mapping task, leading to an increase in false positives. Meanwhile, removing the log schema leads to a decline in both recall and precision, confirming its necessity for interpreting heterogeneous audit logs. In contrast, removing CTI reports caused negligible performance fluctuations. We attribute this to the age of the CADETS dataset, as its specific attack patterns have likely been internalized into the LLM's pre-training data.

Subsequently, we evaluate the impact of context management strategies. Removing the fine context forces the system to evaluate events in isolation, restricting reasoning to the local topology near the POI. This inhibits full graph expansion, resulting in the smallest graph size (21) but a sharp drop in recall (52%). Conversely, removing the coarse context eliminates the tactical constraint needed to determine the attack boundary. Consequently, the system fails to terminate the search appropriately and drifts into benign super nodes, leading to an exploded graph size (125) and a precipitous decline in precision (17%).

D. Case Study

To further illustrate the reasoning capabilities of TRACKAGENT, we visualize the provenance analysis results for case 1 from the OpTC dataset, which represents a typical APT scenario involving RDP lateral movement. Figure 3 in Appendix B depicts the constructed attack subgraph. As shown, the system yields a highly concise graph predominantly comprising correctly identified attack events (red nodes and edges). While the output retains a negligible number of benign events (grey nodes and edges), this volume is significantly reduced compared to the vast scale of the original logs.

The core advantage of our approach is the interpretable reasoning highlighted by the annotations in Figure 3. At the attack's entry point, the system successfully identifies a malicious IP connection (`10.*.*.1`). The LLM explicitly reasons that this network event is the direct reason why `update.exe` was placed on the host. Furthermore, by combining file size attributes with the coarse context, the system determines that this event represents the starting point of the attack (Phishing, T1195). This global awareness allows the system to confidently terminate the backward search at the root cause, avoiding the supernode diffusion.

Similarly, during the lateral movement phase, the system demonstrates deep intent understanding. When analyzing the `whoami.exe` command, the LLM utilizes the fine context to link it with the subsequent creation of `data.zip`. It recognizes that checking user identity is a necessary precursor operation for data leakage, thus correctly classifying these events as part of a System Owner/User Discovery (T1033) campaign rather than isolated benign activities. Finally, as shown in the right part of Figure 3, TRACKAGENT synthesizes this structural information into a comprehensive attack summary and provides actionable mitigation, validating the system's practical utility for security operations.

V. FUTURE WORK

In future work, we plan to extend our evaluation in three key dimensions. First, we will conduct a comparative analysis of different LLM backbones to identify the optimal balance between reasoning capability and inference cost. Second, we intend to provide a fine-grained analysis of token consumption to further assess the feasibility of real-world deployment. Finally, we plan to evaluate TRACKAGENT on newer datasets containing emerging threats. This will allow us to demonstrate the system's unique ability to keep pace with evolving adversarial techniques by leveraging up-to-date CTI reports, effectively bridging the knowledge gap without the need for model retraining.

VI. CONCLUSION

In this paper, we proposed TRACKAGENT, a novel framework that conceptualizes backward tracking as a knowledge-augmented, context-aware reasoning process. By integrating external domain knowledge with a stateful context management model, our approach effectively bridges the gap between low-level log events and high-level attack intent. Preliminary evaluations on DARPA TC, Aurora, and OpTC datasets demonstrate that TRACKAGENT significantly outperforms state-of-the-art baselines, generating highly concise attack graphs without compromising information fidelity. Future work will focus on optimizing the system's computational efficiency and validating its continuous adaptability to novel APT campaigns through dynamic knowledge retrieval.

REFERENCES

[1] Y. Hu, B. Wang, X. Wang, and W. Zang, "Provenance-based advanced persistent threat detection: A survey," *Journal of Information Security and Applications*, vol. 75, p. 103496, 2023.

[2] Z. Li, A. Soltani, A. Yusof, A. C. Risdianto, K. Huang, J. Zeng, Z. Liang, and Y. Chen, "Poster: Towards automated and large-scale cyber attack reconstruction with apt reports." NDSS 2022.

[3] S. T. King and P. M. Chen, "Backtracking intrusions," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 223–236, 2003, seminal work on backward tracing for intrusion investigation.

