SAURONEYES: Disentangling Voluminous Logs to Unveil Camouflaged Attack Intentions

Wei Qiao, Weiheng Wu, Song Liu, Yebo Feng, Zehui Wang, Junrong Liu, Teng Li, Bo Jiang, Zhigang Lu, and Baoxu Liu

***Abstract*—Advanced Persistent Threats (APTs) pose escalating risks to large enterprises and institutions. While current research has predominantly focused on data source analysis for identifying known attack patterns or anomalous behaviors, three critical challenges remain inadequately addressed: 1) APTs demonstrate sophisticated concealment capabilities, embedding malicious operations within legitimate business activities; 2) The sparse nature of APT attacks leads to low-frequency malicious activities that prove exceptionally challenging to detect within massive log datasets; 3) APTs employ multi-stage attack chains, whereas existing solutions exhibit limitations in reconstructing complete attack pathways to enable effective forensic analysis. In this paper, we address the detrimental effects of the sparsity of malevolent interactions and attack intent camouflaging on anomaly detection by introducing SAURONEYES, the pioneering APT detection system tailored to resolve these challenges. SAURONEYES constructs audit logs into both knowledge and interaction views, disentangling these to learn representations through graph learning enhanced with an attention-based neighbor allocation mechanism. Additionally, we incorporate self-supervised contrastive learning to discern the subtle similarities and distinctions among disentangled samples, facilitating a deeper understanding of the inherent structures within system interactions. SAURONEYES thus boasts heightened sensitivity and granular detection capabilities. Finally, SAURONEYES reconstructs the attack chain at the node level and presents an attack-chain that is more accessible for security analysis. Our evaluations in real-world scenarios and simulated attack environments demonstrate that SAURONEYES achieves outstanding accuracy, with an average detection rate of 99%.**

***Index Terms*—Advanced persistent threats (APT) detection, graph neural network (GNN), provenance graph, intrusion detection system (IDS).**

Received 2 April 2025; revised 2 August 2025, 31 August 2025, and 23 September 2025; accepted 24 September 2025. Date of publication 6 October 2025; date of current version 6 November 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFC2206402; in part by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA0460100; in part by the Open Foundation of Key Laboratory of Cyberspace Security, Ministry of Education of China under Grant KLCS20240206; in part by the Program of Key Laboratory of Network Assessment Technology; in part by Chinese Academy of Sciences; in part by the Program of Beijing Key Laboratory of Network Security and Protection Technology; in part by the National Natural Science Foundation of China under Grant 62272370; in part by the Natural Science Basic Research Program of Shaanxi under Grant 2025JC-JCQN-073; in part by the Young Elite Scientists Sponsorship Program by CAST under Grant 2022QNRC001; in part by the Qinchuangyuan Scientist + Engineer Team Program of Shaanxi under Grant 2024QCY-KXJ-149; and in part by the Songshan Laboratory under Grant 241110210200. The associate editor coordinating the review of this article and approving it for publication was Dr. Ming Li. *(Wei Qiao and Weiheng Wu contributed equally to this work.) (Corresponding authors: Junrong Liu; Teng Li.)*

Wei Qiao, Weiheng Wu, Song Liu, Zehui Wang, Junrong Liu, Bo Jiang, Zhigang Lu, and Baoxu Liu are with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100045, China, and also with the School of Cyber Security, University of Chinese Academy of Sciences, Beijing 101408, China (e-mail: qiaowei@iie.ac.cn; wuweiheng@iie.ac.cn; liusong@iie.ac.cn; wangzehui@iie.ac.cn; liujunrong@iie.ac.cn; jiangbo@iie.ac.cn; luzhigang@iie.ac.cn; liubaoxu@iie.ac.cn).

Yebo Feng is with Nanyang Technological University, Singapore 639798 (e-mail: yebo.feng@ntu.edu.sg).

Teng Li is with the School of Cyber Engineering, Xidian University, Xi'an 710126, China, and also with the Songshan Laboratory, Zhengzhou 450018, China (e-mail: litengxidian@gmail.com).

Digital Object Identifier 10.1109/TIFS.2025.3618381

I. INTRODUCTION

AN ADVANCED Persistent Threat (APT) is a long-term, targeted cyberattack conducted by skilled attackers who stealthily infiltrate a system, often through phishing or by exploiting vulnerabilities [1]. Once they gain access, they deploy advanced malware and backdoors to remain undetected while moving through the network to gain control [2]. APTs typically aim to steal sensitive data or disrupt operations, posing a severe threat to large organizations and government systems. Consequently, achieving full-cycle attack attribution has become an urgent need. However, unlike traditional attacks, APTs feature a multi-stage attack chain (e.g., reconnaissance, lateral movement, data exfiltration), making it difficult for conventional single-point event detection methods to establish a complete contextual relationship of the attack. Additionally, APTs incorporate disguised benign behaviors within their attack chain, exhibiting a high degree of stealth that renders detection through traditional feature-matching methods ineffective. Moreover, APTs have long dormancy periods and utilize low-frequency communication, causing traditional threshold-based anomaly detection models to fail. These characteristics of APTs present challenges that render conventional intrusion detection systems ineffective, as they struggle to address low-frequency and complex attack strategies, resulting in higher rates of false positives and false negatives. Therefore, there is a need to propose a high-accuracy APT detection solution that maintains a low false positive rate while consuming minimal resources without compromising system performance. In response, the research community has employed data provenance techniques for executing APT detection [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. These techniques transform system audit logs into provenance graphs, utilizing rich contextual information for fine-grained causal analysis to detect intrusions and deduce security vulnerabilities. Existing research partially relies on

1556-6021 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.

Rule-based methods [14], [15], [16], which employ typical or specific attack patterns to construct rules and match audit records with known attack rules. However, this approach heavily depends on prior knowledge and is costly to develop in environments with frequent new variations, making it challenging to detect common zero-day issues in APT scenarios. Alternatively, some studies utilize **Statistics-based methods** [17], [18], [19] to detect threats by mapping the rarity of elements from various sources in provenance graphs, quantifying the tendency for attacks. The limitation of this approach is its focus only on direct connections (low-order information) within the graphs, which leads to overlooking a vast amount of deeper intents hidden in complex system entity interactions. Most recent research has shifted towards **Learning-based methods** [6], [7], [8], [9], [10], [11], [12], [13]. These methods leverage various deep learning techniques to model APT behaviors or patterns, classify anomalous samples, and detect deviations, enabling the detection of unknown attacks.

Traditional non-learning methods (e.g., rule-based matching, statistical analysis) exhibit inherent limitations in modeling complex entity relationships and dynamic behavioral patterns due to reliance on predefined knowledge bases' completeness. While large language models enhance contextual reasoning via chain-of-thought mechanisms, their design paradigms fundamentally misalign with APT attribution requirements. Large models' data/compute demands clash with scarce APT samples and dynamic threats, while their opacity hinders pinpointing initial breaches and lateral movements, causing dual failures in path inference and forensic clarity. In contrast, graph neural network (GNN)-based provenance frameworks model network entities as heterogeneous provenance graph structures that explicitly preserve attack propagation topology and temporal-causal dependencies. This approach achieves optimal balance across detection accuracy, resource efficiency, and explainability.

While provenance graph-based graph learning methods have achieved promising results, they still face inherent challenges in handling APT attacks: 1) **Camouflage**: Attackers construct semantically coherent pseudo-normal contexts by blending malicious operations with massive benign behaviors (e.g., masquerading as legitimate process invocations or routine file access), causing traditional anomaly detection models to suffer from feature confusion; 2) **Low-frequency**: APT events exhibit extremely low signal-to-noise ratios, with malicious fragments (often $< 0.01\%$) concealed in long-term TB-scale operation logs—the extreme sparsity allows normal high-frequency nodes to dominate graph neural networks' neighborhood aggregation, while existing loss functions struggle to capture long-tailed malicious patterns under class imbalance ratios exceeding $10^4 : 1$; 3) **Multi-stage**: APT attacks follow multi-phase tactics spanning hundreds of entities and dozens of stages, yet current methods relying on fixed-hop subgraph sampling (e.g., 3-hop truncation) fail to model cross-stage long-range dependencies, ultimately hindering complete attack graph reconstruction.

To address these challenges, we propose SAURONEYES, an advanced APT detection framework integrating graph disentanglement algorithms and contrastive learning. It systematically addresses APT characteristics through three innovations: 1) A graph disentanglement algorithm that isolates interactive noise in multi-dimensional entity attributes while revealing latent behavioral intentions; 2) A multi-view contrastive learning mechanism applied to the disentangled graph, which enhances discriminative feature learning through dynamic positive/negative sampling, effectively reducing false positives while amplifying subtle attack patterns; 3) During attack detection, malicious interaction scores are used to distinguish between benign and malicious entities, and critical intents (attributes and relationships) among infected entities are preserved to intuitively reconstruct attack scenarios, thereby providing hierarchical forensic evidence for security analysts.

SauronEyes comprises four collaborative modules for efficient anomalous edge detection and attack chain reconstruction: (1) Graph Construction constructs an Interaction Knowledge Graph (KG) capturing entity attributes (IP, paths, parent-child relationships) and a Graph (IG) reflecting entity interactions (processes, files, sockets), initializing aspect embeddings through gate units coupled with specific aspects. (2) View Disentanglement employs an attention-based neighbor allocation mechanism into path-aware GNNs for KG disentanglement, while integrating an focused lightweight GCN (LightGCN) to disentangle IG. (3) Contrastive Learning enhances representation learning through intra-view and inter-view contrastive mechanisms, mitigating data sparsity. (4) Threat Detection & Attack Reconstruction predicts edge maliciousness via multi-aspect fusion, constructs compact summary graphs by considering overlapping attack behaviors across events.

We have implemented SAURONEYES and evaluated its effectiveness and efficiency on the DARPA Transparent Computing (TC) E3 dataset [20], StreamSpot dataset [21] and the Unicorn Wget dataset [6]. The DARPA dataset includes real-world attacks, while the StreamSpot and Unicorn dataset can simulate attacks in a controlled environment. Experimental results show that SAURONEYES can efficiently capture real attack intents and distinguish between benign and malicious system entity interactions with high precision and recall rates. Moreover, SAURONEYES matches or exceeds the performance of state-of-the-art detection methods. Finally, SAURONEYES accurately reconstructs attack chains.

In summary, we make the following contributions:

- We propose SAURONEYES, an APT detection method suitable for environments with high sparsity of malicious interactions, capable of detecting deep attack intents in massive log data. It also constructs an attack chain that is easier to analyze.
- We capture deep attack intentions by disentangling the camouflaged graphs and model system entity interaction tendencies through disentangled representation learning, thus preventing the detection model from being misled by surface intent-induced representations.
- We introduce a novel self-supervised contrastive learning framework that learns disentangled representations of system entities from knowledge views and interaction views and enhances inter-view and intra-view compar-

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.

