OCR-APT: Reconstructing APT Stories from Audit Logs using Subgraph Anomaly Detection and LLMs

Ahmed Aly
Concordia University
Montreal, Quebec, Canada
ahmed.aly.20211@mail.concordia.ca

Essam Mansour
Concordia University
Montreal, Quebec, Canada
essam.mansour@concordia.ca

Amr Youssef
Concordia University
Montreal, Quebec, Canada
youssef@ciise.concordia.ca

Abstract

Advanced Persistent Threats (APTs) are stealthy cyberattacks that often evade detection in system-level audit logs. Provenance graphs model these logs as connected entities and events, revealing relationships that are missed by linear log representations. Existing systems apply anomaly detection to these graphs but often suffer from high false positive rates and coarse-grained alerts. Their reliance on node attributes like file paths or IPs leads to spurious correlations, reducing detection robustness and reliability. To fully understand an attack's progression and impact, security analysts need systems that can generate accurate, human-like narratives of the entire attack. To address these challenges, we introduce OCR-APT, a system for APT detection and reconstruction of human-like attack stories. OCR-APT uses Graph Neural Networks (GNNs) for subgraph anomaly detection, learning behavior patterns around nodes rather than fragile attributes such as file paths or IPs. This approach leads to a more robust anomaly detection. It then iterates over detected subgraphs using Large Language Models (LLMs) to reconstruct multi-stage attack stories. Each stage is validated before proceeding, reducing hallucinations and ensuring an interpretable final report. Our evaluations on the DARPA TC3, OpTC, and NODLINK datasets show that OCR-APT outperforms state-of-the-art systems in both detection accuracy and alert interpretability. Moreover, OCR-APT reconstructs human-like reports that comprehensively capture the attack story.

CCS Concepts

• **Security and privacy → Intrusion detection systems**.

Keywords

Anomaly Detection, APT Attack Investigation, LLMs, GNNs

1 Introduction

Advanced Persistent Threats (APTs) are among the most insidious forms of cyberattacks. Characterized by stealth, persistence, and adaptability, APTs often evade traditional security mechanisms by exploiting zero-day vulnerabilities and maintaining long-term access through low-profile tactics [3]. As a result, detecting and reconstructing these attacks from system-level audit logs remains a significant challenge for security analysts. Provenance graphs—structured representations of system logs that encode interactions between processes, files, and network entities—offer a promising way to visualize the causal relationships between system

activities [56, 90]. However, the complexity and size of these graphs make human analysis infeasible without intelligent automation.

Security analysts not only require systems that can detect suspicious behaviors but also demand tools that support forensic investigation by reconstructing the complete attack story. Such reconstructions must provide interpretable, human-like reports that map to APT attack stages. Existing systems typically fall short: they generate fragmented outputs or overly technical graphs that are difficult to parse and interpret. This gap motivates the need for more robust and intelligible solutions that go beyond isolated alerts and provide comprehensive insights into how an attack unfolded.

Limitations of Existing Detection Methods: Most prior efforts in APT detection using provenance data fall into two broad categories: heuristic-based and anomaly-based approaches [39]. Heuristic methods rely on signatures or rules derived from known attacks [30, 59], but fail to detect novel threats. Anomaly-based methods, in contrast, identify deviations from expected behavior and thus hold greater promise for detecting zero-day attacks [39, 77]. However, they frequently suffer from high false positive rates [29, 70], producing voluminous alerts that burden security teams with triage tasks. Furthermore, anomaly-based systems often operate at the node level [40, 77, 83] or over the entire graph [29, 41, 57], which creates practical limitations. Node-level alarms lack contextual information, making it difficult to interpret isolated anomalies. Graph-level alarms, on the other hand, are too coarse, obscuring the specific sequences and entities involved in an attack. Recent efforts [46, 70] have shifted toward subgraph-based anomaly detection to strike a balance between granularity and interpretability. These systems identify small, connected sets of anomalous nodes to support better investigation. However, they often rely heavily on fragile node attributes like file paths or IPs, which are easy to obfuscate or manipulate, thereby reducing detection robustness [8, 60].

Challenges in Attack Story Reconstruction: Beyond detection, reconstructing a coherent and human-understandable attack story remains a major unsolved challenge. Many existing systems [2, 23, 80] assume prior knowledge in the form of Points-of-Interest (POIs), such as manually flagged alerts or indicators. This reliance hinders comprehensive log analysis. Moreover, the outputs of these systems often consist of dense graphs or low-level event sequences that lack narrative clarity. Without proper summarization or contextual linking of events to known attack stages—such as those in the MITRE ATT&CK or APT kill-chain frameworks—these systems fail to serve the needs of analysts conducting forensic investigations.

Our Approach: To address the above challenges, we propose OCR-APT[1], a novel system that performs end-to-end reconstruction of APT stories from audit logs. OCR-APT consists of two key components: a GNN-based subgraph anomaly detector, and an LLM-based attack investigator that generates interpretable attack stories. The subgraph anomaly detector leverages a custom graph learning model, OCRGCN, which integrates relational graph convolutional networks (RGCNs) with one-class SVMs. This design captures behavioral patterns over structural relationships, allowing the system to detect anomalies based on context rather than brittle attributes. By training a separate model per node type, OCRGCN identifies abnormal interactions with higher precision.

Detected anomalous nodes are then grouped into subgraphs based on topological and behavioral coherence. Each subgraph is scored for abnormality and filtered to retain those with high investigative value. These subgraphs serve as the basis for the second component: the attack investigator. This module applies a Retrieval-Augmented Generation (RAG) approach to serialize the subgraphs and pass them to a Large Language Model. By modularizing the reconstruction process into validated subtasks, OCR-APT mitigates common issues like LLM hallucination [69]. The output is a structured, stage-wise attack report that identifies Indicators of Compromise (IOCs) and maps events to the APT kill-chain [59].

Impact and Evaluation. We evaluate OCR-APT on three provenance graph datasets: DARPA TC3 [18], OpTC [19], and NODLINK [46]. DARPA TC3 and OpTC are widely recognized benchmarks that reflect realistic, enterprise-scale APT scenarios [90]. The NODLINK dataset provides a controlled simulation environment for multi-stage APTs, enabling direct comparison with state-of-the-art subgraph-based anomaly detection systems. Experimental results demonstrate that OCR-APT consistently outperforms existing systems in both anomaly detection accuracy and the interpretability of generated alerts. Specifically, OCR-APT achieves an average F1-score of 0.96, outperforming both NODLINK (0.248) and FLASH (0.945), thereby advancing the state of subgraph-based detection. Its performance is also comparable to or exceeds that of node-level and time window-based anomaly detection methods.

Beyond quantitative gains, OCR-APT advances usability by producing concise, human-readable attack reports that reconstruct a majority of the APT kill-chain stages. This capability bridges the gap between low-level system telemetry and high-level analyst reasoning. By combining graph learning with natural language generation, OCR-APT delivers not only accurate detections but also actionable insights—streamlining alert triage, reducing investigation time, and pushing the state of the art in APT detection and analysis. Our contributions can be summarized as follows:

- We propose a GNN-based anomaly detection model combined with a one-class classification to accurately identify anomalous nodes and APT-related subgraphs in provenance graphs.
- We introduce an LLM-driven investigation method that reconstructs attack stories from audit logs and generates concise, human-like reports.

- We integrate these components into OCR-APT[2], a complete APT detection and investigation system that identifies anomalies, ranks alerts by severity, and produces interpretable reports to support efficient analyst workflows.
- We conduct extensive evaluations on DARPA TC3, OpTC, and NODLINK datasets. OCR-APT outperforms state-of-the-art anomaly detection systems and successfully reconstructs multi-stage, human-like APT reports.

The remainder of the paper is organized as follows: Section 2 reviews background and limitations of existing systems; Section 3 defines the threat model. Section 4 introduces our system, with Sections 5 and 6 detailing the GNN-based detector and LLM-based attack investigator, respectively. Evaluation results are in Section 7, related work in Section 8, and conclusions in Section 9.

2 Background

Provenance graphs (PGs) are directed, heterogeneous graphs that model audit logs to support causal analysis [56, 90]. They provide a comprehensive view of system activities and information flow, making them effective for uncovering attack traces [39]. A PG consists of diverse node types—such as processes, files, and network flows—linked by edges representing actions like read, write, and execute. The exact schema depends on the underlying operating system; our approach leverages all available node and edge types per host. For example, Appendix A outlines the schema used for FreeBSD-based hosts (CADETS). PGs also include event timestamps, crucial for detecting APTs and reconstructing attack timelines [70], as well as contextual node attributes like command-line arguments, file paths, and IP addresses. These features enrich the analysis of system behavior.

2.1 Limitations of Anomaly Detection Systems

Anomaly-based detection systems learn patterns of normal system behavior and flag deviations as potential threats. In provenance graph analysis, early approaches often detect anomalies at the granularity of entire graphs using clustering [29, 57] or graph classification techniques [36, 41]. However, these methods struggle with interpretability: the alarms often span PGs with millions of nodes [39], despite only a small subset being relevant to the attack [46, 70]. This makes the investigation akin to "searching for a needle in a haystack." Additionally, such coarse-grained approaches risk missing fine-grained anomalies, leading to false negatives [77].

To improve granularity, subsequent systems focus on paths [49, 76]. While more precise, these techniques often lose broader attack context [30]. More recent advancements target even finer units—individual nodes [40, 77, 83], edges [87], time windows [16], or subgraphs [46, 70]. Although these methods enhance interpretability, node- and edge-level systems can overwhelm analysts with numerous isolated alerts lacking contextual history.

Subgraph-based systems like NODLINK [46] attempt to offer better context by constructing coherent attack graphs using Steiner Trees [37, 38]. However, NODLINK's reliance on sentence embeddings [11] limits precision, often resulting in false positives. Moreover, NODLINK relies on node attributes features that may introduce spurious correlations—a common issue in cybersecurity ML,

---

where models learn artifacts (e.g., specific IP ranges) instead of generic attack patterns [8]. FLASH [70] and KAIROS [16] follow similar strategies, using GNNs with node attributes such as process names, command-line arguments, file paths, and IP addresses to inform embeddings. While these semantic features improve detection accuracy, they are also vulnerable to adversarial manipulation, as attackers can change surface-level attributes without altering attack behavior [60]. To counteract this vulnerability, OCR-APT takes a novel approach by avoiding reliance on node attributes. Instead, it uses structural and behavioral features to strengthen robustness against evasion tactics. This strategy ensures consistent anomaly detection, making OCR-APT more accurate and reliable than existing subgraph-level systems, which still struggle with false positives and adversarial manipulation.

Methodological Limitations. Current anomaly detection methods commonly rely on two paradigms: autoencoders [16, 40, 46, 83] and node-type classification [70, 77]. Autoencoders are memory-intensive due to the need to reconstruct large adjacency matrices [52]. In node-type classification, a node is flagged as anomalous if its predicted type (e.g., process, file, network flow) differs from the expected one. However, this assumption does not always hold, as each node type exhibits distinct behavioral patterns. For example, process nodes perform distinct actions that reveal their type, so a malicious process may still be correctly classified and evade detection. OCR-APT mitigates this issue by avoiding type-based classification. Instead, it directly classifies nodes as normal or anomalous based on their behavioral patterns, using a one-class SVM [12]. This one-class classification approach identifies outliers without relying on labeled attack data, enabling the detection of previously unseen attack behaviors.

Moreover, prior works [70, 77] employ GNN models originally designed for homogeneous graphs, such as GraphSAGE [28], which do not consider edge types (i.e., node actions) during embedding computation. As a result, they overlook critical structural context. While GNNs have been actively explored for anomaly detection, the heterogeneous nature of PGs remains underexplored [42]. OCR-APT addresses this gap by using RGCNs [72] to embed nodes while preserving the heterogeneous structure of PGs. Unlike previous methods, OCR-APT incorporates node actions directly into node embeddings, which allows it to account for complex relationships between nodes in attack scenarios. Section 7.4.2 compares OCR-APT with other GNN-based baselines, demonstrating its superior performance in terms of both precision and recall.

Efficiency Considerations. Scalability is critical for handling large-scale enterprise provenance data. While prior systems propose graph reduction and subgraph extraction techniques [4, 5, 33], OCR-APT introduces a memory-efficient approach tailored specifically to anomaly detection. Instead of extracting graphs from known IOCs, OCR-APT constructs causally relevant subgraphs around detected anomalous nodes using three efficient graph queries. This method avoids the need to load the entire PG into memory, which is essential for supporting deployment in resource-constrained environments and ensuring scalability.

2.2 Limitations of Attack Investigation Systems

Attack investigation systems support post-alert analysis by helping security analysts validate threats and understand the attacker's actions [39]. Key challenges include triaging high-priority alerts [31, 32], clustering related alerts [74, 86], and reconstructing comprehensive attack stories from low-level logs [2, 16, 23, 46, 80].

Many reconstruction systems rely on pre-identified POIs [23, 74, 80] or known attack entities [2] as seeds for investigation. This dependence limits generalization: if the initial POI is inaccurate, the derived attack story may be misleading. For instance, ATLAS [2] trains an LSTM-based model on simulated attack sequences. If the starting entity is misclassified, the entire reconstruction may be compromised. Such systems are often unable to detect novel threats outside the scope of their training data. Rule-based systems [33, 59] suffer similar drawbacks, as predefined patterns can only capture known attacks. They are ineffective against polymorphic APTs that exhibit diverse behaviors [29, 39, 77].

Narrative Complexity. Most existing tools generate either attack graphs [9, 33, 46, 59, 68, 80] or attack sequences [2, 23]. While informative, these representations are often complex and hard to interpret, requiring significant analyst effort to extract key insights. Graphs, in particular, pose challenges for visualization and manual inspection [16]. Some approaches focus on alert clustering [54, 74, 86], grouping similar alerts to reduce manual workload. However, they typically lack narrative coherence and contextual depth, which are essential for understanding multi-stage APTs. In contrast, OCR-APT introduces a novel LLM-based module that generates high-quality, human-like attack reports, offering coherent summaries of attack behavior. These reports not only reduce analyst burden but also improve the effectiveness and speed of APT investigations by providing a clear, interpretable story that can easily guide further analysis and response.

3 Threat Model

This study focuses on detecting APTs characterized by a "low and slow" attack approach [29]. While our threat model acknowledges that attackers may use sophisticated zero-day exploits to compromise the system, we assume they must leave distinguishable traces in the system logs. The proposed system requires system logs that are free from attack traces for training. Consistent with previous work, we consider audit logs and kernel-space auditing frameworks as part of our trusted computing base [2, 5, 40, 59, 87]. Attacks involving data poisoning, hardware-level attacks, and side-channel attacks are beyond the scope of this study.

