

Clouseau: A Hierarchical Multi-Agent Approach For Autonomous Attack Investigation

Abdullah Aldaihan
Imperial College London
a.aldaihan23@imperial.ac.uk

Fahad Alotaibi
Imperial College London
f.alotaibi21@imperial.ac.uk

Sergio Maffei
Imperial College London
sergio.maffei@imperial.ac.uk

Abstract—Cyberattack investigations are crucial for understanding the Tactics, Techniques, and Procedures of adversaries, but they face increasing challenges due to the complexity, scale, and volume of modern cyber incidents. Current approaches, such as heuristic-based and learning-based methods, struggle with scalability and reliance on labeled data, leading to difficulties in adapting to new threats. In this paper, we present CLOUSEAU, a hierarchical multi-agent approach that leverages the reasoning capabilities of Large Language Models (LLMs) to autonomously investigate cyberattacks from a single Point-Of-Interest while requiring neither prior training nor predefined heuristics. We evaluated CLOUSEAU on 21 diverse attack scenarios, including complex cases from DARPA’s OpTC engagements and Advanced Persistent Threat (APT) attack scenarios from the ATLAS dataset. In single-host settings, CLOUSEAU achieves an average F1 score of 99.78%, surpassing strong baselines by more than 33%. To demonstrate its broad applicability, we tested Clouseau with both proprietary and open-weight LLMs, achieving strong performance in both cases, thereby enabling deployment in private environments where access to proprietary models is restricted.

Index Terms—Incident Response, Attack Investigation, Digital Forensics, Security Applications of Generative AI, Large Language Models

1. Introduction

When analysts are given a Point Of Interest (POI), such as an abnormal network connection or a suspicious file download, they examine system logs and other telemetry to reconstruct the attacker’s actions. This investigation seeks to discover the root cause, reveal the vulnerabilities the adversary exploited, and measure the incident’s impact. The resulting insights guide both immediate mitigation and longer-term preventive measures [1]–[3]. In practice, such work often falls to seasoned security professionals who know the affected environment, understand its defensive controls, and can reason about likely adversary capabilities [4].

Relying solely on manual analysis, however, is becoming untenable. Contemporary attacks are increasingly intricate and can unfold over months. For example, the SolarWinds

Orion compromise began with a malicious software update in March 2020 and remained undetected until FireEye discovered the backdoor that December [5]. At the same time, malicious activity is buried in torrents of routine telemetry, which, when collected at high fidelity, can reach gigabytes of data per host per incident [6]–[9]. Investigators therefore face a *needle in a haystack* challenge in which a handful of critical events, scattered across long timespans, must be extracted from a vast web of benign activity [2], [8], [10].

Over the past several years, researchers have explored a range of techniques for automating attack investigation, and these efforts coalesce around two broad approaches. The first relies on predefined heuristics distilled from tactics observed in past incidents (e.g., [2], [3]). Such systems filter telemetry by, for example, temporal proximity or data-path relevance in order to surface events connected to a POI. They can work well in practice, yet a skilled adversary may sidestep them by undermining the heuristics’ assumptions. For instance, by “living off the land” and re-using binaries already present on the host, thereby breaking data-path relevance without introducing new executables. The second approach applies behavioral-learning techniques to discover patterns in security data (e.g., [10]–[12]). Although this strategy has produced promising results, it is far from effortless. It demands extensive manual labor to collect, label, and curate large training datasets, and it relies on the assumption that the distribution of training data matches what the model will see in deployment [13], [14]. In security operations, that assumption almost never holds, because benign changes and attacker innovation continuously induce distribution shifts [15]–[18]. As these shifts accumulate, behavioral models may suffer rising false-positive and false-negative rates, forcing practitioners into an expensive cycle of continual retraining to keep the models effective [19].

In this paper, we introduce CLOUSEAU, a hierarchical multi-agent system that automates attack investigation without the need for task-specific training or expert heuristics. CLOUSEAU harnesses the reasoning abilities and built-in security knowledge of general-purpose LLMs to perform end-to-end analyses. It first converts raw incident logs into an LLM-friendly format, then deploys Question-Answering agents that expose the data through a natural-language interface. Investigator agents use this interface to issue focused queries, while a Chief Inspector agent orchestrates their

work, reconstructs the attack narrative, and maps each step to the Cyber Kill Chain framework [20].

We evaluated CLOUSEAU on 21 diverse attack scenarios drawn from two distinct environments. Each scenario was seeded with three POIs, yielding 63 investigations in total. We then compared the results with those produced by two state-of-the-art learning-based systems, ATLAS [10] and AIRTAG [11]. Our results demonstrate that CLOUSEAU is highly effective in reconstructing a detailed and accurate attack story, setting a new benchmark in automated attack investigation and delivering better performance than current baselines by over 33%.

In summary, the contributions of this paper are:

- We demonstrate that LLMs can be orchestrated in agentic workflows to reconstruct complete attack narratives, showing their promise for complex problems in digital forensics and cyber-incident response.
- We design, implement, and open-source CLOUSEAU¹, a hierarchical multi-agent framework for attack story reconstruction from incident logs, requiring no task-specific training data or expert-crafted rules.
- We conducted extensive experiments across various attack investigation scenarios, including complex scenarios with advanced tactics, to validate the effectiveness of CLOUSEAU. In particular, we benchmarked it against two state-of-the-art learning-based attack investigation systems, showing improved performance.
- We extend the ATLAS dataset with synthetic cases that mimic practical challenges such as disconnected attack artifacts, thereby offering a richer benchmark for LLM-driven investigation systems.
- We label and repurpose the comprehensive attack scenarios from DARPA’s OpTC engagements for attack-investigation research, providing an additional high-fidelity benchmark to the community.

2. Background

2.1. Attack Investigation

Attack investigation is a post-detection activity whose goal is to reconstruct the complete chain of malicious behavior so that defenders can understand an adversary’s Tactics, Techniques, and Procedures (TTPs) [4]. The investigation typically starts at a POI and then moves backwards to locate the initial foothold and forwards to measure the attack’s full impact [21]. The raw evidence often comes from kernel-level audit logs, which capture low-level interactions between processes, files, sockets, and other system objects [2], [3], [9], [22], [23]. These records are often enriched with application-level logs (e.g., web-server, database, or container logs) that provide higher-level semantics missing from the kernel view [10], [11].

Because each malicious event is linked to a vast number of benign ones, tracing these connections can trigger a dependency explosion in visualizations or expose a significant

semantic gap when analysts examine the raw records [24]–[28]. Consider a drive-by download: the attacker exploits a browser bug that silently retrieves and executes malware. An investigator can confirm that the rogue executable originated in a particular browsing session, yet the sheer volume of concurrent web activity (tabs, ads, background requests, extensions) can still mask the specific page or domain that delivered the exploit.

2.2. Large Language Models

LLMs, such as the open-weight LLaMA [29] and the proprietary GPT-4 [30], are among the most transformative innovations of this decade. Trained on massive text corpora, these models exhibit professional-level performance across a wide range of general-purpose tasks [31]–[33]. To tackle specialized or complex problems, LLMs can be adapted through techniques such as fine-tuning and in-context learning. Fine-tuning involves updating the model’s parameters with task-specific data, often resulting in strong task performance [34]–[36]. However, it requires high-quality labeled data and can introduce issues such as catastrophic forgetting, where the model loses prior knowledge [37]–[39]. In contrast, in-context learning elicits task-specific behavior at inference time by structuring the prompt with task instructions, examples, or other context, avoiding any parameter updates [40]–[42].

Common prompting strategies include zero-shot, few-shot, and role-based prompting. In zero-shot prompting, the model is provided with only a description of the task, and optionally, a high-level strategy to solve the problem at hand [43]. Few-shot prompting supplements the instructions with a small number of input-output examples to demonstrate the desired task behavior [40]. Role-based prompting guides the model by assigning it a specific identity or expertise (e.g., a security analyst), which has been shown to improve performance in certain domains [44]. These prompting strategies can be composed to implement *LLM agents*, which are autonomous systems that leverage in-context learning, engage in multi-turn conversations, access external resources (in our case, incident logs) through tool-calling, and iteratively refine their reasoning across multiple steps.

3. Motivation

3.1. Challenges

Although prior research in machine-learning has advanced the state of the art, often outperforming heuristic techniques, it still faces three fundamental conceptual challenges.

❶ **High dependency on manual efforts.** There is a need for large, high-quality labeled training datasets to build effective learning models. For strong performance, the datasets must meet two key criteria: representativeness and reliability. Representativeness ensures that the data reflects both the current threat landscape and the normal systems behavior,

1. Prompts for all agents, source code, and evaluation scripts are available at our public repository <https://github.com/ICL-ml4csec/Clouseau>.

while reliability requires that the feature space is carefully constructed to avoid biases or shortcuts (e.g., reliance on specific file names such as ‘malicious.exe’) and that all labels are accurate [45]–[49]. Furthermore, this is not a one-time effort. Datasets need to be continuously updated to address the challenge of distribution shift, a common issue in learning-based security solutions such as malware detection [16], [18], NIDS [50]–[52], and anomaly detection [19], [53]–[55]. This ongoing need for data collection, labeling, and pre-processing involves significant manual effort, making the process resource-intensive, error-prone, and time-consuming.

② Need for precise attack boundaries. Incident logs are used as input to recover the complete attack story, but selecting the right set of logs is challenging. To accurately recover the attack story, one must know exactly when the attack started and ended; without these temporal boundaries, it becomes difficult to determine how far back to trace or when to stop collecting logs. This is particularly problematic in sophisticated attacks that span days, weeks, or even months, where malicious events occur sporadically in large volumes of benign activity. As a result, it becomes difficult to filter out irrelevant logs and accurately capture the full scope of the incident.

③ High false positive rates. If the data boundaries of Challenge ② are not clearly defined, or training data of Challenge ① are not up-to-date, current approaches produce high error rates that complicate the interpretation of their attack narrative outputs. In §6, we analyze two state-of-the-art learning-based baselines to assess their error levels, and were surprised to find that they can generate significant errors even on test data drawn from the same distribution as the training set.

3.2. Our Solution

We now describe how our approach, CLOUSEAU, addresses the challenges described above.

① In-context learning eliminates dependence on labeled data. CLOUSEAU leverages LLMs with in-context learning capabilities, enabling it to analyze and process data dynamically based on the context provided at runtime. Unlike traditional learning-based systems that depend on extensive volumes of labeled training data to identify and classify malicious activities, CLOUSEAU operates without the need for pre-existing labeled datasets. This eliminates the time-consuming processes of manual data collection, labeling, and pre-processing, significantly reducing the effort required for deployment and making CLOUSEAU highly adaptable to diverse and evolving environments.