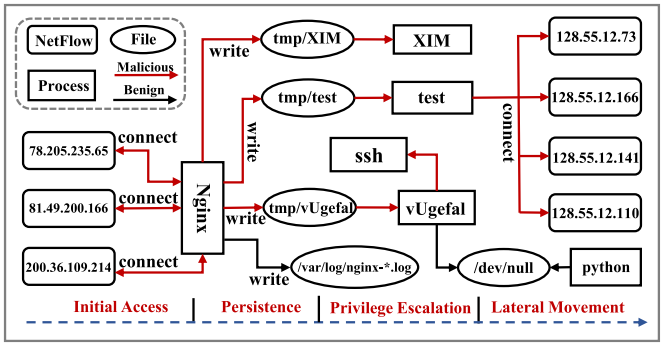


Fig. 1. Abstract diagram sourced from DARPA E3-CADETS, depicting a backdoor attack activity.

isons, providing an effective solution for the high sparsity of malicious entities in real-world log data.

- We conduct extensive experimental evaluations on widely used datasets, demonstrating the high effectivenes.

II. Background & Motivation

In this section, we use APT attacks from DARPA [20] to illustrate the challenges in existing threat detection solutions.

A. Motivating Example

As shown in Fig. 1, this is a real Nginx backdoor attack from the DARPA Transparent Computing Engagement 3 [20]. The attacker `81.49.200.166` launched two cyber attacks against the CADETS system by exploiting an Nginx backdoor. During the initial attack, the perpetrator successfully connected to the vulnerable `Nginx` server running on CADETS. Through shell commands, they downloaded a malicious payload to the `/tmp/vUgefal` directory and successfully escalated privileges for the `vUgefal` process to root level. After exfiltrating sensitive information, the attacker attempted to infect the `sshd` process with the malicious payload, which ultimately caused a system crash before completion. Subsequently, the attacker initiated a second assault by creating the malicious process `XIM` and establishing a persistent `test` process designed to maintain long-term connectivity while conducting port scanning operations across the local network `(128.5.12.*:*)`.

B. Attack Intent Camouflaging

Intent refers to the normal interactive behaviors of system entities themselves, as well as the objectives of malicious behaviors orchestrated by APT attackers [22], [23]. Attack Intent Camouflaging describes a critical detection challenge where an attacker deliberately mixes malicious operations with a large volume of legitimate, benign system activities. By embedding malicious behavior within normal workflows, the attacker's hostile intent becomes mixed with routine operational intents, effectively camouflaging the attack. This tactic makes it exceptionally difficult to discern the true purpose behind a series of actions. For instance, a single process might execute both legitimate and malicious tasks simultaneously, which obscures its overall intent from traditional detection systems. The Nginx backdoor attack in the DARPA dataset serves as a concrete example of Attack Intent Camouflaging.

TABLE I

Obstacles That Limit the Performance of Provenance-Based APT Detection Models. ● Indicate That They Can Solve or Have This Ability, While ○ Indicate the Opposite

Model	Handles Camouflaging	Data Sparsity	Attack Reconstruction	Without Prior Intelligence	Granularity
SauronEyes	●	●	●	●	Edge
POIROT [25]	○	○	●	○	Graph
ThreaTrace [10]	○	●	○	○	Node
MAGIC [24]	●	●	○	○	Node[1]
KAIROS [13]	○	○	●	○	Node[2]
FLASH [26]	○	○	●	●	Node
ShadeWatcher [12]	●	○	○	●	Edge
Unicorn [6]	○	○	○	●	Graph

[1] MAGIC claims to have both batch-level and system-entity-level granularity. This table records the granularity at a finer level.
[2] KAIROS detects the time window and locks the abnormal nodes in it. This table records at the node level.
[3] The mechanisms in other baseline methods listed in the table merely perform simple initialization for distinct node types, failing to effectively find camouflaging intents between different entities.

C. Challenge to Existing Solutions

Previous studies on threat detection have demonstrated remarkable performance in the field of APT detection. Nevertheless, persistent challenges hinder the effectiveness of provenance-based detection models when deployed in increasingly complex real-world environments. These critical limitations, which will be elaborated in subsequent sections, are systematically summarized in TABLE I.

1) Rule-Based Methods: Rule-based detection frameworks [14], [15], [16] incorporate expert knowledge to construct attack-matching patterns through known attack reports and meticulously designed attack chain scoring mechanisms based on ATT&CK. While this expert-driven approach effectively reduces false positive rates, the excessive reliance on manual expertise results in slow pattern updates when confronting sophisticated APT attack vectors and evolving cyberspace landscapes, ultimately leading to significant detection latency in rule-based methodologies.

2) Statistics-Based Methods: Statistical-based approaches [17], [18], [19] detect anomalies by calculating occurrence probabilities of system behaviors and assigning anomaly scores based on their rarity (operating under the assumption that rare entities are more likely to be malicious). However, these shallow statistical measurements fail to capture the genuine intent behind system behaviors. As demonstrated in our motivating example (Fig. 1), the `test` process conducting multiple port scans across different IP addresses would be misclassified as benign due to its high frequency, while overlooking its critical association with the malicious `Nginx` entity. Moreover, initial low-frequency benign activities like file writes by `python` often trigger false positives. These limitations originate from conventional statistical approaches' failure to holistically capture entity attributes and interaction semantics in provenance graphs.

3) Learning-Based Methods: Deep learning-based APT detection leverages large-scale system behavioral data for model training, yet faces fundamental constraints. KAIROS [13] adopts graph neural network encoder-decoder

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.

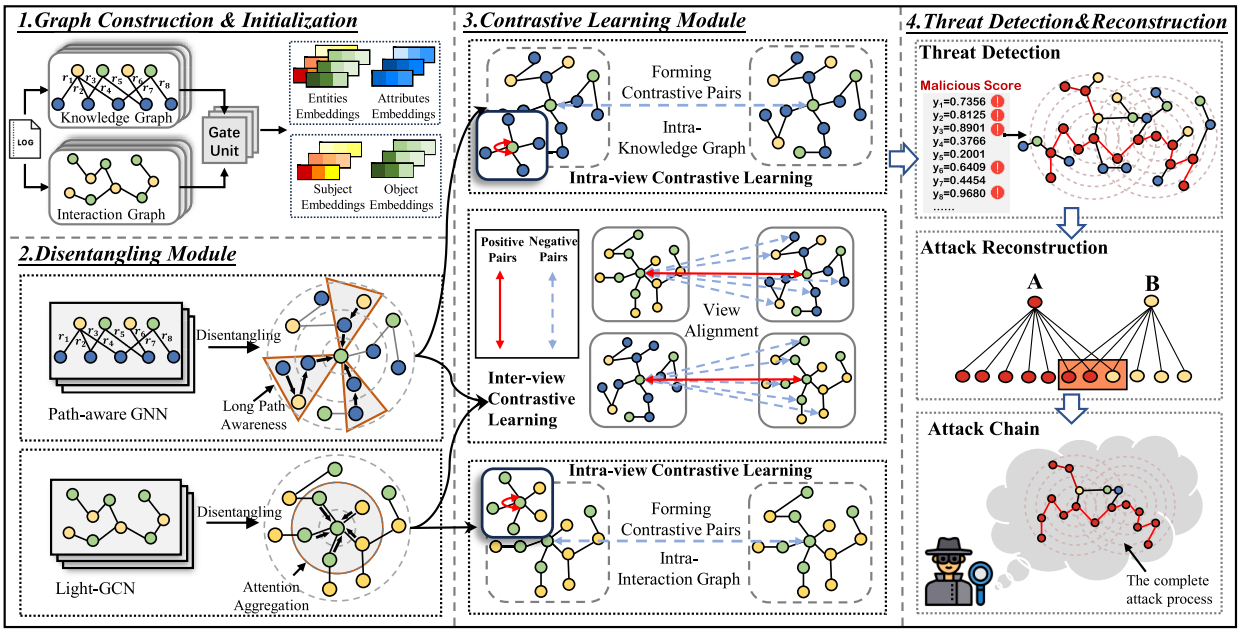


Fig. 2. The framework of SAURONEYES consists of: i) generating initial embeddings from knowledge graphs and interaction graphs; ii) dual-view disentangling by LightGCN and Path-aware GNN respectively; iii) contrastive learning within and between views and iv) threat detection and attack chain reconstruction.

architectures for attack chain reconstruction. UNICORN [6] constructs contextual provenance graphs for stealthy attack detection but suffers from coarse-grained alerting. SHADEWATCHER [12] reformulates APT detection as entity preference prediction via bipartite graph modeling, lacking full-chain attack reconstruction. MAGIC [24] employs masked graph representation learning to abstract benign system patterns, while THREATRACE [10] implements GraphSAGE-based entity aggregation for host-centric threat detection. Current approaches remain fundamentally constrained by severe data imbalance between benign logs and sparse APT patterns, with inadequate mitigation strategies exacerbating false positives.

III. THREAT MODEL AND DESIGN GOAL

A. Threat Model

We assume attackers infiltrate hosts via networks to steal data or manipulate devices, leaving reconstructable audit traces. Aligned with prior models [6], [16], [25], [27], our TCB includes OS kernels and audit frameworks, excluding hardware trojans/evasion attacks beyond kernel auditing [5], [28]. Audit data integrity is ensured via tamper-resistant logging [29], [30].

B. Design Goal

SAURONEYES models audit logs to extract behavioral patterns by disentangling entity attributes and relationship types, enabling precise identification of genuine behavioral intents across diverse entity activities. The system aims to achieve three key objectives for effective APT detection: (1) generating edge-grained granular alerts from massive log volumes to ensure fine-grained threat visibility; (2) maintaining low false positives through intent disentanglement to isolate malicious behaviors; and (3) providing complete alert graphs with contextualized relationships to streamline security analysts' alert verification and attack investigation processes.

IV. OVERVIEW

SAURONEYES is an innovative provenance-based APT detection scheme that utilizes graph disentangling algorithms and graph contrastive learning. It enables fine-grained identification of system anomalies in audit logs, providing a concise and complete attack chain. Fig. 2 illustrates the architecture of SAURONEYES, which consists of four major components:

① **Graph Construction and Initialization** For the host logs of different systems, SAURONEYES models system behavior by focusing on the interactions between system entities and the multiple information flows and attribute features of these entities. SAURONEYES extracts interaction relationships (e.g., Process → File) and entity attributes (e.g., Socket → IP segment) from the logs, which are then converted into two types of graph structures: knowledge graphs (KG) and interaction graphs (IG). The system entity features are initialized as vector embeddings through element-wise gating units.