4 Proposed System Architecture

This section presents the architecture of our OCR-APT system, shown in Figure 1. Each component of OCR-APT is designed to address key limitations identified in existing anomaly detection and attack investigation systems (Section 2.1), and together they achieve our research objectives: fine-grained anomaly detection, resistance to adversarial evasion, scalability to enterprise-scale data, and automated, interpretable threat investigation.
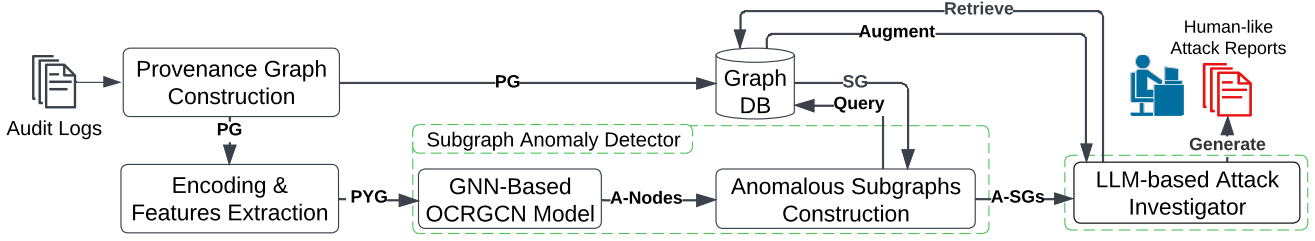
Figure 1: Overall architecture of OCR-APT. This includes constructing provenance graphs (PG), extracting features and encoding the graph into a PyTorch Geometric data object (PYG), detecting anomalous nodes (A-Nodes) with our GNN-based model (OCRGCN), identifying anomalous subgraphs (A-SGs), and generating a human-like APT attack report using LLMs.

To address the scalability and memory-efficiency limitations of existing systems, OCR-APT represents audit logs as an RDF-based[3] provenance graph (PG) and loads them incrementally into an RDF graph database using a disk-backed ingestion strategy [5]. The RDF model encodes system activity as triples of the form ⟨subject, predicate, object⟩, where predicate denotes an event, and subject and object are system entities. This representation enables scalable graph construction and efficient query-based subgraph extraction, overcoming the limitations of systems that require the entire graph to be held in memory.

To address the interpretability challenges of coarse-grained anomaly detection, OCR-APT extracts behavior-based features for each node, including actions, effects, and timing statistics. Normalization ensures consistency across entities, supporting generalized behavior learning. The graph is then encoded for GNN-based modeling. The encoded graph is passed to the subgraph anomaly detector, which overcomes limitations of prior methods, such as reliance on attribute embeddings or autoencoders, by combining node-level anomaly detection with context-aware subgraph analysis. Using OCRGCN, a one-class GNN trained on benign data, it identifies anomalous nodes based on structural and behavioral features without labeled attacks. To improve interpretability and reduce alert fatigue, OCR-APT constructs causally coherent subgraphs around detected anomalies using efficient SPARQL queries. Each subgraph is scored, and those above a threshold are flagged, providing precise and context-rich alerts without static rules.

To support effective post-alert investigation and narrative reconstruction, anomalous subgraphs are passed to the attack investigator module, which uses LLMs to generate concise, human-readable attack reports. It serializes each subgraph into a timestamp-ordered description, summarizes it via LLM prompts to extract IOCs, key actions, and APT kill chain stages, and then composes a complete attack narrative. This report is further enriched through a retrieval-augmented generation (RAG) pipeline that queries the graph for additional context. By integrating precise anomaly detection with automated investigation, OCR-APT reduces false positives, enhances interpretability, and scales to large, heterogeneous provenance graphs—effectively addressing the core challenges outlined in our research objectives.

---
[3]RDF: Resource Description Framework graph model [71]

5 GNN-based Subgraph Anomaly Detection

This section introduces the subgraph anomaly detector, outlining the OCRGCN architecture and subgraph construction algorithm.

5.1 The GNN-based model

The GNN-based model acts as the core component of OCR-APT's subgraph anomaly detection pipeline. Figure 2 illustrates the model's architecture, highlighting its training and inference phases.

5.1.1 The model architecture. Our OCRGCN uses an RGCN-based architecture [72] to capture both graph structure and node behavior by incorporating edge types during the embedding aggregation process. To prevent spurious correlations [8], OCRGCN avoids using node attributes such as IP addresses or file paths, which can cause the model to memorize specific malicious instances rather than learn generalizable attack patterns. While excluded from the model's input, these attributes are retained for the investigation phase, where they assist analysts in interpreting and verifying alerts. Each layer of the model aggregates information from a node's one-hop neighborhood. Nodes exchange messages with their neighbors, embedding information about node types, actions, and initial extracted features, and then update their embeddings based on the aggregated data. After multiple RGCN layers, the model produces a final embedding vector for each node in the provenance graph. These embeddings are then passed to a one-class SVM, which learns a hypersphere that encloses the majority of normal node embeddings. An anomaly score is computed based on the distance of each node's embedding from the center of this hypersphere. Nodes whose scores exceed the hypersphere's radius are flagged as anomalous.

5.1.2 Training Phase. The training begins by extracting behavior-based features from benign provenance graphs, including action frequencies and idle period statistics. Action frequencies reflect behavioral tendencies by computing the proportion of each action type relative to the total number of actions per node. Unlike prior work that uses raw action counts [77], we apply L2 normalization to reduce bias from high-activity benign nodes and allow the model to focus on behavioral patterns (i.e., scaling the action frequency vector so that the sum of squared values equals one). For example, while a frequently used browser may establish many connections, its overall behavior is normal when considering the ratio of connections to other actions like sends, receives, reads, and writes.
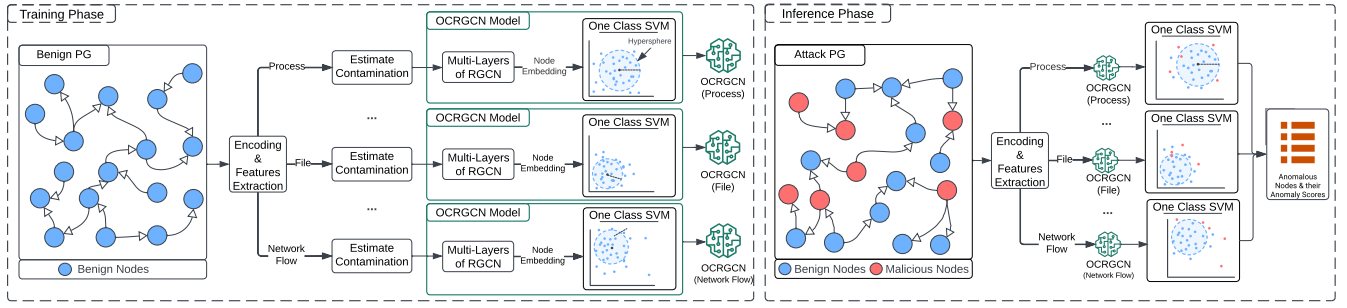
Figure 2: Architecture of the OCRGCN model. The training phase (left) involves encoding benign provenance graphs, estimating contamination factors, learning node embeddings via RGCN layers, and learning the normal hypersphere with a one-class SVM. The inference phase (right) uses the trained models to compute anomaly scores and detect anomalous nodes.

Conversely, a process with an unusually high rate of sending or execution may indicate suspicious behavior. Idle period statistics are derived from event timestamps and include minimum, maximum, and average durations between actions. These statistics are normalized to a 0–1 range using a min-max scaler, based on the dataset's minimum and maximum values. APT-related nodes tend to remain idle longer than benign nodes, making idle period statistics a key indicator. These two features capture key aspects of node behavior and assist in detecting malicious patterns. Importantly, OCR-APT does not rely solely on these raw features; its GNN aggregates them within the graph structure, enabling node representations to reflect both behavioral patterns and structural context. We conducted additional experiments to evaluate alternative temporal features, but ultimately excluded them due to poor generalization across hosts; further details are provided in Appendix E.

After feature extraction, the system splits nodes by type and trains a specific OCRGCN for each type to improve detection accuracy, as normal behavior varies across node types. Each OCRGCN is trained on a single node type but aggregates messages from all neighboring types, preserving cross-type semantics. Following this, each model learns a hypersphere specific to a given node type, enabling anomaly detection tailored to the normal behavior of that type without losing cross-type interaction information. Distinct models also enable precise estimation of the contamination factor, which represents the expected proportion of anomalies. This factor is estimated as the proportion of malicious nodes in the validation set, constrained between Min_{con} and Max_{con}. The maximum constraint ensures the contamination factor aligns with the stealthy nature of APTs. If the validation set contains many malicious nodes, the contamination factor is set to Max_{con}. The minimum constraint ensures the factor is above zero, even when no malicious nodes are present. If no labeled data is available, the system uses Min_{con} as the contamination factor, relying solely on trusted benign logs reflecting normal behavior.

Each OCRGCN model learns a hypersphere that encloses most normal nodes for a specific type and computes the anomaly score threshold based on the contamination factor. The fraction of training nodes allowed outside the hypersphere is controlled by the hyperparameter β, fixed across all node types. If too many nodes are enclosed, the hypersphere becomes too large, reducing its ability

to detect anomalies. The goal is to capture the norm of benign nodes without overfitting. During training, the RGCN model updates its weights to bring normal nodes closer together in the embedding space, while the one-class SVM adjusts the hypersphere's center and radius to fit the normal node embeddings. Training stops early based on the validation set's F1-score, and if no malicious nodes are present, training halts when the true negative rate declines.

5.1.3 Inference Phase. During inference, the system processes provenance graphs containing both benign and malicious traces. It applies the same pre-processing steps to extract features and assigns each node to its corresponding OCRGCN model based on its type. The OCRGCN models compute anomaly scores for the test nodes, and those exceeding the pre-computed threshold are classified as anomalous. Finally, these anomalous nodes and their scores are passed to the subgraph construction module.

5.2 Anomalous Subgraph Construction

OCR-APT constructs anomalous subgraphs using Algorithm 1. The algorithm takes as input a set of anomalous nodes, a connection to the provenance graph database, and two parameters: n_{seed}, which specifies the number of seed nodes for each node type, and max_e, the maximum number of edges allowed in each subgraph. It begins by querying the graph database to retrieve direct connections between anomalous nodes and their one-hop neighbors to form an initial subgraph (lines 3-5). This is done with three SPARQL queries: one for direct edges between anomalous nodes, and two for their neighboring nodes and edges, minimizing traversal overhead.

The nodes are next ranked by anomaly scores, and the top n_{seed} nodes of each type are selected as seeds (lines 6–7). For each seed, the algorithm performs a 1-hop bidirectional traversal (line 9), connecting anomalous nodes through intermediate normal ones to preserve their context. It then retains only the paths that lead to anomalous nodes and constructs a candidate subgraph (lines 10–11), limiting the inclusion of benign nodes. Figure 3 illustrates this process: it begins with individual anomalous nodes, links them via direct connections when possible, expands each by one hop to capture surrounding context, and then prunes paths that do not lead to additional anomalies. The figure shows how anomalous nodes (in red and orange) are connected through normal nodes (in blue), with misclassified nodes (in brown) also identified.
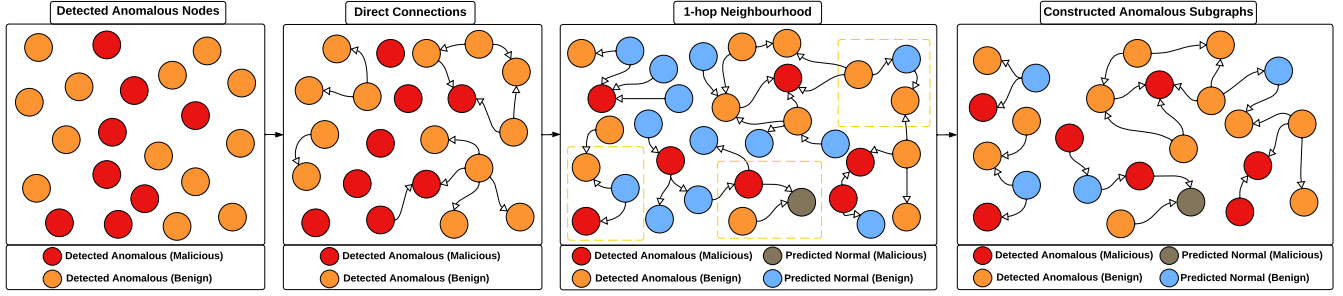
Figure 3: The stages of constructing anomalous subgraphs. OCR-APT starts from anomalous nodes, connects them by direct connections, gets all one-hop neighbor nodes, and keeps only neighbors that lead to other anomalous nodes.

Algorithm 1 Anomalous Subgraphs Construction Algorithm

1: **Input:** Provenance Graph Database (DB_{PG}), Anomalous Nodes (A_{Nodes}), n_{seed}, max_e
2: **Output:** Anomalous Subgraphs (A_{SGs})
3: Query direct connections between A_{Nodes} from DB_{PG}
4: Query 1-hop neighbors of A_{Nodes} from DB_{PG}
5: Construct an initial subgraph ($init_{SG}$)
6: Sort A_{Nodes} based on their anomaly scores
7: Identify $Seeds$ as top n_{seed} A_{Nodes} per node type
8: **for** every $Seeds$ **do**
9:     Traverse $init_{SG}$ for 1-hop forward and backwards
10:     Keep only paths that lead to unvisited A_{Nodes}
11:     Construct a subgraph (sg)
12:     **if** $sg_{edges} <= Max_e$ **then**
13:         Add sg to to the Anomalous Subgraphs (A_{SGs}) list
14:     **else**
15:         Partition sg into smaller subgraphs within Max_e
16:         Add partitioned subgraphs to A_{SGs}
17:     **end if**
18: **end for**
19: Filter out identical subgraphs in A_{SGs}
20: **for** every A_{SGs} **do**
21:     Compute the subgraph anomaly score
22:     Determine the subgraph abnormality level (sg_{ab})
23: **end for**
24: Filter out subgraphs with minor sg_{ab} from A_{SGs}

Once a candidate subgraph is formed, it is either added directly to the set of anomalous subgraphs if it stays within the edge limit max_e, or partitioned into smaller subgraphs (lines 12–17). The Louvain community detection algorithm [10] is used for partitioning. This ensures manageable subgraph sizes for analysis and helps the LLM-based attack investigation maintain narrative coherence. Partitioning does not affect investigation quality, as the LLM-based investigator summarizes each partition individually and merges them into a comprehensive attack report. Some cross-partition edges may be omitted, but the overall attack scenario remains intact. After processing all seed nodes, duplicate subgraphs are removed (line 19) and an anomaly score is computed for each subgraph (line 21). This score is the sum of the scores of its anomalous nodes. The subgraphs are then mapped to abnormality levels using a logarithmic

scale (line 22), and those with low abnormality are filtered out (line 24), reducing false positives. The final set of anomalous subgraphs can be adjusted based on the desired abnormality threshold for further investigation.
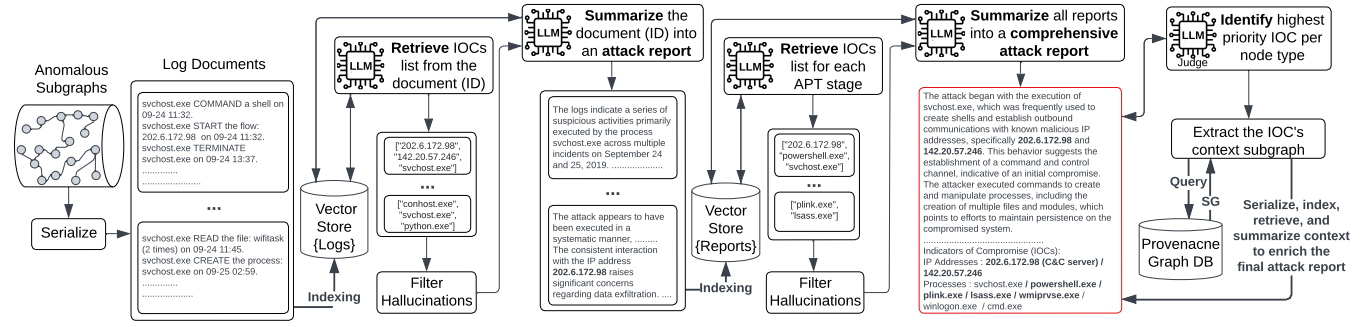
6 LLM-Based Attack Investigation

6.1 The Limits of LLMs in Attack Investigation

To enable LLM-based attack investigation, we examined how LLMs perform when tasked with reconstructing attack stories from system audit logs. This reconstruction process demands high-level reasoning, contextual understanding, and the interpretation of subtle event patterns. We found that LLMs struggle to generate high-quality of human-like reports when asked to perform this task in a single step. Our initial approach used a monolithic prompt to generate full reports directly from anomalous subgraphs, but this often led to hallucinated content, overlooked APT stages, and missing IOCs. These limitations persisted even when evaluating on benchmark datasets that may have been seen during pretraining. This shows the inherent difficulty of complex and analyst-level investigative tasks when attempted all at once.