[4] T. Pasquier, X. Han, T. Goldstein, T. Moyer, D. Eyers, M. Seltzer, and J. Bacon, "Runtime analysis of whole-system provenance," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018, pp. 1601–1616, fundamental work on system-level provenance collection (CamFlow).

[5] P. Fang, P. Gao, C. Liu, E. Ayday, K. Jee, T. Wang, Y. F. Ye, Z. Liu, and X. Xiao, "Back-propagating system dependency impact for attack investigation," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 2461–2478.

[6] Y. Liu, Z. Zhang, X. Zhang, and D. Xu, "Priotracker: Backward tracing of forensic events with priority for attack investigation," in *Proceedings of the IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2018, pp. 482–493.

[7] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and A. Vullikanti, "Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019, pp. 1795–1812.

[8] ——, "Holmes: Real-time apt detection through correlation of suspicious flows," in *Proceedings of the 40th IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, pp. 1137–1152.

[9] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, "Nodoze: Combatting threat alert fatigue with automated provenance triage," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2019.

[10] A. Alsaheel, Y. Nan, S. Ma, L. Yu, G. Walkup, Z. B. Celik, X. Zhang, and D. Xu, "ATLAS: A sequence-based learning approach for attack investigation," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3005–3022.

[11] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler *et al.*, "Emergent abilities of large language models," *Transactions on Machine Learning Research*, 2022. [Online]. Available: https://openreview.net/forum?id=yzkSU5zdwD

[12] OpenAI, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023. [Online]. Available: https://arxiv.org/abs/2303.08774

[13] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK: Design and philosophy," https://attack.mitre.org/, 2018, technical Report.

[14] Z. Li, Q. Chen, R. Chen, Y. Ye, and S. Zhang, "Threat detection and investigation with system-level provenance graphs: A survey," *Computers & Security*, vol. 106, p. 102282, 2021.

[15] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024, provides empirical evidence for the "context maintenance" challenge.

[16] A. Wei, Jerry andr Dafoe, J. Wei, Y. Burda, J. Luketina, J. Bradury, Y.-T. Lo, J. Kiros, and Q. V. Le, "Simple synthetic data reduces sycophancy in large language models," in *International Conference on Machine Learning (ICML)*. PMLR, 2024. [Online]. Available: https://arxiv.org/abs/2308.03958

[17] DeepSeek-AI, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2501.12948

[18] DARPA Information Innovation Office, "Transparent computing (tc) program," https://www.darpa.mil/program/transparent-computing, 2016, the foundational dataset program for provenance analysis.

[19] L. Wang, Z. Li, Y. Jiang, Z. Wang, Z. Guo, J. Wang, Y. Wei, X. Shen, W. Ruan, and Y. Chen, "From sands to mansions: Towards automated cyberattack emulation with classical planning and large language models," *arXiv preprint arXiv:2407.16928*, 2024. [Online]. Available: https://arxiv.org/abs/2407.16928

[20] Five Directions, "Operation transparent computing (optc) dataset," https://github.com/FiveDirections/OpTC, 2020, large-scale provenance dataset representing real-world APT scenarios.

[21] X. Shen, Z. Li, G. Burleigh, L. Wang, and Y. Chen, "Decoding the mitre engenuity att&ck enterprise evaluation: An analysis of edr performance in real-world environments," in *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, 2024, pp. 96–111.

Appendix

A. Reasoning Algorithm

This appendix provides the formal pseudocode for the knowledge-augmented, context-aware reasoning process of TrackAgent. Algorithm 1 details the complete execution flow, illustrating how the system iteratively retrieves external knowledge ($\mathcal{K}_s$), maintains the context ($\mathcal{C}_{fine}$ and $\mathcal{C}_{coarse}$), and constructs structured prompts for the LLM. It specifically formalizes the state update mechanism and the termination condition used to prune the search space efficiently.