② **Disentangling Module** once the initial embeddings are obtained, the disentangling module begins the process of disentangling the knowledge graph (KG) and the interaction graph (IG). For the KG, the module uses a path-aware GNN to encode the multi-faceted attribute information of individual system entities. An attention-based neighbor allocation mechanism is employed for each aspect. For the IG, the disentangling module focuses on separating each system entity's interactions with multiple entities (or aspects) and applies a lightweight graph convo-lutional network (LightGCN) for aggregation. Similarly, an attention-based mechanism is used to filter subsets of neighbors, aggregating more valuable neighbor attributes

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.

③ **Contrastive Learning Module** After obtaining the disentangled entity feature embeddings, the contrastive learning module applies within-view contrastive learning to the embeddings of KG and IG. This module introduces random noise into the embeddings to add perturbations for contrastive enhancement. For any system entity within a view, a pair of augmented embeddings under the same aspect is treated as a positive pair, while representations from different aspects form negative pairs. Once the positive and negative pairs are determined, contrastive loss is calculated separately for both views. Subsequently, the module performs contrastive learning between the two views to align the system entity representations within them.

④ **Threat Detection and Reconstruction** Leveraging multi-view embeddings, SAURONEYES calculates intent prediction scores for interactions by combining inner products across distinct intent dimensions of system entities. When an anomaly score exceeds a predefined threshold, the interaction is flagged as malicious and incorporated into the alert chain. To generate compact attack summary graphs for analysts, SAURONEYES employs overlapping community detection to automatically aggregate discrete alerts into coherent attack events, ensuring both interpretability and structural integrity of reconstructed attack scenarios.

V. DESIGN

In this section, we introduce SAURONEYES, whose purpose is to disentangle the knowledge graph and interaction graph and extract deep-seated intentions, then use contrastive learning to study the intrinsic structural information of entities, ultimately detecting malicious interactions hidden among system entity interactions. TABLE II provides detailed definitions of key concepts and parameters.

TABLE II
SUMMARY OF KEY NOTATIONS AND TERMS

Notation/Term	Description
Core Concepts	
View	A macro-level representation of the system including Interaction Graph and Knowledge Graph .
Aspect	A fine-grained latent factor or intent disentangled within each View.
Subject (s)	The initiator of an interaction, exclusively a process in our model.
Object (o)	The recipient or target of an interaction, which can be a file, socket, or another process.
Graphs and Entities	
IG	Interaction Graph, which models the dynamic, behavioral interactions between subjects and objects.
KG	Knowledge Graph, which models the static attributes of system entities.
E, V, R	Sets of entities, attribute values, and attribute relations in the KG.
e, s, o	A generic entity, a subject entity, and an object entity.
$\mathcal{N}_e$	The set of one-hop neighbors of entity e.
Embeddings and Model Parameters	
d	The dimension of embedding vectors.
K	The number of aspects to be disentangled.
e_e	The initial ID embedding for an entity e.
$e_{e,k}$	The initial embedding for entity e corresponding to the k-th aspect.
$e_{e,k}^{(l)}$	The embedding representation of entity e for the k-th aspect at the l-th GNN layer.
$x_{e,k}^{ig}, x_{e,k}^{kg}$	The final representation of entity e for the k-th aspect, learned from IG and KG respectively.
W_k, b_k	Learnable weight matrix and bias vector for the k-th aspect in the gating unit.
α	Attention score used in neighbor aggregation.
τ	Temperature parameter in the contrastive loss function.
$\lambda_1, \lambda_2, \lambda_3$	Hyperparameters controlling the weights of contrastive losses and L2 regularization.

TABLE III
SYSTEM BEHAVIORS EXTRACTED FROM AUDIT LOGS

System Behavior	Relation Description
Process → R1 → Process	"R1": "fork", "execute", "exit", "clone", etc.
Process → R2 → File	"R2": "read", "open", "close", "write", etc.
Process → R3 → Netflow	"R3": "connect", "send", "recv", "write", etc.
Process → R4 → Memory	"R4": "read", "mprotect", "mmap", etc.

A. Graph Construction and Initialization

1) Graph Construction: SauronEyes extracts system entities including subjects (processes), objects (processes, files, network flows, etc.), and attributes (IP addresses, command-line arguments) from audit logs (TABLE III). After we obtain these four types of entities relationships, we will map the necessary behavioral relationships into attribute relationships between nodes in the KG graph to assist in building attribute edges. The knowledge graph is defined as a set of triples: $KG = \{(e, r, v) \mid e \in E,\ r \in R,\ v \in V\}$, where E denotes system entities(including subject and object), R represents the relationship between entities and attribute values. Subsequently, we enrich the graph by embedding entity attributes and interaction relationships. We abstract the interactions between system entity pairs with causal relationships into a unified interaction graph. Each interaction reflects a causal exchange between two system entities within a specific timestamp. The interaction graph is formally defined as: $IG = \{(s, o) \mid s \in \text{Subject},\ o \in \text{Object}\}$, where each pair (s, o) indicates that subject s has interacted with object o.

2) Embeddings Initialization: Prior to graph disentangling, we initialize multi-aspect embeddings for entities. Formally, we assume there are K distinct aspects. Unlike methods that partition ID embeddings into K chunks [31], we employ element-wise self-gating units to regulate information flow from ID embeddings to each aspect:

$$\mathbf{e}_{e,k} = f_{\text{gate}}^{k}(\mathbf{e}_e) = \mathbf{e}_e \odot \sigma(\mathbf{W}_k \mathbf{e}_e + \mathbf{b}_k), \tag{1}$$

where $\mathbf{e}_e$ is the ID embedding of entities e, $\mathbf{e}_{e,k}$ is the initial embedding of entities e for the k-th aspect, $\mathbf{W}_k \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_k \in \mathbb{R}^d$ are learnable parameters, $\odot$ denotes element-wise multiplication, and σ is the sigmoid function. Similarly, we initialize aspect-specific embeddings $\mathbf{e}_{s,k}$, $\mathbf{e}_{o,k}$, $\mathbf{e}_{r,k}$ and $\mathbf{e}_{v,k}$ for subject s, object o, relations r and attribute values v, respectively.

The self-gating mechanism adaptively learns non-linear gates to modulate ID embeddings across different aspects at element-wise granularity through dimension re-weighting.

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.

This approach offers greater flexibility compared to naively partitioning embeddings into fixed chunks.

B. Disentangling Module

After constructing the audit logs into KG and IG, and initializing the embeddings through gating units, we explore the disentangling of the two views(KG and IG) in the disentangling module. This process differentiates the various aspects between system entity interactions and system entity attributes, aiming to obtain embeddings with clearer expressive power.

1) Disentangling Knowledge Graph: The purpose of this component is to learn disentangled knowledge-aware representations that distinguish various aspects of system entity attributes. For example, in the Knowledge Graph (KG), for the entity node `Nginx`, its connected node `200.36.109.214` reflects IP information, `/var/log/nginx-*.log` indicates storage path information, and `/etc/group` represents the Received objects of `Nginx`. Inspired by [32], we use a path-aware GNN to encode this information. The path-aware GNN aggregates L-hop neighborhood information through deep aggregation while preserving path information (i.e., long-range connectivity).

However, in KG disentanglement tasks, we cannot aggregate all neighbors because only a subset of neighbors is highly relevant to their corresponding aspects. For instance, the IP information `200.36.109.214` directly caused the compromise of `Nginx` and is strongly correlated with the "attack source" aspect, while `/var/log/nginx-*.log` and `/etc/group` are weakly relevant. To better capture the affinity between target entities and their neighbors, we choose similarity-based attention with the assumption that the more similar entity node e is to its neighbor v in the k-th aspect, and the relation is r, the better the representation of node e in that aspect. The attention score is represented as

$$\alpha^k_{(e,r,v)} = \frac{\exp(\mathbf{e}^\top_{e,k}(\mathbf{e}_{r,k} \odot \mathbf{e}_{v,k}))}{\sum_{k' \in K} \exp(\mathbf{e}^\top_{e,k'}(\mathbf{e}_{r,k'} \odot \mathbf{e}_{v,k'}))}. \tag{2}$$

Then, the aggregation at the l-th layer of the k-th aspect can be represented as:

$$\mathbf{e}^{(l+1)}_{e,k} = \frac{1}{\left|\mathcal{N}^{kg}_e\right|} \sum_{(r,v) \in \mathcal{N}^{kg}_e} \alpha^k_{(e,r,v)} \mathbf{e}_{r,k} \odot \mathbf{e}^{(l)}_{v,k}, \tag{3}$$

l specifically refers to the l-th convolutional layer of the GNN, which is conceptually equivalent to the l-hop neighborhood from the central node. Finally, we aggregate the representations from all layers to obtain the knowledge attribute representation at the k-th aspect:

$$\mathbf{x}^{kg}_{e,k} = \mathbf{e}^{(0)}_{e,k} + \cdots + \mathbf{e}^{(L)}_{e,k}. \tag{4}$$

where $e^{(0)}_{e,k}$ represent the initial embeddings of system entities e at the k-th aspect.

2) Disentangling Interaction Graph: The purpose of this component is to learn multi-aspect representations of system entity interactions. The Interaction Graph (IG) emphasizes collaborative signals between entities. For instance, in the interaction sequence `Nginx` → `tmp/test` (and → `/etc/passwd` and → `tmp/vUgefal`), four entities participate in three interactions, where capturing both direct and transitive collaboration patterns is critical. We obtain collaborative information by modeling these connections.

We adopt LightGCN, which employs simple message passing and aggregation mechanisms without feature transformation or nonlinear activation, thereby improving computational efficiency in graph structures constructed from real-world audit logs. However, similar to KG disentanglement, we observe that for key nodes equivalent to "traffic hubs" (e.g., `Nginx`), they may engage in interactions across multiple distinct aspects. Aggregating all neighbors under a single aspect during disentanglement is clearly suboptimal.

