

R+R: From Claims to Crashes: A Systematic Re-evaluation of Graph-Based Network Intrusion Detection Systems

Chenglong Wang^{†,§}, Pujia Zheng^{†,§}, Jiaping Gui^{†,*}, Cunqing Hua[†], Wajih Ul Hassan[‡]

[†]Shanghai Jiao Tong University, Shanghai, China

[‡]University of Virginia, Charlottesville, VA, USA

Email: {wangchenglong25, zhengpujia, jgui, cqhua}@sjtu.edu.cn, hassan@virginia.edu

Abstract—Graph-based Network Intrusion Detection Systems (GIDS) are increasingly used to model complex communication patterns and detect sophisticated enterprise threats, yet the reproducibility and replicability of GIDS research remain underexplored, limiting the reliability and generalizability of published results. We present a rigorous reproduction and replication of five state-of-the-art GIDS across four public datasets and a new large-scale enterprise dataset. Even with original code and configurations, reproducing claimed performance is difficult; detection metrics vary by up to 40 percent due to undocumented assumptions, preprocessing discrepancies, and hyperparameter sensitivity. Models also fail to generalize to real-world enterprise traffic, exhibiting high false positive rates and scalability issues. We identify key implementation factors: graph snapshot size and threshold-setting strategies significantly affect detection performance but are inconsistently documented, and several GIDS are vulnerable to evasion attacks. Beyond confirming known challenges (e.g., parameter sensitivity), our results expose a critical reproducibility crisis in the GIDS literature: without transparent and systematic evaluation, reported results may mislead researchers and practitioners. We provide recommendations to improve reproducibility, replicability, and robustness, and urge the community to adopt rigorous standards for empirical evaluation.

1. Introduction

Modern cyber threats are increasingly sophisticated, often exploiting unknown vulnerabilities to breach enterprise and government systems. These long-term, stealthy attacks aim to extract sensitive data and have caused major economic damage, as seen in breaches at Yahoo [1], Marriott [2], and Home Depot [3]. To counter these threats, Network Intrusion Detection Systems (NIDS) play a key role by monitoring network traffic logs and alerting analysts to suspicious behavior.

Traditional NIDS, such as Zeek and Snort, use static rules to detect anomalies [4–8]. Although interpretable and

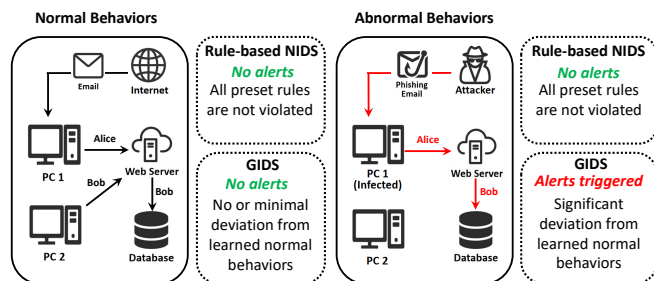


Figure 1: GIDS vs. rule-based NIDS

efficient, these systems often miss novel or zero-day attacks, leading to high false negative rates. They also generate many false positives due to vague or overly broad rules, which causes alert fatigue [9–14] and reduces their practical effectiveness.

To complement rule-based systems, Graph-based NIDS (GIDS) have been developed to model complex relational patterns in network traffic [15–23]. GIDS construct directed graphs from network communications, where nodes represent machines and edges represent communication flows. They utilize graph encoders to generate embeddings of nodes and edges and often employ temporal encoders to capture dynamic behavior over time. By learning typical traffic patterns, GIDS can detect deviations indicative of suspicious behavior [24]. For example, a GIDS may identify a slow and stealthy data exfiltration attack that spreads across multiple machines and occurs over a long period—a pattern that may not trigger any single rule in a traditional NIDS. Figure 1 illustrates such a scenario. While GIDS show promise in detecting such advanced threats, they are not a silver bullet and are best deployed as a complementary layer alongside traditional NIDS [25, 26]. Enterprises often retain rule-based systems for their transparency, low resource consumption, and generally low false positive rates for known threats [27].

Our study aims to tackle the R+R issues in enterprise security research by systematically evaluating state-of-the-art (SOTA) GIDS. Our study seeks to confirm, question, and clarify the results of previous research, ensuring their validity and reliability across various contexts. This is crucial for developing robust and generalizable detection systems capable of effectively combating evolving security

[§] Equal contribution.

^{*} Corresponding author.

This work was supported by the Natural Science Foundation of Shanghai (23ZR1429600).

threats. We conduct a comprehensive assessment of these systems on both public datasets and a newly collected enterprise dataset, measuring their detection performance, efficiency from both temporal and spatial perspectives, and robustness under adversarial attacks. To study existing GIDS, we delineate our research questions into three pivotal domains: Implementation, Evaluation, and Deployment.

IMPLEMENTATION. In the implementation domain, our focus is on the intricacies of GIDS re-implementations and their impact on system performance. Specifically, *RQ1: What are the key factors that influence the re-implementations of GIDS?* This involves a meticulous analysis of various aspects to ensure accurate experimental settings, including the optimal tuning of hyperparameters. This domain emphasizes the necessity of understanding how different configurations and hyperparameters affect the overall effectiveness of the system, enabling us to identify the most impactful settings for developing a functional and optimal system.

EVALUATION. In the evaluation domain, we scrutinize the operational effectiveness and efficiency of GIDS across diverse datasets. *RQ2: How do the state-of-the-art models perform on established public datasets?* This entails a rigorous reevaluation of models on benchmark datasets, focusing on their reproducibility and replicability. *RQ3: How do these models generalize to new datasets derived from real-world enterprise environments?* This question assesses the adaptability and scalability of these models when confronted with data from a newly curated large-scale enterprise network, comparing their performance on synthetic versus real-world data. This evaluation provides insights into how these models can be optimized for better generalizability and applicability across various scenarios.

DEPLOYMENT. In the deployment domain, we address the practical aspects of deploying GIDS in operational settings. *RQ4: How do these models perform from both temporal and spatial perspectives in a production environment?* This question evaluates the scalability and efficiency of the models when processing extensive network traffic data, a critical consideration for enterprise-level deployments. *RQ5: How resilient are these models to adversarial attacks?* This involves stress-testing the models against sophisticated adversarial perturbations to determine their robustness and reliability in real-world attack scenarios. Understanding the robustness and scalability of these models is crucial for ensuring their effective integration into existing security infrastructures.

We evaluate five representative systems: Anomal-E [28], VGRNN [29], PIKACHU [16], EULER [17], and ARGUS [19]. These systems were selected based on the following criteria: (1) **Open-source availability:** Systems with publicly available code and datasets were prioritized to ensure reproducibility and transparency. (2) **Architectural diversity:** The selected models cover a range of graph-based techniques, including variational autoencoders (EULER), temporal graph convolutional networks (PIKACHU),

message-passing networks (ARGUS), and variational graph RNNs (VGRNN). (3) **Relevance:** These systems focus on unsupervised/self-supervised methods and have demonstrated state-of-the-art performance in prior work. We evaluate these systems on LANL [30], DARPA OpTC [31], CIC-IDS-2017 [32] and CTU-13 [33], as well as a new enterprise dataset that captures diverse, real-world traffic and attack patterns. Our study not only confirms and questions results reported by prior GIDS papers, but also clarifies undocumented assumptions, reveals key generalization failures on enterprise traffic, and identifies reproducibility gaps critical for real-world adoption.

Our experimental results reveal substantial reproducibility differences across systems. In the implementation domain, we found that parameters such as snapshot time window, embedding dimension, and detection model significantly influence detection efficiency. Proper tuning of these parameters improves performance. In the evaluation domain, models showed more stable results on the LANL dataset, while discrepancies on other datasets stemmed from suboptimal settings and preprocessing differences. Replication improved results, but performance dropped considerably on both the CIC-IDS-2017 and our large-scale enterprise dataset, highlighting generalization challenges and elevated false positive rates. In the deployment domain, VGRNN and ARGUS achieved the best space efficiency, scaling to 70K nodes, whereas PIKACHU failed on large datasets due to memory and runtime constraints. Finally, VGRNN, EULER, and ARGUS were vulnerable to evasion attacks, especially on the LANL dataset. We conclude with recommendations to improve the reproducibility and replicability of GIDS.

