LARES: Host-centered Lateral Movement Detection via Inductive Graph Reasoning

Tristan Bilot
University of British Columbia, Canada
tbilot@cs.ubc.ca

Nour El Madhoun
LISITE, Isep, France
nour.el-madhoun@isep.fr

Anis Zouaoui
Adservio, France
anis.zouaoui@adservio.fr

Khaldoun Al Agha
Université Paris-Saclay, France
alagha@lri.fr

Thomas Pasquier
University of Oxford, United Kingdom
thomas.pasquier@cs.ox.ac.uk

***Abstract*—Detecting lateral movement (LM) within enterprise networks remains a difficult challenge, with existing state-of-the-art methods often failing to identify such activity effectively. Recent approaches typically rely on unsupervised graph learning to model normal host behavior, using a network graph in which nodes represent hosts and edges correspond to observed events. Although these methods aim to detect LM at the edge level, their practical use is limited by two main shortcomings: (1) they are limited to fixed-size networks and cannot infer LM for newly observed hosts, and (2) they generate a high number of false positive edges, leading to alert fatigue among security analysts. We present LARES, a lightweight, unsupervised framework for detecting LMs in evolving networks without retraining for new hosts. LARES is built on a single observation: lateral movement is causal. It emanates from compromised source hosts and propagates outward to neighbors. Prior methods ignore this structure and score edges as independent anomalies, which is why their precision collapses under realistic class imbalance. LARES mirrors the causal structure of the attack: an inductive encoder learns host behavior not bound to a fixed output host set, generalizing to hosts unseen at training time; a two-stage decoder then localizes compromised sources before tracing their outgoing LM edges. On two realistic enterprise networks, LARES maintains effective detection even as the network grows fivefold, substantially improves precision over recent state-of-the-art methods, and delivers 10–100× faster inference.**

***Note*—This is a preprint version of the paper accepted at the 42nd IEEE Annual Computer Security Applications Conference (ACSAC'26) [1].**

***Code*—https://github.com/TristanBilot/lares**

***Datasets*—https://drive.google.com/drive/folders/1tWf4B1o5zkw6u0xJ1QcYbzsMySjKFoI2?usp=sharing**

***Index Terms*—lateral movement detection, intrusion detection, graph neural networks, inductive learning**

1. Introduction

Detecting lateral movement (LM) within enterprise networks remains a critical challenge in the constantly evolving field of cybersecurity. The sophistication and stealth employed by modern threats, especially advanced persistent threats (APTs), highlight the shortcomings of traditional signature-based network intrusion detection systems (NIDSs). By relying on predefined patterns of known attacks, these systems often struggle to identify new or obfuscated threats that do not conform to established signatures [2]. In contrast, anomaly-based detection methods [3]–[10] provide a more adaptive defense by modeling the normal behavior of networked systems and identifying deviations that may indicate compromise [11], [12]. This shift from signature-based to anomaly-based detection is particularly important for identifying indicators of APTs. Indeed, APTs employ sophisticated strategies that often mimic legitimate behavior, making them effectively invisible to signature-based detection methods [13].

Unsupervised machine learning has shown effectiveness in distinguishing normal from anomalous behavior in network data [14]–[16], and deep learning has gained prominence for behavior-based LM detection by processing raw network data and reducing reliance on handcrafted features [17], [18]. However, these methods typically operate on flat representations (e.g., vectors or grids) that are ill-suited to capture the complex, structured dependencies of advanced threats, where APTs and zero-day exploits often appear as weak structural signals dispersed over irregular and extended timeframes [19], [20]. Graph neural networks (GNNs) address this limitation by modeling fine-grained structure in network graphs, and recent self-supervised approaches learn benign activity without labeled attack data [4]–[7], [19], [21], [22], improving anomaly detection and identifying stealthy behaviors such as LMs. Despite this progress, recent GNN-based LM detection systems face two persistent obstacles that limit practical deployment, as we demonstrate in §2.2: (1) transductive design limited to a fixed host set, limiting deployment to static networks and reducing applicability in dynamic enterprise environments;

and (2) low precision under real-world event volumes, which produces excessive false positives and burdens security analysts [23], [24].

We propose LARES, a LM detection framework built on a single design principle: lateral movement is causal. Attacks originate from compromised source hosts and propagate outward to neighbors, producing localized anomalies anchored at identifiable sources. Prior systems ignore this structure. They treat edges as independent anomalies and predict maliciousness edge-by-edge, which is precisely why their precision collapses under realistic class imbalance. LARES instead mirrors the causal structure of the attack: (i) an inductive graph encoder learns host behavior from local neighborhoods rather than memorizing identities, enabling generalization to hosts unseen during training; (ii) a host-agnostic reconstruction decoder models benign interactions without binding to a fixed host set; and (iii) a two-stage detection procedure first localizes compromised sources, then traces their outgoing LM edges, reconstructing the attack's causal chain.

Contributions

Inductive architecture for evolving networks. We design an inductive architecture that generalizes LM detection to unseen hosts, maintaining high precision under substantial train-test host shift, including settings where many inference-time hosts were unseen during training.
Two-stage detection paradigm. We first identify suspicious source hosts, then run edge-level LM detection, cutting false positives by 14-90$\times$ in the LANL transductive setting, and by 16–107$\times$ across inductive settings.
Lightweight design with practical efficiency. LARES achieves 10–100$\times$ faster inference with a lightweight architecture and comparable memory usage.
Open science. We release LARES' implementation and weights (§A).

2. Background & Motivation

2.1. Related Work

Signature-based NIDS. Signature-based NIDSs are widely adopted in mainstream monitoring tools [25] due to their speed, simplicity, and effectiveness. However, attacks must match a known signature [26] that needs to be carefully tuned to reduce false positive rates [27]. Experts must manually adjust rulesets to balance detection and false alarms, a process that is both time-consuming and essential for optimal performance [28]. While some approaches offer configurable trade-offs between detection and false alarm rates [29], tuning these settings precisely remains difficult [27].
Supervised NIDS. Several studies have applied machine learning and deep learning techniques [30]–[33] to reduce the need for manual tuning by automatically learning and calibrating from network features. However, these methods rely on flat data representations and have increasingly been supplemented by graph-based methods, which more effectively model the complex interdependencies between hosts [19]. Recent advances in GNNs for modeling structural patterns in graphs have made them a popular choice for network intrusion detection. E-GraphSAGE [34], for example, applies a supervised, edge-based variant of GraphSAGE [35] to detect intrusions in network graphs, incorporating edge features to enhance detection accuracy. However, supervised techniques rely on labeled data that are hard to obtain [19], [36], [37].
Unsupervised NIDS. Unsupervised techniques have gained increasing attention for detecting unknown and complex attacks [3]–[7], [12], [38]–[41]. These anomaly-based methods learn patterns of normal host behavior and flag deviations at inference time. By not relying on attack labels, they are well-suited to identifying novel attack variants that may be absent from the training data [38].

E-GraphSAGE was notably extended to unsupervised settings with Anomal-E [7], which leverages graph contrastive learning to learn edge embeddings by distinguishing positive and negative graph samples, maximizing local-global mutual information. However, while Anomal-E's training is unsupervised, it relies on attack data in the training set to cluster attacks as part of its detection pipeline.
Lateral Movement Detection. Lateral movements are a specific phase of advanced attacks, typical of APTs, where attackers expand by moving from an initially compromised host to other systems within the network, often in pursuit of higher privileges or access to sensitive assets. These attacks are stealthy and attempt to mimic benign events, making them difficult to detect with conventional NIDSs that rely on signatures or supervised machine learning [3], [4], [21].

State-of-the-art techniques for LM detection are based on unsupervised learning [3]–[6]. PIKACHU [6] performs unsupervised LM detection by modeling the network as a stream of graph snapshots and capturing both structural and temporal patterns using temporal random walks, Skip-gram [42], and a recurrent neural network (RNN). More recently, PIKACHU has been shown to be outperformed by more powerful systems based on GNNs [3]–[5]. EULER [3], [4] constructs graph snapshots from network events, encodes nodes using a graph convolutional network (GCN) combined with a RNN, and detects malicious edges based on predicted link scores that exceed a threshold determined on a validation set. ARGUS [5] incorporates edge features into message passing and introduces a loss function optimized for average precision (AP). While both systems advance the state of practice, they exhibit high false positive rates in our evaluation and their performance degrades in evolving networks where hosts join after training (see §2.2). JBEIL [21] extends lateral movement detection to evolving networks by employing a temporal graph network (TGN) [43] for inductive link prediction [44]–[46], using contrastive learning to distinguish positive from negative edges. However, based on the available source code [47] and our reproduction efforts, we find no evidence that the system has been evaluated specifically for lateral movement detection, and our experiments show lower performance than

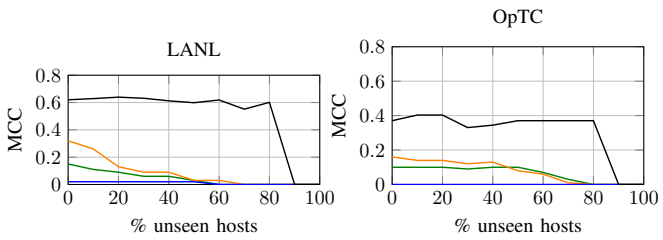


Figure 1: MCC scores of systems on the LANL and OpTC datasets, as the proportion of unseen hosts (i.e., excluded from training), including malicious hosts, increases. ■ EULER, ■ ARGUS, ■ JBEIL, and ■ LARES.

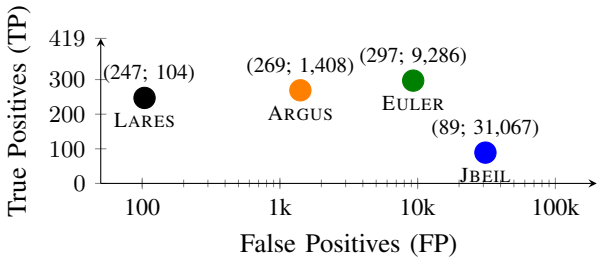


Figure 2: Predicted TPs vs. FPs for state-of-the-art LM detection systems and LARES in the transductive setting on the LANL dataset.

reported (further details in §4.2).

2.2. Limitations of Existing Systems

While prior work has acknowledged challenges in LM detection, EULER and ARGUS recognize false positive issues [4], [5], and JBEIL attempts to address network evolution [21], these systems still fall short of practical deployment requirements. We quantify two specific limitations that remain inadequately addressed.

(1) Incompatibility with Evolving Networks. Enterprise networks are inherently dynamic: employees join and leave, devices are added and removed, and guest access creates transient hosts. A practical LM detection system must handle this dynamism without requiring complete retraining whenever the network topology changes.

Existing systems are fundamentally ***transductive*** [48], [49]: they require knowledge of all network hosts at training time. This constraint stems from two architectural choices: (i) softmax decoders that compute edge probabilities over a fixed set of N hosts, and (ii) RNNs that process temporal dependencies across the same fixed N hosts. As shown in Figure 1, when we exclude hosts from training to simulate network evolution, detection performance collapses. With just 30% of hosts unseen, EULER's MCC drops by 47% on the realistic LANL dataset [50]; ARGUS falls by 69%. At 50% unseen hosts (a realistic scenario for networks with significant turnover), both systems achieve near-zero MCC, rendering them ineffective. While JBEIL [21] claims inductive capability, our experiments show it is ineffective in practice (see §4.2 for details).

