

# LLM-driven Provenance Forensics for Threat Intelligence and Detection

Kunal Mukherjee  
Virginia Tech

Murat Kantarcioglu  
Virginia Tech

**Abstract**—System provenance provides a rich *forensic trail* for analyzing stealthy cyberattacks (e.g., Advanced Persistent Threat (APT) campaigns). However, traditional detection pipelines rely heavily on recognized patterns and lack end-to-end automated mechanisms to integrate external intelligence or reason iteratively about complex attack traces. Therefore, analysts are required to craft ad-hoc queries, correlate disparate evidence, and iteratively reconstruct attack narratives. These approaches suffer from scalability bottlenecks, limited integration of external threat intelligence, and a lack of automated reasoning support for complex, multi-stage attack campaigns.

We introduce PROVSEEK, an LLM-powered agentic framework for automated provenance-driven forensic analysis and threat intelligence extraction. PROVSEEK employs specialized toolchains to dynamically retrieve relevant context by generating precise, context-aware queries that fuse knowledge from threat reports with evidence from system provenance data. The framework resolves provenance queries, orchestrates multiple role-specific agents, and synthesizes structured, ground-truth verifiable forensic summaries. By combining agent orchestration with Retrieval-Augmented Generation (RAG) and chain-of-thought (CoT) reasoning, data-guided filtration using a behavioral model, PROVSEEK enables adaptive multi-step analysis that iteratively refines hypotheses, verifies supporting evidence, and produces scalable, interpretable forensic explanations of attack behaviors. PROVSEEK is designed for automated threat investigation without task-specific training data, enabling forensic-style investigation even when no prior knowledge of the environment.

We conduct a comprehensive evaluation on publicly available DARPA datasets, demonstrating that PROVSEEK outperforms retrieval-based methods for the intelligence extraction task, achieving a 34% improvement in contextual precision/recall; and for threat detection task, PROVSEEK achieves 22%/29% higher precision/recall compared to both a strawman agent and State-Of-The-Art (SOTA) *Provenance-based Intrusion Detection System* (PIDS). In our scalability study, we show PROVSEEK increases its token usage by 1.42x and time required by 1.63x when the database size increases 50x making it optimal for large-scale deployment.

## 1. Introduction

Advanced Persistent Threats (APTs) remain one of the most significant challenges facing enterprises and government agencies today. Characterized by their stealthy, multi-step campaigns, APTs achieve persistence by blending into

normal system activity and maintaining long-term access to critical infrastructures [1, 2]. They not only disrupt operations but also inflict severe socio-economic consequences.

Investigating these sophisticated campaigns requires fine-grained visibility into a system’s complete execution history. System provenance has emerged as a powerful foundation for SOTA forensic investigation and intrusion detection [3, 4]. Provenance data encodes rich causal semantics of processes, files, and network interactions, enabling analysts to reconstruct attack chains [5], reason about dependencies, and uncover hidden adversarial behaviors.

Despite this promise, current provenance-based analysis workflows [6–12] face fundamental challenges as analysts are required to craft ad-hoc queries, manually correlate evidence, and iteratively reason about attack progression across billions of events per day [11], which slows investigations. Moreover, forensic tasks demand context-rich, explainable narratives, yet existing pipelines provide little support for automated reasoning or integration of external threat intelligence, which was highlighted by a recent study [4] as well.

A key challenge in provenance-driven security is scalability. Existing PIDS solutions are not capable of handling large datasets, where billions of logs accumulate daily. Several recent attempts have sought to alleviate scalability shortcomings, such as using federated learning (e.g., [13]) or optimizing inference through vector recycling (e.g., [9]). While these methods improve efficiency, they remain limited for forensics: (i) distributed heuristics are prone to false positives, and (ii) centralized ML pipelines still fail to integrate external threat intelligence or provide fine-grained, analyst-friendly forensic explanations.

This gap motivates a new research question: *Can we design a provenance forensics framework that combines scalability, adaptability, and interpretability by orchestrating multiple reasoning steps and role-specific planning through an intelligent agentic architecture?*

To address this question, we propose PROVSEEK, an Large Language Model (LLM)-powered agentic framework for provenance forensic analysis. PROVSEEK employs three LLM-powered agents (e.g., Investigation Agent, Follow-Up Agent, and Safety Agent) as evidence collection and correlation, and reasoning hub that dynamically plans, coordinates, and evaluates investigations. This architecture allows PROVSEEK to ingest unstructured Cyber Threat Intelligence (CTI) reports, extract Indicators Of Compromises (IOCs), use them to formulate database queries, and return contextualized attack summaries with evidence.

PROVSEEK’s agents uses chain-of-thought (CoT) rea-

soning and routes tasks among a set of specialized tools (e.g., retrieval-augmented generation (RAG) over CTI reports, vector-similarity search for semantic retrieval, provenance database queries, query filtration using behavioral model, and correlating evidence from database queries with retrieved CTI artifacts). CoT reasoning helps the agents to iteratively refine hypotheses, validate evidence against provenance data, and synthesizes results into human-readable forensic narratives. PROVSEEK addresses the scalability issue by rethinking scalability as an *agentic orchestration problem*. Instead of considering every event, PROVSEEK leverages an LLM to translate analyst intent or IoCs into *precise database queries*. By retrieving only the relevant events and further filtering them using a behavioral model to identify only the relevant events, PROVSEEK avoids the scalability bottleneck.

Attack campaigns evolve continuously, reusing known Tactics, techniques, and Procedures (TTPs) while adopting new methods to evade detection. PROVSEEK is designed for automated threat investigation without task-specific training data, enabling forensic-style investigation even when no prior knowledge of the environment. PROVSEEK adapts by employing a dynamic query-planning loop: the Investigation Agent understands the security analyst’s intent (investigation, exploration, or identification), the agent issues an initial SQL or vector search, inspects the intermediate results, and the Follow-up Agent iteratively refines the investigation plan. For example, if intermediate evidence suggests lateral movement, PROVSEEK can automatically expand the query to include different system artifacts (e.g., process, files, IPs) used for lateral movement. PROVSEEK’s iterative refinement allows it to adapt its investigation path in real-time, mirroring how a human analyst adapts their line of questioning based on partial evidence.

In security domain, a system’s output is useful if the analysts can verify it with actual system evidence. Therefore, PROVSEEK is built around *verification-first design*. Each agentic step produces not only an answer but also supporting evidence from the system database. PROVSEEK uses the Safety Agent to identify if there is missing or unclear evidence, and it uses the Follow-up Agent to investigate and gather the missing evidence. By design, the framework ensures that every LLM-generated claim is tied to verifiable ground truth in the underlying database, thereby minimizing the risk of hallucination and CoT reasoning errors.

While RAG usage has shown promise in retrieving relevant context, but simply applying RAG to the provenance pipeline introduces new risks. Provenance queries can easily return millions of candidate events, and dumping this unfiltered evidence into a prompt quickly exceeds token limits, forcing truncation and loss of critical context. PROVSEEK addresses this by using a provenance domain-aware SQL query that returns targeted results, and further filters these results using a behavioral model. This yields a compact, high-signal subset of events that PROVSEEK can precisely correlate with the extracted CTI artifacts.

We conduct a comprehensive evaluation using six publicly available DARPA Transparent Computing (TC) and

OpTC datasets. For *intelligence extraction*, PROVSEEK outperforms retrieval-based methods (vanilla and type-filtered RAG) by 34%/34%/30% in contextual precision, contextual recall, and relevance, demonstrating its ability to produce more accurate and context-aware forensic insights. For *threat detection*, PROVSEEK achieves 22%/29% higher precision and recall compared to baselines, including a naive strawman agent and SOTA PIDS [9–11, 14], highlighting its effectiveness in identifying stealthy attacks. For comprehensive evaluation, we also conduct an ablation study to show how the different components of PROVSEEK affect detection, an overhead study, and an error analysis.

This paper makes the following contributions:

- **Agentic Provenance Forensic Framework.** We introduce PROVSEEK, the first agentic framework for forensic provenance analysis that uses LLM agents to automatically plan investigations by unifying provenance-aware querying, CTI-grounded retrieval, and multi-agent LLM reasoning into a single, verifiable, closed-loop pipeline providing analyst-focused explanations.
- **Comprehensive Evaluation.** Using seven publicly available provenance datasets, we demonstrate that in intelligence extraction task PROVSEEK outperforms retrieval-based methods (vanilla and type-filtered RAG) by 34%/34%/30% in contextual precision, contextual recall, and relevance. For threat detection, PROVSEEK achieves 22%/29% higher precision and recall compared to the naive strawman agent and SOTA PIDS [9, 11]. Finally, we conduct an ablation and error analysis study to show how different components of PROVSEEK affect the detection performances.
- **Scalability and Error Analysis.** We conduct a thorough investigation to show the scalability of PROVSEEK as the token usage increases by 1.42x and time required by 1.63x even when the database size increases 50x. In our error analysis study, we show the majority of errors originate from LLM’s limitation of extracting relevant artifacts from threat reports.

## 2. Background

**System Provenance.** System provenance records fine-grained audit data (e.g., syscall, Windows ETW [15] and Linux audit logs [16]) from real-world enterprise environments. It records information flow and control dependencies across system resources (e.g., process, file, socket) [3, 58]. The resulting provenance log offers rich semantics making it invaluable for forensic analysis. Formally, a provenance log  $\mathcal{L} = \ell_1, \ell_2, \dots$  is a sequence of timestamped events (i.e., interactions between system resources). Each resource is associated with attributes such as file paths, IP addresses, or executable names. Each event  $\ell = (u, v, r, t)$  captures a relation  $r$  occurring at time  $t$  between resources  $u$  and  $v$  (e.g., “process  $u$  writes file  $v$ ”). It preserves causal structure while maintaining the chronological order of interactions, enabling analysts to trace attack progression, discover entry points, and assess the scope of compromise. Provenance logs efficiently capture attackers’ intents and TTPs.

### Provenance-based Machine Learning (ML) Research.

Provenance logs were leveraged for manual forensic reconstruction and has become the base for ML-based PIDS [6, 7, 9, 10, 14, 17, 18], adversarial analysis [8, 19] and interpretability [20]. As shown by recent work [4], there is large performance degradation of PIDS against stealthy APT-style attacks, and ML-based PIDS are limited by scalability and interpretability: billions of events per day cannot be efficiently embedded, and black-box predictions without explanations are unsuitable for real-world deployment.

We surveyed representative provenance-based research published in top-tier security venues over the past seven years. These works span detection [4, 6, 7, 9, 11, 12, 14, 17, 21, 22], evasion and robustness [8, 19], differential privacy [23], synthetic graph generation [24] and explanation [20]. All rely heavily on the DARPA TC and OpTC datasets for reproducibility. Therefore, in this work, to enable easy reproducibility, we chose the seven challenging publicly available DARPA datasets where previous SOTA PIDS struggled (*e.g.*, E3 and E5 CADETS, THEIA, CLEARSCOPE, and OpTC datasets).

**LLMs and Provenance Analytics.** Large Language Models (LLMs) have recently transformed domains such as software engineering and social networks [25, 26], largely through techniques such as prompt engineering [27], CoT reasoning [28], tool usage and RAG [29]. In threat intelligence, LLMs have been utilized to extract knowledge graphs from CTI reports [30] and anomaly detection. To the best of our knowledge, we are the first to develop a provenance-domain-aware agentic framework specifically designed for digital forensics. PROVSEEK differentiates itself by employing an *agentic framework* rather than a sole LLM: the LLM is not the sole decision-maker but a reasoning hub that issues bounded queries, retrieves verifiable provenance evidence, and synthesizes narratives.

## 3. Problem Statement and Threat Model

**Problem Statement.** Provenance-based forensic analysis plays a crucial role in investigating stealthy attacks (*e.g.*, APT campaigns). Currently, forensic analysis is an extremely labor-intensive process that requires specialized domain knowledge. As a result, defenders face a critical gap: timely and thorough investigations require scalable mechanisms to retrieve and correlate evidence, while effective response demands interpretability and verifiable explanations.

To illustrate the challenge, consider an enterprise under attack by an APT actor. The provenance logs would contain colossal amounts of irrelevant noise from background benign processes, and stealthy attackers would craft their attacks so that each step, when viewed in isolation, would resemble benign system activity. Conventional forensic workflows require analysts to craft queries manually and iteratively shift through numerous events, which is both time-consuming and error-prone. This results in missing critical connections across attack stages or discovering them after extensive effort.

- Based on Drakon dropper attack artifacts, was I attacked?
- Can you give information about event <EDGE\_ID>?
- What is the source label of edge <EDGE\_ID>?
- What is the label of node <NODE\_ID>?
- What edges involve node <NODE\_ID>?
- What processes interact with file /var/log/mail?

Figure 1: Example Security Analyst Queries

PROVSEEK’s design goal is to provide: (1) scalable query execution to retrieve relevant context, (2) adaptive investigation planning that iteratively refines reasoning based on intermediate results, and (3) fine-grained contextual evidence of adversarial activity. PROVSEEK employs agentic orchestration, where it automatically orchestrates provenance queries, evidence retrieval, and evidence validation to synthesize coherent forensic narratives. This approach strikes a balance between efficiency and analytical depth, enabling rapid and trustworthy forensic investigations.

Let  $\mathcal{L} = \ell_1, \ell_2, \dots$  denote a provenance log consisting of timestamped events, where each event  $\ell = (u, v, r, t)$  describes a relation  $r$  between system entities  $u$  and  $v$  at time  $t$ . Let  $\mathcal{Q}$  represent a set of analysts’ natural language queries (as seen in Figure 1) containing generic or attack specific IOC (*e.g.*, user got from CTI reports). PROVSEEK employs an LLM-based agent planner  $\mathcal{M}$  that decomposes each query  $q \in \mathcal{Q}$  into a sequence of tool calls  $\pi = \langle a_1, a_2, \dots, a_T \rangle$ , where each  $a_t$  corresponds to an operation (*e.g.*, threat-intelligence extraction, investigation, data retrieval, artifact correlation, safety validation, or summary generation). Each intermediate result is verified through a grounding function  $Verify : (a_t, \mathcal{L}) \mapsto \mathcal{L}_t$ , which maps tool outputs back to specific events in the provenance log. The overall objective is to generate an summary  $\mathcal{X} = f(\mathcal{L}_1, \dots, \mathcal{L}_T)$  consisting of (i) an annotated subset of the log  $\mathcal{L}' \subseteq \mathcal{L}$  capturing the relevant evidence, and (ii) a natural language summary aligned with analyst intent.

**Threat Model.** Our threat model assumes the integrity of on-device provenance collection and secure audit pipelines, consistent with prior provenance-based security research [8, 9, 11, 14]. Specifically, we trust the Trusted Computing Base (TCB), which includes the operating system, auditing framework, and provenance storage mechanisms. The integrity of the output provenance data is assumed to be preserved by existing secure logging and tamper-evident provenance systems.

Our work focuses on automating scalable and interpretable provenance forensic analytics through an agentic framework. We do not consider adversarial attacks against the agents itself (*e.g.*, prompt injection or adversarial example generation) as part of our threat model. These remain important open research challenges orthogonal to this work. Instead, our primary focus is ensuring that PROVSEEK’s outputs remain tethered to verifiable ground-truth provenance data, enabling analysts to validate results even in the presence of stealthy APT activity.

