ANTEATER: A Filter-then-Scrutinize Architecture for End-to-End Attack Investigation

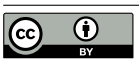
YIMING REN, Institute of Information Engineering, Chinese Academy of Sciences; School of Cyber Security, University of Chinese Academy of Sciences, China
HAOQIANG WANG, Institute of Information Engineering, Chinese Academy of Sciences; School of Cyber Security, University of Chinese Academy of Sciences, China
LINGHAO LI, Institute of Information Engineering, Chinese Academy of Sciences; School of Cyber Security, University of Chinese Academy of Sciences, China
HAOYANG CHEN, Institute of Information Engineering, Chinese Academy of Sciences; School of Cyber Security, University of Chinese Academy of Sciences, China
CHENGXIANG SI*, National Computer Network Emergency Response Technical Team/Coordination Center of China, China
ZHOU ZHOU, Institute of Information Engineering, Chinese Academy of Sciences, China
QINGYUN LIU, Institute of Information Engineering, Chinese Academy of Sciences, China

Audit logs serve as a fundamental data source for system security. However, extracting high-value threat information from massive log data poses significant data management challenges: traditional unsupervised models produce high false positive rates due to their inherent assumption of equating statistical anomalies with malicious activities; Emerging Large Language Model (LLM) based solutions, despite their powerful semantic understanding capabilities, are constrained by high computational costs and context window lengths, making it difficult to detect attacks within massive logs. Furthermore, the outputs of existing methods differ significantly from the practical attack reports required by security analysts.

To overcome these limitations, this paper proposes ANTEATER, an innovative end-to-end attack investigation framework based on raw logs that features a cascading "filter-then-scrutinize" architecture. The "Filter" stage is a lightweight, flow-based anomaly detection model that efficiently filters massive logs and reduces the data scale for investigation. Subsequently, the "Scrutinize" stage is an attack investigation model with a three-agent LLM collaboration. It operates on a provenance graph constructed from the filtered anomalous logs. The agents collaboratively and autonomously explore and reconstruct the attack subgraph, then generate a structured natural-language report. ANTEATER not only effectively mitigates the LLM bottleneck from cost and context window, enabling it to tackle long-term, stealthy attacks, but also bridges the critical gap between raw data detection and the generation of readable attack reports.

*Corresponding author.

Authors' Contact Information: Yiming Ren, Institute of Information Engineering, Chinese Academy of Sciences; School of Cyber Security, University of Chinese Academy of Sciences, China, renyiming@iie.ac.cn; Haoqiang Wang, Institute of Information Engineering, Chinese Academy of Sciences; School of Cyber Security, University of Chinese Academy of Sciences, China, wanghaoqiang@iie.ac.cn; Linghao Li, Institute of Information Engineering, Chinese Academy of Sciences; School of Cyber Security, University of Chinese Academy of Sciences, China, lilinghao@iie.ac.cn; Haoyang Chen, Institute of Information Engineering, Chinese Academy of Sciences; School of Cyber Security, University of Chinese Academy of Sciences, China, chenhaoyang@iie.ac.cn; Chengxiang Si, National Computer Network Emergency Response Technical Team/Coordination Center of China, China, sichengxiang@cert.org.cn; Zhou Zhou, Institute of Information Engineering, Chinese Academy of Sciences, China, zhouzhou@iie.ac.cn; Qingyun Liu, Institute of Information Engineering, Chinese Academy of Sciences, China, liuqingyun@iie.ac.cn.



ACM 2836-6573/2026/6-ART135
https://doi.org/10.1145/3802012

CCS Concepts: • **Security and privacy → Intrusion detection systems**.

Additional Key Words and Phrases: Attack Investigation, Anomaly Detection, Provenance Graph

ACM Reference Format:
Yiming Ren, Haoqiang Wang, Linghao Li, Haoyang Chen, Chengxiang Si, Zhou Zhou, and Qingyun Liu. 2026. ANTEATER: A Filter-then-Scrutinize Architecture for End-to-End Attack Investigation. *Proc. ACM Manag. Data* 4, 3 (SIGMOD), Article 135 (June 2026), 27 pages. https://doi.org/10.1145/3802012

1 Introduction

System logs serve as a fundamental data asset in modern large-scale software infrastructures. Given the rich runtime information they record, automated log analysis techniques, particularly anomaly detection, have become a significant research area in data management [35, 47]. Beyond this, attack investigation represents a more challenging task, as it requires systems to comprehend the deep semantics of log data rather than simply detecting anomalies. It must trace causal dependencies across massive, fragmented records and reveal malicious behaviors hidden among normal system operations [3, 43]. This imposes demanding requirements on both the throughput capacity and the reasoning capability of the data processing pipeline.

When processing evolving patterns shifts and unknown threats in massive log data, supervised learning methods that rely on prior labels have proven inadequate [45, 68]. Consequently, research trends have shifted toward unsupervised anomaly detection frameworks that aim to detect unknown threats by learning the distribution of benign system behaviors [34]. Existing work typically adopts two data representation forms: one directly processes log text sequences and detects outliers through feature encoding [15, 18, 32, 78]; the other transforms logs into structured provenance graphs and captures causal dependencies among entities through topological analysis [36, 53, 72]. However, both traditional approaches face a common bottleneck in handling unstructured data: statistical anomalies do not equate to semantic malice. Whether based on sequence statistics or graph topology, these methods can only reflect deviations in data distribution and lack a deep semantic understanding of log content, resulting in persistently high false positive rates [48, 49].

To address the semantic limitations of traditional methods, recent research has begun exploring the integration of LLMs [31, 46, 63]. With their powerful pre-trained knowledge and reasoning capabilities, LLMs are regarded as ideal tools for bridging the semantic gap and interpreting complex log intentions [58, 59]. However, directly integrating LLMs into high-throughput log analysis pipelines presents significant challenges: their high inference costs and limited context windows result in prohibitive computational overhead and latency bottlenecks, making real-world applications for processing massive raw log streams highly impractical.

Limitations. Existing research presents three key limitations:

L1: *Ineffectiveness in utilizing LLMs.* Although LLMs can be applied to log anomaly detection without additional training, directly using them for detection faces dual constraints in both capability and efficiency. First, cyberattack behaviors are often scattered across massive volumes of normal logs spanning extended time windows [19, 57, 70]. To detect such attacks, the model must process and comprehend an extremely long log sequence within a single context window, which exceeds the capability boundaries of current LLMs. Second, directly inputting large volumes of raw logs into LLMs for analysis also entails prohibitive time overhead and token costs, which are unsustainable in real-world deployment.

L2: *Difficulty in distinguishing malicious behavior from anomalies.* Anomalous behavior merely represents statistical deviations from the system's normal baseline, whereas malicious behavior refers to attack activities intended to compromise the system. Because of the lack of deep semantic understanding of unstructured log content, traditional methods rely on a fragile assumption: that

statistical rarity equates to behavioral malice. However, recent studies on alert measurement have shown that up to 76% of statistical anomalies in real-world environments are benign triggers with no malicious intent [73]. This deficiency in semantic understanding capability leads existing systems to generate numerous false positives in complex detection scenarios, failing to achieve high-precision threat discovery.

L3: *Gap between detection results and attack investigation.* Existing methods suffer from a severe usability gap in their outputs. Although many approaches claim to achieve attack investigation, their outputs are often tens of thousands of fragmented logs that are neither independent nor free of substantial noise, or provenance graphs that have undergone aggressive simplification (e.g., node merging, graph partitioning). However, such simplification inevitably loses critical behavioral and temporal information, making it impossible to reconstruct the attack process on the distorted graph directly, even when anomalous nodes are detected. If one attempts to trace back to the original logs using these coarse-grained nodes, the one-to-many mapping between nodes and log entries dramatically amplifies false positives, rendering the results equally unusable. Empirical data shows that analysts face massive volumes of alerts daily, of which only approximately 0.01% are related to actual attacks [73]. This fragmented, unintegrated output format fails to bridge the gap between detection results and readable reports.

To address the aforementioned limitations, this paper proposes ANTEATER, an end-to-end attack investigation framework based on the "Filter-then-Scrutinize" design philosophy. The core idea behind this architecture is to decouple the attack investigation task and match the optimal processing paradigm according to the requirements and data characteristics of different stages. Specifically, the front-end anomaly detection module ("Filter") employs a streaming generative model-based detector to perform rapid and efficient preliminary filtering of log texts, thereby circumventing the computational costs and context bottlenecks associated with directly applying LLMs (addressing L1). The backend is designed as a tri-agent collaborative attack investigation module ("Scrutinize"), which leverages LLMs to conduct in-depth causal inference and semantic scrutiny on the provenance graph constructed from the filtered logs. By introducing graph structural constraints and multi-agent collaboration, this module can precisely distinguish statistical deviations from genuine malicious intent (addressing L2). The attack chain provenance graph constructed during this process of confirming attack intent, along with the natural language report generated from this graph, serves as a direct output that successfully bridge the gap between raw detection data and readable security results (addressing L3).

We conducted extensive experiments on six datasets. The results demonstrate that ANTEATER can reconstruct attack processes from raw log data and generate readable reports. Specifically, its anomaly detection module achieves an average TPR of 99.8% and FPR of 7.66% across 16 subsets, outperforming existing state-of-the-art (SOTA) methods. Benefiting from a hierarchical architecture, ANTEATER reduces the scale of provenance graphs requiring manual inspection by 92.8%. Compared with directly using LLMs for investigation, token overhead is reduced by 600×, and request interactions are reduced by 40×. Finally, the experiments validate the architecture's scalability and the system's robustness against suboptimal training data and evasion attacks.

Contribution. We summarize our contributions as follows:

• *New paradigm: An end-to-end detection paradigm for attack investigation.* To address the limitations of existing unsupervised methods conflating statistical anomalies with malicious behaviors (L2) and the disconnect between output results and expert requirements (L3), we propose a new paradigm for attack investigation: not only detecting anomalies but also distinguishing malicious intent and generating readable reports. This paradigm bridges the gap between raw log detection and attack investigation requirements.

• *Novel architecture: A cascaded framework based on "Filter-then-Scrutinize."* To mitigate the computational costs and context bottlenecks (L1) faced when directly applying LLMs to massive logs, we designed the ANTEATER architecture. By synergizing efficient filtering from a lightweight front-end model with deep reasoning from a backend LLM, this architecture constitutes the first end-to-end log analysis framework that balances large-scale data-processing efficiency with deep semantic understanding.
• *New techniques: Task-oriented efficient detection and reasoning models.* We designed two key techniques to support this architecture: 1) An unsupervised detection model combining BERT with a flow-based generative model, achieving efficient log filtering without pre-training; 2) A multi-agent collaborative attack investigation engine that, through the division of labor among exploration, reconstruction, and management agents, overcomes the challenge of long-range causal reasoning on provenance graphs and achieves automated attack event reconstruction.

2 Preliminary and Related Work

2.1 Anomaly Detection and Attack Investigation

Log data has become essential in modern computing infrastructure, and extracting high-value information from massive volumes of logs has drawn significant attention from both data management [10, 35, 42, 47] and cybersecurity communities [4, 39, 60, 74]. Among various analytical tasks, anomaly detection represents a key technique [28] and aims to identify events that deviate from the system's benign baseline [41]. However, these anomalous events comprise not only malicious attacks but also system failures or rare benign operations. Furthermore, attack investigation [20] focuses on the in-depth analysis of these suspicious events to reconstruct the complete attack path [21, 76], clarify its intent and impact, ultimately producing attack reports [5, 26]. Based on anomaly detection, attack investigation [8, 27, 33, 68] represents a more advanced and challenging requirement for security analysis.