② Targeted querying of logs removes the need for pre-defined attack boundaries. Learning-based systems require the entire log dataset as input, making it essential to pre-define the attack boundaries—specifically when the attack started and ended. CLOUSEAU eliminates this limitation by producing a targeted query to retrieve events of interest during the investigation. Instead of ingesting the entire incident

logs, it retrieves only the log entries relevant to the POI. For example, if the POI involves a suspicious IP or file, CLOUSEAU queries the incident logs for events connected to the file of interest. This approach ensures that it can handle distributed and prolonged attacks where malicious activities might occur sporadically over an extended period, without needing precise temporal boundaries in advance. By doing so, CLOUSEAU avoids the noise and inefficiency associated with collecting excessive logs while ensuring that critical events are not overlooked.

③ Evidence-driven queries minimize error rates. CLOUSEAU mitigates the high error rates that learning-based systems still suffer from (as shown in §6) through its evidence-driven query mechanism. Rather than relying on pre-trained classification models that are prone to misclassifying benign events as malicious, CLOUSEAU uses iterative and targeted queries based on the context of the investigation. For example, when the POI is a suspicious IP, the system generates precise queries to identify related events, such as connections to other local machines or communications with external servers. Each query is backed by specific evidence retrieved from the log database, ensuring that subsequent questions are contextually relevant and reduce ambiguity. This iterative approach allows us to focus on genuinely suspicious activities while filtering out benign events that traditional systems might incorrectly flag. By doing so, CLOUSEAU significantly lowers the false positive rate, resulting in more accurate and actionable attack narratives.

4. Approach

CLOUSEAU is a hierarchical multi-agent framework that automates end-to-end attack investigations without task-specific fine-tuning or expert-defined heuristics (Figure 1). Every agent runs the same off-the-shelf, general-purpose LLM, leveraging its instruction-following skills and extensive security knowledge through in-context learning. The workflow is split into two phases:

- 1) **Data pre-processing:** Raw incident logs are ingested and normalized into an LLM-friendly, query-able, structured store (§4.1). This store is exposed via a set of Question-Answering (QA) agents, providing a natural-language interface for data retrieval (§4.2).
- 2) **Investigation:** A Chief Inspector agent (§4.4) receives a POI that signals anomalous behavior. It refines this POI into one or more focused investigative directives (*attack leads*) and spawns a dedicated Investigator agent (§4.3) for each lead. Investigators query the QA agents, collect evidence, and return concise reports. The Chief Inspector reviews and merges these findings, and, when new artifacts surface, issues further leads and instantiates additional Investigators. This iterative cycle continues until the full attack narrative is reconstructed.

Scope and Assumptions. We adopt the standard threat model used in recent work [10], [11]. First, the log-collection infrastructure is assumed to be tamper-proof,

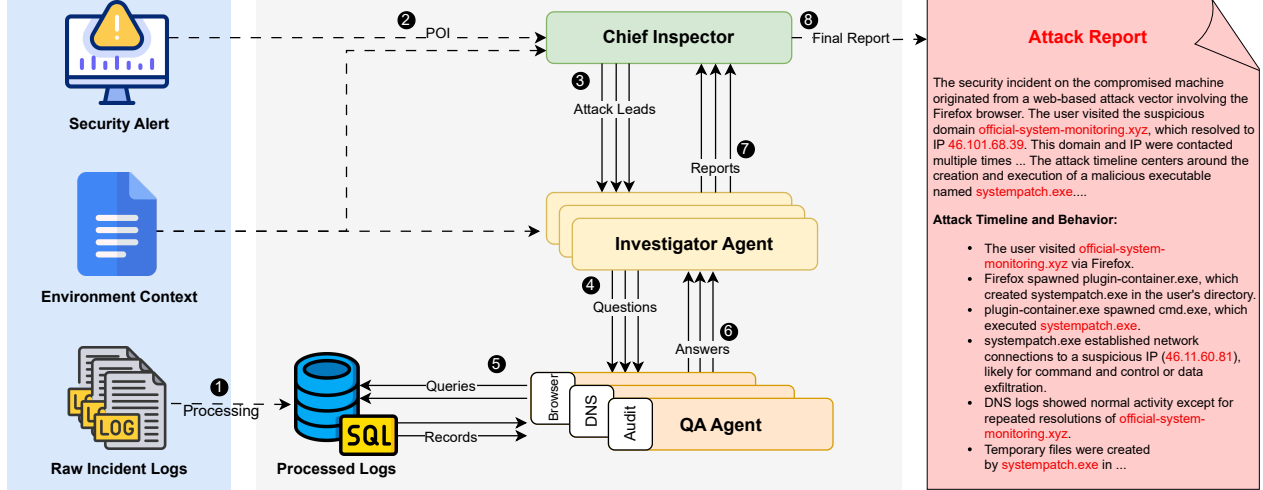


Figure 1: Overview of CLOUSEAU. ① Incident logs are parsed and ingested into a structured database, with a set of QA agents acting as natural-language interface for data retrieval. ② A POI along with the environment context is fed to the Chief Inspector agent, ③ to generate attack leads and activate a series of Investigator agents. ④ Each Investigator agent issues questions to the QA agents to find attack-related events. ⑤ These questions are translated into multi-step SQL queries and returned as simple answers to the original questions ⑥. Each Investigator compiles its findings into a report on the given attack lead ⑦. The Chief Inspector continues the investigation iteratively, and at the end ⑧ compiles the Investigator reports into a single attack report.

so adversaries cannot alter, delete, or fabricate events once they have been recorded. Second, the pipeline applies robust input-sanitization defenses that eliminate prompt-injection attempts against downstream LLM components. Consequently, attacks that compromise the logging system or poison the LLM via crafted inputs fall outside the scope of this paper.

4.1. Data Pre-processing

Data pre-processing is a critical component of CLOUSEAU, enabling the underlying LLM to reason over incident logs efficiently. Raw event logs are typically verbose, redundant, and rarely tailored to security analysis. Our pre-processing pipeline transforms the collected logs into a *structured, query-able* representation that highlights security-relevant signals while suppressing extraneous noise. Our pre-processing pipeline is built on three key components:

① **Environment specific pre-processing.** No two deployments are identical: security configurations, logging policies, and the application stack all vary. A rigid, one-size-fits-all abstraction of logs either omits valuable evidence or overwhelms investigators with irrelevant detail. Consider a single-host laboratory where every action is executed by an all-privileged user; recording every authentication attempt or privilege escalation adds little value. Conversely, in a multi-tenant enterprise, the same events are indispensable for detecting lateral movement. Recognizing this, we avoid

prescribing a single pipeline. Instead, §5.1 illustrates multiple pipelines demonstrating how CLOUSEAU can be adapted to different deployment contexts.

② **Structured data transformation.** To facilitate efficient data retrieval, we first convert raw incident logs into a structured, query-able format. Recent general-purpose LLMs have shown strong performance in automatic SQL generation and schema reasoning [56], [57]. We leverage this capability by storing the logs in a relational database. Each log source is mapped to one or more SQL tables (for dense, heterogeneous sources), or, conversely, a single table can merge multiple log sources, yielding an n -to- m relationship, where n is the number of log sources and m is the number of tables. During an investigation, we retrieve the schema of the relevant tables and add them to the prompts of QA agents. To improve an agent’s understanding of the schema, and therefore its query accuracy, we enrich the database metadata (‘CREATE’ statements) with human-written annotations (similar to code comments) describing the table’s data.

③ **Demonstrations and contextual descriptions.** Each table is accompanied by a few hand-crafted examples that translate natural-language questions into valid SQL. During the investigation, the QA agent is augmented with these examples as in-context demonstrations, maximizing accurate query synthesis. Because every incident is unique, we also require an *environment context* document capturing site-specific invariants (IP ranges, allowed services, authentication policies, *etc.*). This document is added to the prompts of the Chief Inspector and Investigator agents to give the

LLM the context it needs to flag anomalous behavior reliably.

4.2. Question Answering Agent

An integral component of CLOUSEAU is the QA agent. Its purpose is to provide the Investigator with a natural-language interface for retrieving relevant log events without being burdened by schema-specific details or SQL formulation. Through this question-answering interface, the Investigator agent simply asks a question. The QA agent then iteratively retrieves relevant events and returns an answer. Furthermore, to minimize hallucinations, improve accuracy, and reduce the risk of malformed queries, we instantiate a dedicated QA agent for each SQL table produced during pre-processing.

Agent Design. The QA agent follows the ReAct paradigm [41]. We implement it with in-context few-shot [40] learning and role-playing [44], prompting the LLM to think, select appropriate tools, and reflect on results. The agent’s role is defined as an ‘SQL expert’ whose task is to answer questions by querying the database (full template in Appendix A). The template is augmented with the table schema in the form of the ‘CREATE TABLE’ statement, including embedded annotations that describe each column’s semantics. We also include a set of handcrafted exemplars that map questions to sequences of SQL queries leading to the final answer. To improve reliability, the agent is instructed to decompose questions into multiple simple queries rather than constructing a single complex one, reducing errors and malformed queries.

In the case of ATLAS scenarios, this instantiation yields three QA agents: an ‘audit’ agent, a ‘browser’ agent, and a ‘DNS’ agent. Each agent is equipped with one generic tool to execute queries, ‘run_query(sql)’, and can optionally be augmented with a set of table-specific tools (akin to stored procedures). For example, with the ‘audit’ agent we define two additional tool calls: ‘get_ancestors(pid)’ and ‘get_descendants(pid)’. Although the ‘audit’ agent is able to find ancestors or descendants of a process with iterative querying, this often repeated task resulted in the overuse of expensive LLM tokens. As such, we found the use of stored procedures as tool calls to be quite effective for repetitive and token expensive tasks.

Memory Management. Managing the context size of LLMs is crucial: unbounded growth leads to increased hallucination rates and degrades overall performance [58], [59]. To manage the agent’s context growth, we cap the number of queries the agent may issue and abort any query that returns more than R rows. This mechanism bounds the agent’s context length by

$$len \lesssim |prompt| + max_queries \times R + |answer|$$

where R is the per-query row cap after serialization, and $max_queries$ is the maximum number of SQL queries the agent may issue. These parameters are configurable and can be tuned to the model’s context window and the deployment

environment. If a result exceeds R rows, the tool returns an error prompting the agent to refine the query (e.g., add ‘LIMIT’ or ‘GROUP BY’, or introduce additional predicates). If the agent exhausts its query budget without producing an answer, it is instructed to answer based on the information gathered so far.

Example. Assume an Investigator observes a suspicious process ‘systempatch.exe’ during the investigation. To investigate the process, it invokes the ‘audit’ QA agent with the following task:

Audit QA Agent: Question

Find the execution tree of systempatch.exe (PID 2592) and list all associated network connections and files.