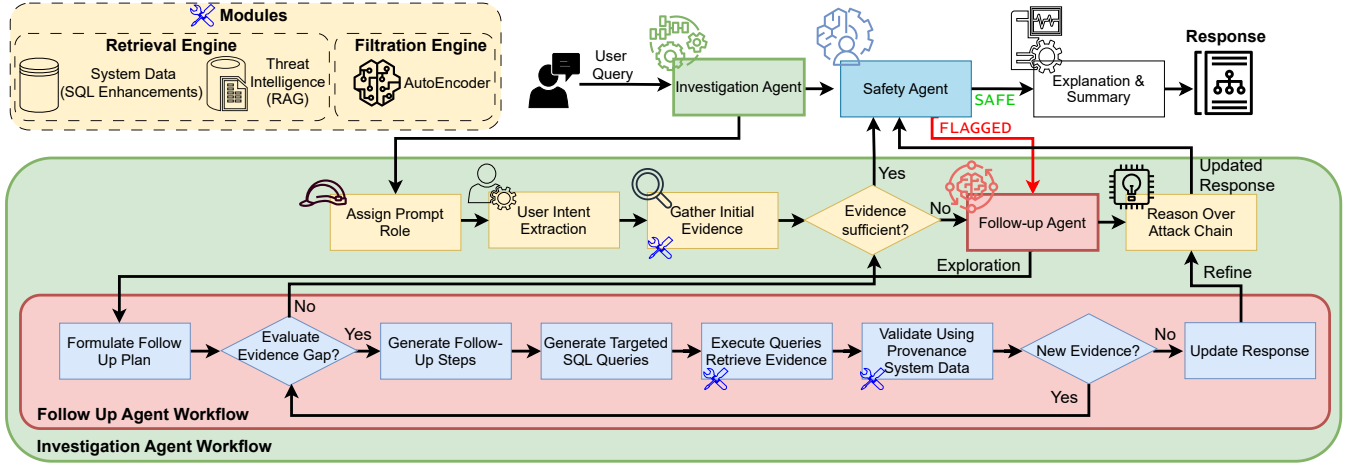


Figure 2: PROVSEEK Agentic Workflow.

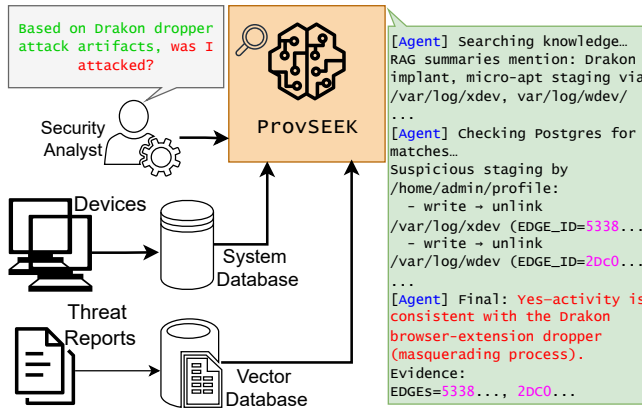


Figure 3: PROVSEEK Overview.

## 4. PROVSEEK: Architecture

PROVSEEK is a LLM-powered threat intelligence extraction and forensic analysis framework as seen in Figure 3, where a user can use natural language to conduct forensic analysis. PROVSEEK expects that it has access to a vector database where CTI reports are parsed and stored, and system provenance databases where system provenance logs are stored. PROVSEEK will extract the intent of the user and conduct investigations until it has gotten concrete evidence from the system database to justify its answer.

PROVSEEK utilizes three main agents: Investigation Agent, Follow-Up Agent, and Safety Agent; and three critical modules that are utilized by the agents: Threat Intelligence Retrieval Engine (using RAG), System Data Retrieval Engine (using SQL enhancements), and Filtration Engine (using AutoEncoder [31]), as seen in Figure 2.

The six main components of PROVSEEK are:

- 1) **Investigation Agent:** Interprets the user’s intent and gathers initial evidence to answer the query. For insufficient evidence, it delegates to the Follow-Up Agent.

- 2) **Follow-Up Agent:** Collects additional evidence to close evidence gaps and reasons over the updated context. After new evidence is extracted and verified, control is handed over to the Safety Agent.
- 3) **Safety Agent:** Ensures that the draft answer contains verifiable evidence. If the answer relies on inconclusive, missing, or unverifiable evidence, it invokes the Follow-Up Agent to gather the missing information. Once satisfied, it summarizes and returns it to the user.
- 4) **Threat Intelligence Retrieval Engine:** Uses RAG to retrieve relevant context from CTI reports and build contextual background for the user’s query.
- 5) **System Data Retrieval Engine:** Executes provenance-aware SQL queries to fetch system-level evidence, ensuring that PROVSEEK does not hallucinate artifacts that cannot be corroborated by system data.
- 6) **Filtration Engine:** Applies a AutoEncoder-based behavioral model to filter out common benign SQL results so that only rare relevant context is passed to the agents when reasoning about the threat scenario.

We will use a motivating security analyst query, “Based on Drakon dropper attack artifact, was I attacked?” to understand PROVSEEK’s components.

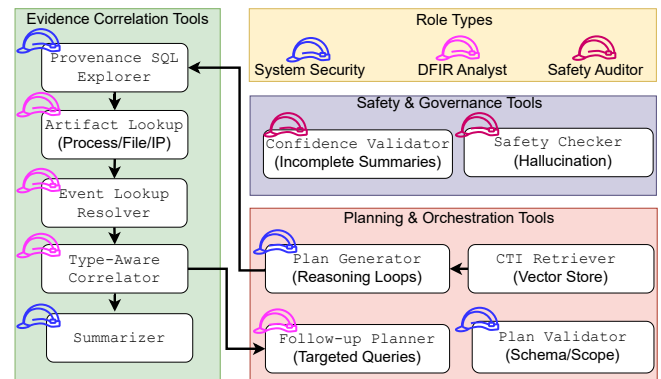


Figure 4: PROVSEEK tools.



## 4.1. Investigation Agent

The Investigation Agent first extracts the intent of the analyst’s query and, based on the intent it gathers initial evidence using different tools [Figure 4](#). These tools can help in *Evidence Correlation* or *Planning & Orchestration*. *Evidence Correlation* tool include: Provenance SQL Explorer, Artifact Lookup, Event Lookup Resolver), and *Planning & Orchestration* tool include: CTI Retriever, Plan Generator). By orchestrating these tools in bounded reasoning loops, the agent ensures investigations remain efficient, precise, and auditable.

PROVSEEK takes on different roles for tools, as shown in [Figure 4](#) and described in [§5.1](#). The Investigation Agent employs differentiated prompt roles that explicitly guide the behavior of underlying LLM. Rather than relying on a single monolithic role, the system frames prompts such as “You are a Digital Forensics and Incident Response (DFIR) analyst” when prioritizing artifact-level correlation, “You are a system security expert” when validating provenance patterns, and “You are a safety auditor for incident-response reports” when reviewing outputs for compliance. This role specialization ensures that the tool is contextually aligned with its intended purpose, using concrete context and improving the interpretability of the generated analysis. The Investigation Agent aggregates the retrieved evidence and correlates them to see if it has enough evidence to answer the user’s query, as seen in [Figure 11](#). If the Investigation Agent could not answer the user’s query because of lack of evidence, it invokes the Follow-up agent.

## 4.2. Follow-Up Agent

Follow-up agent generates structured evidence probing plans when initial evidence is insufficient or new leads emerge, as seen in [Figure 12](#) and uses the following prompt template [Figure 13](#). For example, some artifacts are recovered (e.g., `/var/log/{main/xdev}`) but it is still missing evidence for artifact (e.g., `/var/log/wdev`), it formulates follow-up steps to find those evidences. These plans are bounded in scope, ensuring that exploratory reasoning remains efficient while still progressively expand its investigation scope.

Execution of follow-up plans is handled iteratively, with each step validated against provenance evidence before moving forward. This design ensures that new queries are not only semantically relevant but also grounded in prior results, enabling a logical flow of investigation. Analysts thus benefit from an adaptive system that can pivot to related evidence on demand, reconstruct longer attack chains, and refine hypotheses, all while maintaining accountability and minimizing redundant computation. Once, the Follow-up agent has either found enough evidence to answer the user’s query or have exhausted all its CoT steps it hands over the answer to the Safety Agent.

## 4.3. Safety Agent

Autonomous reasoning introduces risks: such as the answer can have inconclusive evidence or an overly confident answer without having verifiable evidence. The Safety Agent enforces that all the answer contains verifiable evidence required and in case of inconclusive evidence it accurately relays that to the user, as seen in [Figure 14](#).

To further strengthen its operation, the Safety Agent is also assigned an explicit role persona that guides its reasoning as seen in the prompt template [Figure 18](#). Role definition “You are a safety auditor for incident-response automation” direct the agent to critically review investigative actions before execution and further validate the response to decrease the chance of hallucination. In this role, the agent validates query structures, ensures that requests fall within acceptable scopes, and flags potentially unsafe or ambiguous instructions for further scrutiny.

The safety agent can invoke the Follow-up agent when it identifies evidence that needs system justification or identifies an evidence that needs further collaboration as seen in [Figure 15](#) where the Safety Agent flags answer and invokes the Follow-up agent since it is missing Event IDs that are required to verify the evidence. The Follow-up agent gathers the missing evidence and returns the new answer to the Safety Agent that marks the answer safe and returns the answer (as seen in [Figure 5](#)) to the user since it already contains all the information to verify the evidence and the execution trace of PROVSEEK while investigating the query can be seen in [Figure 17](#).

## 4.4. Threat Intelligence Retrieval Engine

Cyber Threat Intelligence (CTI) reports are a rich source of adversarial knowledge, but are presented in unstructured and verbose formats. Analysts often spend significant time sifting through these documents to extract Indicators of Compromise (IOC), leading to delays and inconsistencies in investigative workflows. To alleviate this bottleneck, PROVSEEK integrates a Threat Intelligence Retrieval Engine that applies retrieval-augmented generation (RAG) across an embedded corpus of CTI reports using the prompt template [Figure 19](#). The capability of the engine is dependent on the underlying LLM.

This Engine extracts relevant, complete, and actionable artifacts (e.g., IOCs), separated by OS, process, files and IP as guided by the one-shot example in the prompt template. This translation from unstructured natural language to machine-actionable intelligence reduces cognitive burden on analysts and accelerates the transition from intelligence gathering to forensic validation.

## 4.5. System Data Retrieval Engine

The Data-Retrieval Engine interfaces directly with provenance databases, supporting artifact lookups across various entities and events. Optimized SQL templates handle

#### ProvSEEK Response: Final Assessment & Recommend Actions

##### Conclusion: Compromised (high confidence).

Observed Drakon-aligned TTP: /home/admin/profile writing then unlinking /var/log/mail, preceded by firefox writing to /home/admin/profile. Some auxiliary IoCs were not observed.

**Assessment.** The host exhibits evidence of log tampering and artifact clean-up consistent with MITRE ATT&CK: T1070.004 (Indicator Removal on Host: file deletions of /var/log/mail and transient /tmp/memtrace.so), T1070.002 (Clear Linux System Logs: writes to /var/log/mail immediately preceding deletion), T1574.006 (Hijack Execution Flow: plausible LD\_PRELOAD via transient SO memtrace.so under /tmp), and T1036 (Masquerading: benign-looking executables/paths such as ./gtcache and /home/admin/profile). Overall, the sequence shows staged creation of plausible log/working files followed by removal, aligning with anti-forensic clean-up patterns.

##### Grounded evidence (selected).

```
process(216927) /home/admin/profile → WRITE /var/log/mail (EDGE C284C9F8...)
process(216981) /home/admin/profile → UNLINK /var/log/mail (EDGE 52B5094D...)
process(284573) /home/admin/profile → WRITE /var/log/xdev (EDGE 5338C80F...)
process(216394) /home/admin/profile → UNLINK /var/log/xdev (EDGE 63B9DC37...)
process(216368) /home/admin/profile → WRITE /var/log/wdev (EDGE 2DC0BB3B...)
process(216395) /home/admin/profile → UNLINK /var/log/wdev (EDGE 4850F9E4...)
```

##### Negative indicators (evidence not found).

Files under /deploy/archive/\*, C2 endpoints 104.228.117.212:80 and 5.214.163.155.

**Interpretation.** The WRITE → UNLINK cadence on log-like paths (/var/log/\*) by /home/admin/profile strongly supports anti-forensic behavior (T1070.002, T1070.004). The mention of transient memtrace.so in /tmp is compatible with an LD\_PRELOAD-style hook (T1574.006); the use of innocuous names (./gtcache, /home/admin/profile) aligns with T1036. Absence of known /deploy/archive/\* payloads or the listed C2 IPs in this excerpt limits attribution strength to *medium*.

**Immediate actions:** Isolate host (EDR/VLAN) while preserving forensic access; acquire memory+disk images; preserve /home/admin/profile and /var/log/mail; monitor-first then block suspected egress (141.43.176.203:80, 149.52.198.23:80) via change control.

**Further Notes:** Validate privilege context (writes under /var/log/\*), confirm browser provenance, and sandbox process node IDs 216927/216981.

Figure 5: PROVSEEK final assessment and recommended actions with concrete context.

schema complexity, normalize event records, and collapse duplicates across repeated system activities. This ensures that the evidence retrieved is both accurate and concise, preventing LLM from context explosion and from being overwhelmed by redundant records. By abstracting away the intricacies of database schemas, this engine enables consistent evidence collection across diverse TC datasets. It ensures that each investigative step can reliably ground itself in factual provenance evidence, thereby anchoring the investigation in verifiable system behavior.

A key design choice in the Data Retrieval Engine is the use of event type-specific SQL queries for processes, files, and IPs. Provenance data imposes constraints where the semantics of an entity are tightly coupled with its type: a process may be identified by its command or path, a file by its absolute or relative path, and a network flow by its IP and port tuple. To respect these constraints, the system issues distinct queries tailored to each type rather than relying on a single generalized lookup. This separation reduces ambiguity, ensures semantic correctness, and aligns with the structural guarantees inherent to provenance datasets.

## 4.6. Filtration Engine

Even after the System Data Retrieval Engine issues targeted, provenance-aware SQL queries, the resulting candidate set of SQL events can still contain a large number of benign, high-frequency events (*e.g.*, benign background events). Passing all such results to the agents would both

inflate the context window and increase the risk of false positives. To address this, PROVSEEK incorporates a Filtration Engine that performs rarity-aware post-processing over the retrieved SQL results using an AutoEncoder-based behavioral model [31]. The goal of this component is to down-select to only those events whose behavior is statistically uncommon with respect to benign system activity.

Concretely, each retrieved SQL result, which corresponds to a provenance event, is first embedded into a fixed-dimensional feature vector  $x \in \mathbb{R}^d$  (capturing event type, participating node arguments). The AutoEncoder consists of an encoder  $f_\theta(\cdot)$  and a decoder  $g_\phi(\cdot)$  that are trained jointly on benign, publicly available provenance datasets so as to minimize reconstruction error on common events. During training, the model learns a compressed representation  $z = f_\theta(x)$  and reconstructs  $\hat{x} = g_\phi(z)$ , and the reconstruction loss

$$\mathcal{L}(x) = \|x - \hat{x}\|_2^2 = \|x - g_\phi(f_\theta(x))\|_2^2 \quad (1)$$

serves as a proxy for the rarity of the corresponding SQL event. Intuitively, provenance patterns that frequently occur in benign data are well reconstructed by the AutoEncoder (low  $\mathcal{L}(x)$ ), whereas unusual or attack-like behaviors incur higher reconstruction error.