To address these challenges, we first incorporated Chain-of-Thought (CoT) prompting [79], embedding explicit reasoning steps to help the model logically interpret each subgraph. This improved the coherence of the report and increased the coverage of the APT stage, but hallucinations and factual errors remained. We then designed a multi-stage prompting pipeline that decomposes the investigation into smaller, well-defined subtasks, such as IOC extraction, APT stage mapping, and context summarization. Each stage employs a focused CoT-based prompt, enhanced by in-context learning with domain-specific instructions and CTI concepts. This enables the LLM to reason more effectively within a constrained scope.

Building on these insights, we designed a complete LLM-driven attack investigation mechanism, called attack investigator. It implements our multi-stage prompting strategy within a Retrieval-Augmented Generation (RAG) framework. Each stage uses tailored CoT-based prompts and is connected by an automatic validation mechanism that ensures consistency and preserves report integrity. This design not only improves investigative performance by capturing more attack stages, but also fully mitigates hallucinations observed in the earlier approach. This modular pipeline, with explicit reasoning and automatic validation, improves the overall

Figure 4: Architecture of the LLM-based attack investigator. The system serializes anomalous subgraphs into log documents, indexes them in a vector store, and uses an LLM to generate attack reports. It identifies key IOCs, enriches reports with context subgraphs, and produces a comprehensive report for analysts. The visualized reports are simplified versions of the recovered report from host 501 in the DARPA OpTC dataset.

quality and completeness of the generated attack reports. Hence, our approach enables reliable and context-aware attack report generation that accelerates attack investigations.

6.2 Our Attack Investigator Mechanism

We have designed the attack investigator to reconstruct attacks via a RAG-based pipeline consisting of six stages, as shown in Figure 4. The pipeline transforms detected anomalous subgraphs into human-like attack reports, which capture the main stages of the attack story. In the first stage, anomalous subgraphs are serialized into event log documents and indexed in a vector store. These serialized logs provide a structured representation of the subgraphs for further processing. Stage two involves using an LLM to extract IOCs from each serialized subgraph stored in the vector store. To prevent hallucinations, the system validates the extracted IOCs by checking whether they appear in the corresponding anomalous subgraphs. This ensures that only verified IOCs are retained for subsequent stages. In stage three, the LLM generates an attack report for each subgraph based on the validated IOCs. These reports are indexed into a separate vector store, making them easily retrievable. The final comprehensive attack report is reconstructed in stages four and five. In stage four, the LLM extracts a list of IOCs for each APT stage from all generated attack reports. These individual reports are merged into a comprehensive attack report, as shown in the red box in Figure 4.

The final stage enriches the comprehensive report by iterating over the most critical IOCs. The system employs a mechanism called llm-as-a-judge [89] to identify the most significant IOCs. The system then queries the provenance graph database to retrieve the contextual information for each identified IOC, which is in the form of connected anomalous subgraphs. Each subgraph provides crucial attack context, which is integrated into the comprehensive report. This process enhances the detection of additional APT stages.

6.3 Attack Report Generation

The attack investigator generates attack reports using Algorithm 2, which takes anomalous subgraphs as input and produces a comprehensive narrative that reconstructs the attack story at the subgraph

---

Algorithm 2 Attack Reports Generation Algorithm

1: **Input:** Provenance Graph Database (DB_{PG}), Anomalous Subgraphs (ASG_s)
2: **Output:** Attack Reports (R_{atk}), Comprehensive Attack Report (R_{comp})
3: Serialize ASG_s into log documents (ASG_{docs})
4: Index ASG_{docs} in a vector store for logs (VSt_{logs})
5: Initialize LLM chat engine (LLM_{chat}) with instructions
6: **for** each ASG_{doc} in ASG_{docs} **do**
7:     Extract IOC_{lst} using LLM_{chat} from VSt_{logs}
8:     Filter hallucinations in IOC_{lst}
9:     Summarize ASG_{doc} using LLM_{chat}, append into R_{atk}
10:     Reset LLM_{chat} memory
11: **end for**
12: Index R_{atk} in a vector store for reports (VSt_R)
13: Extract IOC_{lst} per APT stage from R_{atk} using LLM_{chat}
14: Filter hallucinations in IOC_{lst}
15: Summarize R_{atk} into R_{comp} using LLM_{chat}
16: Initialize LLM as a judge (LLM_{judg}) with instructions
17: **for** each $node_{\mathrm{type}}$ in ['IP', 'PROCESS', 'FILE'] **do**
18:     Prompt LLM_{judg} to select most critical IOC in R_{comp}
19:     Query DB_{PG} to retrieve IOC context
20:     Serialize IOC context and index it in VSt_{logs}
21:     Extract IOC_{lst} per APT stage using LLM_{chat}
22:     Filter hallucinations in IOC_{lst}
23:     Summarize context using LLM_{chat}, append into R_{atk}
24:     Enrich R_{comp} with the report using LLM_{chat}
25: **end for**

---

level. Each detected anomalous subgraph represents a fragment of the broader attack scenario. The algorithm begins by serializing each subgraph into a chronological sequence of events (line 3). This process converts the subgraph's edges into natural language sentences that encode subject and object attributes, the action performed, and the associated timestamps. During serialization, a reduction phase condenses duplicate actions occurring within one-second intervals into a more compact representation. For instance, if a process repeatedly reads the same file, the output records the

action once, noting the number of repetitions (e.g., "read X times"). Timestamps are simplified from microseconds to seconds to reduce token overhead and streamline the LLM input.

The serialized subgraph (log document) is segmented into sentence chunks. Embeddings for each chunk are computed and indexed into a vector store (line 4). This enables the LLM to efficiently retrieve relevant context from log documents. We use the 'text-embedding-3-large' model [65] for indexing, due to its superior performance [15]. The algorithm then configures the LLM with domain-specific instructions for attack investigation (line 5). These instructions guide the model's reasoning by narrowing its focus to concepts from operating system security, Cyber Threat Intelligence (CTI), and APT kill-chain stages. They also define key terms, such as IOCs and APT stages, emphasize the importance of avoiding hallucinations, and require the model to produce high-quality, human-like narratives. The task is framed as summarizing detection alerts into concise reports that include: (1) a summary of attack behavior, (2) a breakdown of APT stages, (3) identified IOCs with context, and (4) a minute-by-minute action log. The full set of instructions and prompts is provided in Appendix B.

To enhance performance, the reconstruction process is modularized into subtasks, enabling the LLM to focus on one task at a time. First, the model extracts IOCs from the serialized document (ASG_{doc}) using a dedicated prompt (p_{ioc}) as shown in Equation 1:

$$\{IOC_i\} = f(ASG_{\text{doc}}, p_{\text{ioc}}) \tag{1}$$

This modular design allows the system to validate extracted IOCs and filter out hallucinations. Specifically, any IOCs not found in the source document are excluded (line 8). The validated IOCs ($\{IOC_i\}'$) are then used to guide the LLM in generating an attack report (R_{atk}) using prompt p_{sum} (line 9), as shown in Equation 2. After generating each report, the system resets the LLM's memory to avoid cross-document contamination (line 10).

$$R_{\text{atk}} = f(ASG_{\text{doc}}, \{IOC_i\}', p_{\text{sum}}) \tag{2}$$

Once all subgraphs have been processed, the reports (R_{atk}) are indexed, and the LLM is prompted to extract IOCs per APT stage (stg), supporting the creation of a unified attack report (R_{comp}) (lines 12–15):

$$\{IOC_{\text{stg}}^{(i)}\} = f(\{R_{\text{atk}}^{(i)}\}, stg, p_{\text{ioc,stg}}) \tag{3}$$

$$R_{\text{comp}} = f(\{R_{\text{atk}}^{(i)}\}, \{IOC_{\text{stg}}^{(i)}\}', p_{\text{comp}}) \tag{4}$$

To further enrich R_{comp}, the system applies a RAG-based process. It begins by initializing an LLM "judge" guided by expert-level instructions (line 16). This step enables the fully automated pipeline to evaluate the generated content. The judge LLM selects the most critical IOC per node type using p_{judg} (lines 17–18):

$$IOC = f(R_{\text{comp}}, p_{\text{judg}}) \tag{5}$$

The system then queries the provenance graph to extract subgraphs centered on the selected IOCs (line 19). Graph traversal is limited to one-hop anomalous nodes, filtering out benign context—especially relevant when attackers use "living-off-the-land" techniques by exploiting legitimate system processes. Since such nodes can produce excessive benign context, the filter helps keep

Table 1: Statistics of DARPA TC3, DARPA OpTC, and Simulated NODLINK datasets

Dataset	Host	# Nodes	# Edges	# Nodes Types	# Edges Types	# Malicious Nodes
DARPA TC3	CADETS	696.37 K	8.66 M	6	28	12.81 K
	TRACE	2.48 M	6.98 M	11	24	67.38 K
	THEIA	642.56 K	18.82 M	4	18	25.32 K
DARPA OpTC	201	788.24 K	5.84 M	8	20	71
	501	1.14 M	8.29 M	8	20	418
	51	720.40 K	4.98 M	7	19	200
Simulated NodLink	Ubuntu	23.04 K	14.04 M	3	13	21
	WS12	10.86 K	8.27 M	3	6	47
	W10	62.16 K	7.89 M	3	6	191

the investigation focused. Finally, the extracted context subgraphs (IOC_{ctx}) are indexed, and used to augment the comprehensive report using prompt p_{aug} (lines 20–25):

$$R'_{\text{comp}} = f(R_{\text{comp}}, IOC_{\text{ctx}}, p_{\text{aug}}) \tag{6}$$

The final report, R'_{comp}, offers an accurate reconstruction of the attack story. Analysts can interact with the system by posing follow-up questions to the LLM, such as assessing the security context of a specific entity, evaluating the likelihood of exploitation, or differentiating between malicious and benign behaviors in the subgraphs. They may also identify additional IOCs for further investigation. Overall, attack investigator supports analysts with a user-friendly and effective interface for in-depth incident analysis.

7 Evaluation

This section presents a comprehensive evaluation of OCR-APT. We compare its detection accuracy with state-of-the-art (SOTA) anomaly detection systems, excluding rule-based systems since they target known attacks, while anomaly-based methods detect novel threats. We also study the impact of core components on accuracy via ablation and assess their computational cost. Finally, we evaluate the quality of our LLM-based investigation by comparing generated reports to ground truth reports from simulated attacks.

7.1 Datasets

We evaluated our system on three datasets: DARPA Transparent Computing Engagement 3 (TC3) [18], DARPA Operationally Transparent Cyber (OpTC) [19], and NODLINK simulated dataset [46]. These datasets consist of audit logs collected from diverse operating systems, with a total exceeding 80 million system events. Detailed statistics for each dataset are summarized in Table 1. On average, malicious nodes represent less than 0.01% of the total nodes, which aligns with typical APT behaviors. Therefore, we used the F1-score as the primary evaluation metric, as it effectively measures performance on highly imbalanced datasets [61].

7.1.1 DARPA TC3. The DARPA TC3 dataset, widely used as a benchmark for provenance graph intrusion detection [90], was developed to support research on APT-focused cybersecurity solutions [18]. Over two weeks, adversarial teams executed APT-based attacks and documented their activities in ground truth reports [17].

These reports provide summaries of attack stages, key attack indicators, and detailed event logs with timestamps; however, they do not specify the exact malicious system entities involved as ground-truth labels. Our evaluation covered two Linux-based hosts (THEIA and TRACE) and one FreeBSD-based host (CADETS).

7.1.2 DARPA OpTC. The DARPA OpTC dataset includes data from 1,000 Windows OS hosts simulating a large enterprise environment [6, 19]. It spans seven days, with only benign activity during the first four days. The final three days contain both benign and malicious activity, where a red team conducts APT-style attacks. These attacks cover the full APT lifecycle [20], including initial compromise, internal reconnaissance, command & control, persistence, and trace-covering actions. Following prior studies [5, 70], we focus our evaluation on the three hosts with the highest volume of attack traces, based on the ground truth provided by FLASH. This selection enables fair comparison with existing methods while still presenting a challenging detection setting due to the low proportion of malicious nodes (0.024%).

7.1.3 Simulated NODLINK. This dataset, released by NODLINK [46], simulates the internal environment of a security company, Sangfor. Data were collected from three hosts: an Ubuntu 20.04 server, a Windows Server 2012 (WS 12), and a Windows 10 desktop host (W10). The dataset includes attack descriptions and ground-truth labels, which we used in our evaluation. This dataset enabled us to benchmark our system's performance against NODLINK.

7.2 Evaluation Setup

We train OCR-APT on benign traces (D_b) collected from the provenance graph of a specific host. Then, we test OCR-APT using graphs containing both malicious and benign traces, excluding D_b. We ensure fair and consistent evaluation across the baselines (THREATRACE [77], FLASH [70], MAGIC [40], and KAIROS [40]) using the same datasets, labels, and metrics. For NODLINK [46], reproduction on DARPA TC3 was not possible due to the lack of access to the specific data subset and ground truth labels used in their evaluation. Instead, we relied on the reported TC3 results and reproduced experiments on their simulated dataset using the official system. We also reproduced some baselines: FLASH results closely matched published ones, while THREATRACE showed high variance (e.g., F1-score 0.595 ± 0.434 on TC3). We contacted the authors, who confirmed this instability. Therefore, we rely on original papers to compare OCR-APT with each method's best-performing version. We follow the same evaluation setup as prior work [70, 77], where true positives are anomalous nodes correctly identified as abnormal or those with 2-hop neighbors flagged as abnormal. False positives are benign nodes mistakenly flagged despite having no anomalous nodes within two hops.

7.2.1 Parameter Setting. To optimize detection accuracy, we conducted a hyperparameter tuning experiment to select the default parameters, which were subsequently used in all our evaluations. For our OCRGCN models, we implemented three layers of RGCN, utilizing a 32-dimensional embedding vector and a learning rate[4] of 0.005.