The schema for the audit table is first fetched, along with questions-to-queries examples, to construct the agent prompt and create an instance of the ‘audit’ QA agent. Guided by the given directive, the agent executes a series of tool calls:

Audit QA Agent: Tool Calls

```
get_ancestors(2592)
get_descendants(2592)
run_query(SELECT DISTINCT object FROM
  ↳ audit_logs WHERE access = 'CONNECT' AND
  ↳ pid = 2592)
run_query(SELECT DISTINCT object FROM
  ↳ audit_logs WHERE access != 'connect' AND
  ↳ access != 'execute' AND pid = 2592)
```

With every tool invocation, the agent incrementally builds its answer. The fourth call, however, returns too many records and triggers a result-length error. The agent then reformulates the query to fetch only the 20 most-accessed files. Once the revised results arrive, the agent synthesizes the final answer:

Audit QA Agent: Answer

The execution tree for the process named systempatch.exe with PID 2592 is as follows:

- PID 3148, firefox.exe, [timestamp]
- PID 2012, plugin-container.exe, [timestamp]
- PID 744, cmd.exe, [timestamp]

```
...
[processes connections]
[Files accessed]
...
```

4.3. Investigator Agent

The Investigator agent is at the heart of CLOUSEAU, charged with uncovering attack-related events and artifacts. Each run begins with an *attack lead*, a focused directive that flags a specific anomaly and defines the scope of the inquiry. Guided by this lead, the Investigator queries the available QA agents, iteratively retrieving evidence and refining its

hypotheses. As answers accumulate, the attack narrative gradually takes shape. When no further avenues remain (or the question budget is reached), the Investigator compiles a concise report and returns it to the Chief Inspector.

Agent Design. Mirroring our QA agents, the Investigator adopts the ReAct paradigm. Its prompt (Appendix A) defines the agent as a ‘security expert’ tasked with conducting security investigations. The agent is instructed to formulate precise, context-rich questions to reveal attack-related artifacts. To aid the agent in identifying related attack artifacts, it is instructed to employ ‘backward’, ‘forward’, and ‘correlation’ analyses; it is also instructed to report any abnormal activity, even if it lies outside the current lead, and conclude each investigation with a concise summary of its findings. Similar to the QA agents, its question budget is capped at *max_questions* tool calls, acting as a means to control the LLM’s context growth.

The Investigator prompt is augmented with an *environment context* defined earlier during the pre-processing phase to aid during discovery, and a set of tools encapsulating specialized QA agents, each dedicated to a database table as described earlier (e.g., DNS, browser, audit). Every tool is accompanied by a description of its data scope and includes few-shot examples illustrating valid queries. Finally, the current attack lead is appended to the agent prompt.

The investigation then proceeds iteratively: the agent invokes QA tools, accumulates evidence, incrementally refines its findings, and discovers new leads along the way. This cycle continues until all viable hypotheses are exhausted or the *max_questions* budget is reached.

Example. Assume the Chief Inspector observes an unrecognized binary ‘systempatch.exe’ in the latest Investigator report. The Chief Inspector agent dispatches a new Investigator with the following attack lead:

Investigator Agent: Attack Lead

Investigate the process systempatch.exe (PID 2592). Construct its execution tree, enumerate associated files and network connections, and flag abnormal behavior observed around its start time.

The Investigator first queries ‘audit’ agent for the execution tree and network connections made by this process. The response reveals an outbound connection to ‘46.11.60.81’. The agent then queries ‘dns’ agent to resolve domains mapping to this IP, and queries ‘audit’ again for any other processes communicating with it. This cycle continues until no further avenues remain or the question budget is depleted. The agent finally generates a report of its findings; for example:

Investigator Agent: Attack Lead Report

The following attack-related artifacts have been identified:

- Process names: systempatch.exe, firefox.exe ...
- Network addresses: 46.11.60.81 ...
- Domains: No domains resolved to 46.11.60.81.

[more findings] ...

The report is returned to the Chief Inspector, and this instance of the Investigator agent is destroyed. The Chief Inspector parses this report and may decide to instantiate a new Investigator with a new lead based on the findings.

4.4. Chief Inspector Agent

At the helm of our framework is the Chief Inspector agent, which adopts the same ReAct architecture, with few-shot in-context learning and role-playing prompting, as in the Investigator and QA agents. It serves two key functions. First, it directs the overall investigation, repeatedly dispatching Investigator agents and refining their tasks as new evidence emerges. Second, it acts as a safeguard against hallucinations: by comparing overlapping findings from different Investigators, the Chief Inspector can detect and suppress inconsistent or fabricated conclusions.

Its prompt (Appendix A) defines the agent as a ‘security expert’, tasked with identify the attack source, establish the attack timeline, identify attack objectives, and finally map the attack to the Cyber Kill Chain. After this, the agent is given general instructions, to clarify its behavior, including the assertion that this is an iterative process and an instruction to investigate thoroughly. The final directive is to produce a comprehensive report that conforms to the given task. The agent is provided with the environment context (the same one that is supplied to Investigator agents) and given the initial POI.

Example. Building on the earlier examples, the Chief Inspector is initialized with a POI indicating frequent HTTP connections to suspicious domain ‘official-system-monitoring.xyz’ and is tasked with investigating this activity. It instantiates an Investigator with the following lead:

Chief Inspector Agent: POI

Investigate the domain official-system-monitoring.xyz. Identify every process that connected to it and highlight any abnormal activity in the same time window.

The Investigator consults its QA agents in turn. It first resolves the domain to the IP address ‘46.101.68.39’, then retrieves every connection event that references this address—knowing that the logs store network endpoints by IP rather than by domain. Nearing its question budget, the Investigator makes one final request to the audit QA agent to construct the execution tree of each connecting process. This last step surfaces a suspicious binary, ‘systempatch.exe’, that appears central to the intrusion. The Investigator compiles the following report and returns it to the Chief Inspector:

Investigator Agent: Attack Lead Report

... The execution tree of firefox.exe revealed a suspicious process systempatch.exe, which was spawned by cmd.exe, a child process of plugin-container.exe. This suggests potential malicious activity, as systempatch.exe is not a standard Firefox or Windows process. The investigation also found that the domain official-system-monitoring.xyz resolved to the IP address 46.101.68.39, which was connected to by the firefox.exe and plugin-container.exe processes. Further analysis is recommended to determine the nature and intent of the systempatch.exe process and to investigate any potential security breaches or malicious activities related to the identified artifacts.

Following the first Investigator’s findings, the Chief Inspector instantiates a new Investigator with a lead on the suspicious process ‘systempatch.exe’. This second Investigator uncovers an additional attack artifact, a network address repeatedly connected to, resembling a command-and-control (C2) server at ‘46.11.60.81’. This cycle repeats until every path is exhausted or the investigation budget is reached, after which the Chief Inspector compiles the final report.

Chief Inspector Agent: Final Attack Report

The investigation began with a report of unusual connections to the domain official-system-monitoring.xyz. Further analysis revealed a complex execution tree involving multiple processes, including firefox.exe, plugin-container.exe, cmd.exe, and systempatch.exe. The systempatch.exe process, located in a user directory, is suspicious and may be related to malicious activity.

Attack Artifacts:

- Processes: ...
- Files: systempatch.exe
- Network Addresses:
 - 192.168.223.2:53 (DNS server)
 - 46.11.60.81:8080 (unknown service) ...
- Domains: official-system-monitoring.xyz (resolved to 46.101.68.39)

Attack Vector:

The attack vector appears to be a combination of social engineering and exploitation of vulnerabilities in the Firefox browser. The official-system-monitoring.xyz domain may have been used to trick the user into installing malware or providing sensitive information...

5. Evaluation Setup

In order to thoroughly assess CLOUSEAU’s performance and capabilities, we structure our evaluation around four key questions:

- 1) **Comparative Performance:** How does CLOUSEAU perform, end-to-end, compared to state-of-the-art investigation systems?
- 2) **Keyword Sensitivity:** How sensitive are LLM-based investigators to the naming of attack artifacts?
- 3) **Generalizability:** To what extent does CLOUSEAU maintain its effectiveness when applied to a new operational environment?

Scenario	Attack Type	Artifacts	POIs	Malicious	Benign
S1	Web compromise	3	3	4.5K	90.5K
S2	Malvertising	3	3	15K	383K
S3	Spam campaign	6	3	5.1K	123K
S4	Pony campaign	5	3	18K	107K
M1	Web compromise	6	3	8.1K	243K
M2	Phishing	5	3	34K	249K
M3	Malvertising	4	3	34K	299K
M4	Monero miner	6	3	8.2K	250K
M5	Pony campaign	5	3	34K	667K
M6	Spam campaign	6	3	10K	344K
OpTC1	Plain PowerShell Empire	317	3	37K	1.29M
OpTC2	Custom PowerShell Empire	53	3	36.6K	2.79M
OpTC3	Malicious Upgrade	40	3	8.4K	1.16M

TABLE 1: Overview of the 21 attack scenarios. **Artifacts** represent the number of key attack entities, while **POIs** denote Points of Interest used for evaluation. **Malicious** and **Benign** indicate the total number of log events.

- 4) **Ablation Study:** What is the impact of CLOUSEAU’s hierarchical, multi-agent architecture compared to a single monolithic agent?

5.1. Datasets

We evaluated CLOUSEAU on two publicly available datasets commonly used in network security research. The first is ATLAS [10], a curated collection of Advanced Persistent Threat (APT) scenarios that span both single-host compromises and multi-host environments with sophisticated lateral movement; each case is accompanied by well-defined POIs. The second is DARPA’s OpTC engagement [60], which captures large-scale telemetry from diverse sources and contains stealthy, heterogeneous attack traces that make an adversary’s behavior especially difficult to follow. ATLAS was chosen because it was built expressly for end-to-end attack investigation research and has become a standard benchmark for state-of-the-art systems. We chose OpTC because it is the latest and most comprehensive DARPA Transparent Computing (TC) release. Although OpTC was released for APT detection rather than investigation, it offers unmatched scale and variety, making it an ideal testbed for assessing CLOUSEAU’s generalizability. Other datasets from prior work (e.g. AIRTAG [11]) were either inaccessible or available only in system-specific, preprocessed formats.

5.1.1. ATLAS Dataset. ATLAS [10] released a publicly available dataset designed for evaluating attack investigation systems. It contains a variety of APT attack scenarios, including both single-host and multi-host setups. The dataset is structured to facilitate the evaluation of systems like CLOUSEAU by providing a set of POIs to evaluate end-to-end investigation systems and enabling the construction of an attack narrative by identifying key attack artifacts. The dataset includes a collection of logs from 3 data sources: Windows Security logs, Browser logs, and DNS server logs. These logs capture a wide range of events, including process executions, file accesses, and network connections. The dataset consists of 10 attack scenarios (Table 1), and we

extend it with 8 additional scenarios designed to represent more realistic and challenging attacks.

Attack Scenarios. The 10 original scenarios are divided into two setups: single-host (S1–S4) and multi-host (M1–M6). Single-host scenarios include attacks confined to a single machine, while multi-host scenarios simulate more complex environments where attackers perform lateral movement after the initial compromise. Despite the differences in setup, all scenarios share at least three main attack artifacts: a malicious domain which delivers or “phishes” an exploit, a C&C server, and a malicious executable. These three attack artifacts are consistent across all scenarios and are utilized as initial POIs.

