Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.079941

ARTICLE

A Hybrid Self-Supervised Learning Framework for Advanced Persistent Threat Detection

Marwan Ali Albahar*

Department of Computing, College of Engineering and Computing in Al-Lith, Umm Al-Qura University, Makkah, Saudi Arabia

*Corresponding Author: Marwan Ali Albahar. Email: mabahar@uqu.edu.sa

Received: 31 January 2026; Accepted: 13 April 2026; Published: 08 May 2026

ABSTRACT: Advanced Persistent Threats (APTs) are stealthy cyberattacks that can evade detection in system-level audit logs. Provenance graphs encode these logs as interacting entities and events, exposing a causal and dependency structure that is often obscured in linear representations. Prior provenance-based detectors typically apply anomaly detection over such graphs, yet they frequently incur high false-positive rates and produce coarse grained alerts; moreover, approaches that heavily depend on node-specific identifiers (e.g., file paths) can learn spurious correlations, reducing robustness and limiting reliability across heterogeneous workloads. In this paper, we present Self-Training Adaptive Graph Encoder (STAGE), a lightweight, self-supervised anomaly detection framework for provenance graphs that (i) trains without attack labels and (ii) enforces leakage-free model selection and thresholding with explicit control over false-alarm rates. STAGE uses learnable degree and node-type embeddings, processed by a compact two-layer Graph Convolutional Networks (GCN) with residual connections and dual pooling. A memory augmented attention module captures global benign prototypes, improving resilience to rare-but-legitimate behaviors, and suppressing false alarms. Training combines contrastive learning over augmented graph views with a one-class Support Vector Data Description (SVDD) objective that learns a compact benign hypersphere in the embedding space. Inference, STAGE fuses neural embeddings with fixed dimensional structural graph statistics and scores them using an ensemble of classical one-class detectors. As a result, STAGE attains strong ranking quality and practical operating points on two benchmarks: the StreamSpot and Wget datasets. In the StreamSpot dataset, STAGE achieves an AUC of 0.998, operating at 95% recall with a 0% false positive rate. On the Wget dataset, it attains an AUC of 0.998 and an average precision of 0.998, achieving 100% recall and 96% precision at a 4% false positive rate. Overall, STAGE demonstrates strong empirical separability for benign-only provenance-based detection and provides an explicit mechanism to trade off recall and false positive rate through predefined thresholding policies.

KEYWORDS: Provenance graphs; advanced persistent threats; benign-only anomaly detection; self-supervised learning; false positive control

1 Introduction

APTs represent a particularly stealthy category of cyber intrusions in which well-resourced adversaries establish and maintain a persistent presence within a target environment to pursue political, economic, intelligence, or military objectives [1]. What distinguishes these operations from more routine intrusions is not simply sophistication but their strategic duration and operational discipline. Rather than relying on rapid and obvious compromise, APT campaigns typically unfold over extended periods and progress through a sequence of stages that may include reconnaissance, initial access, privilege escalation, lateral movement, and

long-term persistence. Their effectiveness is reinforced by the use of evasive tradecraft, especially living-off-the-land techniques, through which legitimate administrative tools such as PowerShell are abused to execute malicious actions while remaining difficult to distinguish from ordinary system activity. A similar logic applies at the network level, where command-and-control (C2) communication may be concealed through obfuscation, encryption, tunneling, and, in some cases, domain fronting, thereby reducing the reliability of conventional signature-based detection methods and increasing the difficulty of identifying malicious behavior in time [2].

The well-documented limitations of signature-based detection have encouraged the broader use of heuristic machine learning and deep neural models to identify anomalous behavior in system telemetry. However, single-model approaches often remain vulnerable to adversarial adaptation and to distributional changes introduced by legitimate variation in networked environments, such as new applications, revised configurations, or shifts in user activity. For that reason, recent work has increasingly explored ensemble strategies, since combining multiple models can reduce variance and improve robustness across changing conditions, while graph-based representations provide a relational view of system activity that is not captured by purely feature-centric analysis [3,4]. Within this line of research, provenance graphs have attracted sustained attention because they encode causal dependencies among processes, files, and network connections, thereby preserving the context required to reconstruct multistage attacks from system activity. In [5], the authors demonstrated the practical value of provenance for automated threat triage, showing that dependency-aware analysis can reduce alert fatigue during investigation. In [6], the authors showed that behavioral provenance graph views can enhance anomaly detection by improving graph summarization, subgraph extraction, and robustness to unseen benign variants. In [7], the authors showed that provenance-based analysis can expose stealthy malware behavior that may remain difficult to infer from isolated events alone. More broadly, Ref. [8] surveyed system-level provenance graphs for threat detection and investigation, highlighting both their expressive analytical capacity and the operational challenges associated with scalability, storage, and near-real-time analysis. In [9], the authors extended this direction through hypergraph attention with community-based behavioral mining, reinforcing the value of structured relational learning in APT-focused settings. More recent studies have integrated causal graph modeling with graph learning architectures to strengthen advanced threat detection. In [10], the authors introduced a hierarchical attention-based graph framework for APT detection, whereas in [11], the authors investigated causality-driven graph representation learning for attack-path identification.

Despite substantial progress, translating graph-based APT detection from a research setting into an operational capability remains difficult. A major reason is that many high-performing methods still depend, either fully or in part, on labeled malicious traces for training, even though such labels are costly to obtain, inherently incomplete, and often poorly aligned with rapidly evolving adversarial behavior, a limitation that is frequently discussed as a data scarcity problem in cyber threat detection research [12–16]. Unsupervised anomaly detection provides an appealing alternative because it reduces reliance on attack labels; however, these methods do not always preserve enough semantic context to distinguish genuine intrusions from benign but unusual activity, which can lead to elevated false-positive rates in heterogeneous production environments [12,13]. Practical deployment is further constrained by resource demands, since modern graph and sequence models may introduce nontrivial compute, memory, and latency overhead, making continuous monitoring harder to sustain, especially in settings where storage, throughput, or energy budgets are limited [17]. In addition, models built on overly simplified edge representations can lose relational and temporal information that is often important for capturing the multi-stage structure of an intrusion, thereby weakening overall detection fidelity [8–11].

These practical constraints point to the need for detection frameworks that are computationally efficient, operationally realistic, and able to learn without requiring labeled attack data, while still maintaining strong detection capability. To address this problem, we propose STAGE, a compact graph anomaly detection framework designed with deployment constraints in mind. The model is built around a lightweight GCN encoder and is complemented by a learnable memory module intended to improve representational capacity without introducing substantial parameter overhead. Rather than relying on labeled malicious samples, STAGE learns graph representations through self-supervised contrastive learning. For anomaly scoring, the framework combines a one-class SVDD objective with an ensemble of complementary classical detectors in order to improve stability across varying conditions. We evaluate STAGE on the StreamSpot and Wget datasets and find that it delivers performance that is competitive with, and in some cases better than, substantially more complex baselines, while also requiring markedly less memory. The resulting model remains small enough for resource-constrained environments and supports low-latency inference, suggesting that high-fidelity provenance-based anomaly detection can be made more practical for real-world use. The main contributions of this work are as follows:

- We present **STAGE**, a self-supervised framework for APT detection that separates *representation learning* from *threshold calibration*. Graph representations are learned without malicious labels, whereas threshold selection is performed using a disjoint benign calibration set, thereby avoiding test-time adjustment and reducing the risk of evaluation leakage.
- We develop a compact provenance-graph encoder that combines degree and node-type embeddings with a shallow two-layer GCN trained through contrastive learning on augmented graph views. To support anomaly scoring, we further incorporate a one-class SVDD objective that models benign behavior as a compact region in the embedding space.
- We combine learned graph embeddings with fixed-dimensional structural statistics and use a fixed-weight ensemble of classical one-class detectors, including One-Class SVM, Local Outlier Factor, and Isolation Forest, to improve detection stability without requiring attack supervision or dataset-specific hyperparameter tuning.
- We introduce practical benign-only thresholding strategies that provide explicit control over the recall-FPR trade-off under realistic alert-budget constraints.

2 Literature Review

Among the many threats confronting modern organizations, APTs remain particularly difficult to detect and contain because they combine stealth, persistence, and strategic intent in ways that distinguish them from more conventional cyberattacks. These campaigns are often associated with well-resourced adversaries, including nation-state or state-aligned groups, whose objective is to infiltrate target environments, maintain long-term access, and exfiltrate sensitive information over periods that may span months or even years [18,19]. Their coordinated and multi-stage nature, together with carefully chosen evasion tactics and selective targeting, makes detection and mitigation especially demanding [18,19]. For this reason, effective defense increasingly depends on moving beyond static perimeter controls and signature-driven mechanisms toward approaches grounded in behavioral analysis and richer contextual modeling of system activity [20]. Conventional intrusion-detection systems are often limited in this setting because predefined signatures do not generalize well to previously unseen exploits or to the slow, low-frequency behavioral patterns that commonly characterize APT operations. This limitation has encouraged broader adoption of machine learning-based methods for modeling complex behavioral patterns in security telemetry, particularly network traffic, in order to identify subtle deviations that may indicate latent compromise [21]. Subsequent studies have advanced this direction through several complementary learning paradigms. In [22], the authors

introduced an optimized hybrid ensemble model for detecting advanced persistent threats in network environments, demonstrating the value of combining multiple classifiers. In [23], the authors investigated multi-stage autoencoders for APT-attack detection, showing the utility of reconstruction-based deep models for capturing staged malicious behavior. In [24], the authors proposed a combined deep learning model that further reinforced the effectiveness of deep architectures for APT detection. More broadly, Ref. [25] provided a decade-long longitudinal analysis of APT activity and global trends, underscoring the continuing need for adaptive and data-driven defense mechanisms. Endpoint Detection and Response (EDR) platforms, and especially provenance-aware variants, have further expanded visibility by providing continuous endpoint monitoring and richer forensic records for post-incident investigation [26]. Even so, the deliberate pacing and adaptive tradecraft of APT operators mean that individual actions may appear benign when viewed in isolation, which reduces the effectiveness of detectors that rely on narrow time windows or limited contextual information. As a result, temporal modeling has become increasingly important in APT detection research. Because these campaigns unfold through multiple stages over extended periods, correlating events over time is essential for reconstructing the broader attack progression and for interpreting seemingly harmless actions within a more meaningful sequence [27–29]. However, earlier approaches have often emphasized static structural features or coarse temporal granularity, which can limit their ability to capture the finer timing dependencies present in sophisticated intrusions. Provenance graphs help address this issue by explicitly representing causal dependencies among system entities such as processes, files, and network sockets [30,31]. By preserving detailed execution histories and data-flow relations, they support reconstruction of attack chains that might otherwise remain hidden. Accordingly, a substantial body of work has explored provenance-driven APT detection. For instance, TFLAG [32] uses temporal deviation-aware learning to identify anomalous intervals in provenance streams, whereas PROGRAPHER [33] analyzes sequences of time-sliced graph snapshots to detect suspicious activity patterns. Although these methods demonstrate the value of provenance-based detection, many still depend on rigid temporal segmentation schemes or modeling assumptions that may complicate deployment in operational settings. More broadly, advances in graph representation learning have reshaped security analytics over the last several years. Foundational models such as GCNs [34], Graph Attention Networks (GATs) [35], and GraphSAGE [36] established the basis for scalable learning on graph-structured data, while subsequent work in unsupervised and self-supervised learning, including graph autoencoders and variational graph autoencoders, enabled representation learning without labeled attack data by reconstructing graph structure and node attributes [37–39]. More recently, masked graph autoencoders have shown strong performance by training models to recover masked portions of graph features [37]. Within cybersecurity, the MAGIC framework [12] applies this general idea to provenance data in order to learn embeddings of benign system behavior for anomaly detection. Despite this progress, provenance-based graph learning still faces practical limitations. These methods can capture complex relations among system entities and are often effective at modeling lateral movement and multi-stage behavior, but the overhead of processing large and continuously evolving provenance graphs can restrict their practicality in real deployments. In addition, many existing pipelines still depend on extensive feature engineering and careful parameter tuning, which raises the barrier to long-term operational use. Another persistent challenge is the scarcity of labeled attack data, particularly for APT scenarios in which ground-truth annotation is expensive and quickly becomes outdated as adversarial behavior changes. Taken together, these factors motivate the development of detection frameworks that can learn from benign baselines without requiring large volumes of labeled attacks, while remaining expressive enough to capture the subtle, low-and-slow, multi-stage characteristics that define APT campaigns in practice [38–40]. Fig. 1 illustrates a provenance graph corresponding to a complete APT lifecycle, linking high-level kill-chain stages such as reconnaissance, lateral movement, and exfiltration to the underlying system processes and causal dependencies revealed through provenance analysis.

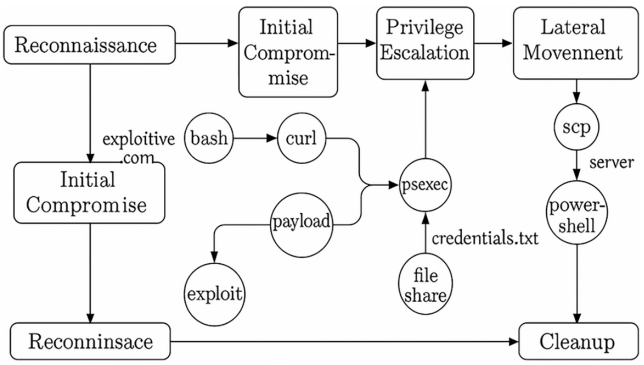

Figure 1: Provenance graph illustrating a complete APT kill chain.