For example, as previously mentioned, the relationships and entities in the attack provenance aspect are the primary causes of the `Nginx` server's operations and associations with `/tmp/vUgefal`, while interactions from other aspects are not. Therefore, we utilize an attention-based neighbor allocation mechanism to refine the IG by inferring the importance of each interaction under different aspects. For the interaction (s, o) under the k-th aspect, its attention score is calculated as

$$\alpha^k_{(s,o)} = \frac{\exp(\mathbf{e}^\top_{s,k}\mathbf{x}^{kg}_{o,k})}{\sum_{k' \in K} \exp(\mathbf{e}^\top_{s,k'}\mathbf{x}^{kg}_{o,k'})}. \tag{5}$$

In the l-th layer at the k-th aspect, the aggregation result can be represented as:

$$\begin{aligned} \mathbf{e}^{(l+1)}_{s,k} &= \frac{1}{\left|\mathcal{N}^{ig}_s\right|} \sum_{o \in \mathcal{N}^{ig}_s} \alpha^k_{(s,o)} \mathbf{e}^{(l)}_{o,k}, \\ \mathbf{e}^{(l+1)}_{o,k} &= \frac{1}{\left|\mathcal{N}^{ig}_o\right|} \sum_{s \in \mathcal{N}^{ig}_o} \alpha^k_{(s,o)} \mathbf{e}^{(l)}_{s,k}, \end{aligned} \tag{6}$$

where N^{ig}_s and N^{ig}_o represent sets of system entities s's and o's neighbors in the interaction graph. Then representations at different layers are summed up as the collaborative representations of the k-th aspect, as follows:

$$\mathbf{x}^{ig}_{s,k} = \mathbf{e}^{(0)}_{s,k} + \cdots + \mathbf{e}^{(L)}_{s,k}, \quad \mathbf{x}^{ig}_{o,k} = \mathbf{e}^{(0)}_{o,k} + \cdots + \mathbf{e}^{(L)}_{o,k}, \tag{7}$$

where where $e^{(0)}_{s,k}$ and $e^{(0)}_{o,k}$ represent the initial embeddings of system entities s and o at the k-th aspect.

C. Contrastive Learning Module

Contrastive learning has enhanced representation learning by minimizing the distance between similar sample pairs and maximizing the distance between dissimilar sample pairs, addressing the issue of data sparsity caused by the scarcity of key samples. SAURONEYES does not require labels at all in the contrastive learning stage, but automatically generates pseudo labels based on the structure of the data itself.

1) Embedding Augmentation: It has been demonstrated in [33] that, for graph contrastive learning, the key factor is the contrastive loss (CL), rather than the augmentation of the graph itself. Therefore, this component designs a random noise-based embedding augmentation method, adding perturbations to the node embeddings for contrastive learning. Specifically, noise is randomly added to the disentangled embeddings at the k-th aspect. We add noise to each embedding in the opposite direction, i.e., the two augmented

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.

embeddings are slightly stretched in opposite directions and slightly shifted, which is expected to improve the linear separability in the projection space. A set of augmented embeddings for the k-th aspect can be represented as

$$f(e) = \begin{cases} {}^1e = e + (\|{}^1e_{rand}\|_2 \times \eta) \odot sign(e) \\ {}^2e = e - (\|{}^2e_{rand}\|_2 \times \eta) \odot sign(e) \end{cases} \tag{8}$$

where $\|\cdot\|_2$ represents L2 normalization, ${}^1e_{rand}$ and ${}^2e_{rand}$ are random embeddings that are uniformly distributed on [0, 1). By normalizing ${}^1e_{rand}$ and ${}^2e_{rand}$, we can convert them into unit length in the projection space. We set a hyperparameter η to control the intensity of the embedding augmentation, to prevent excessive noise from making the model unable to differentiate between similar samples.

2) Intra-View Contrastive Learning: Our goal is to explore the weak dependencies between the disentangled representations. We aim to achieve comprehensive disentangling to obtain more information. We utilize contrastive learning between the disentangled Kg and IG to perform independent representation learning. The steps are as follows: First, we define positive and negative sample pairs. For any embedding in the view, we select a pair of augmented embeddings from the same aspect to form a positive pair and calculate their self-similarity, then choose augmented embeddings from two different aspects to form a negative pair. We use the optimized contrastive loss InfoNCE, which learns more unified entity representations and implicitly alleviates popularity bias. The contrastive loss for the disentangled Knowledge graph can be expressed as:

$$\mathcal{L}^{kg}_{\text{intra}} = \sum_{v\in E,V} \sum_{k\in K} -\log \frac{e^{s(z^{kg}_{v,k}, z^{kg}_{v,k})/\tau}}{\sum_{k'\in K} e^{s(z^{kg}_{v,k}, z^{kg}_{v,k'})/\tau}}, \tag{9}$$

where s(·) represents the calculation of cosine similarity, and τ represents the temperature parameter. In a similar manner, we can derive the contrastive loss for the interaction graph as follows:

$$\mathcal{L}^{ig}_{\text{intra}} = \sum_{n\in s,o} \sum_{k\in K} -\log \frac{e^{s(z^{ig}_{n,k}, z^{ig}_{n,k})/\tau}}{\sum_{k'\in K} e^{s(z^{ig}_{n,k}, z^{ig}_{n,k'})/\tau}}. \tag{10}$$

The total intra-view contrastive loss is the sum of the two contrastive losses mentioned above:

$$\mathcal{L}_{intra} = \mathcal{L}^{kg}_{intra} + \mathcal{L}^{ig}_{intra}. \tag{11}$$

Through this learning, we achieve independent contrastive learning within each of the two views.

3) Inter-View Contrastive Learning: We also aim to investigate the relationship between the KG and IG views. Specifically, our goal is to align the embeddings of IG and KG and perform contrastive learning between their representations. For an entity in one view (e.g., Process, File, or Socket), we extract the embeddings of the same entity learned in the other view as a positive sample pair, and select the embeddings of different entities from the other view as negative sample pairs. The inter-view contrastive loss can be expressed as

$$\mathcal{L}_{\text{inter}} = \sum_{i\in E} \sum_{k\in K} -\log \frac{e^{s(z^{kg}_{i,k}, z^{ig}_{i,k})/\tau}}{\sum_{i'\in E}(e^{s(z^{kg}_{i',k}, z^{ig}_{i,k})/\tau} + e^{s(z^{kg}_{i,k}, z^{ig}_{i',k})/\tau})} \tag{12}$$

D. Threat Detection and Attack Reconstruction Module

1) Threat Detection: After obtaining the contrastive loss for disentangled representations on both the knowledge graph and interaction graph views, we proceed to threat prediction, specifically categorizing system entity interactions into benign and malicious interactions. Given any interaction (u,r,v) with k-th dimension, we combine the embeddings ($z_{u,k}$ and $z_{v,k}$) from both views and apply the inner product to represent the prediction score $\hat{y}^k(u,v)$ for the interaction between system entities u and v being malicious. The specific formula is as follows:

$$z_{s,k} = z^{ig}_{s,k}, \quad z_{o,k} = z^{kg}_{o,k} \oplus z^{ig}_{o,k} \tag{13}$$

$$\hat{y}^k(s,o) = (z_{s,k})^T z_{o,k}. \tag{14}$$

Considering that each interaction may be related to different aspects differently, we adopt a careful fusion of prediction scores from different aspects with $\beta^k_{(u,v)}$ as the attention weight to obtain the final result $\hat{y}(u,v)$, as follows:

$$\beta^k_{(s,o)} = \frac{\exp((h_s \oplus h_o)^T (z_{s,k} \oplus z_{o,k}))}{\sum_{k'\in K} \exp((h_s \oplus h_o)^T (z_{s,k'} \oplus z_{o,k'}))} \tag{15}$$

$$\hat{y}(s,o) = \sum_{k\in K} \beta^k_{(s,o)} \hat{y}^k(s,o). \tag{16}$$

Given the focus of the simple contrastive loss on the intrinsic distribution structure of the sample space, we apply a multi-task learning strategy to jointly train the pairwise BPR loss and the contrastive loss, in order to more comprehensively achieve system entity prediction. As shown below:

$$\mathcal{L}_{BPR} = -\frac{1}{|R|} \sum_{(s,o,s',o')\in E} \ln Sigmoid(\hat{y}_{(s,o)} - \hat{y}_{(s',o')}), \tag{17}$$

$\hat{y}_{(s,o)}$ and $\hat{y}_{(s',o')}$ are both prediction scores, where (s', o') corresponds to a different pair of nodes from (s, o), R represents the set of interactions sampled in each mini-batch.

By combining intra-view and inter-view contrastive losses with BPR loss, we minimize the following objective function to learn the model parameters:

$$\mathcal{L}_{final} = \mathcal{L}_{BPR} + \lambda_1 \mathcal{L}_{intra} + \lambda_2 \mathcal{L}_{inter} + \lambda_3 \|\Theta\|^2_2. \tag{18}$$

Here, θ represents the set of model parameters, λ_1 and λ_2 are hyperparameters controlling the weights of the intra-view and inter-view contrastive losses, respectively, λ_3 is the hyperparameter controlling the L2 regularization term.

2) Attack Reconstruction: In the process of intent disentangling and threat detection, SAURONEYES extracts comprehensive attack intents and behaviors. However, the resulting anomaly graph remains a complex and large node graph. To further reduce the burden on security analysts, SAURONEYES implements attack chain construction. Unlike existing approaches that use the Louvain algorithm [7], [13], [34], we considered the phenomenon of overlapping communities in attack division. As shown in Fig. 4, when nodes belong to multiple clusters simultaneously, communities in the network overlap, and some behaviors may contribute to different attack communities. The Louvain algorithm, based on modularity partitioning, cannot detect this issue. We devised an overlapping community detection method based

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.

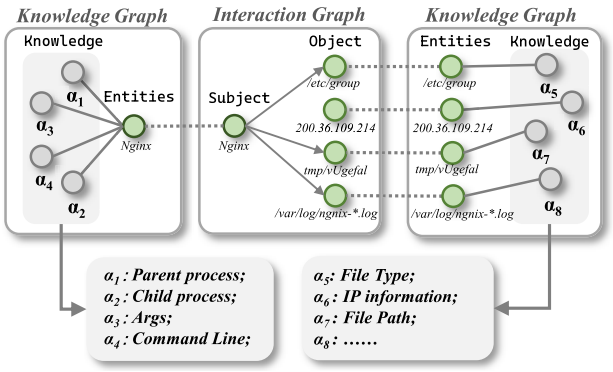


Fig. 3. An example shows the composition of knowledge graph and interaction graph. It can be seen that knowledge graph has multi-faceted attribute knowledge.

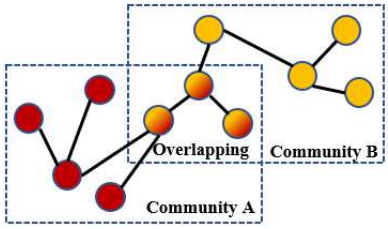


Fig. 4. The Overlapping Phenomenon of Attack Clusters.

on Non-negative Matrix Factorization (NMF) [35], which explicitly models the membership strength of each node to each community. We assign a non-negative latent factor to each node-community pair, indicating the degree of the node's affiliation with the community. The probability of an edge existing between a pair of nodes in the network is then modeled as a function of their shared community affiliations. Based on the node affiliation model, community structures are identified by maximizing the affiliation strength between nodes.