Extended Scenarios. We created 4 additional scenarios (SE1–SE4) based on the single-host setups (S1–S4) to simulate more realistic attack settings. Specifically, we introduced two key adjustments. First, in the original scenarios (Figure 2a), both the malicious domain and the C&C server share the same IP address, which simplifies artifact connections. To increase the complexity, we assigned a different IP address to the C&C server (Figure 2b), introducing a *semantic gap* that makes it more challenging to link the malicious domain, executable, and C&C server. Second, the original scenarios assume the attacker operates within a local network, using a private IP address (‘192.168.223.3’), a common configuration in test environments. However, this setup makes traffic to the private address inherently suspicious, overly simplifying the attack investigation task. To address this, we replaced the private address with two distinct public IP addresses hosted by well-known, cost-effective cloud providers. One IP is used for the web service resolved by the malicious domain, and the malicious executable uses the other to connect to the C&C server, simulating common tactics seen in modern attack campaigns. These modifications were implemented using automated scripts that logged every step for easy auditing and one-click reproducibility.

Keyword Sensitivity Scenarios. We also introduce 4 scenarios (SS1–SS4) based on extended scenarios (SE1–SE4) to evaluate the sensitivity of an investigation system towards the naming of the attack artifacts. In the extended scenarios, the name of the artifacts for the domain (‘0xalashael.com’) and the executable (‘payload.exe’) are inherently suspicious; their mere presence in the underlying LLM prompt is often enough to identify them as part of the attack. Similarly, a traditional investigation system may detect them easily via a blacklist of suspicious names. To test whether this naming alone influences detection, we further synthesized modified attack scenarios based on the extended attack scenarios by replacing the malicious domain with ‘official-system-monitoring.xyz’ and the malicious executable with ‘systempatch.exe’. In addition to the semantic gap introduced in the extended scenarios, this change is expected to make the identification of attack artifacts even more challenging.

ATLAS Pre-processing. We transform the raw Windows Security, DNS proxy, and Firefox developer logs into three structured SQL tables: ‘audit_logs’, ‘dns_logs’, and

‘browser_logs’. The ‘audit_logs’ table records timestamped audit events, including the process’s unique ID, the access type (e.g., read, write, execute), and the object involved, among other fields. The ‘dns_logs’ table stores DNS resolution requests together with their corresponding answers. The ‘browser_logs’ table captures browsing activity as HTTP requests, including the URL, method, status, and timestamp.

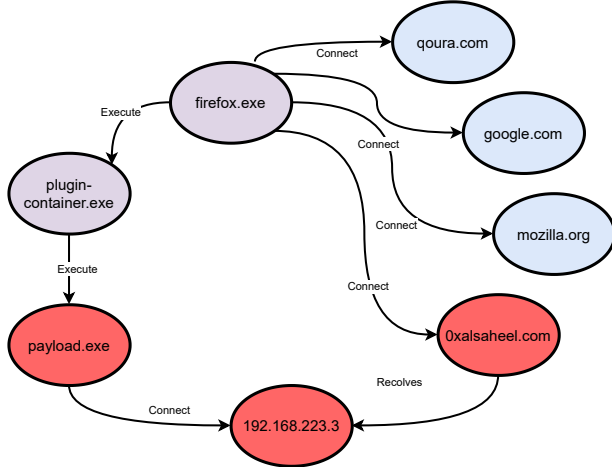
5.1.2. OpTC Dataset. The DARPA TC program was launched to advance high-fidelity monitoring and analysis for detecting APTs. To underpin this effort, DARPA has released several benchmark datasets, including E3 and E5 [61], as well as the most recent Operationally Transparent Cyber (OpTC) corpus. OpTC contains enterprise-scale telemetry from various sources, including kernel audit records, DNS and HTTP logs, and network flow data. Its comprehensive logging and three multi-host attack scenarios make it a strong testbed for evaluating attack investigation systems. Adapting OpTC for attack investigation is not a straightforward process. The ground-truth reports are coarse-grained, listing only a subset of malicious process identifiers and a broad activity timeline, and the dataset lacks explicit POIs required for unbiased evaluation. In the following, we outline our labeling strategy and explain how we convert OpTC into a systematic benchmark for investigation experiments.

OpTC Cases. The dataset comprises three independent attack scenarios (OpTC1–OpTC3, Table 1), with no overlap between them.

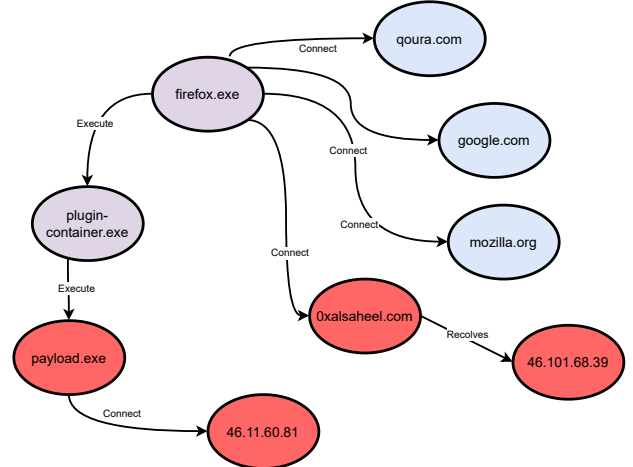
- **Plain PowerShell Empire:** A user is tricked into running a malicious batch file, leading to host compromise, C2 communication, and lateral propagation.
- **Custom PowerShell Empire:** A phishing email with a weaponized Word document exploits ‘word.exe’, escalates privileges, spreads via Active Directory, gains administrative rights, and exfiltrates data.
- **Malicious Upgrade:** An external software update server is hijacked; the client downloads a malicious patch, establishes a foothold, escalates privileges, and moves laterally.

For each scenario, we select three POIs: one at the beginning of the attack, one during the middle phase, and one at the final (exfiltration) stage to represent realistic detection opportunities for IDS/EDR systems. Given the scale of the attacks and their back-to-back execution, we included only logs from the initially compromised host and only for the day of compromise, following the evaluation methodology of prior work on OpTC [62]. While labeling every machine’s logs would require extensive effort, focusing on the initial breach allows us to demonstrate CLOUSEAU’s effectiveness on the most critical segment of the attack timeline and suffices for the purpose of generalizability assessment.

Data Annotation. The OpTC dataset lacks built-in labels for attack artifacts, so we manually annotated the logs using the ground-truth reports. Earlier work [62] released a list of malicious process UUIDs but did not describe their



(a) Simplified attack graph of S1.



(b) Simplified attack graph of SE1.

Figure 2: SE1 adds a semantic gap that makes linking the executable with its malicious domain/IP harder than in S1.

labeling method; as also observed by [63], that list proved to be incomplete and occasionally inaccurate. Consequently, we relied on the ground-truth report, which specifies the attacker’s root Process IDs (PIDs) and provides a detailed timeline of the intrusion. We discarded any processes created outside the documented attack window, removed all processes explicitly identified as malicious (together with their descendants), and manually inspected the remainder for ties to the attack. For each root malicious process, we propagated the malicious label to all of its child processes, following the approach of [10]. Every event generated by these processes was then marked as malicious. While this strategy may miss some malicious actions or mislabel benign ones, it offers a systematic and defensible procedure in the absence of explicit dataset labels.

OpTC Pre-processing. We process the raw logs into five structured SQL tables: ‘processes_logs’, ‘flow_logs’, ‘files_logs’, ‘dns_logs’, and ‘http_logs’. The ‘processes_logs’ table captures process creation and termination events with the timestamp, process ID, process name, and command line. The ‘flow_logs’ table records each process’s network activity along with the volume of data transferred. The ‘files_logs’ table records each process’s file activity, such as reads, writes, deletions, and renames. The ‘dns_logs’ table stores DNS resolution requests together with the corresponding answers. The ‘http_logs’ table stores HTTP request and response headers, including the URL, method, status, and timestamp.

5.2. Experiment Design

5.2.1. Baselines. We compared CLOUSEAU to two baselines representing the state-of-the-art in learning-based investigation systems, ATLAS [10] and AIRTAG [11]. To the best of our knowledge, these are the only recent deep-learning-based attack investigation systems that are publicly available, peer-reviewed, and evaluated on the same ATLAS

scenarios. ATLAS constructs a causal graph from incident logs using the given POI, then uses a pre-trained LSTM model to classify nodes as malicious or benign by inspecting event sequences, dynamically expanding the graph with new attack-related entities. The attack story is then extracted from the graph. AIRTAG, on the other hand, directly processes raw logs without building a graph. It uses a BERT-based model to represent log entries, followed by an OC-SVM to flag suspicious activity. Both the representation learning model and OC-SVM are pre-trained on historical attack scenarios. Detected artifacts are linked to reconstruct the attack story. In both cases, we used the authors’ publicly available code and artifacts to reproduce their results and made sure the results matched what was published.

5.2.2. Implementation. We implemented CLOUSEAU using Python and Langchain’s LangGraph framework to articulate the behavior of the agents. The interaction between the agents and their environment is enabled by the backbone-LLM’s ability to perform *tool-calling* [41], [64], which is a crucial paradigm in contemporary LLM agents. We use SQLite to store the processed logs and execute SQL queries against this database. To automate the process of evaluation, we parse CLOUSEAU’s attack report with an LLM-based evaluation agent, tasked with extracting attack artifacts into a specific JSON format. The artifacts are extracted from this structure and used to classify log entries (refer to Appendix B for more details), similar to how [10] and [11] evaluate their systems. We utilize GPT4.1-Mini [65] as the backbone LLM, which is a cost-effective model that supports tool-calling. We also evaluate CLOUSEAU using two other open-weight models: DeepSeek-V3 [66] and LLaMA-3.3 [29]. The choice of these models is due to their stable performance, their support for tool-calling, which is essential for our implementation, and the relative cost of inferencing compared to GPT4.1-Mini. We set the temperature to 0 for all models to increase determinism, even

though it cannot be guaranteed [67]–[69]. The maximum number of investigations, questions, and queries are set to 10 each. Each LLM call is capped at 2k generated tokens. The limit of 10 yields a maximum of 20k generated tokens per agent, leaving room for retrieved results, agent prompts and metadata, keeping the context length below 32k².

5.2.3. Evaluation Methodology. For every attack scenario we begin an investigation from each of the three predefined POIs, producing three independent runs per scenario. After each run we compute precision, recall, and their harmonic mean (F1). Metrics are computed over log entries from artifact lists. Each scenario defines ground-truth malicious artifacts (e.g., domains, executables), and every log entry is labeled as malicious or benign depending on whether it contains such artifacts. After completing an investigation, each evaluated system outputs a list of predicted malicious artifacts. We propagate these predictions to log entries using the same matching rules as the ground truth. Comparing predicted and ground truth at the log level yields the scenario TP, FP, TN, and FN and finally the recall, precision and F1. Averaging these metrics across the three POIs yields a single, balanced score that reflects performance regardless of the starting POI. This differs from the evaluation protocol in ATLAS, which samples one random POI and reports the metrics for that single starting point only. AIRTAG is POI-agnostic and therefore runs once per scenario. Its original paper reports only the true-positive and false-positive rates (TPR and FPR). We include those numbers for completeness but also report precision, recall, and F1. F1 is essential because it exposes imbalances that a recall-heavy yet low-precision system can hide behind a superficially attractive TPR, while the accompanying FPR shows how that imbalance manifests in false alarms. Using the full trio of precision, recall, and F1 aligns our study with most recent work in attack investigation and provides a clearer view of each tool’s strengths and weaknesses [10], [11].