At inference time, the Filtration Engine computes  $\mathcal{L}(x)$  for each candidate SQL result and assigns an anomaly score  $s(x) = \mathcal{L}(x)$ . A threshold  $\tau$  is selected on a held-out benign validation set, and the filtration decision is given by

$$x \text{ is anomalous} \iff \mathcal{L}(x) > \tau. \quad (2)$$

Let  $\mathcal{C}$  denote the set of all SQL results returned by the Data Retrieval Engine for a given query. The Filtration Engine passes forward only the subset  $\mathcal{A} = \{x \in \mathcal{C} \mid \mathcal{L}(x) > \tau\}$ , ensuring that the downstream agents reason primarily over rare and potentially attack-relevant events while filtering out common, benign provenance patterns.

#### 4.7. Explanation & Summary

PROVSEEK’s Safety Agent uses the summarization tool with the prompt template as seen in Figure 16, to translates structured evidence into human-interpretable narratives as seen in Figure 5, where it not only provides the conclusion and interpretation but also grounded evidence, what evidence it could not find, and the possible immediate actions. This ensures the summaries are concise, actionable, and aligned with analyst needs, producing outputs that highlight causal relationships while maintaining semantic precision. This interpretive layer enhances analyst trust, reduces triage times, and ensures accountability in security workflows.

### 5. Provenance Tools and Considerations

The architecture of PROVSEEK was designed not only as a modular pipeline but also as a resilient system capable of handling the nuanced challenges of provenance-driven security investigations. This section explains the tools and design considerations required for *verifiable foreics*.

Table 1: Summary of PROVSEEK tools.

Name	Description
<b>Evidence Correlation Tools</b>	
Provenance SQL Explorer	Executes provenance-aware SQL queries to extract system-level evidence.
Artifact Lookup	Searches for processes, files, and IPs in provenance databases.
Event Lookup Resolver	Retrieves event for given process IDs to characterize system interactions.
Type-Aware Correlator	Correlates artifacts (e.g., process→file) to reconstruct causal chains.
Summarizer	Summarizes the artifacts explored.
<b>Planning &amp; Orchestration Tools</b>	
CTI Retriever	Performs RAG over a CTI vectorDB to build context regarding IoCs.
Plan Generator	Decomposes analyst queries into structured and reproducible investigation steps.
Follow-up Planner	Formulates new targeted queries when evidence gaps are detected during investigation.
Plan Validator	Enforces schema and scope constraints to ensure generated queries are correct and consistent.
<b>Safety &amp; Governance Tools</b>	
Safety Checker	Validates agent reasoning and enforces operational guardrails.
Confidence Validator	Detects incomplete or speculative summaries and enforces evidence-backed reporting.

#### 5.1. Tools.

The tools in PROVSEEK span three dimensions: *evidence correlation*, *planning & orchestration*, and *safety & governance* as described in Table 1. Evidence correlation tools provide direct interfaces to the provenance databases, abstracting schema complexity and supporting artifact-centric lookups. Planning tools bridge the gap between unstructured CTI reports and structured forensic workflows, enabling agents to generate reproducible and auditable query sequences. Safety and governance tools act as guardrails, validating both inputs and outputs to prevent answers without evidence and mitigate hallucinations. Finally, PROVSEEK integrates *role specialization*, where prompts are assigned professional personas (e.g., DFIR analyst) to align reasoning context with investigative goals.

#### 5.2. Considerations.

**Ground Truth Verification.** A primary challenge in provenance analysis is ensuring that retrieved evidence corresponds to verifiable system-level events. PROVSEEK integrates ground truth verification by enforcing database-backed validation of artifacts before they are sent to analysts. Rather than relying on inferred associations, evidence must be corroborated against system provenance logs. This avoids speculative reasoning and anchors each investigative step in concrete, queryable system behavior.

**Follow-up Exploration.** Another design enhancement concerns follow-up exploration. Investigations rarely end with a single query; analysts often need to refine questions, pivot to related artifacts, or expand the scope of their search. PROVSEEK supports iterative exploration through bounded CoT reasoning loops, where outputs from one stage can seed subsequent queries. This enables the agent to progressively refine hypotheses, recover related processes or files, and map broader attack surfaces without requiring manual re-specification from the analyst.

Investigations often involve joining heterogeneous artifacts, such as linking processes with files and IP connections. Therefore, using an Event-Type Resolver to enforce edge semantics, and a Type-Aware Correlator that links processes→files and processes→IPs without cross-type leakage, PROVSEEK decomposes complex tasks into small manageable tasks. Also, instead of overloading a single agent with complex forensic task, PROVSEEK decomposes tasks across specialized Agents using different engines and tools.

**Hallucination Mitigation.** A recurring risk with LLM-driven agents is hallucination when generated outputs diverge from ground truth. To mitigate this, PROVSEEK anchors reasoning in authoritative CTI reports and provenance logs. PROVSEEK validates every hypothesized artifact against provenance DBs before sending it to the analyst. Concretely, the CTI Retriever proposes candidate artifacts from CTI (e.g., process/file/IP), which are then verified via Artifact Lookup. This tool performs entity-specific

searches with adaptive SQL templates, emitting concrete edge-level evidence (edge id, event type, source/destination labels) rather than free-text matches. Subsequently, Event Lookup Resolver cross-checks the *operation/event\_type* by node id, and Source/Destination Label Resolvers extracts human-readable labels from edge ids. Because artifact hypotheses must resolve to concrete node and edge identifiers, the agent cannot “invent” a process or file without a corresponding row match, and typed validators further prevent cross-type leakage (e.g., a file basename being misinterpreted as a process). This strong coupling forces summaries to be traceable to specific events.

**Query & Token Management.** Large provenance databases introduces a significant operational challenge of query explosion. Certain artifact lookups may return excessive results if not carefully constrained, which can overwhelm the LLM’s context window. To address this, PROVSEEK applies query optimization strategies such as duplicate collapse, type-specific queries, and scoped matching, and further incorporates an AutoEncoder-based Filtration Engine to suppress high-frequency benign patterns before they reach the reasoning layer.

PROVSEEK issues *type-specific* SQL queries where, for files and processes, it searches both *fullpaths* and *basenames*; for IPs, it attempts exact *IP:port* matches before falling back to *IP-only* lookups. Results are deduplicated to prevent inflation of evidence counts and then passed through the Filtration Engine, to discard the results whose reconstruction loss falls below the rarity threshold, effectively filtering out common, routine system activity. These mechanisms keep queries bounded and prevent unnecessary token consumption during the reasoning process by limiting both the number and diversity of events that reach the agents.

In addition, reasoning depth is explicitly bounded: the *Follow-up Agent* conditionally increases the step budget only when prior outputs *explicitly* recommend deeper probing. Follow-up Agent use explicit tool calls for each extracted artifact, ensuring that expansions remain structured, semantically relevant, and focused on statistically rare, potentially attack-relevant events.

**Evidence Verification.** PROVSEEK will provide answers that can be verified using system events. When an answer cannot be fully verified, PROVSEEK will explicitly state which artifact(s) it could not validate and outline potential follow-up steps the analyst should take. In addition, PROVSEEK clearly indicates when its results are inconclusive. Thus, beyond efficiency, verifiability remains a central focus of PROVSEEK.

Collectively, these design details demonstrate how PROVSEEK addresses real operational constraints that undermine many provenance-based systems. By embedding ground truth verification, iterative exploration through bounded queries, and hallucination control, PROVSEEK provides an analyst-aligned investigative platform that is both powerful and trustworthy.

## 6. Evaluation

In this section, we evaluate PROVSEEK’s effectiveness in automated intelligence extraction and forensic investigation. We aim to answer the following research questions:

- RQ1:** How effective is PROVSEEK in extracting actionable artifacts from threat reports (§6.2)?
- RQ2:** How well does PROVSEEK identify threats and summarize post-detection context (§6.3)?
- RQ3:** How much does each component of PROVSEEK contribute to its overall performance (§6.4)?
- RQ4:** How scalable is PROVSEEK (§6.5)?
- RQ5:** What is the impact of different LLM backends on PROVSEEK’s performance (§6.6)?
- RQ6:** What errors are encountered by PROVSEEK (§6.7)?

### 6.1. Experiment Protocol

We first measure PROVSEEK performance in extracting actionable system artifacts (e.g., process, files and IP addresses) from threat reports. We compare PROVSEEK to vanilla RAG pipeline and a type-filtered RAG variant. The vanilla RAG pipeline retrieves artifacts without provenance awareness, often introducing irrelevant or semantically inconsistent evidence. The type-filtered RAG pipeline improves upon this by constraining retrieval to artifacts of the correct provenance entity and event types, but it still lacks reasoning about multi-hop causal relationships. By comparing against these baselines, we *highlight how PROVSEEK’s provenance domain-aware query not only retrieves better artifacts but also captures richer attack context*.

Next, we measure how well PROVSEEK identifies threats and summarizes the post-detection context, so we compare PROVSEEK against multiple baselines spanning both provenance-based Intrusion Detection System (IDS) and Strawman Agent. Specifically, for threat detection, we benchmark against SOTA PIDS Kairos [14], MAGIC [10], Flash [9] and Orthurus [11]. Then, we conduct an ablation study by removing the components of PROVSEEK and measuring its detection accuracy to understand each component’s significance. Next, we study the scalability of PROVSEEK across database sizes to understand how much token usage and latency is expected as database size grow.

Then, to understand the impact of different backend LLMs, we perform an ablation study across different LLMs, GPT-4o, GPT-4.1, and GPT-5, measuring detection accuracy, token usage, cost and time required. Finally, we conduct an error analysis study to understand what kind of errors are encountered by PROVSEEK.

**Datasets.** Our evaluation leverages seven publicly available provenance datasets that capture diverse attack scenarios, including the largest publicly released provenance dataset to date [9] (e.g., DAPRA OpTC dataset). Following prior work [4], we use the most challenging publicly available provenance datasets: DARPA OpTC [32], TC E3 [33] and E5 [34]. E3 and E5 datasets contain multi-day system traces from CADETS, THEIA, and CLEARSCOPE. DARPA OpTC



dataset also contains multi-host device traces. We utilize the ground truth annotations for E3 and E5 datasets from [11]. But for DAPRA OpTC dataset we manually created and verified the ground truth, since they are not publicly available. To evaluate retrieval and identification quality, we additionally employ synthetic Q&A pairs generated using [35], which produces security-specific question-answer pairs from DARPA threat reports tailored for RAG pipelines.

To avoid information leakage, we explicitly exclude benign data from the dataset under evaluation when training the AutoEncoder and calibrating  $\tau$ . That is, for any target provenance dataset used in experiments, its benign events are never incorporated into the training or validation sets of the Filtration Engine. This setting accurately reflects deployment scenarios where the model must generalize to previously unseen systems and workloads.

**Metrics.** To rigorously assess PROVSEEK, we employ a comprehensive set of evaluation metrics spanning intelligence extraction, threat detection, and ablation study tasks. For intelligence extraction, we use *contextual precision* and *recall* to measure whether retrieved evidence fragments are both relevant and complete relative to the analyst’s query. We further measure *relevance*, which captures semantic alignment between generated answers and threat reports, and *faithfulness*, which evaluates whether responses remain grounded in retrieved provenance evidence rather than introducing unsupported hallucinations. These metrics are particularly important in the security domain, where spurious or misleading intelligence can significantly degrade analyst trust and decision-making.

For detection and ablation studies, we measure *precision*, *recall*, *F1-score*, *Token Usage* (for LLM-based approaches), and *Time* by comparing the post-detection summaries against the ground-truth IOC. This allows us to evaluate not only whether threats are identified, but also the relative utilizing of the different components of PROVSEEK. For overhead study, token usage and time is measured against the database size. For the error analysis study, we measure the number of errors produced by PROVSEEK as it attempts to answer the queries. Together, these metrics provide a balanced view of both effectiveness and operational feasibility of PROVSEEK.

## 6.2. Artifact Extraction Performance

PROVSEEK’s threat artifact extraction performance snapshot is shown in Figure 6 measured across DARPA OpTC and E5 datasets (THEIA and CLEARSCOPE); detailed results across all the datasets (e.g., E3, E5, OpTC) are provided in Table 4. Across all the datasets, PROVSEEK achieves 34% improvements in contextual precision and recall compared to vanilla RAG, with corresponding gains in relevance and faithfulness. Even when compared to type-filtered RAG, which is a stronger baseline, PROVSEEK improves by 18%, showcasing the impact of context reasoning. These results show that provenance-guided intelligence extraction not only suppresses spurious associations but also enables richer contextual reasoning about attacks.

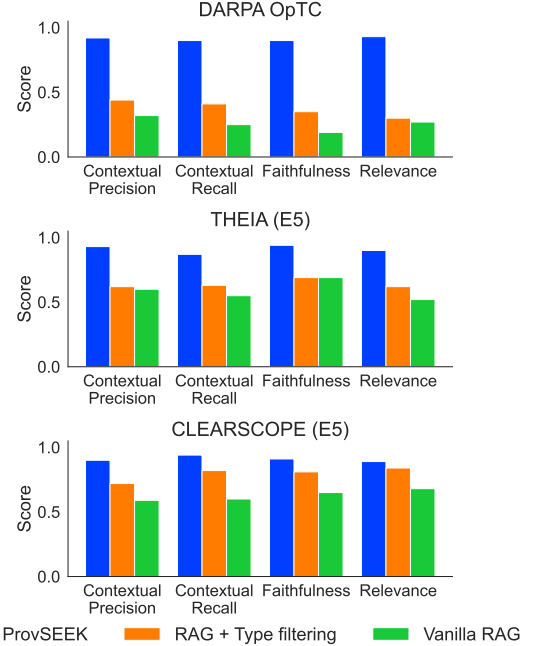


Figure 6: Intelligence extraction performance on DARPA (E5) dataset. (complete results in Table 4).

Both vanilla and type-filtered RAG perform poorly in THEIA with contextual precision often dipping near 0.5-0.6 and worse for DARPA OpTC (going below 0.3). But, PROVSEEK consistently performs well (i.e., precision and recall in the 0.87-0.94 range), demonstrating robustness to complex persistent attack traces. Type-filtered RAG narrows the recall gap on CLEARSCOPE (E5), but fails to maintain precision, highlighting the difficulty of reconciling relevance with correctness without provenance reasoning.

On all the datasets, PROVSEEK sustains both high contextual precision ( $\geq 0.90$ ) and recall ( $\geq 0.91$ ), while baselines fluctuate significantly. In particular, DARPA OpTC shows how naive retrieval pipelines cannot be deployed in the real-world where large CTI reports can mislead relevant intelligence extraction (recall dropping to 0.25 for vanilla RAG), whereas provenance-aware correlation stabilizes results.