The contamination factor was set to range between $Min_{con} = 0.001$ and $Max_{con} = 0.05$. Following prior work [51], we set $\beta = 0.5$. The number of seed nodes n_{seed} for subgraph construction was set to 15, with a maximum of 5000 edges per subgraph (max_e). We assess the abnormality levels of the constructed subgraphs as follows. Subgraphs with an anomaly score below 10 are classified as having minor abnormalities. Those with scores between 10 and 100 exhibit moderate abnormalities. Scores between 100 and 1000 indicate significant abnormalities, and scores exceeding 1000 are categorized as critical. In our evaluation, subgraphs with moderate abnormalities or higher are labeled as anomalous.

7.2.2 Infrastructure. Our experiments were conducted on a Linux system equipped with 64 cores and 256 GB of RAM. We developed OCR-APT using Python and Bash scripts, leveraging PyTorch Geometric [66] for training GNN models and NetworkX [7] for subgraph construction. Our OCRGCN is built on top of the PyGOD [51] library. Provenance graphs are stored in the GraphDB [63] RDF graph database, which supports the RDF-star format used in our system [5]. We developed our RAG-based pipeline using LlamaIndex [50], which offers a vector store and API calls for various LLMs. For our main LLM, we used GPT-4o-mini [64] with temperature set to 0 to ensure accurate and deterministic results [73]. As part of our ablation study, we tested OpenAI's embedding models and selected 'text-embedding-3-large' [65] for indexing due to its strong performance. The entire system was implemented in approximately 5,000 lines of code.

7.3 Evaluation of Detection Accuracy

We compared OCR-APT with state-of-the-art (SOTA) anomaly detection systems across various granularities: nodes (THREATRACE and MAGIC), time windows (KAIROS), and subgraphs (NODLINK and FLASH). To enable unified evaluation, Table 2 reports the detection accuracy of all systems at the node level, where any node within an anomalous time window or subgraph is labeled anomalous. Results for the SOTA systems are drawn from their original papers[5]. Overall, OCR-APT achieved comparable or superior accuracy to existing node-level detectors. However, these systems do not support subgraph-level anomaly detection, which is essential for our LLM-based investigator to reconstruct attack reports.

On the DARPA TC3 dataset, OCR-APT achieved higher recall than KAIROS[6]. KAIROS detects anomalies over 15-minute time windows, each manually labeled based on ground truth. While it attains 100% recall at the window level, its node-level recall caps at 95%, likely due to malicious nodes falling outside the labeled windows, which may contain both benign and malicious entities. In contrast, OCR-APT provides comparable accuracy while operating at the subgraph level rather than fixed time windows. This ensures that anomalous subgraphs consist only of causally connected anomalous nodes, incorporating benign nodes only when they serve as bridges between anomalous events.

Detecting anomalies at the subgraph level improves alert validation and interpretability but is more challenging than node-level

---

[4]The learning rate determines the step size for parameter updates during training [84].

[5]As THREATRACE does not report evaluation results on the OpTC dataset, we present the results produced and reported by FLASH.
[6]KAIROS was also evaluated on the DARPA OpTC dataset, but its results were not reported at the node level, preventing direct comparison.

Table 2: Detection accuracy of OCR-APT in comparison with SOTA anomaly detection systems on DARPA TC3, DARPA OpTC, and Simulated NODLINK datasets.

Dataset	System	Precision	Recall	F1-Score
TC3 (CADETS)	THREATRACE	0.90	0.99	0.95
	MAGIC	0.94	0.99	0.97
	KAIROS	1.00	0.95	0.97
	NODLINK	0.14	1.00	0.25
	FLASH	0.95	0.99	0.97
	OCR-APT	**1.00**	**1.00**	**1.00**
TC3 (TRACE)	THREATRACE	0.72	0.99	0.83
	MAGIC	0.99	0.99	0.99
	NODLINK	0.25	0.98	0.40
	FLASH	0.95	0.99	0.97
	OCR-APT	**1.00**	**1.00**	**1.00**
TC3 (THEIA)	THREATRACE	0.87	0.99	0.93
	MAGIC	0.98	0.99	0.99
	KAIROS	1.00	0.95	0.97
	NODLINK	0.23	1.00	0.37
	FLASH	0.93	0.99	0.96
	OCR-APT	**1.00**	**1.00**	**1.00**
OpTC (201)	THREATRACE	0.84	0.85	0.84
	FLASH	0.90	0.92	0.91
	OCR-APT	**1.00**	**0.88**	**0.94**
OpTC (501)	THREATRACE	0.85	0.87	0.86
	FLASH	0.94	0.92	0.93
	OCR-APT	**1.00**	**1.00**	**1.00**
OpTC (51)	THREATRACE	0.86	0.87	0.86
	FLASH	**0.94**	**0.92**	**0.93**
	OCR-APT	0.89	0.77	0.82
NODLINK (Ubuntu)	NODLINK	0.04	0.38	0.07
	OCR-APT	**0.95**	**1.00**	**0.97**
NODLINK (WS 12)	NODLINK	0.10	0.84	0.17
	OCR-APT	**0.74**	**0.93**	**0.82**
NODLINK (W10)	NODLINK	0.14	0.68	0.23
	OCR-APT	**0.95**	**0.99**	**0.97**

detection. OCR-APT achieved perfect accuracy across all DARPA TC3 hosts, whereas NODLINK struggled with a high false-positive rate, reporting a maximum precision of just 0.25 on the TRACE host. NODLINK uses sentence embeddings, which fail to capture the graph structure. This limitation negatively impacts its accuracy, as observed in the NODLINK dataset[7]. In contrast, OCRGCN leverages both graph structure and node behavior, leading to superior performance. OCR-APT consistently outperformed NODLINK, with its lowest F1-score being 0.82 on the simulated WS 12 host. In this case, OCR-APT missed 3 out of 47 malicious entities and produced 13 false positives among 10,860 benign nodes. Due to the small size of the simulated dataset, minor errors had a considerable impact on evaluation metrics. OCR-APT enhances interpretability through subgraph-level detection without compromising accuracy.

---

[7]As NODLINK's authors did not provide per-host results on their simulated dataset, we executed it using their public scripts and metrics without modification.

Overall, OCR-APT outperformed all detectors across all hosts, except for OpTC 51, where FLASH achieved higher detection accuracy. In that case, the adversary launched a malicious upgrade attack by installing a backdoored version of Notepad-Plus. During the update process, the backdoor connected to the attacker's server to download both legitimate updates and a malicious binary. This behavior confused the anomaly detection model, which failed to flag the malicious binary. However, our LLM-based attack investigator successfully identified both the malicious binary and the command-and-control server in the generated attack report. Furthermore, OCR-APT demonstrates greater robustness than SOTA anomaly detection systems by avoiding reliance on node features that are susceptible to adversarial manipulation.

Robustness to evasion remains a critical factor for anomaly detection systems. A growing concern in this area is mimicry attacks, where adversaries inject benign activities into attack graphs to evade detection while preserving the core malicious behavior. Provenance-based intrusion detection systems that operate at the graph or path level have proven vulnerable to such tactics [25]. However, the same study suggests that focusing on finer-grained —such as nodes, edges, or subgraphs—can mitigate this risk [25], a strategy that has shown promise in several recent systems [16, 70, 75]. OCR-APT's subgraph-level detection naturally aligns with these insights and offers a promising defense against such evasion. As part of future work, we aim to assess its robustness against a broad range of mimicry and evasion techniques [25, 60].

7.4 Ablation Study

This section evaluates OCRGCN against existing GNN-based anomaly detectors using the simulated NODLINK dataset, chosen for its manageable size. Some baselines failed to run on larger datasets (e.g., CADETS from DARPA TC3) due to memory constraints. We also conduct ablation studies to assess the impact of key components and tune hyperparameters for optimal performance. Each experiment is repeated 10 times, and average results are reported.

7.4.1 The OCRGCN Models. We developed six variations of OCR-APT: one with our GNN model, and the rest with existing GNN-based anomaly detection models implemented using the PyGOD [51] library. These models include AnomalyDAE [22], CONAD [81], CoLA [53], GAE [43], and OCGNN [78]. Table 3 presents a comparison of the detection accuracy and efficiency of OCR-APT using OCRGCN model versus general detectors.

Overall, OCRGCN consistently outperforms these detectors in accuracy, as it captures edge types when aggregating node embeddings. In contrast, these detectors are designed for homogeneous graphs and do not incorporate edge types. For example, OCGNN is a one-class classification method similar to OCRGCN, but it does not capture edge types. As a result, OCGNN suffers from low precision and struggles to differentiate between normal and anomalous nodes. CoLA, a self-supervised learning method for graph anomaly detection, achieves slightly higher precision than OCRGCN in the W10 host. However, its performance is inconsistent, with an F1-score of approximately 0.4 in the other two hosts. Autoencoding-based methods, such as GAE, CONAD, and AnomalyDAE, exhibit inconsistent detection accuracy. Notably, both CONAD and AnomalyDAE failed to detect any anomalous nodes in the WS12 host. Besides, these

Table 3: Evaluating APT detection accuracy and efficiency of OCR-APT on the Simulated NODLINK dataset using various GNN-based anomaly detection models. OCRGCN is our novel GNN-based model.

Host	Model	Precision	Recall	F1-Score	Detection Time (s)	Occupied Memory (GB)
Ubuntu	AnomalyDAE	0.24	0.61	0.34	1,411.67	53.88
	CONAD	0.34	1.00	0.51	1,431.23	53.93
	GAE	**1.00**	**0.94**	**0.97**	928.60	46.27
	CoLA	0.32	0.59	0.40	182.33	13.58
	OCGNN	0.04	1.00	0.07	941.84	31.43
	OCRGCN	**0.95**	**1.00**	**0.97**	828.01	41.78
WS12	AnomalyDAE	0.00	0.00	0.00	111.16	8.26
	CONAD	0.00	0.00	0.00	124.34	9.03
	GAE	0.26	0.93	0.40	115.13	7.47
	CoLA	0.30	0.47	0.37	31.80	7.80
	OCGNN	0.32	0.98	0.48	108.39	7.80
	OCRGCN	**0.74**	**0.93**	**0.82**	30.87	10.82
W10	AnomalyDAE	0.82	0.99	0.90	266.69	63.69
	CONAD	0.82	0.99	0.90	264.55	59.45
	GAE	0.70	1.00	0.83	159.69	7.26
	CoLA	**0.98**	**0.99**	**0.99**	41.43	7.59
	OCGNN	0.75	1.00	0.86	173.54	7.69
	OCRGCN	**0.95**	**0.99**	**0.97**	59.21	10.48

Table 4: Comparison of APT Detection accuracy and efficiency of OCR-APT on Simulated NODLINK dataset with all system components, and without the behavior-based features (B-Feat), the type-specific models (TS-Mod), and the subgraph anomaly detection (SG-Det).

Host	Version	Precision	Recall	F1-Score	Detection Time (s)	Occupied Memory (GB)
Ubuntu	Without B-Feat	0.41	1.00	0.58	1,380.69	52.83
	Without TS-Mod	1.00	0.44	0.62	24.01	18.97
	Without SG-Det	0.58	1.00	0.73	34.89	19.61
	OCR-APT	**0.95**	**1.00**	**0.97**	828.01	41.78
WS12	Without B-Feat	0.00	0.00	0.00	57.02	10.65
	Without TS-Mod	0.22	0.93	0.36	12.37	10.47
	Without SG-Det	0.32	1.00	0.48	22.58	10.82
	OCR-APT	**0.74**	**0.93**	**0.82**	30.87	10.82
W10	Without B-Feat	0.89	0.99	0.94	134.26	10.30
	Without TS-Mod	0.80	0.99	0.89	78.74	10.10
	Without SG-Det	0.80	1.00	0.89	20.28	10.48
	OCR-APT	**0.95**	**0.99**	**0.97**	59.21	10.48

methods are typically memory-intensive, as they scale quadratically with the number of nodes due to the reconstruction of the complete graph adjacency matrix [52].

7.4.2 OCR-APT System Components. We conducted ablation experiments to assess the impact of OCR-APT's components. Four variations of the system were created: the full system, one without behavior-based features (*Without B-Feat*), one without type-specific models (*Without TS-Mod*), and one without subgraph anomaly detection (*Without SG-Det*). Table 4 shows the detection accuracy and efficiency of each variation.

The variation *Without B-Feat* relies on features from prior work, THREATRACE [77], excluding statistics of the node idle phase and normalization techniques. This variant failed to detect any anomalous nodes in the WS12 host and reduced precision in the other two hosts. These results align with our hypothesis that the statistics of node idle periods assist in distinguishing benign nodes from those associated with APT activity. Additionally, our behavior-based features enhance time efficiency due to feature normalization.

The *Without TS-Mod* variant employs a single OCRGCN model for all node types. While this approach improves time and memory efficiency, it significantly compromises detection accuracy. In the Ubuntu host, recall dropped from 100% to 44%, while precision in the WS12 host declined sharply from 74% to 22%. These findings indicate that the model struggles to learn the normal behavior of different node types when relying on a single model. Training multiple OCRGCN models—one per node type—assists in capturing variations in benign behavior, as normal behavior patterns differ across node types. We acknowledge that benign traffic may exhibit multiple behavior patterns; therefore, exploring clustering-based methods may offer a promising direction for future work.

In Table 4, the *Without SG-Det* variant performs only node-level detection, avoiding the time overhead of subgraph construction and detection. This variant reduces precision across all hosts, underscoring the value of our subgraph anomaly detection in filtering

false positives. The results show that subgraph detection improves precision while maintaining high recall. OCR-APT's subgraph construction reduces false positives by filtering out subgraphs with low abnormality scores. Even if a node's anomaly score is inaccurate, it will not trigger an alert unless it belongs to a highly abnormal subgraph. These findings highlight the importance of each core component in enhancing OCR-APT's effectiveness.

7.4.3 Hyperparameter Tuning. To optimize the F1-score, hyperparameter tuning was performed using Bash scripts to systematically evaluate a range of parameter configurations. For GNN model training, this included variations in the number of RGCN layers {2, 3, 4}, graph embedding vector sizes {32, 64, 92}, and learning rates {0.005, 0.001, 0.0005}. We also varied the β parameter of the one-class SVM in {0.3, 0.4, 0.5, 0.6, 0.7} and observed minimal impact on performance; thus, we used PyGOD's [51] default value ($\beta = 0.5$). For subgraph construction, we considered parameters such as the number of seed nodes {10, 15, 20} and the maximum edges per subgraph {5000, 10000}. We also evaluated the impact of using two-hop expansion during subgraph construction. While two-hop expansion slightly reduces false negatives, it significantly increases false positives. For example, in THEIA, false negatives reduced from 5 to 2, but false positives rose sharply from 0 to 21.5 K, reducing precision from 1.0 to 0.5. Other datasets showed minimal impact (see Appendix D for more results). Based on these experiments, we adopt one-hop expansion as the default configuration for subgraph construction. While the selected default parameters were applied consistently across all hosts, future datasets with varying levels of complexity may benefit from dataset-specific hyperparameter tuning to ensure optimal performance.