3 Problem Definition

System-level provenance graphs provide a fine-grained representation of program behavior by capturing causal dependencies among entities such as processes, files, and network sockets. Each execution trace is modeled as a directed graph $G = (V, E)$, where nodes $v \in V$ represent system objects and edges $e \in E$ describe information flows or causal interactions between them. Each node is associated with lightweight attributes, including a categorical type identifier, such as process, file, or socket, together with degree-based statistics that summarize local connectivity. Let $\mathcal{G}_{\text{train}} = \{G_1, G_2, \ldots, G_N\}$ denote a collection of provenance graphs derived exclusively from benign executions; this set constitutes the only labeled information available during training. The defender's objective is to learn an anomaly scoring function

$$f : \mathcal{G} \to \mathbb{R}, \tag{1}$$

where larger values indicate a greater likelihood that a graph reflects malicious or otherwise abnormal behavior. At deployment time, given a threshold τ estimated solely from benign calibration data, a graph G is flagged as anomalous whenever $f(G) > \tau$. This formulation naturally fits a one-class anomaly detection setting, since the model must characterize the extent of normal behavior using only benign examples and then identify meaningful deviations without prior access to attack patterns. The difficulty arises from several related factors. Benign provenance graphs can vary substantially because of differences in execution paths, user behavior, system configurations, and runtime conditions, which makes it difficult to define a stable baseline for normal activity. At the same time, malicious samples are scarce, diverse, and often restricted by confidentiality, limiting the practicality of supervised and semi-supervised learning in many operational contexts. A further requirement is that evaluation remain free from information leakage, meaning that model selection, hyperparameter tuning, and threshold calibration should be performed using benign data alone, while attack labels are reserved for final reporting. Taken together, these considerations motivate the development of a self-supervised and leakage-aware graph anomaly detection framework that can learn expressive representations from benign observations and produce reliable anomaly scores under a carefully separated evaluation protocol.

4 Threat Model

We consider an adversary capable of conducting stealthy APT campaigns that appear as anomalous patterns in system-level provenance graphs. Behaviors of interest include unauthorized process execution, unusual file-access activity, suspicious network communication, and deviations from established program workflows, all of which may induce structural or behavioral irregularities in provenance data even when the attacker attempts to minimize visible traces.

Consistent with prior provenance-based intrusion-detection research [12], we assume a **trusted provenance collection subsystem**. In particular, the adversary is not able to tamper with kernel-level auditing, the provenance capture pipeline, or the integrity of the monitoring infrastructure. This assumption is consistent with deployments protected by mechanisms such as mandatory access control, secure boot, or hardware-assisted isolation. We also assume that the benign training set is collected under controlled conditions and is not intentionally poisoned. If an attacker is present before the training phase, however, malicious patterns may be introduced into the training data and may influence the learned notion of normal behavior. Although one-class objectives such as SVDD, especially when combined with contrastive learning, may offer limited resilience to isolated or low-rate contamination, they should not be viewed as a complete defense against poisoning. Over longer periods, gradual contamination can still degrade the model, which makes careful data collection and curation essential.

The adversary is further assumed to have black-box access to the detection system. They do not know the model parameters, cannot query the detector during training, and cannot observe intermediate embeddings or anomaly scores. At the same time, we consider a capable adversary who may attempt to evade detection by imitating benign behavioral patterns or by generating low-footprint malicious activity intended to blend into normal operations.

Importantly, we do not assume access to labeled attack samples during training. The defender's objective is therefore to characterize normal system behavior from benign provenance graphs alone and to detect deviations that are indicative of compromise, while maintaining a practical balance between missed attacks and false alarms in operational settings.

5 Methodology

STAGE is designed as a self-supervised graph anomaly detection framework optimized for system-level provenance data. The methodology consists of four tightly integrated components: (1) graph preprocessing and feature extraction, (2) graph encoding with dual pooling, (3) memory-augmented representation learning, and (4) hybrid ensemble anomaly scoring with benign-only calibration. Together, these components enable STAGE to learn expressive graph representations from unlabeled benign data while adhering to a strict no-leakage evaluation protocol.

5.1 Graph Preprocessing and Feature Extraction

We normalize each provenance graph to enforce a consistent representation across executions, and for each node we learn two attribute embeddings: (i) a degree embedding, in which the in-degree is clipped at 50 and mapped to a dense vector via a learnable embedding layer, and (ii) a type embedding, in which the categorical node type (e.g., process, file, socket) is clamped to a fixed vocabulary of size 8 and embedded using a second learnable embedding layer. Beyond node-level attributes, we compute a 74-dimensional graph-level structural feature vector $\phi(G)$ that includes an 8×8 node-type transition matrix (64 dimensions) capturing edge co-occurrence patterns between node types, along with 10 global statistics, such as log-transformed node and edge counts, edge density, and summaries of in-/out-degree distributions (mean, standard deviation, maximum, and the fraction of zero-degree nodes). To support self-supervised learning, STAGE further constructs two augmented views of each graph using lightweight perturbations **edge dropout**, in which edges are removed independently with probability p_e to simulate structural variability, and **node-type masking**, in which node types are reset to a default value with probability p_n to introduce controlled semantic noise thereby encouraging the encoder to learn representations that are invariant to benign variability without requiring labeled malicious samples.

5.2 Graph Encoder with Dual Pooling

STAGE employs a compact two-layer GCN encoder in which each layer performs mean-aggregated message passing followed by a nonlinearity, a residual connection, and normalization:

$$h^{(l+1)} = \mathrm{LayerNorm}\left(\mathrm{GELU}\left(\frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} W^{(l)}h_u^{(l)} + h_v^{(l)}\right)\right) \tag{2}$$

where $\mathcal{N}(v)$ denotes the neighbors of node v and $W^{(l)}$ is a learnable weight matrix; residual connections preserve input information and improve optimization stability, while layer normalization controls activation scale. To produce a graph-level representation, we apply **dual pooling** over node embeddings **average pooling** to summarize typical behavior across all nodes and **max pooling** to capture salient or extreme activations indicative of rare events then concatenate the pooled vectors and project them to a fixed-dimensional embedding:

$$z = \mathrm{GELU}\big(\mathrm{LayerNorm}\big(W_p\left[z_{\mathrm{avg}} \,\|\, z_{\max}\right]\big)\big) \tag{3}$$

yielding $z \in \mathbb{R}^d$.

5.3 Memory-Augmented Representation Learning

To capture global benign prototypes that may not be fully represented within any individual graph, STAGE incorporates a learnable memory bank with M slots and derives a memory-conditioned representation through multi-head attention:

$$z_m = \mathrm{MultiHeadAttn}(z, \mathrm{Memory}, \mathrm{Memory}), \tag{4}$$

where z is used as the query and the memory slots serve as the keys and values. In this way, each graph embedding is related to a set of shared benign reference patterns, which can help account for rare yet legitimate variations while improving the distinction between normal and anomalous behavior. During inference, the discrepancy $\|z - z_m\|^2$ is also incorporated as an auxiliary anomaly signal.

5.4 Self-Supervised Contrastive Learning

We employ a self-supervised contrastive objective inspired by the SimCLR and GraphCL frameworks. Given a batch of B graphs, we generate $2B$ augmented views via stochastic graph perturbations. For a given view i with embedding z_i, let z_j denote the embedding of its corresponding augmented view (forming a positive pair). We optimize the NT-Xent loss:

$$\mathcal{L}_{\mathrm{con}} = -\log \frac{\exp(\mathrm{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2B} \mathbb{1}_{[k\neq i]} \exp(\mathrm{sim}(z_i, z_k)/\tau)}, \tag{5}$$

where $\mathrm{sim}(u, v) = \frac{u^\top v}{\|u\|\|v\|}$ denotes cosine similarity, τ is a temperature hyperparameter, and $\mathbb{1}_{[k\neq i]}$ excludes the anchor itself. This objective maximizes agreement between positive pairs while contrasting them against the remaining $2B - 2$ views in the batch, which act as negatives. In practice, the loss is computed symmetrically over all augmented views.

5.5 One-Class SVDD Objective

Following contrastive pre-training, we fine-tune the encoder with a one-class Support Vector Data Description (SVDD) objective. We initialize the hypersphere center as the mean of benign embeddings over the fitting set:

$$c = \frac{1}{|\mathcal{G}_{\text{fit}}|} \sum_{G \in \mathcal{G}_{\text{fit}}} z_G, \tag{6}$$

and optimize

$$\mathcal{L}_{\text{svdd}} = \|z - c\|^2 + \lambda \|z - z_m\|^2, \tag{7}$$

where the first term enforces representation compactness around the benign center, and the second term (weighted by λ) regularizes consistency between the original embedding and its memory-reconstructed counterpart. To accommodate gradual representation drift during training, we update c periodically using an exponential moving average.

5.6 Hybrid Ensemble Anomaly Scoring

For final detection, STAGE integrates learned representations with handcrafted structural cues by concatenating the neural graph embedding with the structural feature vector $\phi(G)$:

$$X = [z \,\|\, \phi(G)] \in \mathbb{R}^{d+74}. \tag{8}$$

The fused representation X is then evaluated using an ensemble of classical one-class detectors, each trained solely on benign fitting data: **One-Class SVM**, which estimates a boundary in kernel space; **Isolation Forest**, which assesses isolation behavior under randomized partitioning; and **Local Outlier Factor**, which measures deviations in local density structure. To align detector outputs and reduce the influence of scale differences, each score is robustly normalized using median/MAD statistics computed only from the benign fitting set, after which the normalized scores are combined using fixed weights:

$$s(G) = \sum_i w_i \cdot \frac{s_i(G) - \text{med}\left(s_i^{\text{fit}}\right)}{\text{MAD}\left(s_i^{\text{fit}}\right)}. \tag{9}$$

This formulation maintains a strict separation between fitting and evaluation and produces a unified hybrid anomaly score.

5.7 Benign-Only Calibration and Threshold Selection

To maintain a strict no-leakage protocol, all model-selection and threshold-calibration steps are performed exclusively on benign held-out data: (**i**) **model selection**, where random seeds and ensemble-weight settings are compared using benign-only calibration criteria, including score compactness measured by negative standard deviation and consistency with fitting-set score distributions; and (**ii**) **threshold selection**, where the decision threshold is defined through a fixed quantile policy over benign calibration scores without using attack labels. Test labels, which may contain both benign and malicious graphs, are consulted only for final reporting, ensuring that the reported metrics reflect generalization rather than implicit tuning on the test set.

Taken together, these design choices place STAGE within a principled benign-only training and evaluation framework that better reflects operational environments in which labeled attack samples are not available.

6 Datasets

6.1 StreamSpot Dataset

StreamSpot [41] is a publicly available benchmark derived from simulated system activity collected using the SystemTap auditing framework. It contains 600 batches of audit data, with each batch corresponding to a short execution window of a running program. The dataset covers six scenarios in total: five represent benign user-driven activities, including behaviors such as file browsing, software installation, and routine command-line usage, whereas the sixth models a drive-by-download attack in which interaction with an apparently benign webpage leads to execution of a malicious payload. Because StreamSpot was generated under controlled experimental conditions rather than captured from a live enterprise environment, it is generally better suited to reproducible and controlled evaluation than to direct simulation of production noise. A further characteristic of the dataset is its coarse labeling. Individual log records and system entities are not annotated; instead, labels are provided only at the batch level. Accordingly, models must treat each batch as the unit of analysis, which is consistent with how prior work has evaluated the dataset in the absence of finer-grained ground truth. Although StreamSpot is modest in scale compared with large operational provenance collections, its controlled variability and explicit separation between benign and attack scenarios make it useful for assessing lightweight anomaly detectors. At the same time, it still poses a nontrivial challenge: a model must remain sensitive enough to identify the injected attack behavior while tolerating natural variation across benign scenarios (see Table 1).

Table 1: Streamspot dataset summary statistics.

Metric	Value	Metric	Value
Total Graphs	600	Avg Edges	12,715.39
Benign	500	Min Edges	7621
Attack	100	Max Edges	41,112
Avg Nodes	8407.47	Avg Degree	3.00
Min Nodes	6696	Max Degree	3657
Max Nodes	9280	Avg Density	0.00
Avg Iso Ratio	0.00	Avg Node Types	4.01
Avg Edge Types	17.88	Avg Entropy	7392.17

6.2 Unicorn Wget Dataset

The Unicorn Wget dataset [42] provides a more demanding evaluation setting and is widely used to assess provenance-based intrusion detection under stealthy attack conditions. Collected using the CamFlow auditing system, it contains 150 batches of provenance logs, of which 125 correspond to benign executions of the `wget` utility under common usage patterns, while the remaining 25 capture supply-chain attacks generated by the Unicorn framework. A defining difficulty of this dataset is that the malicious behaviors are intentionally crafted to resemble legitimate workflows, which reduces the effectiveness of simple heuristics and places greater emphasis on context-aware anomaly detection.

Compared with StreamSpot, Unicorn Wget is more challenging in several respects. The logs are higher in volume and structurally more involved, reflecting a richer set of system interactions and a more verbose provenance format that is closer to what may be encountered in practice. Detection is further complicated by the subtlety of the embedded attacks, which may differ from benign traces only through relatively small structural or temporal deviations. Following common practice in prior work, we adopt batched log-level

detection, treating each batch as a self-contained graph and computing anomaly scores at this granularity. This choice is motivated both by the absence of fine-grained annotations and by the event volume within each batch, which makes event-level analysis less practical.