## 6.3. Threat Detection Performance

We next evaluate how accurately PROVSEEK detects threats and summarizes them against provenance ground truth. We visualize PROVSEEK’s performance for DARPA OpTC and THEIA E5 dataset in Figure 7 and the comprehensive results for all the datasets are shown in Table 5. PROVSEEK achieves higher F1 by balancing both precision (improving by 22%) and recall (improving by 29%) comparing all the SOTA PIDS across datasets. PROVSEEK consistently outperforms most baselines across datasets or delivers performance comparable to the best-performing baseline. The contrast is especially pronounced on the DARPA OpTC dataset, where most SOTA PIDS struggle.

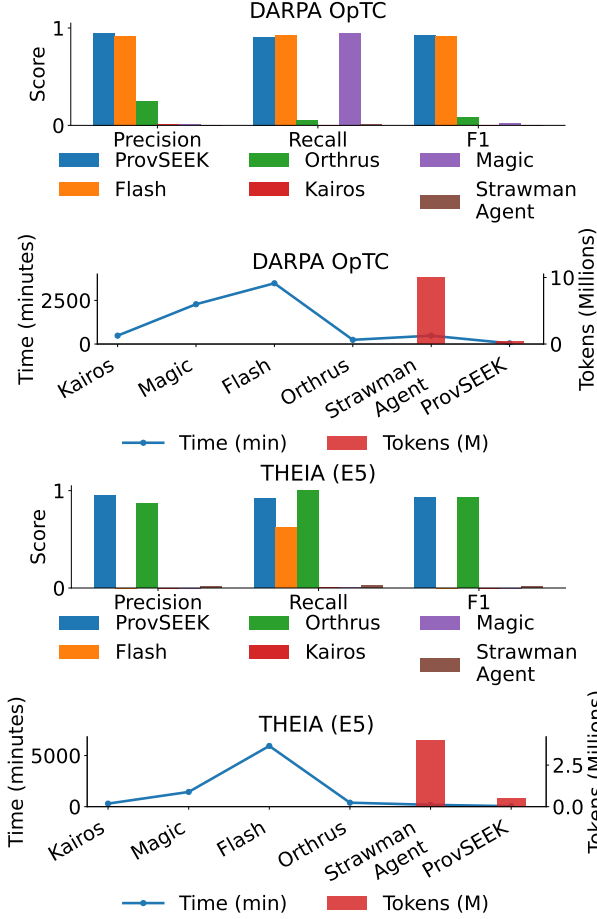


Figure 7: Threat detection performance on DARPA (OpTC and E5-Theia) dataset, (complete results in Table 5).

The Strawman Agent does not provide any benefits because without being able to extract the important attack artifacts (*i.e.*, processes, files and IP) it cannot find them in the system database. Noticeably, the Strawman Agent used 2000% (2M vs 200K) more token compared to PROVSEEK without providing any significant result because of its aimless pursuit of trying to find attack artifacts. The runtime of PROVSEEK is comparable to that of the best-performing PIDS, with most runs completing in under 45 minutes.

While SOTA PIDS either miss most threats or misidentify on irrelevant artifacts, PROVSEEK provides structured summaries that align with ground-truth IOCs. Importantly, these summaries not only flag malicious behaviors but also preserve causal chains, allowing analysts to reason about attack progression. In cases where it could not get the right answer or have inconclusive result as the all step budget is used up, it notifies the user of what artifacts it could not find so that the user can focus their effort into identifying those artifacts or provide better guidance to PROVSEEK. Overall, these results confirm that PROVSEEK is capable of delivering both faithful and operationally efficient detection.

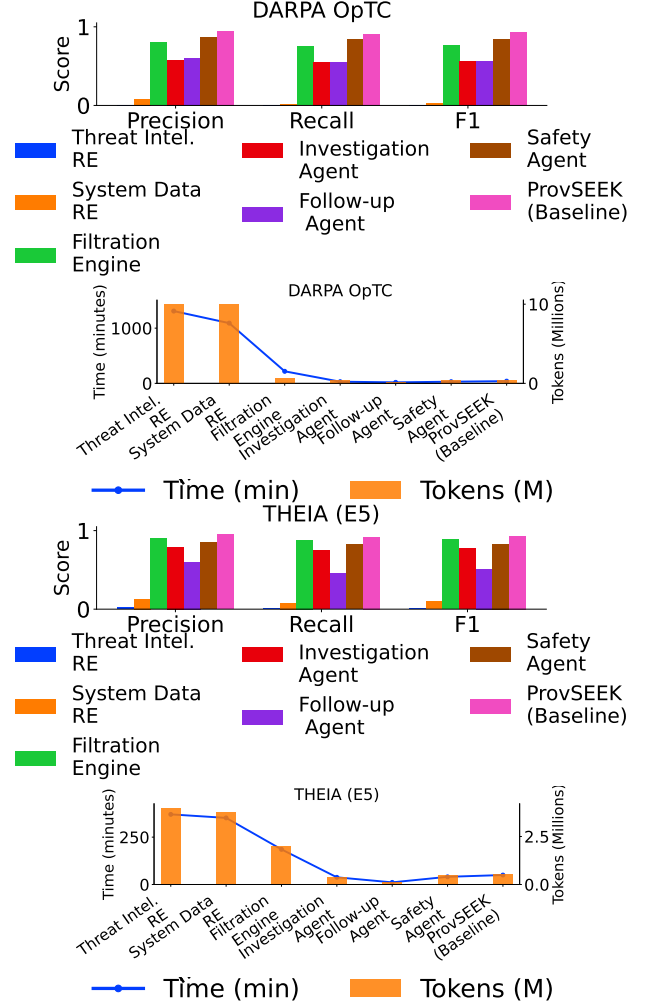


Figure 8: Ablation study of how removing different components affect PROVSEEK's threat detection performance on DARPA (OpTC and E5-Theia) dataset, (complete results in Table 6).

#### 6.4. Ablation Study

We next conduct an ablation study to understand how each component of PROVSEEK contributes to threat detection performance and resource usage. We visualize the ablation results for DARPA OpTC and THEIA E5 in Figure 8, and report the full results across all datasets in Table 6.

The ablation clearly shows that the *Follow-up Agent* is the most critical component for maintaining high detection performance. Removing the follow-up loop substantially degrades both recall and F1, because the system can no longer decompose complex analyst queries into step-wise evidence gathering and iterative hypothesis refinement. For instance, on CADETS (E3), the F1-score drops from 0.89 with full PROVSEEK to 0.55 without the Follow-up Agent (a loss of 0.34), and on THEIA (E5) it falls from 0.93 to 0.51 (a loss of 0.42). Similar drops (*e.g.*, from 0.93 to 0.57 on OpTC) are visible across all datasets in Table 6. At the same time, removing the Follow-up Agent reduces token usage

and latency because the system stops issuing additional targeted probes: token usage shrinks by roughly 300K-400K tokens (e.g., 540K to 122K on THEIA (E5)), and runtime falls from tens of minutes to under 10-20 minutes.

This trade-off highlights that the follow-up loop is where PROVSEEK does the heavy lifting of reasoning over artifacts and closing evidence gaps, and that its computational cost is directly tied to more exhaustive and accurate investigations. Ablation study of the Investigation Agent shows a smaller but noticeable effect: removing the Investigation Agent degrades F1 more mildly (e.g., from 0.93 to 0.77 on THEIA (E5)), because the Follow-up Agent can still try to recover evidence, but the system loses the initial structured plan for correlating artifacts and assembling attack chains.

This is evident in Figure 8, where removing the Threat Intelligence Retrieval Engine (RE) or System Data Retrieval Engine (RE) degrades performance, demonstrating that provenance-aware retrieval is essential for both accuracy and scalability. Without Threat Intelligence Retrieval Engine (RE), PROVSEEK no longer has a principled way to identify which artifacts (files, processes, IPs) are worth querying in the provenance database, and the LLM is forced to “guess” (*i.e.*, hallucinate) from its pre-training which indicators might be relevant. This leads to near-random detection performance while consuming massive resources: on OpTC F1 collapses to 0.00 while token usage explodes to 10M tokens and over 21 hours of compute. A similar pattern appears when we remove the System Data Retrieval Engine (RE): detection remains extremely poor (F1 in the 0.03-0.10 range), but token usage and run time remain nearly as high as the “no-Threat Intelligence Retrieval Engine (RE)” setting (e.g., 10M tokens and 18.18 hours on OpTC). These experiments confirm that naive or unoptimized database querying causes severe context explosion: the agent repeatedly fetches large volumes of low-value events and then spends most of its budget reasoning over irrelevant or noisy artifacts. The combination of CTI-guided RAG and provenance-aware SQL templates in PROVSEEK is therefore crucial: Threat Intelligence Retrieval Engine (RE) focuses the search on semantically relevant IOCs, while the SQL querying enhancements that enforce type- and schema-aware constraints keep the retrieved context bounded and interpretable.

The remaining components: the Filtration Engine, Investigation Agent, and Safety Agent, show a finer trade-off between interpretability and overhead. Removing the Filtration Engine step yields only a modest F1 decline for smaller datasets (e.g., 0.93 to 0.88 on THEIA (E5)), but becomes significant for large datasets like CLEARSCOPE (E5). Removing the Filtration Engine, increases token usage and run time because irrelevant queries are not filtered out and the the system must reason over more raw edges without structured consolidation.

Investigation Agent ablation study, as discussed above, shows that initial investigation planning matters, but its impact is still smaller than that of the follow-up loop, since the Follow-up Agent continues to search for evidence that can support or refute a hypothesis. In contrast, removing the *Safety Agent* results in only a small performance drop (typ-

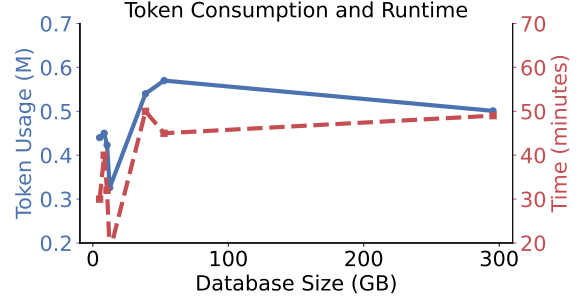


Figure 9: PROVSEEK resource usage with database size.

ically 0.04-0.07 F1), while token usage decreases slightly (e.g., from 423K to 401K on CADETS (E3) and from 570K to 513K on CLEARSCOPE (E5)), and latency changes by only a few minutes. This reflects that the Safety Agent primarily drives extra verification and re-writing passes: it flags over-confident or under-evidenced answers, triggers additional follow-up plans, and enforces evidence-backed reporting. When it is removed, PROVSEEK saves the extra queries and rewrites, but at the cost of weaker guarantees that every claim is backed by explicit database evidence. Overall, the ablation results support our design choice: PROVSEEK’s strongest gains come from the synergy between follow-up reasoning, Threat Intelligence Retrieval Engine (RE), and provenance-aware SQL querying, while the other agents refine the balance between accuracy, robustness, and resource usage.

## 6.5. Scalability of PROVSEEK Study

We evaluate the scalability of PROVSEEK by varying the size of the provenance database and measuring the corresponding runtime and token usage (as seen in Figure 9). As the database size grows from small CLEARSCOPE E3 deployments (4.98GB) up to 50x CADET E5 deployment (295.45GB) the token usage increases by 1.42x and time required per investigation increases by 1.63x because from PROVSEEK’s design: CTI guided RAG narrows the search space to a small set of candidate artifacts, provenance domain-aware SQL enhancements aggressively discard benign edges, and the Filtration Engine further filters out common system events, so the agent reasons over a compact, rare-event based context rather than the full database.

More interestingly, LLM token usage remains tightly concentrated around  $\approx 500K$  tokens across all datasets (ranging from 326K on THEIA E3 to 570K on CLEARSCOPE E5), despite nearly two orders of magnitude variation in database size. As databases get larger, the Follow-up Agent may need to perform additional iterations when RAG extraction is less than optimal. Conversely, in smaller databases, suboptimal RAG behavior can sometimes lead to relatively high token usage because the agent explores a broader fraction of the database when it “does not know what to look for” and chases more speculative artifacts. Overall, these results show that PROVSEEK scales optimally to the largest provenance

database while keeping token usage roughly constant and runtime within tens of minutes, indicating that it is practical to deploy in real-world, large-scale system environments.

Model	P (↑)	R (↑)	F1 (↑)	Token Usage (↓)	Cost (↓)	Time (min.) (↓)
CADETS (E3)						
GPT-4o	0.75	0.75	0.75	276K	\$0.69	20 min
GPT-4.1	0.86	0.92	0.89	611K	\$1.22	44 min
GPT-5	0.90	0.88	0.89	423K	\$0.52	32 min
THEIA (E3)						
GPT-4o	0.78	0.75	0.76	234K	\$0.59	24 min
GPT-4.1	0.96	0.89	0.92	502K	\$1.00	30 min
GPT-5	0.99	0.87	0.93	326K	\$0.40	18 min
CLEARSCOPE (E3)						
GPT-4o	0.79	0.77	0.78	200K	\$0.50	25 min
GPT-4.1	0.91	0.87	0.89	480K	\$0.96	45 min
GPT-5	0.95	0.89	0.92	440K	\$0.55	30 min

Table 2: Impact of different LLMs on PROVSEEK’s performance.

## 6.6. Ablation Study: LLM Backends

We compare three LLM backends: GPT-4o, GPT-4.1, and GPT-5, while keeping the rest of PROVSEEK unchanged (as seen in Table 2). Across all three E3 datasets, GPT-5 consistently achieves the strongest detection performance while maintaining moderate latency and token usage. For example, on CADETS (E3), GPT-5 attains  $F1 \approx 0.89$  using 423K tokens in 32 min, slightly improving F1 over GPT-4.1 ( $F1 \approx 0.89$ ) while using fewer tokens (611K) and lower latency (44 min). On the structurally complex THEIA (E3), GPT-5 reaches  $F1 \approx 0.93$  with 326K tokens and 18 min, compared to GPT-4.1’s slightly lower F1 ( $F1 \approx 0.92$ ) at 502K tokens and 30 min.

Introducing the explicit cost metric reveals that GPT-5 also provides the best accuracy-per-dollar trade-off. Although GPT-4.1 often attains strong recall, it does so at the highest monetary cost and token budget: on CADETS (E3), GPT-4.1 costs \$1.22 per investigation versus \$0.52 for GPT-5; on THEIA (E3), the gap widens to \$1.00 vs. \$0.40; and on CLEARSCOPE (E3), GPT-4.1 costs \$0.96 compared to \$0.55 for GPT-5. In all cases, GPT-5 delivers the highest F1-score while being strictly cheaper than GPT-4.1 and only slightly more expensive than GPT-4o on CLEARSCOPE, yielding the highest F1-per-dollar ratio across datasets. This suggests that GPT-5’s internal reasoning strategy works well with PROVSEEK’s bounded reasoning loops and role-aware orchestration: it can deploy deeper reasoning when necessary while avoiding the runaway token usage seen with GPT-4.1.

GPT-4o remains the most frugal in raw token counts (e.g., 276K on CADETS and 200K on CLEARSCOPE) and often attains the lowest or near-lowest latency (20-25 min), but this efficiency comes at a clear cost in detection quality. Its F1-scores lag significantly behind GPT-5 on every dataset (e.g., 0.75 vs.  $\approx 0.89$  on CADETS, 0.76 vs.  $\approx 0.93$  on THEIA), indicating that simply minimizing token expenditure does not yield effective agent planning or provenance reasoning.