LARES adopts an ***inductive*** design [35], [51] that learns generalizable structural patterns rather than memorizing host identities, maintaining high MCC even when 50% of hosts are absent from training. We describe the inductive components in §3.3 and §3.4.

(2) Excessive False Positives. Even when operating in their intended transductive setting, existing systems generate too many false positives (FPs) to be practical. As shown in Figure 2, on the LANL dataset, EULER produces roughly 31 FPs for every detected LM edge, while ARGUS reduces this to a 5:1 ratio. On OpTC, which has extreme class imbalance (see Table 1), both EULER and ARGUS produce roughly 30 FPs for every true detection. Under such imbalance, metrics like FPR and AUC-ROC used in prior work [4]–[6] obscure the true scale of FPs—e.g., ARGUS [5] reports an FPR of 0.0005 and AUC of 0.9983 on LANL, yet still generates 1,309 FPs. We instead report raw TP and FP counts to directly reflect analyst workload. These high volumes (Figure 2) lead to analyst alert fatigue [52] and burnout [24]. Analysts already face hundreds of alerts daily [23], making systems that add thousands more FPs unusable in practice [53].

The problem stems from detecting attacks at the edge level. Enterprise networks have far more edges than nodes, and malicious edges make up a tiny fraction of all edges. Finding anomalies at this level means distinguishing rare malicious events from millions of benign ones—even tiny per-edge error rates compound into thousands of false alerts.

LARES uses a two-stage approach tailored to LM detection: first identify anomalous hosts, then examine their emitted edges as potential candidates for LM. This cuts false positives by an order of magnitude compared to prior methods.

3. Design

LARES, illustrated in Figure 3, instantiates the causal design principle from §1. Each component mirrors a facet of LM attacks: hosts as the point of compromise, edges as the propagation mechanism, and source-anchored anomalies as the detection signal. (a) Network events are first transformed into a sequence of graph snapshots, each capturing system activity over a fixed time window (§3.2). (b) The graph encoder computes node and edge embeddings by capturing host-specific structural features from a subset of hosts seen at training time (§3.3). (c) The decoder models benign behavior by reconstructing edge embeddings through an autoencoder, while enabling inference over arbitrary hosts (§3.4). (d) Malicious hosts are identified by applying thresholding and unsupervised clustering to the most anomalous nodes. Lateral movements are then inferred based on these detected hosts (§3.5). Steps (b) and (c) address limitation **(1)**, enabling detection of hosts and events unseen during training, while step (d) targets limitation **(2)**, reducing the volume of false positives through two-stage detection. Finally, unlike recent NIDSs that grow in complexity [54], our

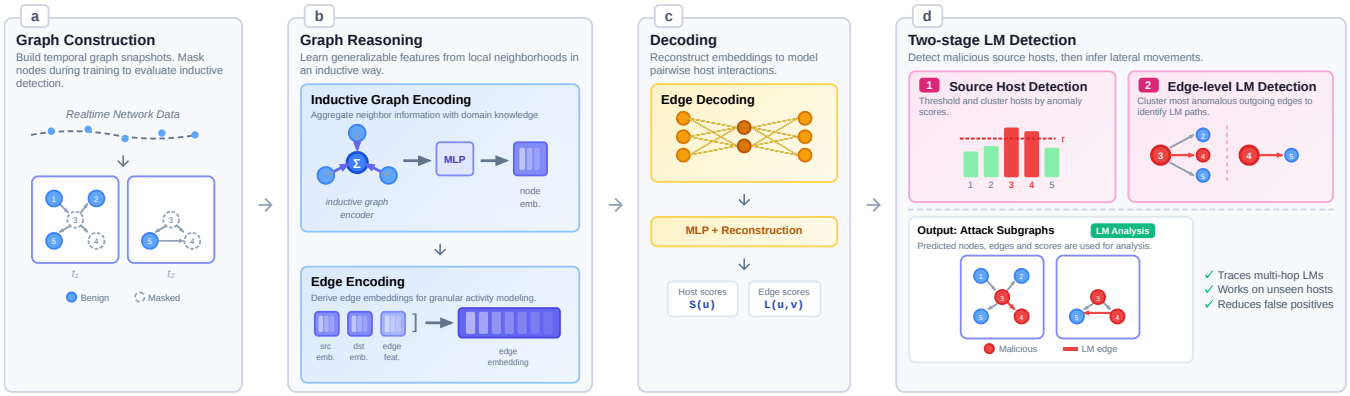


Figure 3: Architecture of LARES. **(a)** Graph snapshots from network events (masking only for inductive evaluation). **(b)** An inductive encoder learns generalizable structural patterns from host neighborhoods, enabling inference on unseen hosts. **(c)** A host-agnostic reconstruction decoder produces host scores $S(u)$ and edge scores $\mathcal{L}_{(u,v)}$ without retraining. **(d)** Two-stage detection first localizes anomalous source hosts via clustering, then identifies their malicious outgoing edges, exploiting the causal structure of LM attacks.

design remains deliberately simple; we show this simplicity is sufficient (§5.5) and greatly improves inference time (§5.8).

3.1. Threat Model

LARES targets network-observable lateral movement in enterprise networks (remote service exploitation, credential-based movement) where an adversary with initial access to one internal host seeks to expand to others, possibly blending malicious activity with normal traffic. The defender has network monitoring (e.g., NetFlow, authentication logs) and a benign training period, but no labeled attack data (unsupervised setting). Given this benign-only training, mimicry attacks [55] are a known risk, which we evaluate in §5.6.

3.2. Graph Construction

LARES represents the network as a temporal graph $G = \{S_1, \ldots, S_T\}$ where each snapshot $S_t = \{X_{n_t}, X_{e_t}, E_t\}$ captures node features X_{n_t}, edge features X_{e_t}, and the edges E_t over a fixed-duration window sized to fit in memory. Nodes are hosts (identified by local IP) and edges are observed inter-host events; multi-edges within a snapshot are merged following [4], [5]. Edge features encode port/protocol frequencies, while node features are one-hot vectors identifying each host (§3.3).

3.3. Graph Reasoning

The graph reasoning component is designed to address two key objectives: (i) enable LM detection in evolving networks where hosts join after training, and (ii) overcome the transductive limitations of existing encoders that embed a fixed number of hosts into their architecture.

To achieve these objectives, we design an inductive graph encoder that learns *generalizable structural patterns* characterizing benign vs. malicious host behavior, rather than memorizing the identities of specific hosts.

1) Inductive Graph Encoder. We design an encoder that derives host representations from local neighborhood structure, allowing both known and newly introduced hosts to obtain meaningful embeddings from connectivity patterns. Benign host activity exhibits distinctive structural signatures, for instance from workstations, developers, or administrators interacting with network resources. The encoder should learn these patterns and detect deviations independently of host identity. It must also address a key challenge: enterprise networks are dense, with nodes exhibiting high in-degree (e.g., 205 average in-degree per graph in OpTC; see Table 1). Such nodes are vulnerable to *over-squashing* [56], where information from many neighbors is compressed into a fixed-size embedding. Degree-based aggregation, used in GCN-based approaches [4], [5], exacerbates this by averaging away distinguishing signals. We instead employ sum aggregation, which accumulates detailed information from all neighbors and guarantees that distinct neighborhood configurations yield distinct embeddings [57]. Additionally, network events encode rich domain knowledge through edge attributes such as ports and protocols, which the encoder must incorporate to reason inductively. Transductive encoders often ignore or underutilize this information. We design our message-passing mechanism to explicitly incorporate edge weights (capturing interaction frequency) and edge features (capturing communication characteristics), enabling the encoder to reason from the full richness of network telemetry.

Formally, we compute the embedding of a node v as:

$$\mathbf{h}_v^{(k+1)} = \text{MLP}\left(\mathbf{h}_v^{(k)} + \sum_{u \in N(v)} \phi\left(\mathbf{h}_u^{(k)}, w_{uv}, e_{uv}\right)\right). \quad (1)$$

Here, $\mathbf{h}_u^{(k)}$ and $\mathbf{h}_v^{(k)}$ denote the embeddings of the source and destination nodes at layer k, respectively. At the first layer,

these embeddings are initialized via a linear transformation of the node features: $\mathbf{h}_u^{(0)} = \mathbf{W}_p \mathbf{x}_u$, where $\mathbf{W}_p$ is a projection matrix and $\mathbf{x}_u$ is the feature vector of node u, consisting of a unique one-hot encoded vector. MLP denotes a multi-layer perceptron, ϕ is defined as a message function, described in Equation 2, while e_{uv} and w_{uv} represent the edge features and the edge weight of edge (u, v), respectively. The message function ϕ is designed to aggregate the information from the source node u and the associated edge weight and edge features:

$$\phi\left(\mathbf{h}_u, w_{uv}, e_{uv}\right) = \sigma\left(\mathbf{W}_m\left(w_{uv}\mathbf{h}_u + \mathbf{W}_e e_{uv}\right)\right), \quad (2)$$

where $\mathbf{W}_m$ and $\mathbf{W}_e$ are two weight matrices and σ is the Tanh activation function. We show in §5.5 that this encoder design significantly outperforms traditional encoders.

Edge Features. We use three network attributes as edge features: *Source Ports*, *Destination Ports*, and *Protocols*. These features carry useful information about network activity and are encoded as categorical variables. We chose these features due to their universal availability across network monitoring tools and their presence in most datasets. In this work, edge features encode the frequency of port types and protocols. Assigning a dimension to every port is impractical, so we reserve unique dimensions for common low-range ports (e.g., 80, 135, 139) and group high-range ports (above 5355) into one unit. This keeps the vector compact while preserving key information. Edge vectors are then standardized per snapshot: $e_{uv} = \sigma\left(\frac{x_{uv}-\mu_\epsilon}{\Sigma_\epsilon}\right)$, where σ is the Sigmoid function, x_{uv} is the raw feature vector, and μ_ϵ, Σ_ϵ are the mean and standard deviation of the snapshot's features. This standardization differentiates edges by assigning higher weights to frequent interactions and lower weights to infrequent ones. Our ablation study (see §5.5) shows that LARES maintains robust detection performance even when these features are excluded, demonstrating that its effectiveness does not depend on any single feature.

Node Features. Following established practice in prior work [3]–[5], we encode each host as a unique one-hot vector, capturing host-specific behaviors and semantics. A host appearing only at inference receives a new dimension, absent during training: its projection weights in $\mathbf{W}_p$ are untrained and encode no identity. Its embedding instead derives from connectivity, as message passing (Equations 1 and 2) propagates representations learned for its neighbors; identity weights for the new host are only learned at the next retraining (§5.9). The ablation study in §5.5 demonstrates that these features consistently enhance performance.

2) Edge Encoding. At this stage, the node embeddings effectively profile the activity of hosts by leveraging both the graph topology and edge features. However, since LMs inherently occur between two hosts, anomaly scores should be calculated at the edge level. These edge-level scores are then leveraged in the two-stage detection process: first, identifying suspicious hosts, and second, detecting edges that are likely to correspond to lateral movements (see §3.5).