2.2 Log Text and Provenance Graph

Logs serve as the fundamental data carrier for system operational states and user activity trails, primarily generated by operating systems and applications, providing core evidence for attack forensics [18]. Specifically, raw audit logs contain not only timestamps as temporal information but also detailed records of subject entities (e.g., process IDs), object entities (e.g., file paths and IP addresses), and specific operation primitives. To enable cross-domain attack investigation, the heterogeneity of multi-source logs in data schemas must be eliminated [40]. Therefore, we adopt the provenance graph as a unified canonical data model [14]. Formally, we model the provenance graph as a directed graph $G = (V, E)$. In this structure, the node set V corresponds to various entities in the system (encompassing categories such as Process, File, and Socket), while the edge set E captures the information flow and causal dependencies between entities (specifically manifested as system calls such as Read, Write, Execute, and Connect). Through this explicit causal encoding, the provenance graph reconstructs discrete log entries into a cohesive graph structure, thereby providing robust support for complex attack provenance.

2.3 Research Status

Supervised Methods. Supervised detection methods learn attack patterns by training models on known attack logs and use these patterns to identify malicious behavior in new logs. HOLMES [49] employs prerequisite-consequence patterns to detect an attack chain by correlating suspicious activities based on the information flow between them. For known attack events, POIROT [48] abstracted their behavior into a provenance graph; for test data, it discovered behaviors similar to

known attacks through inexact graph pattern matching. ATLAS [2] proposed that different attacks might share similar abstract strategies. Based on this, it constructed a set of candidate sequences related to anomalous entities. It used a sequence-based model to identify entities similar to those in known attack sequences, thereby reconstructing the attack process. These studies commonly relied on a labeled training set covering known attacks. However, this assumption is often untenable. Because attacks continuously evolve, models that depend on predefined attack patterns struggle to detect novel or highly customized threats. [62].

Unsupervised Methods. Unsupervised methods model behavioral patterns and identify anomalous behaviors that deviate from those patterns. UNICORN [25] explores provenance graphs using multi-hop exploration and evolutionary modeling, discovering attack behaviors in the system through graph analysis. ThreaTrace [69] uses an inductive graph neural network to learn the connectivity structure of each benign entity in the provenance graph. During detection, the model predicts the types of all nodes and flags those with incorrect predictions as anomalous. PROGRAPHER [72] splits provenance graphs by time and combines graph embedding (graph2vec) with sequence learning (TextRCNN) to detect anomalies in the provenance graphs. MAGIC [36] applies graph masking techniques to the detection task [29]; it models the system environment with a GAT-based model and ultimately detects anomalous entities by analyzing node embedding differences.

NeuralLog [78] directly constructs a Transformer-based classifier without first parsing the logs. RT-Log [35] dynamically captures feature interactions between log fields through adaptive relational modeling and learns environment-invariant representations to achieve cross-environment anomaly detection. AIRTAG [15] takes log text as input and combines BERT with a one-class support vector machine to achieve fine-grained log detection. PreLog [41] captures log semantics by designing log-specific, entry-level and sequence-level pre-training objectives and transfers knowledge to downstream tasks using prompt-tuning.

Although unsupervised methods alleviate dependence on attack samples, their core flaw is equating statistical anomalies with semantic maliciousness. This paradigm labels any behavior that deviates from the benign baseline as a threat, unable to distinguish between normal system operations and true attack behaviors.

LLM-based Methods. With the rise of LLMs, some researchers have begun to apply these models to log detection tasks. [30, 67]. LogGPT [56] is a ChatGPT-based framework for log-based anomaly detection. It parses log text to extract structured information and leverages ChatGPT's language interpretation capabilities to transfer knowledge from large corpora to log-based anomaly detection [9]. LogPrompt [44] used LLMs to perform zero-shot log analysis tasks through a set of high-level prompting strategies tailored for logs. ITDLM [61] is an LLM fine-tuning-based solution for precise anomaly detection in behavioral logs. However, these LLM-based detection methods directly submit massive logs to the LLM for evaluation, resulting in prohibitive computational overhead and inference latency.

2.4 Knowledge Graph-Augmented LLM

Although LLMs have achieved success, they still face two core limitations in complex knowledge reasoning tasks [64, 65]. First, their high training costs cause knowledge updates to lag, making it difficult to meet timeliness requirements [54, 66]. Second, their reasoning process is opaque and lacks explainability, leading to unreliable results, such as hallucinations [71, 75]. These issues can be partially addressed by introducing external knowledge into the LLM reasoning process. Knowledge Graphs (KGs) provide structured, explicit, and editable knowledge representations that effectively mitigate the limitations of LLMs [50]. RoG [46] generated relation paths based on a KG as faithful plans. It then used these plans to retrieve effective reasoning paths for the LLM to perform inference. TKG [31] is an extension of conventional KGs that adds a temporal dimension,

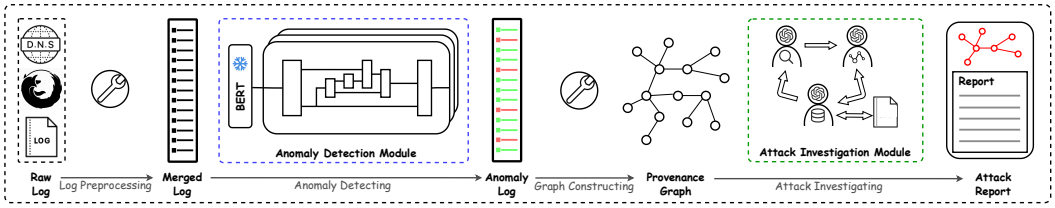


Fig. 1. Overview of the ANTEATER Workflow: An Automated Process from Raw Logs to Attack Reports.

significantly enhancing an LLM's ability to handle complex temporal questions. Subsequently, ToG [63] proposed a new paradigm of iterative interaction between the LLM and KG. In this paradigm, KGs and LLMs work in tandem, complementing each other's capabilities at every step of graph reasoning, thereby enhancing the LLM's ability to solve complex problems. However, these methods typically rely on explicit query entities and static, trustworthy knowledge bases, making them difficult to directly apply to attack investigation tasks that require autonomous exploration of unknown attack entry points within noisy, dynamic provenance graphs.

2.5 Threat Model

Our work is based on the complete trust in the content of audit logs. Ensuring the integrity and authenticity of log data is beyond the scope of this work. Therefore, consistent with prior work [2, 15, 25, 36, 48, 72], we assume that attackers have no means to interfere with or manipulate the generation of audit logs.

2.6 Problem Definition

Based on our threat model, this research focuses on solving the end-to-end attack investigation problem. We expect the solution to be as follows: design an automated framework, ANTEATER, that inputs a massive volume of raw audit logs collected from end-hosts. The framework filters anomalous logs that deviate from normal distributions in large-scale log data and identifies attack intents by inferring causal correlations from them. Ultimately, it outputs structured provenance graphs that document attack events and readable attack reports, without relying on prior attack features or signatures, or requiring human intervention.

3 Overview of System Design

To address the new paradigm outlined above, we propose a revised understanding of the attack investigation task: an attack is a combination of anomalous system behavior and malicious intent. Based on this, we propose ANTEATER, a two-stage framework for attack investigation. As illustrated in Fig. 1, this framework includes an anomaly detection module and an attack investigation module.

The core task of the anomaly detection module is to serve as a high-throughput filter, rapidly selecting high-value candidate logs from massive raw logs. Given the enormous volume and lack of structure in raw log data, we avoid expensive graph construction and instead design a lightweight unsupervised detector based on flow-based models. This model leverages BERT to transform logs into high-dimensional semantic vectors and performs density estimation by learning invertible transformations of benign states, identifying anomalies directly in the vector space through low-probability tail regions. This design achieves extreme computational efficiency while preserving semantic representation capability, reducing millions of logs to a small-scale anomaly candidate set and mitigating the computational cost and context bottlenecks of directly applying LLMs (*L1*).

The core task of the attack investigation module is to perform deep causal analysis and semantic reasoning within a controlled data scope to confirm attack intent. Benefiting from the filtering stage, we can construct provenance graphs that capture rich details of system entity interactions without concerning for scale explosion. Based on this structured data, we design a collaborative framework consisting of three agents: explorer ($A_{explorer}$), reconstructor ($A_{reconstructor}$), and manager ($A_{manager}$). This system performs autonomous breadth exploration and depth reasoning on the graph, distinguishing statistical anomalies from malicious behaviors (*L2*) by verifying whether anomalous events can form complete attack chains that satisfy logical and temporal constraints. Furthermore, it connects fragmented attack nodes into logically coherent attack provenance graphs and generates natural language attack reports, filling the gap in result usability (*L3*).

4 Details of ANTEATER

4.1 Anomaly Log Detection Model

This section details the ANTEATER anomaly detection module, designed to meet the requirements of the "filter-then-scrutinize" architecture while maintaining efficiency and low overhead. Although anomaly detection has been extensively studied, existing text-based methods typically rely on expensive domain-adaptive pre-training , which violates our lightweight design principle. To directly leverage pre-trained BERT's semantic capabilities without additional training, we propose an unsupervised detection model based on Normalizing Flows.

Our model selection is based on an analysis of the characteristics of the BERT embedding space. BERT embeddings exhibit a highly structured anisotropic distribution rather than an isotropic one [16, 17, 37]. Traditional generative models (e.g., VAE) assume that latent variables follow isotropic Gaussian priors, which forces lossy compression when processing non-Gaussian BERT embeddings and destroys rich semantic information [38]. In contrast, Normalizing Flows learn invertible and differentiable transformations for precise probability density estimation on log embeddings, avoiding information loss from approximate inference and enabling direct use of frozen BERT parameters for feature extraction.

We introduce this module in two stages: training and detection. During training, we use BERT as the log text encoder and design a flow-based model to learn the system's benign operational state (Sec. 4.1.1). During detection, we evaluate the anomaly degree of logs using the learned normal behavior distribution and forward anomalous logs to the attack investigation module (Sec. 4.1.2).

4.1.1 ***Training.*** We first preprocessed heterogeneous logs by consolidating multi-line entries into a single-line format. We then used a pre-trained BERT model as a frozen feature extractor. Each entry was tokenized with BERT's WordPiece vocabulary, and the [CLS] token's encoding served as its vector representation. Since the encoder's parameters were frozen across all stages, no further pre-training on log data was required, thereby substantially reducing ANTEATER's computational requirements.