Overall, Table 2 supports our broader insight from §6.2 and §6.3: “thinking” capacity alone is insufficient; models like GPT-5 that can coordinate tool calls, respect PROVSEEK’s role decomposition, and adaptively modulate their depth of reasoning achieve the best joint trade-off between detection accuracy, latency, token usage, and cost.

Failure Reasons	OpTC	THEIA	CADETS	CLEARSCOPE
Total Errors	5/50	8/50	13/50	14/50
Hallucinated artifact	-	-	2/50	-
Benign artifact extracted	1/50	2/50	3/50	3/50
Non-system artifact extraction	2/50	3/50	2/50	3/50
Artifact absent from DB	-	2/50	1/50	2/50
Context explosion	-	-	2/50	1/50
Incorrect correlation	-	-	1/50	3/50
Inconclusive evidence	2/50	1/50	3/50	2/50

Table 3: Failure analysis across DARPA OpTC and E5 datasets. Cells contain counts of failures out of 50 queries per dataset (complete result in Table 7)

## 6.7. Error Analysis Study

We generate fifty user prompts per dataset, twenty-five generic questions that a security response analyst can ask an automated forensics agent (*i.e.*, PROVSEEK) and twenty-five dataset-specific questions, and we manually inspect every failure causes. Table 3 summarizes the distribution of failure modes DARPA OpTC and E5 datasets, (extended results across the datasets are in Table 7. While Figure 20 shows the concrete prompt set for THEIA (E3). The largest single-host datasets, CADETS (E5) and CLEARSCOPE (E5), account for the most failures across almost every category. In particular, they exhibit more cases of context explosion (2/50 and 1/50, respectively), incorrect correlation (1/50 and 3/50), and even the only hallucinated artifacts observed in our study (2/50 on CADETS (E5)). These patterns are consistent with scalability findings: as the underlying provenance data grow, there are more opportunities for the agent to retrieve less relevant artifacts and over-extend its reasoning chain.

A substantial fraction of failures arise from limitations of the underlying LLM in performing domain-specific named entity recognition over system logs and CTI reports. Benign artifact extraction is common across datasets (e.g., 4/50 on CADETS (E5) and 3/50 on CLEARSCOPE (E5)), where PROVSEEK identifies legitimate processes or files that are not actually part of the malicious activity and then spends follow-up steps trying to corroborate them. Non-system artifact extraction shows up at similar rates (e.g., 3/50 for THEIA (E5) and 3/50 for CLEARSCOPE (E5)), reflecting cases where the model mislabels natural-language tokens as system entities. A recurring example is the token `elevate`: the DARPA reports describe red-team activity using Metasploit, which has a command named `elevate`. The LLM frequently misinterprets this as a process or file name, causing PROVSEEK to issue provenance queries for a non-existent



artifact. These errors exemplify that high-quality, system-artifact-aware foundation models, or a dedicated Named Entity Recognition (NER) layer for files, processes, and IPs, remain an open subproblem for provenance-based forensics.

The remaining categories expose a mix of data and orchestration limitations. In several cases, artifacts mentioned in the narrative reports are absent from the actual provenance databases (e.g., 3/50 failures due to “Artifact absent from DB” on CADETS (E5) and 2/50 on CLEARSCOPE (E5)), leaving PROVSEEK unable to validate otherwise correct hypotheses. Context explosion is observed only on the largest dataset (CADETS (E5) and CLEARSCOPE (E5)), typically when non-system or benign artifacts misguide overly broad SQL queries and the Follow-up Agent becomes trapped in a long loop over non-relevant events. This also explains why “Inconclusive evidence” appears across nearly all datasets (e.g., 2/50 on CADETS (E3), 3/50 on CADETS (E5), and 2/50 on CLEARSCOPE (E5)): the agent exhausts its bounded chain-of-thought step budget without accumulating sufficiently grounded evidence to declare a clear attack or benign verdict.

## 7. Discussion

**Dependence on LLM.** A core limitation of PROVSEEK lies in its reliance on LLMs as the reasoning backbone. Despite their impressive ability to parse cyber threat intelligence and orchestrate tool calls, LLMs remain vulnerable to inconsistent reasoning when faced with inconclusive or confusing evidence. In high-stakes security environments, even a small fraction of misleading outputs could hinder investigations or increase analyst workload, showcasing the need for continuous validation of answers against explicit, provenance-backed evidence.

**Incomplete Domain Knowledge Integration.** Another limitation stems from the challenge of embedding deep domain expertise into LLM-driven reasoning. While PROVSEEK incorporates provenance context and role assignment to constrain the LLM’s behavior, the LLM may not fully capture nuanced system semantics, such as mistaking Metasploit commands (e.g., *Elevate*) for actual process names. Unlike seasoned human analysts, LLMs struggle with Named Entity Recognition (NER) from system reports. Developing a dedicated NER layer for LLMs for system artifacts remains an open subproblem for provenance-based forensics.

**Replacement of Human Expertise or IDS.** PROVSEEK reduces analyst burden by automating forensic analysis; it is not a replacement for human expertise or IDS. Security investigations often demand contextual judgment, creative hypothesis generation, and organizational awareness that current LLMs cannot replicate. PROVSEEK should be viewed as a tool that accelerates triage and provides interpretable evidence to augment, rather than replace, expert analysts.

**Adversarial Manipulation.** Evasion attacks against IDS have highlighted vulnerabilities in problem-space manipulation and mimicry strategies [8, 19]. Although PROVSEEK’s verification-first design makes direct hallucination negli-

gible, attackers could attempt adversarial prompt injection to influence investigation planning. Extending PROVSEEK with robustness checks against adversarial manipulation represents an important next step.

## 8. Related Work

**Provenance-based Investigation.** PIDS have become a cornerstone for attack detection [4, 8–14, 17–20, 22–24, 36] and investigation [37–42] support post-hoc attack investigation and incident response for attack triage, correlation, and interactive querying over audit logs and provenance graphs. However, these systems are orthogonal to our research direction: they are not designed for fully automated threat investigation in a zero-training setting. In contrast, PROVSEEK is explicitly designed for automated threat investigation without task-specific training data, enabling forensic-style investigation even when no prior knowledge of the environment.

**LLMs for Security Analytics.** LLMs have been applied across diverse cybersecurity tasks, including software vulnerability detection [43, 44], fuzzing [45], automated patching and vulnerability repair [46], threat detection (e.g., DDoS and phishing) [47], penetration testing [48], and malware reverse engineering [49].

PROVSEEK differentiates itself by employing an *agentic framework*: the LLM is not the sole decision-maker but a reasoning hub that retrieves verifiable provenance evidence, and synthesizes human-readable narratives. By enforcing a verification-first design grounded in system provenance, PROVSEEK complements prior LLM-based security by helping mitigate key scalability and interpretability gaps, while opening a new direction for grounded, agentic security analytics.

## 9. Conclusion

We propose PROVSEEK, an agentic framework for provenance forensics that delivers scalable and interpretable security analytics by automatically fusing provenance evidence into all results. The core of the system is an LLM-based, provenance-aware reasoning hub that orchestrates specialized agents and tools, coordinating tasks like querying, filtering, and CTI integration. This architecture establishes a practical, grounded “verification-first” approach to agentic forensics in real-world environments.

We demonstrate that, for the intelligence extraction task, PROVSEEK outperforms baseline retrieval-based methods by 34%/34%/30% in contextual precision, contextual recall, and relevance, respectively; and for the threat detection task, it achieves 22%/29% higher precision and recall compared to a naive agentic baseline and SOTA PIDS [9–11, 14]. Ablation experiments quantify the contribution of each PROVSEEK component, while a scalability analysis shows that token usage and latency increase by only 1.42 $\times$  and 1.63 $\times$  even as the database size grows 50 $\times$ . An error analysis further reveals that most residual failures stem from

LLM limitations in extracting relevant artifacts from threat reports. Overall, PROVSEEK represents a step toward a new *grounded agentic forensics* paradigm, where LLMs serve not as opaque oracles but as verifiable security tools.

## 10. Ethics Considerations

We have considered the risks and benefits of our research [50] and, to the best of our knowledge, it raises no ethical concerns. All experiments use publicly available, ethically collected datasets that contain no sensitive information.

## 11. Open Science

**Datasets Availability.** The DARPA TC datasets are publicly available [darpa:ground, darpa:ground2] and include textual descriptions of the attacks. For our experiments, we used the preprocessed DARPA TC dataset released by a prior study [4], available at <https://ubc-provenance.github.io/PIDSMaker/ten-minute-install/>. In addition, we used datasets with annotated nodes (labeled as benign or attack-related) provided by another study [11], which are publicly accessible at <https://zenodo.org/records/14641608>.

**Software Artifacts Availability.** The source code is publicly available at <https://anonymous.4open.science/r/provseek-anonymous-DDDD>.

## 12. LLM usage considerations

**Originality.** LLMs were used for editorial purposes in this manuscript, and all outputs were inspected by the authors to ensure accuracy and originality.

**Transparency of LLM usage in PROVSEEK.** PROVSEEK is an agentic framework whose backbone is an LLM. In our implementation, we primarily use models served via the OpenAI API, with GPT-5 as the default backend. For agentic orchestration, we employ smolagents in combination with LangGraph. Tools are implemented using CodeAgent and ToolCallingAgent from smolagents; these wrappers encapsulate LLM calls and can be configured to use any compatible LLM endpoint (e.g., via vLLM or Ollama) without changing the agent logic. In our evaluation, we explicitly evaluate the impact of different LLM backends (GPT-4o, GPT-4.1, and GPT-5) on detection performance, token usage, time, and cost. We did not evaluate older or larger models beyond this set to avoid unnecessary resource usage.

LLMs in PROVSEEK are used for intent understanding, tool planning, and explanation generation; they never directly manipulate system state or modify the provenance databases. All security-relevant conclusions are grounded in explicit system events returned by SQL queries over provenance logs and retrievals from CTI reports. Any ideas or hypotheses proposed by the LLM (e.g., candidate artifacts or investigation paths) are independently validated against the underlying databases via our tools (Artifact Lookup,

Event Lookup Resolver, etc.) before being presented as evidence. A key limitation of our approach is dependency on LLM APIs, which may evolve over time and affect exact reproducibility of reasoning behavior; to mitigate this, we (i) describe the prompts, tools, and agent workflows in detail, and (ii) show that PROVSEEK can operate with different backends, not only a single model.

**Responsibility, data, and environmental impact.** The SQL results consumed by the Safety Agent and Investigation Agent are derived from publicly available, large-scale DARPA-led simulations and provenance datasets, which are widely used in prior work for reproducible evaluation. These datasets do not contain personal user data, and we do not collect or process any additional human subject data for this work. All CTI reports and related documents used for retrieval-augmented generation are publicly released for research use.

To conduct our experiments, we ran PROVSEEK primarily on an NVIDIA L4 GPU for approximately two weeks of clock time (including all datasets and LLM-backend ablations). Using standard methodology for estimating ML carbon impact, this corresponds to an estimated footprint of 7.26 kg CO<sub>2</sub>-eq. We justified this footprint as follows: (i) an LLM-backed agentic architecture is central to our research question of scalable, explainable provenance forensics; (ii) we restricted our study to a small set of modern, efficient LLMs (GPT-4o, GPT-4.1, GPT-5) rather than exhaustively sweeping over many model families; and (iii) we minimized the number of queries and tokens by aggressively filtering results returned from databases. In particular, PROVSEEK translates analyst intent into targeted provenance-aware SQL, applies an autoencoder-based filtration step to discard high-frequency benign events, and terminates further querying once the accumulated evidence is sufficient and verified by the Safety Agent. These design choices limit unnecessary LLM calls and keep token usage roughly constant even as database size scales, thereby reducing both cost and environmental impact while still supporting our methodological goals.

## References

- [1] D. Legezo, *Wildpressure targets industrial in the middle east*, <https://tinyurl.com/mr2n8hdu>, Accessed: April 6, 2023, 2019.
- [2] T. H. Team, *Tortoiseshell group targets it providers in saudi arabia in probable supply chain attacks*, <https://shorturl.at/hmvGT>, (Accessed on 11/09/2021), 2021.
- [3] M. A. Inam et al., “Sok: History is a vast early warning system: Auditing the provenance of system intrusions,” in *IEEE Symposium on Security and Privacy (SP)*, 2023.
- [4] T. Bilot et al., “Sometimes simpler is better: A comprehensive analysis of state-of-the-art provenance-based intrusion detection systems,” in *34th USENIX Security Symposium (SEC)*, 2025.