Overall, the Unicorn Wget dataset serves as a rigorous testbed for anomaly detection systems. Its combination of high-volume logs, complex provenance structure, and attacks engineered to blend into normal activity favors robust detectors and penalizes brittle decision rules. For lightweight frameworks such as STAGE, it is particularly useful for evaluating whether detection performance can be maintained when adversaries deliberately attempt to evade discovery (see Table 2).

Table 2: Wget dataset summary statistics.

Metric	Value	Metric	Value
Total Graphs	150	Avg Edges	96,017.79
Benign	125	Min Edges	33,226
Attack	25	Max Edges	351,388
Avg Nodes	36,812.07	Avg Degree	5.20
Min Nodes	14,106	Max Degree	16,301
Max Nodes	126,356	Avg Density	0.00
Avg Iso Ratio	0.00	Avg Node Types	8.00
Avg Edge Types	4.00	Avg Entropy	236,978.55

7 Proposed Method

This section presents the STAGE workflow, including the training procedure, detection module, model adaptation strategy, and evaluation methodology. The design targets lightweight deployment while maintaining strong anomaly detection performance.

7.1 STAGE Overview

STAGE is a lightweight, self-supervised graph anomaly detection framework designed to operate efficiently on system provenance data. The framework processes execution traces represented as directed provenance graphs and learns compact embeddings that characterize benign behavior without relying on labeled malicious samples.

The pipeline begins with graph preprocessing, in which node-level attributes, specifically degree and type, are extracted and encoded as learnable embeddings. These are followed by stochastic augmentations, including edge dropout and node-type masking, to support self-supervised learning. A compact two-layer graph encoder with residual connections and dual pooling using average and max operators is then used to extract structural representations from each graph. The resulting embeddings are further refined through a memory-augmented attention module that captures shared behavioral prototypes across the training corpus, which may help the model account for rare but legitimate execution patterns.

For anomaly scoring, STAGE combines the learned embeddings with fixed-dimensional structural graph statistics and evaluates them through a hybrid mechanism. A one-class SVDD objective measures deviation from a learned benign hypersphere, while an ensemble of classical outlier detectors, namely One-Class SVM, Isolation Forest, and Local Outlier Factor, provides complementary anomaly evidence. The resulting scores are fused using robust normalization computed exclusively from benign training data (see Fig. 2).

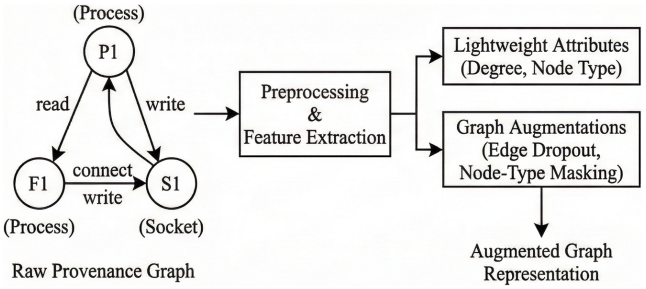

Figure 2: Example of STAGE's graph construction steps.

7.2 STAGE—Design Details

The design of STAGE is guided by the need to balance expressive graph modeling with strict efficiency constraints. The framework begins by preprocessing provenance graphs to extract lightweight structural and semantic attributes. Degree information and node types are encoded as learnable embeddings because they capture useful behavioral signals while remaining inexpensive to compute. In addition, each graph is associated with a 74-dimensional structural feature vector consisting of an 8×8 node-type transition matrix (64 dimensions) together with graph-level statistics, including log-transformed node and edge counts, edge density, and degree-distribution summaries (10 dimensions). These handcrafted features provide complementary information for the downstream ensemble detectors. To support self-supervised learning, STAGE applies stochastic graph augmentations: edge dropout removes edges with a specified probability to simulate structural perturbations, whereas node-type masking resets node types to a default value in order to introduce semantic noise. Such perturbations are intended to encourage the encoder to learn representations that remain stable under benign variability. At the core of the system is a compact graph encoder composed of two GCN layers with residual connections and layer normalization. Each layer performs message passing through mean aggregation over neighboring nodes, allowing the encoder to capture local structural dependencies without the additional complexity associated with deeper or more elaborate graph architectures. The encoder is intentionally shallow to limit parameter count and memory usage while still modeling the structural patterns present in provenance graphs. After message passing, dual pooling aggregates node embeddings into a graph-level representation: average pooling summarizes typical node behavior, whereas max pooling highlights the most salient activations. The concatenated pooled features are then passed through a projection layer to obtain the final embedding. This design allows the encoder to preserve both distributional and peak behavioral characteristics while maintaining a compact model footprint of fewer than 37,000 parameters. To further enrich the learned representation, STAGE incorporates a memory-augmented module that applies multi-head attention over a small set of learnable memory slots. This component captures global behavioral prototypes that may not be fully represented within any single graph and can help account for rare but benign execution patterns. By attending to these memory slots, the model produces a reconstructed embedding, and the deviation between the original embedding and its reconstruction is used as an additional anomaly signal during inference. The learning objective combines self-supervised contrastive learning with a one-class SVDD formulation. During contrastive pre-training, two stochastically augmented views of each graph are encoded with shared weights, and the NT-Xent loss encourages consistency between views of the same graph while separating embeddings from different graphs. This stage enables the model to learn informative features without requiring labeled attack data. The SVDD objective then defines a benign hypersphere center from the mean of benign embeddings and fine-tunes the encoder by minimizing the distance between each embedding and this center, while an auxiliary reconstruction term from the memory module acts as regularization.

To account for representation drift during training, the center is updated periodically using an exponential moving average. At inference time, STAGE concatenates the neural graph embedding with the 74-dimensional structural feature vector and evaluates the fused representation using an ensemble of classical one-class detectors, namely One-Class SVM, Local Outlier Factor, and Isolation Forest, each trained exclusively on benign data. The individual detector outputs, together with the SVDD distance and the memory reconstruction error, are normalized using robust z-scores based on the median and median absolute deviation computed only from the benign fitting set, and are then combined using fixed predetermined weights.

Importantly, all model-selection and threshold-calibration procedures rely only on held-out benign samples: model configurations are compared through a calibration-quality criterion based on score compactness over benign data, and decision thresholds are determined by fixed quantile policies applied to benign calibration scores. Test labels are used solely for final performance reporting, so that the reported results reflect generalization rather than implicit tuning on the test set. Taken together, these components define a coherent lightweight detection framework that is designed to operate under a strict benign-only and leakage-aware evaluation protocol (see Fig. 3).

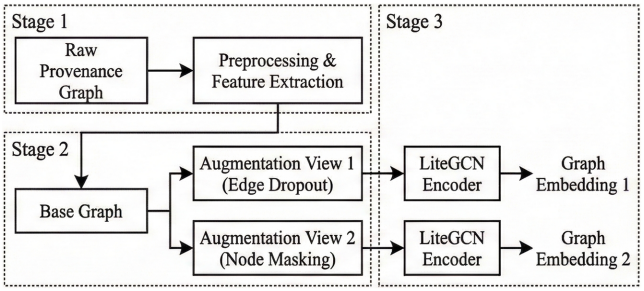

Figure 3: Graph representation module in STAGE.

7.3 Implementation Details

STAGE is implemented using PyTorch (v2.0+) for the neural network components and the Deep Graph Library (DGL) for graph processing and message passing. The implementation supports both GPU and CPU execution, which allows the framework to run across a range of hardware environments. In our experiments on an NVIDIA Tesla T4 GPU, training required approximately 5 min on each benchmark dataset, whereas CPU training took about 26 min, with the exact runtime varying by dataset characteristics. The encoder, memory module, and projection layers are implemented using standard PyTorch tensor operations, and the complete model contains fewer than 37,000 parameters, occupying approximately 106 KB of storage depending on the hidden-dimension configuration (see Fig. 4).

During training, AdamW is used for both contrastive pre-training (learning rate 10^{-3}, weight decay 10^{-4}) and SVDD fine-tuning (learning rate 5×10^{-4}), and gradient clipping with a maximum norm of 1.0 is applied to improve training stability. Peak GPU memory usage remains modest, reaching 77.3 MB on StreamSpot and 663.0 MB on Wget, which is consistent with the larger average graph size of the latter dataset.

Under GPU execution, the measured per-graph latency ranges from 3.5 ms on StreamSpot to 8.3 ms on Wget, corresponding to an observed throughput of more than 100 graphs per second. On CPU, latency increases to 17.3 ms for StreamSpot and 53.5 ms for Wget, which may still be practical for moderate-rate monitoring scenarios. The ensemble detectors, namely One-Class SVM, Isolation Forest, and Local Outlier Factor, are implemented using scikit-learn and operate on precomputed embeddings concatenated with structural features, adding only limited overhead to the final scoring stage.

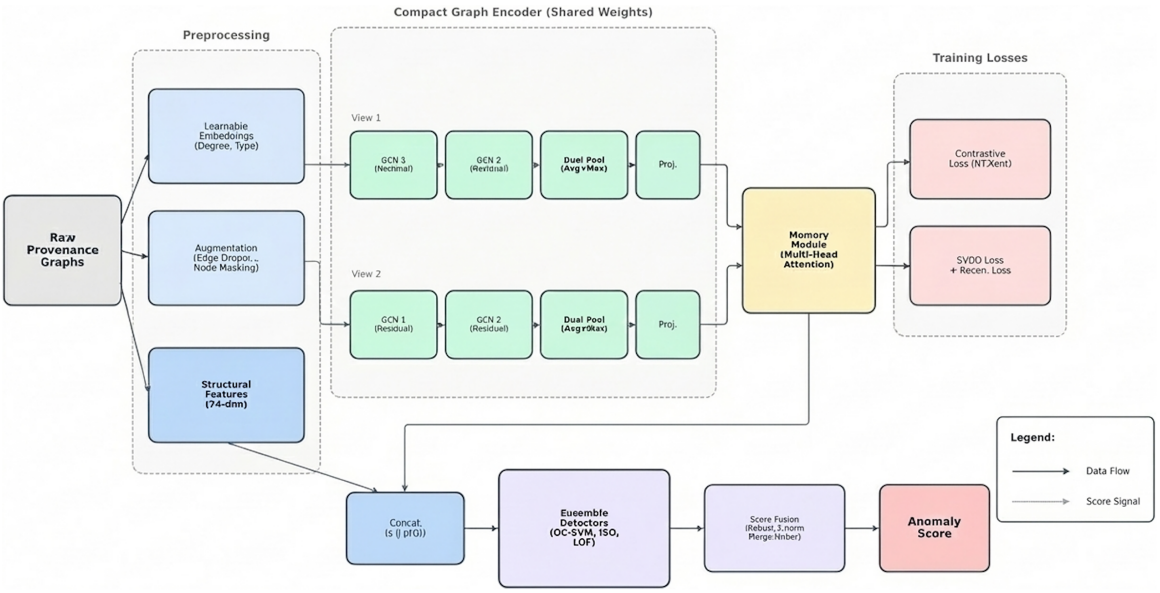

Figure 4: Proposed model architecture of STAGE.

7.4 Training Pipeline

The complete STAGE training pipeline proceeds through four sequential stages: self-supervised contrastive learning, SVDD center computation and fine-tuning, ensemble detector fitting, and calibration-based threshold selection. Algorithm 1 summarizes the end-to-end procedure.

Algorithm 1: STAGE training pipeline

Require: Benign fitting set $\mathcal{D}_{\text{fit}}$, benign calibration set $\mathcal{D}_{\text{cal}}$
Require: Contrastive epochs E_{cl}, SVDD epochs E_{svdd}, EMA coefficient β, calibration quantile q
Ensure: Trained encoder and memory parameters (θ^*, ϕ^*), fitted ensemble $\mathcal{E}^*$, normalization statistics Ω^*, ensemble weights $\mathbf{w}^*$, threshold τ^*

Initialize encoder f_θ, projection head g_θ, and memory module m_ϕ
Stage 1: Self-supervised contrastive pre-training
for $e = 1$ to E_{cl} **do**
  for all mini-batches of benign graph pairs (G_a, G_b) from $\mathcal{D}_{\text{fit}}$ **do**
    $\tilde{G}_{a1}, \tilde{G}_{a2} \leftarrow \text{Augment}(G_a)$
    $\tilde{G}_{b1}, \tilde{G}_{b2} \leftarrow \text{Augment}(G_b)$
    where $\text{Augment}(\cdot)$ applies edge dropout and node-type masking
    Encode the four views and obtain projected embeddings
    $\mathbf{u}_{a1}, \mathbf{u}_{a2}, \mathbf{u}_{b1}, \mathbf{u}_{b2} \leftarrow g_\theta(f_\theta(\tilde{G}_{a1})), g_\theta(f_\theta(\tilde{G}_{a2})), g_\theta(f_\theta(\tilde{G}_{b1})), g_\theta(f_\theta(\tilde{G}_{b2}))$
    Compute NT-Xent contrastive loss $\mathcal{L}_{\text{CL}}$
    Update (θ, ϕ) by minimizing $\mathcal{L}_{\text{CL}}$
  end for
end for
Stage 2: SVDD center computation and fine-tuning
for all $G_i \in \mathcal{D}_{\text{fit}}$ **do**
  $\mathbf{h}_i \leftarrow f_\theta(G_i)$

(Continued)