Unlike approaches that rely on approximate inference and latent space regularization, our proposed model employs Normalizing Flows to directly learn the distribution of logs representing normal system behavior. We formalize this task as the estimation of a probability density over the log embedding vectors. Given a log sequence embedding vector $\mathbf{x} \in \mathbb{R}^D$, we aim to learn its probability distribution $p_Z(\mathbf{z})$. The core of this learning process is to construct an invertible and differentiable deep neural network, $f_\theta : \mathbb{R}^D \to \mathbb{R}^D$, which is defined by parameters θ. This function, f_θ, is capable of creating a mapping from a data vector x with a complex, unknown distribution to a vector z in a latent space, where z follows a simple and tractable base distribution $p_Z(\mathbf{z})$. This base distribution is set to the standard multivariate normal distribution, i.e., $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and its probability density function is:

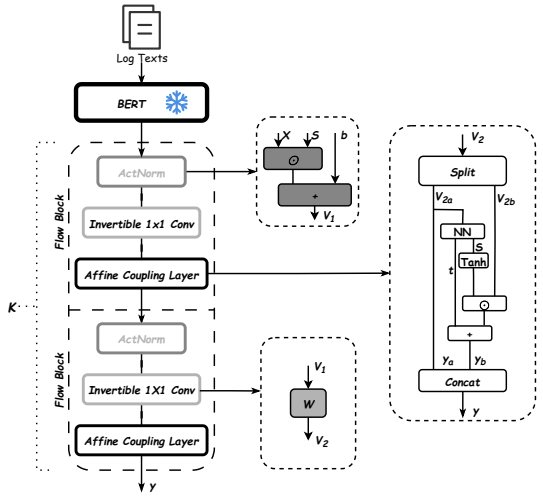


Fig. 2. Anomaly Detection Module Architecture.

$$p_Z(\mathbf{z}) = \frac{1}{(2\pi)^{D/2}} \exp\left(-\frac{1}{2}\mathbf{z}^T\mathbf{z}\right) \tag{1}$$

According to the Change of Variables Theorem from probability theory, we can calculate the probability density of the original data point x using the function f_θ and the base distribution $p_Z(\mathbf{z})$:

$$p_X(\mathbf{x};\theta) = p_Z(f_\theta(\mathbf{x})) \left|\det\left(\frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}}\right)\right| \tag{2}$$

where $\frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian matrix of the transformation function f_θ with respect to the input x, and $|\det(\cdot)|$ denotes the absolute value of its determinant. This determinant term quantifies the local change in spatial volume caused by the transformation f_θ near the point x. For ease of optimization, we operate in the log domain. The Log-Likelihood of x can be expressed as:

$$\log p_X(\mathbf{x};\theta) = \log p_Z(f_\theta(\mathbf{x})) + \log\left|\det\left(\frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}}\right)\right| \tag{3}$$

The model's training objective is to optimize the parameters θ via Maximum Likelihood Estimation (MLE). This is equivalent to minimizing the average Negative Log-Likelihood (NLL) loss function $\mathcal{L}(\theta)$ over the entire training dataset $\mathcal{D}$:

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \log p_X(\mathbf{x};\theta) \tag{4}$$

The model f_θ is trained to learn an optimal transformation. This transformation maps the embedding vectors of the training data to the central, high-probability region of the standard normal distribution, while maximizing the volume expansion of the region where benign log embeddings reside. Ultimately, the model constructs a precise probabilistic description of normal system behavior.

As shown in Fig. 2, the model f_θ is composed of three core invertible transformation components that form the foundation of the Normalizing Flow architecture: Activation Normalization (ActNorm), Invertible 1x1 Convolution (Invertible1x1Conv), and the Affine Coupling Layer (AffineCouplingLayer).

To stabilize the training of our deep flow model, we use ActNorm, a normalization operation that performs an affine transformation, $\mathbf{y} = \mathbf{s} \odot \mathbf{x} + \mathbf{b}$, independently for each feature channel. Its key feature is data-dependent initialization, where the scale s and bias b parameters are initialized based on the mean and standard deviation of the first mini-batch. After initialization, they are treated as standard trainable parameters.

The invertible 1x1 convolution enables effective information flow across feature dimensions by applying a learnable permutation. This permutation is a linear transformation, $\mathbf{y} = \mathbf{W}\mathbf{x}$, using an invertible weight matrix $\mathbf{W} \in \mathbb{R}^{D \times D}$. To ensure initial stability and invertibility, $\mathbf{W}$ is initialized as a random orthogonal matrix via QR decomposition of a random Gaussian matrix.

The Affine Coupling Layer is the primary source of the model's nonlinear capabilities. It operates by splitting the D-dimensional feature vector x along the feature dimension into two D/2-dimensional sub-vectors, $\mathbf{x}_a$ and $\mathbf{x}_b$. The transformation rule is defined as follows:

$$\mathbf{y}_a = \mathbf{x}_a \tag{5}$$

$$\mathbf{y}_b = \mathbf{x}_b \odot \exp(\tanh(s(\mathbf{x}_a))) + t(\mathbf{x}_a) \tag{6}$$

where an independent neural network (a multi-layer perceptron in our implementation) takes $\mathbf{x}_a$ as input to generate a scale vector s and a translation vector t. The symbol $\odot$ denotes element-wise multiplication. To enhance training stability, we apply a tanh activation function to the scale factor s. Furthermore, a zero-initialization strategy is employed, where the weights and biases of the final layer of the neural network used to compute s and t are initialized to zero. This makes each coupling layer behave as an identity transformation at the beginning of training, which facilitates gradient propagation in deep models.

The three components—ActNorm, Invertible1x1Conv, and Affine-CouplingLayer—are combined in sequence to form a reusable, invertible computational unit termed a "Flow-Block". Our model is constructed by stacking K of these Flow-Blocks. As an input vector x passes through this stack, the logarithmic change in its probability density is cumulatively recorded after each block.

4.1.2 ***Detection***. After the model training is complete, the objective of the detection stage is to use the function f_θ, which has learned the distribution of normal behavior, to evaluate the anomaly level of a log entry. For a new input log sequence and its embedding vector $\mathbf{x}_{test}$, we define its anomaly score $S(\mathbf{x}_{test})$ as its exact log-likelihood under the trained model. This score can be efficiently computed with a single forward pass:

$$\begin{aligned} S(\mathbf{x}_{test}) &= \log p_X(\mathbf{x}_{test}; \_\text{trained}) \\ &= \log p_Z(f_\theta(\mathbf{x}_{test})) + \log \left| \det \left(\frac{\partial f_\theta(\mathbf{x}_{test})}{\partial \mathbf{x}_{test}} \right) \right| \end{aligned} \tag{7}$$

This score intuitively quantifies the degree to which the test log entry $\mathbf{x}_{test}$ conforms to the learned normal pattern. The final anomaly decision is made by comparing this score to a threshold τ, which is empirically determined on a validation set:

$$\text{Label}(\mathbf{x}_{test}) = \begin{cases} \text{Normal} & \text{if } S(\mathbf{x}_{test}) \geq \tau \\ \text{Abnormal} & \text{if } S(\mathbf{x}_{test}) < \tau \end{cases} \tag{8}$$

A normal log embedding, which conforms to the patterns seen during training, is mapped by the function f_θ to the central, high-probability region of the latent space Z. Its corresponding Jacobian log-determinant also tends to be positive or a small negative value. Consequently, it yields a high overall log-likelihood score.

Conversely, an anomalous log embedding represents a pattern unseen by the model. When processed by the transformation function f_θ, which is optimized for normal data, two effects typically occur: 1) The mapped vector, Z, falls into the low-probability tail regions of the base distribution, making $\log p_Z(Z)$ a large negative number. 2) The transformation undergoes a drastic local spatial compression to maintain bijection, which causes the Jacobian log-determinant, $\log|\det(J)|$, to also become a large negative number. The combination of these two effects results in a final log-likelihood score, $S(\mathbf{x}_{abnormal})$, that is significantly lower than that of a normal log, thus enabling efficient and reliable anomaly detection.

4.2 Attack Investigation Module

This section presents the design of our attack investigation module, an LLM-based multi-agent framework for recovering attack events from anomalous logs produced by the detection model. We first identify the challenges of using LLMs for attack investigation, then detail ANTEATER's workflow, showing how its components collaborate to overcome these challenges. Finally, we describe the design of the three agents.

Challenges. We detail several technical challenges below.

(C1) ***Unknown Entry Point.*** Unlike in KG-augmented LLM tasks introduced in Sec. 2.4, where the entry point for reasoning can be identified from the question, attack investigation lacks such prior knowledge. In a noisy provenance graph, it is difficult to determine a proper entry point for the LLM to initiate its investigation.

(C2) ***Context Window Bottleneck.*** Although the filter stage reduces log volume, the resulting provenance graph still exceeds the context window limits of current LLMs. We observed that even the one-hop neighborhood of a single high-degree node can exhaust the context capacity. This leads to forgetting of core task instructions, known as failure in long-range dependency modeling, and prevents the LLM from forming a global understanding of the attack scenario.

(C3) ***Fallacies of Local Reasoning and Cascading Errors.*** Constrained by input length, the LLM can only perform local reasoning on provenance subgraphs, lacking a global perspective. This makes it prone to local hallucinations, misinterpreting benign behavior as attacks. In multi-turn reasoning, these false positives are treated as facts for subsequent judgments, leading to cascading errors that severely contaminate the final investigation results.

(C4) ***Catastrophic Forgetting in the Iterative Process.*** In iterative reasoning, while using prior conclusions as context can aid current judgments, the LLM's limited context window can lead to catastrophic forgetting. As the investigation deepens, new information crowds out and overwrites early core clues, such as attack nodes and critical paths. This ultimately causes the model to lose its macro-level memory of the main attack trajectory, hindering its ability to reconstruct a complete and coherent attack chain.

Workflow. The aforementioned challenges show that using an LLM as a black-box classifier for attack investigation is impractical, requiring a new mechanism for interaction and collaboration. Our investigation module addresses this by deeply analyzing temporal and causal correlations in anomalous logs to reconstruct potential attacks. Designed to operate directly on raw log texts, this module is fully decoupled from the anomaly detection phase. This enables it to integrate with any detection tool, including third-party systems, thereby significantly improving the framework's versatility and adaptability.

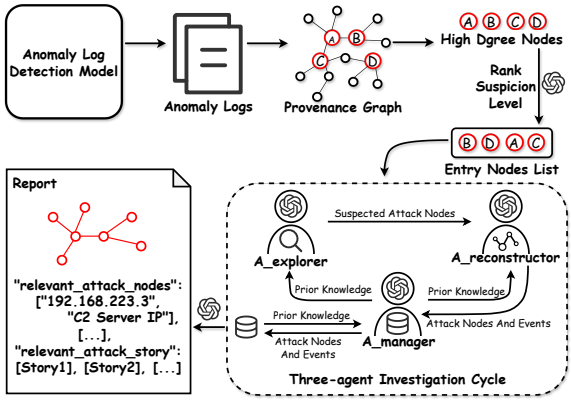


Fig. 3. Workflow of the Attack Investigation Module.

Fig. 3 illustrates the workflow of the attack investigation module, which begins by constructing a provenance graph from the anomalous logs. To address the challenge of entry point selection (C_1), we first identify the M nodes with the highest in- and out-degrees as initial candidates. This strategy is based on two key rationales. First, high-degree nodes occupy critical structural positions and aggregate rich anomalous behavior information. Second, this approach effectively mitigates the cold-start problem arising from an empty investigation knowledge base at the initial stage, thereby providing necessary contextual support for subsequent reasoning. To further optimize efficiency, we abstract the attributes, topological features, and associated edge information of these nodes into preliminary behavioral profiles. We then leverage the LLM's semantic reasoning capability to rank their suspicion levels. The resulting priority list serves as a scheduling guide, directing the system to prioritize high-risk nodes. This approach quickly establishes the backbone of potential attack behaviors and lays the foundation for subsequent depth-first investigation.