7.5 Evaluation of Recovered Attack Reports

We evaluated the quality of our LLM-based attack investigator by comparing our recovered attack reports to the ground truth reports provided by DARPA [17, 20]. Table 5 provides a summary of the detected IOCs and APT stages across all reports. The APT stages include Initial Compromise (IC), Internal Reconnaissance (IR), Command and Control (C&C), Privilege Escalation (PE), Lateral Movement (LM), Maintain Persistence (MP), Data Exfiltration (DE),

Table 5: Evaluation of recovered attack reports using both commercial (GPT-4o-mini) and local (LLAMA3-8B) LLMs on DARPA TC3 and OpTC datasets. The table shows the number of detected IOCs and APT attack stages, with total counts in parentheses. Detected stages are highlighted in green, while missed stages are shown in red.

LLM	Dataset	Host	# Detected IOCs	# Detected APT Stages	Detected APT Stages
GPT-4o-mini	DARPA TC3	CADETS	11 (16)	5 (6)	IC, MP, PE, C&C, IR, CT
		TRACE	6 (7)	4 (4)	IC, MP, C&C, IR
		THEIA	5 (7)	5 (6)	IC, MP, PE, C&C, IR, CT
	DARPA OpTC	201	5 (6)	5 (7)	IC, MP, PE, C&C, IR, LM , CT
		501	7 (11)	5 (8)	IC, MP, PE, C&C, IR, LM, DE, CT
		51	8 (10)	4 (6)	IC , MP, PE, C&C, IR, LM
LLAMA3-8B	DARPA TC3	CADETS	10 (16)	5 (6)	IC, MP, PE, C&C, IR, CT
		TRACE	5 (7)	4 (4)	IC, MP, C&C, IR
		THEIA	5 (7)	5 (6)	IC, MP, PE, C&C, IR, CT
	DARPA OpTC	201	2 (6)	3 (7)	IC, MP, PE, C&C, IR, LM , CT
		501	7 (11)	5 (8)	IC, MP, PE, C&C, IR, LM, DE, CT
		51	7 (10)	4 (6)	IC , MP, PE, C&C, IR, LM

and Covering Tracks (CT). In most cases, our recovered reports covered the majority of APT stages (highlighted in green). They clearly specify artifacts, such as command-and-control servers, malicious executable files, and exploited processes involved in the attacks. For example, reports recovered from the TRACE host captures all performed attack stages, including initial compromise, persistence, command and control, and internal reconnaissance. Simplified versions of the recovered reports[8] are provided in Appendix C. Missed stages (highlighted in red) in DARPA TC3 dataset primarily resulted from OS log parsing issues. This led to the missing of process attributes on the CADETS host and file attributes on the THEIA host.

For the DARPA OpTC dataset, the recovered reports captured most APT stages but struggled to identify the lateral movement and initial compromise phases. Detecting lateral movement was beyond the scope of this work, as OCR-APT does not process network traffic logs. In future work, we plan to address this limitation by integrating network traffic analysis with specialized detectors. The initial compromise stage was challenging to detect because the initial payload files remained inactive during the attack. As a result, the anomaly detection models did not flag them as suspicious. However, further analysis revealed that these overlooked artifacts were directly connected to detected IOCs. To mitigate missing IOCs, our approach enriches reports with subgraphs surrounding key IOCs. The LLM-based investigator explores these anomalous subgraphs to uncover related artifacts missed by the detection model. For example, on the OpTC 51 host, it identified a malicious binary and a C&C IP that had been initially overlooked. This enrichment helps the LLM infer additional threats and improves overall detection. Despite these limitations, the recovered reports provide clear and detailed accounts of the attack scenarios. They align well with the attack timestamps from the ground truth and reference most key artifacts. Moreover, the reports are written in a human-like narrative style, similar to CTI reports.

To evaluate our system in a practical setting, we conducted experiments using locally deployed open-source LLMs. These models preserve data privacy by eliminating the need to transmit sensitive

---

[8]Full versions of the reports are available at https://github.com/CoDS-GCS/OCR-APT/tree/main/recovered_reports

Table 6: Detection accuracy of OCR-APT and FLASH on DARPA TC3 without neighbor-based assumptions (original metric results in brackets).

Dataset	System	Precision	Recall	F1-Score
CADETS	FLASH	0.65 (0.93)	1.00 (1.00)	0.79 (0.96)
	OCR-APT	**1.00 (1.00)**	**1.00 (1.00)**	**1.00 (1.00)**
TRACE	FLASH	0.66 (0.95)	0.99 (0.99)	0.79 (0.97)
	OCR-APT	**0.87 (1.00)**	**1.00 (1.00)**	**0.93 (1.00)**
THEIA	FLASH	0.72 (0.92)	0.99 (1.00)	0.84 (0.96)
	OCR-APT	**0.98 (1.00)**	**1.00 (1.00)**	**0.99 (1.00)**

system logs to commercial providers, offering a cost-effective solution for long-term use. We ran the pipeline on a local machine with 4 CPU cores, an 8GB GPU, and 22GB of RAM. An ablation study guided the selection of the most effective local LLM and embedding model. We evaluated six local LLMs and seven embedding models. The best setup—LLaMA3 (8B)[1] paired with IBM's open-source 'granite-embedding-125m-english'[26]—achieved comparable performance to ChatGPT. As shown in Table 5, this configuration detected the same APT stages as ChatGPT on all hosts except DARPA OpTC 501, where it missed two stages. While the quality of the comprehensive reports declined, the overall investigation results remained reliable and informative. This strong performance demonstrates the effectiveness of our pipeline, even when using lightweight, locally deployed models. LLAMA3's relatively small size further suggests that OCR-APT's performance is not driven by memorization of benchmark datasets.

Furthermore, OCR-APT systematically validates generated reports against detected anomalies, ensuring that outputs are grounded in actual data rather than relying on prior model knowledge or LLM hallucinations. This is achieved by modularizing the investigation into subtasks, each guided by specialized Chain-of-Thought (CoT) prompts. To assess the impact of this design, we compared it against a baseline that uses a single CoT-based prompt to generate the entire attack report, skipping intermediate steps like IOC extraction and validation. The baseline produced fewer detected APT stages and, more critically, frequent hallucinations—including fabricated entities like `malicious.exe`, `suspicious_process.exe`, and `vulnerable_service.exe`, which were not present in the source provenance graph. These results highlight the advantages of our modular pipeline and its integrated validation mechanism.

Our OCR-APT automatically analyzes audit logs and generates valuable insights in the form of human-like security reports. These reports cover most APT stages and include key IOCs. Hence, they provide a clear overview of the attack progression and highlight critical indicators. These reports save security analysts significant time and enable them to quickly identify key patterns. This leads to more focused and efficient investigations.

7.6 Discussion and Limitations

7.6.1 Evaluation Metric. We follow the evaluation setup used in prior work [16, 40, 70, 77], which treats neighboring nodes of compromised ones as part of the attack. Although this assumption may not always hold [16], it ensures consistency with existing system

evaluations. To further assess OCR-APT, we conducted additional experiments using a stricter metric that considers only directly identified malicious nodes as true positives, without relying on neighbor-based assumptions. Table 6 presents the detection results of OCR-APT and FLASH (using FLASH's official implementation) under this setting. The results show that OCR-APT maintains high precision and recall, with only minor precision drops in some cases. In contrast, FLASH exhibits a significant decline in precision. For instance, on the CADETS host, FLASH's precision dropped from 0.93 to 0.65, while OCR-APT remained stable. On TRACE, FLASH fell from 0.95 to 0.66, whereas OCR-APT declined slightly to 0.87. On THEIA, FLASH dropped from 0.92 to 0.72, while OCR-APT maintained a high precision of 0.98. These results highlight OCR-APT's reliability under stricter evaluation and its advantage over existing baselines. A broader assessment of evaluation metrics is a promising direction for future work toward establishing best practices in anomaly detection benchmarking. Though narrative clarity remains challenging to quantify, OCR-APT's structured reports offer more interpretable outputs than prior methods, encouraging future efforts to formalize this aspect.

7.6.2 Multiple Attack Handling. One limitation of our approach lies in the subgraph construction process, which may inadvertently merge multiple attacks into a single subgraph when they share system entities (e.g., processes). While this can be useful for capturing shared infrastructure or correlated activity, it may also obscure the boundaries between causally unrelated attacks, potentially confusing the investigation reports. Although our system can manage causally disconnected attacks to some extent, accurately distinguishing them within a shared subgraph remains challenging. Future work could address this by segmenting subgraphs based on behavioral signatures or by enhancing the LLM investigation module to better identify boundaries between separate attacks.

7.6.3 Model Generalization. The model's ability to generalize is influenced by the extent to which benign behavior is represented in the training data—a known limitation of anomaly detection. Our approach mitigates this by incorporating structural and behavioral features that support generalization, as reflected in the consistently high precision observed across different hosts. However, unseen benign patterns can still lead to false positives. To address this, future work could investigate model adaptation strategies that incorporate analyst feedback through semi-supervised learning.

8 Related Work

Recent research on provenance-based APT detection [90] can be categorized into two main approaches: heuristic-based and anomaly-based methods [39]. In Section 2, we discussed the limitations of anomaly detection systems. This section complements the discussion on related work by focusing on heuristic techniques and the emerging role of LLMs in cybersecurity, highlighting how OCR-APT differs from existing work.

LLMs in Cybersecurity. LLMs have been applied across diverse cybersecurity tasks, including software vulnerability detection [48, 55, 73], fuzzing [62], automated patching [44, 67], threat detection (e.g., DDoS and phishing) [27, 45, 47], penetration testing [21], and malware reverse engineering [35]. In threat intelligence, LLMs help

extract knowledge graphs from CTI reports [15, 24, 88], with benchmarks like AttackSeqBench [85] assessing LLM effectiveness. The potential of LLMs for anomaly detection has been explored in a recent survey [14]. In contrast, OCR-APT uniquely applies LLMs to reconstruct APT stories from anomalous subgraph alerts. By combining subgraph-level anomaly detection with LLM-driven tasks, such as IOC extraction, stage identification, and report generation, OCR-APT produces interpretable and context-rich reports.

Heuristic-based Detection. These systems identify malicious behavior through rules, graph matching, or supervised learning on known attacks. Rule-based approaches [30, 33, 34, 59] rely on expert-defined specifications derived from TTPs, but they are prone to high false positives or miss zero-day threats [77]. CAPTAIN [75] improves this by tuning rules with benign data. Graph matching systems [4, 5, 58] compare suspicious subgraphs to predefined query graphs derived from CTI reports. While automated graph construction is possible [58], these systems struggle with novel behaviors not covered in the queries. Similarly, supervised models [13, 82] trained on labeled datasets are limited by the scarcity and cost of real APT data. APT-KGL [13] augments training data by mining TTPs and CTI reports but still lacks generalization to unseen threats. Though some systems incorporate Relational GCNs (RGCNs) [5, 13], they typically focus on rule-based or supervised learning paradigms. In contrast, OCR-APT adopts a fully anomaly-based approach, detecting deviations from normal behavior at a fine-grained subgraph level—enabling identification of both known and unknown APTs.

In summary, OCR-APT's core innovation lies in integrating subgraph-level anomaly detection with LLM-based attack reconstruction. This approach avoids reliance on static rules or labeled attack data, offering greater adaptability to emerging threats. By converting anomaly alerts into detailed, human-readable reports, OCR-APT enhances both detection and interpretability. This makes OCR-APT a robust and versatile solution for APT defense.

9 Conclusion

We proposed OCR-APT, a system that automatically detects APTs and recovers attack reports from provenance graphs. We developed OCR-APT based on our novel GNN-based subgraph anomaly detection and LLM-based investigation. Hence, OCR-APT overcomes the limitations of existing systems. The LLM-based attack investigator generates concise and human-like reports that help analysts efficiently assess and prioritize threats. Comprehensive evaluations on the DARPA TC3, OpTC, and NODLINK datasets show that OCR-APT consistently outperforms state-of-the-art subgraph anomaly detection systems in detection accuracy. It also enhances the interpretability of results. Additionally, the ablation study demonstrates that OCR-APT effectively balances detection accuracy with memory and time efficiency. By integrating GNN-based detection with LLM-guided interpretation, OCR-APT significantly advances APT detection and streamlines alert verification, bridging the gap between low-level telemetry and high-level analyst insight.

References

[1] AI@Meta. 2024. Llama 3 Model Card. (2024). https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[2] Abdulellah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z Berkay Celik, Xiangyu Zhang, and Dongyan Xu. 2021. ATLAS: A Sequence-based Learning Approach for Attack Investigation. In *USENIX Security Symposium*.

[3] Adel Alshamrani, Sowmya Myneni, Ankur Chowdhary, and Dijiang Huang. 2019. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys & Tutorials* 21, 2 (2019), 1851–1877.

[4] Enes Altinisik, Fatih Deniz, and Hüsrev Taha Sencar. 2023. Provg-searcher: a graph representation learning approach for efficient provenance graph search. In *Proceedings of the 2023 ACM SIGSAC conference on computer and communications security*. 2247–2261.

[5] Ahmed Aly, Shahrear Iqbal, Amr Youssef, and Essam Mansour. 2024. MEGR-APT: A Memory-Efficient APT Hunting System Based on Attack Representation Learning. *IEEE Transactions on Information Forensics and Security* 19 (2024), 5257–5271.

[6] Md Monowar Anjum, Shahrear Iqbal, and Benoit Hamelin. 2021. Analyzing the usefulness of the DARPA OpTC dataset in cyber threat detection research. In *Proceedings of the ACM Symposium on Access Control Models and Technologies*. 27–32.

[7] Pieter Swart Aric Hagberg, Dan Schult. 2024. NetworkX: Network Analysis in Python. https://github.com/networkx/networkx Accessed: 2025-03-06.

[8] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2022. Dos and don'ts of machine learning in computer security. In *USENIX Security Symposium*.

[9] Bibek Bhattarai and Howie Huang. 2022. SteinerLog: Prize Collecting the Audit Logs for Threat Hunting on Enterprise Network. In *Proceedings of the ACM on Asia Conference on Computer and Communications Security*. 97–108.

[10] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[11] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the association for computational linguistics* 5 (2017), 135–146.

[12] Abdenour Bounsiar and Michael G Madden. 2014. One-class support vector machines revisited. In *International Conference on Information Science & Applications (ICISA)*. IEEE, 1–4.

[13] Tieming Chen, Chengyu Dong, Mingqi Lv, Qijie Song, Haiwen Liu, Tiantian Zhu, Kang Xu, Ling Chen, Shouling Ji, and Yuan Fan. 2022. APT-KGL: An Intelligent apt Detection System Based on Threat Knowledge and Heterogeneous Provenance Graph Learning. *IEEE Transactions on Dependable and Secure Computing* (2022).

[14] Wenrui Cheng, Tiantian Zhu, Chunlin Xiong, Haofei Sun, Zijun Wang, Shunan Jing, Mingqi Lv, and Yan Chen. 2025. SoK: Knowledge is All You Need: Last Mile Delivery for Automated Provenance-based Intrusion Detection with LLMs. *arXiv preprint arXiv:2503.03108* (2025).

[15] Yutong Cheng, Osama Bajaber, Saimon Amanuel Tsegai, Dawn Song, and Peng Gao. 2024. CTINEXUS: Leveraging Optimized LLM In-Context Learning for Constructing Cybersecurity Knowledge Graphs Under Data Scarcity. *arXiv preprint arXiv:2410.21060* (2024).