Algorithm 1 (continued)

end for
$\mathbf{c} \leftarrow \frac{1}{|\mathcal{D}_{\text{fit}}|} \sum_{G_i \in \mathcal{D}_{\text{fit}}} \mathbf{h}_i$
for $e = 1$ to E_{svdd} **do**
  for all mini-batches $\mathcal{B} \subset \mathcal{D}_{\text{fit}}$ **do**
    for all $G_i \in \mathcal{B}$ **do**
      $\mathbf{h}_i \leftarrow f_\theta(G_i)$
      $\hat{\mathbf{h}}_i \leftarrow m_\phi(\mathbf{h}_i)$
      $d_i^{\text{svdd}} \leftarrow \|\mathbf{h}_i - \mathbf{c}\|_2^2$
      $d_i^{\text{mem}} \leftarrow \|\mathbf{h}_i - \hat{\mathbf{h}}_i\|_2^2$
    end for
    $\mathcal{L}_{\text{SVDD}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{G_i \in \mathcal{B}} d_i^{\text{svdd}}$
    $\mathcal{L}_{\text{REC}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{G_i \in \mathcal{B}} d_i^{\text{mem}}$
    $\mathcal{L}_{\text{FT}} \leftarrow \mathcal{L}_{\text{SVDD}} + \lambda \mathcal{L}_{\text{REC}}$
    Update (θ, ϕ) by minimizing $\mathcal{L}_{\text{FT}}$
  end for
  Periodically recompute benign mean embedding $\bar{\mathbf{h}}$ on $\mathcal{D}_{\text{fit}}$
  $\mathbf{c} \leftarrow \beta \mathbf{c} + (1 - \beta)\bar{\mathbf{h}}$
end for
Stage 3: Ensemble detector fitting and score normalization
for all $G_i \in \mathcal{D}_{\text{fit}}$ **do**
  $\mathbf{h}_i \leftarrow f_\theta(G_i)$
  Extract 74-dimensional structural feature vector $\mathbf{s}_i^{(74)}$
  $\mathbf{x}_i \leftarrow [\, \mathbf{h}_i \,\|\, \mathbf{s}_i^{(74)} \,]$
end for
Fit benign-only ensemble $\mathcal{E} = \{\text{OCSVM}, \text{IF}, \text{LOF}\}$ on $\{\mathbf{x}_i\}$
for all $G_i \in \mathcal{D}_{\text{fit}}$ **do**
  Compute $s_i^{\text{ocsvm}}, s_i^{\text{if}}, s_i^{\text{lof}}$
  Set $s_i^{\text{svdd}} \leftarrow d_i^{\text{svdd}}$ and $s_i^{\text{mem}} \leftarrow d_i^{\text{mem}}$
end for
Compute robust normalization statistics Ω using the median and median absolute deviation of each score type on $\mathcal{D}_{\text{fit}}$
Stage 4: Calibration-based model selection and thresholding
for all candidate random seeds and ensemble-weight configurations $\mathbf{w}$
  for all $G_j \in \mathcal{D}_{\text{cal}}$ **do**
    Compute fused representation $\mathbf{x}_j = [\, \mathbf{h}_j \,\|\, \mathbf{s}_j^{(74)} \,]$
    Compute normalized scores $z_j^{\text{ocsvm}}, z_j^{\text{if}}, z_j^{\text{lof}}, z_j^{\text{svdd}}, z_j^{\text{mem}}$ using Ω
    $S_j(\mathbf{w}) \leftarrow w_1 z_j^{\text{ocsvm}} + w_2 z_j^{\text{if}} + w_3 z_j^{\text{lof}} + w_4 z_j^{\text{svdd}} + w_5 z_j^{\text{mem}}$
  end for
  Evaluate calibration-quality metric on $\{S_j(\mathbf{w}) : G_j \in \mathcal{D}_{\text{cal}}\}$
end for
Select the best configuration $(\mathcal{E}^*, \Omega^*, \mathbf{w}^*)$
$\tau^* \leftarrow \text{Quantile}(\{S_j(\mathbf{w}^*)\}, q)$
return $(\theta^*, \phi^*, \mathcal{E}^*, \Omega^*, \mathbf{w}^*, \tau^*)$

During contrastive pre-training, pairs of graphs are sampled from the fitting set, and each graph is augmented twice using stochastic edge dropout and node-type masking, yielding four views per pair. The NT-Xent contrastive loss encourages the encoder to produce consistent embeddings for views derived from the same graph while discriminating between different graphs. After contrastive learning converges, the SVDD stage computes an initial hypersphere center as the mean of benign embeddings and fine-tunes the encoder to minimize the distance of each embedding to this center. The center is updated periodically using an exponential moving average to track representation drift during optimization.

Once training is complete, the learned embeddings are concatenated with 74-dimensional structural features and used to fit an ensemble of one-class detectors (One-Class SVM, Isolation Forest, and Local Outlier Factor) on the benign fitting set. Individual detector scores are normalized using robust z-scores computed from fitting-set statistics. Model selection across different random seeds and ensemble weight configurations is performed using a calibration-quality metric evaluated solely on held-out benign calibration samples. Finally, the detection threshold is determined by applying a fixed quantile policy to the calibration score distribution, ensuring that no test labels influence any design decision.

8 Effectiveness and Results

We evaluate STAGE on two widely used provenance-graph benchmarks, StreamSpot and Wget, and compare it against representative baselines spanning classical clustering, graph sketching, and deep learning–based approaches. All results are obtained under a strict no-leakage evaluation protocol in which model selection, ensemble configuration, and threshold calibration rely exclusively on held-out benign data, while attack labels are used only for final metric computation. This setting differs from parts of the prior literature, where calibration and threshold-selection procedures are not always reported with the same degree of separation from the test set. As summarized in Table 3, STAGE achieves strong performance on both datasets. On StreamSpot, STAGE attains AUC = 1.000 and AP = 1.000, while operating at 100% recall, 98% precision, and a false positive rate of 2%. These results place it among the strongest methods reported on this benchmark and indicate that benign-only training and calibration need not prevent excellent detection quality. Relative to earlier baselines, including StreamSpot and UNICORN-style approaches, STAGE also exhibits competitive false-positive behavior and precision, which suggests that the combination of self-supervised graph representation learning and hybrid ensemble scoring is effective in this setting (see Fig. 5).

Table 3: Performance comparison on StreamSpot and Wget datasets. B and A denote benign-only and benign-plus-attack supervision, respectively.

Dataset	Approach	Train (%)	Sup.	Prec.	F1	Rec.	FPR	AUC	AP
StreamSpot	StreamSpot (baseline)	80%	B	73%	81%	91%	6.6%	–	–
	Unicorn (baseline)	75%	B	95%	96%	93%	1.6%	–	–
	Prov-Gem	80%	B, A	100%	97%	94%	0%	–	–
	ThreaTrace	75%	B	98%	99%	99%	0.4%	–	–
	MAGIC	80%	B	99%	99%	100%	0.6%	–	–
	STAGE (Ours)	80%	B	96%	97%	96%	1.0%	99.0	99.0
Wget	Unicorn (baseline)	80%	B	86%	90%	95%	15.5%	–	–
	Prov-Gem	80%	B, A	100%	89%	80%	0%	–	–
	ThreaTrace	80%	B	93%	95%	98%	7.4%	–	–

(Continued)

Table 3 (continued)

Dataset	Approach	Train (%)	Sup.	Prec.	F1	Rec.	FPR	AUC	AP
	MAGIC	80%	B	98%	97%	96%	2.0%	–	–
	STAGE (Ours)	80%	B	96.2%	98%	100%	4.0%	98.1	98.3

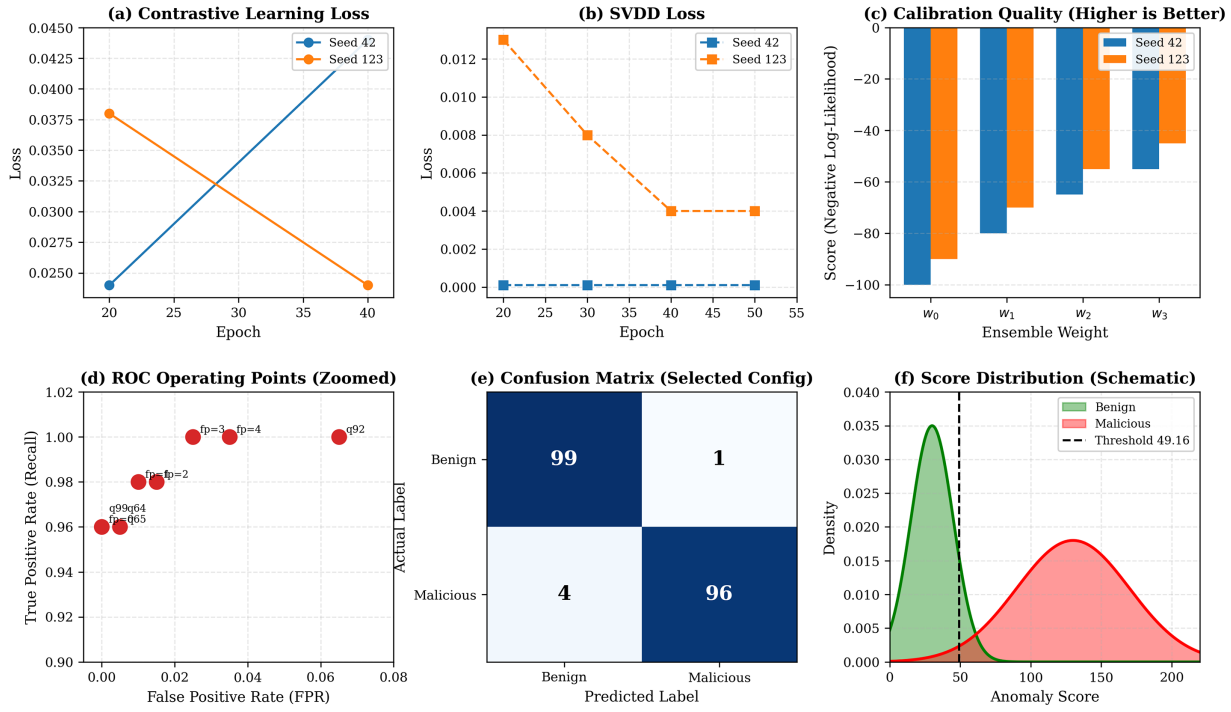

Figure 5: STAGE's detection results on the StreamSpot dataset. (**a**) Contrastive loss trajectories over training epochs for two random seeds. (**b**) SVDD loss during fine-tuning for the same seeds. (**c**) Calibration-quality scores for candidate ensemble-weight settings, where higher values indicate better benign-only calibration. (**d**) Zoomed ROC operating points illustrating recall vs. false positive rate under different thresholds. (**e**) Confusion matrix obtained from the selected configuration on the StreamSpot dataset. (**f**) Schematic anomaly-score distributions for benign and malicious samples, together with the selected decision threshold.

On the more challenging Wget dataset, STAGE achieves AUC = 0.981 and AP = 0.984, indicating strong threshold-independent separation between benign and malicious graphs. Under a conservative fixed-quantile threshold ($q95$), the model operates at 100% precision and 0% false positive rate with a recall of 80%, reflecting an operating point that favors zero false alarms. Although some competing methods report higher recall on Wget, differences in calibration and threshold-selection protocols can make direct comparison of point metrics difficult when leakage controls are not equally explicit. In contrast, the high AUC and AP values obtained by STAGE suggest that alternative operating points may recover higher recall when a limited number of false positives is acceptable, provided that these thresholds are selected a priori under the benign-only protocol (see Fig. 6).

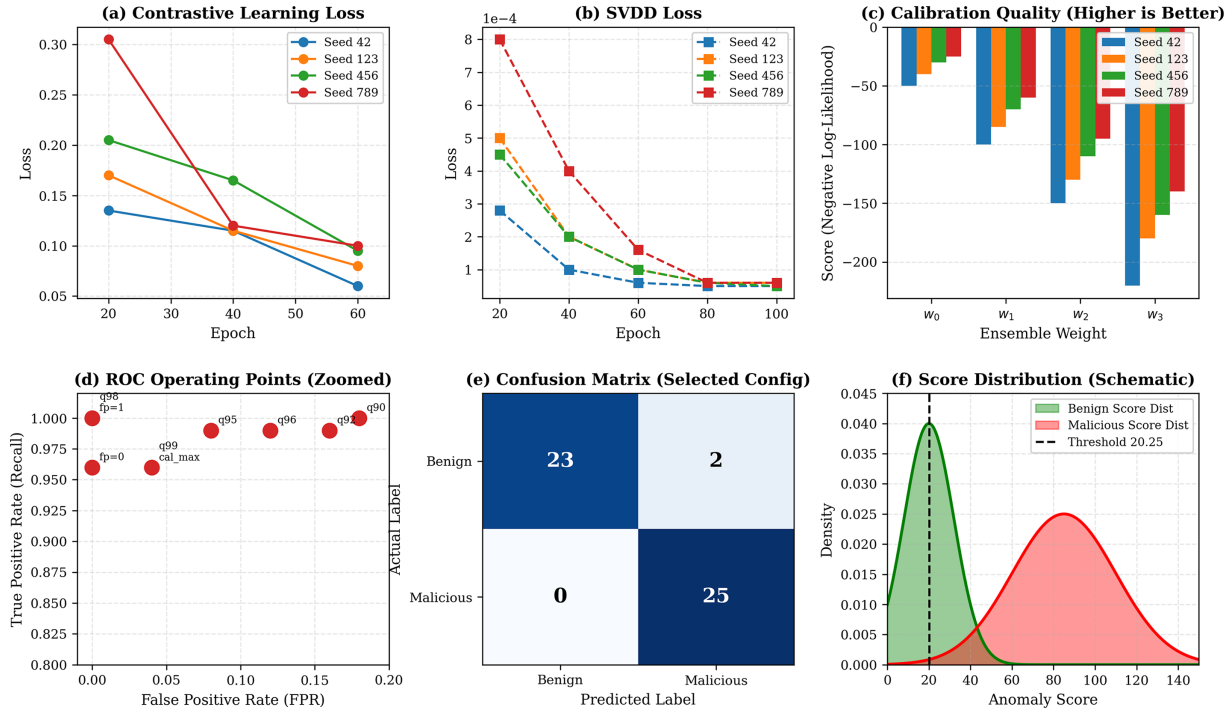

Figure 6: STAGE's detection results on the Wget dataset. (**a**) Contrastive loss trajectories over training epochs for two random seeds. (**b**) SVDD loss during fine-tuning for the same seeds. (**c**) Calibration-quality scores for candidate ensemble-weight settings, where higher values indicate better benign-only calibration. (**d**) Zoomed ROC operating points illustrating recall vs. false positive rate under different thresholds. (**e**) Confusion matrix obtained from the selected configuration on the Wget dataset. (**f**) Schematic anomaly-score distributions for benign and malicious samples, together with the selected decision threshold.