Subsequently, for each entry node, we initiate an iterative three-agent investigation cycle. First, A_manager retrieves prior knowledge related to the current node under investigation from the investigation knowledge base it maintains. A_explorer then receives this knowledge, analyzes the node's neighborhood, and uses the LLM to form a preliminary list of suspected behaviors and nodes. This list is then submitted to A_reconstructor for reconstruction and summarization. It determines whether the related behaviors in the list provided by A_explorer and the relevant information given by A_manager can form attack fragments with causal relationships. It also verifies whether the timestamp order of events is logically consistent. Subgraphs that meet these conditions are transformed into structured, time-stamped attack event fragments, and the role each node plays in the attack is inferred. Critically, A_reconstructor also identifies the next key node that is most worthy of in-depth investigation. Finally, the analysis results from A_reconstructor (newly confirmed attack nodes, roles, and story fragments) are sent to the A_manager for decision-making and integration. A_manager again calls the LLM to non-redundantly merge these new findings into the investigation knowledge base. If A_reconstructor proposes a new investigation target, the system recursively repeats this cycle in a depth-first manner on that node, continuing until a preset investigation depth is reached or no new leads are found. The information flow during the cycle is shown in Fig. 4. The attack investigation concludes once all entry nodes have been processed.

Finally, the LLM is invoked again to generate a natural language report based on the attack events stored in the investigation knowledge base. Below, we will elaborate on the three core agents

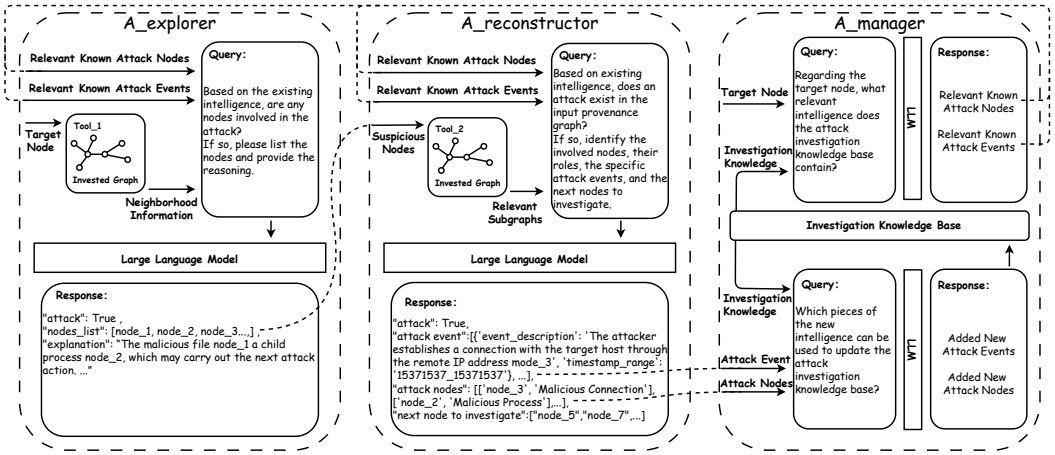


Fig. 4. Information Flow in the Three-Agent Investigation Loop. Specifically, the "attack event" and "attack nodes" within the response from A_reconstructor serve as the fundamental data structures that constitute the investigation knowledge base.

that constitute this investigation module, focusing on how their collaborative design addresses the challenges we have proposed.

Details of A_explorer. A_explorer acts as a tactical-level investigator. To address C_2, we designed it to be responsible only for local exploration and preliminary node labeling. For a node under investigation, Tool_1 is first used to acquire the node's one-hop neighborhood information. Concurrently, to mitigate C_3 and C_4 by incorporating broader context, A_explorer queries A_manager for prior knowledge related to the target node, including known attack nodes and events. This combined information is used to construct a structured prompt that directs the LLM to perform a preliminary security assessment of all nodes within the one-hop neighborhood.

The function of Tool_1 (see Appendix C[1]) is to collect a node's adjacent nodes, their attributes, and the properties of connecting edges (direction, type, timestamp). All edges are sorted chronologically by their start timestamps to provide a clear temporal event flow for the LLM. To address C_2, our tool triggers semantic compression when the number of neighbor nodes is large. Tool_1 examines the one-hop neighborhood edge property distribution of first-order neighbor nodes. Nodes with identical property distributions are merged, thereby compressing the context. To preserve fine-grained detection, Tool_1 maintains a local mapping table during this compression, ensuring that labels from A_explorer can be accurately resolved back to the original nodes.

Details of A_reconstructor. To address C_3 and mitigate false positives caused by LLM hallucinations, we designed a series of explainability tasks to constrain the LLM. For the suspected attack nodes identified by A_explorer, A_reconstructor shifts the process from ambiguous pattern matching to evidence-based reasoning by providing descriptions of attack events and assigning identities to the nodes involved. It then plans the next node to investigate.

A_reconstructor receives the list of suspicious nodes from A_exp-lorer and uses Tool_2 to extract the relevant subgraph from the provenance graph. Concurrently, to mitigate C_4, it incorporates existing clues from A_manager into the data to construct its prompt.

The prompt for A_reconstructor decomposes the task into multiple logically sequential sub-steps, guiding the LLM's reasoning with instructions. First, it determines if the current information

[1]Appendix files can be obtained from our open-source website [6].

constitutes a meaningful attack fragment. If an attack is identified, it directs the LLM to generate a structured, time-stamped narrative designed for easy storage and management by A_manager. Next, it infers the roles of the nodes involved in the attack events (e.g., C2 Server IP, Dropper; see Appendix A for the role list), prompting a deep semantic analysis. Finally, it requires the LLM to propose the next node to investigate. The preceding, progressively detailed tasks help the LLM make a reasonable investigation plan based on the existing attack events.

Tool_2 (see Appendix C), used by A_reconstructor, takes the list of suspicious nodes from A_explorer and extracts the associated provenance subgraph. It retains only the edges whose source and destination nodes are both within the suspicious set. The resulting edges are sorted chronologically and returned to A_reconstructor.

Details of A_manager. A_manager serves as the administrator of the investigation knowledge base, responsible for integrating, maintaining, and retrieving global information. It plays a critical role in C_3 and C_4. At the initial stage of each node's investigation, A_manager retrieves relevant attack nodes and event descriptions from the knowledge base that are associated with the node under investigation. This intelligence serves as prior knowledge for A_explorer and A_reconstructor, thereby mitigating local hallucinations during the LLM's reasoning process. Furthermore, after A_reconstructor completes its analysis of attack events and node roles, A_manager is responsible for updating the knowledge base. In this phase, it compares A_reconstructor's findings with existing information and re-validates whether the new data can further confirm or expand the attack chain, ultimately completing the knowledge base update.

5 Evaluation

In this section, we describe our model implementation, the evaluation datasets, and the baseline models. We introduce the evaluation metrics within their respective experimental sections. In Sec. 5.2, we evaluated the anomaly detection module of ANTEATER. In Sec. 5.3, we demonstrated the prevalence of L3 through quantitative analysis and visualization, and we proved that ANTEATER effectively bridges this gap and addresses L2. In Sec. 5.4 and 5.5, we experimentally demonstrated that the new architecture and techniques in ANTEATER effectively address L1. In Sec. 5.6, by analyzing the attack events in the datasets, we investigated whether the mechanisms designed for ANTEATER's attack investigation module are reasonable and effective in addressing the practical challenges (C_1, C_2, C_3, and C_4). Finally, we conducted robustness and hyper-parameter analyses.

5.1 Experiment Setup

5.1.1 Implementation. ANTEATER is implemented using Python 3.10.3 and CUDA 12.2. The anomaly detection model uses a learning rate of 0.001, with 10 flow blocks and a hidden dimension of 256. The pre-trained BERT model used is bert-base-uncased [22]. The number of entry nodes for the attack investigation module is set to 10. Experiments were conducted on a server equipped with an NVIDIA A100 GPU, 64 CPUs at 2.10GHz, and 251 GB of memory. The method for constructing provenance graphs is consistent with prior research, directly utilizing the open-source code from ATLAS [55]. We did not perform any compression or pruning operations on the provenance graphs. Our focus is on the effectiveness of the method under this general provenance graph structure, in order to broaden the applicability of our approach.

5.1.2 Datasets and Labels. Our experimental evaluation utilized three open-source datasets for network attack detection and three for log anomaly detection. The DEPIMPACT [13] and ATLAS data-sets [55], which provide detailed attack descriptions, were used to evaluate ANTEATER's attack investigation capabilities. The DARPA E3 datasets [12], due to a lack of fine-grained event

labels, were used alongside the HDFS, BGL [77], and Spirit [52] datasets to assess the anomaly detection module.

The DEPIMPACT dataset is a publicly available dataset provided by the DEPIMPACT study. It primarily consists of three sub-datasets: Shellshock, Data Leakage, and VPN Filter. The Shellshock scenario simulated an attack that uses a covert C2 channel for lateral movement and credential theft. The Data Leakage scenario demonstrated a data theft process involving multi-host staging. The VPN Filter scenario simulated the multi-stage deployment of VPNFilter malware to establish a persistent C2 channel.

The ATLAS dataset contains 10 sub-datasets, where S1-S4 simulated attacks on a single host, and M1-M6 simulated cross-host attacks. The dataset included three types of logs: system, DNS, and firefox. Detailed information of the datasets is provided in Sec. 6.1 of the original ATLAS paper [2].

The DARPA Transparent Computing program dataset is one of the most widely used benchmark datasets for provenance-based intrusion detection and attack investigation. We used its three core sub-datasets: CADETS, THEIA, and TRACE. These datasets were captured during professional red-team vs. blue-team exercises and contain real attack scenarios embedded within benign background activities. The THEIA and TRACE datasets were collected from Ubuntu Linux systems, while the CADETS dataset was collected from a FreeBSD system.

The other three log anomaly detection datasets comprise real-world system logs. The BGL and Spirit datasets were collected from supercomputers at Lawrence Livermore and Sandia National Laboratories, respectively, and their logs contain explicit alert labels. The HDFS dataset originated from a Hadoop-based cloud environment, in which log sequences were labeled as normal or abnormal by experts based on their associated block IDs.

When evaluating the anomaly detection module on datasets with only entity-level annotations, we followed AIRTAG's labeling methodology, using the provided entity-level labels to label the log data. All log entries related to malicious entities were marked as malicious, and the rest as benign. For the attack investigation evaluation, we manually labeled the edges of the provenance graphs by combining the attack process descriptions from the dataset papers with the entity-level labels. For the ATLAS dataset, we performed cross-validation by splitting the data into S and M groups[2]. For all other datasets, we used an 80% training data split. Our experimental results and code are available on our anonymous website [6].

5.1.3 Baseline models. We compare our work against seven related works introduced in Sec. 2.3. Three of these are models that take provenance graphs as input—ThreaTrace [69], PROGRAPHER [72], and MAGIC [36]—and four process log texts: NeuralLog [78], RT-Log [35], PreLog [41], and AIRTAG [15]. It should be noted that the graph-based methods can only be evaluated on datasets that provide graph-type input. On the three log anomaly detection datasets, we compare our work only against the log anomaly detection models.