[16] Zijun Cheng, Qiujian Lv, Jinyuan Liang, Yan Wang, Degang Sun, Thomas Pasquier, and Xueyuan Han. 2024. Kairos: Practical intrusion detection and investigation using whole-system provenance. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3533–3551.

[17] DARPA. 2018. TC3 Ground Truth Report. https://drive.google.com/file/d/1mrs4LWkGk-3zA7t7v8zrhm0yEDHe57QU/view Accessed: 2025-03-06.

[18] DARPA. 2018. Transparent Computing Engagement 3 (TC3) Data Release. https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md Accessed: 2025-03-06.

[19] DARPA. 2020. Operationally Transparent Cyber (OpTC) Data Release. https://github.com/FiveDirections/OpTC-data Accessed: 2025-03-06.

[20] DARPA. 2020. OpTC Ground Truth Report. https://drive.google.com/file/d/1lX8kfrdZGJwaqSdwTlEBGwmz069lZWh-/view Accessed: 2025-03-06.

[21] Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. 2024. PentestGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing. In *USENIX Security Symposium*. 847–864.

[22] Haoyi Fan, Fengbin Zhang, and Zuoyong Li. 2020. Anomalydae: Dual autoencoder for anomaly detection on attributed networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5685–5689.

[23] Pengcheng Fang, Peng Gao, Changlin Liu, Erman Ayday, Kangkook Jee, Ting Wang, Yanfang Fanny Ye, Zhuotao Liu, and Xusheng Xiao. 2022. Back-Propagating System Dependency Impact for Attack Investigation. In *USENIX Security Symposium*. 2461–2478.

[24] Romy Fieblinger, Md Tanvirul Alam, and Nidhi Rastogi. 2024. Actionable cyber threat intelligence using knowledge graphs and large language models. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 100–111.

[25] Akul Goyal, Xueyuan Han, Gang Wang, and Adam Bates. 2023. Sometimes, you aren't what you do: Mimicry attacks against provenance graph host intrusion detection systems. In *Network and Distributed System Security (NDSS) Symposium*.

[26] IBM Granite Embedding Team. 2024. Granite Embedding Models. https://github.com/ibm-granite/granite-embedding-models/

[27] Michael Guastalla, Yiyi Li, Arvin Hekmati, and Bhaskar Krishnamachari. 2023. Application of large language models to ddos attack detection. In *International Conference on Security and Privacy in Cyber-Physical Systems and Smart Vehicles*. Springer, 83–99.

[28] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[29] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. 2020. UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats. In *Network and Distributed Systems Security (NDSS) Symposium*.

[30] Wajih Ul Hassan, Adam Bates, and Daniel Marino. 2020. Tactical provenance analysis for endpoint detection and response systems. In *IEEE Symposium on Security and Privacy (SP)*. IEEE, 1172–1189.

[31] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. 2019. Nodoze: Combatting threat alert fatigue with automated provenance triage. In *Network and Distributed System Security (NDSS) Symposium*.

[32] Wajih Ul Hassan, Ding Li, Kangkook Jee, Xiao Yu, Kexuan Zou, Dawei Wang, Zhengzhang Chen, Zhichun Li, Junghwan Rhee, Jiaping Gui, et al. 2020. This is why we can't cache nice things: Lightning-fast threat hunting using suspicion-based hierarchical storage. In *Annual Computer Security Applications Conference*. 165–178.

[33] Md Nahid Hossain, Sadegh M Milajerdi, Junao Wang, Birhanu Eshete, Rigel Gjomemo, R Sekar, Scott Stoller, and VN Venkatakrishnan. 2017. SLEUTH: Real-time attack scenario reconstruction from COTS audit data. In *USENIX Security Symposium*. 487–504.

[34] Md Nahid Hossain, Sanaz Sheikhi, and R Sekar. 2020. Combating dependence explosion in forensic analysis using alternative tag propagation semantics. In *2020 IEEE symposium on security and privacy (SP)*. IEEE, 1139–1155.

[35] Peiwei Hu, Ruigang Liang, and Kai Chen. 2024. Degpt: Optimizing decompiler output with llm. In *Proceedings 2024 Network and Distributed System Security Symposium*, Vol. 267622140.

[36] Zeqi Huang, Yonghao Gu, and Qing Zhao. 2022. One-Class Directed Heterogeneous Graph Neural Network for Intrusion Detection. In *The International Conference on Innovation in Artificial Intelligence (ICIAI)*. 178–184.

[37] Frank K Hwang and Dana S Richards. 1992. Steiner tree problems. *Networks* 22, 1 (1992), 55–89.

[38] Makoto Imase and Bernard M Waxman. 1991. Dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics* 4, 3 (1991), 369–384.

[39] Muhammad Adil Inam, Yinfang Chen, Akul Goyal, Jason Liu, Jaron Mink, Noor Michael, Sneha Gaur, Adam Bates, and Wajih Ul Hassan. 2022. SoK: History is a Vast Early Warning System: Auditing the Provenance of System Intrusions. In *IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 307–325.

[40] Zian Jia, Yun Xiong, Yuhong Nan, Yao Zhang, Jinjing Zhao, and Mi Wen. 2024. MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning. In *USENIX Security Symposium*. 5197–5214.

[41] Maya Kapoor, Joshua Melton, Michael Ridenhour, Siddharth Krishnan, and Thomas Moyer. 2021. PROV-GEM: Automated Provenance Analysis Framework using Graph Embeddings. In *IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 1720–1727.

[42] Hwan Kim, Byung Suk Lee, Won-Yong Shin, and Sungsu Lim. 2022. Graph anomaly detection with graph neural networks: Current status and challenges. *IEEE Access* (2022).

[43] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[44] Ummay Kulsum, Haotian Zhu, Bowen Xu, and Marcelo d'Amorim. 2024. A case study of llm for automated vulnerability repair: Assessing impact of reasoning and patch validation feedback. In *Proceedings of the ACM International Conference on AI-Powered Software*. 103–111.

[45] Qingyang Li, Yihang Zhang, Zhidong Jia, Yannan Hu, Lei Zhang, Jianrong Zhang, Yongming Xu, Yong Cui, Zongming Guo, and Xinggong Zhang. 2024. DoLLM: How Large Language Models Understanding Network Flow Data to Detect Carpet Bombing DDoS. *arXiv preprint arXiv:2405.07638* (2024).

[46] Shaofei Li, Feng Dong, Xusheng Xiao, Haoyu Wang, Fei Shao, Jiedong Chen, Yao Guo, Xiangqun Chen, and Ding Li. 2024. NODLINK: An Online System for Fine-Grained apt Attack Detection and Investigation. In *Network and Distributed System Security (NDSS) Symposium*.

[47] Yuexin Li, Chengyu Huang, Shumin Deng, Mei Lin Lock, Tri Cao, Nay Oo, Hoon Wei Lim, and Bryan Hooi. 2024. KnowPhish: Large Language Models Meet Multimodal Knowledge Graphs for Enhancing Reference-Based Phishing Detection. In *33rd USENIX Security Symposium (USENIX Security 24)*. 793–810.

[48] Jie Lin and David Mohaisen. 2025. From Large to Mammoth: A Comparative Evaluation of Large Language Models in Vulnerability Detection. In *Network and Distributed System Security (NDSS) Symposium.*

[49] Fucheng Liu, Yu Wen, Dongxue Zhang, Xihe Jiang, Xinyu Xing, and Dan Meng. 2019. Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In *Proceedings of the ACM SIGSAC conference on computer and communications security.* 1777–1794.

[50] Jerry Liu. 2022. *LlamaIndex.* https://github.com/jerryjliu/llama_index

[51] Kay Liu, Yingtong Dou, Xueying Ding, Xiyang Hu, Ruitong Zhang, Hao Peng, Lichao Sun, and S Yu Philip. 2024. Pygod: A python library for graph outlier detection. *Journal of Machine Learning Research* 25, 141 (2024), 1–9.

[52] Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, et al. 2022. Bond: Benchmarking unsupervised outlier node detection on static attributed graphs. *Advances in Neural Information Processing Systems* 35 (2022), 27021–27035.

[53] Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. 2021. Anomaly detection on attributed networks via contrastive self-supervised learning. *IEEE transactions on neural networks and learning systems* 33, 6 (2021), 2378–2392.

[54] Yushan Liu, Xiaokui Shu, Yixin Sun, Jiyong Jang, and Prateek Mittal. 2022. RAPID: real-time alert investigation with context-aware prioritization for efficient threat discovery. In *Proceedings of the Annual Computer Security Applications Conference.* 827–840.

[55] Guilong Lu, Xiaolin Ju, Xiang Chen, Wenlong Pei, and Zhilong Cai. 2024. GRACE: Empowering LLM-based software vulnerability detection with graph structure and in-context learning. *Journal of Systems and Software* 212 (2024), 112031.

[56] Yang Lv, Shaona Qin, Zifeng Zhu, Zhuocheng Yu, Shudong Li, and Weihong Han. 2022. A Review of Provenance Graph based apt Attack Detection: Applications and Developments. In *IEEE International Conference on Data Science in Cyberspace (DSC).* IEEE, 498–505.

[57] Emaad Manzoor, Sadegh M Milajerdi, and Leman Akoglu. 2016. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 1035–1044.

[58] Sadegh M Milajerdi, Birhanu Eshete, Rigel Gjomemo, and VN Venkatakrishnan. 2019. Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.* 1795–1812.

[59] Sadegh M Milajerdi, Rigel Gjomemo, Birhanu Eshete, Ramachandran Sekar, and VN Venkatakrishnan. 2019. Holmes: real-time apt detection through correlation of suspicious information flows. In *IEEE Symposium on Security and Privacy (SP).* IEEE, 1137–1152.

[60] Kunal Mukherjee, Joshua Wiedemeier, Tianhao Wang, James Wei, Feng Chen, Muhyun Kim, Murat Kantarcioglu, and Kangkook Jee. 2023. Evading Provenance-BasedML detectors with adversarial system actions. In *32nd USENIX Security Symposium (USENIX Security 23).* 1199–1216.

[61] Gireen Naidu, Tranos Zuva, and Elias Mmbongeni Sibanda. 2023. A Review of Evaluation Metrics in Machine Learning Algorithms. In *Computer Science On-line Conference.* Springer, 15–25.

[62] Yaroslav Oliinyk, Michael Scott, Ryan Tsang, Chongzhou Fang, Houman Homayoun, et al. 2024. Fuzzing BusyBox: Leveraging LLM and Crash Reuse for Embedded Bug Unearthing. In *33rd USENIX Security Symposium (USENIX Security 24).* 883–900.

[63] Ontotext. 2025. GraphDB. https://www.ontotext.com/products/graphdb/ Accessed: 2025-03-06.

[64] OpenAI. 2024. GPT-4o-mini. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/. Accessed April 2025.

[65] OpenAI. 2024. text-embedding-3-large. https://platform.openai.com/docs/guides/embeddings/embedding-models. Accessed April 2025.

[66] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, and et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems.* Curran Associates, Inc.

[67] Hammond Pearce, Benjamin Tan, Baleegh Ahmad, Ramesh Karri, and Brendan Dolan-Gavitt. 2023. Examining zero-shot vulnerability repair with large language models. In *2023 IEEE Symposium on Security and Privacy (SP).* IEEE, 2339–2356.

[68] Kexin Pei, Zhongshu Gu, Brendan Saltaformaggio, Shiqing Ma, Fei Wang, Zhiwei Zhang, Luo Si, Xiangyu Zhang, and Dongyan Xu. 2016. Hercule: Attack story reconstruction via community discovery on correlated log graph. In *Proceedings of the Annual Conference on Computer Security Applications.* 583–595.

[69] Gabrijela Perković, Antun Drobnjak, and Ivica Botički. 2024. Hallucinations in llms: Understanding and addressing challenges. In *MIPRO ICT and Electronics Convention (MIPRO).* IEEE, 2084–2088.

[70] Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. 2024. Flash: A comprehensive approach to intrusion detection via provenance graph representation learning. In *2024 IEEE Symposium on Security and Privacy (SP).* IEEE, 3552–3570.

[71] David Wood Richard Cyganiak and Markus Lanthaler. 2014. RDF 1.1 concepts and abstract syntax. (2014).

[72] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference.* Springer.

[73] Saad Ullah, Mingji Han, Saurabh Pujar, Hammond Pearce, Ayse Coskun, and Gianluca Stringhini. 2024. Llms cannot reliably identify and reason about security vulnerabilities (yet?): A comprehensive evaluation, framework, and benchmarks. In *2024 IEEE Symposium on Security and Privacy (SP).* IEEE, 862–880.

[74] Thijs Van Ede, Hojjat Aghakhani, Noah Spahn, Riccardo Bortolameotti, Marco Cova, Andrea Continella, Maarten van Steen, Andreas Peter, Christopher Kruegel, and Giovanni Vigna. 2022. Deepcase: Semi-supervised contextual analysis of security events. In *IEEE Symposium on Security and Privacy (SP).* IEEE, 522–539.

[75] Lingzhi Wang, Xiangmin Shen, Weijian Li, Zhenyuan Li, R Sekar, Han Liu, and Yan Chen. 2025. Incorporating gradients to rules: Towards lightweight, adaptive provenance-based intrusion detection. In *Network and Distributed System Security (NDSS) Symposium.*

[76] Qi Wang, Wajih Ul Hassan, Ding Li, Kangkook Jee, Xiao Yu, Kexuan Zou, Junghwan Rhee, Zhengzhang Chen, Wei Cheng, Carl A Gunter, et al. 2020. You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis. In *Network and Distributed Systems Security (NDSS) Symposium.*

[77] Su Wang, Zhiliang Wang, Tao Zhou, Hongbin Sun, Xia Yin, Dongqi Han, Han Zhang, Xingang Shi, and Jiahai Yang. 2022. Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning. *IEEE Transactions on Information Forensics and Security* 17 (2022), 3972–3987.

[78] Xuhong Wang, Baihong Jin, Ying Du, Ping Cui, Yingshui Tan, and Yupu Yang. 2021. One-class graph neural networks for anomaly detection in attributed networks. *Neural computing and applications* 33 (2021), 12073–12085.

[79] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS).* https://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

[80] Zhiqiang Xu, Pengcheng Fang, Changlin Liu, Xusheng Xiao, Yu Wen, and Dan Meng. 2022. Depcomm: Graph summarization on system audit logs for attack investigation. In *IEEE Symposium on Security and Privacy (SP).* IEEE, 540–557.

[81] Zhiming Xu, Xiao Huang, Yue Zhao, Yushun Dong, and Jundong Li. 2022. Contrastive attributed network anomaly detection with data augmentation. In *Pacific-Asia conference on knowledge discovery and data mining.* Springer, 444–457.

[82] Na Yan, Yu Wen, Luyao Chen, Yanna Wu, Boyang Zhang, Zhaoyang Wang, and Dan Meng. 2022. Deepro: Provenance-based APT Campaigns Detection via GNN. In *IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom).* IEEE, 747–758.

[83] Fan Yang, Jiacen Xu, Chunlin Xiong, Zhou Li, and Kehuan Zhang. 2023. PROGRAPHER: An Anomaly Detection System based on Provenance Graph Embedding. In *USENIX Security Symposium.* 4355–4372.

[84] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316.