To achieve this, we derive edge embeddings from the corresponding node embeddings and reintegrate domain-specific edge features, which might be lost during message passing due to over-squashing. Formally, we compute edge embeddings by concatenating the embeddings of the two end nodes with the corresponding edge feature vector:

$$\mathbf{h}_{uv} = \left[\mathbf{h}_u, \mathbf{h}_v, \mathbf{W}_g e_{uv}\right], \quad (3)$$

where $\mathbf{h}_{uv}$ represents the embedding of an edge (u, v), $\mathbf{h}_u$ and $\mathbf{h}_v$ are the embedding of the two end nodes, $\mathbf{W}_g$ is a weight matrix that weighs the importance of each feature, and $[.,.]$ represents the concatenation operation.

LANL Statistics	**Value**
# Nodes	13,183
# Edges	2,498,716
# Root Attack Nodes	1
# Malicious Edges	419
% Malicious Edges	0.0167%
$\bar{x}$ Degree	0.70
$\bar{x}$ Edges / Snapshot	7572
$\bar{x}$ Unique Src / Snapshot	5836
$\bar{x}$ Unique Dst / Snapshot	113

OpTC Statistics	**Value**
# Nodes	1114
# Edges	72,712,765
# Root Attack Nodes	3
# Malicious Edges	59
% Malicious Edges	0.0000811%
$\bar{x}$ Degree	205
$\bar{x}$ Edges / Snapshot	180,877
$\bar{x}$ Unique Src / Snapshot	507
$\bar{x}$ Unique Dst / Snapshot	347

TABLE 1: Preprocessed LANL and OpTC statistics: # = occurrences, % = percentage, $\bar{x}$ = mean.

3.4. Decoding

Recent LM detectors rely on softmax decoders [4], [5] that require the final layer to align with a fixed set of nodes, making them unsuitable for inductive settings. We instead shift from classification to a reconstruction task: an autoencoder AE compresses edge embeddings and reconstructs them, distilling the patterns of benign interactions so that anomalous edges produce high reconstruction error. Our ablation (§5.5) shows this is essential for both inductive inference and precision.

Formally, the reconstruction score of an edge (u, v) and the snapshot training loss are

$$\mathcal{L}_{(u,v)} = \left(\mathbf{h}_{uv} - \mathrm{AE}(\mathbf{h}_{uv})\right)^2, \qquad \mathcal{L}_t = \frac{1}{|E_t|}\sum_{(u,v)\in E_t} \mathcal{L}_{(u,v)}, \quad (4)$$

where AE is a bottleneck autoencoder with two linear layers.

3.5. Two-stage LM Detection

State-of-the-art systems [3]–[5], [21] directly calculate edge-level anomaly scores based on the reconstruction loss from their decoders, followed by thresholding to distinguish between benign and anomalous edges. However, this approach often results in the detection of isolated anomalous network events, leading to extremely high false positive rates due to the sheer volume of such network events (edges in Table 1). In practice, lateral movements are not single, standalone events. They originate from compromised nodes and unfold as a sequence of related actions.

Building on this insight, we introduce a two-stage detection approach: (i) We first identify the root nodes responsible for lateral movement (i.e., nodes surrounded by a high

proportion of anomalous activity), and (ii) we then detect the edges in their neighborhood that correspond to lateral movements toward adjacent hosts. This strategy significantly reduces the search space and lowers the number of false positives by focusing on the most malicious hosts.

Because each attack can be causally traced back to a compromised root, focusing on hosts first enables precise localization of the threat; direct edge detection instead highlights dispersed activity without capturing the underlying causal structure. LARES therefore identifies anomalous hosts via thresholding, isolates the most likely malicious ones via clustering, and then clusters their outgoing edges by reconstruction score to select the attack edges.

Stage 1 - Source Host Detection. We derive host scores from edge scores so that a single anomalous LM is not diluted by surrounding benign activity. We define the anomaly score of host u as the maximum reconstruction loss across its outgoing edges:

$$S(u) = \max_{(u,v)\in E_t} \mathcal{L}_{(u,v)}. \tag{5}$$

Max aggregation ensures even one highly anomalous event raises the host score, preventing attackers from evading detection by mixing in benign traffic [58], [59] (see §5.6). We then flag hosts whose score exceeds a threshold τ (the maximum score observed on validation) and apply K-means [60] to the flagged hosts, retaining only the highest-scoring cluster. Unlike pure thresholding [4], [5], [12], clustering adapts to the score distribution and further filters benign anomalies [38].

Stage 2 - Edge-level LM Detection. Edge detection is now restricted to the outgoing edges of detected hosts. For each detected host u, we apply K-means to the scores $\mathcal{L}_{(u,v)}$ of its above-threshold outgoing edges and designate the higher-scoring cluster as malicious. By coupling host-level localization with edge-level clustering, LARES can reconstruct multi-hop LM paths when each hop's source is flagged (§5.4), providing analysts with host-centered summaries instead of the dispersed alerts produced by prior approaches [4]–[7], [34], [61]–[63]; quantifying per-chain hop coverage, however, would require datasets containing several multi-hop campaigns, none of which is publicly available today.

4. Experimental Setup & Implementation

LARES' framework can be deployed across diverse environments with minimal integration effort. Generally, our evaluation methodology follows practices established by previous work. We evaluate LARES on two widely used network security datasets and provide details on their characteristics and the feature preprocessing steps applied. We compare LARES against three state-of-the-art LM detection systems. We implemented all models with PyTorch Geometric [64] and experiments ran on a server with an NVIDIA Tesla V100 GPU (32GB) and an Intel Xeon Gold 6148 CPU (20 cores, 100GB RAM).

4.1. Datasets

We evaluate LARES on the well-established LANL [50] and OpTC [65] datasets (see descriptions in §C), following the methodology used in recent LM detection works [3]–[5], [21]. Although other network security datasets exist [66]–[68], they do not capture LM scenarios, and their attack-heavy distributions hinder unsupervised learning of benign behavior. Nevertheless, we study the generalization on the larger intrusion detection landscape with a recent enterprise dataset [68] in §5.10. The LANL and OpTC remain the only public datasets providing realistic lateral movement scenarios with proper temporal separation and sufficient scale.

Train-Test Split. In OpTC, we train on benign activity from the first 4.5 days, validate on the next 0.5 days, and test on the final 3 days containing attacks, following a similar approach as in EULER and ARGUS. For LANL, training is performed on the first 39 hours of benign data, followed by 3 hours for validation, and testing extends through day 14, aligning with ARGUS and following the protocols of Kent et al. [69].

Ground Truth. For both datasets, we follow the same methodology as EULER and ARGUS, labeling an edge as malicious if it is associated with a malicious label within a given snapshot. In this work, we define a node as a root attack node if it is the source of a malicious edge and is identified in the ground truth files as initiating the attack. Based on this definition, our analysis identified one root attack node in the LANL dataset and three in the OpTC dataset. The resulting ground truth labels are publicly available as part of our datasets release.

Graph Construction. Our methodology for creating graph snapshots follows EULER's and ARGUS' approach to ensure a fair comparison. For the OpTC dataset, network flows are extracted from the `eCar` directory [70], specifically focusing on `START` events associated with `FLOW` objects to capture the initiation of host communications. Labels are assigned using the red team ground truth. For the LANL dataset, we parse events from `auth.txt` and apply labels using `redteam.txt`.

The preprocessing involves the following steps: (i) extracting events and their associated features into CSV files representing one-minute intervals; and (ii) combining these files into graph snapshots stored in tensor format, where nodes correspond to IP addresses and edges represent network flows (OpTC) or authentication events (LANL). We select node and edge features following the methodology established in previous work [3]–[5].

4.2. Baselines

We compare LARES against LM detection systems meeting the criteria in Table 6 (§D): published in top-tier venues, targeting network-level lateral movement, and providing open-source implementations for reproducible comparison. For fairness, all baselines are evaluated using their origi-

nal implementations with hyperparameters tuned via grid search.

EULER (NDSS'22) [3], [4]. EULER is a distributed system that creates graph snapshots from network events, generates node embeddings with a GCN, and processes them with a RNN for temporal feature extraction. The encoder is trained to predict edges, detecting malicious ones if their probability score exceeds a threshold calculated on a validation set.

ARGUS (S&P'24) [5]. Following a similar approach, ARGUS incorporates network features into its message-passing mechanism and uses a different loss function to improve average precision (AP).

Inductive variant. The GCN-based encoders used in EULER and ARGUS are not suited for inductive settings. To enable a fair comparison with LARES, we evaluate an additional variant that replaces the transductive GCN with an inductive graph attention network (GAT) [71]. Similar variants have been previously benchmarked in EULER [4].

JBEIL (S&P'24) [21]. JBEIL introduces an inductive method for temporal link prediction in dynamic network graphs, using a TGN. The available implementation [47] supports only an edge prediction task, which distinguishes positive from negative edges via negative sampling. Our experiments suggest that the reported performance in the original paper corresponds to this setup. However, this task formulation does not align with the stated objectives of lateral movement detection, as the evaluation metrics are based on predicted edge types (positive vs. negative) rather than ground truth labels indicating attack or benign activity. Furthermore, we note that while JBEIL uses augmented threat data generated with a custom version of an empirical attack synthesis framework [72], did not release these supplementary data, nor did they mention how they integrated synthetic attack data using "lateral-movement-simulato" [73]. We also note that independent work [53] reported similar issues when reproducing the JBEIL's published results.

To enable a fair comparison with other baselines, we adapt JBEIL to a proper edge-level LM detection task[1]. Specifically, we reinterpret the scores assigned to predicted positive edges as indicators of benign or attack behavior, and evaluate them directly against ground truth labels, consistent with the approach in EULER and ARGUS.

Non-graph alternative. To evaluate the added value of leveraging graph structure over flat feature representations, we include a traditional one-class SVM [74] trained solely on the network flow features used in LARES. This baseline isolates the contribution of edge features and serves as a simpler, non-graph alternative for comparison.

4.3. Metrics

Malicious edges are True Positives (TPs), benign ones are True Negatives (TNs); misclassifications are False Positives (FPs) or False Negatives (FNs). We report Recall $= \frac{\text{TP}}{\text{TP+FN}}$ and Precision $= \frac{\text{TP}}{\text{TP+FP}}$. Given the extreme class imbalance noted in §2.2, FPR and AUC-ROC become overly optimistic, so we use the Matthews Correlation Coefficient (MCC) [38], [75] as our primary metric. MCC balances TP/FP/TN/FN in a single score in $[-1, 1]$ (see §E); it should only be compared across classifiers on the same dataset. We use average precision (AP) to assess performance in a threshold-independent manner.

1. We will make the code publicly available.

5. Evaluation

This section evaluates the core contributions of LARES, specifically its ability to detect LM paths at the edge level. We also evaluate LARES' ability to achieve detection with improved precision compared to the state-of-the-art, as a practical system should minimize false positives, prioritizing precision over exhaustive true positive identification with high recall [38], [53].

We present the edge-level performance results in §5.2 and §5.3. For completeness, we also evaluate all baselines on the node-level task, aiming to detect the root attack hosts responsible for LMs. However, we emphasize that EULER and ARGUS were originally developed for edge-level detection, and JBEIL for edge prediction. Therefore, this evaluation focuses on edge-level detection and we provide intermediate node-level results in §F.

We address nine research questions: how LARES performs transductively (**RQ1**, §5.2) and inductively on evolving networks (**RQ2**, §5.3); how its outputs benefit analysts (**RQ3**, §5.4); the role of its key components (**RQ4**, §5.5); its robustness to adversarial attacks (**RQ5**, §5.6); the influence of hyperparameters (**RQ6**, §5.7); its scalability versus prior systems (**RQ7**, §5.8); its robustness to concept drift (**RQ8**, §5.9); and its inductive generalization to broader intrusion detection beyond LM (**RQ9**, §5.10).