5.2 Effectiveness of Anomaly Detection Module

To evaluate the effectiveness of ANTEATER's anomaly detection module, we conducted experiments on both network attack datasets and generic log anomaly datasets. We evaluated our model using the TPR and FPR metrics from prior work. TPR represents the ratio of malicious entries correctly identified as malicious to the total number of malicious entries. FPR is the ratio of benign entries incorrectly classified as malicious to the total number of benign entries. For the selection of threshold τ, we follow the common practice in unsupervised detection by relying solely on the distributional characteristics of benign data. We compute the log-likelihood scores for all samples in the validation

[2]For example, the model for testing S1 is trained on the benign logs of S2, S3, and S4.

Table 1. Performance comparison of the ANTEATER anomaly detection module and other detection methods on attack detection datasets. We use ANTEATER-A to denote the anomaly detection module of ANTEATER.

Method	Metric	S1	S2	S3	S4	M1	M2	M3	M4	M5	M6	Trace	Theia	Cadets	Shellshock	Data Leakage	VPN Filter
ThreaTrace	TPR	0.673	0.774	0.805	0.693	0.703	0.782	0.883	0.695	0.733	0.698	0.693	0.765	0.865	0.642	0.752	0.782
	FPR	0.305	0.265	0.312	0.221	0.459	0.423	0.377	0.348	0.449	0.411	0.305	0.362	0.225	0.339	0.301	0.241
PROGRAPHER	TPR	0.821	0.804	0.795	0.835	0.809	0.839	0.811	0.805	0.753	0.838	0.909	0.895	0.892	0.855	0.810	0.880
	FPR	0.257	0.300	0.255	0.155	0.234	0.321	0.233	0.322	0.355	0.349	0.253	0.201	0.350	0.260	0.410	0.322
MAGIC	TPR	0.850	0.895	0.875	0.840	0.810	0.869	0.805	0.705	0.802	0.875	0.898	0.915	0.902	0.880	0.769	0.753
	FPR	0.305	0.311	0.330	0.295	0.295	0.350	0.380	0.204	0.344	0.369	0.355	0.243	0.642	0.321	0.399	0.491
NeuralLog	TPR	0.903	0.898	0.922	0.860	0.869	0.899	0.921	0.890	0.877	0.932	0.911	0.881	0.834	0.849	0.871	0.881
	FPR	0.112	0.124	0.151	0.104	0.091	0.210	0.099	0.109	0.111	0.098	0.115	0.134	0.148	0.232	0.105	0.280
AIRTAG	TPR	0.982	0.989	0.980	0.990	0.987	**0.993**	0.982	**0.992**	0.988	0.979	0.903	0.855	0.769	**0.901**	**0.912**	0.876
	FPR	**0.052**	**0.025**	0.072	0.135	0.089	0.101	0.085	**0.079**	**0.054**	0.052	0.128	0.189	0.203	0.201	0.057	0.252
ANTEATER-A	TPR	**0.982**	**0.990**	**0.989**	**0.991**	**0.990**	0.986	**0.992**	0.990	**0.993**	**0.990**	**0.952**	**0.921**	**0.899**	0.900	0.892	**0.917**
	FPR	0.055	0.057	**0.060**	**0.065**	**0.065**	**0.088**	**0.081**	0.088	0.067	**0.042**	**0.089**	**0.101**	**0.097**	**0.098**	**0.049**	**0.124**

Table 2. Performance on the anomaly detection datasets.

Dataset	Metric	NeuralLog	RT-Log	PreLog	ANTEATER-A
HDFS	TPR	**1**	0.982	**1**	0.995
	FPR	0.132	0.085	0.062	**0.053**
BGL	TPR	0.988	0.968	0.982	**0.990**
	FPR	0.019	0.122	**0.010**	0.081
Spirt	TPR	0.991	0.954	**0.996**	0.982
	FPR	0.014	0.017	**0.009**	0.049

Table 3. Performance comparison of the ANTEATER attack investigation module driven by different LLMs.

		S1	S2	S3	S4	M1	M2	M3	M4	M5	M6	Shellshock	Data Leakage	VPN Filter
qwen-max-latest[7]	Recall	0.910	**0.921**	**0.911**	**0.952**	0.910	**0.936**	0.902	0.890	0.906	**0.920**	**0.840**	0.852	0.822
	Precision	**0.979**	**0.990**	**0.982**	**0.977**	**0.985**	**0.972**	**0.989**	**0.952**	**0.990**	**0.990**	0.882	**0.866**	**0.849**
gpt-o1[1]	Recall	0.905	0.906	0.901	0.912	0.897	0.901	**0.922**	**0.903**	**0.923**	0.904	0.810	**0.862**	**0.832**
	Precision	0.942	0.978	0.960	0.945	0.956	0.934	0.909	0.902	0.911	0.918	**0.932**	0.791	0.777
DeepSeek-R1-0528[24]	Recall	0.915	0.895	0.899	0.941	0.921	0.886	0.872	0.860	0.912	0.860	0.708	0.792	0.805
	Precision	0.924	0.911	0.888	0.895	0.905	0.948	0.929	0.887	0.943	0.891	0.782	0.689	0.676

set and treat the lowest 1% of scores as statistical tail outliers. This value is then used as the cutoff threshold for determining anomalies in the test set.

For the evaluation on the attack detection datasets, we compared ANTEATER against the SOTA log text based model AIRTAG, the anomaly detection model NeuralLog, and the SOTA provenance graph based models ThreaTrace, PROGRAPHER, and MAGIC. As shown in Tab. 1, in the log-entry-level evaluation, ANTEATER's performance surpassed that of NeuralLog and all graph-based models, and it outperformed AIRTAG on most sub-datasets. Notably, AIRTAG's superior performance relies on a secondary pre-training phase on the target log data, whereas ANTEATER directly uses a pre-trained BERT model from Hugging Face [22].

An analysis of the evaluation results for the provenance graph methods reveals that although these methods performed exceptionally well in the entity-level evaluations within their respective papers (e.g., MAGIC achieved an entity-level TPR of 99% and an FPR of only 0.13% on the DARPA dataset), they performed poorly in our log-entry-level evaluation. The root cause lies in a mismatch of evaluation granularity. The entities misreported by these methods are often core nodes in the provenance graph with high degrees. According to the labeling method in Sec. 5.1.2 (an edge is considered malicious if either of its nodes is malicious), a single misreported core benign entity causes all its associated edges (i.e., log entries) to be incorrectly labeled, thereby sharply amplifying

the FPR. In contrast, since most neighbors of a malicious entity are also malicious, the TPR is less affected by this issue.

To further validate its generalization ability, we also evaluated ANTEATER on generic anomaly detection datasets against several anomaly log detection models. As shown in Tab. 2, ANTEATER also demonstrated performance comparable to the SOTA model PreLog. However, the PreLog model requires an additional pre-training process, whereas ANTEATER does not. **In summary, the experimental results strongly support our design in Sec. 4.1. The anomaly detection module of ANTEATER, with its extremely low deployment cost and generalization capability without pre-training, perfectly meets the requirements of the "filter-then-scrutinize" architecture for a lightweight and efficient first-stage filter.**

5.3 Effectiveness of Attack Investigation

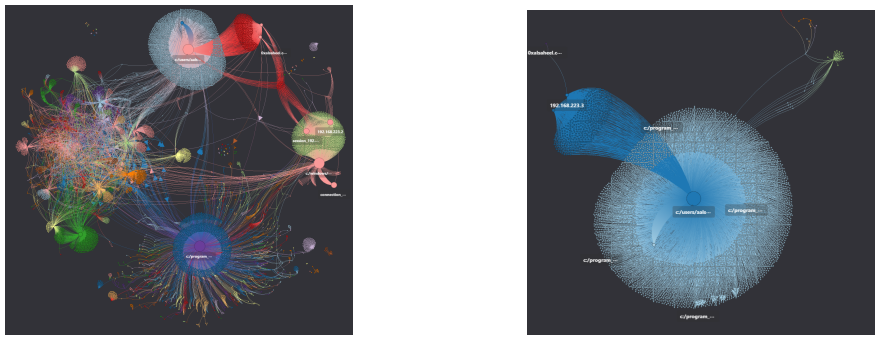
The Necessity of Attack Investigation. Traditional anomaly detection models, even with excellent performance metrics, often yield outputs unsuitable for direct security analysis due to low precision. Using the S4 dataset with 3.5 million logs as an example, despite ANTEATER achieving SOTA TPR and FPR, the extreme class imbalance between normal and attack logs (18,540 entries) results in a massive number of false positives even with a very low FPR. Specifically, the module detected 18,373 true attack logs, which were overwhelmed by 227,546 benign false positives, resulting in an extremely low signal-to-noise ratio. This discrepancy between excellent metrics and unusable results reflects the Limitation L3 we proposed. As shown in Fig. 5 [3], the significant difference in the scale of the provenance graph before and after being processed by the attack investigation module visually illustrates this point. Therefore, deep noise reduction and refinement of anomaly detection outputs are necessary prerequisites for effective attack investigation.

Evaluation Design. Another objective of our evaluation is to validate ANTEATER's ability to distinguish between statistical anomalies and semantic malice (L2). However, the lack of explicit labels for "benign anomalies" in all current public datasets makes direct quantitative evaluation challenging. Nevertheless, our experimental design provides an indirect yet effective evaluation method. As described in Sec. 5.2, the output of the anomaly detection module is a highly mixed dataset, composed of a small fraction of genuinely malicious logs (true positives) and a large volume of benign anomaly logs (false positives). This dataset simulates a challenging real-world scenario where the investigation module must accurately identify and isolate true attack activities from a vast number of events that appear anomalous but harmless. Therefore, the success of the attack investigation module on this task directly reflects its ability to distinguish benign anomalies from malicious ones.

We constructed a provenance graph from the logs output by the anomaly detection module to serve as input for the attack investigation module. We used the graph's edges (i.e., log entries) as the basic unit for evaluation, employing precision and recall as the core metrics. Notably, our recall calculation is end-to-end: any attack log missed during the Filter stage is counted as a false negative in the final evaluation to assess the performance of the entire framework[4]. As shown in Tab. 3, we evaluated the performance of three different LLMs on the attack investigation task. The results show that most LLMs are capable of this task, with the qwen-max-latest model demonstrating more rigorous performance in most cases. Overall, the attack investigation module helps analysts reduce the analysis scale of the provenance graph by an average of 92.8%, while retaining over 89.7% of

[3]This figure illustrates scale differences only, as the large number of nodes limits clarity in the paper. Detailed visualizations and plotting code are available on our website [6].

[4]The attack report consists of text with a non-fixed structure, making its quality difficult to evaluate directly. Nevertheless, given that the report is derived from the output provenance graph by an LLM, its quality can be quantified by the quality of the underlying graph.



(a) Before Investigation (b) After Investigation

Fig. 5. Comparison of Provenance Graph Scale Before and After the Attack Investigation Module.

Table 4. Comparison of Time and Memory Consumption.