[85] Javier Yong, Haokai Ma, Yunshan Ma, Anis Yusof, Zhenkai Liang, and Ee-Chien Chang. 2025. AttackSeqBench: Benchmarking Large Language Models' Understanding of Sequential Patterns in Cyber Attacks. *arXiv preprint arXiv:2503.03170* (2025).

[86] Jun Zeng, Zheng Leong Chua, Yinfang Chen, Kaihang Ji, Zhenkai Liang, and Jian Mao. 2021. WATSON: Abstracting Behaviors from Audit Logs via Aggregation of Contextual Semantics.. In *Network and Distributed System Security (NDSS) Symposium.*

[87] Jun Zengy, Xiang Wang, Jiahao Liu, Yinfang Chen, Zhenkai Liang, Tat-Seng Chua, and Zheng Leong Chua. 2022. Shadewatcher: Recommendation-guided cyber threat analysis using system audit records. In *IEEE Symposium on Security and Privacy (SP).* IEEE, 489–506.

[88] Yongheng Zhang, Tingwen Du, Yunshan Ma, Xiang Wang, Yi Xie, Guozheng Yang, Yuliang Lu, and Ee-Chien Chang. 2025. AttacKG+: Boosting attack graph construction with Large Language Models. *Computers & Security* 150 (2025), 104220.

[89] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems* 36 (2024).

[90] Michael Zipperle, Florian Gottwalt, Elizabeth Chang, and Tharam Dillon. 2022. Provenance-based Intrusion Detection Systems: A Survey. *ACM Computing Surveys (CSUR)* (2022).

A Provenance Graphs Schema

Figure 5 presents the schema of the provenance graph for the CADETS host, illustrating system entities and the events connecting them. The schema includes a diverse range of system entities such as processes, files, and network flows. The relationships between
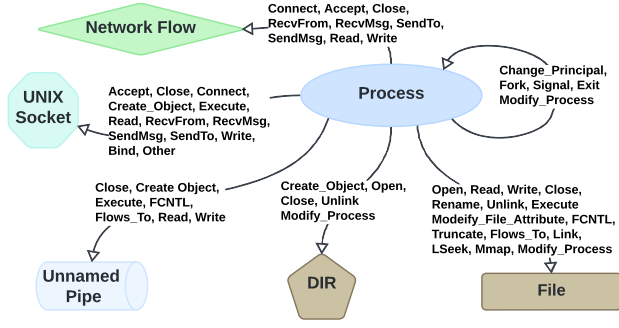
Figure 5: Schema of provenance graphs for the CADETS host.

these entities include actions such as 'read', 'write', and 'execute', as well as network communications like 'send' and 'receive'.

B   LLM Prompts

In this section, we present the prompts and instructions used by our LLM-based attack investigation module. The system begins by configuring the LLM as an APT investigator, responsible for producing factual and well-structured attack reports.

Investigator Instructions: You are an advanced persistent threat (APT) attack investigator, skilled at summarizing log events related to anomaly detection alerts into comprehensive attack reports. You possess deep expertise in APTs, Cyber Threat Intelligence (CTI), and operating system security.
Guidelines: Focus on delivering factual, high-quality analysis in a human-like narrative. Ensure all information is accurate and directly sourced from the document. Do not introduce any details not present in the document, avoiding any fabrications or hallucinations. Keep a detailed account of the attack's execution, including specific timestamps. All responses should be formatted in Markdown.
Definitions: The APT stages are: Initial Compromise, Internal Reconnaissance, Command and Control, Privilege Escalation, Lateral Movement, Maintain Persistence, Data Exfiltration, and Covering Tracks. Indicators of Compromise (IOCs) include: External IP addresses. Suspicious or executable files suspected to be potential threats. Processes with moderate to high likelihood of exploitation.
Your task is to generate an attack report that includes the following sections: A concise summary of the attack behavior, detailing key events and actions taken during the incident. Where applicable, specify the corresponding stage of the APT attack. A table of IOCs detected in the document. Based on your cybersecurity expertise, add a concise security context beside each detected IOC, including the legitimate usage and exploitation likelihood. A list of chronological log of actions, organized by minute.

The investigation workflow is broken down into a sequence of subtasks, each guided by a specialized prompt. For each anomalous subgraph, the LLM is first prompted to extract a list of relevant IOCs, returned in Python list format.

Retrieve IOCs Prompt: The provided document contains log events related to anomaly detection alerts. Extract the list of IOCs from the document ASG_{doc}. Return the output only as a Python list, formatted as: ['IOC1', 'IOC2', 'IOC3', etc].

The LLM then uses the IOC list to summarize the serialized subgraph into a structured attack report.

Summarize Report Prompt: Based on the logs in document ASG_{doc} and the extracted IOCs list: $[IOC_{lst}]$. Summarize the ASG_{doc} document into an attack report.
The attack report includes the following sections: A concise summary of the attack behavior, detailing key events and actions taken during the incident. Where applicable, specify the corresponding stage of the APT attack. A table of IOCs detected in the document. Based on your cybersecurity expertise, add a concise security context beside each detected IOC, including the legitimate usage and exploitation likelihood. A list of chronological log of actions, organized by minute.

After that, the system prompts the LLM to extract the top IOCs associated with each APT stage from attack reports.

Retrieve IOCs per APT stage Prompt: The provided reports names are: $[R_{atk}]$. Extract the three highest-priority IOCs related to the stage: stg from each provided reports. Focus on external IP addresses, suspicious or executable files, malicious processes, and exploitable processes. Return the output only as a Python list, formatted as: ['IOC1', 'IOC2', 'IOC3', etc].

The LLM then compiles all attack reports and IOCs into a comprehensive attack report.

Summarize Comprehensive Report Prompt: Based on the provided reports and the extracted IOCs list: $[IOC_{lst}]$. Summarize all provided reports into a comprehensive attack report. Consider all external IP addresses, suspicious or executable files, malicious processes, and exploitable processes referenced in the provided reports.

Next, the system initializes a second LLM as a judge, with a role-specific instruction set for a security analyst, who prioritizes IOCs for deeper inspection.

Analyst Judge Instructions: You are a highly skilled security analyst specializing in Advanced Persistent Threats (APTs), Cyber Threat Intelligence (CTI), and operating system security. Your expertise includes reviewing attack reports and providing actionable insights.
The APT attack stages are: Initial Compromise, Internal Reconnaissance, Command and Control, Privilege Escalation, Lateral Movement, Maintain Persistence, Data Exfiltration, and Covering Tracks.
Your task is to analyze the provided attack report and identify key Indicators of Compromise (IOCs) for further investigation. IOCs include external IP addresses, processes with moderate to high exploitation likelihood, and associated suspected files. Focus on identifying IOCs whose contextual analysis could uncover additional APT attack stages, enabling a comprehensive understanding of the full attack scenario. Prioritize IOCs directly tied to malicious activity, such as command-and-control IPs or malicious executable binaries, while deprioritizing general system processes or indicators linked to benign activities.

The judge LLM is prompted to select the highest-priority IOC in the comprehensive report to guide further investigation.

Select IOC by LLM Judge Prompt: Review the attack report to identify the highest-priority $node_{type}$ IOC for further investigation, that could aid in uncovering additional APT attack stages. Return the IOC only, formatted as 'IOC'.

The selected IOC is used to query historical context from the graph database, which is summarized into additional reports. The final step prompts the LLM to enrich the comprehensive report using this additional context.

Enrich Comprehensive Report Prompt: Enrich the comprehensive attack report R_{comp} by incorporating the summary of the attack report R_{atk}. Consider all external IP addresses, suspicious or executable files, malicious processes, and exploitable processes referenced in the provided reports.

C Recovered Attack Reports

In this section, we present simplified versions of recovered reports.

Figure 6 shows the comprehensive attack report recovered from the TRACE host in DARPA TC3 dataset. The report details the initial compromise stage, where the attacker leveraged IP 128.55.12.73 to deliver a malicious executable as an email attachment through the thunderbird process. It identifies the executable file tcexec, which was downloaded to disk, renamed, and had its attributes modified to maintain persistence. Moreover, it captures the use of the server 162.66.239.75 for command-and-control activities.

Figure 7 presents the attack report recovered from host '51' in the DARPA OpTC dataset, detailing the key stages of a malicious upgrade attack. The attack involved the delivery of the malicious payload ckfgw.exe through a compromised Notepad Plus upgrade process. The recovered report accurately identified several key artifacts, including the malicious binary ckfgw.exe, the command

---

Summary of Attack Behavior

The analysis of the provided reports indicates a coordinated Advanced Persistent Threat (APT) attack characterized by multiple stages, including Initial Compromise, Internal Reconnaissance, Command and Control, Lateral Movement, Maintain Persistence, and Data Exfiltration. The attack involved a series of suspicious activities primarily centered around the execution of the process tcexec, which facilitated various malicious actions.

Key Events:

- **Initial Compromise**: The execution of the tcexec process marked the beginning of the attack. This process was responsible for memory allocation, library loading, and file writing operations, indicating potential exploitation of system vulnerabilities. The logs show the loading of critical libraries such as ld-linux-x86-64.so.2, libc.so.6, and libpthread.so.0, which are essential for executing processes on Linux systems. This could indicate an attempt to exploit vulnerabilities in these libraries. The first instance of loading the library ld-linux-x86-64.so.2 occurred on **2018-04-13 at 13:59** by Thunderbird, indicating a potential entry point of the attack.

- **Command and Control**: The attacker established connections to several external IP addresses, including 128.55.12.55, 128.55.12.67, 128.55.12.73, and 162.66.239.75. These connections suggest attempts to communicate with command and control servers, which are often used to issue commands to compromised systems. Notably, the logs for 128.55.12.55 indicate a pattern of connection and disconnection, characteristic of C2 activity. The consistent communication with the IP address 162.66.239.75 through multiple SENDMSG and RECVMSG actions suggests that this IP is being used for C2 purposes.

- **Internal Reconnaissance**: The logs show multiple connection attempts to various IP addresses, indicating the attacker's efforts to explore the network and identify additional targets for exploitation. The logs reveal multiple connections to various internal IP addresses, indicating an exploration of the network environment, which is typical during the reconnaissance phase of an APT.

- **Maintain Persistence**: The logs from the context file indicate that the file tcexec underwent multiple operations, including being closed, created, modified, renamed, and updated. These actions are indicative of an attacker establishing control over the file, which may serve as a means to execute further malicious activities. The LOADLIBRARY operation involving tcexec suggests that the file is being used to load additional libraries, potentially for executing malicious code or maintaining persistence. The repeated loading of the library ld-linux-x86-64.so.2 by various applications, including Thunderbird and Python3, raises concerns about potential exploitation.

Indicators of Compromise (IoCs)

The following external IP addresses and suspicious files were identified as potential indicators of compromise (IoCs) throughout the reports:

IoC	Security Context
128.55.12.1	Potentially a command and control server; legitimate usage may include internal network services.
128.55.12.55	External IP address involved in multiple connections; high likelihood of exploitation, associated with C2 activity.
128.55.12.67	External IP address with multiple connection attempts; could be associated with malicious activity.
128.55.12.73	External IP address involved in data transfer; potential command and control server.
128.55.12.103	External IP address with connections; may indicate lateral movement or data exfiltration.
128.55.12.110	External IP address involved in connections; potential for exploitation.
128.55.12.141	External IP address with multiple connection attempts; could indicate malicious intent.
128.55.12.166	External IP address involved in connections; potential command and control server.
162.66.239.75	Known external IP address associated with C2 activities. High likelihood of exploitation.
tcexec	The file tcexec is likely a legitimate executable. However, its repeated modifications and library loading suggest it may be exploited for malicious purposes, indicating a moderate to high likelihood of exploitation.
ld-linux-x86-64.so.2	A critical system library; legitimate usage but can be exploited if compromised.
libc.so.6	Standard C library; essential for many applications. Exploitation can lead to privilege escalation.
libpthread.so.0	Library for multi-threading; legitimate but can be targeted for exploitation.
ld.so.cache	Cache for dynamic linker; legitimate usage but can be manipulated for malicious purposes.

Chronological Log of Actions

2018-04-13

- **13:59**: Thunderbird LOADLIBRARY the file: ld-linux-x86-64.so.2
- **14:01**: CREATE_OBJECT the file: tcexec / MODIFY_FILE_ATTRIBUTES the file: tcexec / RENAME the file: tcexec / UPDATE the file: tcexec / CLOSE the file: tcexec
- **14:02**: python3 LOADLIBRARY the file: ld-linux-x86-64.so.2 (2 times)
- **14:13**: Thunderbird LOADLIBRARY the file: ld-linux-x86-64.so.
- **14:20**: tcexec LOADLIBRARY the file: ld-linux-x86-64.so.2 / tcexec MMAP a memory (2 times) / tcexec LOADLIBRARY the file: tcexec / pine EXECUTE the process: tcexec / tcexec MPROTECT a memory / tcexec WRITE a fileChar
- **14:21**: tcexec CONNECT the flow: 128.55.12.73 / tcexec WRITE a fileChar (2 times) / tcexec CLOSE the flow: 128.55.12.73 / tcexec CONNECT the flow: 128.55.12.55 / tcexec CLOSE the flow: 128.55.12.55 / tcexec OPEN a fileDir / tcexec CONNECT the flow: 128.55.12.67 (2 times)
- **........ ....**
- **14:25**: tcexec CLOSE the flow: 103.12.253.24
- **14:28**: tcexec EXIT the process: tcexec

The logs and identified IoCs suggest a sophisticated APT attack with multiple stages and indicators of compromise. Immediate action is recommended to investigate the identified IP addresses and mitigate any potential threats to the network. Further analysis and monitoring of the affected systems are essential to prevent future incidents.

Figure 6: A simplified version of the comprehensive attack report recovered from the 'TRACE' host in DARPA TC3 dataset.

---

and control server 53.192.68.50, the shell script cmd.exe used to execute commands, and the scanned IP address 142.20.56.52. It also captured schtasks.exe, which was employed to establish persistence via scheduled tasks. Notably, all these artifacts, along with their timestamps, align with the details provided in the ground truth report. Additionally, the recovered report detected an additional executable binary, biGuWCmNsuCIG.exe, written by cKfGW.exe but not explicitly mentioned in the ground truth report.

Summary of Attack Behavior

The analysis of the provided reports indicates a coordinated attack involving multiple stages of an Advanced Persistent Threat (APT) on September 25, 2019. The attack primarily utilized the `python.exe`, `svchost.exe`, and other processes to execute malicious activities, including data exfiltration, internal reconnaissance, and potential command and control communications. The presence of various suspicious files and external IP addresses suggests a well-planned operation aimed at compromising sensitive information and maintaining persistence within the target environment.

Key Events:

- **Internal Reconnaissance**: The logs show extensive file reading activities, including multiple `.pyc` files and configuration files, indicating that the attacker was exploring the environment to identify potential targets and vulnerabilities. The use of commands via `cmd.exe`, `wmiprvse.exe`, and other system processes suggests that the attacker was gathering information about the system and its environment. The communication pattern between `142.20.56.52` and `53.192.68.50` further indicates ongoing reconnaissance activities. Additionally, the logs from the `context_ckfgw` document show repeated inbound messages from `239.255.255.250`, suggesting that the attacker was gathering information about the network environment.
- **Command and Control**: Outbound connections were established to external IP addresses, notably `142.20.61.132`, `10.20.2.66`, and `53.192.68.50`, which are suspected command and control servers. The repeated communication with these addresses suggests attempts to receive commands or exfiltrate data. The consistent outbound connections to `53.192.68.50`, particularly from processes like `biguwcmnsucig.exe` and `GoogleUpdate.exe`, indicate that the attacker was likely maintaining a command and control channel to execute further commands or exfiltrate data. The logs also indicate consistent communication with the multicast address `239.255.255.250`, which may suggest a C2 channel being established for further instructions or data exfiltration.
- ....