We summarize our contributions as follows.

- We collect and release a large-scale, real-world enterprise dataset (over 10B events across 101 days), enabling realistic evaluation of GIDS beyond synthetic and academic settings.
- We design a robust evaluation framework that supports both artifact reuse and re-implementation to systematically assess reproducibility and replicability.
- We conduct the first extensive cross-system analysis of five SOTA GIDS on public and enterprise datasets, highlighting discrepancies in detection accuracy, generalization, and resource efficiency.
- We assess robustness under evasion attacks and show that all evaluated GIDS are vulnerable to minimal perturbations, revealing critical security gaps.
- We provide actionable recommendations for future GIDS development and release the source code of this project at <https://github.com/wcl-sjtu/GIDSREP>.

2. Background

2.1. Graph-based IDS: GIDS

In enterprise environments, network traffic is routinely audited and stored in logs that capture user behaviors.

TABLE 1: Comparison of SOTA GIDS.

System	Graph Type	Node Emb.	Edge Emb.	Graph Encoder	Temporal Encoder	Streaming	Supervised	Datasets Used In The Paper	Open Source
NETWALK [15] (KDD 2018)	Static	Yes	No	Network Walk Autoencoder	No	Yes	No	UCI Messages, Digg, arXiv hep-th, DBLP	Yes
VGRNN [29] (NeurIPS 2019)	Dynamic	Yes	No	GCN	GRNN	Yes	No	Cora, Citeseer, Pubmed	Yes
Anomal-E [28] (KBS 2022)	Static	Yes	Yes	GraphSAGE	No	No	No	NF-UNSW-NB15-v2, NF-CSE-CIC-IDS2018-v2	Yes
E-GraphSAGE [34] (NOMS 2022)	Static	Yes	Yes	GraphSAGE	No	Yes	Yes	ToN-IoT, NF-TON-IoT, BoT-IoT, NF-BoT-IoT	Yes
EULER [17] (TOPS 2023)	Dynamic	Yes	No	GNN	GRU / LSTM	Yes	No	LANL, OpTC	Yes
PIKACHU [16] (NOMS 2022)	Dynamic	Yes	No	Random Walk + Skip-gram	GRU	No	No	LANL, OpTC	Yes
ARGA [35] (ACM 2023)	Static	Yes	No	GAE	No	Yes	No	CTU-13, ToN-IoT	No
[36] (ARES 2023)	Static	Yes	No	GraphSAGE	No	Yes	Yes	TON-IoT	No
Jbeil [37] (IEEE S&P 2024)	Dynamic	Yes	Yes	TGN	GRU	Yes	No	LANL, Pivoting	Yes
ARGUS [19] (IEEE S&P 2024)	Dynamic	Yes	Yes	MPNN	GRU	Yes	No	LANL, OpTC	Yes
K-GetNID [38] (IEEE TIFS 2024)	Dynamic	Yes	Yes	GNN	T-GNN	Yes	Yes	CIC-IDS-2017, UNSW-NB15	No
PNCG [39] (IEEE TII 2024)	Dynamic	Yes	Yes	MPNN	Bi-LSTM	Yes	Yes	ToN-IoT, BoT-IoT, UNSW-NB15, DoHBrw2020	No
E-GRACL [40] (J SUPERCOMPUT 2024)	Static	Yes	Yes	GraphSAGE	No	Yes	No	NF-BoT-IoT-v2, NF-ToN-IoT-v2, NF-CSE-CIC-IDS2018-v2	No

* The “Streaming” column indicates whether predictions can be performed incrementally on ingested network traffic data.

* Gray-highlighted rows represent systems evaluated in this study. NETWALK was excluded due to prohibitive execution time (48 hours per run on LANL), making it impractical for real-world deployment, Jbeil due to reimplementing challenges, despite reaching out to the authors and receiving no response. Supervised learning methods such as E-GraphSAGE require extensive labeled datasets for training; however, this is impractical in large-scale enterprise environments, and for non-open-source approaches such as ARGA and K-GetNID, we have also contacted the respective authors but received no response.

These logs consist of discrete events, where an event typically represents a network flow or a distinct connection record between two hosts, encapsulating information such as source/destination IPs, ports, timestamps, and protocol. Security analysts analyze these events to detect adversarial actions [41]. However, as networks grow, traffic volume increases dramatically, often reaching terabytes per day in medium-sized enterprises, making attack detection increasingly challenging. Traditional NIDS rely on rule-based detection, leading to high false negative rates and missed anomalies.

To address these limitations, GIDS have emerged as a key solution. Unlike traditional systems, GIDS autonomously learn behavioral patterns by constructing directed graphs from network communications, where nodes represent machines and edges represent communication flows. They use graph encoders to embed node and edge features and temporal encoders to capture dynamic behavior over time. Graph Neural Networks (GNNs) are typically employed for both encoding tasks [16, 29, 37, 42]. By modeling normal traffic behavior, GIDS flag deviations as suspicious, demonstrating strong detection performance even against zero-day exploits and offering greater adaptability than rule-based NIDS [17, 19, 24]. Table 1 summarizes recent GIDS systems. The selection was based on a systematic survey until October 2024, prioritizing state-

of-the-art models with diverse architectures.

2.2. Comparison with HIDS

While Graph-based IDS (GIDS) focus on network-level intrusion detection, it is important to differentiate them from Host-based IDS (HIDS), which operate at different abstraction levels and with distinct data sources.

HIDS monitor activities on individual hosts or endpoints, analyzing system logs, file system changes, and application behaviors. High-profile evaluation frameworks such as DARPA E3 [43] and E5 [44] have significantly advanced HIDS research by providing comprehensive host-level audit data. However, these datasets consist primarily of system logs designed for host-centric detection. In contrast, our work focuses on NIDS using network logs (e.g., NetFlow, Zeek logs). The domains differ fundamentally in data granularity, threat models, and methodologies. Network logs capture macro-level interactions between hosts across the enterprise, enabling detection of lateral movement and command-and-control communications, but at a coarser granularity compared to host-level audit trails.

HIDS typically employ provenance-based analysis [12, 45–54] to model system behavior using fine-grained audit logs (e.g., system calls), constructing process-level causality graphs. These systems excel at detecting attack provenance

by tracing the lineage of malicious activities through detailed dependency chains. In contrast, GIDS utilize network communication graphs, where hosts are nodes and communication flows are edges. This approach is designed for macro-level network interaction analysis, which is more scalable for enterprise-wide detection but faces distinct challenges in temporal dynamics and robustness against adversarial perturbations.

2.3. Challenges in R+R of GIDS

GIDS have demonstrated outstanding performance on public datasets, showing great potential in cybersecurity defense [15–20, 23]. However, the reproducibility of these results remains unexplored, making it difficult to assess their validity and reliability in different contexts, especially new scenarios.

A critical limitation lies in the narrow scope of public datasets used for evaluation. While public datasets are representative of real-world environments, significant disparities exist compared to actual industry settings. The main differences include the larger scale of network traffic, more complex network structures, and a wider array of network threats faced by enterprises, which are not fully captured in datasets like LANL and OpTC. Additionally, in real-world environments, NIDS must respond to attacks promptly with limited resources, processing massive amounts of data efficiently while maintaining high accuracy. The robustness of models is also crucial, as they must remain accurate and stable under hostile conditions.

To bridge this gap, we leverage a new dataset collected from an anonymous enterprise to evaluate the adaptability of existing GIDS to contemporary cybersecurity challenges. We have selected five representative systems for evaluation, as they represent recent advancements and have consistently showed high performance in detection accuracy, scalability, and computational efficiency, demonstrating strong performance in detecting complex and multi-stage attacks. We re-evaluate these models on both public datasets and our own dataset in terms of effectiveness and efficiency.

3. Approach: GIDSREP

We formalize our evaluation methodology as an open, reproducible framework named GIDSREP, with its workflow illustrated in Figure 2. Given raw data as input, GIDSREP generates experimental results through four modules: (1) the Data Processing Module, which cleans, formats, and partitions data for each model; (2) the Detection Assessment Module, which evaluates detection performance using default and tuned hyperparameters; (3) the Robustness Assessment Module, which tests model resilience against adversarial attacks; and (4) the Efficiency Assessment Module, which measures space and time performance. Each module is described in detail below.