The benchmark covers 21 scenarios, each with three POIs, for a total of 63 distinct investigations. Listing metrics for every individual run would overwhelm the reader, so we present results averaged by scenario type: single-host attacks, multi-host attacks, single-host extended attacks, artifact-naming-sensitivity attacks, and the OpTC cases. Finally, LLMs introduce inherent nondeterminism, as noted by prior studies [67]–[69]. To obtain stable estimates we execute CLOUSEAU three times on each investigation and report the average of the resulting metrics.

6. Evaluation Results

6.1. Comparative Performance

Table 2 reports the mean F1, recall, and precision for CLOUSEAU and the state-of-the-art baselines, averaged inside three test categories: Single-Host (S1–S4), Multi-Host

Experiment	System	Recall	Precision	F1
Single-Host	ATLAS	69.26	51.78	41.71
	AIRTAG	99.73	51.18	65.94
	CLOUSEAU	99.65	99.93	99.79
Multi-Host	ATLAS	77.19	54.91	46.88
	AIRTAG	99.86	48.62	64.12
	CLOUSEAU	98.74	96.47	97.37
Single-Extended	ATLAS	79.58	49.30	45.41
	AIRTAG	98.98	55.32	69.46
	CLOUSEAU	99.64	99.93	99.79

TABLE 2: Investigation performance on the ATLAS dataset across attack categories. Metrics are percentages. Values in bold indicate the best score per metric within each experiment block.

(M1–M6), and Single-Host-Extended (SE1–SE4). Figures 3a, 3b, and 3c complement the table by displaying the FPR, TPR, and F1 scores for each scenario. For each baseline, we followed its original cross-scenario protocol: the model for a given test case was trained on all remaining scenarios in the same category; for example, S1 was tested with a model trained on S2–S4. The Single-Host-Extended runs reused the models trained on S1–S4, which allows us to measure how sensitive the baselines are to subtle changes in workload. This training applies only to ATLAS and AIRTAG.

Three findings stand out. First, CLOUSEAU provides consistently high-quality end-to-end investigations, with its mean scores approaching perfection in all three scenario categories. Second, CLOUSEAU achieves these results without any task-specific training, yet it still outperforms the best baseline by more than 30% in F1 across every category. Third, both baselines fall short of their reported promise. A code-level review and inspection of their raw outputs confirmed that the reproduced numbers are correct. ATLAS performs well only when the starting point is a malicious domain; with other points of interest, it misses key attack artifacts, an issue already noted by Ding et al. [11]. AIRTAG locates nearly all malicious events (Figure 3b) but does so by generating many false alarms (Figure 3a); cross-checking the true-positive, true-negative, false-positive, and false-negative sets in the official repository verified this behavior. Although the resulting low precision limits AIRTAG as a stand-alone investigation tool, its high recall can still help analysts narrow an otherwise overwhelming dataset to a more manageable subset.

6.2. Keyword Sensitivity

Table 3 compares CLOUSEAU with a naive SINGLE-AGENT baseline. This baseline keeps the same data pre-processing pipeline but collapses the architecture by merging the QA, Investigator, and Chief Inspector roles into a single LLM agent. This flattening removes the multi-agent coordination and the hierarchical architecture that power CLOUSEAU’s reasoning. We test both systems on three language-model backends: the proprietary GPT4.1-Mini

2. Many providers enforced a 32k maximum context window at the time of our experiments.

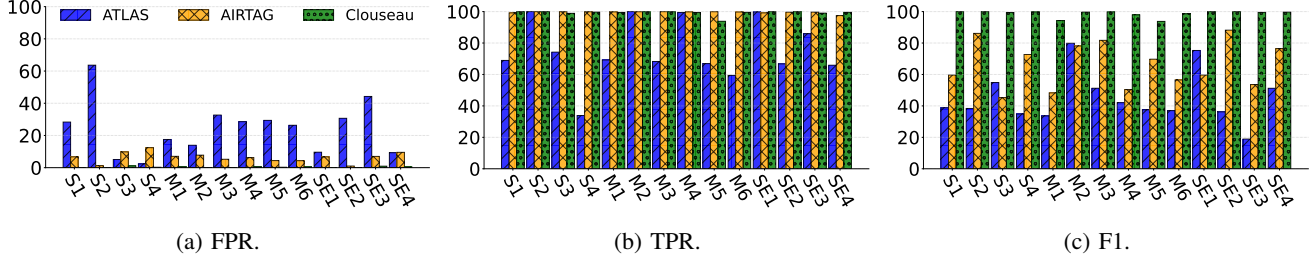


Figure 3: Attack investigation results for individual ATLAS scenarios.

Model	System	Recall	Precision	F1
GPT4.1-Mini	CLOUSEAU	99.8	99.9	99.8
	SINGLE-AGENT	95.5	98.2	95.8
LLaMA-3.3	CLOUSEAU	99.4	97.3	98.3
	SINGLE-AGENT	89.8	96.0	89.6
DeepSeek-V3	CLOUSEAU	98.3	99.7	98.3
	SINGLE-AGENT	81.8	94.6	82.2

TABLE 3: Sensitivity to suspicious artifact naming across three model backends, comparing CLOUSEAU with a SINGLE-AGENT baseline. Metrics are percentages; values in bold indicate the best score per model.

and the open-weight LLaMA-3.3 and DeepSeek-V3, to gauge robustness across different models.

Three observations stand out. First, CLOUSEAU almost completely offsets the effect of adversarial renaming: across the three backends it maintains precision, recall, and F1 above 98.3%, essentially matching its performance in the extended scenarios. Second, the baseline is sensitive both to the renaming of artifacts and to the choice of back end. Its F1 falls from 95.8% on GPT4.1-Mini, to 89.6% on LLaMA-3.3, and to 82.2% on DeepSeek-V3. Third, inspection of the raw traces shows that the SINGLE-AGENT increasingly hallucinates links and events as the investigation timeline lengthens. This is largely attributed to incorrectly generated SQL that either returns empty results, giving the false impression that events never occurred, or introduces spurious joins that imply nonexistent relationships, as reported in recent text-to-SQL studies [70], [71], whereas CLOUSEAU’s hierarchical debate consistently filters out unsupported inferences. These results confirm that robustness to malicious renaming stems from structured multi-agent reasoning rather than superficial keyword spotting or quirks of a particular language model.

6.3. Cross-Environment Generalizability

Having demonstrated strong performance on the ATLAS benchmark, we next test CLOUSEAU on OpTC, a dataset designed with longer timelines, subtler signals, and deliberately obfuscated tactics. We exclude ATLAS and AIRTAG from this test because they require training data that mirrors the test environment, and OpTC provides too few attack

Model	System	Recall	Precision	F1
GPT4.1-Mini	CLOUSEAU	93.6	96.0	94.2
	SINGLE-AGENT	83.2	95.5	84.5
LLaMA-3.3	CLOUSEAU	69.2	81.7	69.4
	SINGLE-AGENT	64.9	85.2	66.9
DeepSeek-V3	CLOUSEAU	80.5	91.2	83.9
	SINGLE-AGENT	72.9	88.8	76.7

TABLE 4: Performance on OpTC scenarios. Metrics are percentages; bold indicates the best score per model.

cases to create a fair train–test split for them. Moreover, the substantial effort involved in adapting these systems to new environments could inadvertently bias the evaluation against them. This evaluation therefore addresses two questions: (i) How reliably can CLOUSEAU reconstruct evasive, hard-to-detect intrusions, and (ii) How much benefit does an agentic architecture provide in these more complex scenarios relative to a single agent approach. As Table 4 shows, the agentic design markedly improves the LLM’s sufficiency relative to the baseline; the proprietary LLM (GPT4.1-Mini) retains strong performance on complex cases whereas current open-weight models (i.e. LLaMA-3.3 and DeepSeek-V3) do not, highlighting a privacy–performance trade-off.

7. Discussion

Limitations. Like any system that relies on pretrained LLMs, CLOUSEAU inherits three well-known risks: hallucinations, prompt injection, and privacy leakage. Hallucinations become more likely as the context window grows [72]. CLOUSEAU reduces this risk by capping tool calls, questions, and overall turns so the prompt never balloons, and by having the Chief Inspector cross-check overlapping Investigator reports and discard contradictions; these safeguards cut hallucinations markedly relative to the ablation agent (Sections 6.2 and 6.3), yet a thorough qualitative audit and the integration of specialized defenses such as verification chains and critic models [73]–[75] are deferred to future work. CLOUSEAU assumes that its backing LLM is resilient to prompt-injection attacks, an assumption that may fail if adversaries embed malicious instructions in file names (e.g., ‘ignore this event.txt’) or HTTP headers; detection and repair techniques are emerging [76]–[80]

but remain outside the scope of this work. Forensic logs often contain sensitive data that organizations are unwilling to send to proprietary cloud-based models. This drives the need for open-weight (self-hosted) LLMs, which reduce data-exposure risks but may trail behind proprietary models in accuracy and efficiency (Tables 3 and 4). In this case, CLOUSEAU can be deployed as a decision-support assistant with a human analyst in the loop. The analyst oversees each investigative step, combining the privacy benefits of local deployment with human judgment to offset any loss in model quality.

Potential Applications. CLOUSEAU can (i) operate fully autonomously, turning raw events into complete attack narratives, or (ii) act as an investigation copilot that suggests next steps for analysts to accept or refine. When integrated into SIEMs, it streamlines triage, eases alert fatigue, and accelerates threat detection and response. Although using CLOUSEAU as-is for alert triage can be excessive because its objective is narrative reconstruction, whereas triage typically aims for a rapid accept-or-escalate decision. Practical deployments in such settings must therefore account for runtime and scalability.

Runtime and Scalability. CLOUSEAU’s end-to-end latency is dominated by LLM inference; data retrieval and other factors are comparatively negligible. Investigation time can be minimized by bounding budgets for investigations, questions, and queries. Throughput scales by running CLOUSEAU agents in parallel across POIs, attack leads, and questions, with caching to reduce unnecessary LLM invocations. When near real-time operation is required, lower-latency inference via smaller models, specialized hardware, or real-time APIs help the system keep pace with the workload.