5.1. Experiments

We conduct a series of experiments to evaluate LARES' effectiveness in detecting LMs under inductive settings, and while producing fewer false positives, thereby addressing the initial limitations **(1)** and **(2)** introduced in §2.2. We model network growth by withholding a subset of hosts during training and introducing them only at inference: withheld hosts contribute no edges or gradients during training, inducing a train-test host shift that mirrors expansion to new machines. Specifically, `Exp0` serves as a baseline transductive evaluation, whereas `Exp1`, `Exp2`, and `Exp3` represent different inductive scenarios. In `Exp1`, malicious hosts are included during training, while in `Exp2` and `Exp3`, they are excluded. All systems are evaluated under exactly the same conditions (i.e., the present/excluded nodes are the same).

Exp0. No hosts are removed; the training set remains unchanged. This transductive setting, as used in EULER and ARGUS, evaluates performance when both attacker and victim hosts are seen during training.

Exp1. We randomly exclude 30% of hosts from training, while ensuring all malicious hosts remain included. This

Method	Exp	LANL TP	LANL FP	LANL TN	LANL FN	LANL Recall	LANL Precision	LANL MCC	LANL AP	OpTC TP	OpTC FP	OpTC TN	OpTC FN	OpTC Recall	OpTC Precision	OpTC MCC	OpTC AP
EULER	Exp0	297	9,286	2.49M	122	0.71	0.03	0.15	0.18	27	1,179	72.71M	32	0.46	0.02	0.10	0.08
	Exp1	201	13,283	2.49M	218	0.48	0.01	0.08	0.06	24	1,209	72.71M	35	0.41	0.02	0.09	0.12
	Exp2	156	14,423	2.48M	263	0.37	0.01	0.06	0.09	21	837	72.71M	38	0.36	0.02	0.09	0.07
	Exp3	199	16,767	2.48M	220	0.47	0.01	0.07	0.05	24	998	72.71M	35	0.41	0.02	0.10	0.13
EULER$_{ind}$	Exp0	267	9,488	2.49M	152	0.64	0.03	0.13	0.16	28	2,235	72.71M	31	0.48	0.01	0.08	0.06
	Exp1	192	7,593	2.49M	227	0.46	0.02	0.11	0.08	28	1,762	72.71M	31	0.47	0.02	0.09	0.12
	Exp2	166	9,599	2.49M	253	0.40	0.02	0.08	0.11	26	2,134	72.71M	33	0.44	0.01	0.07	0.05
	Exp3	205	13,089	2.49M	214	0.49	0.02	0.09	0.07	12	1,310	72.71M	47	0.20	0.01	0.04	0.07
ARGUS	Exp0	269	1,408	2.50M	150	0.64	0.16	0.32	0.29	39	944	72.71M	20	0.67	0.04	0.16	0.19
	Exp1	163	5,847	2.49M	256	0.39	0.03	0.10	0.13	39	989	72.71M	20	0.67	0.04	0.16	0.14
	Exp2	137	5,582	2.49M	282	0.33	0.02	0.09	0.07	32	1,238	72.71M	27	0.54	0.03	0.12	0.15
	Exp3	102	6,881	2.49M	317	0.24	0.01	0.06	0.09	34	1,444	72.71M	25	0.58	0.02	0.12	0.10
ARGUS$_{ind}$	Exp0	190	2,228	2.50M	229	0.45	0.08	0.19	0.22	33	2,112	72.71M	26	0.56	0.02	0.09	0.07
	Exp1	177	2,376	2.50M	242	0.42	0.07	0.17	0.14	33	2,083	72.71M	26	0.56	0.02	0.09	0.12
	Exp2	164	1,524	2.50M	255	0.39	0.10	0.19	0.16	32	3,768	72.71M	27	0.54	0.01	0.07	0.10
	Exp3	157	1,902	2.50M	262	0.37	0.08	0.17	0.20	17	1,982	72.71M	42	0.29	0.01	0.05	0.03
JBEIL	Exp0	89	31,067	2.47M	330	0.21	0.00	0.02	0.04	3	149,394	72.56M	56	0.05	0.00	0.00	0.02
	Exp1	89	36,922	2.46M	330	0.21	0.00	0.02	0.01	3	148,923	72.56M	56	0.05	0.00	0.00	0.01
	Exp2	88	44,121	2.45M	331	0.21	0.00	0.02	0.05	3	177,890	72.53M	56	0.05	0.00	0.00	0.03
	Exp3	88	56,760	2.44M	331	0.21	0.00	0.02	0.00	1	169,251	72.54M	58	0.02	0.00	0.00	0.02
OC-SVM	Exp0	45	2,739	2.50M	374	0.12	0.02	0.04	0.07	19	395	72.71M	40	0.32	0.05	0.12	0.09
	Exp1	45	2,901	2.50M	374	0.12	0.02	0.04	0.02	19	648	72.71M	40	0.27	0.02	0.10	0.13
	Exp2	49	3,255	2.50M	370	0.13	0.01	0.04	0.06	19	887	72.71M	40	0.27	0.02	0.08	0.06
	Exp3	41	3,994	2.49M	378	0.11	0.01	0.03	0.05	17	861	72.71M	42	0.27	0.02	0.07	0.10
LARES	Exp0	247	104	2.50M	172	0.61	0.70	0.65	0.70	27	63	72.71M	32	0.46	0.30	0.37	0.39
	Exp1	241	116	2.50M	178	0.59	0.68	0.63	0.66	27	63	72.71M	32	0.46	0.30	0.37	0.39
	Exp2	241	116	2.50M	178	0.59	0.68	0.63	0.66	24	66	72.71M	35	0.41	0.27	0.33	0.37
	Exp3	241	156	2.50M	178	0.59	0.61	0.60	0.62	27	63	72.71M	32	0.49	0.30	0.37	0.36

TABLE 2: Edge-level detection results on the LANL and OpTC datasets.

inductive setting tests generalization when only part of the network is observed.

Exp2. We randomly exclude 30% of hosts, ensuring *all* malicious ones are also excluded. This tests inductive performance when no malicious hosts are known.

Exp3. Building on Exp2, this experiment excludes 50% of hosts to simulate high-turnover networks or frequent guest access. It evaluates robustness and generalization in highly inductive settings with limited training data.

5.2. Transductive Detection Performance

We provide the edge-level detection results of systems across all experiments in Table 2. We first focus on the transductive setting (Exp0), which is the standard setting used in EULER, ARGUS and most NIDSs [6], [7], [34]. In this setting, all hosts in the test set are seen during training, meaning training and testing take place within the exact same network.

LANL. Consistent with the original TPs and FPs reported in EULER and ARGUS, both systems exhibit high FPs and low precision on LANL. ARGUS yields 1,408 FPs for 269 TPs, EULER 9,286 FPs for 297 TPs, and JBEIL 31,067 FPs for 89 TPs. The poor performance of JBEIL indicates that the system fails under the same evaluation setup used for EULER and ARGUS (see §4.2), when assessed against actual ground truth labels rather than simply distinguishing between positive and negative edges. In contrast, LARES detects 247 TPs with only 104 FPs, achieving 70% precision. This represents a 90× reduction in FPs compared to EULER and a 14× reduction compared to ARGUS.

OpTC. High class imbalance (72M edges, 59 malicious) challenges all baselines. Interestingly, the non-GNN-based OC-SVM leads among baselines with 5% precision, followed by ARGUS at 4% (39 TPs, 944 FPs). LARES excels, detecting 28 malicious edges with 62 FPs for 31% precision. This leads to 19× and 15× reduction in FPs compared to EULER and ARGUS, respectively.

5.3. Inductive Detection Performance

In this evaluation we focus on the inductive settings (Exp1, Exp2, Exp3 in Table 2, which represent the primary contribution of LARES: enabling LM detection in evolving networks where hosts are unseen during training.

LANL. Most systems achieve high recall but near 0% precision. ARGUS' GAT variant performs best, with 7–10% precision across experiments, yet its 160 TPs among 1,500+ FPs provide limited value for security analysts [38], [53]. LARES largely outperforms baselines, maintaining high precision between 61% and 68% and solid TP counts in all inductive experiments. In Exp3, it divides FPs by 107× compared to EULER and 44× compared to ARGUS, while detecting more TPs.

OpTC. ARGUS, the top baseline, reaches only 4% precision, declining further in more inductive experiments. OC-SVM's simple architecture achieves a steady 2% precision across all experiments. Conversely, LARES sustains ∼30% precision despite class imbalance, detecting nearly half the attack edges in a robust way across all experiments. In Exp3, it divides FPs by 16× compared to EULER and 23× compared to ARGUS. While JBEIL performs better on this task, it remains the worst of our 3 baselines.

When does detection fail? To probe the limits of inductive robustness, we gradually increase the proportion of unseen

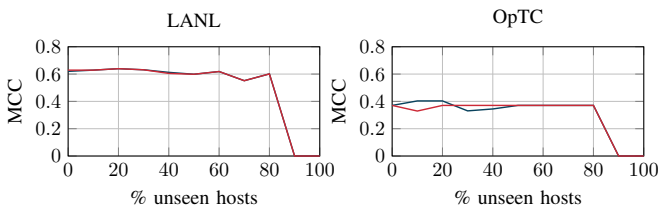
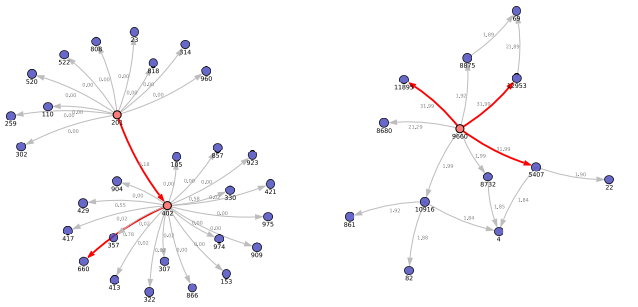


Figure 4: Evolution of MCC as the number of unseen hosts (i.e., removed from training) increases. ■ shows MCC when malicious hosts are excluded from training, whereas ■ shows MCC when they are included.



(a) Two lateral movements initiated from malicious hosts in the OpTC "Plain PowerShell Empire" campaign.

(b) Lateral movement attempts via authentication from a malicious host to multiple benign hosts in LANL dataset's 12th day.

Figure 5: Attack subgraphs generated by LARES extracting 2-hop neighbors of detected hosts in a time window: gray edge = benign; red edge = malicious; blue node = benign; red node = malicious; **bold** edge = predicted as malicious.

hosts. Figure 4 shows that LARES' MCC degrades only gradually up to 80% host removal, but collapses at 90%. The sharp drop occurs because nearly all neighbors of malicious nodes are then absent from training, causing a distribution shift at test time (concept drift). Such a gap is unlikely in practice, as LARES can be retrained frequently with minimal overhead (§5.8). Even excluding all malicious hosts from training does not significantly degrade performance, confirming that LARES learns generalizable inductive features rather than memorizing host-specific behavior.