Method	Time consumption (s)		Peak Memory consumption (GB)	
	Training	Detection	Training	Detection
MAGIC	528	704	1.7	1.9
AIRTAG	20 hours+	295	40	10
ANTEATER	**139**	**202**	**1.4**	**1.4**

attack-related information. **This result strongly demonstrates that ANTEATER effectively bridges the gap between detection and investigation (L3) and successfully addresses the challenge of distinguishing anomalies from malice (L2).**

5.4 Overhead Analysis

For *L1* in Sec. 1, some prior work chose to delegate the entire task of log detection to an LLM by submitting all log for judgment, a method we believe would face enormous cost and time overhead in practice. In this section, on the ATLAS S4 dataset, we first compare the overhead of ANTEATER's anomaly detection module with SOTA methods. As shown in Tab. 4, our model achieves the lowest time and memory overhead. This efficiency advantage stems from our use of normalizing flows to directly perform density estimation on frozen BERT , avoiding the costs of complex graph construction and pre-training. The low computational load brought by this design enables it to comfortably handle high-throughput logs.

Subsequently, we compare the overhead of ANTEATER's attack investigation module with the following two schemes. Scheme 1: Following prior work, all logs are sent directly to the LLM for processing. Scheme 2: The ANTEATER framework is employed, in which its anomaly detection module first performs preliminary filtering, and the detection results are then sent to the LLM for processing. Scheme 3: Using the framework provided by ANTEATER, the anomaly detection module first performs preliminary detection, then the detection results are used to construct a provenance graph, after which the LLM traverses and judges all nodes using one-hop neighborhoods as the minimum unit.

We use the number of requests to represent time overhead. For both comparison schemes, we truncate the log text using the maximum input length of deepseek-r1. For ANTEATER, its attack investigation module can generate requests without manual intervention. When comparing the cost advantage of ANTEATER with previous LLM-based methods, we selected tokens to evaluate

Table 5. Comparison of the advantages of the ANTEATER architecture and the ANTEATER attack investigation module in terms of token consumption and number of requests.

	Tokens	Requests	Recall	Precision
Scheme 1	193871284	3815	0.630	0.003
Scheme 2	6245494	123	0.968	0.447
Scheme 3	953723	225	0.905	0.677
ANTEATER	**312963**	**96**	**0.941**	**0.895**

the scheme's costs. We did not select the methods mentioned in Sec. 2.4 as baselines due to the following three reasons: 1) these methods rely on explicit query entities as entry points (the entry point is unknown in this task, C_1); 2) they assume that knowledge graph is static and trustworthy (this task contains substantial noise and false positives); 3) they focus on single-path reasoning rather than breadth-first search. Forcibly adapting them would require fundamental modifications to their core logic, causing them to lose their representativeness as baselines.

The results are shown in Tab. 5. Compared with Scheme 1, adopting the "filter-then-scrutinize" architecture (Scheme 2) reduced both cost and time overhead by approximately 31 times. This demonstrates the significant effectiveness of the front-end anomaly detection module in data reduction. Building on this, the ANTEATER multi-agent investigation module further reduces cost and time by approximately 20 and 1.3 times, respectively, compared with the ANTEATER framework without this module. Compared to Scheme 3, which uses the LLM to traverse all nodes on the graph, ANTEATER's attack investigation module achieves 3× cost savings and 2.3× time savings. We finally calculated the proportion of nodes reviewed by ANTEATER during the entire attack investigation process relative to the input provenance graph. 67% of the nodes were examined by ANTEATER, while 33% of the nodes were pruned due to ANTEATER's heuristic depth-first search strategy based on entry nodes, which implements early termination for irrelevant branches determined not to involve attack behaviors, thereby effectively avoiding unnecessary traversal of non-attack regions.

Overall, the complete ANTEATER framework, compared with directly calling the LLM, reduced token overhead and request rounds by more than 600 and 40 times, respectively. **This result not only validates the feasibility of the "filter-then-scrutinize" architecture for handling massive data but also highlights the core value of our multi-agent investigation mechanism in improving task efficiency and reducing costs (*L1*).**

5.5 Decoupling Test

In the overhead analysis, we noted that even without using the full ANTEATER system, adopting our proposed "filter-then-scrutinize" architecture can effectively reduce time and cost. In fact, there are many different logging systems in real-world environments. We designed ANTEATER as a decoupled attack investigation framework, thereby greatly enhancing its applicability. In this section, we will conduct a decoupling test on the ANTEATER framework. Specifically, we selected MAGIC and AIRTAG as anomaly detection models to evaluate ANTEATER's effectiveness. Because different anomaly detection models exhibit varying performance, we can use this to evaluate the impact of the upstream detection model's data quality on ANTEATER's attack investigation model performance.

As shown in Fig. 6, the performance of the ANTEATER framework is influenced by the performance of the upstream model and the distribution of its output results. In comparison, we found that the TPR results are relatively consistent with the anomaly detection model's performance. This is because the upstream model's recall rate for attack-related behaviors directly affects the

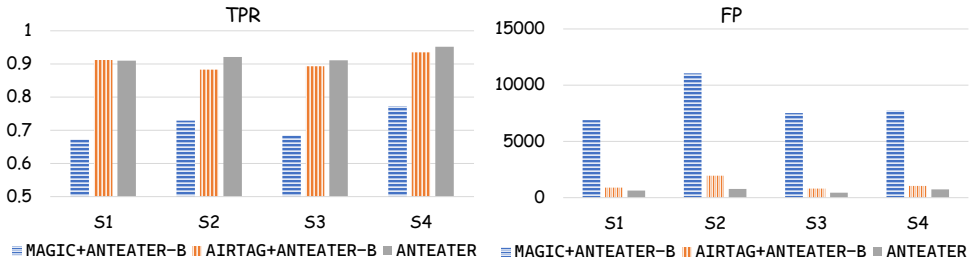


Fig. 6. Impact of Different Anomaly Detection Methods on the Performance of the Attack Investigation Module. Note: We use the absolute count of FP for evaluation instead of the FPR, as varying input data scales across models make FPR an unreliable measure of the final output noise.

completeness of the provenance graph produced by the attack investigation module. However, the approach using MAGIC as the anomaly detection model failed, producing a large number of false positives. In the evaluation in Sec. 5.2, the over-smoothing problem encountered by GNN encoders leads to classification failure for high-degree nodes. ANTEATER's attack investigation module uses high-degree core nodes as entry points for its investigation. While our designed attack investigation module can stop at nodes with no attack behavior when faced with typical upstream model noise and remains largely unaffected, false positives from MAGIC on high-frequency nodes make our attack investigation entry points incorrect, thereby triggering a large number of false positives. **This result not only shows that our model has a certain degree of robustness to upstream model noise but also, together with the overhead analysis in Sec. 5.4, confirms the advantages of our scheme of performing anomaly detection on text and attack investigation on provenance graphs.**

5.6 Case study and mechanism analysis

In this section, we conduct a qualitative analysis using the ATLAS S4 attack scenario. To concretely demonstrate the model's capability to identify complex attack patterns, behavioral chains, and causal dependencies, we perform a fine-grained deconstruction of the reasoning process and its underlying design mechanisms in multi-agent collaboration. We revisit the four challenges (C_1, C_2, C_3, and C_4) proposed in Section 4.2, and use them as benchmarks to retrospectively analyze investigation effectiveness under different configurations through comparative experiments, focusing on analyzing the underlying reasons why introducing specific architectural features yields improvements. We present cases of reasoning interruption and failure that occur in the absence of complete multi-agent collaboration mechanisms, and contrast how the full architecture corrects local fallacies through inter-agent interaction and checks-and-balances, thereby achieving precise reconstruction of the complete attack chain. This process-oriented analysis empirically evaluates the applicability and reliability of LLMs for this specific task, demonstrating how our architecture enables LLMs to produce trustworthy security investigation conclusions through system-level task decoupling and state constraints.

The left side of Fig. 7 shows the attack provenance graph from the S4 dataset, while the right side presents the attack report generated based on ANTEATER's investigation knowledge base. Output directly without any manual processing, this report summarizes the core attack flow. We also display the identity information of the involved nodes, derived from the knowledge base, to demonstrate the extent of ANTEATER's understanding of the attack.

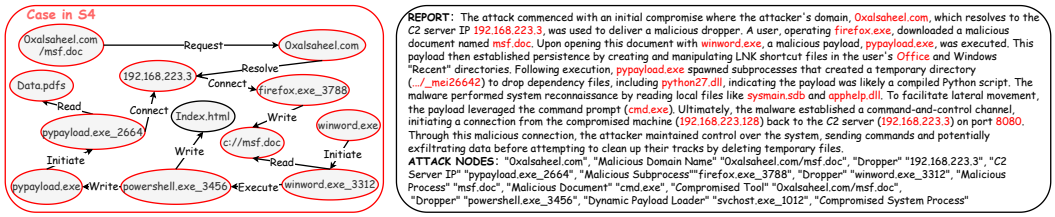


Fig. 7. Provenance graph of the S4 dataset with the attack report and node roles automatically generated by ANTEATER.

To investigate the necessity of different agent collaboration mechanisms in the attack investigation, we establish a baseline using only A_explorer equipped with the entry node selection strategy and Tool_1 (Tool_1 ensures the model's basic operation without errors caused by C_2). Experiments show that using A_explorer alone triggers substantial false positives regarding Windows host and system services (e.g., svchost.exe, spoolsv.exe), which is due to the LLM's limited contextual understanding of the processes within a single query. Second, it fails to detect malicious nodes at the periphery of the attack provenance graph (e.g., 0xalsaheel.com/msf.doc). If a node is not confirmed as malicious in the previous round, topological exploration along that direction terminates. The failure of this single-query mode occurs because combining multiple tasks, such as topological exploration and anomaly discrimination, within a limited context severely impairs the LLM's reasoning capabilities, thereby triggering C_3. This demonstrates the necessity of decoupling topological exploration from complex logical reasoning.

Next, we attempted to introduce a simple memory mechanism for A_explorer, incorporating the attack nodes identified in the previous round as prior knowledge into the prompt for the subsequent round. However, experiments demonstrate that, because of A_explorer's inherently high false-positive rate, introducing a memory mechanism not only failed to reduce false negatives but also caused C_3 to accumulate and amplify, ultimately resulting in investigation performance worse than the baseline. This reveals that, in the absence of accurate local reasoning, the memory mechanism treats local fallacies from the previous round as prior facts. This auto-regressive characteristic causes subsequent reasoning to completely detach from the actual causal relationships of the graph topology, further exacerbating noise propagation.

Therefore, we suspended the memory mechanism affected by C_3 and introduced A_reconstructor to conduct a deep analysis of suspicious subgraphs, requiring it to detail attack behaviors, infer node roles, and select the direction for the next step of the investigation. In the experiment, we compared introducing A_reconstructor with and without requiring it to predict the roles of attack nodes. We found that without this mechanism, A_reconstructor had a limited effect on reducing false positives, whereas incorporating the role prediction mechanism effectively reduced false positives related to system services. This demonstrates that requiring the LLM to describe attack behaviors and infer node roles is essentially forcing the model to perform externalized reasoning. This mechanism breaks the LLM's inertia of relying solely on internal parameters for fuzzy pattern matching, forcing it to seek explicit causal events within the complex provenance graph as logical support. Consequently, it effectively curbs C_3 through graph structure constraints, significantly improving the reliability of local reasoning.

However, we further discovered that relying solely on the collaboration between A_explorer and A_reconstructor still presents issues. If A_explorer cannot determine whether a node is suspicious, the model's false negative rate does not improve. Moreover, since both rely on local views, they tend to direct the investigation toward areas where judgments were ambiguous in the previous round.