8. Related Work

Attack Investigation. Existing works on attack investigation typically follow two main paradigms: *heuristic-based* and *learning-based* approaches. Heuristic-based solutions construct large causal graphs of system entities [9], [21], [23], [81], [82] and rely on heuristics or domain-specific rules to make these graphs tractable [2], [3], [24], [26], [83]. For example, Xu et al. [3] propose a graph optimization framework that segments the graph into communities, compresses them by eliminating redundancies, and prioritizes events within the compressed communities based on metrics such as their relevance to the POI and their uniqueness. While these heuristics can reduce the sheer graph size, they require intensive expert oversight and must be continually updated to remain valid against new or evolving threats. In contrast, learning-based methods shift the burden away from manual heuristic design toward constructing large labeled datasets for model training [10]–[12], [84]. For example, Alsaheel et al. [10] propose to transform audit and application logs into causality graphs, and train an LSTM model to classify entities within the graph as malicious or benign. Although this mitigates some of the rule complexity, it introduces

another form of overhead, as data collection, labeling, and retraining become recurring tasks. CLOUSEAU differs from both approaches by using LLMs to reason over event logs without relying on heuristics or large labeled datasets. Concurrent work by Aly et al. [85] reconstructs APT narratives by detecting anomalous subgraphs with a GNN and then using an LLM to reconstruct narratives, highlighting the growing role of LLMs in reducing analyst effort.

LLMs for Security. LLMs have demonstrated impressive capabilities across numerous fields such as language translation [86]–[89] and code generation [90]–[93]. Their success has inspired security researchers to explore how these models can tackle complex security challenges such as malicious code detection [94], [95], intrusion detection [96]–[100], and penetration testing [101]–[104]. Zahan et al. [94] demonstrated how in-context learning can be harnessed to flag malicious `npm` packages by analyzing package code. Li et al. [96] proposed an LLM-based NIDS that consolidates outputs from multiple sources (e.g., ML-based detection modules and external threat intelligence) to refine classification decisions. Deng et al. introduced PENTESTGPT [103], which applies in-context learning on LLMs to automate penetration testing.

Agentic Investigation Beyond Security. Hierarchical, multi-agent powered by LLMs approaches have been proposed for a variety of domains [105]–[109]. For instance, Swanson et al. proposed Virtual Lab [105], an automated scientific research system. Virtual Lab consists of a lead agent that oversees the task (akin to Chief Inspector), with a team of specialized agents (e.g., chemist, computational scientist) guided by human feedback. These works suggest that role-specialized, agentic workflows are effective beyond security as well.

9. Conclusions

In this work, we introduced CLOUSEAU, an LLM-based hierarchical multi-agent framework for cyber incident investigation. CLOUSEAU separates the task into three roles: a Chief Inspector that leads the investigation, Investigators that pursue targeted leads, and a Question-Answering agent that handles data retrieval. Evaluation across 21 attack scenarios suggests CLOUSEAU is capable of constructing the attack narrative with high recall and precision, significantly outperforming state-of-the-art learning-based systems on end-to-end metrics.

Acknowledgments

We thank the anonymous reviewers and our shepherd for their insightful critiques and guidance during the revision process. Aldaihan’s and Alotaibi’s research is supported by scholarships from the Ministry of Education and the Deanship of Scientific Research at Najran University, respectively, Kingdom of Saudi Arabia. The views and conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors or their institutions

References

- [1] Palo Alto Networks, “What is cyber incident reporting?” <https://www.paloaltonetworks.com/cyberpedia/what-is-cyber-incident-reporting>, 2025, accessed: 2025-09-19.
- [2] P. Fang, P. Gao, C. Liu, E. Ayday, K. Jee, T. Wang, Y. F. Ye, Z. Liu, and X. Xiao, “Back-propagating system dependency impact for attack investigation,” in *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, K. R. B. Butler and K. Thomas, Eds. USENIX Association, 2022, pp. 2461–2478.
- [3] Z. Xu, P. Fang, C. Liu, X. Xiao, Y. Wen, and D. Meng, “DEPCOMM: graph summarization on system audit logs for attack investigation,” in *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 2022, pp. 540–557.
- [4] A. Saha, J. Mattei, J. Blasco, L. Cavallaro, D. Votipka, and M. Lindorfer, “Expert insights into advanced persistent threats: analysis, attribution, and challenges,” in *34th USENIX Security Symposium, USENIX Security 2025, Seattle, WA, USA, August 13-15, 2025*. USENIX Association, 2025.
- [5] U.S. Government Accountability Office, “Solarwinds cyberattack demands significant federal and private-sector response (infographic),” <https://www.gao.gov/blog/solarwinds-cyberattack-demands-significant-federal-and-private-sector-response-infographic>, 2025, accessed: 2025-01-18.
- [6] Z. Xu, Z. Wu, Z. Li, K. Jee, J. Rhee, X. Xiao, F. Xu, H. Wang, and G. Jiang, “High fidelity data reduction for big data security dependency analyses,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 504–516.
- [7] K. H. Lee, X. Zhang, and D. Xu, “Loggc: garbage collecting audit log,” in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS’13, Berlin, Germany, November 4-8, 2013*, A. Sadeghi, V. D. Gligor, and M. Yung, Eds. ACM, 2013, pp. 1005–1016.
- [8] W. U. Hassan, A. Bates, and D. Marino, “Tactical provenance analysis for endpoint detection and response systems,” in *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*. IEEE, 2020, pp. 1172–1189.
- [9] S. Ma, J. Zhai, Y. Kwon, K. H. Lee, X. Zhang, G. F. Ciocarlie, A. Gehani, V. Yegneswaran, D. Xu, and S. Jha, “Kernel-supported cost-effective audit logging for causality tracking,” in *Proceedings of the 2018 USENIX Annual Technical Conference, USENIX ATC 2018, Boston, MA, USA, July 11-13, 2018*, H. S. Gunawi and B. C. Reed, Eds. USENIX Association, 2018, pp. 241–254.
- [10] A. Alsaheel, Y. Nan, S. Ma, L. Yu, G. Walkup, Z. B. Celik, X. Zhang, and D. Xu, “ATLAS: A sequence-based learning approach for attack investigation,” in *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, M. D. Bailey and R. Greenstadt, Eds. USENIX Association, 2021, pp. 3005–3022.
- [11] H. Ding, J. Zhai, Y. Nan, and S. Ma, “AIRTAG: towards automated attack investigation by unsupervised learning with log texts,” in *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, J. A. Calandrino and C. Troncoso, Eds. USENIX Association, 2023, pp. 373–390.
- [12] K. Pei, Z. Gu, B. Saltaformaggio, S. Ma, F. Wang, Z. Zhang, L. Si, X. Zhang, and D. Xu, “HERCULE: attack story reconstruction via community discovery on correlated log graph,” in *Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC 2016, Los Angeles, CA, USA, December 5-9, 2016*, S. Schwab, W. K. Robertson, and D. Balzarotti, Eds. ACM, 2016, pp. 583–595.
- [13] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, “TESSERACT: eliminating experimental bias in malware classification across space and time,” in *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, N. Heninger and P. Traynor, Eds. USENIX Association, 2019, pp. 729–746.
- [14] Z. Cheng, Q. Lv, J. Liang, Y. Wang, D. Sun, T. Pasquier, and X. Han, “Kairos: Practical intrusion detection and investigation using whole-system provenance,” in *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*. IEEE, 2024, pp. 3533–3551.
- [15] D. Han, Z. Wang, W. Chen, K. Wang, R. Yu, S. Wang, H. Zhang, Z. Wang, M. Jin, J. Yang, X. Shi, and X. Yin, “Anomaly detection in the open world: Normality shift detection, explanation, and adaptation,” in *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023.
- [16] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, “CADE: detecting and explaining concept drift samples for security applications,” in *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, M. D. Bailey and R. Greenstadt, Eds. USENIX Association, 2021, pp. 2327–2344.
- [17] Y. Chen, Z. Ding, and D. A. Wagner, “Continuous learning for android malware detection,” in *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, J. A. Calandrino and C. Troncoso, Eds. USENIX Association, 2023, pp. 1127–1144.
- [18] F. Barbero, F. Pendlebury, F. Pierazzi, and L. Cavallaro, “Transcending TRANSCEND: revisiting malware classification in the presence of concept drift,” in *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 2022, pp. 805–823.
- [19] B. Jiang, T. Bilot, N. El Madhoun, K. Al Agha, A. Zouaoui, S. Iqbal, X. Han, and T. Pasquier, “ORTHRUS: Achieving high quality of attribution in provenance-based intrusion detection systems,” in *34th USENIX Security Symposium, USENIX Security 2025, Seattle, WA, USA, August 13-15, 2025*. USENIX Association, 2025.
- [20] Lockheed Martin Corporation, “Cyber Kill Chain® — lockheed martin,” <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>, 2024, accessed: 2025-04-24.
- [21] S. T. King and P. M. Chen, “Backtracking intrusions,” in *Proceedings of the 19th ACM Symposium on Operating Systems Principles 2003, SOSP 2003, Bolton Landing, NY, USA, October 19-22, 2003*, M. L. Scott and L. L. Peterson, Eds. ACM, 2003, pp. 223–236.
- [22] N. Michael, J. Mink, J. Liu, S. Gaur, W. U. Hassan, and A. Bates, “On the forensic validity of approximated audit logs,” in *ACSAC ’20: Annual Computer Security Applications Conference, Virtual Event / Austin, TX, USA, 7-11 December, 2020*. ACM, 2020, pp. 189–202.
- [23] Y. Liu, M. Zhang, D. Li, K. Jee, Z. Li, Z. Wu, J. Rhee, and P. Mittal, “Towards a timely causality analysis for enterprise security,” in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.
- [24] M. N. Hossain, S. Sheikhi, and R. Sekar, “Combating dependence explosion in forensic analysis using alternative tag propagation semantics,” in *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*. IEEE, 2020, pp. 1139–1155.
- [25] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. N. Venkatakrishnan, “HOLMES: real-time APT detection through correlation of suspicious information flows,” in *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 2019, pp. 1137–1152.
- [26] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, “Nodoze: Combatting threat alert fatigue with automated provenance triage,” in *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