5.4. Qualitative Analysis

During inference, LARES' predictions enable the extraction of host-centered subgraphs around detected nodes, supporting interpretation of inferred LM paths. Figure 5 illustrates an example in which LARES identifies LMs missed by other baselines during `Exp3`. In graph (a), LARES detects two root attack hosts within a single time window and predicts one outgoing LM edge from each. From an analyst's perspective, this reveals a successful lateral movement chain from host 201 to 402, followed by propagation to host 660. In graph (b), host 9660 is first flagged due to abnormal behavior, after which three high-scoring outgoing edges are identified. These edges correspond to failed authentication

Exp	Encoder	Decoder	Detection	Features	LANL			OpTC		
					Recall	Precision	MCC	Recall	Precision	MCC
Exp0	EULER	–	–	–	0.45	0.47	0.46	0.00	0.00	0.00
	ARGUS	–	–	–	0.00	0.00	0.00	0.41	0.40	0.40
	LSTM	–	–		0.45	0.46	0.47	0.47	0.31	0.38
	–	EULER	–	–	0.52	0.45	0.49	0.41	0.27	0.33
	–	ARGUS	–	–	0.50	0.57	0.53	0.00	0.00	0.00
	–	–	Edge	–	0.56	0.27	0.39	0.56	0.01	0.08
	–	–	–	Node	0.45	0.52	0.48	0.41	0.27	0.33
	–	–	–	Edge	0.49	0.56	0.52	0.42	0.28	0.34
	–	–	–	–	0.60	0.70	0.65	0.46	0.30	0.37
Exp1	EULER	–	–	–	0.45	0.47	0.46	0.00	0.00	0.00
	ARGUS	–	–	–	0.00	0.00	0.00	0.41	0.40	0.40
	LSTM	–	–		0.00	0.00	0.00	0.51	0.25	0.36
	–	EULER	–	–	0.54	0.47	0.50	0.41	0.13	0.23
	–	ARGUS	–	–	0.55	0.48	0.51	0.00	0.00	0.00
	–	–	Edge	–	0.56	0.09	0.22	0.56	0.01	0.08
	–	–	–	Node	0.43	0.36	0.39	0.41	0.27	0.33
	–	–	–	Edge	0.53	0.24	0.36	0.41	0.27	0.33
	–	–	–	–	0.59	0.68	0.63	0.46	0.30	0.37
Exp2	EULER	–	–	–	0.45	0.47	0.46	0.00	0.00	0.00
	ARGUS	–	–	–	0.00	0.00	0.00	0.41	0.40	0.40
	LSTM	–	–		0.00	0.00	0.00	0.47	0.47	0.47
	–	EULER	–	–	0.54	0.47	0.50	0.00	0.00	0.00
	–	ARGUS	–	–	0.55	0.48	0.51	0.00	0.00	0.00
	–	–	Edge	–	0.56	0.09	0.22	0.58	0.01	0.09
	–	–	–	Node	0.43	0.36	0.39	0.42	0.28	0.34
	–	–	–	Edge	0.53	0.24	0.36	0.41	0.27	0.33
	–	–	–	–	0.59	0.68	0.63	0.46	0.30	0.37
Exp3	EULER	–	–	–	0.45	0.52	0.49	0.00	0.00	0.00
	ARGUS	–	–	–	0.00	0.00	0.00	0.41	0.27	0.33
	LSTM	–	–		0.00	0.00	0.00	0.44	0.29	0.36
	–	EULER	–	–	0.56	0.49	0.52	0.00	0.00	0.00
	–	ARGUS	–	–	0.56	0.06	0.19	0.00	0.00	0.00
	–	–	Edge	–	0.44	0.51	0.47	0.58	0.01	0.09
	–	–	–	Node	0.43	0.36	0.39	0.41	0.27	0.33
	–	–	–	Edge	0.50	0.48	0.49	0.41	0.27	0.33
	–	–	–	–	0.59	0.61	0.60	0.46	0.30	0.37

TABLE 3: Ablation Study of LARES.

attempts. Early detection of such attempts enables proactive mitigation by blocking access from the compromised host.

5.5. Ablation Study

We ablate each component of LARES to identify which addresses the two initial challenges: **(1)** the transductive design of prior systems, and **(2)** their high false-positive rate (Table 3).

Encoder. Substituting LARES' encoder with EULER's or ARGUS' GCN yields inconsistent behavior: EULER's fails on OpTC but succeeds on LANL, while ARGUS' shows the opposite. LARES' encoder delivers strong and consistent performance across both datasets and all inductive experiments, mitigating limitation **(1)**. Adding an LSTM after the GNN brings no benefit and fails on LANL inductively, while inflating runtime—hence its omission.

Decoder. Replacing the reconstruction decoder with EULER's cross-entropy decoder or ARGUS' AP-loss decoder consistently degrades performance and yields near-zero precision on OpTC: transductive decoders learn host-specific edge probabilities and cannot generalize. LARES' reconstruction decoder is host-agnostic, addressing **(1)**.

Detection. Replacing two-stage detection by direct edge thresholding (as in EULER and ARGUS) sharply reduces precision, particularly on imbalanced OpTC. Pre-selecting malicious hosts narrows the search space, and clustering retains only the most concentrated anomalies, addressing **(2)**; this ablation is the necessity test for the causal design principle of §3.5.

Features. Removing node (one-hot) or edge (port/protocol frequencies) features both hurt recall and precision, but LARES retains solid performance without edge features,

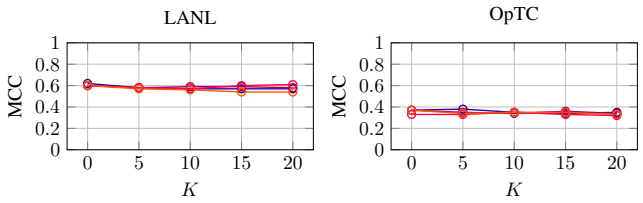


Figure 6: Effect of poisoning attack, adding K malicious edges during training, on edge-level MCC. Exp0, Exp1, Exp2, and Exp3.

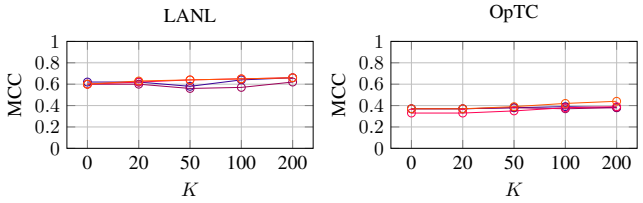


Figure 7: Effect of evasion attack, adding K benign edges to malicious hosts during inference. Exp0, Exp1, Exp2, and Exp3.

suggesting it relies on graph structure rather than on rare port/protocol patterns used in attacks [12], [76]. One-hot node features still help inductively (Exp3), as GNNs propagate the learned host information to unseen nodes through message passing.

5.6. Robustness to Adversarial Attacks

Adversarial attacks pose a significant threat to network-based detection systems [77]. We assess LARES' robustness against two common adversarial strategies: poisoning and evasion attacks.

Poisoning Attack. We evaluate robustness to training-time manipulation by injecting K malicious edges into the training set, following the methodology of [5]. Figure 6 shows minimal MCC decrease as K increases on both datasets. This robustness is expected for reconstruction-based anomaly detectors [12], [38]: since the autoencoder minimizes average reconstruction error across all edges, a small number of poisoned samples ($K \ll |E|$) has minimal impact on the learned benign manifold, and malicious edges remain outliers with high reconstruction error.

Evasion Attack. We evaluate whether attackers can evade source host detection by diluting malicious activity with benign traffic. We inject K benign edges into malicious hosts during attack windows, selecting edges previously emitted during training to simulate typical background traffic. Figure 7 shows stable MCC scores that even improve for higher K values. This counterintuitive result occurs because injected traffic increases overall activity volume, making attack windows more anomalous relative to training, a phenomenon observed in prior mimicry attack studies [55]. More fundamentally, LARES' max-aggregation for host scoring (Eq. 5) ensures malicious edge scores are

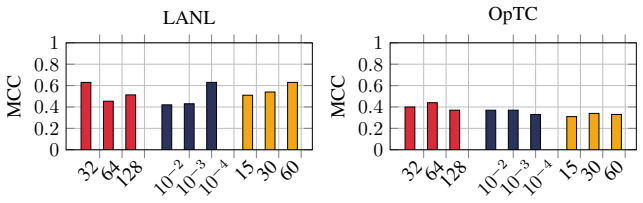


Figure 8: Evaluation of LARES' edge-level MCC with various hyperparameter values (Exp2). Embedding Size, Learning Rate, and Snapshot Size.

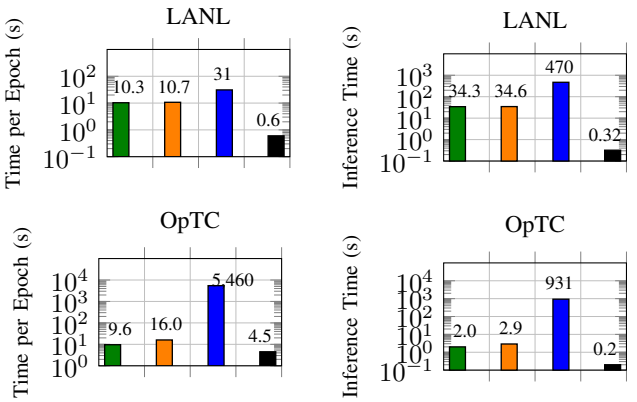


Figure 9: Runtime and memory usage for LANL and OpTC datasets (Exp0). EULER, ARGUS, JBEIL, and LARES.

preserved regardless of surrounding benign traffic; attackers cannot reduce their most anomalous event's score by adding benign ones.

5.7. Influence of Hyperparameters

Figure 8 shows LARES' MCC across different hyperparameter settings, including node embedding size, learning rate, and snapshot size (in minutes). LARES maintains stable detection across various hyperparameter settings. We attribute this robustness to its two-stage strategy, which identifies malicious nodes with high confidence before predicting edges. Since edge-level detection is restricted to the local neighborhood of detected nodes, the search space for potential malicious edges remains limited. This localized approach reduces the risk of false positives in unrelated regions of the graph.

5.8. Scalability

We measure runtime and memory usage in Figure 9. EULER, although distributed, is run on a single GPU-equipped host to enable fair comparison.

Training Time. LARES achieves at least a 17× per-epoch speedup on LANL and 2× on OpTC, as it avoids the costly RNN operations over N nodes used in EULER and ARGUS.

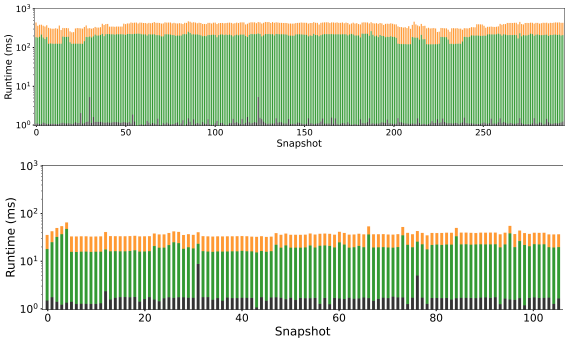


Figure 10: Log-scale inference times on LANL (up) and OpTC (down), with snapshot sizes of 60 min and 30 min, respectively. ■ EULER, ■ ARGUS, and ■ LARES.

Inference Time. At inference, LARES is at least 100× faster on LANL and 10× faster on OpTC across the full test set. JBEIL, built on TGN, suffers from sequential edge processing: unlike node-scaling encoders (EULER, ARGUS, LARES), its encoder scales with edge count, which dominates in network graphs. Per-snapshot inference (Figure 10) drops to 1–2 ms, enabling near real-time detection with a sliding window approach.