To accomplish this, SAURONEYES extracts a subgraph G_p containing malicious activities (edges) from the interaction graph, where G_p serves as the input graph for attack reconstruction. The community detection in BIGCLAM [35] is performed by leveraging the affiliation strength A_{uv} between nodes u and v.

$$A_{uv} = F_u \cdot F_v = \sum_{k=1}^{K} F_{uk} \times F_{vk} \tag{19}$$

Here, F_u and F_v are the community attribute vectors of nodes u and v, respectively. By maximizing the affiliation strength between node pairs while considering the actual connections in the network, overlapping community partitioning is achieved. The optimization function is:

$$\sum_{(u,v)\in E} \log\left(1 - \exp(-A_{uv})\right) - \sum_{(u,v)\notin E} A_{uv}. \tag{20}$$

To the best of our knowledge, SAURONEYES is the first to address the issue of overlapping attack communities. For instance, Hercule [7] and Kairos [13] also utilize community detection but do not consider that attack nodes may simultaneously belong to multiple anomalous behavior communities.

VI. EVALUATION

We assessed SAURONEYES from the following Seven perspectives:

RQ1: How effective is SAURONEYES in detecting APTs compared to current state-of-the-art appraoches?

RQ2: Are the key components in our SAURONEYES framework really improving the overall performance?

RQ3: How do hyperparameters affect SAURONEYES' efficacy?

RQ4: What is the performance overhead of SAURONEYES when handling large-scale data in real-world scenarios?

RQ5: Can SAURONEYES accurately reconstruct attack behavior from the original provenance graph?

RQ6: How resilient is SAURONEYES to adversarial attacks?

RQ7: How is the generalization ability of SAURONEYES when confronting concept drift?

A. Experimental Settings

1) Datasets: SAURONEYES was evaluated using three authoritative open-source datasets: The **StreamSpot** dataset [1] employs six experimental scenarios (five benign, one drive-by download attack) to assess rapid attack detection. The **Unicorn Wget** dataset [2] contains 150 CamFlow logs (125 benign/25 covert supply-chain attacks) simulating kill-chain obfuscation tactics, challenging systems to discern malicious patterns mimicking legitimate workflows. The **DARPA-E3** dataset [3] from real adversarial engagements combines APT activities with benign operations in an enterprise network, reflecting Advanced Persistent Threats' stealth characteristics. Their complementary designs – controlled experiments (StreamSpot/Unicorn) and operational realism (DARPA-E3) – ensure comprehensive evaluation of detection accuracy, anti-evasion robustness, and practical applicability.

2) Metrics: (1) To evaluate SauronEyes' anomaly detection capability, we use two evaluation metrics based on IDS ThreaTrace [10] and Unicorn [6]. We perform graph-level and node-level anomaly detection on both system entity logs and batch logs. We use Accuracy, Precision, Recall and F1-score for evaluation. (2) To evaluate attack community overlap, we use Normalized Mutual Information (NMI) to measure the similarity of community partitions and ensure the algorithm accurately reflects the true structure. The Jaccard similarity coefficient assesses node set overlap, while the F1 score measures precision and recall. This provides a comprehensive evaluation of SAURONEYES' performance in identifying communities and avoiding errors.

The Jaccard similarity coefficient is defined as:

$$J(U,V) = \frac{|U \cap V|}{|U \cup V|} \tag{21}$$

where U represents the groundtruth community and V represents the detected partitioned community. NMI is defined as

$$NMI(U,V) = \frac{2 \cdot MI(U,V)}{H(U) + H(V)} \tag{22}$$

where $MI(U,V)$ is the mutual information and $H(U)$, $H(V)$ are the entropies.

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.

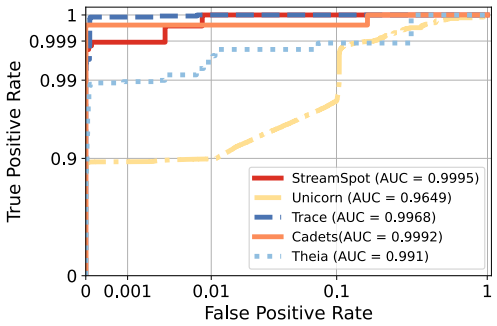


Fig. 5. ROC curves on all datasets.

TABLE IV
OVERVIEW OF THE ROW EXPERIMENTAL DATASETS

Dataset	of Nodes	of Edges in millions	Attack batch/node	Size (GB)
StreamSpot	999,999	89.8	100 batches	2.8
Unicorn Wget	39,606,900	145.9	25 batches	76.6
DARPA E3-Trace	3,288,676	4	68,082 nodes	15
DARPA E3-Cadets	1,627,035	2.8	12,846 nodes	18
DARPA E3-Theia	1,623,966	3.3	25,319 nodes	18

3) Implementation: We implemented the SAURONEYES prototype using Python 3.11. We used PyTorch [36] to implement the model. For the hyperparameters in SAURONEYES, we set all embedding dimensions to 64, the learning rate to 0.001, and the L2 regularization parameter to 0.0001. We chose a batch size of 1024 for log detection in our experiments. We used the Adam [37] optimizer to optimize the model. While these hyperparameters might have other potential choices, in Section VI-D, we discuss the impact of hyperparameter settings on the final performance of SAURONEYES. All experiments were performed on a server running Ubuntu 22.04.5 LTS with an Intel(R) Xeon(R) Silver 4208 CPU (14 cores, 14 threads, base frequency 2.10 GHz) and 256 GB of RAM.

B. Detection Performance (RQ1)

1) Detection Results: Experimental results demonstrate that SAURONEYES achieves robust performance across diverse operational scenarios, with comprehensive evaluations conducted on both batch processing logs and entity-level system logs (TABLE V). Furthermore, the ROC curves presented in Fig. 5 validate the model's superior classification accuracy under varying operational conditions, confirming its adaptability and detection effectiveness in heterogeneous logging environments.

2) SauronEyes Vs. State-of-the-Art: In batch log analysis, SAURONEYES achieved near-perfect detection on the Streamspot dataset with simple attack patterns, while conventional graph-level baselines suffered notable degradation on the complex, highly camouflaged attacks in Unicorn Wget. Though node-level methods like Threatrace maintained decent accuracy, SAURONEYES reduced false positives significantly via edge-level intent-disentanglement detection. This approach

TABLE V
COMPARISON OF DETECTION RESULTS BETWEEN SAURONEYES AND OTHER STATE-OF-THE-ART SOLUTIONS

Datasets	System	Accuracy	Precison	Recall	F1-Score
StreamSpot	StreamSpot	93%▼6.0%	73%▼26.3%	91%▼8.1%	81%▼18.2%
	Unicorn	99%▼0.0%	95%▼4.0%	97%▼2.02%	96%▼3.0%
	ThreaTrace	99%▼0.0%	98%▼1.0%	99%▼0.0%	98%▼1.0%
	SAURONEYES	99%▲2.06%[1]	99%▲11.7%	99%▲3.5%	99%▲7.6%
Unicorn Wget	Unicorn	90%▼9.1%	86%▼10.4%	95%▼4.0%	90%▼7.2%
	Prov-Gem	97%▼2.0%	100%▲4.2%	80%▼19.2%	89%▼8.3%
	ThreaTrace	95%▼4.0%	93%▼3.1%	98%▼1.0%	95%▼2.1%
	SAURONEYES	99%▲5.3%	96%▲3.2%	99%▲8.8%	97%▲6.2%
DARPA E3 TRACE	Log2vec	97%▼3.0%	54%▼42.6%	78%▼21.2%	64%▼33.3%
	DeepLog	96%▼4.0%	41%▼56.4%	68%▼31.3%	51%▼46.9%
	ThreaTrace	98%▼2.0%	72%▼23.4%	99%▼0.0%	83%▼13.5%
	FLASH	99%▼1.0%	95%▲1.1%	99%▼0.0%	97%▲1.0%
	SAURONEYES	100%▲2.5%	94%▲43.5%	99%▲15.1%	96%▲30.2%
DARPA E3 CADETS	Log2vec	98%▼1.0%	49%▼47.9%	85%▼14.1%	62%▼35.4%
	DeepLog	95%▼4.0%	23%▼75.5%	74%▼25.3%	35%▼63.5%
	ThreaTrace	99%▼0.0%	90%▼4.3%	99%▼0.0%	95%▼1.0%
	FLASH	99%▼0.0%	92%▼2.1%	99%▼0.0%	96%▼0.0%
	OMNISEC[2]	99%▼0.0%	98%▲4.0%	94%▼5.1%	96%▼0.0%
	SAURONEYES	99%▲1.0%	94%▲33.5%	99%▲9.8%	96%▲25.0%
DARPA E3 THEIA	Log2vec	99%▼0.0%	62%▼35.4%	66%▼34.0%	64%▼34.7%
	DeepLog	98%▼1.0%	16%▼83.3%	14%▼86.0%	15%▼84.7%
	ThreaTrace	99%▼0.0%	87%▼9.4%	99%▼1.0%	93%▼5.1%
	FLASH	99%▼0.0%	92%▼4.2%	99%▼1.0%	96%▼2.0%
	OMNISEC	99%▼0.0%	100%▲4.0%	95%▼5.0%	97%▼1.0%
	SAURONEYES	99%▲0.2%	96%▲24.6%	100%▲25.4%	98%▲25.0%

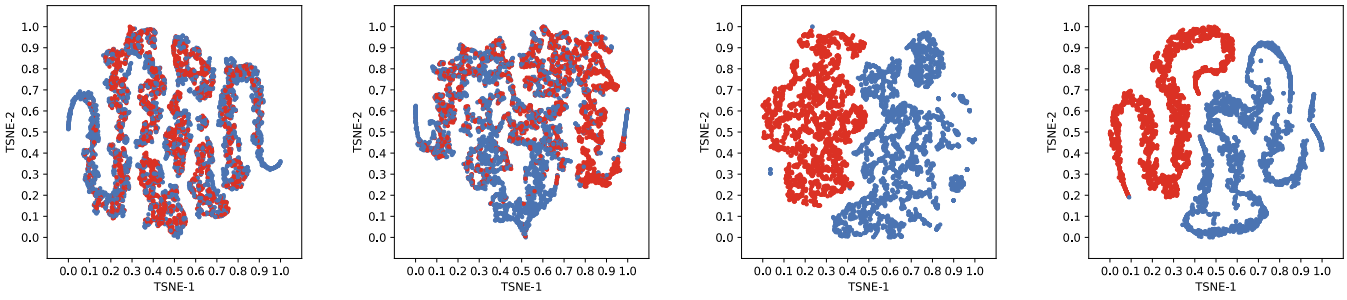
▼% represents the percentage of reduction, ▲% represents the percentage of improvement.
[1] Improvement compared to the average of all detection systems except SAURONEYES.
[2] OMNISEC incorporates expert knowledge in the prompt.

separates malicious behaviors from benign activities at edge granularity, enhancing stealthy attack identification. For entity-level log in DARPA E3, Log2vec detects anomalies in logs within graphs using node embeddings and clustering methods, aggregating information from node neighbors. However, Log2vec's nodes are logs rather than system entities, making it naturally inferior to provenance graph-based detection. DeepLog, being a log-level detector, cannot capture the contextual information in system entity interactions, making it difficult to detect anomalies within these interactions. As node-level detector, ThreaTrace and FLASH are comparable to SauronEyes in all aspects, but is significantly inferior to precision and F1-score in THEIA. Compared to the latest LLM-integrated prompt engineering solutions, although OMNISEC [38] incorporates expert knowledge to achieve lower false positives, its recall rate is significantly lower than SAURONEYES. Its missed detections of attacks could make the system more vulnerable to being compromised by attackers.