DATA PROCESSING MODULE. This module begins by loading raw data from public datasets (LANL, OpTC,

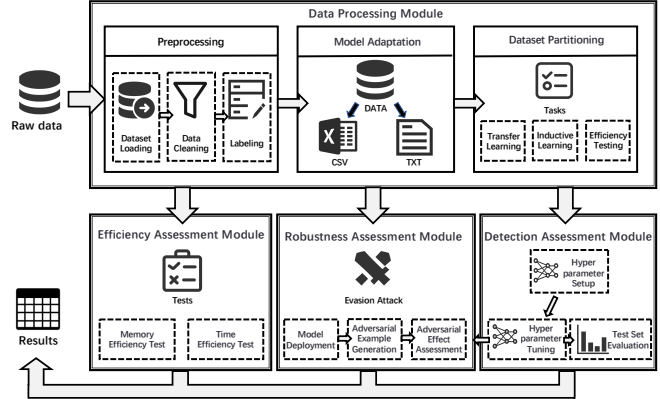


Figure 2: Workflow of GIDSREP

CIC-IDS-2017, CTU-13) and a newly collected large-scale enterprise dataset. Real-world data often contains noise or errors, so we apply preprocessing steps such as filtering invalid entries, removing outliers, and smoothing time-related data to ensure clean inputs. Each event is labeled as normal or malicious. Public datasets use red team documentation or built-in labels, while the enterprise dataset is labeled using simulated attacks with recorded timestamps and corresponding traffic logs. To enable efficient validation and testing, the enterprise dataset is split into smaller subsets, allowing optimal model parameter selection without high computational cost. Since datasets differ in format, we convert them according to model requirements. For instance, EULER and ARGUS use batched text files, while PIKACHU expects CSV files. Finally, for inductive learning and efficiency evaluations, the training set includes all logs prior to the first attack timestamp, ensuring the model is trained only on pre-attack data.

DETECTION ASSESSMENT MODULE. We begin by replicating each model’s original experimental setup, reusing the published hyperparameters and dataset configurations to ensure consistency with prior results. This establishes a reproducibility baseline. Next, we apply grid search to explore a range of hyperparameter values and identify the optimal configuration on the testing dataset, aiming to improve detection accuracy while minimizing false positives and false negatives. This tuning step is essential for enhancing model performance beyond its default settings. Finally, we evaluate each model using standard metrics (see Section 4) under its optimized configuration. For the large-scale enterprise dataset, we directly evaluate on the testing set to reflect real-world deployment scenarios and assess practical performance.

ROBUSTNESS ASSESSMENT MODULE. Through simulating white-box adversarial attacks, we apply perturbations aimed at evading detection to the testing set. These perturbations are designed to challenge the model’s ability to detect threats and test its resilience under adversarial conditions. Utilizing the optimal parameter model obtained from the Detection Assessment Module, we conduct evasion attack testing. This allows us to evaluate the model’s performance in detecting

TABLE 2: Statistics of four public datasets and our newly collected large-scale enterprise dataset.

Dataset	Scenario	# Hosts/IPs	# Events	Duration (days)
LANL		17,649	1,051,430,459	58
DARPA OpTC		814	10,109,338	8
CIC-IDS-2017		19,060	2,099,976	5
CTU-13	neris_1	607,565	2,824,639	1
	neris_2	367,263	2,753,884	1
Our Dataset		18,425,098	10,338,002,425	101

attacks, even when subjected to adversarial perturbations. The goal is to assess how effectively the model maintains its detection accuracy under conditions designed to deceive it. This process helps identify potential vulnerabilities within the model and highlights areas that need improvement, particularly with regard to robustness. By understanding the model’s weaknesses in the presence of adversarial interference, we can develop strategies for enhancing its defense mechanisms and ensure that it remains reliable and effective in real-world, adversarial environments.

EFFICIENCY ASSESSMENT MODULE. During the entire training and testing phases, we monitor memory usage closely. By incrementally increasing the number of nodes (hosts) in the dataset, we determine the maximum scale the model can handle before encountering out-of-memory (OOM) errors. This helps identify the limits of the model’s scalability and resource requirements. Additionally, we record both the training and testing durations to assess how efficiently the model utilizes computational resources. This timing data provides valuable insight into the computational efficiency of each model. Comparing models based on these metrics highlights the differences in speed and scalability, emphasizing those models that perform well under resource constraints. It also helps assess the feasibility of deploying each model in real-world enterprise environments, where computational resources may be limited. This evaluation ensures that we select models that not only deliver high performance but also operate efficiently in practical settings.

4. Evaluation Results

In this section, we discuss the details of the experiments we conducted to address each of the RQs defined in Section 2. For each RQ, we describe the approach we used to capture the relevant metrics, present the results we obtained, and discuss the implications of these results with respect to each of the RQs.

ENVIRONMENT SETTINGS. We conducted our experiments on a workstation equipped with an Intel i9-14900K 32-core processor and 128 GB of RAM, running the Ubuntu 22.04.4 LTS operating system. RQ1 to RQ4 were conducted using the CPU, while RQ5 used the GPU. Our GPU was an NVIDIA GeForce RTX 4090 with 24 GB of VRAM. The runtime environments for the models were consistent with those in the original papers.

DATASET PREPARATION. To evaluate SOTA GIDS, we use four public datasets and one new enterprise dataset.

Key characteristics are summarized in Table 2. The **LANL** dataset contains 58 days of genuine human-made internal corporate traffic. Its red team attacks are also human-conducted, simulating realistic advanced persistent threat (APT) scenarios like lateral movement. The **OpTC** dataset includes one week of logs from 1,000 machines. Both benign activity and attacks are synthetically generated by the **FLARE** program to emulate a government agency network and MITRE ATT&CK techniques. The **CIC-IDS-2017** dataset spans five days in a lab environment. Benign traffic is synthetically generated to mimic human user protocols, while attacks (e.g., DoS, Brute Force) are executed via scripts. We use its improved version from [55]. The **CTU-13** dataset is a lab-captured botnet dataset. Benign traffic synthetically emulates an SME network. We select the two longest Neris botnet attacks following [35] and sample background traffic to 20K nodes for scalability¹.

Additionally, we assess generalizability on a **newly collected dataset** comprising over 10 billion events from a real enterprise network over 101 days. The benign traffic is entirely human-made, reflecting authentic enterprise behavior. We simulate three APT attacks (Sandworm, Wizard Spider, OilRig) in a controlled domain environment following MITRE guidelines, and merge the attack traffic as in [57]. Attack statistics are provided in Appendix Table 6. A detailed description of the datasets is included in Appendix B.

EVALUATION METRICS. Following prior work [17, 19], we define edges with at least one malicious event as true positives (TP) and those with only normal events as true negatives (TN). False positives (FP) and false negatives (FN) refer to edges incorrectly classified as malicious and normal, respectively. For LANL and OpTC, malicious events are extracted from red-team documentation. In our enterprise dataset, attacks were simulated with recorded start/end times and associated traffic, excluding benign activity during execution to ensure precise labeling. All datasets underwent manual verification for labeling completeness. We assess model performance using five metrics: true positive rate (TPR), false positive rate (FPR), precision, average precision (AP), and area under the ROC curve (AUC), which plots TPR against FPR across thresholds. Specifically, $TPR = \frac{TP}{TP+FN}$, $FPR = \frac{FP}{FP+TN}$, $Precision = \frac{TP}{TP+FP}$, and $AP = \sum_{\tau} (R_{\tau} - R_{\tau-1})P_{\tau}$, where R_{τ} and P_{τ} denote recall and precision at threshold τ .

RQ1: What are the key factors that influence the re-implementations of GIDS?

REPRODUCIBILITY & REPLICATION SETUP. To address this research question, we leveraged the Detection Assessment Module to fine-tune the implementation parameters of the target models and evaluated them using

1. Several other network attack datasets exist, such as CIC-IDS-2018 [32], ToN-IoT [56], BoT-IoT [56], and UNSW-NB15 [56]. However, we did not select these datasets due to their limitations. Specifically, they either mix attack and normal behaviors based on timestamps or contain over 90% attack traffic, which makes them unsuitable for training models in the target GIDS.