Memory. EULER and LARES peak around 2 GB, while ARGUS demands more due to its AP loss, which scales with edges per snapshot. Encoders scale with nodes and decoders with edges (see §G); the snapshot size bounds edges processed at once, preventing memory spikes.

5.9. Concept Drift

Like other ML-based detectors, LARES may be affected by concept drift as benign behavior diverges from training-time patterns [78], [79]. Its low training cost makes periodic retraining a practical mitigation: at 0.6 s/epoch on LANL and 4.5 s/epoch on OpTC (Figure 9), retraining from scratch on a sliding window of recent traffic completes in minutes on a single GPU with under 2 GB of memory, well within standard SOC maintenance windows [53]. Online learning is avoided due to known poisoning risks [80], [81]. To quantify drift directly, we train LARES on a fixed-size benign window that we slide backward in time, progressively increasing the gap between training and the held-out attack period. Figure 11 shows that MCC degrades gradually as this gap grows: on LANL, MCC drops from 0.65 at no gap to 0.54 at a 7-day gap (a 20% relative degradation); on OpTC, from 0.37 to 0.30 at a 3-day gap (19%). This gradual degradation, combined with LARES' minutes-scale retraining cost, supports nightly retraining as a viable operational strategy. Existing public datasets do not yet capture multi-month drift in a benchmarkable way [12], which we view as important future work.

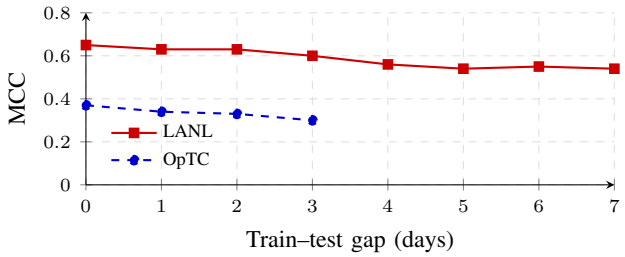


Figure 11: MCC of LARES as a function of the temporal gap between the training window and the held-out attack period.

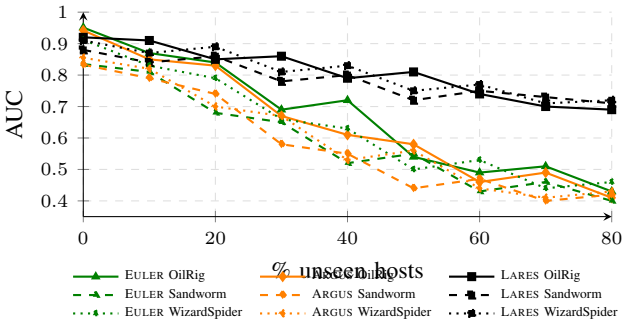


Figure 12: Inductive generalization to intrusion detection, tested across three APT scenarios from [68].

5.10. Inductive Generalization to Intrusion Detection

Wang et al. [68] report that ARGUS and EULER, despite high AUC on a recent large-scale enterprise dataset, collapse in precision under real traffic volume: on the Sandworm APT scenario, ARGUS detects 528 true positives among 1.5M false positives. They attribute this to extreme class imbalance combined with edge-level thresholding that scales poorly. We evaluate LARES on the three APT scenarios in their dataset (OilRig, Sandworm, and WizardSpider) under the same inductive protocol as Figure 4. These scenarios contain full APT campaigns rather than LM-only traces: lateral movement is present in each, with Sandworm in particular relying heavily on internal propagation, but is mixed with credential abuse, C&C beaconing, and command execution. We therefore evaluate LARES as a general edge-level inductive intrusion detector on these scenarios, treating any malicious flow as a positive regardless of attack stage. Figure 12 reports AUC for direct comparability with the numbers in [68]. EULER and ARGUS degrade rapidly under inductive conditions across all three scenarios, trending toward random performance by 50% unseen, while LARES retains an advantage throughout. The two pathologies the paper targets, transductivity and edge-level thresholding, compound on enterprise traffic, and only the inductive design preserves the ranking quality of the anomaly score.

6. Conclusion

LARES is an unsupervised LM detection system for evolving networks that minimizes false positives via a two-stage process: first identifying malicious hosts, then predicting LM edges. It achieves significantly higher precision than recent state-of-the-art systems, even when all malicious and many benign hosts are excluded during training, while its simpler design yields substantially faster inference.

References

[1] T. Bilot, A. Zouaoui, K. Al Agha, N. El Madhoun, and T. Pasquier, "LARES: Host-centered Lateral Movement Detection via Inductive Graph Reasoning," in *Annual Computer Security Applications Conference (ACSAC'26)*. IEEE, 2026.

[2] K. Stouffer, J. Falco, K. Scarfone *et al.*, "Guide to Industrial Control Systems (ICS) Security," *NIST special publication*, vol. 800, no. 82, pp. 16–16, 2011.

[3] I. J. King and H. H. Huang, "Euler: Detecting network lateral movement via scalable temporal link prediction," in *Network and Distributed System Security Symposium (NDSS'22)*. Internet Society, 2022.

[4] ——, "Euler: Detecting network lateral movement via scalable temporal link prediction," *ACM Transactions on Privacy and Security*, 2023.

[5] J. Xu, X. Shu, and Z. Li, "Understanding and Bridging the Gap Between Unsupervised Network Representation Learning and Security Analytics," in *Symposium on Security and Privacy (S&P'24)*. IEEE, 2024.

[6] R. Paudel and H. H. Huang, "Pikachu: Temporal Walk Based Dynamic Graph Embedding for Network Anomaly Detection," in *Network Operations and Management Symposium (NOMS'22)*. IEEE/IFIP, 2022.

[7] E. Caville, W. W. Lo, S. Layeghy, and M. Portmann, "Anomal-E: A self-supervised network intrusion detection system based on graph neural networks," *Knowledge-Based Systems*, vol. 258, p. 110030, 2022.

[8] X. Han, X. Yu, T. Pasquier, D. Li, J. Rhee, J. Mickens, M. Seltzer, and H. Chen, "SIGL: Securing software installations through deep graph learning," in *Security Symposium (USENIX Sec'21)*. USENIX, 2021.

[9] S. Wang, Z. Wang, T. Zhou, H. Sun, X. Yin, D. Han, H. Zhang, X. Shi, and J. Yang, "Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3972–3987, 2022.

[10] F. Yang, J. Xu, C. Xiong, Z. Li, and K. Zhang, "PROGRAPHER: An Anomaly Detection System based on Provenance Graph Embedding," in *Security Symposium (USENIX Sec'23)*. USENIX, 2023.

[11] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, pp. 949–961, 2019.

[12] Z. Cheng, Q. Lv, J. Liang, Y. Wang, D. Sun, T. Pasquier, and X. Han, "KAIROS: Practical Intrusion Detection and Investigation using Whole-system Provenance," in *Symposium on Security and Privacy (S&P'24)*. IEEE, 2024.

[13] P. Chen, L. Desmet, and C. Huygens, "A study on advanced persistent threats," in *International Conference on Communications and Multimedia Security*. IFIP, 2014.

[14] M. Rabbani, Y. Wang, R. Khoshkangini, H. Jelodar, R. Zhao, S. Bagheri Baba Ahmadi, and S. Ayobi, "A review on machine learning approaches for network malicious behavior detection in emerging technologies," *Entropy*, vol. 23, no. 5, p. 529, 2021.

[15] Q. Xiao, J. Liu, Q. Wang, Z. Jiang, X. Wang, and Y. Yao, "Towards network anomaly detection using graph embedding," in *International Conference on Computational Science*. Springer, 2020.

[16] R. O. Shittu, "Mining intrusion detection alert logs to minimise false positives & gain attack insight," Ph.D. dissertation, City University London, 2016.

[17] Y. Wu, D. Wei, and J. Feng, "Network attacks detection methods based on deep learning techniques: a survey," *Security and Communication Networks*, vol. 2020, pp. 1–17, 2020.

[18] H. M. Soliman, G. Salmon, D. Sovilj, and M. Rao, "A Graph Neural Network Approach for Scalable and Dynamic IP Similarity in Enterprise Networks," in *International Conference on Cloud Networking (CloudNet'20)*. IEEE, 2020.

[19] T. Bilot, N. El Madhoun, K. Al Agha, and A. Zouaoui, "Graph neural networks for intrusion detection: A survey," *IEEE Access*, 2023.

[20] Y. Guo, "A review of machine learning-based zero-day attack detection: Challenges and future directions," *Computer communications*, vol. 198, pp. 175–185, 2023.

[21] J. Khoury, D. Klisura, H. Zanddizari, G. D. L. T. Parra, P. Najafirad, and E. Bou-Harb, "Jbeil: Temporal Graph-Based Inductive Learning to Infer Lateral Movement in Evolving Enterprise Networks," in *Symposium on Security and Privacy (S&P'24)*. IEEE, 2024.

[22] T. Bilot, N. El Madhoun, K. Al Agha, and A. Zouaoui, "A Benchmark of Graph Augmentations for Contrastive Learning-Based Network Attack Detection with Graph Neural Networks," in *Cyber Security in Networking Conference (CSNet'23)*. IEEE, 2023.

[23] B. A. Alahmadi, L. Axon, and I. Martinovic, "99% false positives: a qualitative study of soc analysts' perspectives on security alarms," in *Security Symposium (USENIX Sec'22)*. USENIX, 2022.

[24] S. C. Sundaramurthy, A. G. Bardas, J. Case, X. Ou, M. Wesch, J. McHugh, and S. R. Rajagopalan, "A human capital model for mitigating security analyst burnout," in *Symposium on Usable Privacy and Security (SOUPS'15)*. USENIX, 2015.

[25] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks," in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.

[26] A. Khalid, A. Zainal, M. A. Maarof, and F. A. Ghaleb, "Advanced persistent threat detection: A survey," in *International Cyber Resilience Conference (CRC'21)*. IEEE, 2021.

[27] J. Díaz-Verdejo, J. Muñoz-Calle, A. Estepa Alonso, R. Estepa Alonso, and G. Madinabeitia, "On the detection capabilities of signature-based intrusion detection systems in the context of web attacks," *Applied Sciences*, vol. 12, no. 2, p. 852, 2022.

[28] H. Holm and M. Ekstedt, "Estimates on the effectiveness of web application firewalls against targeted attacks," *Information Management & Computer Security*, vol. 21, no. 4, pp. 250–265, 2013.

[29] ModSecurity, *https://github.com/owasp-modsecurity/ModSecurity, [Accessed on 15th September 2026]*, 2025.

[30] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Symposium on Security and Privacy (S&P'10)*. IEEE, 2010.

[31] S. A. R. Shah and B. Issac, "Performance comparison of intrusion detection systems and application of machine learning to snort system," in *Future Generation Computer Systems 80*, 2018, pp. 157–170.

[32] A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," in *Knowledge-Based Systems*, vol. 189, 2020.

[33] C. Fu, "Realtime robust malicious traffic detection via frequency domain analysis," in *Conference on Computer and Communications Security (CCS'21)*. ACM, 2021.

[34] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-graphsage: A graph neural network based intrusion detection system for IoT," in *Network Operations and Management Symposium (NOMS'22)*. IEEE/IFIP, 2022.

[35] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Conference on Neural Information Processing Systems (NeurIPS'17)*, 2017.

[36] J. L. Guerra, C. Catania, and E. Veas, "Datasets are not enough: Challenges in labeling network traffic," *Computers & Security*, vol. 120, p. 102810, 2022.