- [5] E. Hutchins, *Cyber Kill Chain® — Lockheed Martin*, <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>, 2021.
- [6] X. Han et al., “Sigl: Securing software installations through deep graph learning,” in *30th USENIX Security Symposium (SEC)*, 2021.
- [7] Q. Wang et al., “You are what you do: Hunting stealthy malware via data provenance analysis,” in *Network and Distributed System Security Symposium (NDSS)*, Feb. 2020.
- [8] K. Mukherjee et al., “Evading provenance-based ml detectors with adversarial system actions,” in *32nd USENIX Security Symposium (SEC)*, 2023.
- [9] M. U. Rehman, H. Ahmadi, and W. U. Hassan, “Flash: A comprehensive approach to intrusion detection via provenance graph representation learning,” in *IEEE Symposium on Security and Privacy (SP)*, 2024.
- [10] Z. Jia, Y. Xiong, Y. Nan, Y. Zhang, J. Zhao, and M. Wen, “{Magic}: Detecting advanced persistent threats via masked graph representation learning,” in *32nd USENIX Security Symposium (SEC)*, 2024.
- [11] B. Jiang et al., “ORTHRUS: Achieving High Quality of Attribution in Provenance-based Intrusion Detection Systems,” in *34th USENIX Security Symposium (SEC)*, 2025.
- [12] L. Wang et al., “Incorporating gradients to rules: Towards lightweight, adaptive provenance-based intrusion detection,” in *Network and Distributed System Security Symposium (NDSS)*, 2025.
- [13] K. Mukherjee et al., “Proviot: Detecting stealthy attacks in iot through federated edge-cloud security,” in *International Conference on Applied Cryptography and Network Security*, Springer, 2024.
- [14] Z. Cheng et al., “Kairos: Practical intrusion detection and investigation using whole-system provenance,” in *IEEE Symposium on Security and Privacy (SP)*, 2024.
- [15] M. Team, *Event tracing*, <http://tinyurl.com/4usynccm>, 2021.
- [16] Redhat, *The linux audit framework*, <https://github.com/linux-audit/>, 2017.
- [17] J. Zengy et al., “Shadewatcher: Recommendation-guided cyber threat analysis using system audit records,” in *IEEE Symposium on Security and Privacy (SP)*, 2022.
- [18] J. Zeng, Z. L. Chua, Y. Chen, K. Ji, Z. Liang, and J. Mao, “Watson: Abstracting behaviors from audit logs via aggregation of contextual semantics,” in *Network and Distributed System Security Symposium (NDSS)*, Feb. 2021.
- [19] A. Goyal, X. Han, G. Wang, and A. Bates, “Sometimes, you aren’t what you do: Mimicry attacks against provenance graph host intrusion detection systems,” in *Network and Distributed System Security Symposium (NDSS)*, Feb. 2023.
- [20] K. Mukherjee et al., “Interpreting gnn-based ids detections using provenance graph structural features,” 2023.
- [21] F. Yang, J. Xu, C. Xiong, Z. Li, and K. Zhang, “{Prographer}: An anomaly detection system based on provenance graph embedding,” in *32nd USENIX Security Symposium (SEC)*, 2023.
- [22] A. Goyal, G. Wang, and A. Bates, “R-caid: Embedding root cause analysis within provenance-based intrusion detection,” in *IEEE Symposium on Security and Privacy (SP)*, 2024.
- [23] K. Mukherjee, J. Yu, P. De, and D. M. Divakaran, “ProvdP: Differential privacy for system provenance dataset,” in *International Conference on Applied Cryptography and Network Security*, Springer, 2025, pp. 189–219.
- [24] T. Wang et al., “Provcreator: Synthesizing complex heterogenous graphs with node and edge attributes,” *arXiv preprint arXiv:2507.20967*, 2025.
- [25] OpenAI, “Gpt-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023. [Online]. Available: <https://arxiv.org/abs/2303.08774>
- [26] Y. Gao et al., “Retrieval-augmented generation for large language models: A survey,” *arXiv preprint arXiv:2312.10997*, 2023. [Online]. Available: <https://arxiv.org/abs/2312.10997>
- [27] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, “Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing,” *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023. DOI: [10.1145/3560815](https://doi.org/10.1145/3560815) [Online]. Available: <https://arxiv.org/abs/2107.13586>
- [28] J. Wei et al., “Chain-of-thought prompting elicits reasoning in large language models,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 24 824–24 837. [Online]. Available: <https://arxiv.org/abs/2201.11903>
- [29] P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 9459–9474. [Online]. Available: <https://arxiv.org/abs/2005.11401>
- [30] Y. Cheng, O. Bajaber, S. A. Tsegai, D. Song, and P. Gao, “CTINEXUS: Leveraging optimized LLM in-context learning for constructing cybersecurity knowledge graphs under data scarcity,” *arXiv preprint arXiv:2410.21060*, 2024.
- [31] Google, *Intro to autoencoders*, 2021. [Online]. Available: <https://www.tensorflow.org/tutorials/generative/autoencoder>
- [32] DARPA and Five Directions, *Operationally transparent cyber (optc) data release*, <https://github.com/FiveDirections/OpTC-data>, GitHub repository. Accessed: 2025-11-12, 2020.
- [33] DARPA, *Darpa transparent computing (e3)*, <http://tinyurl.com/yaknev56>, 2018.
- [34] DARPA, *Darpa transparent computing (e5)*, <http://tinyurl.com/y4wrf74u>, 2020.
- [35] M. AI, *Synthetic data kit: A toolkit for generating synthetic question-answer pairs*, GitHub repository,



2024. [Online]. Available: <https://github.com/meta-llama/synthetic-data-kit>

- [36] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, “Unicorn: Runtime provenance-based detector for advanced persistent threats,” in *Network and Distributed System Security Symposium (NDSS)*, Feb. 2020.
- [37] T. Pasquier et al., “Runtime analysis of whole-system provenance,” in *ACM Conference on Computer and Communications Security (CCS)*, 2018.
- [38] Z. Xu, P. Fang, C. Liu, X. Xiao, Y. Wen, and D. Meng, “Depcomm: Graph summarization on system audit logs for attack investigation,” in *IEEE Symposium on Security and Privacy (SP)*, 2022.
- [39] P. Fang et al., “Back-propagating system dependency impact for attack investigation,” in *USENIX Security Symposium*, 2022.
- [40] W. U. Hassan et al., “NoDoze: Combatting threat alert fatigue with automated provenance triage,” in *Network and Distributed System Security Symposium (NDSS)*, 2019.
- [41] W. U. Hassan, M. A. Nouredine, P. Datta, and A. Bates, “OmegaLog: High-fidelity attack investigation via transparent multi-layer log analysis,” in *Network and Distributed System Security Symposium (NDSS)*, 2020.
- [42] P. Gao et al., “SAQL: A stream-based query system for real-time abnormal system behavior detection,” in *USENIX Security Symposium*, 2018.
- [43] J. Lin and D. Mohaisen, “From large to mammoth: A comparative evaluation of large language models in vulnerability detection,” in *Network and Distributed System Security (NDSS) Symposium*, 2025.
- [44] S. Ullah, M. Han, S. Pujar, H. Pearce, A. Coskun, and G. Stringhini, “LLMs cannot reliably identify and reason about security vulnerabilities (yet?): A comprehensive evaluation, framework, and benchmarks,” in *2024 IEEE symposium on security and privacy (SP)*, IEEE, 2024, pp. 862–880.
- [45] Y. Oliynyk, M. Scott, R. Tsang, C. Fang, H. Homayoun, et al., “Fuzzing busybox: Leveraging LLM and crash reuse for embedded bug unearthing,” in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 883–900.
- [46] H. Pearce, B. Tan, B. Ahmad, R. Karri, and B. Dolan-Gavitt, “Examining zero-shot vulnerability repair with large language models,” in *2023 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2023, pp. 2339–2356.
- [47] Y. Li et al., “KnowPhish: Large language models meet multimodal knowledge graphs for enhancing reference-based phishing detection,” in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 793–810.
- [48] G. Deng et al., “PentestGPT: Evaluating and harnessing large language models for automated penetration testing,” in *USENIX Security Symposium*, 2024, pp. 847–864.

#### Chat Intro (shown by default)

```
How can I help you?
Some example things you can ask me are:
- Based on the attack artifacts used in the
  privilege escalation attacks, was I attacked?
- Can you give me the source label for this
  event '5338C80F-2428-2415-3146-410000000010'?
- Can you give me label for this node '284573'?
- Can you give me information about this source
  id '658599'?
```

Figure 10: PROVSEEK chat templates used for the intro message.

- [49] P. Hu, R. Liang, and K. Chen, “DeGPT: Optimizing decompiler output with LLM,” in *Proceedings 2024 Network and Distributed System Security Symposium*, 2024.
- [50] T. Kohnno, Y. Acar, and W. Loh, “Ethical frameworks and computer security trolley problems: Foundations for conversations,” in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 5145–5162.
- [51] LangChain, *Langchain: Building applications with llms through composability*, GitHub repository, 2023. [Online]. Available: <https://github.com/langchain-ai/langchain>
- [52] Chroma, *Chroma: The ai-native open-source embedding database*, GitHub repository, 2023. [Online]. Available: <https://github.com/chroma-core/chroma>
- [53] OpenAI, *Openai api*, Accessed: 2025-08-22, 2023. [Online]. Available: <https://platform.openai.com>
- [54] HuggingFace, *Smolagents: Lightweight multi-agent framework for llm orchestration*, GitHub repository, 2024. [Online]. Available: <https://github.com/huggingface/smolagents>
- [55] LangGraph, *Langgraph: Multi-actor applications with llms*, GitHub repository, 2024. [Online]. Available: <https://github.com/langchain-ai/langgraph>
- [56] A. Abid, A. Abdalla, et al., *Gradio: Build machine learning web apps — in python*, GitHub repository, 2023. [Online]. Available: <https://github.com/gradio-app/gradio>
- [57] C. AI, *Deepeval: Open-source llm evaluation framework*, GitHub repository, 2024. [Online]. Available: <https://github.com/confident-ai/deepeval>
- [58] S. T. King and P. M. Chen, “Backtracking intrusions,” in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2003.

## Appendix

### 1. Implementation

PROVSEEK is implemented in python and centers around an agentic architecture built using the LangChain [51] framework for LLM orchestration. The system integrates a PostgreSQL backend database containing the provenance data for structured provenance queries,



a ChromaDB [52] vector store for retrieval-augmented CTI analysis, and OpenAI’s [53] text-embedding-ada-002 model for semantic embedding of threat intelligence.

In addition to the core architecture, PROVSEEK leverages several modern libraries to support orchestration, interaction, and evaluation. For agentic orchestration, we employ smolagents [54] in combination with LangGraph [55]. The tools are implemented using CodeAgent and ToolCallingAgent from smolagents [54]. The analyst-facing interface is implemented in Gradio [56], providing a lightweight web-based UI for interactive investigation and visualization of provenance evidence. PROVSEEK displays Figure 10 template during initialization so that users know what kind of forensic investigations they conduct.

For evaluation, we integrate deepeval [57], which enables systematic benchmarking of agent outputs against ground-truth provenance queries and explanations. To generate diverse evaluation prompts, we further utilize Meta’s synthetic-data-kit [35], which produces realistic question-answer pairs tailored to security scenarios. This pipeline allows us to rigorously test PROVSEEK’s accuracy, interpretability, and robustness under a wide range of analyst-driven investigation tasks.

```
Investigation Agent

[Threat Intelligence Retrieval using RAG]
Summary: Drakon dropper (in THEIA E3) wrote
implant/tools to disk; attempted injections via
/tmp/memtrace.so and /var/log/xdev, /var/log/wdev;
privilege-escalated via /var/log/mail.
OS: [Ubuntu 12.04 x64]
Processes: [profile, gtcache]
Files: [/tmp/memtrace.so, /var/log/mail,
/var/log/xdev, /var/log/wdev]
IPs: [104.228.117.212:80, 141.43.176.203:80,
5.214.163.155, 149.52.198.23]

[Evidence Retrieval and Correlation from System Data]
Scope: [THEIA E3]
edge=C284C9F8-... | proc(216927) /home/admin/profile
EVENT_WRITE → file(513608) /var/log/mail
edge=52B5094D-... | proc(216981) /home/admin/profile
EVENT_UNLINK → file(513608) /var/log/mail
edge=5338C80F-... | proc(284573) /home/admin/profile
EVENT_WRITE → file (658599) /home/admin/xdev
...
```

Figure 11: Investigation Agent Output

## 2. Extended Artifact Extraction Performance

Table 4 reports detailed artifact extraction results for PROVSEEK and different RAG pipelines across all publicly available provenance datasets. These numbers complement the high-level trends discussed in §6.2: PROVSEEK reliably recovers key IOCs (files, processes, IPs) needed for downstream provenance queries, with only modest variation. The table also highlights that stronger RAG pipelines tend to improve contextual recall (fewer missed indicators) without

```
Follow-up Agent

Scope: [THEIA E3]
Artifacts Gathered:
- Processes: /home/admin/profile
- Files: /var/log/mail, /var/log/xdev,
/tmp/memtrace.so
- IPs: 141.43.176.203:80 (hits via firefox/fluxbox),
149.52.198.23:80 (hit by /var/log/mail)
Findings align with files dropped in /var/log/*
misuse pattern and prior IoCs from RAG, but
system interaction related to /var/log/wdev and
/tmp/memtrace.so are missing. Therefore, we should
find them.
New Evidence: [THEIA E3]
edge=2DC0BB3B-... | proc(216368) /home/admin/profile
EVENT_WRITE → file(512304) /var/log/wdev
edge=4850F9E4-... | proc(216368) /home/admin/profile
EVENT_UNLINK → file(512304) /var/log/wdev
edge=01FD23F0-... | proc(216470) /usr/bin/gtcache
EVENT_WRITE → file(512410) /tmp/memtrace.so
...
```

Figure 12: Follow-up Agent Output

degrading precision, which is critical in a security setting where missed IOCs can hide entire attack paths. Overall, Table 4 shows that the CTI extraction stage is robust enough to lead the subsequent provenance analysis with high-quality artifacts.

## 3. Extended Threat Detection Performance

Table 5 provides the full breakdown of PROVSEEK’s threat detection performance, including per-dataset precision, recall, and F1-scores for each SOTA PIDS and Strawman Agent. These results extend §6.3 by showing that the trends we highlight there hold consistently across all DARPA TC datasets: PROVSEEK maintains high precision (low false-alarm rates) while preserving strong recall even on structurally complex datasets such as THEIA and OpTC. The table also makes clear that detectors which better coordinate tool calls and reason over multi-hop evidence yield more reliable security decisions.

## 4. Extended Ablation Details

Table 6 expands the ablation results summarized in §6.4 by providing per-component precision, recall, F1-score, token usage, and runtime across all datasets. The table quantifies the qualitative trends visible in Figure 8: removing RAG or the SQL enhancements severely harms detection accuracy while inflating both token usage and latency, reflecting context explosion over irrelevant events; dropping the Follow-up Agent produces the largest F1 degradation among the orchestration components, underscoring its central role in iterative evidence gathering; and disabling the Safety Agent yields only modest accuracy drops but slightly reduced resource use, confirming that it primarily contributes additional verification and re-writing passes. Taken together,

### Follow-up Investigator Plan (auto-generated)

You are a Digital Forensics and Incident Response follow-up investigator. Execute the following plan using the available tools. Be concise and avoid redundant queries. After your plan execution Finally, call `safety_check` and `create_attack_summary` with all evidence gathered.

Plan:

DB: {db}

Step budget: {followup\_steps}

- investigate process: run `check_artifacts_in_postgres({db}:{process})` (for each extracted process)
- investigate file: run `check_artifacts_in_postgres({db}:{file})` (for each extracted file)
- investigate ip: run `check_artifacts_in_postgres({db}:{port})` (for each extracted IP)
- get edge type → `get_event_type_from_source_id` & `get_event_type_from_destination_id`
- aggregate edge ids → `get_source_labels_from_edge_ids` & `get_destination_labels_from_edge_ids`
- aggregate node labels → `get_process_labels_from_ids` & `get_file_labels_from_ids` & `get_ip_labels_from_ids`
- check is the answer is safe by triggering `safety_check` with the compiled context
- produce `create_attack_summary` with the compiled context

Figure 13: Prompt template for the follow-up investigator to create a tool-grounded plan for evidence aggregation and safety review.

### Safety Agent

status: **SAFE**

Does all claim are backed by edge IDs: **YES**

**Context:** Evidence on THEIA E3 DB of defense evasion via log tampering and masquerading, plus suspicious HTTP egress likely for C2/exfiltration. Firefox wrote `/home/admin/profile`. A process labeled `/home/admin/profile` then wrote and unlinked multiple log-like files (`/var/log/mail`, `/var/log/wdev`, `/var/log/xdev`). Firefox and fluxbox initiated anomalous HTTP connections to 141.43.176.203:80. Critically, a process named `/var/log/mail` initiated HTTP to 149.52.198.23:80, strongly suggesting execution of a masqueraded payload from a log path.