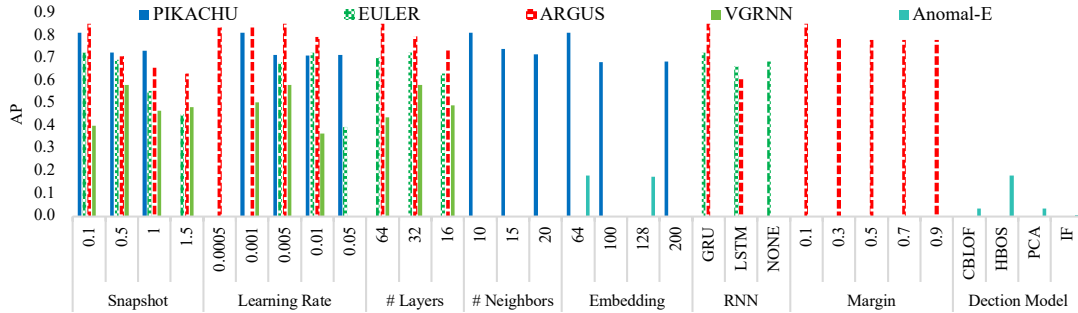


Figure 3: Impact of key implementation parameters on AP. Notice that some models lack results corresponding to certain parameters due to the absence of those parameters. Table 7 in the appendix explains each parameter on the X axis.

the OpTC dataset. We focused on the OpTC dataset because it is the common dataset utilized in the original papers of the target models. For Anomal-E, we conducted experiments using CIC-IDS-2017, as OpTC does not possess flow features. Table 8 in the appendix shows the specific implementation parameters. We adopted the sum of AP and AUC as the criterion for parameter optimization. After obtaining the optimal settings, we assessed the detection efficacy of GIDS by adjusting a specific parameter while maintaining all other settings at their optimal values. Then, we meticulously analyzed the impact of the implementation parameters on the model’s performance, and identified those having a significant influence on the overall effectiveness of the target models as the key parameters.

RESULTS. Figure 3 shows the impact of key implementation parameters. For ease of comparison, we have also included the results of these parameters (if applicable) across models. We focus on two metrics, AP and AUC, which are utilized for assessing the performance of the models in their original papers.

OUTCOME ANALYSIS. Snapshot size significantly affects the AP performance of VGRNN, EULER, and ARGUS. As the time window increases, AP drops for EULER and ARGUS, while VGRNN’s AP initially rises before declining. This is because GNNs compress all events between two nodes into a single edge, erasing temporal dynamics. Larger snapshots increase edge/event compression, making it harder to distinguish attack events from benign ones. Smaller snapshots, while preserving event granularity, reduce contextual information and limit learned behavioral features. Thus, selecting an appropriate snapshot size is essential. Learning rate notably affects EULER but has little impact on other models. PIKACHU’s performance varies with embedding dimension, while Anomal-E is most influenced by the choice of anomaly detection model. These parameters, however, have minimal effect on AUC, as shown in Figure 6 in the appendix.

Finding: For GIDS, the size of the snapshot has a significant impact on their detection performance in terms of the AP score, while other parameters

exhibit varying degrees of influence. Nonetheless, these implementation parameters have minimal influence on the AUC metric. This observation aligns with existing works [16, 17, 19, 29] that suggest AP be a primary optimization goal for enhancing the detection efficacy of GIDS.

RECOMMENDATION. We recommend that future GIDS research (i) report not only final parameter values but also parameter sensitivity analyses to disclose how performance varies with hyperparameter drift, (ii) include complete, dataset-specific configuration files and threshold-setting procedures, and (iii) standardize evaluation using AP rather than AUC as the primary metric of comparison. Given that AUC remains relatively stable under re-implementation variance, while AP varies substantially, relying on AUC alone may mask reproducibility failures and overstate detection robustness.

RQ2: How do the state-of-the-art models perform on established public datasets?

REPRODUCIBILITY AND REPLICATION SETUP. To validate the evaluation results of existing GIDS, we conducted reproducibility and replication experiments on established public datasets. For reproduction, we adopted the original papers’ environmental settings and used the publicly available source code and datasets. On the LANL dataset, models followed different processing strategies: EULER used all NTLM-tagged events over 58 days, ARGUS used the first 14 days, and PIKACHU filtered out certain users (e.g., administrators) before sampling normal events. Since these preprocessed datasets are publicly available, we reused them directly. For the OpTC dataset, however, only the authors of ARGUS shared their preprocessing scripts; the authors of EULER and PIKACHU provided only the raw data, which could not be directly used. Thus, we used the ARGUS-preprocessed version for all models.

For replication, we re-evaluated models on the same datasets using fine-tuned parameters. For ARGUS and EULER, we tuned the number of GNN layers, snapshot size, epochs, learning rate, threshold weight, and patience. We also varied margin parameters for ARGUS and tested different RNN modules in EULER. For

TABLE 3: Performance comparison of target models and traditional NIDS on the CIC-IDS-2017 and CTU-13 datasets.

Attack	Type	Model	TPR	FPR	Prec.	AUC	AP
CIC-IDS 2017	GIDS	ARGUS_ft	0.290	0.439	0.001	0.448	0.001
		ARGUS	0.310	0.534	0.001	0.391	0.001
		EULER	0.299	0.236	0.001	0.510	0.006
		VGRNN	0.223	0.034	0.007	0.570	0.002
		PIKACHU	0.860	0.140	0.722	0.953	0.917
		Anomal-E	0.893	0.062	0.597	0.915	0.798
	NIDS	IF	0.527	0.192	0.536	0.729	0.433
		LOF	0.045	0.052	0.268	0.462	0.283
		OC-SVM	0.453	0.015	0.928	0.638	0.618
CTU-13 (neris_1)	GIDS	ARGUS_ft	0.977	0.095	0.734	0.964	0.769
		ARGUS	0.995	0.114	0.707	0.989	0.941
		EULER	0.982	0.073	0.779	0.984	0.910
		VGRNN	0.987	0.089	0.760	0.972	0.904
		PIKACHU	0.973	0.033	0.608	0.977	0.476
		Anomal-E	1.000	0.078	0.398	0.961	0.398
	NIDS	IF	0.009	0.011	0.045	0.911	0.226
		LOF	0.010	0.010	0.382	0.612	0.096
		OC-SVM	0.539	0.069	0.288	0.637	0.185
CTU-13 (neris_2)	GIDS	ARGUS_ft	0.642	0.195	0.818	0.847	0.852
		ARGUS	0.890	0.104	0.930	0.947	0.944
		EULER	0.978	0.097	0.941	0.958	0.979
		VGRNN	0.936	0.127	0.933	0.950	0.939
		PIKACHU	0.960	0.040	0.847	0.978	0.881
		Anomal-E	1.000	0.105	0.684	0.947	0.684
	NIDS	IF	0.026	0.010	0.320	0.893	0.521
		LOF	0.149	0.055	0.382	0.565	0.254
		OC-SVM	0.304	0.258	0.212	0.532	0.248

* In the table, "Prec." stands for precision, and "NIDS" traditional NIDS.
 * The highlighted cells represent the best results among columns.

PIKACHU, we tuned the embedding dimension, learning rate, and neighbor sampling. To comprehensively evaluate GIDS’s performance, we also incorporated three traditional unsupervised NIDS methods—Isolation Forest (IF) [58], Local Outlier Factor (LOF) [59], and OC-SVM [60]—into our assessment and optimized their hyperparameters. These optimal parameter settings are listed in Table 8 in Appendix D. Since Anomal-E requires edge features per communication, and the LANL and OpTC datasets lack statistically meaningful edge-level information, we evaluate Anomal-E and traditional NIDS on the CIC-IDS-2017 and CTU-13 datasets.

RESULTS. The results of R+R experiments on the LANL and OpTC datasets are shown in Figure 4. For ease of comparison, we have aggregated the experimental results and the original ones along the X axis. For each of the metrics, each model has three bars, presented from left to right as the reproduction, replication, and original results. The experimental results on the CIC-IDS-2017 and CTU-13 datasets are shown in Table 3.