[37] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys & Tutorials*, 2015.

[38] B. Jiang, T. Bilot, N. El Madhoun, K. Al Agha, A. Zouaoui, S. Iqbal, X. Han, and T. Pasquier, "ORTHRUS: Achieving High Quality of Attribution in Provenance-based Intrusion Detection Systems," in *Security Symposium (USENIX Sec'25)*. USENIX, 2025.

[39] S. Li, F. Dong, X. Xiao, H. Wang, F. Shao, J. Chen, Y. Guo, X. Chen, and D. Li, "NODLINK: An Online System for Fine-Grained APT Attack Detection and Investigation," in *Network and Distributed System Security Symposium (NDSS'24)*. The Internet Society, 2024.

[40] Z. Jia, Y. Xiong, Y. Nan, Y. Zhang, J. Zhao, and M. Wen, "MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning," in *Security Symposium (USENIX Sec'24)*. USENIX, 2024.

[41] A. Goyal, G. Wang, and A. Bates, "R-CAID: Embedding Root Cause Analysis within Provenance-based Intrusion Detection," in *Symposium on Security and Privacy (S&P'24)*. IEEE, 2024.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *International Conference on Learning Representations - Workshop Track (ICLR'13)*, 2013.

[43] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal Graph Networks for Deep Learning on Dynamic Graphs," in *International Conference on Machine Learning (ICML'20)*, 2020.

[44] T. N. Kipf and M. Welling, "Variational Graph Auto-Encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[45] M. Qin and D.-Y. Yeung, "Temporal link prediction: A unified framework, taxonomy, and review," *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–40, 2023.

[46] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and S. Y. Philip, "Graph self-supervised learning: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 5879–5900, 2022.

[47] Jbeil, *https://github.com/surender08/Jbeil, [Accessed on 15th September 2026]*, 2024.

[48] T. Joachims, "Transductive Inference for Text Classification using Support Vector Machines," in *International Conference on Machine Learning (ICML'99)*, 1999.

[49] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations (ICLR'17)*, 2017.

[50] A. D. Kent, "Comprehensive, multi-source cyber-security events data set," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2015.

[51] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," in *International Conference on Learning Representations (ICLR'20)*, 2020.

[52] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, "NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage," in *Network and Distributed System Security Symposium (NDSS'19)*. The Internet Society, 2019.

[53] F. Dong, S. Li, P. Jiang, D. Li, H. Wang, L. Huang, X. Xiao, J. Chen, X. Luo, Y. Guo, and X. Chen, "Are we there yet? An Industrial Viewpoint on Provenance-based Endpoint Detection and Response Tools," in *Conference on Computer and Communications Security (CCS'23)*. ACM, 2023.

[54] K. Wolsing, L. Thiemt, C. v. Sloun, E. Wagner, K. Wehrle, and M. Henze, "Can Industrial Intrusion Detection Be SIMPLE?" in *European Symposium on Research in Computer Security (ESORICS'22)*. Springer, 2022.

[55] 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 *Network and Distributed System Security Symposium (NDSS'23)*. The Internet Society, 2023.

[56] J. Topping, F. Di Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein, "Understanding over-squashing and bottlenecks on graphs via curvature," in *International Conference on Learning Representations (ICLR'22)*, 2022.

[57] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations (ICLR'19)*, 2019.

[58] P. Fogla and W. Lee, "Evading Network Anomaly Detection Systems: Formal Reasoning and Practical Techniques," in *Conference on Computer and Communications Security (CCS'06)*. ACM, 2006.

[59] H. Yan, X. Li, W. Zhang, R. Wang, H. Li, X. Zhao, F. Li, and X. Lin, "Automatic evasion of machine learning-based network intrusion detection systems," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2023.

[60] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[61] J. Lan, J. Z. Lu, G. G. Wan, Y. Y. Wang, C. Y. Huang, S. B. Zhang, Y. Y. Huang, and J. N. Ma, "E-minBatch GraphSAGE: An Industrial Internet Attack Detection Model," *Security and Communication Networks*, vol. 2022, no. 1, p. 5363764, 2022.

[62] X. Sun and J. Yang, "HetGLM: Lateral Movement Detection by Discovering Anomalous Links with Heterogeneous Graph Neural Network," in *International Performance, Computing, and Communications Conference (IPCCC'22)*. IEEE, 2022.

[63] B. Bowman, C. Laprade, Y. Ji, and H. H. Huang, "Detecting lateral movement in enterprise computer networks with unsupervised graph AI," in *International Symposium on Research in Attacks, Intrusions and Defenses (RAID'20)*, 2020.

[64] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," *ICLR Workshop*, 2019.

[65] R. Arantes, C. Weir, H. Hannon, and M. Kulseng, "Operationally transparent cyber (optc)," 2021. [Online]. Available: https://dx.doi.org/10.21227/edq8-nk52

[66] V. Kanimozhi and T. P. Jacob, "Artificial Intelligence based Network Intrusion Detection with Hyper-Parameter Optimization Tuning on the Realistic Cyber Dataset CSE-CICIDS2018 using Cloud Computing," in *International Conference on Information Communication and Signal Processing (ICCSP'19)*. IEEE, 2019.

[67] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Annual Military Communications and Information Systems Conference (MilCIS'15)*. IEEE, 2015.

[68] C. Wang, P. Zheng, J. Gui, C. Hua, and W. Hassan, "R+R: From Claims to Crashes: A Systematic Re-evaluation of Graph-Based Network Intrusion Detection Systems," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC'25)*. ACM, 2025.

[69] A. D. Kent, "Cyber security data sources for dynamic network research," in *Dynamic Networks and Cyber-Security*. World Scientific, 2016, pp. 37–65.

[70] OpTCNCR, *https://drive.google.com/drive/u/0/folders/1n3kkS3KR31KUegn42yk3-e6JkZvf0Caa, [Accessed on 15th September 2026]*, 2020.

[71] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," in *International Conference on Learning Representations (ICLR'18)*, 2018.

[72] G. Ho, M. Dhiman, D. Akhawe, V. Paxson, S. Savage, G. M. Voelker, and D. Wagner, "Hopper: Modeling and detecting lateral movement," in *Security Symposium (USENIX Sec'21)*. USENIX, 2021.

[73] grantho, *https://github.com/grantho/lateral-movement-simulator, [Accessed on 15th September 2026]*, 2025.

[74] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[75] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA)-Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[76] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "HOLMES: Real-time APT Detection through Correlation of Suspicious Information Flows," in *Symposium on Security and Privacy (S&P'19)*. IEEE, 2019.

[77] X. Xu, Q. Hao, Z. Yang, B. Li, D. Liebovitz, G. Wang, and C. A. Gunter, "How to Cover up Anomalous Accesses to Electronic Health Records," in *Security Symposium (USENIX Sec'23)*. USENIX, 2023.

[78] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[79] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting Concept Drift in Malware Classification Models," in *USENIX Security Symposium (USENIX Sec'17)*. USENIX, 2017.

[80] R. G. Margiotta, S. Goldt, and G. Sanguinetti, "Attacks on online learners: A teacher-student analysis," *Conference on Neural Information Processing Systems (NeurIPS'23)*, 2023.

[81] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," in *Conference on Computer and Communications Security (CCS'18)*. ACM, 2018.

[82] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs," in *AAAI Conference on Artificial Intelligence (AAAI'20)*, 2020.

[83] J. Chen, X. Wang, and X. Xu, "GC-LSTM: Graph convolution embedded LSTM for dynamic network link prediction," *Applied Intelligence*, 2022.

[84] E. Hajiramezanali, A. Hasanzadeh, K. Narayanan, N. Duffield, M. Zhou, and X. Qian, "Variational graph recurrent neural networks," *Conference on Neural Information Processing Systems (NeurIPS'19)*, 2019.

[85] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *SIGMOD International Conference on Management of Data (SIGMOD'00)*. ACM, 2000.

[86] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *International Conference on Data Mining (ICDM'08)*. IEEE, 2008.

[87] G. Apruzzese, F. Pierazzi, M. Colajanni, and M. Marchetti, "Detection and threat prioritization of pivoting attacks in large networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 404–415, 2017.

[88] Q. Liu, J. W. Stokes, R. Mead, T. Burrell, I. Hellen, J. Lambert, A. Marochko, and W. Cui, "Latte: Large-scale lateral movement detection," in *Military Communications Conference (MILCOM'18)*. IEEE, 2018.

[89] B. Bowman, I. J. King, R. Melamed, and H. H. Huang, "NETHAWK: Hunting Advanced Persistent Threats via Structural and Temporal Graph Anomalies," in *2024 IEEE Conference on Communications and Network Security (CNS'24)*. IEEE, 2024.

[90] J. Gui, M. Nie, J. Guo, F. Zou, M. U. Rehman, and W. U. Hassan, "A Principled Approach for Detecting APTs in Massive Networks via Multi-Stage Causal Analytics," in *International Conference on Computer Communications (INFOCOM'25)*. IEEE, 2025.

[91] M. U. Rehman, H. Ahmadi, and W. U. Hassan, "Flash: A Comprehensive Approach to Intrusion Detection via Provenance Graph Representation Learning," in *Symposium on Security and Privacy (S&P'24)*. IEEE, 2024.

[92] B. Jiang, T. Bilot, N. El Madhoun, K. Al Agha, A. Zouaoui, S. Iqbal, X. Han, and T. Pasquier, "Orthrus: Achieving high quality of attribution in provenance-based intrusion detection systems," in *Security Symposium (USENIX Sec'25). USENIX*, 2025.

[93] B. Manral, G. Somani, K.-K. R. Choo, M. Conti, and M. S. Gaur, "A systematic survey on cloud forensics challenges, solutions, and future directions," *ACM Computing Surveys (CSUR)*, 2019.

Appendix A. Availability

Code and weight

Our code and the instructions to reproduce our results can be found here: https://github.com/TristanBilot/lares.

Preprocessed datasets

We note that varying preprocessing steps and labeling across studies [3]–[6] result in inconsistent comparisons, making exact reproduction challenging. We provide preprocessed versions of the LANL and OpTC datasets with the labels used in this paper to help future reproduction efforts: https://drive.google.com/drive/folders/1tWf4B1o5zkw6u0xJ1QcYbzsMySjKFoI2?usp=sharing.

Appendix B. Prior Evaluation

Table 4 shows the baselines used for the evaluation of state-of-the-art systems [3]–[5], [21].

Method	Baselines
EULER	VGRNN, EGCN [82], GCN, SAGE, GAT, GC-LSTM [83]
ARGUS	EULER, VGRNN [84], PIKACHU, Netwalk, local outlier factor [85], isolation forest [86]
JBEIL	EULER, GraphSAGE
LARES (ours)	EULER, ARGUS, JBEIL, 1-class SVM

TABLE 4: Baselines Used by Each System

Table 5 shows the datasets used to evaluate each system. We exclude the Pivoting dataset [87] as it contains only benign pivoting events, not the malicious lateral movements we aim to detect.