Algorithm 1: The knowledge-augmented, context-aware reasoning process

```
Input: Provenance Graph $G$, Point-of-Interest $e_{poi}$,
       Knowledge Base $\mathcal{KB}$ (CTI, TTPs, Schemas)
Output: Reconstructed Attack Graph $G_{attack}$
1  $Frontier \leftarrow$ Queue($e_{poi}$)
2  $G_{attack} \leftarrow \{e_{poi}\}$
3  $\mathcal{C}_{fine} \leftarrow \emptyset$;   $\mathcal{C}_{coarse} \leftarrow \emptyset$
4  while $Frontier$ is not empty do
5      $e_{curr} \leftarrow Frontier$.dequeue()
6      $\mathcal{K}_s \leftarrow$ RetrieveKnowledge($e_{curr}$)
7      $\mathcal{C}_{fine}, \mathcal{C}_{coarse} \leftarrow$ FetchContext($e_{curr}$)
8      $Prompt \leftarrow$ Construct($e_{curr}, \mathcal{K}_s, \mathcal{C}_{fine}, \mathcal{C}_{coarse}$)
9      $Output \leftarrow$ LLM_Reasoning($Prompt$)
10     if $Output.is\_relevant$ == 'yes' then
11         $\mathcal{C}_{coarse} \leftarrow$
             Update($\mathcal{C}_{coarse}, Output.techniques$)
12         $\mathcal{C}_{fine} \leftarrow$ Update($\mathcal{C}_{fine}, e_{curr}$)
13         $G_{attack\_new} \leftarrow G_{attack} \cup \{e_{curr}\}$
14         if $\frac{|G_{attack}|}{|G_{attack\_new}|} <$ threshold then
15             break
16         end
17         $Neighbors \leftarrow G$.get_predecessors($e_{curr}$)
18         $Frontier$.enqueue($Neighbors$)
19     end
20 end
21 return $G_{attack}$
```

B. Case Study Details

This appendix presents the detailed visualization of the provenance subgraph reconstructed by TrackAgent, as discussed in Section IV-D.

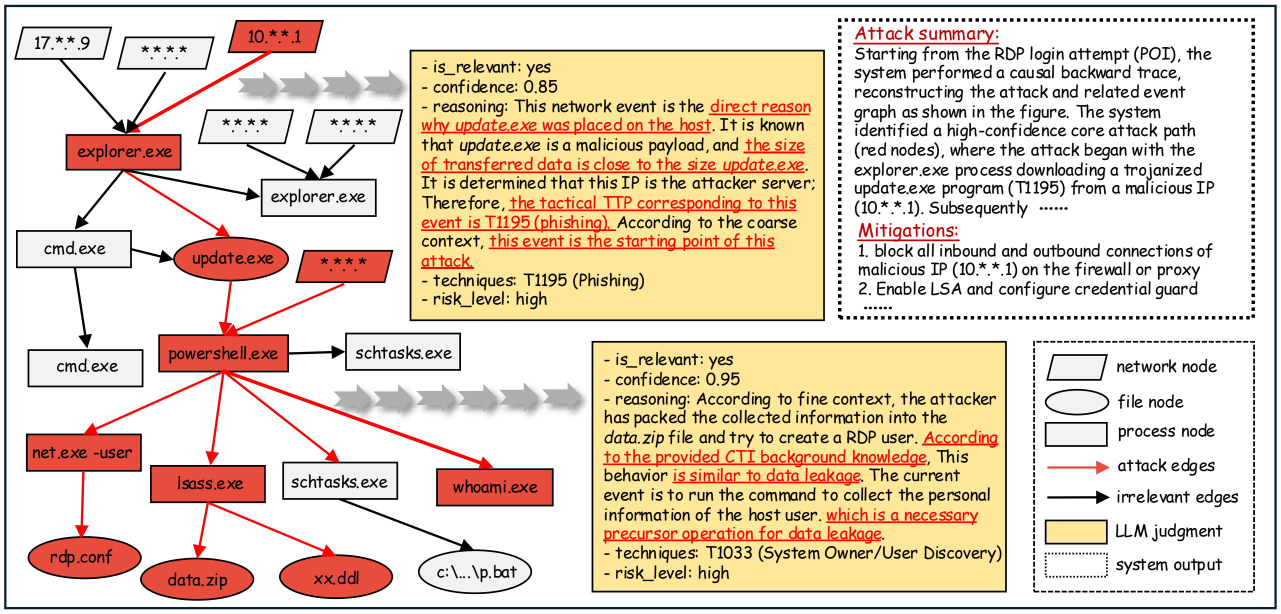
The figure illustrates the system's output regarding the RDP lateral movement campaign. In the visualization, red nodes and edges represent the malicious entities and activities identified by the system (True Positives), while grey nodes denote the benign background activities. The annotations (yellow boxes) highlight the specific reasoning logic generated by the Large Language Model (LLM) based on the components we design.