Table 6. Robustness evaluation of ANTEATER in different scenarios. In the table, we use ANTEATER+ to denote direct feedback retrieval and ANTEATER+L to denote LLM-guided feedback retrieval.

Variant / Scenario	ANTEATER-A		ANTEATER		ANTEATER+		ANTEATER+L	
	TPR	**FPR**	**Recall**	**Prec.**	**Recall**	**Prec.**	**Recall**	**Prec.**
0.5%	0.701	0.021	0.612	0.897	0.873	0.911	0.853	0.909
1%	0.991	0.065	0.952	0.977	0.980	0.971	0.973	0.981
2%	0.996	0.117	0.952	0.949	0.983	0.939	0.980	0.952
4%	0.996	0.179	0.951	0.945	0.979	0.910	0.975	0.931
100%	0.991	0.065	0.952	0.977	0.980	0.971	0.973	0.981
75%	0.950	0.094	0.903	0.892	0.971	0.899	0.964	0.883
50%	0.857	0.153	0.843	0.867	0.926	0.869	0.919	0.872
25%	0.744	0.301	0.653	0.849	0.920	0.783	0.894	0.774
Dirty Data	0.990	0.067	0.949	0.977	0.984	0.964	0.979	0.973
Miss Top Nodes	—		0.321	0.503	0.898	0.706	0.884	0.905

For instance, although a malicious process is indeed associated with svchost.exe, the lack of global historical context causes subsequent investigations to overlook its background as a controlled process, leading to incorrect judgments about its neighboring nodes. We therefore reintroduced the memory mechanism, using the attack behavior descriptions generated by A_reconstructor as prior knowledge and requiring incremental updates in each round. Analysis revealed that, although the model acquired long-chain reasoning capability, attack descriptions propagated across multiple rounds were influenced by C_4. After more than 10 rounds, the attack narrative began to deviate from the main path, leading to increased false positives. This is because, constrained by C_2, as new reasoning results continuously flood in, the core attack clues accumulated early on are gradually overwritten. This indicates that unidirectional dynamic memory propagation lacks a management and filtering mechanism, making it highly susceptible to C_4 due to drift in core reasoning.

Finally, to address C_4, we introduced A_manager to centrally manage the investigation results. At the beginning of each investigation round, A_manager retrieves intelligence related to the target node from the investigation knowledge base, serving as prior knowledge for A_explorer and A_reconstructor. Furthermore, it summarizes the main attack events based on existing data to prevent the investigation process from deviating. Introducing A_manager essentially achieves the decoupling of state management and logical reasoning at the system architecture level. Acting as an external knowledge base independent of the LLM context window, it organizes fragmented reasoning into structured facts, endowing the main attack path with immutability. This fundamentally overcomes the bottleneck of relying solely on the LLM to maintain long-term memory, successfully addressing the C_3 and C_4 challenges encountered when components are incomplete. Ultimately, the complete ANTEATER model successfully completed the investigation task.

5.7 Robustness Analysis

In this section, we aimed to evaluate the robustness of the ANTEAT-ER framework against three key challenging scenarios: log distribution shift, imperfect training data and targeted detection-evasion attacks. First, for the log distribution shift scenario, we simulate the real-world challenge where validation and test data distributions are inconsistent. In practical applications, obtaining a validation set that accurately reflects the deployment environment is difficult, often resulting in a suboptimal threshold τ. Since ANTEATER's threshold τ is determined by the tail percentile of the validation set distribution, we introduce perturbations to τ by varying the percentile. By observing performance fluctuations under threshold deviations, we evaluate the model's robustness and parameter sensitivity. Furthermore, to simulate imperfect training data, we conducted two sets

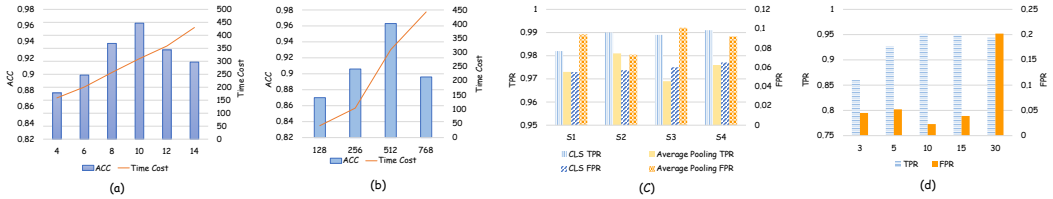


Fig. 8. Effect of Different Hyperparameters on ANTEATER's Performance and Efficiency.

of experiments: 1) a data sparsity test, where we gradually reduced the training data volume to simulate scenarios of insufficient labeling or concept drift; and 2) a data contamination test, where we directly used uncleaned data for training to evaluate the model's tolerance to noise. Finally, to simulate targeted detection evasion attacks, we manually removed the top five attack nodes with the highest in- and out-degrees and their associated edges from the provenance graph before it was input to the attack investigation module [5]. This simulated a scenario in which a skilled attacker successfully evaded detection, resulting in the loss of critical node information.

To address these challenges, we designed and evaluated two feedback enhancement mechanisms. The first, direct feedback retrieval, uses attack entities confirmed by the investigation module as keywords to retrieve matching logs from the original input, which are then fed back to the module for secondary analysis. The second, LLM-guided feedback retrieval, builds on the first approach by introducing an LLM to perform a secondary screening of the confirmed attack entities. The LLM selects the nodes with the highest investigative value as retrieval keywords to improve the precision of the supplementary analysis.

As shown in Table 6, the detection results indicate that, in the log distribution shift experiment, we find that varying the threshold can cause false positives or false negatives in the anomaly detection module. However, increasing the provenance graph size through false positives has minimal impact on ANTEATER's performance. Although false negatives reduce recall, both feedback enhancement mechanisms effectively mitigate this negative effect. In the data sparsity test, the performance of ANTEATER's anomaly detection module gradually declined as the training data were reduced. However, both feedback mechanisms effectively restored the investigation's integrity, stabilizing the recall rate at around 90%. Regarding the data contamination issue, the results showed high robustness: the model trained on noisy data exhibited almost no negative impact on performance, indicating that it is insensitive to a small amount of training noise. In the targeted detection evasion attack experiment, the performance of the basic ANTEATER was significantly degraded when key node information was missing. However, both of our designed feedback enhancement mechanisms showed significant effects: direct feedback retrieval successfully restored the recall to 89.8%, while LLM-guided feedback retrieval substantially improved precision with a minimal sacrifice in recall. **Together, these results validate that ANTEATER, when combined with feedback mechanisms, is a robust and reliable end-to-end attack investigation framework.**

5.8 Hyperparameter Analysis

This section examines the impact of several hyperparameters on ANTEATER's performance. The anomaly detection module comprises three key parameters: the number of flow blocks (K), the hidden-layer embedding dimension (Z), and the BERT encoding method. For the attack investigation module, we analyze the number of entry nodes (M). The K and Z are critical determinants of

[5]Inspired by recent research on evasion attacks against intrusion detection models [23, 51], we simulate a successful evasion attack by removing the attack nodes and their associated information before the investigation begins.

model capacity. Generally, increasing these values enhances the model's expressive power but also prolongs training time. As shown in Fig. 8(a) and 8(b), excessively large values lead to overfitting. Through grid search, we identified their optimal configurations.

We also compared different log text embedding methods. Fig. 8(c) demonstrates that using the [CLS] token outperforms average pooling of all output embeddings. This is because average pooling indiscriminately dilutes the information carried by critical anomaly-indicating words, whereas the [CLS] embedding preserves a more accurate representation in BERT's latent space, enabling our flow-based model to better assess anomaly probabilities.

Fig. 8(d) shows the impact of the M on the attack investigation module. When M is too small, the TPR drops significantly because reconstructing complete attack chains from limited entry nodes demands more consecutive correct judgments from ANTEATER, reducing fault tolerance. Conversely, an oversized M raises the FPR, as lower-ranked entry nodes lack sufficient contextual support for independent judgments—and A_manager sometimes fails to provide effective prior knowledge, leading to increased false alarms.

6 Discussion and Limitations

6.1 Application and Integration of LLMs

Integrating LLMs into high-throughput data management systems faces a fundamental tension between semantic depth and processing efficiency. ANTEATER demonstrates that a hierarchical pipeline architecture effectively addresses this challenge. We argue that LLMs should not function as isolated black boxes, but rather adopt a task-aware data-model coupling strategy: lightweight operators optimized for specific tasks handle high-throughput pruning of massive data, while expensive LLM reasoning is concentrated on semantically dense, controlled data subsets. This design overcomes the scalability bottlenecks of LLM cost and latency at the architectural level, while maximizing data privacy through localized pre-filtering. This architecture provides a practical blueprint for the data management community to handle hierarchical tasks that require both massive unstructured data filtering and deep semantic mining.

6.2 Dependency on Logging System

Similar to previous log based detection methods [15, 36], the effectiveness of this framework highly depends on the integrity and visibility of the logs. If attackers can modify or delete logs, or if their activities fall outside the monitored scope, the system is at risk of failure. Additionally, because the framework relies on logs to infer behavior, it struggles to detect attacks that don't leave unique traces in the logs. Examples include completely in memory Fileless Attacks or "Living-off-the-Land" attacks [11], whose behaviors are indistinguishable from normal administrative operations.

7 Conclusion

This paper presents ANTEATER, an end-to-end attack investigation framework based on raw logs. The framework introduced a multi-agent system to collaboratively perform autonomous exploration, causal reasoning, and attack chain reconstruction on a provenance graph. This design aimed to mitigate the limitations of traditional methods for processing long term massive logs, distinguishing benign anomalies from malicious, and meeting practical operational needs. Evaluation results showed that ANTEATER outperformed existing SOTA methods in reducing false positive rates and generated structured and readable attack reports.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 62472119 and the National Cyber Security-National Science and Technology Major Project under Grant 2025ZD1500204.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[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 *30th USENIX security symposium (USENIX security 21)*. 3005–3022.

[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 amp; Tutorials* (Jan 2019), 1851–1877. doi:10.1109/comst.2019.2891891

[4] 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.

[5] Franciso J. Aparicio-Navarro, Konstantinos G. Kyriakopoulos, Ibrahim Ghafir, Sangarapillai Lambotharan, and Jonathon A. Chambers. 2018. Multi-Stage Attack Detection Using Contextual Information. In *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*. 1–9. doi:10.1109/milcom.2018.8599708

[6] Authors. 2025. The paper website. https://sites.google.com/view/anteatermaterials

[7] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923* (2025).

[8] Zahra Bazrafshan, Hashem Hashemi, Seyed Mehdi Hazrati Fard, and Ali Hamzeh. 2013. A survey on heuristic malware detection techniques. In *The 5th conference on information and knowledge technology*. IEEE, 113–120.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[10] Xiaolei Chen, Jia Chen, Jie Shi, Peng Wang, and Wei Wang. 2025. EPAS: Efficient Online Log Parsing via Asynchronous Scheduling of LLM Queries. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE, 4025–4037.

[11] CrowdStrike. 2024. *Living off the Land (LotL) Attack*. https://www.crowdstrike.com/en-us/cybersecurity-101/cyberattacks/living-off-the-land-attack/ Accessed on: 2024-03-08.

[12] DARPA i2o. 2024. Transparent Computing. https://github.com/darpa-i2o/Transparent-Computing Accessed: 2024-12-03.