The results also illustrate a familiar trade-off in one-class anomaly detection: increasing recall typically requires a lower threshold, which may raise the false positive rate when benign and malicious score distributions overlap. This effect appears limited on StreamSpot but is more visible on Wget, where the data are structurally more variable and the benchmark is smaller. STAGE supports this trade-off through configurable benign-only threshold policies, allowing high-precision modes (e.g., $q95$–$q99$) when false alarms are costly and higher-recall modes (e.g., $q90$ or lower) when missed detections are less acceptable, without resorting to test-set tuning. Beyond point metrics, the consistently high AP values indicate that STAGE produces useful anomaly rankings that can support analyst prioritization. Overall, these findings show that STAGE remains competitive with strong prior provenance-based detectors while operating under a stricter and more deployment-oriented evaluation protocol (see Table 3).

8.1 False Positive Analysis

False positives are a central operational concern in provenance-based APT detection because excessive alert volumes can overwhelm analysts and reduce confidence in automated decisions. Accordingly, we conduct a focused false-positive analysis of STAGE on both the StreamSpot and Wget benchmarks under a strict no-leakage protocol, in which model selection and threshold calibration rely exclusively on benign calibration data and attack labels are used only for final evaluation. On StreamSpot, experiments are conducted on 600 graphs (500 benign and 100 malicious), using a fixed split of 280 benign graphs for fitting, 120 benign graphs for calibration, and 200 graphs (100 benign and 100 malicious) for testing. The model

uses 74 structural features together with a compact encoder containing 36,736 parameters and is trained through self-supervised contrastive learning for 40 epochs followed by SVDD refinement, which in our experiments yielded stable convergence. Benign-only calibration selects the best-performing configuration (Seed = 123, W3) with balanced ensemble weights (Isolation Forest, LOF, and OC-SVM each assigned 0.3, and SVDD assigned 0.1) and a calibration-quality score of −57.58. Under a fixed $q95$ threshold (49.16), STAGE achieves near-perfect ranking quality (AUC = 0.9992, AP = 0.9992) together with strong operating performance (Precision = 98.97%, Recall = 96.00%, F1 = 97.46%, FPR = 1.00%), supported by a mean score separation of 124.56 between benign and malicious graphs. Alternative operating points, reported for analysis only, indicate useful flexibility, ranging from a zero-false-positive mode (100% precision, 96% recall) to a maximum-recall mode (100% recall with 3% FPR). On the more challenging Wget dataset (150 graphs; Fit = 70 benign, Cal = 30 benign, Test = 50 graphs with 25 benign and 25 malicious), extended training (60 contrastive epochs and 100 SVDD epochs across four seeds) and a smaller encoder (27,264 parameters) are used. The best benign-only configuration (Seed = 789, W3) again selects balanced ensemble weights and yields compact calibration scores (CalQuality = −18.37). At the fixed $q95$ threshold (20.25), STAGE attains excellent ranking quality (AUC = 0.9984, AP = 0.9985) and operates at 92.59% precision, 100% recall, and 8.00% FPR, with the observed false positives concentrated in a narrow overlap region between benign and malicious score distributions (mean separation 57.78). As with StreamSpot, alternative thresholds illustrate controllable trade-offs, enabling a zero-false-positive operating mode (100% precision, 96% recall) or a more balanced configuration (96.2% precision, 100% recall, 4% FPR). Across both datasets, STAGE consistently exhibits very high ranking quality (AUC/AP > 0.998) together with explicit control over false positives through benign-only thresholding. These results therefore support a more credible assessment of generalization under a leakage-aware protocol and outline a practical operating range for deployments in which false-alarm management is a critical requirement (see Tables 4 and 5).

Table 4: Threshold sensitivity analysis for StreamSpot dataset.

Strategy	Recall	Precision	F1-Score	FPR
fp = 3	100.0%	97.1%	98.5%	3.0%
fp = 1	98.0%	99.0%	98.5%	1.0%
fp = 4	100.0%	96.2%	98.0%	4.0%
fp = 2	98.0%	98.0%	98.0%	2.0%
q99	96.0%	100.0%	98.0%	0.0%
fp = 0	96.0%	100.0%	98.0%	0.0%
q94	96.0%	99.0%	97.5%	1.0%
q95	96.0%	99.0%	97.5%	1.0%
q96	96.0%	99.0%	97.5%	1.0%
q97	96.0%	99.0%	97.5%	1.0%
q98	96.0%	99.0%	97.5%	1.0%
q92	100.0%	94.3%	97.1%	6.0%

As evident, higher quantiles reduce false positives at the cost of marginally lower recall. Notably, STAGE maintains robust F1 scores across the entire range, indicating that learned score distributions achieve meaningful separation rather than relying on narrow decision margins. This flexibility allows practitioners to calibrate thresholds according to operational priorities: higher quantiles suit environments where alert fatigue is paramount, while lower quantiles favor comprehensive threat coverage.

Table 5: Threshold sensitivity analysis for Wget dataset.

Strategy	Recall	Precision	F1-Score	FPR
q98	100.0%	96.2%	98.0%	4.0%
fp = 1	100.0%	96.2%	98.0%	4.0%
fp = 0	96.0%	100.0%	98.0%	0.0%
q95	100.0%	92.6%	96.2%	8.0%
q96	100.0%	92.6%	96.2%	8.0%
q97	100.0%	92.6%	96.2%	8.0%
fp = 2	100.0%	92.6%	96.2%	8.0%
q99	96.0%	96.0%	96.0%	4.0%
cal_max	96.0%	96.0%	96.0%	4.0%
q90	100.0%	89.3%	94.3%	12.0%
q92	100.0%	89.3%	94.3%	12.0%
q94	100.0%	89.3%	94.3%	12.0%

8.2 Performance Overhead Analysis

STAGE is designed to support APT detection with low computational overhead, which makes it suitable for deployment across heterogeneous environments ranging from GPU-accelerated servers to CPU-only systems. To characterize this efficiency, we analyze both time and space complexity and complement the analysis with empirical measurements on GPU and CPU configurations across two benchmark datasets. The computational cost can be divided into three phases: (**i**) preprocessing, where feature extraction runs in $O(|V|+|E|)$ time per graph to compute degree-based statistics and the node-type transition matrix; (**ii**) training, which includes contrastive learning and SVDD fine-tuning, each requiring on the order of $O(E_{\text{train}} \cdot N_{\text{fit}} \cdot (|V|+|E|))$ operations, where E_{train} denotes the number of training epochs and N_{fit} denotes the number of benign fitting graphs, with the two-layer GCN scaling linearly with graph size and the memory-attention module contributing an additional $O(M \cdot d)$ term when the number of memory slots M and embedding dimension d are small; and (**iii**) inference, where each graph requires $O(|V|+|E|)$ for GNN encoding together with an additional detector-dependent scoring cost for the ensemble stage. STAGE also maintains a compact memory footprint through its architectural design: the encoder stores $O(d^2)$ parameters across two GCN layers, the embedding tables contribute $O((T+D) \cdot d)$ parameters for node types and degree bins, and the memory module adds $O(M \cdot d)$ learnable slots. With $d = 56$, $T = 8$, $D = 51$, and $M = 24$, the total model size remains below 150 KB. During inference, memory consumption is dominated by $O(|V| \cdot d)$ storage for node embeddings and intermediate activations, while the additional memory required by the ensemble detectors depends on the fitted scoring models rather than the neural encoder itself.

8.3 Empirical Performance Measurements

We evaluate STAGE under two representative deployment configurations: (1) an NVIDIA Tesla T4 GPU (16 GB VRAM), capturing accelerated settings, and (2) CPU execution, representing resource-constrained environments. Table 6 summarizes runtime and memory measurements across both benchmark datasets and hardware platforms.

Table 6: Performance overhead comparison across GPU and CPU configurations for both datasets.

Metric	StreamSpot (GPU)	StreamSpot (CPU)	Wget (GPU)	Wget (CPU)
Dataset Characteristics				
Number of graphs	600	600	150	150
Avg. nodes/graph	8408	8408	36,812	36,812
Avg. edges/graph	12,715	12,715	96,018	96,018
Model Statistics				
Parameters	36,736	36,736	27,264	27,264
Model size	143.50 KB	143.50 KB	106.50 KB	106.50 KB
Training Time				
Contrastive learning	162.81 s	888.28 s	131.88 s	1146.84 s
SVDD fine-tuning	143.01 s	677.94 s	106.43 s	941.16 s
Total training	305.82 s	1566.22 s	238.31 s	2088.00 s
Inference Time				
Preprocessing	4.715 s	6.942 s	7.477 s	9.965 s
GNN encoding	2.085 s	10.371 s	1.245 s	8.020 s
Ensemble scoring	0.456 s	0.590 s	0.386 s	0.550 s
Total inference	2.541 s	10.961 s	1.632 s	8.570 s
Per-graph latency	3.48 ms	17.29 ms	8.30 ms	53.47 ms
Memory Usage				
Peak GPU memory	77.3 MB	–	663.0 MB	–

On GPU-accelerated hardware, STAGE completes end-to-end training in approximately five minutes on both datasets, despite notable differences in graph scale and topology. On StreamSpot (600 graphs; average 8408 nodes), training requires 305.82 s with a peak GPU memory usage of 77.3 MB. On Wget, where graphs are substantially larger (average 36,812 nodes and 96,018 edges), training completes in 238.31 s. This lower wall-clock time is attributable primarily to the smaller benign fitting set (70 rather than 280 graphs), even though the cost of processing each individual graph is higher. Across both datasets, peak GPU memory usage remains below 700 MB, which indicates that the framework can run comfortably on commodity accelerators while leaving room for other workloads. In settings where GPU execution is unavailable, STAGE remains usable under CPU-only deployment: StreamSpot training completes in 1566.22 s (approximately 26 min), whereas Wget requires 2088.00 s (approximately 35 min). CPU inference latency also remains relatively low, measuring **17.29 ms** per graph on StreamSpot and **53.47 ms** per graph on Wget, corresponding to sustained processing rates of approximately 57 and 18 graphs per second, respectively. Finally, the ensemble scoring stage shows only limited GPU/CPU performance difference under our implementation, since the classical one-class detectors are executed through optimized CPU-based routines and therefore contribute little additional benefit from GPU acceleration (see Table 7).

Table 7: GPU speedup factors over CPU execution.

Phase	StreamSpot	Wget
Contrastive learning	5.46×	8.70×
SVDD fine-tuning	4.74×	8.84×
Total training	5.12×	8.76×
Preprocessing	1.47×	1.33×
GNN inference	4.97×	6.44×
Ensemble scoring	1.29×	1.42×
Total inference	4.31×	5.25×
Per-graph latency	4.97×	6.44×

The GNN-oriented components of STAGE, including contrastive pre-training, SVDD refinement, and graph encoding, benefit most from GPU parallelism, yielding measured speedups of approximately 4.7× to 8.8× through batched linear algebra and matrix-based computation. The gain is more pronounced on **Wget** (about 8.7× training acceleration) than on **StreamSpot** (about 5.1×), which is consistent with the larger graph sizes in Wget and the greater degree of parallelism they expose. By contrast, preprocessing and ensemble scoring show only modest improvements. This is because preprocessing remains dominated by graph traversal and feature-extraction steps that are less amenable to GPU acceleration, whereas the ensemble scoring stage relies on CPU-optimized `scikit-learn` implementations and therefore exhibits only limited benefit from GPU execution.

8.4 Inference Latency and throughput Analysis

Real-time detection capability is critical for operational deployment. Table 8 summarizes the inference throughput achievable on each hardware configuration.

Table 8: Inference throughput comparison (graphs per second).