C. Ablation Study (RQ2)

In this section, we discuss the effectiveness of individual components in SAURONEYES and investigate whether removing or modifying these components impacts the final performance. The performance results are shown in Fig. 7. In Fig. 6, we select logs around the attack periods from E3-CADETS and used t-SNE to plot 2D visualized node representations after different disentanglement scenarios in the ablation experiments of the disentanglement module.

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.



(a) Method: GCN + MLP (b) Method: GCN + IG-Disentangling (c) Method: KG-Disentangling + MLP (d) Method: SAURONEYES

Fig. 6. 2D visualization of node representations on E3-CADETS using t-SNE. Red nodes indicate anomalous entities, while blue nodes indicate benign entities.

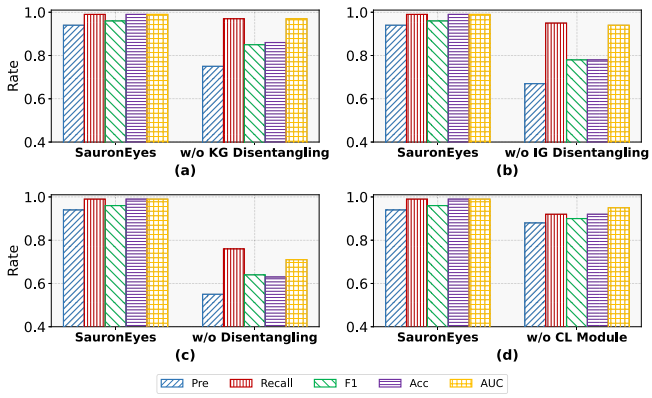


Fig. 7. Effects of Different Modules on the Detection Performance of SAURONEYES.

TABLE VI
COMPARISON OF DETECTION RESULTS BETWEEN **SAURONEYES (ORIGINAL)** AND **SAURONEYES (ABLATION)**

Datasets	Methods[1]	Accuracy	Precision	Recall	F1-Score
StreamSpot	SE(original)	99%	99%	99%	99%
	SE(ablation)	92%▼7.1%	84%▼15.2%	85%▼14.1%	84%▼15.2%
Unicorn Wget	SE(original)	99%	96%	99%	97%
	SE(ablation)	90%▼9.1%	85%▼11.5%	89%▼10.1%	87%▼10.3%
DARPA E3 TRACE	SE(original)	100%	94%	99%	96%
	SE(ablation)	87%▼13.0%	81%▼13.8%	88%▼11.1%	84%▼12.5%
DARPA E3 CADETS	SE(original)	99%	94%	99%	96%
	SE(ablation)	90%▼9.1%	77%▼18.1%	87%▼12.1%	81%▼15.6%
DARPA E3 THEIA	SE(original)	99%	96%	100%	98%
	SE(ablation)	82%▼17.2%	74%▼22.9%	80%▼20.0%	76%▼22.4%

▼% represents the percentage of reduction.
[1] We use the same attention parameters as the SE(original) and perform intra-view contrastive learning to control the variance.

1) Removing Disentanglement: To validate the necessity of disentanglement mechanisms, we conducted ablation studies by replacing key modules in the knowledge and interaction views(see Fig. 7). For knowledge disentanglement, removing the module and substituting it with a Graph Convolutional Network (GCN) caused precision to drop to 75%. This phenomenon suggests that GCN alone fails to deeply capture the intricate relationships among system calls, impeding the model's ability to uncover malicious behavioral logic and resulting in frequent false positives. For interaction disentanglement, replacing the module with a Multilayer Perceptron (MLP) led to a more pronounced degradation (67% precision), underscoring the critical role of disentangling dynamic interaction patterns. When all decoupling modules were replaced with the traditional scheme, the final model performance significantly declined, with Precision at only 55%. This occurs because GCN+MLP encoding fails to disentangle entities' multifaceted intentions, allowing misleading signals to distort their authentic behavioral patterns.

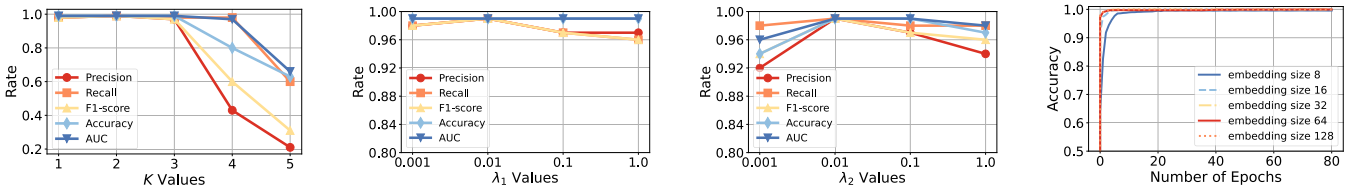
2) Removing Contrastive Learning Module: We removed the contrastive loss component within and between views. From Fig. 7, it can be seen that removing this module also reduced detection performance. This is because the two views not only lost the self-supervised ability to capture the structural relationships of system entities within themselves but also could not transfer information between the views.

3) Removing Multi-View: To study the significance of the respective disentangling operations and dual-view contrastive learning in dual views, we replaced the original SAURONEYES design with a combination of 'PG construction + PG disentanglement + contrastive learning + threat detection'. The decoupling operation is modified to aggregate PG using LightGCN with a multi-attention mechanism, and contrastive learning is to upsample positive and negative sample pairs from different aspects of the same node on PG. As shown in TABLE VI, the ablated version of SauronEyes has a significant performance degradation on various datasets, which reveals the importance of multi-view decoupling and contrastive learning in revealing the deep intentions of APT.

D. Hyperparameter Impact on Performance (RQ3)

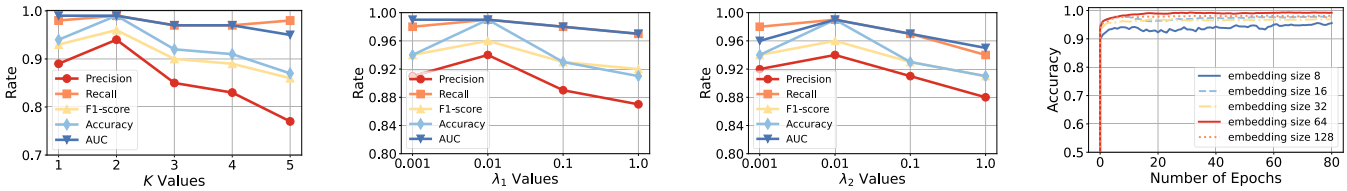
In this section, we independently modify specific hyperparameters to reflect their impact on the detection of APTs by SAURONEYES and provide interpretable explanations for these impacts. We selected StreamSpot at the batch log level (Fig. 8)

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.



a. K b. λ_1 c. λ_2 d. *embedding*

Fig. 8. Impact of different hyperparameters on detection performance on StreamSpot(Batch-level log).



a. K b. λ_1 c. λ_2 d. *embedding*

Fig. 9. Impact of Hyperparameter Variations on Detection Performance in DARPA E3-CADETS(Entity-level log).

and DARPA E3-CADETS at the system entity level (Fig. 9) as the experimental subjects for hyperparameter analysis.

1) Impact of Disentangling Dimension K: The disentangling dimension K concretely represents the degree of disentangling of different aspects of a single system entity in the two views. We analyzed the range $[1, 2, 3, 4, 5]$. We observe that increasing the disentangling dimensions can enhance attack detection accuracy, as it allows SAURONEYES to capture system entity interaction tendencies from a broader perspective and more diverse aggregation levels. However, excessively high K values result in too much dimension separation, compromising the independence of each dimension's attributes of the system entities. We found that when $K = 2$, the model performs best across all datasets.

2) Impact of Contrastive Loss Weights: The contrastive loss weight parameters λ_1 and λ_2 control the impact of intra-view and inter-view contrastive learning on the final loss for the knowledge view and the interaction view, respectively. We varied λ_1 and λ_2 within the range [0.001, 0.01, 0.1, 1]. When $\lambda_1 = 0.01$ and $\lambda_2 = 0.01$, SAURONEYES performs optimally on each dataset. The reason is: excessively high contrastive learning losses may cause the model to focus too much on the relative distances between samples, neglecting the main task loss. Conversely, setting the contrastive learning loss weights too low, meaning the contrastive learning loss occupies too small a proportion in the overall loss function.

3) Impact of Embedding Size: The embedding operation maps node features to edge embeddings, specifically representing edges in a graph as low-dimensional vectors that retain the original graph structure and attribute information. From the results: As the embedding size increases, the model accuracy improves significantly and gradually stabilizes (20 epochs for StreamSpot and 40 for CADETS), reaching near-perfect accuracy (close to 1.0). In both datasets, the model with an embedding size of 64 performs best and has the highest final accuracy.