[13] DepImpact. 2023. DepImpact Source. https://github.com/depimpact/depimpact/tree/master/depimpact-source-master Accessed: 2025-06-06.

[14] Hailun Ding, Juan Zhai, Dong Deng, and Shiqing Ma. 2023. The case for learned provenance graph storage systems. In *32nd USENIX Security Symposium (USENIX Security 23)*. 3277–3294.

[15] Hailun Ding, Juan Zhai, Yuhong Nan, and Shiqing Ma. 2023. AIRTAG: Towards Automated Attack Investigation by Unsupervised Learning with Log Texts. In *32nd USENIX Security Symposium (USENIX Security 23)*. 373–390.

[16] Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516* (2014).

[17] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016).

[18] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 1285–1298.

[19] Brown Farinholt, Mohammad Rezaeirad, Paul Pearce, Hitesh Dharmdasani, Haikuo Yin, StevensLe Blond, Damon Mccoy, and Kirill Levchenko. [n. d.]. To Catch a Ratter: Monitoring the Behavior of Amateur DarkComet RAT Operators in the Wild. ([n. d.]).

[20] Peng Gao, Xusheng Xiao, Zhichun Li, Fengyuan Xu, Sanjeev R Kulkarni, and Prateek Mittal. 2018. {AIQL}: Enabling efficient attack investigation from system monitoring data. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. 113–126.

[21] Ibrahim Ghafir, Mohammad Hammoudeh, Vaclav Prenosil, Liangxiu Han, Robert Hegarty, Khaled Rabie, and Francisco J. Aparicio-Navarro. 2018. Detection of advanced persistent threat using machine-learning correlation analysis. *Future*

Generation Computer Systems (Dec 2018), 349–359. doi:10.1016/j.future.2018.06.055

[22] Google and Hugging Face. 2018. BERT base uncased. https://huggingface.co/google-bert/bert-base-uncased. Accessed: 2025-06-06.

[23] 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 *30th Network and Distributed System Security Symposium*.

[24] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[25] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. 2020. Unicorn: Runtime provenance-based detector for advanced persistent threats. *arXiv preprint arXiv:2001.01525* (2020).

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

[27] 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 Symposium*.

[28] Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R Lyu. 2021. A survey on automated log analysis for reliability engineering. *ACM computing surveys (CSUR)* 54, 6 (2021), 1–37.

[29] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 594–604.

[30] Peiwei Hu, Ruigang Liang, and Kai Chen. [n. d.]. DeGPT: Optimizing Decompiler Output with LLM. ([n. d.]).

[31] Rikui Huang, Wei Wei, Xiaoye Qu, Wenfeng Xie, Xianling Mao, and Dangyang Chen. 2024. Joint Multi-Facts Reasoning Network For Complex Temporal Question Answering Over Knowledge Graph. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 10331–10335.

[32] Shaohan Huang, Yi Liu, Carol Fung, He Wang, Hailong Yang, and Zhongzhi Luan. [n. d.]. Improving Log-Based Anomaly Detection by Pre-Training Hierarchical Transformers. ([n. d.]).

[33] Muhammad Adil Inam, Yinfang Chen, Akul Goyal, Jason Liu, Jaron Mink, Noor Michael, Sneha Gaur, Adam Bates, and Wajih Ul Hassan. 2023. Sok: History is a vast early warning system: Auditing the provenance of system intrusions. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2620–2638.

[34] Yang Ji, Sangho Lee, Evan Downing, Weiren Wang, Mattia Fazzini, Taesoo Kim, Alessandro Orso, and Wenke Lee. 2017. Rain: Refinable attack investigation with on-demand inter-process information flow tracking. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 377–390.

[35] Peng Jia, Shaofeng Cai, Beng Chin Ooi, Pinghui Wang, and Yiyuan Xiong. 2023. Robust and transferable log-based anomaly detection. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–26.

[36] 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 *33rd USENIX Security Symposium (USENIX Security 24)*. 5197–5214.

[37] Durk P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems* 31 (2018).

[38] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[39] Christopher Kruegel. 2004. Intrusion Detection and Correlation: Challenges and Solutions. (Nov 2004).

[40] Pavlos Lamprakis, Ruggiero Dargenio, David Gugelmann, Vincent Lenders, Markus Happe, and Laurent Vanbever. 2017. Unsupervised detection of APT C&C channels using web request graphs. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 366–387.

[41] Van-Hoang Le and Hongyu Zhang. 2024. Prelog: A pre-trained model for log analytics. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–28.

[42] Zeyan Li, Jie Song, Tieying Zhang, Tao Yang, Xiongjun Ou, Yingjie Ye, Pengfei Duan, Muchen Lin, and Jianjun Chen. 2025. Adaptive and Efficient Log Parsing as a Cloud Service. In *Companion of the 2025 International Conference on Management of Data*. 512–524.

[43] 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 2019 ACM SIGSAC conference on computer and communications security*. 1777–1794.

[44] Yilun Liu, Shimin Tao, Weibin Meng, Feiyu Yao, Xiaofeng Zhao, and Hao Yang. 2024. Logprompt: Prompt engineering towards zero-shot and interpretable log analysis. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*. 364–365.

[45] Yushan Liu, Mu Zhang, Ding Li, Kangkook Jee, Zhichun Li, Zhenyu Wu, Junghwan Rhee, and Prateek Mittal. 2018. Towards a Timely Causality Analysis for Enterprise Security.. In *NDSS*.

[46] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061* (2023).

[47] Lei Ma, Lei Cao, Peter M VanNostrand, Dennis M Hofmann, Yao Su, and Elke A Rundensteiner. 2024. Pluto: Sample Selection for Robust Anomaly Detection on Polluted Log Data. *Proceedings of the ACM on Management of Data* 2, 4 (2024), 1–25.

[48] 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 2019 ACM SIGSAC conference on computer and communications security*. 1795–1812.

[49] 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 *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1137–1152.

[50] Chancharik Mitra, Brandon Huang, Trevor Darrell, and Roei Herzig. 2024. Compositional chain-of-thought prompting for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14420–14431.

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

[52] Adam Oliner and Jon Stearley. 2007. What supercomputers say: A study of five system logs. In *37th annual IEEE/IFIP international conference on dependable systems and networks (DSN'07)*. IEEE, 575–584.

[53] 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 32nd Annual Conference on Computer Security Applications*. doi:10.1145/2991079.2991122

[54] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. 2020. KILT: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252* (2020).

[55] Purseclab. 2023. ATLAS. https://github.com/purseclab/ATLAS. Accessed: 2025-06-06.

[56] Jiaxing Qi, Shaohan Huang, Zhongzhi Luan, Shu Yang, Carol Fung, Hailong Yang, Depei Qian, Jing Shang, Zhiwen Xiao, and Zhihui Wu. 2023. Loggpt: Exploring chatgpt for log-based anomaly detection. In *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 273–280.

[57] Santiago Quintero-Bonilla and Angel Martín del Rey. 2020. A new proposal on the advanced persistent threat: A survey. *Applied Sciences* 10, 11 (2020), 3874.

[58] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[59] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[60] R Sekar, Y Guang, S Verma, and T Shanbhag. 1999. A high-performance network intrusion detection system. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*. 8–17.

[61] Chengyu Song, Linru Ma, Jianming Zheng, Jinzhi Liao, Hongyu Kuang, and Lin Yang. 2024. Audit-llm: Multi-agent collaboration for log-based insider threat detection. *arXiv preprint arXiv:2408.08902* (2024).

[62] Yunfei Su, Mengjun Li, ChaoJing Tang, and Rongjun Shen. 2016. A Framework of APT Detection Based on Dynamic Analysis. In *Proceedings of the 2015 4th National Conference on Electrical, Electronics and Computer Engineering*. doi:10.2991/nceece-15.2016.187

[63] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M Ni, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697* (2023).

[64] Alon Talmor and Jonathan Berant. 2018. The Web as a Knowledge-base for Answering Complex Questions. *arXiv: Computation and Language,arXiv: Computation and Language* (Mar 2018).

[65] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. *arXiv: Computation and Language,arXiv: Computation and Language* (Nov 2018).

[66] SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. 2024. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313* 6 (2024).

[67] Haoqiang Wang, Yiwei Fang, Yichen Liu, Ze Jin, Emma Delph, Xiaojiang Du, Qixu Liu, and Luyi Xing. 2025. Hidden and Lost Control: on Security Design Risks in IoT User-Facing Matter Controller. In *Network and Distributed System Security (NDSS) Symposium 2025*. doi:10.14722/ndss.2025.240266

[68] 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 *NDSS*.

[69] 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.

[70] Xu Wang, Kangfeng Zheng, Xinxin Niu, Bin Wu, and Chunhua Wu. 2016. Detection of command and control in advanced persistent threat based on independent access. In *2016 IEEE International Conference on Communications (ICC)*. doi:10.1109/icc.2016.7511197

[71] Yu Xia, Rui Wang, Xu Liu, Mingyan Li, Tong Yu, Xiang Chen, Julian McAuley, and Shuai Li. 2024. Beyond chain-of-thought: A survey of chain-of-x paradigms for llms. *arXiv preprint arXiv:2404.15676* (2024).

[72] Fan Yang, Jiacen Xu, Chunlin Xiong, Zhou Li, and Kehuan Zhang. 2023. PROGRAPHER: An Anomaly Detection System based on Provenance Graph Embedding. In *32nd USENIX Security Symposium (USENIX Security 23)*. 4355–4372.

[73] Limin Yang, Zhi Chen, Chenkai Wang, Zhenning Zhang, Sushruth Booma, Phuong Cao, Constantin Adam, Alexander Withers, Zbigniew Kalbarczyk, Ravishankar K Iyer, et al. 2024. True attacks, attack attempts, or benign triggers? an empirical measurement of network alerts in a security operations center. In *33rd USENIX Security Symposium (USENIX Security 24)*. 1525–1542.

[74] Yan Zhai, Peng Ning, and Jun Xu. 2006. Integrating IDS alert correlation and OS-level dependency tracking. *Lecture Notes in Computer Science,Lecture Notes in Computer Science* (Jan 2006).

[75] Hang Zhang, Yeyun Gong, Xingwei He, Dayiheng Liu, Daya Guo, Jiancheng Lv, and Jian Guo. 2023. Noisy pair corrector for dense retrieval. *arXiv preprint arXiv:2311.03798* (2023).

[76] G. Zhao, K. Xu, L. Xu, and B. Wu. 2015. Detecting APT Malware Infections Based on Malicious DNS and Traffic Analysis. *IEEE Access* (Jan 2015), 1132–1142. doi:10.1109/access.2015.2458581

[77] Jinyang Zhu, Shilin He, Jinyu Liu, Pinjia He, Qi Zheng, and Michael R Lyu. 2020. Loghub: A large collection of system log datasets towards automated log analytics. *arXiv preprint arXiv:2008.06448* (2020).

[78] Fei Zuo, Xiaopeng Li, Patrick Young, Lannan Luo, Qiang Zeng, and Zhexin Zhang. 2019. Neural Machine Translation Inspired Binary Code Similarity Comparison beyond Function Pairs. In *Proceedings 2019 Network and Distributed System Security Symposium*. doi:10.14722/ndss.2019.23492

Received October 2025; revised January 2026; accepted February 2026