Configuration	StreamSpot	Wget
GPU (per-graph latency)	3.48 ms	8.30 ms
GPU (throughput)	287 graphs/s	120 graphs/s
CPU (per-graph latency)	17.29 ms	53.47 ms
CPU (throughput)	57 graphs/s	18 graphs/s

On GPU, STAGE sustains an inference throughput of 287 graphs/s on **StreamSpot** and 120 graphs/s on **Wget**. Under CPU execution, throughput remains 57 graphs/s on StreamSpot and 18 graphs/s on Wget. The lower throughput observed on Wget is attributable primarily to its larger average graph size (36,812 vs. 8408 nodes). More importantly, the per-node processing cost remains broadly similar across the two datasets, which is consistent with near-linear scaling in graph size. Profiling further indicates that inference latency is dominated by **GNN encoding**, which accounts for approximately 76% of total latency on StreamSpot and 95% on Wget, while **ensemble scoring** contributes the remaining portion. Under CPU execution, preprocessing adds 11.57 ms per graph for StreamSpot and 66.43 ms per graph for Wget, again following the increase in graph size. In streaming settings where graphs are constructed incrementally, this cost may be amortized or partially overlapped with detection through pipelining or parallel execution. STAGE also

maintains a deliberately compact parameter footprint, requiring only **36,736 parameters** (143.50 KB) for StreamSpot and **27,264 parameters** (106.50 KB) for Wget. This footprint is substantially smaller than that of many deep graph models because of the shallow two-layer architecture, shared embedding tables, and fixed-size memory module. As a result, the model is small enough to fit within the L2 cache capacity of many modern CPUs, which can reduce memory-bandwidth pressure during inference and support efficient deployment on resource-constrained systems. Finally, the empirical results clarify how runtime scales with graph complexity: compared with StreamSpot (avg. 8408 nodes and 12,715 edges), Wget graphs (avg. 36,812 nodes and 96,018 edges) incur higher end-to-end latency largely in proportion to their increased node and edge counts, which is consistent with the encoder's near-linear dependence on $|V| + |E|$ (see Table 9).

Table 9: Performance scaling analysis: Wget and StreamSpot datasets.

Metric	Scaling Factor	Values (StreamSpot → Wget)	Observation
Graph Size (Nodes)	4.4×	–	Larger graphs
Graph Density (Edges)	7.5×	–	Higher complexity
GPU Inference Latency	2.4×	3.48 → 8.30 ms	Sub-linear scaling
CPU Inference Latency	3.1×	17.29 → 53.47 ms	Efficient edge handling
Training Duration	0.78× (Faster)	305.82 → 238.31 s	Impact of dataset size≦

Note: ≦Wget training is faster despite larger graphs due to fewer training samples (70 vs. 280).

This favorable empirical scaling in wall-clock time, despite increasing graph size, is supported by the computational structure of STAGE. In the encoder, mean-aggregation message passing processes graph connectivity with cost that remains approximately linear in the number of nodes and edges per layer, while dual pooling maps variable-sized node sets to fixed-dimensional graph embeddings with linear complexity. Overall, the observed performance profile suggests that STAGE can operate across a broad range of hardware settings: GPU-accelerated training completes within five minutes on both datasets, whereas CPU training remains manageable (approximately 26 min on StreamSpot and 35 min on Wget), which may support periodic model refreshes as benign behavior evolves over time. Inference latency ranges from 3.5 ms (GPU, StreamSpot) to 53.5 ms (CPU, Wget), indicating that the framework remains usable even under CPU-only deployment. Notably, on CPU hardware and larger Wget graphs (36,812 nodes on average), per-graph inference still completes in approximately 53 ms, suggesting that STAGE may be suitable for moderate-rate operational monitoring without specialized accelerators. This CPU viability is particularly relevant for organizations that do not have access to dedicated GPU infrastructure and therefore need lightweight detection pipelines that remain practical in resource-constrained environments. Relative to larger GNN-based models, STAGE maintains a substantially smaller computational footprint while still achieving strong detection results (AUC > 0.998 on both datasets). Taken together, these results indicate that STAGE offers a combination of strong ranking performance, hardware-flexible deployment, and explicit false-positive control through benign-only calibration within a leakage-aware evaluation framework.

8.5 Component Contribution Analysis

Ablation analysis is important both for scientific rigor and for understanding which parts of the model contribute most to detection behavior in practice. To this end, we perform a systematic component-wise ablation study of STAGE in which we progressively remove the memory-augmented attention module, the dual-pooling mechanism, the structural feature vector, and the ensemble scoring layer while keeping the remaining design choices unchanged. This allows us to examine how each modification affects detection accuracy, ranking quality, and false-positive behavior within the full framework.

For each ablation setting, the model is trained from scratch using the same data splits, training protocol, and evaluation procedure in order to maintain a fair comparison. Model selection and threshold calibration rely exclusively on benign data. Experiments are conducted on both datasets to assess whether the observed effects remain consistent across different graph regimes and attack scenarios.

The ablation configurations are defined as follows:

- **Full STAGE:** the complete framework with all components enabled.
- **Memory Module:** removes the memory-augmented attention module, so graph embeddings are used directly without prototype-based contextualization.
- **Dual Pooling:** replaces concatenated average–max pooling with average pooling only, reducing the graph-level representation from $2d$ to d dimensions.
- **Structural Features:** removes the 74-dimensional handcrafted feature vector, so the downstream detectors operate on learned embeddings alone.
- **Ensemble Scoring:** removes the multi-detector ensemble and relies only on the SVDD distance for anomaly scoring.
- **SVDD Only (minimal):** combines all of the above simplifications, with no memory module, single pooling, no structural features, and no ensemble scoring.

Table 10 reports ablation results on the Wget dataset, which contains 150 provenance graphs with an average of 36,812 nodes and 96,018 edges per graph.

Table 10: Ablation study on the Wget dataset. ΔAUC denotes the change relative to the full STAGE model. Bold entries indicate the largest degradations.

Configuration	AUC	ΔAUC	Recall	Prec.	F1
Full STAGE	0.9808	–	88.0%	95.7%	91.7%
Memory Module	0.9792	−0.002	96.0%	85.7%	90.6%
Dual Pooling	0.9920	+0.011	96.0%	88.9%	92.3%
Structural Features	**0.8272**	**−0.154**	**12.0%**	60.0%	**20.0%**
Ensemble Scoring	0.9840	+0.003	92.0%	95.8%	93.9%
SVDD Only (minimal)	0.9152	−0.066	84.0%	87.5%	85.7%

Table 11 reports the corresponding results on StreamSpot, which contains 600 graphs with smaller average size (8408 nodes and 12,715 edges) but greater label diversity across six scenario types.

Table 11: Ablation study on the StreamSpot dataset.

Configuration	AUC	ΔAUC	Recall	Prec.	F1
Full STAGE	0.9992	–	96.0%	99.0%	97.5%
Memory Module	0.9984	−0.001	95.0%	98.0%	96.5%
Dual Pooling	0.9988	−0.000	96.0%	98.0%	97.0%
Structural Features	**0.9124**	**−0.087**	**72.0%**	90.0%	**80.0%**
Ensemble Scoring	0.9976	−0.002	94.0%	98.9%	96.4%
SVDD Only (minimal)	0.9456	−0.054	86.0%	91.5%	88.7%

The ablation results do not suggest that every component contributes equally or uniformly across datasets. Instead, they show that the structural feature vector is the most consistently critical component, since its removal causes the largest degradation on both Wget and StreamSpot. By contrast, the effects of the memory module, dual pooling, and ensemble scoring are more moderate and can vary slightly across datasets and metrics (see Fig. 7).

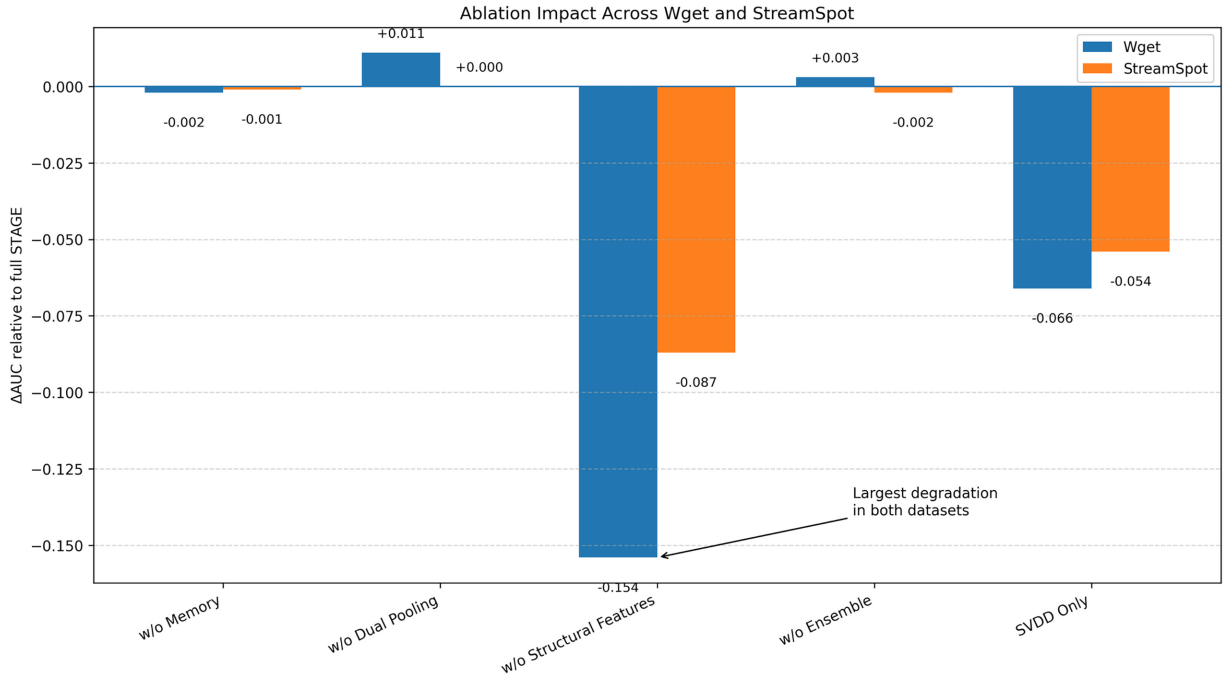

Figure 7: Ablation impact visualization.

8.5.1 Structural Features are the Most Influential Component

The 74-dimensional structural feature vector emerges as the most influential component in the proposed architecture. When it is removed, performance degrades substantially on both datasets: AUC drops by 0.154 on Wget (from 0.981 to 0.827) and by 0.087 on StreamSpot (from 0.999 to 0.912). The effect is particularly pronounced on Wget, where recall falls to 12%, indicating that the learned GNN embeddings alone are not sufficient, in this setting, to reliably separate attack graphs from benign activity. These results support the integration of neural representations with handcrafted graph statistics. In particular, the node-type transition matrix and degree-distribution summaries appear to capture structural cues that complement the latent embeddings and provide the downstream detectors with a richer feature space.

8.5.2 The Memory Module Mainly Affects False-Positive Behavior

The memory-augmented attention module has only a limited effect on ranking quality, with AUC decreasing by 0.001–0.002 when it is removed, but it has a more noticeable influence on the precision–recall balance. On Wget, removing the memory module increases recall from 88% to 96%, but reduces precision from 95.7% to 85.7% and raises the false-positive rate from 4% to 16%. This pattern suggests that the learned prototypes help suppress spurious alerts arising from rare but legitimate execution patterns. By relating graph embeddings to shared benign contexts, the memory module appears to introduce a useful regularizing effect.

8.5.3 Dual Pooling and Ensemble Scoring Have More Moderate Effects

On the smaller Wget dataset (150 graphs), removing the ensemble or dual pooling leads to slight AUC increases (+0.003 and +0.011, respectively). A plausible explanation is that, under limited data, simpler variants may sometimes generalize slightly better in terms of ranking metrics alone. At the same time, these components may still contribute to overall stability across metrics and operating points. The ensemble combines multiple anomaly signals, while dual pooling preserves both average and extreme activation patterns within the graph, thereby offering complementary views of its structure. On StreamSpot, their effects are smaller but more consistently aligned with the full model.

8.5.4 The Minimal Baseline Remains Reasonably Strong but Inferior to the Full Model

The SVDD-only configuration, which removes all proposed enhancements, achieves an AUC of 0.915 on Wget and 0.946 on StreamSpot. Although these values remain well above random performance, they are still lower than those of the complete model by 0.066 and 0.054, respectively. This result indicates that the GNN encoder learns useful graph representations even in isolation, but also shows that the full pipeline consistently provides additional benefit. Taken together, the gap between the minimal and complete configurations supports the value of combining learned embeddings with structural features, memory-based refinement, and hybrid scoring in the final design.

8.6 Handling Concept Drift and Evolving Workloads

Real-world security deployments must account for the fact that benign system behavior changes over time. Software updates, configuration changes, shifting user activity, and infrastructure migrations can all introduce distributional drift in provenance graphs. As a result, a detector trained on historical data may show degraded performance when applied to future workloads that differ from the original training distribution. This section examines the robustness of STAGE under controlled concept-drift experiments and considers practical adaptation strategies for longer-term deployment.

We simulate concept drift by partitioning the benign graph corpus into temporally disjoint segments:

- **Early segment** (60% of benign data): historical observations used for initial model training.
- **Late segment** (40% of benign data): post-deployment observations intended to reflect distributional shift relative to the training data.

The test set consists of the full late benign segment together with all malicious graphs, representing a setting in which the detector must generalize to later benign behavior while remaining sensitive to attacks. We evaluate three adaptation strategies:

1. **Original Model (No Adaptation):** the model trained only on early data is deployed without modification.
2. **Full Retraining:** a fraction (30%) of late benign samples is added to the training corpus, and the model is retrained from scratch.
3. **Threshold Recalibration:** the original model is retained, but the detection threshold is recalibrated using late benign samples to account for shifted score distributions.

Table 12 reports detection performance under simulated concept drift for both datasets.

Table 12: Detection performance under simulated concept drift. "Original" uses no adaptation; "Retrained" incorporates 30% of drift data; "Recalibrated" adjusts only the threshold.