Table of Indicators of Compromise (IoCs)

IoC	Security Context
python.exe	Legitimate Python executable; high likelihood of exploitation if used to run malicious scripts.
cKfGW.exe	Suspicious executable; potential for malicious activity.
53.192.68.50	External IP address; potential command and control server.
142.20.61.132	External IP address; potential command and control server.
142.20.56.52	External IP address; potential command and control server.
239.255.255.250	Multicast address; legitimate in certain contexts but often exploited for reconnaissance.
... ...	... ...
biguwcmnsucig.exe	Executable file exhibiting suspicious behavior; high likelihood of exploitation.
conhost.exe	Legitimate Windows process; can be exploited for malicious purposes. Moderate exploitation risk.
taskhostw.exe	Legitimate Windows process; can be exploited for malicious purposes. Moderate exploitation risk.
backgroundtaskHost.exe	Legitimate Windows process; can be exploited for malicious purposes. Moderate exploitation risk.
cmd.exe	Command-line interface that can be used for legitimate or malicious commands. Moderate risk.
schtasks.exe	Windows task scheduler; can be used to create scheduled tasks for persistence. Moderate risk.
wmiprvse.exe	Windows Management Instrumentation process; can be exploited for reconnaissance. Moderate risk.
csrss.exe	Client/Server Runtime Subsystem; critical for Windows, can be targeted for exploitation. High risk.
cscript.exe	Windows script host for executing scripts; can be used for malicious scripts. Moderate risk.
GoogleUpdate.exe	Legitimate updater for Google applications; can be exploited for persistence. Moderate risk.

Chronological Log of Actions

September 25, 2019

- **09:19**: `START_INBOUND` the flow: 239.255.255.250 (19 times) / `MESSAGE_INBOUND` the flow: 239.255.255.250 (16 times) / `MESSAGE_OUTBOUND` the flow: 239.255.255.250 (2 times) / `START_OUTBOUND` the flow: 239.255.255.250 / `READ` the file: SVCHOST.EXE-135A30D8.pf
- **09:20**: The process `python.exe` was invoked to read multiple files, including sensitive files such as `node_id.txt` and `ncr.key`. / `START_INBOUND` the flow: 239.255.255.250 (18 times) / `MESSAGE_INBOUND` the flow: 239.255.255.250 (15 times)
- **09:23 - 10:11**: `START_INBOUND` the flow: 239.255.255.250 (171 times) / `MESSAGE_INBOUND` the flow: 239.255.255.250 (94 times)
- **10:27**: The process `conhost.exe` initiated outbound communication to the IP address 53.192.68.50.
- **10:30**: The process `biguwcmnsucig.exe` initiated outbound communication to the IP address 53.192.68.50.
- **10:47**: The process `biguwcmnsucig.exe` initiated outbound communication to the IP address 53.192.68.50.
- **10:48**: The process `taskhostw.exe` initiated outbound communication to the IP address 53.192.68.50.
- **10:49**: The process `biguwcmnsucig.exe` initiated outbound communication to the IP address 53.192.68.50.
- **10:54**: The process `backgroundtaskHost.exe` initiated outbound communication to the IP address 53.192.68.50.
- **10:55**: The process `biguwcmnsucig.exe` initiated outbound communication to the IP address 53.192.68.50.
- **10:57**: The process `cmd.exe` initiated outbound communication to the IP address 53.192.68.50.
- **10:59**: The process `schtasks.exe` initiated outbound communication to the IP address 53.192.68.50.
- **11:00**: The process `conhost.exe` initiated outbound communication to the IP address 53.192.68.50.
- ... ...

This report highlights the suspicious activities and potential indicators of compromise that warrant further investigation to mitigate any potential threats. Immediate actions should be taken to secure the environment and analyze the extent of the compromise.

Figure 7: A simplified version of the comprehensive attack report recovered from host '51' in the DARPA OpTC dataset, where the red team performed a Malicious Upgrade attack.

Summary of Attack Behavior

The analysis of the provided reports indicates a coordinated and sophisticated attack, likely an advanced persistent threat (APT), characterized by multiple stages of exploitation and manipulation. The logs span various timestamps, primarily focusing on the activities of the `svchost.exe` process, which is commonly exploited by attackers to execute malicious actions while masquerading as a legitimate system process.

Key Events and Stages Identified:

- **Internal Reconnaissance**:
  - The logs indicate multiple read and write operations on various files, including suspicious files such as `8d273d55-059f-4c89-9fd2-587b4bad1ce4_5-1-5-21-4190936083-3304963419-1584388968-1105_35.rslc`, `setuptools-0.7.2-py2.7.egg`, and `kickoff.log`, which may have been used to gather information about the system and its configurations. The use of `NETSTAT.EXE` and `PING.EXE` further suggests that the attacker was gathering information about the network and available hosts. The repeated access to `svchost.exe` and its associated prefetch files (`SVCHOST.EXE-CA1952BB.pf`, `SVCHOST.EXE-25622318.pf`, and `SVCHOST.EXE-824A39CF.pf`) indicates an attempt to gather information about the system and its services.
- **Command and Control**:
  - Repeated outbound messages to `202.6.172.98`, `142.20.57.246`, and other internal IP addresses highlight the establishment of a C2 channel, allowing the attacker to maintain control over the compromised system. The consistent outbound messages from various processes to the IP address `202.6.172.98` suggest that this IP may be a command and control server, facilitating further instructions to the compromised system.
- **Privilege Escalation**:
  - The use of legitimate modules such as `Microsoft.Powershell.Commands.Diagnostics.ni.dll` and `Microsoft.WSMan.Management.ni.dll` indicates attempts to escalate privileges and gain unauthorized access to sensitive system functionalities.
- ....

Table of Indicators of Compromise (IoCs)

IoC	Security Context
202.6.172.98	An external IP address frequently used for C2 communications. High likelihood of exploitation.
142.20.57.246	Another external IP address involved in inbound communications. Potentially malicious.
10.20.0.2	Internal IP address; potential C2 server.
10.50.2.101	Internal IP address; potential C2 server.
10.50.5.11	Internal IP address; potential C2 server.
svchost.exe	A legitimate Windows process that can be exploited for malicious purposes. Moderate to high risk.
... ...	... ...
powershell.exe	Legitimate Windows process often used for scripting; can be exploited for malicious commands.
lsass.exe	Windows process for managing security policies; can be targeted for credential harvesting.
cmd.exe	Command-line interpreter; can be used for executing commands and scripts maliciously.
wmiprvse.exe	Windows Management Instrumentation process; can be exploited for remote management tasks.
GoogleUpdate.exe	Legitimate updater for Google applications; can be misused for persistence.
taskhostw.exe	Windows process for running tasks; can be exploited for scheduled tasks.
conhost.exe	Console host process; can be used to execute commands in a console window.
compattelrunner.exe	Windows process for compatibility telemetry; can be exploited for persistence.
NETSTAT.EXE	Network utility to display active connections; can be used for reconnaissance.
PING.EXE	Utility for testing network connectivity; can be used for reconnaissance.
schtasks.exe	Utility for managing scheduled tasks; can be exploited for persistence.
... ...	... ...

Chronological Log of Actions

September 24, 2019

- **10:30**: NGentask.exe, conhost.exe, csrss.exe, ngen.exe, services.exe, and svchost.exe read SVCHOST.EXE-CA1952BB.pf.
- **12:22**: lsass.exe, services.exe, and svchost.exe read svchost.exe.
- ... ...
- **14:10**
  - svchost.exe wrote mantra.log (8 times).
  - svchost.exe sent outbound messages to 202.6.172.98 (7 times).
- **14:19**: services.exe and svchost.exe read svchost.exe.
- **14:28**: lsass.exe and services.exe read svchost.exe.
- **16:23**
  - wmiprvse.exe, AUDIODG.EXE, lsass.exe, and services.exe read SVCHOST.EXE-824A39CF.pf.
  - lsass.exe and services.exe read SVCHOST.EXE::$EA and svchost.exe.

The logs from the provided reports reveal a complex and multi-faceted attack involving the exploitation of legitimate processes and the establishment of C2 communications. The identified IoCs warrant immediate investigation and remediation actions to mitigate the risks associated with this APT attack. Continuous monitoring and analysis of network traffic and system behavior are essential to prevent further exploitation and ensure the integrity of the affected systems.

Figure 8: A simplified version of the comprehensive report recovered from host '501' in DARPA OpTC dataset, where the red team performed a Custom PowerShell Empire attack.

Figure 8 showcases the attack report recovered from host '501' of the DARPA OpTC dataset, highlighting a Powershell Empire attack scenario. The report successfully identified key elements of the attack, including the `powershell.exe` script injected during the initial compromise, the command and control server at `202.6.172.98`, the windows management instrumentation process `wmiprvse.exe` exploited for privilege escalation, and the `schtasks.exe` process utilized to manage and automate scheduled tasks. Furthermore, it highlighted the network utility process `netstat.exe` used for reconnaissance and the scanned IP address `142.20.57.246`.

D Multi-Hop Expansions Analysis

Table 7 compares one-hop and two-hop expansion strategies across multiple datasets. While two-hop expansion slightly reduces false negatives in some cases (e.g., TC3 THEIA, OpTC 201, and OpTC 51), it often introduces a large number of false positives, significantly reducing precision. For example, in THEIA, false negatives reduced from 5 to 2, but false positives rose sharply from 0 to 21.5 K, reducing

Table 7: Impact of one-hop vs. two-hop expansion on detection performance across datasets.

Dataset	Expansion	FP	FN	Precision	Recall	F1-Score
TC3	one-hop	0	2	1.00	1.00	1.00
(CADETS)	two-hop	0	2	1.00	1.00	1.00
TC3	one-hop	173	1	1.00	1.00	1.00
(TRACE)	two-hop	269	1	1.00	1.00	1.00
TC3	**one-hop**	**0**	**5**	**1.00**	**1.00**	**1.00**
(THEIA)	two-hop	21.5K	2	0.50	1.00	0.67
OpTC	one-hop	0	7	1.00	0.88	0.94
(201)	two-hop	3	6	0.95	0.90	0.92
OpTC	one-hop	0	0	1.00	1.00	1.00
(501)	two-hop	2	0	0.99	1.00	1.00
OpTC	one-hop	17	39	0.89	0.77	0.82
(51)	two-hop	26	29	0.84	0.83	0.84
NODLINK	one-hop	1	0	0.95	1.00	0.97
(Ubuntu)	two-hop	0	0	1.00	1.00	1.00
NODLINK	one-hop	13	3	0.74	0.93	0.82
(WS 12)	two-hop	13	3	0.74	0.93	0.82
NODLINK	one-hop	9	1	0.95	0.99	0.97
(W 10)	two-hop	0	1	1.00	0.99	1.00

precision from 1.0 to 0.5. In most datasets, the gains from two-hop expansion are minimal or negligible. Based on this trade-off, we selected one-hop expansion as the default expansion method.

E Feature Selection Analysis

As part of our feature selection process, we evaluated two additional temporal features: Lifespan, defined as the duration between a node's first and last observed actions, and Cumulative Active Time, defined as the total time between consecutive actions with gaps under one second. To assess their effectiveness, we created three system variants: our proposed system, one with the Lifespan feature (*With Lifespan*), and one with the Cumulative Active Time feature (*With CumActive*). OCR-APT uses two core behavioral features—normalized action frequency and idle period statistics—selected for their ability to generalize across diverse hosts.

As shown in Table 8, the Lifespan feature yielded minor improvements on a few hosts. On OpTC 51, it slightly boosted recall compared to OCR-APT, and on Ubuntu and W10, it slightly improved precision by eliminating false positives. However, its performance dropped sharply on other hosts. On WS12, it led to complete failure—F1 score fell to zero due to missed detections ($TP = 0$). Similarly, on OpTC 201, Lifespan caused a substantial drop in precision (from 1.00 to 0.49), severely degrading the F1 score.

The Cumulative Active Time feature showed a modest benefit only on W10, where it slightly reduced false positives compared to OCR-APT. On all other hosts, however, it either did not improve performance or led to degradation. On WS12, its inclusion once again led to detection failure, replicating the poor performance observed with Lifespan on this host. On OpTC 201, it sharply reduced precision (to 0.44), and on Ubuntu, it introduced a large number of false positives (23, compared to just one in OCR-APT).

Overall, although both features offered marginal gains on a few hosts, their lack of stability and the significant performance drops on others led us to exclude them from the final system. We also

Table 8: Impact of Lifespan and Cumulative Active Time (CumActive) Features on Detection Performance across datasets.

Dataset	Version	FP	FN	Precision	Recall	F1-Score
TC3 (CADETS)	With Lifespan	0	1	1.00	1.00	1.00
	With CumActive	18	1	1.00	1.00	1.00
	OCR-APT	0	2	1.00	1.00	1.00
TC3 (TRACE)	With Lifespan	38	0	1.00	1.00	1.00
	With CumActive	180	0	1.00	1.00	1.00
	OCR-APT	173	1	1.00	1.00	1.00
TC3 (THEIA)	With Lifespan	137	5	0.99	1.00	1.00
	With CumActive	153	5	0.99	1.00	1.00
	OCR-APT	0	5	1.00	1.00	1.00
OpTC (201)	With Lifespan	57	6	0.49	0.90	0.63
	With CumActive	55	16	0.44	0.73	0.55
	OCR-APT	**0**	**7**	**1.00**	**0.88**	**0.94**
OpTC (501)	With Lifespan	4	0	0.99	1.00	0.99
	With CumActive	1	0	1.00	1.00	1.00
	OCR-APT	0	0	1.00	1.00	1.00
OpTC (51)	With Lifespan	20	36	0.87	0.79	0.83
	With CumActive	17	39	0.89	0.77	0.82
	OCR-APT	17	39	0.89	0.77	0.82
NODLINK (Ubuntu)	With Lifespan	0	0	1.00	1.00	1.00
	With CumActive	23	0	0.44	1.00	0.61
	OCR-APT	1	0	0.95	1.00	0.97
NODLINK (WS 12)	With Lifespan	78	40	0.00	0.00	0.00
	With CumActive	54	40	0.00	0.00	0.00
	OCR-APT	**13**	**3**	**0.74**	**0.93**	**0.82**
NODLINK (W 10)	With Lifespan	0	1	1.00	0.99	1.00
	With CumActive	3	1	0.98	0.99	0.99
	OCR-APT	9	1	0.95	0.99	0.97

excluded features such as *most active hours* due to limited generalizability and potential dataset bias. Simulated datasets may not reflect real-world attacker behavior, as adversaries can evade detection by operating during typical business hours—periods that are increasingly difficult to define due to flexible work schedules and remote access. Our findings validate the selected feature set, though a broader exploration of alternative temporal features remains an open direction for future research.