OUTCOME ANALYSIS. Overall, the performance of PIKACHU, ARGUS and EULER is more consistent on the LANL dataset than on the OpTC dataset. Among the results, the performance difference in some metrics can be significant, even for the same model. Below we present a detailed analysis of the performance comparison.

In the reproduction experiments, the results of all metrics except AUC on the OpTC dataset showed significant

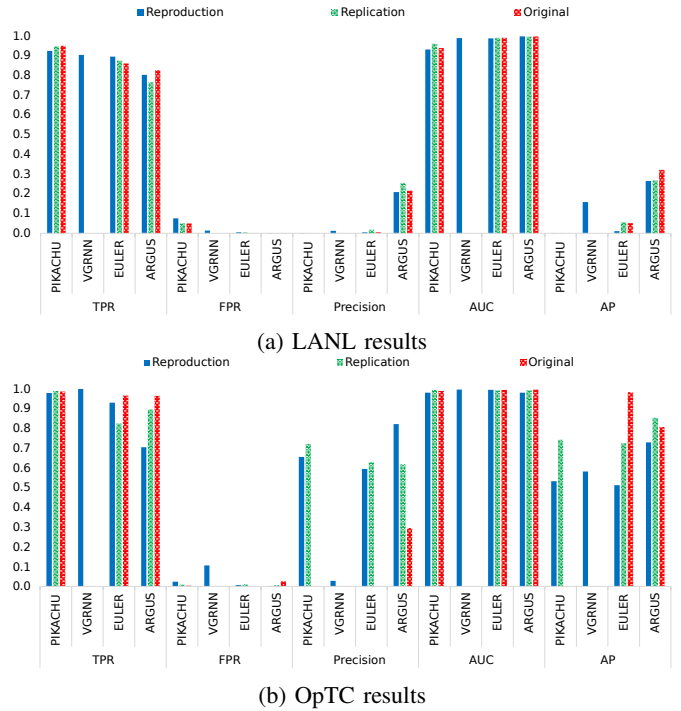


Figure 4: Performance comparison of target models on the LANL and OpTC datasets. Notice that some original results of VGRNN, PIKACHU and EULER are not plotted in the figure since they are not provided in their original papers.

differences with the original results. We identified multiple reasons that could contribute to such differences: 1) The experimental settings that are publicly available are not optimal, which is also reflected by the replication results. 2) The preprocessed dataset of ARGUS which we utilized as input for all models might differ from the ones used by PIKACHU and EULER in their original evaluations. 3) There is randomness in the model’s calculation of the detection threshold. In anomaly detection, the model generates a score for each edge based on learned behavioral features, automatically calculates a threshold using the validation set, and classifies edges with scores below this threshold as anomalies. However, EULER and ARGUS randomly select 5% of events in each snapshot as the validation set. Hence, the calculated threshold can vary significantly, leading to fluctuations in threshold-related evaluation metrics (TPR, FPR and precision). For example, in ARGUS, both TPR and FPR scores are lower than the original results.

The results also demonstrate that compared to other metrics, AUC is insufficient to assess R+R of GIDS. In addition, we observed different versions of EULER were evaluated in the original paper [17]. We further conducted reproduction experiments of these versions on the OpTC dataset, and found our results are consistent with those in the ARGUS paper, but significantly different from those in the original paper of EULER [17]. We outline more discussions on it in Appendix E.

In the replication experiments, considering the randomness in calculating detection thresholds, we focused more on the AUC and AP scores, which are independent of threshold calculation. We selected the parameters that yielded the highest AUC and AP scores as the best tuning results. As shown in Figure 4a, in the LANL dataset, the performance of the target models was relatively stable, and the optimal results from parameter tuning were close to the parameter settings in the original paper. However, in the OpTC dataset, although the results of the replication experiments still differed from the results in the original paper, they were significantly better than the reproduction results. This indicates that evaluating target models in different environments may require re-tuning the parameters.

For the experiments conducted on the CIC-IDS-2017 dataset, PIKACHU and Anomal-E exhibited good performance, while the remaining GIDS models performed poorly, with AUCs close to or below 0.5—indicating ineffective detection. In contrast, on the CTU-13 dataset, several GIDS (such as ARGUS, EULER, and VGRNN) achieved strong performance, with TPRs above 97% and higher AP values than PIKACHU and Anomal-E in certain scenarios. When comparing against traditional NIDS (IF, LOF, and OC-SVM), we observe that GIDS generally outperform NIDS across both datasets. Specifically, on CIC-IDS-2017, most GIDS achieve higher TPR and AUC than NIDS, although OC-SVM attains the lowest FPR and highest precision. On CTU-13, GIDS consistently achieve higher TPR and AUC, while NIDS exhibit TPRs below 60%, indicating their inferior detection capability relative to GIDS. These differences can be attributed to the distinct nature of attacks in each dataset: CIC-IDS-2017 contains high-intensity, short-duration attacks (e.g., DDoS) that are challenging for certain GIDS to capture after temporal graph merging, while CTU-13 consists of long-duration botnet activities that align better with the relational anomaly detection strengths of models like ARGUS and EULER. In RQ3, we further analyze the reasons behind the outstanding performance of PIKACHU and Anomal-E.

Finding: SOTA GIDS do not perform consistently across public datasets. Without detailed experimental documentation, such as model parameters, data preprocessing scripts, and environmental settings, it is challenging to accurately reproduce the experiments and validate the results.

RECOMMENDATION. To improve reproducibility of GIDS on public datasets, we recommend that future work provide full experimental documentation, including (i) preprocessed versions of each dataset used in the paper, (ii) deterministic validation strategies for thresholding, and (iii) clearly versioned code with fixed seeds and configuration files. Given the observed metric divergence even under identical settings, authors should report both reproduction and replication performance, and explicitly distinguish dataset

versions or preprocessing variants.

RQ3: How do these models generalize to new datasets derived from real-world enterprise environments?

REPLICATION SETUP. To answer this research question, we constructed an evaluation dataset based on real-world network traffic that is collected from massive networks. Compared to public datasets, this dataset is much larger in scale, encompassing a new scenario with a wider range of behaviors. To prevent exceeding the model’s processing limits (see RQ4), we first selected a random day of enterprise network data and employed random sampling to identify a subset of nodes and their corresponding communications as background traffic. The number of chosen nodes is close to that of nodes in the LANL dataset. Then, we created three evaluation datasets, each based on a distinct attack scenario, as input for the target models. We set the attack duration to be in the last four hours of the day and leveraged all snapshots (i.e., events within time windows) before the occurrence of the first attack event as the training set, and the remaining snapshots for testing. For VGRNN, EULER, and ARGUS, 5% of the edges in the training set are selected to calculate the anomaly score threshold.

For PIKACHU, we followed the method described in the original paper to sample node pairs. Assuming there are n nodes with anomalous communications, we randomly sampled $2,000 \times n$ normal nodes. To retain as much communication data related to the anomalous nodes as possible, we first extracted the normal nodes that communicated with the anomalous nodes, denoted as set V_1 , and the remaining nodes were denoted as set V_2 . We then sampled 80% of the nodes from set V_1 . If the number of sampled normal nodes was still less than $2,000 \times n$, we continued sampling from V_2 . Finally, we extracted all the sampled nodes and their communications as the input for the PIKACHU model.

For ARGUS and Anomal-E, we also incorporated six communication features to characterize user behaviors for the control experiments conducted in the original paper [19]. These features include the mean and standard deviation of communication duration, the number of packets, and the number of bytes transmitted.

RESULTS. The results of these experiments are shown in Table 4. Here, “ARGUS_ft” represents the detection performance of the model when considering communication features between nodes, and “ARGUS” represents the detection performance without accounting for these features. These results are optimized through fine-tuning the target models on three datasets. For ease of analysis, we have highlighted the best results for each attack dataset.

OUTCOME ANALYSIS. Overall, the results show despite a high TPR, the FPR of all models increases significantly, and the precision score drops noticeably, indicating that many false positives were generated during detection. For example, for the Sandworm attack dataset, the ARGUS

TABLE 4: Evaluation results on our large-scale dataset.