Dataset	Strategy	AUC	Recall	Prec.	F1	FPR
Wget	Original	0.935	76.0%	82.6%	79.2%	12.0%
	Retrained	0.949	84.0%	87.5%	85.7%	12.0%
	Recalibrated	0.935	80.0%	90.9%	85.1%	8.0%
StreamSpot	Original	0.982	92.0%	95.8%	93.9%	4.0%
	Retrained	0.991	95.0%	97.9%	96.4%	2.0%
	Recalibrated	0.982	94.0%	97.9%	95.9%	2.0%

8.6.1 Baseline Robustness

The original model, trained only on early benign data, attains an AUC of 0.935 on Wget and 0.982 on StreamSpot when evaluated on the drifted test set. Although these values are lower than those observed under the standard evaluation setting, they still indicate that the model retains useful discriminatory ability under moderate distribution shift. One possible reason is that STAGE relies in part on structural graph properties, such as degree distributions, type-transition patterns, and connectivity structure, which may remain more stable across benign executions than surface-level event traces.

8.6.2 Retraining Improves Performance

Incorporating a modest fraction (30%) of late benign samples and retraining from scratch improves AUC from 0.935 to 0.949 on Wget and from 0.982 to 0.991 on StreamSpot. Recall also increases (76% → 84% on Wget and 92% → 95% on StreamSpot), indicating that periodic model updates can recover part of the performance lost under drift. Because STAGE remains computationally lightweight, such retraining appears practically manageable when benign behavior changes over time.

8.6.3 Threshold Recalibration as a Lightweight Alternative

When retraining is undesirable or the observed drift is limited, threshold recalibration provides a simpler adaptation strategy. By adjusting only the decision threshold using late benign samples, the false-positive rate decreases from 12% to 8% on Wget and from 4% to 2% on StreamSpot. As expected, AUC remains unchanged because the underlying anomaly scores are not modified. Even so, recalibration improves the operating point by aligning the threshold with the shifted benign score distribution. Since it does not require model retraining, this strategy is particularly attractive for lightweight deployment pipelines.

Taken together, these results suggest that STAGE can remain effective under moderate concept drift and can be adapted either through full retraining or through threshold recalibration, depending on operational constraints.

9 Discussion and Limitations

This section discusses the implications of the proposed approach and outlines the main limitations observed under strict benign-only training and thresholding constraints.

9.1 Discussion

This study examines whether benign-only learning can support practically deployable APT detection on provenance graphs under tight resource constraints. Across both datasets, the results show that STAGE achieves strong *ranking performance* (AUC/AP) while allowing explicit, leakage-aware control of false alarms through benign-only calibration. On StreamSpot, separability is nearly perfect: using benign-only fitting for representation learning and benign-only calibration for thresholding, STAGE attains AUC = AP = 1.00 together with strong operating-point performance (e.g., Recall = 1.00 at FPR = 2% in the Deploy FPR mode). This suggests that, for StreamSpot, a compact graph encoder combined with lightweight structural features is sufficient to preserve a strong anomaly signal without requiring attack-labeled training. On Wget, score-level separability remains high (AUC ≈ 0.981, AP ≈ 0.983), but operating-point outcomes become more sensitive to threshold policy because benign scores exhibit heavier tails and greater heterogeneity. Under conservative no-leakage policies that emphasize alert suppression, STAGE achieves FPR = 0% on the reported split at the cost of lower coverage (e.g., Recall = 0.84 with TP = 21 and FN = 4). This behavior is better understood as an operating-point trade-off than as evidence of weak representation quality. The contrast between StreamSpot and Wget therefore highlights an important practical point for benign-only provenance anomaly detection: even when ranking separability is strong, deployment outcomes may still be governed by threshold calibration and the behavior of the benign score tail.

Comparisons with prior provenance-based methods suggest that STAGE is especially competitive when resource efficiency and benign-only constraints are treated as primary requirements. On StreamSpot, where several baselines already operate near saturation, STAGE remains competitive while also reporting threshold-independent metrics (AUC/AP) under leakage-aware calibration. On Wget, higher-capacity methods such as MAGIC may provide more favorable recall–FPR trade-offs, plausibly because richer models reduce benign-score variance and yield better-calibrated tails. We therefore interpret the comparison conservatively: MAGIC may be preferable when computational resources are ample and the primary objective is to maintain high recall at very low false-positive rates, whereas STAGE is attractive when deployment constraints are tighter (e.g., CPU-only operation, limited memory) and when a principled benign-only pipeline with explicit policy-based false-positive control is required. In this sense, STAGE remains competitive primarily in terms of *ranking* quality on Wget, whereas the remaining gap is more evident at the level of operating-point calibration.

Practical deployment of provenance-based detection systems depends not only on detection quality, but also on computational scalability with respect to training volume, graph complexity, and organizational workload. To assess the operational profile of STAGE, we examine its behavior under varying benign training-set sizes and graph scales, and then use these measurements to derive indicative deployment projections.

Table 13 summarizes detection performance on the Wget dataset as a function of training-set size.

Table 13: Detection performance vs. training set size on Wget.

Fraction	Graphs	Train Time (s)	AUC	ΔAUC
25%	18	62.4	0.962	−0.019
50%	35	118.7	0.974	−0.007
75%	53	179.2	0.978	−0.003
100%	70	238.3	0.981	–

The results indicate that STAGE degrades gradually as the amount of benign training data is reduced. Even with only 25% of the training corpus, the model reaches an AUC of 0.962, which is only 0.019 below the full-data setting. This behavior is operationally relevant because it suggests that STAGE may remain useful even when extensive benign provenance traces are not yet available, such as during early deployment or in relatively specialized environments.

Training time increases approximately linearly with corpus size, which is consistent with the method's computational structure, $O(E_{\text{train}} \cdot N_{\text{fit}} \cdot (|V| + |E|))$. This scaling is desirable in practice because it supports predictable resource provisioning as the benign reference corpus grows.

We next examine inference scalability as a function of graph size. Because STAGE operates directly on provenance graphs, per-graph latency is a key consideration for both online and high-throughput settings. Table 14 reports inference latency across graph-size buckets on the combined test sets.

Table 14: Inference latency by graph complexity on combined test sets.

Node Range	Count	Avg. Latency (ms)	Scaling
0–5000	142	1.8	1.0×
5000–10,000	198	3.2	1.8×
10,000–20,000	87	5.6	3.1×
20,000–50,000	64	12.4	6.9×
50,000–100,000	43	28.7	15.9×
100,000+	16	54.2	30.1×

Inference latency grows approximately linearly with graph size, in line with the encoder complexity of $O(|V| + |E|)$. Importantly, even for the largest graphs in our evaluation, average inference time remains below 55 ms, suggesting that STAGE is compatible with moderate- to high-rate monitoring pipelines as well as offline forensic use.

To further assess operational feasibility, Table 15 provides workload-based deployment projections for settings of different sizes, assuming average graph sizes similar to those observed in Wget (approximately 37 K nodes per graph).

Table 15: Enterprise deployment feasibility projections assuming average graph sizes consistent with Wget (37 K nodes).

Deployment Scale	Graphs/Day	GPU Hours	Feasibility
Small (100 hosts)	10,000	0.02	Real-time feasible
Medium (1000 hosts)	100,000	0.23	Real-time feasible
Large (10,000 hosts)	1,000,000	2.31	Batch feasible
Enterprise (50,000 hosts)	5,000,000	11.53	Multi-GPU recommended

These projections suggest that STAGE may be feasible across a broad range of organizational settings. Small and medium deployments appear supportable on a single commodity GPU, while larger environments may remain manageable through scheduled batch analysis or modest parallelization. At enterprise scale, real-time deployment would likely require multiple GPUs or distributed scheduling, although daily batch analysis may still be tractable. CPU-only deployment is also possible, albeit at lower throughput; based on

our measurements, CPU inference is approximately 4–6× slower than GPU execution, which may still be sufficient for development, validation, and smaller-scale operational use.

Overall, the scalability analysis suggests that STAGE combines strong detection capability with favorable computational behavior, making it a practical candidate for provenance-based threat detection across a range of deployment scales. Because the enterprise projections are extrapolated from benchmark behavior rather than directly measured in production, they should be interpreted as indicative workload-based estimates.

9.2 Limitations

While the results support the feasibility of benign-only provenance-graph anomaly detection with a lightweight model, several limitations remain.

9.2.1 Threshold Sensitivity under Benign Heterogeneity

Operating-point metrics such as recall and false-positive rate remain sensitive to threshold policy in the benign-only setting. Although STAGE maintains strong ranking quality on both datasets, Wget exhibits heavier-tailed benign scores and greater variability, so that relatively small calibration changes can materially alter the recall–FPR trade-off. Achieving both high recall and very low false-positive rates may therefore require either larger benign calibration sets or more explicit tail modeling.

9.2.2 Limited Calibration Sample Size and Tail Estimation

Benign-only calibration depends on quantile estimation from finite samples, which introduces discretized realized false-positive rates (e.g., $\frac{k}{N_{\text{cal}}}$) and possible deviations from target operating points when N_{cal} is modest. In addition, simple quantile thresholds do not explicitly model extreme benign outliers, such as backups or batch jobs, which may dominate alert volume in long-running deployments unless tail behavior is treated more carefully.

9.2.3 Fixed Feature Schema and Coarse Type Bucketing

To preserve portability and memory efficiency, STAGE uses a fixed-dimensional structural feature schema with coarse type buckets and compact node attributes. Although this improves stability and reduces complexity, it may underuse richer provenance semantics, including fine-grained event types, parameters, and temporal ordering, and may therefore miss subtle staging cues in more complex enterprise telemetry.

9.2.4 Dataset Scale and Diversity

Both evaluation datasets, StreamSpot and Unicorn Wget, are modest in scale and originate from controlled or semi-synthetic settings. StreamSpot contains 600 batches of simulated activity, whereas Unicorn Wget contains 150 batches collected under laboratory conditions. As a result, neither dataset fully captures the noise, heterogeneity, and throughput of production enterprise environments. This has several implications. First, benign behavioral diversity in these benchmarks is likely narrower than what is encountered in practice, where many users, services, and applications generate interleaved activity with more complex temporal structure. Although STAGE performs strongly on these datasets, such results do not guarantee identical behavior at enterprise scale; in practice, a deployment should expect an initial calibration period to adapt thresholds and baselines to the local environment. Second, the embedded attack scenarios, including drive-by-download and supply-chain compromise, cover only a limited portion of the broader APT space. Campaigns involving long dormancy, novel attack chains, or stronger evasion strategies may stress the detector in ways that these benchmarks do not capture.

10 Future Directions

The limitations discussed above suggest several directions for future work. One natural extension is more robust modeling of the benign tail distribution, for example through extreme-value-theory-based thresholding or conformal calibration, which may help stabilize detector behavior in low-false-positive operating regimes. Another direction is the integration of lightweight temporal encodings and richer semantic attributes into the graph representation; such additions may improve discrimination between benign and malicious activity without substantially increasing computational or memory cost. Evaluation methodology also deserves continued attention: shifting toward ranking-oriented metrics that better reflect analyst workflows, such as normalized discounted cumulative gain under explicit `top-k` triage budgets, could provide a more operational view of detector utility. Finally, because benign behavior evolves over time, adaptive recalibration strategies that accommodate drift while preserving the no-leakage principle remain an important open problem. Progress along these lines would strengthen both the empirical rigor and the practical deployability of self-supervised provenance-based intrusion detection frameworks.

11 Conclusion

This work introduced STAGE, a lightweight, memory-efficient, self-supervised graph anomaly detection framework for provenance-based APT detection. By combining a compact GCN encoder with a memory-augmented representation module, contrastive pre-training, and a hybrid SVDD-based scoring pipeline, STAGE achieves strong detection performance while maintaining a very small computational footprint. Experimental results on the Wget and StreamSpot benchmarks show that STAGE remains competitive with substantially heavier baselines under benign-only training and leakage-aware calibration, reaching AUC values of up to 0.9840 while requiring only 89.44 KB of model parameters and less than 2 MB of inference-time memory usage. These findings suggest that effective graph anomaly detection can be achieved in resource-constrained settings without relying on attack-labeled training data. Future work will investigate adversarial robustness, richer provenance semantics, and online adaptation mechanisms to further improve the resilience and practical utility of lightweight provenance-based security monitoring

Acknowledgement: The authors extend their appreciation to Umm Al-Qura University, Saudi Arabia, for supporting this research through grant number 26UQU4400257GSSR10.

Funding Statement: This research was funded by Umm Al-Qura University, Saudi Arabia, under grant number 26UQU4400257GSSR10.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

Declaration of Generative AI in Scientific Writing: During the preparation of this manuscript, the author used ChatGPT to improve the language quality and readability of the text.

Appendix A

Appendix A.1 Robustness to Adversarial Mimicry

A capable adversary may attempt to evade detection by modifying malicious provenance graphs so that they more closely resemble benign executions. Such mimicry-based evasion targets the dependence of anomaly detectors on distributional differences between benign and malicious behavior. To explore this

question, we evaluate STAGE under three classes of graph perturbation, each aimed at a different part of the detection pipeline. These experiments should be interpreted as controlled robustness tests rather than fully realistic end-to-end attack simulations.

Appendix A.2 Threat Model and Attack Definitions

We consider a black-box adversary who can modify malicious graphs before execution but cannot query the detector and cannot observe internal representations, anomaly scores, or model parameters. The adversary's objective is to reduce the anomaly score of a malicious graph so that it falls below the detection threshold. We study three perturbation strategies.