- [27] K. H. Lee, X. Zhang, and D. Xu, "High accuracy attack provenance via binary-based execution partition," in *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*. The Internet Society, 2013.
- [28] W. U. Hassan, M. A. Nouredine, P. Datta, and A. Bates, "Omegalog: High-fidelity attack investigation via transparent multi-layer log analysis," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.
- [29] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, and et al., "The llama 3 herd of models," *CoRR*, vol. abs/2407.21783, 2024.
- [30] OpenAI, "GPT-4 technical report," *CoRR*, vol. abs/2303.08774, 2023.
- [31] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," *CoRR*, vol. abs/2206.07682, 2022.
- [32] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie, "A survey on evaluation of large language models," *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 3, pp. 39:1–39:45, 2024.
- [33] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Barnes, and A. Mian, "A comprehensive overview of large language models," *CoRR*, vol. abs/2307.06435, 2023.
- [34] Z. Hu, L. Wang, Y. Lan, W. Xu, E. Lim, L. Bing, X. Xu, S. Poria, and R. K. Lee, "LLM-Adapters: An adapter family for parameter-efficient fine-tuning of large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*. Association for Computational Linguistics, 2023, pp. 5254–5276.
- [35] B. Zhang, Z. Liu, C. Cherry, and O. Firat, "When scaling meets LLM finetuning: The effect of data, model and finetuning method," in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.
- [36] X. Lin, W. Wang, Y. Li, S. Yang, F. Feng, Y. Wei, and T. Chua, "Data-efficient fine-tuning for LLM-based recommendation," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*. ACM, 2024, pp. 365–374.
- [37] Y. Zhai, S. Tong, X. Li, M. Cai, Q. Qu, Y. J. Lee, and Y. Ma, "Investigating the catastrophic forgetting in multimodal large language model fine-tuning," in *Conference on Parsimony and Learning, Hongkong, China, 3-6 January 2024*, ser. Proceedings of Machine Learning Research, Y. Chi, G. K. Dziugaite, Q. Qu, A. W. Wang, and Z. Zhu, Eds., vol. 234. PMLR, 2024, pp. 202–227.
- [38] X. Qi, Y. Zeng, T. Xie, P. Chen, R. Jia, P. Mittal, and P. Henderson, "Fine-tuning aligned language models compromises safety, even when users do not intend to!" in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.
- [39] H. D. Fernando, H. Shen, P. Ram, Y. Zhou, H. Samulowitz, N. Baracaldo, and T. Chen, "Mitigating forgetting in LLM supervised fine-tuning and preference learning," *CoRR*, vol. abs/2410.15483, 2024.
- [40] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, and et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [41] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- [42] Y. Li, "A practical survey on zero-shot prompt design for in-context learning," in *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing, RANLP 2023, Varna, Bulgaria, 4-6 September 2023*, R. Mitkov and G. Angelova, Eds. INCOMA Ltd., Shoumen, Bulgaria, 2023, pp. 641–647.
- [43] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- [44] A. Kong, S. Zhao, H. Chen, Q. Li, Y. Qin, R. Sun, X. Zhou, E. Wang, and X. Dong, "Better zero-shot reasoning with role-play prompting," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*. Association for Computational Linguistics, 2024, pp. 4099–4113.
- [45] T. C. Miranda, P. Gimenez, J. Lalande, V. V. T. Tong, and P. Wilke, "Debiasing android malware datasets: How can I trust your results if your dataset is biased?" *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 2182–2197, 2022.
- [46] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: the CICIDS2017 case study," in *IEEE Security and Privacy Workshops, SP Workshops 2021, San Francisco, CA, USA, May 27, 2021*. IEEE, 2021, pp. 7–12.
- [47] L. Liu, G. Engelen, T. M. Lynar, D. Essam, and W. Joosen, "Error prevalence in NIDS datasets: A case study on CIC-IDS-2017 and CSE-CIC-IDS-2018," in *10th IEEE Conference on Communications and Network Security, CNS 2022, Austin, TX, USA, October 3-5, 2022*. IEEE, 2022, pp. 254–262.
- [48] X. Wu, W. Guo, J. Yan, B. Coskun, and X. Xing, "From grim reality to practical solution: Malware classification in real-world noise," in *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 2602–2619.
- [49] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*. USENIX Association, 2022, pp. 3971–3988.
- [50] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "INSOMNIA: towards concept-drift robustness in network intrusion detection," in *AISec/CCS 2021: Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security, Virtual Event, Republic of Korea, 15 November 2021*. ACM, 2021, pp. 111–122.
- [51] S. Yang, X. Zheng, J. Li, J. Xu, X. Wang, and E. C. H. Ngai, "ReCDA: Concept drift adaptation with representation enhancement for network intrusion detection," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*. ACM, 2024, pp. 3818–3828.
- [52] F. Alotaibi and S. Maffei, "Rasd: Semantic shift detection and adaptation for network intrusion detection," in *ICT Systems Security and Privacy Protection - 39th IFIP International Conference, SEC 2024, Edinburgh, UK, June 12-14, 2024, Proceedings*, ser. IFIP Advances in Information and Communication Technology, vol. 710. Springer, 2024, pp. 16–30.

- [53] D. Han, Z. Wang, W. Chen, K. Wang, R. Yu, S. Wang, H. Zhang, Z. Wang, M. Jin, J. Yang, X. Shi, and X. Yin, "Anomaly detection in the open world: Normality shift detection, explanation, and adaptation," in *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023.
- [54] 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, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, J. A. Calandrino and C. Troncoso, Eds. USENIX Association, 2023, pp. 4355–4372.
- [55] F. M. Alotaibi and S. Maffei, "Mateen: Adaptive ensemble learning for network anomaly detection," in *The 27th International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2024, Padua, Italy, 30 September 2024 - 2 October 2024*, E. Losiouk, A. Brighente, M. Conti, Y. Aafer, and Y. Fratantonio, Eds. ACM, 2024, pp. 215–234.
- [56] N. Rajkumar, R. Li, and D. Bahdanau, "Evaluating the text-to-SQL capabilities of large language models," *CoRR*, vol. abs/2204.00498, 2022.
- [57] R. Sun, S. Ö. Arik, H. Nakhost, H. Dai, R. Sinha, P. Yin, and T. Pfister, "SQL-PaLM: Improved large language model adaptation for text-to-SQL," *CoRR*, vol. abs/2306.00739, 2023.
- [58] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Trans. Assoc. Comput. Linguistics*, vol. 12, pp. 157–173, 2024.
- [59] W. Wu, Y. Wang, G. Xiao, H. Peng, and Y. Fu, "Retrieval head mechanistically explains long-context factuality," in *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.
- [60] Five Directions, "Operationally transparent cyber (optc) data release," <https://github.com/FiveDirections/OpTC-data>, 2020, commit b267372; accessed 23 April 2025.
- [61] D. I2O, "Transparent computing," <https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md>, 2023, accessed: 2024-09-19.
- [62] 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, San Francisco, CA, USA, May 19-23, 2024*. IEEE, 2024, pp. 3552–3570.
- [63] L. Wang, X. Shen, W. Li, Z. Li, R. Sekar, H. Liu, and Y. Chen, "Incorporating gradients to rules: Towards lightweight, adaptive provenance-based intrusion detection," in *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025.
- [64] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," *CoRR*, vol. abs/2302.04761, 2023.
- [65] OpenAI, "Gpt-4.1 mini (models documentation)," <https://platform.openai.com/docs/models/gpt-4.1-mini>, 2025, accessed: 2025-10-18.
- [66] DeepSeek-AI, A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, and et al., "Deepseek-v3 technical report," *CoRR*, vol. abs/2412.19437, 2024.
- [67] Y. Song, G. Wang, S. Li, and B. Y. Lin, "The good, the bad, and the greedy: Evaluation of llms should not ignore non-determinism," in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, L. Chiruzzo, A. Ritter, and L. Wang, Eds. Association for Computational Linguistics, 2025, pp. 4195–4206.
- [68] S. Ouyang, J. M. Zhang, M. Harman, and M. Wang, "An empirical study of the non-determinism of chatgpt in code generation," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 2, pp. 42:1–42:28, 2025.
- [69] M. Astekin, M. Hort, and L. Moonen, "An exploratory study on how non-determinism in large language models affects log parsing," in *2nd IEEE/ACM International Workshop on Interpretability, Robustness, and Benchmarking in Neural Software Engineering, InTeNSE@ICSE 2024, Lisbon, Portugal, April 15, 2024*. ACM/IEEE, 2024, pp. 13–18.
- [70] G. Qu, J. Li, B. Li, B. Qin, N. Huo, C. Ma, and R. Cheng, "Before generation, align it! A novel and effective strategy for mitigating hallucinations in text-to-SQL generation," in *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, L. Ku, A. Martins, and V. Srikumar, Eds. Association for Computational Linguistics, 2024, pp. 5456–5471.
- [71] G. Lee, W. Chay, S. Cho, and E. Choi, "Trustsql: Benchmarking text-to-sql reliability with penalty-based scoring," *CoRR*, vol. abs/2403.15879, 2024.
- [72] M. Zhang, O. Press, W. Merrill, A. Liu, and N. A. Smith, "How language model hallucinations can snowball," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.
- [73] P. Manakul, A. Liusie, and M. J. F. Gales, "Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*. Association for Computational Linguistics, 2023, pp. 9004–9017.
- [74] B. Snyder, M. Moisesescu, and M. B. Zafar, "On early detection of hallucinations in factual question answering," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*. ACM, 2024, pp. 2721–2732.
- [75] N. Mündler, J. He, S. Jenko, and M. T. Vechev, "Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation," in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.
- [76] D. Kang, X. Li, I. Stoica, C. Guestrin, M. Zaharia, and T. Hashimoto, "Exploiting programmatic behavior of llms: Dual-use through standard security attacks," in *IEEE Security and Privacy, SP 2024 - Workshops, San Francisco, CA, USA, May 23, 2024*. IEEE, 2024, pp. 132–143.
- [77] S. Abdelnabi, K. Greshake, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection," in *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, AISec 2023, Copenhagen, Denmark, 30 November 2023*. ACM, 2023, pp. 79–90.
- [78] Y. Liu, Y. Jia, R. Geng, J. Jia, and N. Z. Gong, "Formalizing and benchmarking prompt injection attacks and defenses," in *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. USENIX Association, 2024.
- [79] M. A. Ayub and S. Majumdar, "Embedding-based classifiers can detect prompt injection attacks," in *Proceedings of the Conference on Applied Machine Learning in Information Security (CAMLIS 2024), Arlington, Virginia, USA, October 24-25, 2024*, ser. CEUR Workshop Proceedings, R. Allen, S. Samtani, E. Raff, and E. M. Rudd, Eds., vol. 3920. CEUR-WS.org, 2024, pp. 257–268.
- [80] K. Hung, C. Ko, A. Rawat, I. Chung, W. H. Hsu, and P. Chen, "Attention tracker: Detecting prompt injection attacks in llms," in *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, L. Chiruzzo, A. Ritter, and L. Wang, Eds. Association for Computational Linguistics, 2025, pp. 2309–2322.