Attack	Model	TPR	FPR	Prec.	AUC	AP
OilRig	ARGUS_ft	0.976	0.176	0.021	0.918	0.022
	ARGUS	1.000	0.115	0.030	0.942	0.030
	EULER	1.000	0.070	0.048	0.951	0.032
	VGRNN	1.000	0.065	0.051	0.951	0.031
	PIKACHU	1.000	0.002	0.649	0.999	0.560
	Anomal-E	0.875	0.006	0.264	0.934	0.231
Sandworm	ARGUS_ft	1.000	0.383	0.006	0.872	0.009
	ARGUS	1.000	0.227	0.010	0.833	0.006
	EULER	1.000	0.264	0.017	0.835	0.006
	VGRNN	0.973	0.243	0.009	0.813	0.006
	PIKACHU	0.934	0.067	0.017	0.961	0.016
	Anomal-E	1.000	0.115	0.005	0.942	0.005
Wizard-Spider	ARGUS_ft	0.737	0.153	0.008	0.800	0.010
	ARGUS	0.849	0.072	0.017	0.855	0.011
	EULER	0.849	0.106	0.012	0.912	0.019
	VGRNN	0.849	0.069	0.018	0.854	0.012
	PIKACHU	0.967	0.033	0.014	0.998	0.548
	Anomal-E	0.984	0.046	0.008	0.969	0.008

* In the table, Prec. denotes precision.

model (without edge features) detected 528 attack events but misclassified 1,529,202 normal events as anomalies, resulting in a false positive count that is 2,896 times the number of detected attack events, which is a substantial cost.

In addition, the detection performance of ARGUS_ft (considering edge features) decreases in all OilRig and WizardSpider attack datasets and increases in the Sandworm dataset. This is contrary to the trend outlined in the original paper. We observed that the original paper only evaluated the impact of including edge features on the LANL dataset but failed to introduce edge features in another public dataset, OpTC. Therefore, to verify the generalization of incorporating edge features during the training process, we suggest conducting experiments on more datasets. From another perspective, certain attack behaviors may exhibit edge features that resemble those of normal behaviors, leading to the failure of methods that incorporate such features. To explore this, we further analyzed ARGUS_ft’s performance by computing the cosine similarity of edge features (duration, packet count, and byte count) between normal and attack edges in our enterprise dataset. We found over 93% of attack edges in the OilRig scenario had a feature similarity greater than 0.85 with normal edges, concretely explaining the reduced precision.

Surprisingly, on all attack datasets, the detection performance of PIKACHU is significantly better than that of EULER and ARGUS, which is contrary to the conclusions drawn in their original papers. Anomal-E also demonstrates outstanding performance. There are three possible reasons for this discrepancy. (1) PIKACHU and Anomal-E do not rely on discrete time graphs during the testing phase and do not merge edges within the same temporal graph. Therefore, a high volume of true negatives contribute to the superior detection results of these two methods. (2) PIKACHU adopts a distinct threshold-setting approach during evaluation. As discussed in RQ2, EULER and ARGUS calculate the detection threshold using a validation set randomly sampled from the training set. However, PIKACHU directly utilizes the testing set as

the validation set (also known as transductive learning) instead of sampling from the training set. After computing the scores for all edges in the testing set, the optimal detection threshold is determined based on the ground truth. This dynamic threshold calculation method, reliant on the ground truth, grants an additional advantage to PIKACHU. (3) PIKACHU performs additional sampling on the dataset. Restricted by memory limitations, PIKACHU cannot process all data. Hence, prior to data analysis, it samples normal communications according to the ratio of normal to abnormal communications. For example, in the Sandworm attack dataset, although the number of attack events remains unchanged after sampling, the number of normal events only accounts for 48% of the original count, which also provides an additional advantage. Therefore, we used a smaller dataset that included 8,745 normal nodes and their communications, and removed the pre-sampling stage for PIKACHU. The results are presented in Figure 8 in the appendix. It is evident that although PIKACHU still performs the best, the score gap has significantly narrowed.

Finding: Most of SOTA GIDS do not generalize well to our new dataset, which is derived from real-world enterprise environments. The performance of recently proposed GIDS may not necessarily surpass that of previous ones. Therefore, to enhance the generalization ability of GIDS, it is necessary to evaluate them on a wider range of intrusion detection datasets to better represent real-world scenarios.

RECOMMENDATION. To improve the generalization of GIDS to enterprise environments, we recommend that future research (i) incorporate evaluations on operational-scale datasets that reflect diverse attack types and benign behaviors, (ii) avoid relying solely on time-windowed graph compression, which may mask temporal distinctions crucial for anomaly detection, and (iii) explicitly document and compare threshold-setting and sampling strategies, as these significantly influence generalization outcomes. Furthermore, we encourage the use of benchmarking practices that isolate true generalization (e.g., fixed training distributions with unseen test behaviors) and the release of sanitized enterprise traces to foster more realistic evaluation baselines.

RQ4: How do these models perform from both temporal and spatial perspectives in a production environment?

SETUP. In order to determine the time and space efficiency of the target models, we utilized the Efficiency Assessment Module to measure their running time and memory usage during execution. Specifically, we conducted the evaluation using the network traffic collected from an anonymous enterprise. The enterprise dataset contains voluminous traffic, enabling us to scale the graph size. However, none of the target models can process the entire traffic for a single day

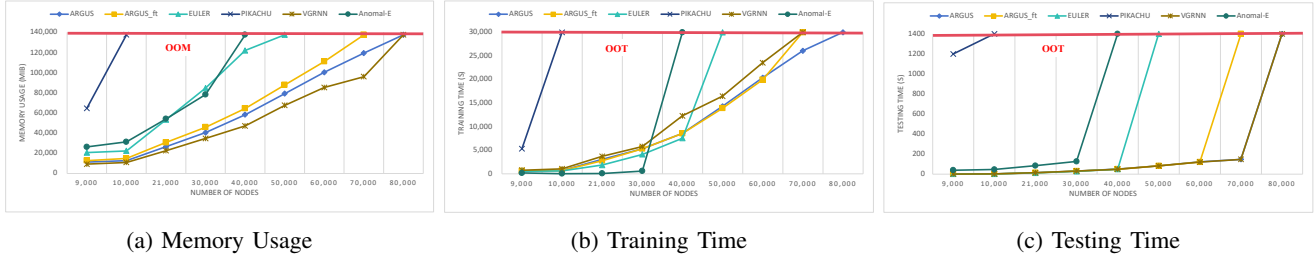


Figure 5: Memory usage and training/testing time of the target models. The red horizontal line represents the maximum memory or the time budgets available in the experimental environment.

in this dataset. Therefore, we randomly chose one day’s network data and sampled a subset of nodes along with their communication events. We gradually increased the number of nodes until it exceeded the processing capacity of the target models. Throughout this process, we recorded the memory usage and time performance of the target models under different dataset sizes, during both the training and detection phases. We partitioned the dataset into training and testing following the same strategy outlined in RQ3.

RESULTS. The results are shown in Figure 5, where the X axis represents the number of nodes in the dataset. Figure 5a depicts the memory usage of the target models, while Figures 5b and 5c illustrate the training time and testing time, respectively. The red line in the figures represents the maximum memory (128G) or the time budgets (30,000 seconds for training and 1,400 seconds for testing) available in the experimental environment. Reaching this red line indicates that the model will encounter an OOM (Out of Memory) or OOT (Out of Time) error when processing datasets beyond this scale.

OUTCOME ANALYSIS. The results show that VGRNN and ARGUS have the highest space efficiency. When edge features are disregarded, ARGUS can manage data with up to 70K nodes. However, incorporating edge features elevates memory overhead, causing OOM errors during the processing of data with 70K nodes. In contrast, EULER can handle up to 40K nodes, exhibiting lower space efficiency compared to ARGUS. Upon analyzing the design of EULER, we discovered that even when configuring the number of workers and threads to 1, the system still generates both worker and leader processes during runtime. These processes jointly consume memory, and additional memory is utilized for torch inter-process communication, thereby rendering EULER less space-efficient than ARGUS, which operates serially. Anomal-E demonstrate low efficiency and can only accommodate 30K nodes. This is mainly because Anomal-E does not introduce discrete-time graphs and trains the traffic graph as a whole. Conversely, PIKACHU’s space efficiency is markedly inferior to several other models, encountering OOM errors with more than 10K nodes. This limitation potentially makes it unsuitable for large-scale network data in enterprise environments. The primary reason is that PIKACHU processes the entire training set as a single batch

during anomaly detection. Large-scale matrix multiplication and differentiation operations lead to a substantial increase in memory usage.