TABLE VII
COMPUTATIONAL OVERHEAD PROFILING

Phase	Time (s)	Peak Memory (MB)
Graph Construction	67.66	4,874.74
Training	2,713.81	16,489.59
Testing	26.96	17,303.00

E. Performance Overhead (RQ4)

SAURONEYES is designed to achieve detection with low and reasonable overhead. We tested the time and peak memory overhead of SAURONEYES at different stages, including **(1)Graph Construction**, **(2)Training**, and **(3)Testing**. We provide specific runtime performance overheads of SAURONEYES on CADETS-E3. It is worth noting that, GPUs are generally lacking in real production environments, we cannot require large-scale hosts in the industry to use GPUs for threat detection, with CPU being the dominant resource when GPUs are unavailable, our experiments mainly focus on CPU-based testing. During the graph construction phase, we measured the total time taken by SAURONEYES to construct a usable graph from the preprocessed dataset, as well as the memory changes before and after the process. In the training phase, we assessed the training overhead by measuring the time required for SAURONEYES to generate the complete model, and the peak memory usage during this process was recorded. In the final testing phase, we recorded the duration of APT predictions made by the trained model on input system entities, as well as the peak memory consumption.

TABLE VII presents the complete performance overhead of SAURONEYES. As shown, SAURONEYES has very low time overhead, and its memory overhead remains within the range that can be tolerated by real-world threat detection hosts. SAURONEYES shows significant advantages in practical

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.

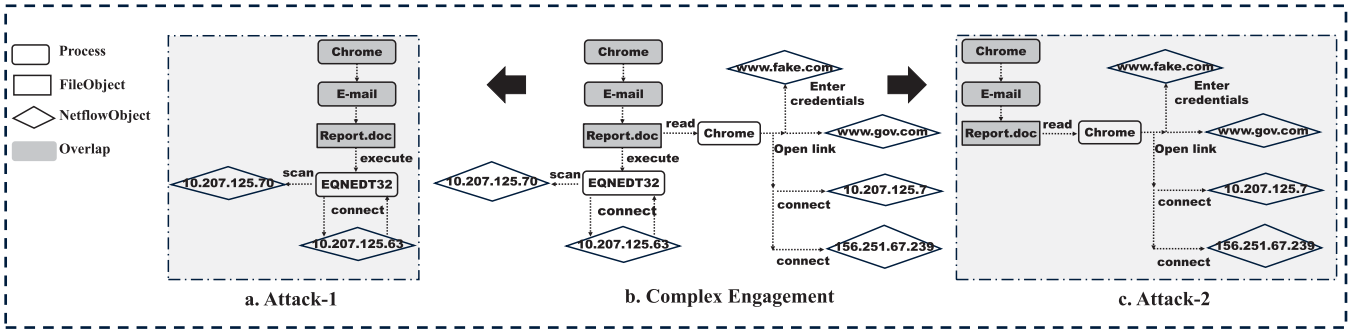
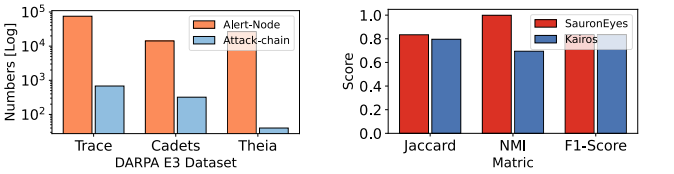


Fig. 10. Case Study: An Example of a Multi-Stage Complex Attack with Overlapping Attack Activities.

threat detection. For example, the dataset used for testing in CADETS-E3 takes almost two days to collect and amounts to approximately 4GB of log data, whereas SAURONEYES completes the prediction in just 27 seconds. The training overhead primarily stems from multiple rounds of iteration over millions of nodes and edges: each round involves multi-layer neighbor aggregation in the GNN and the computation of a complex objective function comprising the BPR loss and two types of contrastive losses (intra-view and cross-view).



a. Number of Generated Attack Chains in Entity-level Logs.

b. Completeness and Accuracy of SauronEyes Attack Chains.

Fig. 11. Attack Reconstruction Performance of SauronEyes.

F. Attack Reconstruction (RQ5)

The ultimate goal of SAURONEYES is to provide security analysts with a comprehensive and concise attack chain. In the context of the attack chain, we believe that ① a single attack chain should encompass the entirety of an attack activity, ② the attack behavior should effectively indicate the attack stages, thereby creating a directed flow of attack information, and ③ this approach aims to expedite analyst feedback and the handling of false positives (FP).

We simulated multiple attacks by an adversar, completing a sophisticated APT campaign. This showcases SAURONEYES' capability to segment and reconstruct different attack events, ensuring the attack chain remains concise and complete. Fig. 10(b) presents an overview of the attack scenario tracing, where the attacker initiated the campaign with a phishing email, leading to two distinct attack campaigns.

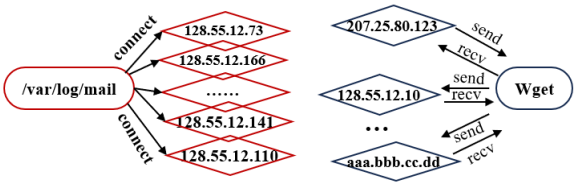
- In the first campaign depicted in Fig. 10(a), the attacker begins by crafting a simple message in a phishing email, enticing the recipient to download a file from the email. In the second step, once the recipient clicks on the Word document in the email, the attacker exploits `CVE-2017-11882`, triggering a stack overflow in the `EQNEDT32` module, which then initiates a reverse TCP connection to the C&C server at IP address `10.207.125.63`. In the third step, the attacker starts performing port scans for internal reconnaissance and establishes a stealthy connection between the target host and the attacker.
- In the second campaign depicted in Fig. 10(c), the attacker embeds a phishing link within the Word document attached to the initial phishing email. In the first step, the phishing email entices the recipient to download the file from the email. In the second step, the report.doc included a link to a website hosted at `www.gov.cn`, address `156.251.67.239`. The website hosted a form asking for name, e-mail address, and password. In the third step, the user unfortunately clicked on the link, entered the requested information, and submitted it. The results were sent back to `www.fake.com`, address `10.207.125.7`. The attacker now has access to victim's e-mail account, including contact information for other company employees.

As illustrated in Fig. 10(b), overlapping attack communities emerge when distinct attack activities share common entry points, challenging the reconstruction framework in Section V-C.3. Conventional community detection methods (e.g., Louvain algorithm adopted by Kairos and Hercules) fail to resolve such overlaps, causing incomplete attack chain extraction in aggregated community assignments. SAURONEYES overcomes this limitation through overlapping community detection, effectively distinguishing interdependent attack chains from shared entry points. This enables precise reconstruction of two distinct attack chains (Figs. 10(a)(c)) that traditional approaches would erroneously merge.

As shown in Fig. 11b, SAURONEYES demonstrates outstanding performance across various metrics, enabling a more comprehensive reconstruction of attack chains. Additionally, as illustrated in Fig. 11a, SAURONEYES significantly reduces the number of items that investigators need to review through its attack reconstruction capabilities. This facilitates a more efficient and targeted attack analysis, allowing for precise identification of attack stages, and simplifies the analysis of false positives and feedback mechanisms.

Using DARPA detection examples, we demonstrate how attack reconstruction significantly reduces security analysts' alert processing time. We illustrate this with both a false positive and a genuine attack alert. Fig. 12(a) shows a

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.



(a) Port Scanning (b) Wget Download

Fig. 12. Alert Generation Graph.

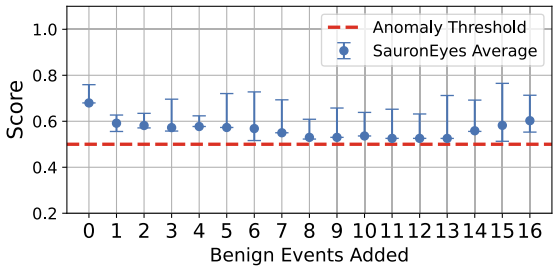


Fig. 13. Resilience Against Adversarial Attacks.

malicious `mail` process conducting a port scan across 6000 ports. The massive number of raw alert nodes makes it difficult for analysts to identify the core system event. The reconstructed attack graph in Figure 11 effectively reveals the event logic. Similarly, Fig. 12(b) depicts `Wget` communicating with multiple IPs for downloads. This behavior structurally resembles malicious scanning, triggering a false positive alert. Here again, hundreds of disconnected alert nodes obscure the event context, while the structured visualization in Fig. 12 enables analysts to quickly identify and filter it. Attack chains dramatically reduce the number of items analysts must review, as they are far fewer than individual anomaly nodes.

G. Robustness Against Adversarial Attacks (RQ6)

SAURONEYES is a provenance-based detection system. Goyal [39] et al. noted that intrusion detection systems (IDS) with graph-level granularity, such as Unicorn [6], StreamSpot [21] and ProvDetector [27] are highly sensitive to "adversarial attacks". These adversarial attacks initiate evasion attacks by manipulating the distribution graph encoding, aiming to create misleading similarities between the node neighborhood distribution of the attack graph and that of the benign source graph. To assess the robustness of SAURONEYES against adversarial attacks, we modifyed the node neighborhood in the attack graph to mimic that in the benign source graph.

Fig. 13 shows the anomaly scores (min/avg/max, y-axis) of attack nodes as benign edges (x-axis) are added. While some node scores decrease slightly with benign structures, they remain above the detection threshold. Although a few nodes evade detection, the stable average score ensures persistent event alerts. SAURONEYES maintains robustness by leveraging a decoupling module to discern structural nuances and contrastive learning to amplify node similarity patterns.

It is worth mentioning that, after inserting more camouflage events, the effectiveness of the adversarial attack surprisingly decreases. This phenomenon has also been noted in FLASH [26]. This occurs because the continuous addition of benign nodes causes a shift in the node distribution learned by our model. This change leads to an increase in the anomaly score, which in turn helps to separate the camouflaged events from the truly malicious ones.

TABLE VIII
PERFORMANCE OF CONCEPT DRIFT

Add Edge Ratio	Drift degree[1]	Accuracy	Precision
10%	0.087	$0.9875_{\blacktriangledown 0.594\%}$	$0.9307_{\blacktriangledown 1.116\%}$
15%	0.143	$0.9689_{\blacktriangledown 2.466\%}$	$0.9170_{\blacktriangledown 2.571\%}$
20%	0.183	$0.9604_{\blacktriangledown 3.322\%}$	$0.9036_{\blacktriangledown 3.322\%}$
Original Graph	0	$0.9934_{\blacktriangle 2.170\%}$[2]	$0.9412_{\blacktriangle 2.628\%}$

▼% represents the percentage of reduction, ▲% represents the percentage of improvement.
[1] Use KL divergence to calculate the degree of concept drift. The larger the value, the more serious the concept drift.
[2] Comparison of the average values of the original graph without adding random edges and various offset schemes.