Figure 14: Safety Agent Output

Method	Contextual Precision(↑)	Contextual Recall(↑)	Relevance(↑)	Faithfulness(↑)	Contextual Precision(↑)	Contextual Recall(↑)	Relevance(↑)	Faithfulness(↑)
CADETS (E3)					CADETS (E5)			
Vanilla RAG	0.53 (-0.37)	0.52 (-0.39)	0.59 (-0.32)	0.58 (-0.31)	0.58 (-0.35)	0.69 (-0.25)	0.59 (-0.32)	0.52 (-0.38)
RAG + Type filtering	0.70 (-0.20)	0.69 (-0.22)	0.80 (-0.11)	0.82 (-0.07)	0.63 (-0.30)	0.82 (-0.12)	0.77 (-0.14)	0.63 (-0.27)
<b>PROVSEEK</b>	<b>0.90</b>	<b>0.91</b>	<b>0.91</b>	<b>0.89</b>	<b>0.93</b>	<b>0.94</b>	<b>0.91</b>	<b>0.90</b>
THEIA (E3)					THEIA (E5)			
Vanilla RAG	0.51 (-0.41)	0.58 (-0.31)	0.66 (-0.24)	0.55 (-0.33)	0.60 (-0.33)	0.55 (-0.32)	0.52 (-0.38)	0.69 (-0.25)
RAG + Type filtering	0.71 (-0.21)	0.84 (-0.05)	0.65 (-0.25)	0.63 (-0.25)	0.62 (-0.31)	0.63 (-0.24)	0.62 (-0.28)	0.69 (-0.25)
<b>PROVSEEK</b>	<b>0.92</b>	<b>0.89</b>	<b>0.90</b>	<b>0.88</b>	<b>0.93</b>	<b>0.87</b>	<b>0.90</b>	<b>0.94</b>
CLEARSCOPE (E3)					CLEARSCOPE (E5)			
Vanilla RAG	0.65 (-0.26)	0.53 (-0.41)	0.54 (-0.34)	0.50 (-0.40)	0.59 (-0.31)	0.60 (-0.34)	0.68 (-0.21)	0.65 (-0.26)
RAG + Type filtering	0.76 (-0.15)	0.77 (-0.17)	0.80 (-0.08)	0.82 (-0.08)	0.72 (-0.18)	0.82 (-0.12)	0.84 (-0.05)	0.81 (-0.10)
<b>PROVSEEK</b>	<b>0.91</b>	<b>0.94</b>	<b>0.88</b>	<b>0.90</b>	<b>0.90</b>	<b>0.94</b>	<b>0.89</b>	<b>0.91</b>
DARPA OpTC								
Vanilla RAG	0.32 (-0.60)	0.25 (-0.65)	0.27 (-0.66)	0.19 (-0.71)	-	-	-	-
RAG + Type filtering	0.44 (-0.48)	0.41 (-0.49)	0.30 (-0.63)	0.35 (-0.55)	-	-	-	-
<b>PROVSEEK</b>	<b>0.92</b>	<b>0.90</b>	<b>0.93</b>	<b>0.90</b>	-	-	-	-

Table 4: Intelligence extraction performance on DARPA dataset.

Method	Precision(↑)	Recall(↑)	F1+score(↑)	Token Usage(↓)	Time(↓)	Precision(↑)	Recall(↑)	F1+score(↑)	Token Usage(↓)	Time(↓)
CADETS (E3)						CADETS (E5)				
Kairos [14]	0.01 (-0.94)	0.00 (-0.91)	0.00 (-0.93)	-	8.00 hr (+7.50 hr)	0.00 (-0.85)	0.01 (-0.84)	0.00 (-0.85)	-	14.00 hr (+13.18 hr)
MAGIC [10]	0.01 (-0.94)	0.95 (0.04)	0.02 (-0.91)	-	38.00 hr (+37.50 hr)	0.00 (-0.85)	1.00 (+0.15)	0.00 (-0.85)	-	87.00 hr (+86.18 hr)
Flash [9]	0.92 (-0.03)	0.93 (0.02)	0.92 (-0.00)	-	58.00 hr (+57.50 hr)	0.36 (-0.49)	0.49 (-0.36)	0.00 (-0.85)	-	207.00 hr (+206.18 hr)
Orthrus [11]	0.25 (-0.70)	0.05 (-0.86)	0.08 (-0.85)	-	4.00 hr (+3.50 hr)	0.17 (-0.68)	0.02 (-0.83)	0.04 (-0.81)	-	7.00 hr (+6.18 hr)
Strawman Agent	0.01 (-0.89)	0.01 (-0.87)	0.01 (-0.88)	2M (+1.60M)	2.00 hr (+1.50 hr)	0.02 (-0.83)	0.03 (-0.82)	0.02 (-0.83)	6M (+5.50M)	5.50 hr (+4.68 hr)
ProvSEEK	0.90	0.88	0.89	423.0K	32.0 min	0.85	0.85	0.85	501.0K	49.0 min
THEIA (E3)						THEIA (E5)				
Kairos [14]	1.00 (+0.01)	0.03 (-0.84)	0.06 (-0.87)	-	1.50 hr (+1.20 hr)	0.00 (-0.95)	0.01 (-0.91)	0.00 (-0.93)	-	5.00 hr (+4.17 hr)
MAGIC [10]	0.00 (-0.99)	0.97 (+0.10)	0.00 (-0.00)	-	23.00 hr (+22.70 hr)	0.00 (-0.95)	0.01 (-0.91)	0.00 (-0.93)	-	24.00 hr (+23.17 hr)
Flash [9]	0.00 (-0.99)	0.18 (-0.69)	0.00 (-0.00)	-	12.00 hr (+11.70 hr)	0.00 (-0.95)	0.62 (-0.3)	0.00 (-0.93)	-	99.00 hr (+98.17 hr)
Orthrus [11]	0.52 (-0.47)	0.40 (-0.47)	0.45 (+0.45)	-	45.00 min (+27.00 min)	0.87 (-0.08)	1.00 (+0.08)	0.93 (-0.00)	-	6.50 hr (+5.67 hr)
Strawman Agent	0.05 (-0.94)	0.08 (-0.79)	0.06 (-0.87)	2M (+1.67M)	1.50 hr (+1.20 hr)	0.02 (-0.93)	0.03 (-0.89)	0.02 (-0.91)	4M (+3.46M)	3.00 hr (+2.17 hr)
ProvSEEK	0.99	0.87	0.93	326.0K	18.0 min	0.95	0.92	0.93	540.0K	50.0 min
CLEARSCOPE (E3)						CLEARSCOPE (E5)				
Kairos [14]	0.00 (-0.95)	0.01 (-0.88)	0.00 (-0.92)	-	33.0 min (+3 min)	0.25 (-0.65)	0.00 (-0.85)	0.00 (-0.87)	-	4.00 hr (3.25 hr)
MAGIC [10]	0.00 (-0.95)	0.97 (0.08)	0.00 (-0.92)	-	2.50 hr (+2 hr)	0.00 (-0.9)	1.00 (0.15)	0.00 (-0.87)	-	15.00 hr (14.25 hr)
Flash [9]	0.00 (-0.99)	0.18 (-0.69)	0.00 (-0.92)	-	37.00 hr (+36.5 hr)	0.00 (-0.9)	0.29 (-0.56)	0.00 (-0.87)	-	49.00 hr (48.25 hr)
Orthrus [11]	0.25 (-0.7)	0.05 (-0.84)	0.08 (-0.84)	-	10.0 min (+20 min)	0.33 (-0.57)	0.07 (-0.78)	0.12 (-0.76)	-	4.00 hr (3.25 hr)
Strawman Agent	0.03 (-0.92)	0.07 (-0.82)	0.04 (-0.88)	6M (+5.56M)	5.50 hr (+5 hr)	0.07 (-0.83)	0.01 (-0.84)	0.02 (-0.86)	7M (6.43M)	6.00 hr (5.25 hr)
ProvSEEK	0.95	0.89	0.92	440.0K	30.0 min	0.90	0.85	0.87	570.0K	45.0 min
DARPA OpTC										
Kairos [14]	0.01 (-0.94)	0.00 (-0.91)	0.00	-	8.00 hr (+7.20 hr)	-	-	-	-	-
MAGIC [10]	0.01 (-0.94)	0.95 (0.04)	0.02	-	38.00 hr (+37.20 hr)	-	-	-	-	-
Flash [9]	0.92 (-0.03)	0.93 (0.02)	0.92	-	58.00 hr (+57.20 hr)	-	-	-	-	-
Orthrus [11]	0.25 (-0.7)	0.05 (-0.86)	0.08	-	4.00 hr (+3.20 hr)	-	-	-	-	-
Strawman Agent	0.00 (-0.95)	0.01 (-0.9)	0.00 (-0.93)	10M (+9.5M)	8.00 hr (+7.20 hr)	-	-	-	-	-
ProvSEEK	0.95	0.91	0.93	450.0K	40 min	-	-	-	-	-

Table 5: Comparing threat detection performance of PROVSEEK against SOTA PIDS and Strawman Agent using ground truth IoCs.

Removing	Precision(↑)	Recall(↑)	F1+score(↑)	Token Usage(↓)	Time(↓)	Precision(↑)	Recall(↑)	F1+score(↑)	Token Usage(↓)	Time(↓)
CADETS (E3)						CADETS (E5)				
Threat Intelligence RE	0.01 (-0.89)	0.01 (-0.87)	0.01 (-0.88)	2M (+1.60M)	2.52 hr (+1.90 hr)	0.02 (-0.83)	0.04 (-0.81)	0.03 (-0.82)	6M (+5.5M)	9.78 hr (+8.96 hr)
System Data RE	0.09 (-0.81)	0.02 (-0.86)	0.03 (-0.86)	2M (+1.60M)	2.52 hr (+1.90 hr)	0.10 (-0.75)	0.08 (-0.77)	0.09 (-0.76)	5.50M (+5M)	8.97 hr (+8.15 hr)
Filtration Engine	0.87 (-0.03)	0.82 (-0.06)	0.84 (-0.05)	1M (+577K)	1.26 hr (+43.60 min)	0.82 (-0.03)	0.78 (-0.07)	0.80 (-0.05)	1M (+499K)	1.63 hr (+48.80 min)
Investigation Agent	0.85 (-0.05)	0.85 (-0.03)	0.85 (-0.04)	376K (-47K)	28.40 min (-3.60 min)	0.78 (-0.07)	0.77 (-0.08)	0.77 (-0.08)	437K (-64K)	42.70 min (+6.30 min)
Follow-up Agent	0.60 (-0.30)	0.50 (-0.38)	0.55 (-0.34)	113K (-310K)	8.60 min (-23.40 min)	0.59 (-0.26)	0.44 (-0.41)	0.50 (-0.35)	126K (-375K)	12.30 min (+36.70 min)
Safety Agent	0.87 (-0.03)	0.80 (-0.08)	0.83 (-0.06)	401K (-22K)	30.30 min (-1.70 min)	0.80 (-0.05)	0.78 (-0.07)	0.79 (-0.06)	481K (-20K)	47.00 min (+2.00 min)
ProvSEEK	0.90	0.88	0.89	423.0K	32.0 min	0.85	0.85	0.85	501.0K	49.0 min
THEIA (E3)						THEIA (E5)				
Threat Intelligence RE	0.03 (-0.96)	0.05 (-0.82)	0.04 (-0.89)	1.10M (+774K)	1.01 hr (+42.70 min)	0.02 (-0.93)	0.01 (-0.91)	0.01 (-0.92)	4M (+3.46M)	6.17 hr (+5.34 hr)
System Data RE	0.10 (-0.89)	0.08 (-0.79)	0.09 (-0.84)	2M (+1.67M)	1.84 hr (+1.54 hr)	0.13 (-0.82)	0.08 (-0.84)	0.10 (-0.84)	3.80M (+3.26M)	5.86 hr (+5.03 hr)
Filtration Engine	0.95 (-0.04)	0.82 (-0.05)	0.88 (-0.05)	380K (+54K)	21.00 min (+3.00 min)	0.90 (-0.05)	0.88 (-0.04)	0.89 (-0.04)	2M (+1.46M)	3.09 hr (+2.25 hr)
Investigation Agent	0.87 (-0.12)	0.81 (-0.06)	0.84 (-0.09)	280K (-46K)	15.50 min (+2.50 min)	0.79 (-0.16)	0.75 (-0.17)	0.77 (-0.17)	410K (-130K)	38.00 min (+12.00 min)
Follow-up Agent	0.66 (-0.33)	0.56 (-0.31)	0.61 (-0.32)	150K (-176K)	8.30 min (+9.70 min)	0.60 (-0.35)	0.45 (-0.47)	0.51 (-0.42)	122K (-418K)	11.30 min (+34.70 min)
Safety Agent	0.90 (-0.05)	0.84 (-0.05)	0.87 (-0.05)	400K (+40K)	27.30 min (+2.70 min)	0.85 (-0.05)	0.82 (-0.03)	0.83 (-0.04)	513K (-57K)	40.50 min (+4.50 min)
ProvSEEK	0.99	0.87	0.93	326.0K	18.0 min	0.95	0.92	0.93	540.0K	50.0 min
CLEARSCOPE (E3)						CLEARSCOPE (E5)				
Threat Intelligence RE	0.01 (-0.94)	0.00 (-0.89)	0.00 (-0.92)	6M (+5.56M)	6.82 hr (+6.32 hr)	0.02 (-0.88)	0.03 (-0.82)	0.02 (-0.85)	7M (+6.43M)	9.21 hr (+8.46 hr)
System Data RE	0.07 (-0.88)	0.04 (-0.85)	0.05 (-0.87)	6M (+5.56M)	6.82 hr (+6.32 hr)	0.08 (-0.82)	0.05 (-0.8)	0.06 (-0.81)	6M (+5.43M)	7.89 hr (+7.14 hr)
Filtration Engine	0.87 (-0.08)	0.82 (-0.07)	0.84 (-0.07)	1.2M (+760K)	1.36 hr (+51.80 min)	0.84 (-0.06)	0.80 (-0.05)	0.82 (-0.05)	910K (+340K)	1.20 hr (+26.8 min)
Investigation Agent	0.85 (-0.1)	0.70 (-0.19)	0.77 (-0.15)	378K (-62K)	25.80 min (+4.20 min)	0.77 (-0.13)	0.73 (-0.12)	0.75 (-0.12)	510K (-60K)	40.3 min (+4.7 min)
Follow-up Agent	0.60 (-0.35)	0.50 (-0.39)	0.55 (-0.37)	113K (-327K)	7.70 min (+22.30 min)	0.65 (-0.25)	0.44 (-0.41)	0.52 (-0.35)	130K (-440K)	10.3 min (+34.7 min)
Safety Agent	0.90 (-0.05)	0.84 (-0.05)	0.87 (-0.05)	400K (-40K)	27.30 min (+2.70 min)	0.85 (-0.05)	0.82 (-0.03)	0.83 (-0.04)	513K (-57K)	40.5 min (+4.5 min)
ProvSEEK	0.95	0.89	0.92	440.0K	30.0 min	0.90	0.85	0.87	570.0K	45.0 min
DARPA OpTC										
Threat Intelligence RE	0.00 (-0.95)	0.01 (-0.90)	0.00 (-0.93)	10M (+9.5M)	21.82 hr (+19.82 hr)	-	-	-	-	-
System Data RE	0.08 (-0.87)	0.02 (-0.89)	0.03 (-0.90)	10M (+9.5M)	18.18 hr (+16.18 hr)	-	-	-	-	-
Filtration Engine	0.80 (-0.15)	0.75 (-0.16)	0.77 (-0.16)	700K (+350K)	3.64 hr (+1.64 hr)	-	-	-	-	-
Investigation Agent	0.58 (-0.37)	0.55 (-0.36)	0.56 (-0.36)	400K (-50K)	30 min (-10.90 min)	-	-	-	-	-
Follow-up Agent	0.60 (-0.35)	0.55 (-0.36)	0.57 (-0.36)	120K (-330K)	18.10 min (-22 min)	-	-	-	-	-
Safety Agent	0.87 (-0.08)	0.84 (-0.07)	0.85 (-0.07)	390K (-60K)	30 min (-8.00 min)	-	-	-	-	-
ProvSEEK	0.95	0.91	0.93	450.0K	40 min	-	-	-	-	-

Table 6: Ablation study evaluating how removing each component of PROVSEEK impacts threat detection performance.