In terms of time efficiency, ARGUS, EULER, and VGRNN exhibit similar data processing time during the testing phase. However, EULER requires less time to train datasets of comparable size. We observed that all three models adopt the GNN + RNN architecture. In the RNN component, ARGUS and EULER utilize the same GRU module, while VGRNN employs a more intricate GC-LSTM module. Meanwhile, ARGUS adopts a more sophisticated Message Passing Neural Network (MPNN) for the GNN module, while EULER and VGRNN utilize a simpler GCN module. Due to the complexity of their model architectures, ARGUS and VGRNN experience longer training durations under identical computing resources. Compared with the aforementioned three models, Anomal-E has longer testing time but shorter training time. This disparity arises because Anomal-E uses GNN to generate node embeddings, featuring a simpler model architecture. However, during the testing phase, Anomal-E needs to traverse all edges in the graph simultaneously to generate anomaly scores, which is a time-consuming process. Compared with all other models, PIKACHU performs significantly worse in terms of time efficiency. For example, on a dataset with 9,000 nodes, the training time of PIKACHU is 9.5 times that of EULER, and the testing time is 537 times longer. This is attributed to PIKACHU’s two-stage training process: node embedding and anomaly detection training. It first performs random walks on each Dynamic Temporal Graph (DTG) and regenerates node embeddings via GRU. Then, it trains the anomaly detection component based on these embeddings, prolonging the overall training time. During the testing phase, models like ARGUS and EULER can directly obtain anomaly scores of all edges in a single DTG through reconstruction loss, while PIKACHU needs to traverse all edges in the graph to generate adjacent joint embeddings, resulting in substantial time consumption.

Finding: GIDS fail to handle large-scale datasets collected from massive networks. The trade-offs between model complexity, memory usage, and computational efficiency highlight the importance of optimizing architecture design for scalability and

performance.

RECOMMENDATION. To ensure GIDS can scale in production environments, we recommend that future research prioritize architectural designs that explicitly consider memory and runtime constraints. In particular, models should adopt modular, streaming-compatible encoders, leverage subgraph batching or hierarchical aggregation to reduce resource overhead, and avoid single-batch training pipelines where possible. Evaluation should report time and memory usage under increasing dataset sizes, alongside detection performance, to facilitate deployment-readiness comparisons. Additionally, researchers should provide configurable resource caps and failure-triggering thresholds (e.g., for OOM/OOT) as part of artifact releases to promote fair benchmarking under practical constraints.

RQ5: How resilient are these models to adversarial attacks?

SETUP. To address this research question, we employed the SOTA evasion attack method [61] targeting GNN-based models in the Robustness Assessment Module. Unlike poisoning attacks, evasion attacks do not modify the training set; instead, they only introduce adversarial perturbations into the testing set. For instance, an attacker can add adversarial edges by controlling decoy connections, which interferes with the judgment of GIDS on real attack edges. Compared to poisoning attacks that require alterations to the training data, evasion attacks are more practical in real-world scenarios because they only involve introducing perturbations during the attack phase without changing the internal network data used for model training. We excluded the evaluations of PIKACHU and Anomal-E for two reasons. Firstly, both the node embedding and anomaly detection processes in PIKACHU and Anomal-E are divided into two distinct stages, and evasion attacks are unable to update the node embeddings. Secondly, and most importantly, the anomaly scores obtained through PIKACHU and Anomal-E are not differentiable with respect to the adjacency matrix. Additionally, we excluded the CIC-IDS-2017 dataset because methods such as ARGUS are ineffective in detecting anomalies (as discussed in RQ2). Therefore, we applied evasion attacks to three datasets (LANL, OpTC, and our own dataset) using models that had already been trained with optimal parameters. We varied the number of adversarial edges (e.g., 2, 5, 10, 20, and 50), and measured the performance of the target models. To the best of our knowledge, no such attempts have been made in related works.

RESULTS. The results for the VGRNN, EULER and ARGUS are shown in Table 5. In this table, K represents the number of added adversarial edges (i.e., target events). r_{tgt} and r_{cov} denote the average evasion rate of the target event and the covering event (i.e., edges made to disguise a target event), respectively. And r_{atk} denotes the average success rate of the attack. For each target event, the attack

is considered successful only if both the target event and the covering event are classified as normal. The light blue-shaded cells indicate the highest r_{atk} value for each K .

OUTCOME ANALYSIS. The results show that VGRNN, EULER and ARGUS perform similarly on the LANL dataset. Evasion attacks achieve the highest success rate on the LANL dataset, with all attack events fully covered by the insertion of just two adversarial edges. This underscores the current models’ lack of robustness, allowing attackers to evade detection through adversarial attacks. For the OpTC dataset, EULER and ARGUS nearly cover all attack events by inserting ten adversarial edges.

In contrast, on our own dataset and for VGRNN on the OpTC dataset, the success rate of adversarial attacks is not high. Even with the insertion of 50 adversarial edges, the attack events cannot be fully covered. A potential reason for this is related to the detection capability of GIDS. As discussed in RQ2 and RQ3, although these target models can detect most of attack events in the dataset, they also exhibit a high false positive rate. This indicates that these models struggle to accurately distinguish between attack events and normal events, resulting in a lowered threshold for anomaly detection. In the adversarial attack experiments, we discovered that the most effective adversarial edges for covering attack events had scores below the detection threshold and were themselves classified as attacks. In contrast, adversarial edges with scores above the detection threshold could not effectively conceal the attack events. From this perspective, a decrease in the model’s detection accuracy could paradoxically improve its robustness against adversarial attacks.

Finding: GIDS suffer from brittle detection boundaries, adding just a few adversarial edges can fully evade detection across models. Surprisingly, models with higher false positives exhibit greater resilience, suggesting that overly aggressive filtering may inadvertently absorb adversarial noise. Temporal compression and non-adaptive thresholds further amplify vulnerability, exposing a fundamental fragility in current GNN-based intrusion detection design.

RECOMMENDATION. To enhance the adversarial robustness of GIDS, we recommend integrating gradient-based adversarial training or certified defense strategies into the model training pipeline. Future work should evaluate models under both white-box and black-box threat models, and report robustness metrics alongside traditional detection scores. Moreover, GIDS should expose differentiable attack surfaces during training where feasible, or otherwise adopt embedding-level defense mechanisms to secure post-hoc detection stages. Finally, we advocate for the standardization of robustness benchmarks and the release of attack scripts and ground-truth mappings to enable consistent, reproducible evaluation across research efforts.

TABLE 5: Evasion attack performance of VGRNN, EULER, and ARGUS on LANL, OpTC, and Our Dataset. Each cell shows $r_{tgt}/r_{cov}/r_{atk}$ for the corresponding model and dataset.

K	LANL			OpTC			Our Dataset		
	VGRNN	EULER	ARGUS	VGRNN	EULER	ARGUS	VGRNN	EULER	ARGUS
0	0.12/1.00/0.12	0.13/1.00/0.13	0.18/1.00/0.18	0.00/1.00/0.00	0.03/1.00/0.03	0.02/1.00/0.02	0.03/1.00/0.03	0.03/1.00/0.03	0.04/1.00/0.04
2	0.68/1.00/0.68	1.00/1.00/1.00	1.00/1.00/1.00	0.00/1.00/0.00	0.38/1.00/0.38	0.20/1.00/0.20	0.04/1.00/0.04	0.03/0.99/0.00	0.04/0.98/0.03
5	1.00/1.00/1.00	1.00/1.00/1.00	1.00/1.00/1.00	0.24/1.00/0.24	0.69/1.00/0.69	0.50/1.00/0.50	0.04/1.00/0.04	0.03/0.98/0.00	0.04/0.97/0.03
10	1.00/1.00/1.00	1.00/1.00/1.00	1.00/1.00/1.00	0.28/1.00/0.28	1.00/1.00/1.00	0.98/1.00/0.98	0.04/1.00/0.04	0.03/0.98/0.00	0.04/0.97/0.03
20	1.00/1.00/1.00	1.00/1.00/1.00	1.00/1.00/1.00	0.75/1.00/0.75	1.00/1.00/1.00	1.00/1.00/1.00	0.04/1.00/0.04	0.03/0.96/0.00	0.13/0.99/0.13
50	1.00/1.00/1.00	1.00/1.00/1.00	1.00/1.00/0.99	0.93/1.00/0.88	1.00/1.00/1.00	1.00/1.00/1.00	0.04/1.00/0.04	0.03/0.94/0.00	0.22/1.00/0.21