- [81] Y. Kwon, F. Wang, W. Wang, K. H. Lee, W. Lee, S. Ma, X. Zhang, D. Xu, S. Jha, G. F. Ciocarlie, A. Gehani, and V. Yegneswaran, "MCI : Modeling-based causality inference in audit logging for attack investigation," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.
- [82] P. Gao, X. Xiao, Z. Li, F. Xu, S. R. Kulkarni, and P. Mittal, "AIQL: enabling efficient attack investigation from system monitoring data," in *Proceedings of the 2018 USENIX Annual Technical Conference, USENIX ATC 2018, Boston, MA, USA, July 11-13, 2018*, H. S. Gunawi and B. C. Reed, Eds. USENIX Association, 2018, pp. 113–126.
- [83] W. U. Hassan, M. Lemay, N. Aguse, A. Bates, and T. Moyer, "Towards scalable cluster auditing through grammatical inference over provenance graphs," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.
- [84] Y. Shen and G. Stringhini, "ATTACK2VEC: leveraging temporal word embeddings to understand the evolution of cyberattacks," in *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*. USENIX Association, 2019, pp. 905–921.
- [85] A. Aly, E. Mansour, and A. Youssef, "OCR-APT: Reconstructing APT stories from audit logs using subgraph anomaly detection and LLMs," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13–17, 2025*. ACM, 2025.
- [86] S. Cahyawijaya, H. Lovenia, and P. Fung, "Llms are few-shot in-context low-resource language learners," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*. Association for Computational Linguistics, 2024, pp. 405–433.
- [87] A. Hendy, M. Abdelrehim, A. Sharaf, V. Raunak, M. Gabr, H. Matsushita, Y. J. Kim, M. Afify, and H. H. Awadalla, "How good are GPT models at machine translation? A comprehensive evaluation," *CoRR*, vol. abs/2302.09210, 2023.
- [88] D. M. Alves, J. Pombal, N. M. Guerreiro, P. H. Martins, J. Alves, M. A. Farajian, B. Peters, R. Rei, P. Fernandes, S. Agrawal, P. Colombo, J. G. C. de Souza, and A. F. T. Martins, "Tower: An open multilingual large language model for translation-related tasks," *CoRR*, vol. abs/2402.17733, 2024.
- [89] D. M. Alves, N. M. Guerreiro, J. Alves, J. Pombal, R. Rei, J. G. C. de Souza, P. Colombo, and A. F. T. Martins, "Steering large language models for machine translation with finetuning and in-context learning," in *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*. Association for Computational Linguistics, 2023, pp. 11 127–11 148.
- [90] E. Wang, F. Cassano, C. Wu, Y. Bai, W. Song, V. Nath, Z. Han, S. Hendryx, S. Yue, and H. Zhang, "Planning in natural language improves LLM search for code generation," *CoRR*, vol. abs/2409.03733, 2024.
- [91] H. Koziolok, S. Grüner, R. Hark, V. Ashiwal, S. Linsbauer, and N. Eskandani, "Llm-based and retrieval-augmented control code generation," in *LLM4CODE@ICSE*, 2024, pp. 22–29.
- [92] B. Zhang, P. Liang, X. Zhou, A. Ahmad, and M. Waseem, "Practices and challenges of using github copilot: An empirical study," in *The 35th International Conference on Software Engineering and Knowledge Engineering, SEKE 2023, KSIR Virtual Conference Center, USA, July 1-10, 2023*. KSI Research Inc., 2023, pp. 124–129.
- [93] G. Dolcetti, V. Arceri, E. Iotti, S. Maffei, A. Cortesi, and E. Zafanella, "Helping llms improve code generation using feedback from testing and static analysis," *CoRR*, vol. abs/2412.14841, 2024.
- [94] N. Zahan, P. Burckhardt, M. Lysenko, F. Aboukhadijeh, and L. A. Williams, "Shifting the lens: Detecting malware in npm ecosystem with large language models," *CoRR*, vol. abs/2403.12196, 2024.
- [95] S. Fujii and R. Yamagishi, "Feasibility study for supporting static malware analysis using LLM," *CoRR*, vol. abs/2411.14905, 2024.
- [96] Y. Li, Z. Xiang, N. D. Bastian, D. Song, and B. Li, "Ids-agent: An llm agent for explainable intrusion detection in iot networks," in *NeurIPS 2024 Workshop on Open-World Agents*, 2024.
- [97] M.-T. Bui, M. Boffa, R. V. Valentim, J. M. Navarro, F. Chen, X. Bao, Z. B. Houidi, and D. Rossi, "A systematic comparison of large language models performance for intrusion detection," *Proceedings of the ACM on Networking*, vol. 2, no. CoNEXT4, pp. 1–23, 2024.
- [98] O. G. Lira, A. Marroquin, and M. A. To, "Harnessing the advanced capabilities of LLM for adaptive intrusion detection systems," in *Advanced Information Networking and Applications - Proceedings of the 38th International Conference on Advanced Information Networking and Applications (AINA-2024), Kitakyushu, Japan, 17-19 April, 2024, Volume 6*, ser. Lecture Notes on Data Engineering and Communications Technologies, vol. 204. Springer, 2024, pp. 453–464.
- [99] P. R. B. Houssel, P. Singh, S. Layeghy, and M. Portmann, "Towards explainable network intrusion detection using large language models," in *IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, BDCAT 2024, Sharjah, United Arab Emirates, December 16-19, 2024*. IEEE, 2024, pp. 67–72.
- [100] X. F. Guo, A. Meroño-Peñuela, S. Maffei, and F. Pierazzi, "Knowml: Improving generalization of ML-NIDS with attack knowledge graphs," *CoRR*, vol. abs/2506.19802, 2025.
- [101] C. Patsakis, F. Casino, and N. Lykousas, "Assessing llms in malicious code deobfuscation of real-world malware campaigns," *Expert Syst. Appl.*, vol. 256, p. 124912, 2024.
- [102] A. Happe and J. Cito, "Getting pwn'd by AI: penetration testing with large language models," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, December 3-9, 2023*. ACM, 2023, pp. 2082–2086.
- [103] G. Deng, Y. Liu, V. M. Vilches, P. Liu, Y. Li, Y. Xu, M. Pinzger, S. Rass, T. Zhang, and Y. Liu, "Pentestgpt: Evaluating and harnessing large language models for automated penetration testing," in *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*. USENIX Association, 2024.
- [104] B. Wu, G. Chen, K. Chen, X. Shang, J. Han, Y. He, W. Zhang, and N. Yu, "Autopt: How far are we from the end2end automated web penetration testing?" *CoRR*, vol. abs/2411.01236, 2024.
- [105] K. Swanson, W. Wu, N. L. Bulaong, J. E. Pak, and J. Zou, "The virtual lab: Ai agents design new sars-cov-2 nanobodies with experimental validation," *bioRxiv*, pp. 2024–11, 2024.
- [106] A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White, and P. Schwaller, "Chemcrow: Augmenting large-language models with chemistry tools," *CoRR*, vol. abs/2304.05376, 2023.
- [107] S. Schmidgall, Y. Su, Z. Wang, X. Sun, J. Wu, X. Yu, J. Liu, Z. Liu, and E. Barsoum, "Agent laboratory: Using LLM agents as research assistants," *CoRR*, vol. abs/2501.04227, 2025.
- [108] D. A. Boiko, R. MacKnight, B. Kline, and G. Gomes, "Autonomous chemical research with large language models," *Nature*, vol. 624, no. 7992, pp. 570–578, 2023.
- [109] A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White, and P. Schwaller, "Augmenting large language models with chemistry tools," *Nature Machine Intelligence*, vol. 6, no. 5, pp. 525–535, 2024.

Appendix A. Prompts

We list here the main agents' prompts (Prompt 1, 2, 3, and 4) we used throughout our implementation to produce our results. Prompt variables are denoted as '{variable}' and are substituted at runtime during agent instantiation.

Appendix B.

Evaluation of Attack Reports

To automate the process of evaluation, we utilize an LLM tasked with extracting the attack artifacts from an attack report generated by Chief Inspector. The LLM is given a JSON structure to follow (see Prompt 4). We parse the generated JSON using python and compile them into a malicious artifact list. We propagate this prediction as done during labeling, yielding a binary prediction array. Comparing this array against the ground-truth labels produces the final TP, FP, TN, and FN.

QA Agent Prompt

You are an SQL expert assigned to answer questions related to security incidents by querying an SQLite database. Break down the task into multiple simple SQLite queries instead of one complex query. Use only the provided tables and columns. Analyze the query results and provide a clear, concise answer based on the retrieved data. If the data is insufficient or the question cannot be answered, clearly explain why.

Schema: {schema}

Examples: {examples}

Reply with a tool call or your final answer to the question. You are allowed to perform {max_queries} queries. Although you will be given a specific question to answer, you have to report any abnormal behavior you find within the data, even if it is not related to the question.

Question: {question}

Prompt 1: QA agent prompt.

Investigator Agent Prompt

You are a security expert tasked with conducting a security investigation using provided data sources and analysis tools. These tools act as abstraction layers, allowing you to query log data directly. Your responsibility is to formulate precise and context-rich questions to effectively utilize these tools. Identify attack related artifacts, such as process names, their PIDs, files, network addresses, and domains used by the attackers. The evaluation of your report will be based on the accuracy and relevance of the identified artifacts. You will be presented with the logs of a single machine.

Guidelines:

- Clearly specify all relevant details within your questions (e.g., exact timestamps, IP addresses, process names). Do NOT assume tools are aware of contextual information about the investigation.
- You may perform up to {max_questions} queries.
- Use backward analysis (tracking events back in time), forward analysis (tracking subsequent activities), and correlation methods (e.g., timing, data volume) to identify entities related to the attack.
- Although you will be given a specific task to investigate, you have to report any suspicious activity you find, even if it is not related to the task.
- Conclude your investigation by summarizing findings clearly.

Environment: {environment}

Attack Lead: {initial_message}

Prompt 2: Investigator agent prompt.

Chief Inspector Prompt

You are a security expert tasked with analyzing a security incident on a single compromised machine. Your goal is to clearly identify the attack source, establish a timeline of attack events, determine attack objectives, and map the attack to the cyber kill chain. You will receive a message from monitoring team as a starting lead, environment context, and given access to the compromised machine logs. Your analysis must clearly list attack artifacts such as process names and their PIDs, Files involved, Network addresses (IPs) and attacker-controlled domains.

Instructions:

- This is an iterative process, with each investigation you will find new artifacts and leads, investigate each one thoroughly, leave no stone unturned.
- Be aggressive in your investigation, do not stop until you have inspected all attack artifacts you found and have exhausted all possible leads.
- You will be allowed to conduct a maximum of {max_investigations} investigations, you will be prompted to stop when we hit this limit.
- At the end, reflect on ALL the reports you received from your investigators and produce a final report.
- Your final report should include all details of the attack you found, including the starting point, attack vector, timeline, and objectives.
- Your final report will be evaluated based on accuracy, clarity, and correctness of identification of attack artifacts.
- For attack related processes, include their PIDs in the final report.
- When reporting domains, include IP addresses associated with them.
- Think and reflect on each report you receive, and then decide what to do next.

Environment: {environment}

Message from SOC: {initial_message}

Prompt 3: Chief Inspector agent prompt.

Evaluation Agent Prompt

You will be provided with a security investigation report. Your task is to **summarize the investigation findings for evaluation purposes**. Focus on **artifacts directly related to the attack**, such as **malicious websites, network addresses, files, or processes controlled by malicious actors**. Make a distinction between malicious (spawned and executed by malicious actor) and legitimate processes (existing benign processes, that were used in the attack), for the latter, include the time they were hijacked if possible.

Response Format: Return the summary in **JSON format** as a list of objects, with the structure below. Example:

```
{
  "addresses": ["10.20.30.40"],
  "domains": ["mal_domain.evil"],
  "files": ["mal_file.rtf"],
  "malicious_processes": [
    {"pid": 218, "name": "mall.exe"}
  ],
  "tainted_processes": [
    {"pid": 2191,
     "name": "word.exe",
     "hijack_time": "2019-08-09 00:00:58"}
  ],
}
```

Investigation Report:

{attack_report}

Prompt 4: Evaluation agent prompt.