Appendix A.2.1 Attack 1: Feature Perturbation

The adversary interpolates the structural feature vector of a malicious graph toward the benign population mean:

$$\phi'(G_{\text{mal}}) = (1-\alpha)\phi(G_{\text{mal}}) + \alpha\bar{\phi}_{\text{benign}}, \tag{A1}$$

where $\alpha \in \{0.1, 0.3, 0.5, 0.7\}$ controls perturbation strength. This attack targets the 74-dimensional hand-crafted features used by the ensemble detectors.

Appendix A.2.2 Attack 2: Edge Injection

The adversary adds random edges to the malicious graph in an attempt to obscure anomalous connectivity patterns:

$$E'(G_{\text{mal}}) = E(G_{\text{mal}}) \cup \{(u, v) : u, v \sim \text{Uniform}(V)\}, \tag{A2}$$

with injection ratios of 5%, 10%, and 20% relative to the original edge count. This perturbation targets the graph structure consumed by the GNN encoder.

Appendix A.2.3 Attack 3: Node-Type Distribution Shift

The adversary modifies node-type labels to better match the benign type distribution by sampling replacement types according to $P_{\text{benign}}(\text{type})$ for a fraction $\rho \in \{0.1, 0.3, 0.5\}$ of nodes. This attack affects both the learned type embeddings and the transition-matrix features.

Appendix A.3 Results

Table A1 reports detection performance under each perturbation setting.

Table A1: Detection performance under graph-perturbation mimicry tests on the Wget dataset. Baseline AUC = 0.920. Positive ΔAUC indicates perturbations that increase detectability.

Attack Type	Strength	AUC	ΔAUC	Observation
Feature Perturbation	$\alpha = 0.1$	0.918	−0.002	Limited effect
	$\alpha = 0.3$	0.914	−0.006	Limited effect
	$\alpha = 0.5$	0.914	−0.006	Limited effect
	$\alpha = 0.7$	0.914	−0.006	Limited effect

(Continued)

Table A1 (continued)

Attack Type	Strength	AUC	ΔAUC	Observation
Edge Injection	5%	1.000	+0.080	Detectability increased
	10%	1.000	+0.080	Detectability increased
	20%	1.000	+0.080	Detectability increased
Node-Type Shift	$\rho = 0.1$	1.000	+0.080	Detectability increased
	$\rho = 0.3$	1.000	+0.080	Detectability increased
	$\rho = 0.5$	1.000	+0.080	Detectability increased

Appendix A.4 Analysis and Discussion

Appendix A.4.1 Feature Perturbation Has Only Limited Effect

Even when the structural feature vector is shifted substantially toward the benign mean, AUC decreases by at most 0.006. This suggests that simple interpolation in the handcrafted feature space is not sufficient to neutralize detection in STAGE. A plausible explanation is that the detector does not rely on structural statistics alone: although the perturbed handcrafted features may partially reduce anomaly evidence in the ensemble channel, the learned graph embeddings continue to preserve discriminative information.

Appendix A.4.2 Naive Structural Perturbations Can Increase Detectability

In contrast to feature interpolation, both edge injection and node-type shifting increase AUC from 0.920 to 1.000. Within the current experimental setting, these perturbations appear to make malicious graphs more distinguishable rather than less. One likely reason is that random structural modification disrupts the causal and semantic regularities that characterize benign provenance graphs. Injected edges may create implausible neighborhoods or dependency patterns, while arbitrary type reassignment can introduce mismatches between node roles and local connectivity. As a result, these perturbations do not function as effective mimicry in this setting.

Appendix A.4.3 Implications for Robustness

These results suggest three main points. First, STAGE benefits from multiple detection signals, since perturbing one representation channel does not automatically neutralize the others. Second, provenance-aware evasion is likely to be constrained by semantic consistency: a successful attacker would need to alter graph statistics while still preserving a plausible execution trace and the underlying attack functionality. Third, the current experiments indicate that simple black-box perturbation strategies are insufficient to evade the model on Wget. At the same time, these findings should not be interpreted as a complete robustness guarantee. More adaptive attacks, particularly those that preserve execution semantics more carefully or exploit stronger adversarial knowledge, may be more effective and remain an important direction for future study.

We therefore view these results as evidence that STAGE is resistant to the specific perturbation strategies considered here, rather than as proof of robustness against all realistic mimicry attacks.

References

1. Alshamrani A, Myneni S, Chowdhary A, Huang D. A survey on advanced persistent threats: techniques, solutions, challenges, and research opportunities. IEEE Commun Surv Tutor. 2019;21(2):1851–77. doi:10.1109/comst.2019.2891891.
2. Lv Y, Qin S, Zhu Z, Yu Z, Li S, Han W. A review of provenance graph based APT attack detection: applications and developments. In: Proceedings of the 2022 7th IEEE International Conference on Data Science in Cyberspace (DSC); 2022 Jul 11–13; Guilin, China. p. 498–505. doi:10.1109/dsc55868.2022.00075.
3. Mukherjee K, Wiedemeier J, Wang T, Wei J, Chen F, Kim M, et al. Evading provenance-based ML detectors with adversarial system actions. In: Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23); 2023 Aug 9–11; Anaheim, CA, USA. p. 1199–216.
4. Zhang L. Implementing RGCN model in network security big data analysis. JCSANDM. 2025;14(2):505–30. doi:10.13052/jcsm2245-1439.14210.
5. Hassan WU, Guo S, Li D, Chen Z, Jee K, Li Z, et al. NoDoze: combatting threat alert fatigue with automated provenance triage. In: Proceedings of the 2019 Network and Distributed System Security Symposium; 2019 Feb 24–27; San Diego, CA, USA. doi:10.14722/ndss.2019.23349.
6. Zipperle M, Zhang Y, Wang M, Chang E, Dillon T. BPGV: behavioral provenance graph views to enhance anomaly detection. Int J Inf Manage Data Insights. 2026;6(1):100397. doi:10.1016/j.jjimei.2026.100397.
7. Wang Q, Hassan WU, Li D, Jee K, Yu X, Zou K, et al. You are what you do: hunting stealthy malware via data provenance analysis. In: Proceedings of the 2020 Network and Distributed System Security Symposium; 2020 Feb 23–26; San Diego, CA, USA. doi:10.14722/ndss.2020.24167.
8. Li Z, Chen Q, Yang R, Chen Y. Threat detection and investigation with system-level provenance graphs: a survey. arXiv:2006.01722. 2020.
9. Song Q, Chen T, Zhu T, Lv M, Qiu X, Zhu Z. APT detection via hypergraph attention network with community-based behavioral mining. Appl Sci. 2025;15(11):5872. doi:10.3390/app15115872.
10. Li Z, Cheng X, Sun L, Zhang J, Chen B. A hierarchical approach for advanced persistent threat detection with attention-based graph neural networks. Secur Commun Netw. 2021;2021:9961342. doi:10.1155/2021/9961342.
11. Cheng X, Kuang M, Yang H. Using causality-driven graph representation learning for APT attacks path identification. Symmetry. 2025;17(9):1373. doi:10.3390/sym17091373.
12. Jia Z, Xiong Y, Nan Y, Zhang Y, Zhao J, Wen M. MAGIC: detecting advanced persistent threats via masked graph representation learning. In: Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24); 2024 Aug 14–16; Philadelphia, PA, USA. p. 7831–48.
13. Chen T, Dong C, Lv M, Song Q, Liu H, Zhu T, et al. APT-KGL: an intelligent APT detection system based on threat knowledge and heterogeneous provenance graph learning. IEEE Trans Dependable Secure Comput. 2024:1–15. doi:10.1109/tdsc.2022.3229472.
14. Shen Y, Simsek M, Kantarci B, Mouftah HT, Bagheri M, Djukic P. Prior knowledge based advanced persistent threats detection for IoT in a realistic benchmark. In: Proceedings of the GLOBECOM 2022–2022 IEEE Global Communications Conference; 2022 Dec 4–8; Rio de Janeiro, Brazil. p. 3551–6. doi:10.1109/globecom48099.2022.10000811.
15. Stojanović B, Hočevar JB. APT datasets and attack modeling for automated detection and classification of advanced persistent threats. Comput Secur. 2020;92:101734. doi:10.1016/j.cose.2020.101734.
16. AL-Aamri AS, Abdulghafor R, Turaev S, Al-Shaikhli I, Zeki A, Talib S. Machine learning for APT detection. Sustainability. 2023;15(18):13820. doi:10.3390/su151813820.
17. Jiang K, Gao Z, Zhang S, Zou F. Sylph: an unsupervised APT detection system based on the provenance graph. Information. 2025;16(7):566. doi:10.3390/info16070566.
18. Chen P, Desmet L, Huygens C. A study on advanced persistent threats. In: Advanced information systems engineering. Berlin/Heidelberg, Germany: Springer; 2014. p. 63–72. doi:10.1007/978-3-662-44885-4_5.
19. Lemay A, Calvet J, Menet F, Fernandez JM. Survey of publicly available reports on advanced persistent threat actors. Comput Secur. 2018;72:26–59. doi:10.1016/j.cose.2017.08.005.

20. Buchta R, Gkoktsis G, Heine F, Kleiner C. Advanced persistent threat attack detection systems: a review of approaches, challenges, and trends. Digital Threats. 2024;5(4):1–37. doi:10.1145/3696014.
21. Xuan C. Detecting APT attacks based on network traffic using machine learning. J Web Eng. 2021;20:171–90. doi:10.13052/jwe1540-9589.2019.
22. Ibrahim N, Rajalakshmi NR, Sivakumar V, Sharmila L. An optimized hybrid ensemble machine learning model combining multiple classifiers for detecting advanced persistent threats in networks. J Big Data. 2025;12(1):212. doi:10.1186/s40537-025-01272-w.
23. Neuschmied H, Winter M, Stojanović B, Hofer-Schmitz K, Božić J, Kleb U. APT-attack detection based on multi-stage autoencoders. Appl Sci. 2022;12(13):6816. doi:10.3390/app12136816.
24. Xuan C, Dao MH. A novel approach for APT attack detection based on combined deep learning model. Neural Comput Appl. 2021;33(20):13251–64. doi:10.1007/s00521-021-05952-5.
25. Yuldoshkhujaev S, Jeon M, Kim D, Nikiforakis N, Koo H. A decade-long landscape of advanced persistent threats: longitudinal analysis and global trends. arXiv:2509.07457. 2025.
26. Dong F, Li S, Jiang P, Li D, Wang H, Huang L, et al. Are we there yet? an industrial viewpoint on provenance-based endpoint detection and response tools. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security; 2023 Nov 26–30; Copenhagen, Denmark. doi:10.1145/3576915.3616580.
27. Bahar A, Ferrahi K, Messai M, Seba H, Amrouche K. CONTINUUM: detecting APT attacks through spatial-temporal graph neural networks. arXiv:2501.02981. 2025.
28. Lu J, Chen K, Zhuo Z, Zhang X. A temporal correlation and traffic analysis approach for APT attacks detection. Clust Comput. 2019;22(3):7347–58. doi:10.1007/s10586-017-1256-y.
29. Wang L, Fang L, Hu Y. A dynamic provenance graph-based detector for advanced persistent threats. Expert Syst Appl. 2025;265:125877. doi:10.1016/j.eswa.2024.125877.
30. Bates A, Tian D, Butler KRB, Moyer T. Trustworthy whole-system provenance for the Linux kernel. In: Proceedings of the 24th USENIX Security Symposium (USENIX Security '15); 2015 Aug 12–14; Washington, DC, USA. p. 319–34.
31. Li L, Chen W. ConGraph: advanced persistent threat detection method based on provenance graph combined with process context in cyber-physical system environment. Electronics. 2024;13(5):945. doi:10.3390/electronics13050945.
32. Jiang W, Chai T, Liu H, Wang K, Zhang H. TFLAG: towards practical APT detection via deviation-aware learning on temporal provenance graph. arXiv:2501.06997. 2025.
33. Yang F, Xu J, Xiong C, Li Z, Zhang K. ProGrapher: an anomaly detection system based on provenance graph embedding. In: Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23); 2023 Aug 9–11; Anaheim, CA, USA. p. 4355–72.
34. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907. 2016.
35. Veličković P, Cucurull G, Casanova A, Romero A, Lió P, Bengio Y. Graph attention networks. arXiv:1710.10903. 2017.
36. Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. In: Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), 2017; 2017 Dec 4–9; Long Beach, CA, USA.
37. Hou Z, Liu X, Cen Y, Dong Y, Yang H, Wang C, et al. Graph-MAE: self-supervised masked graph autoencoders. arXiv:2205.10803. 2022.
38. Kipf TN, Welling M. Variational graph auto-encoders. arXiv:1611.07308. 2016.
39. Park J, Lee M, Chang HJ, Lee K, Choi JY. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In: Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV); 2019 Oct 27–Nov 2; Seoul, Republic of Korea. doi:10.1109/iccv.2019.00662.
40. Salehi A, Davulcu H. Graph attention autoencoders. arXiv:1905.10715. 2019.
41. Manzoor E, Milajerdi SM, Akoglu L. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 1996 Aug 2–4; Portland, OR, USA. doi:10.1145/2939672.2939783.
42. Han X, Pasquier T, Bates A, Mickens J, Seltzer M. Unicorn: runtime provenance-based detector for advanced persistent threats. In: Proceedings of the 2020 Network and Distributed System Security Symposium; 2020 Feb 23–26; San Diego, CA, USA. doi:10.14722/ndss.2020.24046.