System	Datasets
EULER	LANL
ARGUS	LANL, OpTC
JBEIL	LANL, Pivoting
PIKACHU	LANL, OpTC

TABLE 5: Datasets Used by Evaluated Systems

System	Venue	Network-Level LM	Open Source	Included
EULER [4]	NDSS'22	✓	✓	✓
ARGUS [5]	S&P'24	✓	✓	✓
JBEIL [21]	S&P'24	✓	✓	✓
PIKACHU [6]	NOMS'22	✓	✓	✗[a]
HOPPER [72]	USENIX'21	✓	✗	✗[b]
NETHAWK [89]	CNS'24	✓	✗	✗[b]
NETGUARDIAN [90]	INFOCOM'25	✓	✗	✗[b]
PROGRAPHER [10]	USENIX'23	✗[c]	✓	✗
FLASH [91]	S&P'24	✗[c]	✓	✗
KAIROS [12]	S&P'24	✗[c]	✓	✗
ORTHRUS [92]	USENIX'25	✗[c]	✓	✗

[a]Outperformed by EULER/ARGUS [4], [5]. [b]Code unavailable. [c]Provenance-based (host-level, different scope).

TABLE 6: Baseline selection. We include systems from top-tier venues (2021–2025) that target network-level LM detection and provide open-source implementations.

TABLE 7: Node-level detection results on LANL and OpTC datasets.

		LANL						OpTC					
Method	Exp	TP	FP	TN	FN	Recall	Precision	MCC	TP	FP	TN	FN	Recall	Precision	MCC
EULER	Exp0	1	23	13.2k	0	1.00	0.04	0.20	1	23	1088	2	0.33	0.04	0.11
	Exp1	1	22	13.2k	0	1.00	0.04	0.21	1	20	1091	2	0.33	0.05	0.12
	Exp2	1	22	13.2k	0	1.00	0.04	0.21	1	25	1086	2	0.33	0.04	0.11
	Exp3	1	25	13.2k	0	1.00	0.04	0.20	1	26	1085	2	0.33	0.04	0.10
$\text{EULER}_{\text{SAGE}}$	Exp0	1	79	13.1k	0	1.00	0.01	0.11	1	34	1077	2	0.33	0.03	0.09
	Exp1	1	76	13.1k	0	1.00	0.01	0.11	1	34	1077	2	0.33	0.03	0.09
	Exp2	1	89	13.1k	0	1.00	0.01	0.11	1	57	1054	2	0.33	0.02	0.07
	Exp3	0	84	13.1k	1	0.00	0.00	0.00	1	68	1043	2	0.33	0.01	0.06
$\text{EULER}_{\text{GAT}}$	Exp0	1	26	13.2k	0	1.00	0.04	0.19	1	33	1078	2	0.33	0.03	0.09
	Exp1	1	26	13.2k	0	1.00	0.04	0.19	1	34	1077	2	0.33	0.03	0.09
	Exp2	1	28	13.2k	0	1.00	0.03	0.19	1	39	1072	2	0.33	0.03	0.08
	Exp3	1	19	13.2k	0	1.00	0.05	0.22	1	39	1072	2	0.33	0.03	0.08
ARGUS	Exp0	1	39	13.1k	0	1.00	0.02	0.16	2	18	1093	1	0.67	0.11	0.25
	Exp1	1	49	13.1k	0	1.00	0.02	0.14	2	17	1094	1	0.67	0.11	0.26
	Exp2	1	45	13.1k	0	1.00	0.02	0.15	2	25	1086	1	0.67	0.07	0.22
	Exp3	0	44	13.1k	1	0.00	0.00	0.00	0	17	1094	3	0.00	0.00	0.00
$\text{ARGUS}_{\text{SAGE}}$	Exp0	0	100	13.1k	1	0.00	0.00	0.00	1	73	1038	2	0.33	0.01	0.06
	Exp1	0	101	13.1k	1	0.00	0.00	0.00	1	67	1044	2	0.33	0.01	0.06
	Exp2	0	122	13.1k	1	0.00	0.00	0.00	1	89	1022	2	0.33	0.01	0.05
	Exp3	0	109	13.1k	1	0.00	0.00	0.00	0	90	1021	3	0.00	0.00	0.00
$\text{ARGUS}_{\text{GAT}}$	Exp0	1	25	13.2k	0	1.00	0.04	0.20	2	14	1097	1	0.67	0.13	0.28
	Exp1	1	16	13.2k	0	1.00	0.07	0.24	2	15	1096	1	0.67	0.12	0.28
	Exp2	0	13	13.2k	1	0.00	0.00	0.00	2	12	1099	1	0.67	0.14	0.31
	Exp3	0	12	13.2k	1	0.00	0.00	0.00	1	13	1098	2	0.33	0.07	0.15
JBEIL	Exp0	1	376	12.8k	0	1.00	0.00	0.05	0	329	782	3	0.00	0.00	0.00
	Exp1	0	329	12.9k	1	0.00	0.00	0.00	0	218	893	3	0.00	0.00	0.00
	Exp2	0	341	12.8k	1	0.00	0.00	0.00	0	257	854	3	0.00	0.00	0.00
	Exp3	0	407	12.8k	1	0.00	0.00	0.00	0	299	812	3	0.00	0.00	0.00
OC-SVM	Exp0	0	42	13.1k	1	0.00	0.00	0.00	2	29	1082	1	0.67	0.07	0.20
	Exp1	0	33	13.1k	1	0.00	0.00	0.00	1	37	1074	2	0.33	0.03	0.09
	Exp2	0	33	13.1k	1	0.00	0.00	0.00	1	37	1074	2	0.33	0.03	0.09
	Exp3	0	38	13.1k	1	0.00	0.00	0.00	1	37	1074	2	0.33	0.03	0.09
LARES	Exp0	1	1	13.2k	0	1.00	0.50	0.71	2	1	1110	1	0.67	0.67	0.67
	Exp1	1	1	13.2k	0	1.00	0.50	0.71	2	1	1110	1	0.67	0.67	0.67
	Exp2	1	1	13.2k	0	1.00	0.50	0.71	2	1	1110	1	0.67	0.67	0.67
	Exp3	1	1	13.2k	0	1.00	0.50	0.71	2	1	1110	1	0.67	0.67	0.67

Appendix C. Datasets

We evaluate the performance of LARES and baseline methods on two well-established datasets, commonly adopted in recent state-of-the-art LM detection studies [3]–[6], [21].

LANL. The LANL Comprehensive, Multi-Source Cyber-Security Events dataset [50] is a large-scale collection of anonymized security event data from the Los Alamos National Laboratory's enterprise network. It includes diverse data types such as authentication logs, process execution records, DNS queries, and network flow data. Collected over a 58-day period, the dataset captures both normal operational activity and a simulated red team campaign, comprising 1.05 billion authentication events labeled as either benign or malicious across a real-world network of 17,684 machines. Owing to its realistic setting and the availability of ground-truth labels, the dataset has been widely used in cybersecurity research [19]. It is particularly well-suited for studying lateral movement in APT scenarios, as it contains labeled malicious events corresponding to such behavior. APT campaigns often involve abnormal authentication patterns as attackers move laterally across a system. These patterns serve as valuable indicators of compromise at the network level [72], [88].

DARPA OpTC. The DARPA OpTC dataset [65], developed as part of the DARPA Transparent Computing (TC) program, is a large-scale repository of host-level telemetry data aimed at enhancing system transparency and facilitating the detection of APTs. It comprises 1.1 TB of data and 17.4 billion events collected over one week from a real network of more than 1,000 machines, encompassing four days of benign activity and three days of coordinated attack campaigns. Each day of the attack campaign features a distinct scenario, incorporating malicious events at both the system and network levels. The dataset captures both host-level activities and network flow information. In our setting, we focus exclusively on network events to construct input graphs, and the task is framed around detecting lateral movement.

Appendix D. Baseline Selection

Baselines are selected following these criterions (Table 6): published in top-tier venues, targeting network-level lateral movement, and providing open-source implementations for reproducible comparison.

Appendix E. Matthews Correlation Coefficient

The Matthews Correlation Coefficient (MCC) [75] is used to assess the quality of binary classifications.

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{6}$$

Appendix F. Node-level Detection Results

We report node-level detection results in Table 7. For LARES, these correspond to the first stage of the two-stage detection process, which focuses on identifying the source hosts of LMs. For the baselines, we adapt their edge-level outputs to the node level by applying the same strategy used in LARES, assigning each host a score based on the highest-scoring outgoing edge (see §3.5).

F.1. Transductive Detection Performance

LANL. Baselines like EULER and ARGUS detect the single malicious root node but suffer from low precision, with EULER producing 23 false positives and ARGUS 39. Variants like SAGE and GAT show minimal improvement, except for $\text{ARGUS}_{\text{GAT}}$, which reaches 4% precision. In contrast, LARES

TABLE 8: Detailed complexity analysis of LARES.

Layer	Complexity
GNN Encoder	$\mathcal{O}\left(\lvert N\rvert d^2 + \lvert E\rvert d^2\right)$
Edge Encoding Layer	$\mathcal{O}\left(\lvert E\rvert \left(d_e^2 + d.d_e + d\right)\right)$
Total Encoder	$\mathcal{O}\left(\left(\lvert E\rvert + \lvert N\rvert\right) d^2 + \lvert E\rvert d.d_e\right)$
Total Decoder	$\mathcal{O}\left(\lvert E\rvert d^2\right)$
Total LARES	$\mathcal{O}\left(\left(\lvert E\rvert + \lvert N\rvert\right) d^2 + \lvert E\rvert d.d_e\right)$

detects the malicious host with only one FP, achieving 50% precision.

OpTC. EULER and its variants detect one of three malicious nodes, while ARGUS and $\text{ARGUS}_{\text{GAT}}$ detect two, though with low precision (11% and 13%, respectively). LARES also detects two TPs with just one FP, missing the third node but achieving 67% precision.

F.2. Inductive Detection Performance

LANL. Most baselines achieve 100% recall in `Exp1` and `Exp2` by detecting the single root node, but only EULER and its GAT variant succeed in `Exp3`. EULER and ARGUS show low precision (peaking at 4% and 2%), with GAT offering no clear gain and SAGE underperforming. OC-SVM detects no TPs, indicating edge features alone are insufficient in this dataset. Conversely, LARES' inductive design detects the malicious host with only a single FP among 13.2k hosts. In particular, results in `Exp3` show that LARES sustains accuracy despite training on half the network, excluding the malicious host.

OpTC. EULER and its variants detect one of three attack root nodes with low precision (up to 5%). ARGUS outperforms EULER in `Exp1` and `Exp2` but fails in `Exp3`. $\text{ARGUS}_{\text{GAT}}$ improves detection across all experiments, unlike the SAGE variants.

JBEIL fails to detect any hosts with acceptable precision on both datasets. LARES consistently detects 2/3 TPs with just one FP in all experiments, showing strong performance in inductive scenarios.

F.3. Practical Implication

In deployment, nodes flagged as malicious are typically isolated from the network and subjected to forensic analysis [93] to determine the nature and impact of the compromise. This process is complex and requires significant time and effort from well-trained employees. Previous systems, due to their high false positive rates, generate excessive alerts that lead to unnecessary investigations.

Appendix G. Complexity Analysis

In Table 8, we present a detailed complexity analysis of LARES, accounting for various parameters such as $|N|$ and $|E|$, the number of nodes and edges, where d is the node embedding size and d_e is the edge feature size. Both the encoder and decoder scale linearly with the number of edges and scale quadratically with node embedding size. Only the encoder scales with $|N|$, as the decoder performs operations at the edge level exclusively. Overall, the architecture exhibits no distinct bottlenecks, though the node embedding size requires careful management due to its quadratic impact on complexity.