C. Structured Prompt Details

To facilitate reproducibility, this appendix presents the core prompt templates used in TrackAgent.

Figure 5 depicts the reasoning prompt, which synthesizes retrieved knowledge and hierarchical context to guide the LLM's decision-making.

To ensure the output is machine-parsable for automated graph expansion, we enforce the strict JSON schema shown in Figure 6.



- is_relevant: yes
- confidence: 0.85
- reasoning: This network event is the direct reason why *update.exe* was placed on the host. It is known that *update.exe* is a malicious payload, and the size of transferred data is close to the size *update.exe*. It is determined that this IP is the attacker server; Therefore, the tactical TTP corresponding to this event is T1195 (phishing). According to the coarse context, this event is the starting point of this attack.
- techniques: T1195 (Phishing)
- risk_level: high

Attack summary:
Starting from the RDP login attempt (POI), the system performed a causal backward trace, reconstructing the attack and related event graph as shown in the figure. The system identified a high-confidence core attack path (red nodes), where the attack began with the explorer.exe process downloading a trojanized update.exe program (T1195) from a malicious IP (10.*.*.1). Subsequently ......
Mitigations:
1. block all inbound and outbound connections of malicious IP (10.*.*.1) on the firewall or proxy
2. Enable LSA and configure credential guard
......

- is_relevant: yes
- confidence: 0.95
- reasoning: According to fine context, the attacker has packed the collected information into the *data.zip* file and try to create a RDP user. According to the provided CTI background knowledge, This behavior is similar to data leakage. The current event is to run the command to collect the personal information of the host user. which is a necessary precursor operation for data leakage.
- techniques: T1033 (System Owner/User Discovery)
- risk_level: high

Fig. 3: The detailed backtracking result on the OptC case 1 scenario. The annotations detail the LLM's reasoning process in determining the attack entry point and lateral movement intent.

Finally, Figure 4 illustrates the fine context update prompt. Implementing the memory decay strategy described in Section III, it instructs the model to compress historical events into semantic abstractions while preserving granular details for recent activities.

Fine Context Update Query (Simplified)

Role Play
You are a forensic investigation chronicler. Your task is to maintain a running narrative of a cyber attack investigation...

Input Data
Current context: The attacker compromised the host via RDP... [Existing summary of previous events]
New event: Process: `cmd.exe`, Operation: `Execute`, Args: `/c powershell -enc ...`

Update Strategy
Update the narrative by integrating the new event using the following decay logic:

- **Recent Focus (High Granularity):** Retain specific technical artifacts (e.g., arguments, IPs) for the new event.
- **Historical Compression (Semantic Abstraction):** Abstract older events into high-level descriptions of intent.
- **Causality Preservation:** Explicitly state the causal dependency between the historical context and the new event.

Task Constraint
Output ONLY the updated narrative as a single coherent paragraph. Do not include explanations.

Fig. 4: The structured prompt template used in the context management module for updating the fine context.

Tracking Reasoning Query (Simplified)

Role Play
You are a Senior Digital Forensics Analyst. Your goal is to filter false positives and reconstruct the attack chain accurately.

Context Integration
Coarse Context: [T1074.001, T1056, T1059.001, T1048 ...]
Fine Context: The attacker collected system credentials via a malicious component disguised as a legitimate library file ...

External Knowledge
CTI: A state government organization was notified that documents containing host and user credentials ...
MITRE Relationships: Technique Context (T1074.001): Local Data Staging. Description: Adversaries may ...
Log Schema: Field: subject. Description: The UUID of the event's subject (Actor), typically a process ...

CoT Instructions
Analyze the candidate event by first checking causality, then analyzing intent (alignment with identified Tactics), and finally verifying consistency ...

Task Constraint
Determine relevance based on the evidence above. Output the result strictly according to the specified JSON format.

Fig. 5: The structured prompt template used in TRACKAGENT for attack reasoning.

Output Format Specification

Constraint
Output ONLY a valid JSON object. Do not include markdown formatting or extra text.

JSON Schema

```
{
    "is_relevant": boolean (true/false),
    "confidence": float (0.0 - 1.0),
    "reasoning": string (Step-by-step justification...),
    "mitre_technique": string (e.g., T1074),
    "risk_level": string (Low/Medium/High)
}
```

Fig. 6: The required JSON output format for the reasoning module.