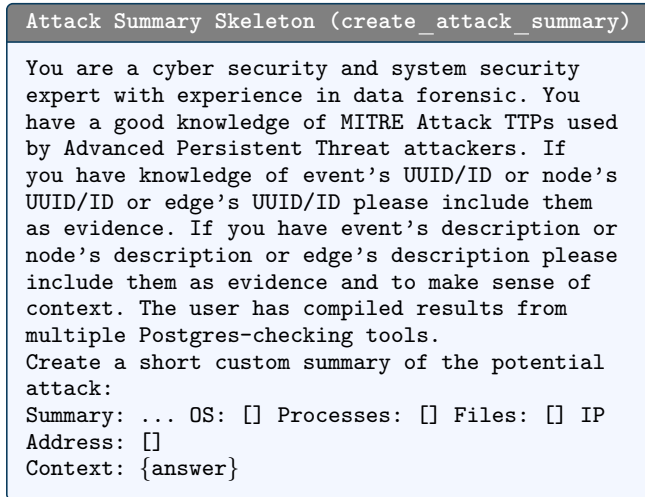


Figure 16: Prompt template for evidence-backed summarization.

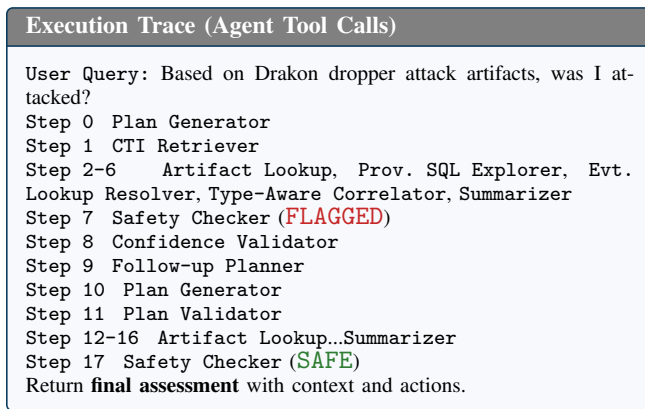


Figure 17: PROVSEEK's execution trace for resolving query.

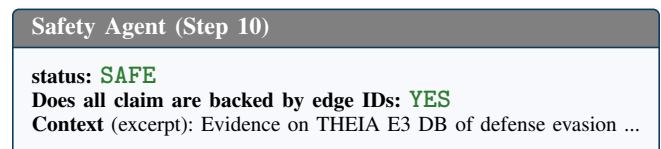
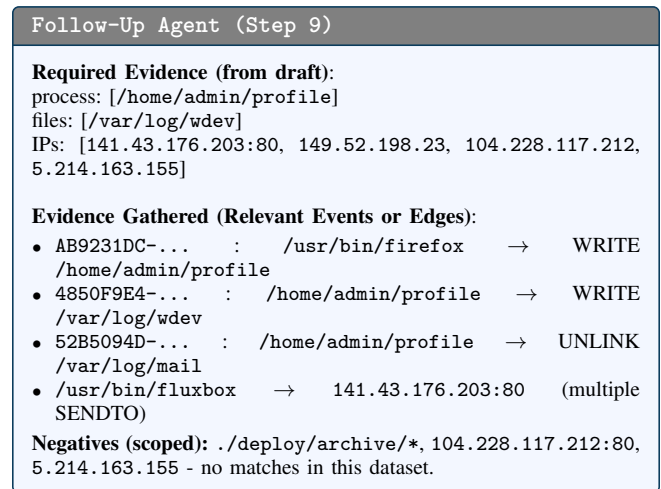
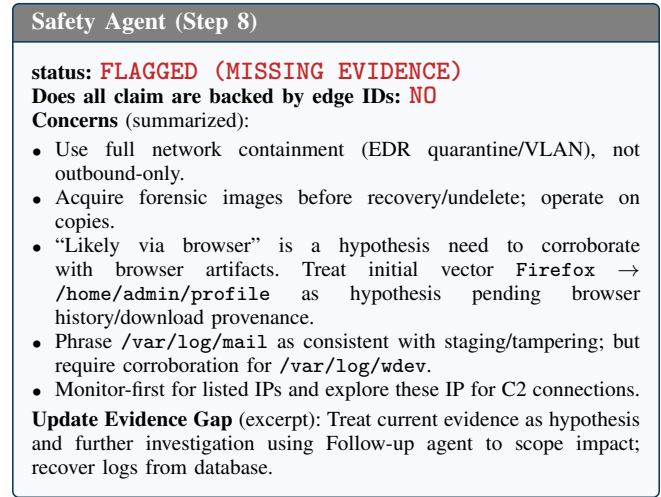


Figure 15: Safety agent flags over-confident answers and answers that do not contain verifiable evidence. It also generates Follow-up investigation plan to bridge the gap.

Table 6 shows that PROVSEEK's design is not redundant, each module contributes a distinct performance gain of that matters in operational deployments.

## 5. Extended Error Analysis

Table 7 reports detailed detail error analysis study across all publically available provenance datasets. Figure 20 visualizes the prompts used for error analysis over 50 representative user queries. We observe that the prompts that cause the most errors cluster around underspecified or highly abstract questions (for example, queries that omit concrete artifacts such as general questions), as well as prompts that implicitly require cross-scenario reasoning beyond the available provenance window.



#### Safety Agent (Prompt)

You are a safety auditor who is needed to generate incident-response reports and triage plans. Your job is to detect overconfidence, unverifiable claims, evidence gap, unsafe instructions, and missing caveats in the current answer. Only approve drafts whose claims are backed by explicit edge IDs and/or node IDs and/or artifacts. Otherwise, mark as flagged and suggest fixes. If flagged, trigger the follow-up agent investigator to close gaps.

- Evidence binding. Each claim about processes/files/IPs should reference concrete edge or node IDs.
- No speculation. Avoid attribution/C2 claims without artifacts; use hedging + next steps if unsure.
- Procedural safety. No risky actions (e.g., tampering live evidence) before imaging/preservation.
- Scope hygiene. DB scope is correct; no querying unrelated databases.
- Data sensitivity. No leakage of secrets/keys/PII; redact if needed.
- Mitigation language. \Monitor-first" vs \block now" is justified with evidence and risk.

If your analysis recommends ‘‘further investigation’’ or ‘‘forensic investigation’’, you MUST call the tool `auto_followup_investigation` with your current draft as `final_answer_text` before you produce your final answer.

Output the answer as: { "status": "SAFE" | "FLAGGED", "concerns": [".."], "suggested\_rewrite": ".." }  
Current Answer: {answer}

#### Safety Investigation Loop (control loop)

Step S1 (Audit): Inspect draft. If missing evidence, set status="flagged" and populate concerns.

Step S2 (Plan): Emit follow-up asks (process/file/IP) needed to clear concerns.

Step S3 (Investigate): Follow-up agent runs: `check_artifacts_in_postgres` → `collect_edge_ids` → `get_source_labels_from_edge_ids`, `get_destination_labels_from_edge_ids` → `create_attack_summary`.

Step S4 (Re-audit): Re-run Safety Agent on revised draft.

Step S5 (Summarize/Plan): If the answer is **SAFE**, then run summarizer tool else plan and investigate.

Typical states:

**SAFE** status="safe" ⇒ return final assessment.

**FLAGGED** status="flagged" ⇒ trigger follow-up plan.

Figure 18: Prompt template for Safety Agent and an example closed-loop handshake with the Follow-up Investigator to turn flagged answers into evidence-backed, verifiable answers.

#### RAG Prompt Template (rag\_with\_reasoner)

You are a cyber security and system security expert with experience in data forensic. You have a good knowledge of MITRE Attack TTPs used by steamlthy actors such as Advanced Persistent Threat attackers, Ransomware and Malware writers. You have read threat reports and are trying to answer questions. Be concise and specific about the process names, ip addresses, and file paths when producing a result. If there isn't sufficient information, provide a better query to improve the RAG results.

Give the answer in the below format:

Answer: Summary: [] OS: [] Processes: [] Files: [] IP Address: []

Example 1:

Question: For the TC5 Theia Attack, can you summarize ‘‘8.6 THEIA - Firefox Drakon APT BinFmt-Elevate Inject’’?

Summary: The attacker gains C2 connections to the target by exploiting Firefox by browsing a website. The attacker then gains root privileges using the installed elevate driver. Further the attacker injected a shellcode (using process injection technique) into the sshd process which wrote a file `sshdlog` into the disk.

OS: [Ubuntu 12.04]

Processes: [sshd, scp, insmod]

Files: [/var/log/sshdlog, read\_scan.ko, load\_helper.ko, /e5/dist/sshd-linux-x64]

IP Address: [128.55.12.167:55430, 128.55.12.167:8108, 128.55.12.110, 189.141.204.211, 208.203.20.42]

Context: {context from vectorDB}

Question: {user query}

Figure 19: Prompt template for the tool using retrieval (RAG).

Failure Reasons	OpTC	THEIA (E3)	THEIA (E5)	CADETS (E3)	CADETS (E5)	CLEARSCOPE (E3)	CLEARSCOPE (E5)
Total Errors	5/50	5/50	8/50	6/50	13/50	3/50	14/50
Hallucinated artifact	-	-	-	-	2/50	-	-
Benign artifact extracted	1/50	2/50	2/50	-	3/50	1/50	3/50
Non-system artifact extraction	2/50	1/50	3/50	1/50	2/50	1/50	3/50
Artifact absent from DB	-	1/50	2/50	2/50	1/50	-	2/50
Context explosion	-	-	-	-	2/50	-	1/50
Incorrect correlation	-	-	-	1/50	1/50	-	3/50
Inconclusive evidence	2/50	1/50	1/50	2/50	3/50	1/50	2/50

Table 7: Failure analysis across datasets. Cells contain counts of failures out of 50 queries per dataset.

#### Generic Queries

1. Based on the system artifacts you see, can you determine whether this host was successfully compromised?
2. For this intrusion scenario, list every file artifact that appears to be linked to the attack.
3. Which file paths in the environment are associated with the suspected dropper and backdoor activity?
4. Enumerate all files that the suspected malicious activity touched (read, write, or delete) on this system.
5. What file nodes (UUIDs and paths) are associated with the suspected incident in the provenance data?
6. Based on the system artifacts related to a potential privilege-escalation attempt, can you tell me if a successful attack occurred?
7. Which log and configuration files were accessed during the suspected compromise window?
8. In this scenario, which files does the suspected malicious component interact with across the system?
9. Retrieve the complete list of file artifacts that appear to belong to this incident.
10. Which files under sensitive configuration and user home directories are associated with the suspected attack?
11. What files tied to the suspected dropper and any follow-on backdoor activity can you find in the ground-truth or labeled data?
12. Identify all file paths that show attacker-like activity (e.g., transient shared objects, log tampering, profile modifications).
13. Which file nodes in the provenance data capture the dropper's actions and the subsequent backdoor behavior?
14. Which files were written, read, or deleted in this case according to the database-backed provenance logs?
15. Provide all file UUIDs, paths, and node sources that appear related to this incident.
16. What file-based artifacts best characterize this compromise from a forensic standpoint?
17. Which browser-related configuration or native-messaging files and related artifacts are implicated in this scenario?
18. List all file artifacts (including temporary directories, log directories, and user profile paths) associated with this attack set.
19. In the labeled ground-truth for this case, which file objects are explicitly marked as part of the attack?
20. Summarize the file-level evidence for this scenario based on the available provenance or database records.
21. In this incident, how did the attacker's processes behave and what process identifiers (PIDs or node IDs) are involved?
22. Which attacker-controlled or suspicious processes (with subject strings) are present in the provenance for this compromise?
23. List all malicious or suspicious process subjects and node IDs linked to the attack in this dataset.
24. Retrieve the browser and helper processes used by the attacker in this scenario, along with their identifiers.
25. How do the dropper and backdoor processes appear in the data (process subjects plus node IDs), and how are they related?

#### Database Specific Queries

26. Which process nodes represent the execution of gtcache and pass\_mgr in the THEIA E3 Drakon case?
27. Show all process subjects under admin/Downloads/firefox that are associated with the THEIA E3 Drakon attack.
28. For THEIA E3, what attacker process behavior is recorded for the Drakon/Firefox incident set?
29. Identify all process UUIDs and subjects that correspond to attacker activity in THEIA E3 Drakon/Firefox.
30. Which processes launched from the malicious Firefox bundle participate in the THEIA E3 Drakon/Firefox compromise?
31. Summarize the attacker's process tree (subjects and node IDs) for the THEIA E3 Drakon/Firefox attacks.
32. What native-messaging host processes are implicated in the THEIA E3 Drakon/Firefox scenario?
33. Which Firefox-related processes (downloads, backdoor, in-memory execution) show up in THEIA E3?
34. In THEIA E3, which processes correspond to the Drakon browser extension executing its payload?
35. Provide all process-level artifacts (subjects and node indices) for the Drakon/Firefox attack in THEIA E3.
36. How does the sequence of Firefox and helper processes look in the THEIA E3 Drakon/Firefox ground-truth data?
37. Which attacker processes are linked specifically to the in-memory Drakon backdoor in THEIA E3?
38. From the THEIA E3 database, what evidence do we have about the lifecycle of attacker processes in Drakon/Firefox?
39. List every process subject string tied to the Drakon dropper and Firefox backdoor in the THEIA E3 attack set.
40. In the THEIA E3 Drakon/Firefox attack chain, which file artifacts appear in the provenance graph?
41. For the DARPA TC3 THEIA-E3 Drakon/Firefox scenario, list all files that were created, modified, or deleted by the attacker.
42. Using the THEIA E3 Drakon/Firefox dataset, which file paths are associated with the malicious browser extension and backdoor activity?
43. From the THEIA E3 Drakon dropper + Firefox backdoor events, enumerate the key log, profile, and shared-object files involved in the compromise.
44. Looking at the THEIA E3 Drakon/Firefox traces, what files in /etc, /var/log, /tmp, and /home/admin are tied to the attack?
45. In the THEIA E3 Drakon/Firefox incident, which attacker-controlled processes (by subject string and internal ID) are visible in the provenance data?
46. For the THEIA E3 Drakon dropper scenario, summarize the process subjects and their node IDs that correspond to the malicious Firefox and native-messaging hosts activity.
47. Within the THEIA E3 Drakon/Firefox attack set, which process nodes show the transition from the downloaded Firefox binary to /usr/bin/firefox?
48. For the THEIA E3 Drakon/Firefox backdoor, identify all attacker processes tied to the gtcache native messaging host and provide their IDs.
49. Using the THEIA E3 Drakon/Firefox provenance, which process subjects and file artifacts together best characterize the attacker's behavior path?
50. Enumerate the key log, profile, and shared-object files involved in the compromise in Drakon dropper + Firefox backdoor events.

Figure 20: Fifty user prompts to invoke different failure case in DARPA Theia E3. Twenty-five user prompts are generic questions that a security response analyst will ask an automated forensics agent (*i.e.*, PROVSEEK) and twenty-five are dataset specific questions.