H. Concept Drift (RQ7)

SAURONEYES is designed as a periodic training and near-real-time detection system, rather than a pure real-time online learning system. However, adapting to environmental changes is an important challenge commonly faced by IDS, such as concept drift. To verify the concept drift robustness of SauronEyes, we randomly generate edges in different time periods of its training set, and add edge drift to the nodes in the target time window according to the timestamp. For example, we select time windows in CADETS with an interval of one hour as the benchmark to generate edge relationships. To ensure that the randomly sampled interaction relationships meet the characteristics of sparse attacks, the randomly sampled edge relationships are only selected from common edge connections such as "read", "open", "close", and "write". We use CADETS as the experimental target, calculate the KL divergence between the sampled new graph and the original graph, and determine whether the new graph has produced concept drift according to the criteria of [32]. Experiments are conducted on the new and old training sets and fixed test sets. The results are shown in TABLE VIII. The detection accuracy and precision decrease slightly, proving that SauronEyes has obvious concept drift robustness. The current design in our paper does not include an online learning mechanism. In practice, the most feasible way to adapt to environmental changes and new threats is to periodically retrain the model with new log data.

VII. RELATED WORK

A. Provenance Detection

Advanced Persistent Threat (APT) attacks exhibit prolonged incubation periods, necessitating enterprises to maintain operational logs for over six months - a requirement that imposes significant storage overhead and highlights the

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.

critical challenge of provenance graph reduction and compression. References [15] and [24] have proposed viable solutions addressing both graph simplification and semantic preservation. Furthermore, contemporary provenance detection methodologies can be broadly categorized into three principal approaches: statistical analysis [17], [18], [19], specification-based detection [9], [11], [40], and learning-driven prediction [6], [7], [8], [9], [10], [11], [12], [13], [41], as comprehensively discussed in Section II.

B. Disentangled Representation Learning

Disentangled representation learning aims to separate the underlying factors of data by embedding objects from multiple perspectives [42], [43]. It has been applied to various fields such as text [44], image [45] and knowledge graph embedding [46]. Efforts have also been made in disentangled representation learning for predictive classification. DGCF [47] performs disentangled representation learning on embedded segmentation graph neural networks based on intent disentanglement.

C. Contrastive Learning

As an efficient self-supervised learning method, contrastive learning (CL) has achieved significant advancements in computer vision [48] and natural language processing [49]. The idea of contrastive learning is to pull positive samples together and push negative samples apart in the embedding space. Researchers have found that applying contrastive learning to graph representation learning yields excellent performance. Velickovic et al. [50] proposed Deep Graph Infomax (DGI) by maximizing the consistency between node representations and graph representations in a local-global contrastive paradigm.

VIII. Conclusion

The escalating sophistication of APT attacks and the dynamic evolution of network environments have rendered existing detection schemes inadequate for uncovering genuine attack patterns beneath adversaries' complex activities. To address this challenge, we present SAURONEYES- an innovative APT detection system that employs graph disentanglement algorithms and contrastive learning to effectively decouple intricate interaction patterns within massive audit logs(Camouflage), while resolving the inherent issue of sparse malicious interactions(Low-frequency). Our solution further implements hierarchical attack-chain reconstruction to reveal comprehensive threat scenarios(Multi-stage). Extensive evaluations across multiple datasets and detection scenarios demonstrate that SAURONEYES achieves exceptional operational performance, attaining an average detection accuracy of 99%. These results significantly advance the state-of-the-art in APT detection capabilities.

References

[1] E. J. Khaleefa and D. A. Abdulah, "Concept and difficulties of advanced persistent threats (apt): Survey," *Int. J. Nonlinear Anal. Appl.*, vol. 13, no. 1, pp. 4037–4052, 2022.

[2] A. Sharma, B. B. Gupta, A. K. Singh, and V. K. Saraswat, "Advanced persistent threats (APT): Evolution, anatomy, attribution and countermeasures," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 7, pp. 9355–9381, Jul. 2023.

[3] A. Gehani and D. Tariq, "SPADE: Support for provenance auditing in distributed environments," in *Proc. ACM/IFIP/USENIX Int. Conf. Distrib. Syst. Platforms Open Distrib. Process.*, 2012, pp. 101–120.

[4] Y. Ji et al., "RAIN: Refinable attack investigation with on-demand inter-process information flow tracking," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Dallas, TX, USA, Oct. 2017, pp. 377–390.

[5] A. Bates, D. Tian, R. B. K. Butler, and T. Moyer, "Trustworthy whole-system provenance for the Linux kernel," in *Proc. USENIX Conf. Secur. Symp. (SEC)*, Austin, TX, USA, Aug. 2015, pp. 319–334.

[6] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "Unicorn: Runtime provenance-based detector for advanced persistent threats," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020.

[7] K. Pei et al., "HERCULE: Attack story reconstruction via community discovery on correlated log graph," in *Proc. Annu. Conf. Comput. Security Appl. (ACSAC)*, 2016, pp. 583–595.

[8] Y. Shen, E. Mariconti, P.-A. Vervier, and G. Stringhini, "Tiresias: Predicting security events through deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, 2018, pp. 592–605.

[9] A. Alsaheel et al., "Atlas: A sequence-based learning approach for attack investigation," in *Proc. 30th USENIX Secur. Symp. (USENIX Secur.)*, 2021, pp. 3005–3022.

[10] S. Wang et al., "THREATRACE: Detecting and tracing host-based threats in node level through provenance graph learning," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3972–3987, 2022.

[11] Z. Li, X. Cheng, L. Sun, J. Zhang, and B. Chen, "A hierarchical approach for advanced persistent threat detection with attention-based graph neural networks," *Secur. Commun. Netw.*, vol. 2021, pp. 1–14, May 2021.

[12] J. Zengy et al., "SHADEWATCHER: Recommendation-guided cyber threat analysis using system audit records," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 489–506.

[13] Z. Cheng et al., "Kairos: Practical intrusion detection and investigation using whole-system provenance," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2024, pp. 3533–3551.

[14] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 1172–1189.

[15] N. Hossain et al., "SLEUTH: Real-time attack scenario reconstruction from COTS audit data," in *Proc. 26th USENIX Secur. Symp. (USENIX Secur.)*, 2017, pp. 487–504.

[16] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. N. Venkatakrishnan, "HOLMES: Real-time APT detection through correlation of suspicious information flows," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 1137–1152.

[17] W. U. Hassan et al., "NoDoze: Combatting threat alert fatigue with automated provenance triage," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019.

[18] Y. Liu et al., "Towards a timely causality analysis for enterprise security," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018.

[19] W. U. Hassan et al., "This is why we can't cache Nice things: Lightning-fast threat hunting using suspicion-based hierarchical storage," in *Proc. Annu. Comput. Security Appl. Conf. (ACSAC)*, 2020, pp. 165–178.

[20] (2020). *Darpa Transparent Computing Program Engagement 3 Data Release*. [Online]. Available: https://github.com/darpa-i2o/Transparent-Computing

[21] (2016). *The Streamspot Dataset*. [Online]. Available: https://github.com/sbustreamspot/sbustreamspot-data

[22] H. Yue, T. Li, D. Wu, R. Zhang, and Z. Yang, "Detecting APT attacks using an attack intent-driven and sequence-based learning approach," *Comput. Secur.*, vol. 140, May 2024, Art. no. 103748.

[23] H. Li et al., "MIRDETECTOR: Applying malicious intent representation for enhanced APT anomaly detection," *Comput. Secur.*, vol. 157, Oct. 2025, Art. no. 104588.

[24] Z. Jia, Y. Xiong, Y. Nan, Y. Zhang, J. Zhao, and M. Wen, "MAGIC: Detecting advanced persistent threats via masked graph representation learning," in *Proc. 33rd USENIX Secur. Symp. (USENIX Secur.)*, 2023, pp. 5197–5214.

[25] S. M. Milajerdi, B. Eshete, R. Gjomemo, and V. N. Venkatakrishnan, "POIROT: Aligning attack behavior with kernel audit records for cyber threat hunting," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 1795–1812.

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.

[26] M. Ur Rehman, H. Ahmadi, and W. Ul Hassan, "Flash: A comprehensive approach to intrusion detection via provenance graph representation learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2024, pp. 3552–3570.
[27] Q. Wang et al., "You are what you do: Hunting stealthy malware via data provenance analysis," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020.
[28] T. Pasquier et al., "Practical whole-system provenance capture," in *Proc. Symp. Cloud Comput.*, Sep. 2017, pp. 405–418.
[29] R. Paccagnella et al., "Custos: Practical tamper-evident auditing of operating systems using trusted execution," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020.
[30] R. Paccagnella, K. Liao, D. Tian, and A. Bates, "Logging to the danger zone: Race condition attacks and defenses on system audit frameworks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 1551–1574.
[31] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, "Disentangled graph collaborative filtering," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 1001–1010.
[32] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining Knowl. Discovery*, vol. 30, no. 4, pp. 964–994, Jul. 2016.
[33] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, "Are graph augmentations necessary?: Simple graph contrastive learning for recommendation," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2022, pp. 1294–1303.
[34] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.
[35] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, Feb. 2013, pp. 587–596.
[36] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.
[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[38] W. Cheng et al., "Omnisec: LLM-driven provenance-based intrusion detection via retrieval-augmented behavior prompting," *Available SSRN J.*, 2025.
[39] A. Goyal, X. Han, G. Wang, and A. Bates, "Sometimes, you aren't what you do: Mimicry attacks against provenance graph host intrusion detection systems," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2023.
[40] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 1777–1794.
[41] G. Wang, N. Ivanov, B. Chen, Q. Wang, T. Nguyen, and Q. Yan, "Graph learning for interactive threat detection in heterogeneous smart home rule data," in *Proc. ACM Manage. Data*, 2023, vol. 1, no. 1, pp. 1–27.
[42] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[43] F. Locatello et al., "Challenging common assumptions in the unsupervised learning of disentangled representations," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 4114–4124.
[44] V. John, L. Mou, H. Bahuleyan, and O. Vechtomova, "Disentangled representation learning for non-parallel text style transfer," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 424–434.
[45] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016.
[46] J. Wu et al., "DisenKGAT: Knowledge graph embedding with disentangled graph attention network," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2021, pp. 2140–2149.
[47] X. Wang et al., "Learning intents behind interactions with knowledge graph for recommendation," in *Proc. Web Conf.*, Apr. 2021, pp. 878–887.
[48] R. D. Hjelm et al., "Learning deep representations by mutual information estimation and maximization," in *Proc. Int. Conf. Learn. Represent.*, 2018.
[49] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 3111–3119.
[50] P. Veličković, W. Fedus, W. L. Hamilton, P. Lió, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *Proc. ICLR (Poster)*, 2018, p. 4.

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on April 02,2026 at 15:57:29 UTC from IEEE Xplore. Restrictions apply.