5. Related Work

Performance of GIDS. Various studies have been carried out to analyze the performance of existing GIDS. Most of these studies [24, 62, 63] merely reviewed existing evaluation results as analytical basis but did not reproduce or re-evaluate existing NIDS. The work by Apruzzese *et al.* [64] is the most similar to ours. They also conducted extensive evaluation experiments to reveal the gap between research and practice in the NIDS field. However, their work only evaluated traditional machine learning algorithms, such as random forest and logistic regression. These algorithms are supervised, requiring extensive labeled datasets for training. In large-scale enterprise environments, obtaining high-quality, comprehensive labels for attacks—especially novel or stealthy threats—is often impractical. In contrast, we assess SOTA GIDS using large-scale datasets, of which four are publicly available and one is commercially collected from real-world production environments, and we compared the performance differences between GIDS and traditional unsupervised NIDS. We believe our study complements related works to provide a holistic view of existing GIDS.

Robustness of GIDS. Several studies [24, 65] have evaluated the robustness of GIDS. Pujol-Perich *et al.* [66] evaluated the robustness of GNN-based NIDS under two adversarial attacks by modifying packet sizes and arrival times. Their research indicated that learning the relationships between different flows can strengthen the model’s robustness. Zhou *et al.* [67] proposed an adversarial attack method that significantly reduces the detection accuracy of GIDS in IoT environments. Apruzzese *et al.* [68] analyzed the threat model of existing adversarial attack methods, re-modeled the capabilities of attackers in real-world scenarios, and evaluated the impact of adversarial attacks on NIDS under this threat model. However, most of these evaluation works focused on adversarial attacks established in IoT and SDN environments and only added disturbances at the packet level. In contrast, we leverage adversarial attack methods [61] by incorporating access behaviors into the network to assess the robustness of GIDS.

6. Conclusion

GIDS hold great promise for detecting advanced threats, but their real-world readiness remains uncertain. This study takes a critical step toward closing that gap by

systematically examining their reproducibility, scalability, and robustness. Through extensive evaluations across public and enterprise datasets, we uncover that reproducing published results is fraught with challenges—undocumented preprocessing steps, hyperparameter sensitivities, and non-deterministic thresholding mechanisms lead to significant performance variations. While some findings (e.g., parameter influence and evasion vulnerabilities) align with known issues, our work reveals a deeper reproducibility crisis: without access to precise experimental setups, even careful reimplementations struggle to match claimed performance. This underscores a pervasive lack of rigor in empirical evaluation within the GIDS community. We urge researchers to prioritize transparency by sharing full experimental artifacts (code, data, configurations) and adopting standardized evaluation protocols. By doing so, we can foster more reliable and resilient intrusion detection systems that truly translate from research to practice.

DISCUSSION ON REAL-WORLD APPLICABILITY. Security systems operate as part of a layered defense. While GIDS can detect advanced and stealthy attacks, they should complement—not replace—traditional rule-based NIDS, which offer interpretability and low false positives for known threats. GIDS are best deployed where advanced persistent threats (APTs) are a concern and sufficient computational and analyst resources exist.

Our evaluation shows that current GIDS models still produce high false positive rates (e.g., up to thousands of false alerts per true positive in some cases), which could overwhelm security analysts if not properly tuned or integrated with alert correlation systems, or lead to scalability issues (e.g., OOM errors). Therefore, enterprises considering GIDS should prioritize models with lower FPR and higher precision (e.g., PIKACHU in our tests) and ensure they have the operational capacity to triage additional alerts. Future work must improve alert explainability and integrate GIDS into hybrid frameworks that combine rule-based and learning-based strengths.

7. Ethical Statement

The collection and usage of the enterprise dataset in this study were conducted in full compliance with the data governance and security policies of the anonymous enterprise. All data was thoroughly anonymized and sanitized to remove any personally identifiable information (PII) and sensitive operational details before analysis.

References

- [1] "Yahoo discloses hack of 1 billion accounts," <https://techcrunch.com/2016/12/14/yahoo-disclosethacker-of-1-billionaccounts/>, 2016.
- [2] "The marriott data breach," <https://www.consumer.ftc.gov/blog/2018/12/marriottdata-breach>, 2018.
- [3] "Home depot confirms data breach at u.s., canadian stores," <http://www.npr.org/2014/09/09/347007380/homedepot-confirms-databreach-at-u-s-canadian-stores>, 2014.
- [4] "The zeek network security monitor," <https://github.com/zeek/zeek>, 2024.
- [5] "Suricata," <https://github.com/OISF/suricata>, 2024.
- [6] "Snort," <https://www.snort.org/>, 2024.
- [7] "Modsecurity web application firewall," <https://github.com/owasp-modsecurity/ModSecurity>, 2024.
- [8] "Yara," <https://github.com/VirusTotal/yara>, 2024.
- [9] B. A. Alahmadi, L. Axon, and I. Martinovic, "99% false positives: A qualitative study of {SOC} analysts' perspectives on security alarms," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 2783–2800.
- [10] 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 systems security symposium*, 2019.
- [11] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *IEEE Symposium on Security and Privacy (S&P)*, 2020.
- [12] M. A. Inam, Y. Chen, A. Goyal, J. Liu, J. Mink, N. Michael, S. Gaur, A. Bates, and W. U. Hassan, "Sok: History is a vast early warning system: Auditing the provenance of system intrusions," in *IEEE Symposium on Security and Privacy (S&P)*, 2023.
- [13] F. Meng, J. Gui, F. Zou, Y. Li, and Y. Wu, "Distr: Detecting multi-stage iot botnets through contextual traffic and causal analytics," *Computers & Security*, p. 104531, 2025.
- [14] K. Mukherjee, J. Wiedemeier, Q. Wang, J. Kamimura, J. J. Rhee, J. Wei, Z. Li, X. Yu, L.-A. Tang, J. Gui *et al.*, "Proviot: Detecting stealthy attacks in iot through federated edge-cloud security," in *International Conference on Applied Cryptography and Network Security*. Springer, 2024, pp. 241–268.
- [15] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2672–2681.
- [16] R. Paudel and H. H. Huang, "Pikachu: Temporal walk based dynamic graph embedding for network anomaly detection," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–7.
- [17] I. J. King and H. H. Huang, "Euler: Detecting network lateral movement via scalable temporal link prediction," *ACM Transactions on Privacy and Security*, vol. 26, no. 3, pp. 1–36, 2023.
- [18] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.
- [19] J. Xu, X. Shu, and Z. Li, "Understanding and bridging the gap between unsupervised network representation learning and security analytics," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 3590–3608.
- [20] Q. Cheng, Y. Shen, D. Kong, and C. Wu, "Step: Spatial-temporal network security event prediction," *arXiv preprint arXiv:2105.14932*, 2021.
- [21] S. Wang, Z. Chen, X. Yu, D. Li, J. Ni, L.-A. Tang, J. Gui, Z. Li, H. Chen, and P. S. Yu, "Heterogeneous graph matching networks for unknown malware detection," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 3762–3770.
- [22] P. Wang, J. Gui, Z. Chen, J. Rhee, H. Chen, and Y. Fu, "A generic edge-empowered graph convolutional network via node-edge mutual enhancement," in *Proceedings of the web conference 2020*, 2020, pp. 2144–2154.
- [23] L. Cai, Z. Chen, C. Luo, J. Gui, J. Ni, D. Li, and H. Chen, "Structural temporal graph neural networks for anomaly detection in dynamic graphs," in *Proceedings of the 30th ACM international conference on Information & Knowledge Management*, 2021, pp. 3747–3756.
- [24] T. Bilot, N. El Madhoun, K. Al Agha, and A. Zouaoui, "Graph neural networks for intrusion detection: A survey," *IEEE Access*, 2023.
- [25] A. Hussain, E. M. Tordera, X. Masip-Bruin, and H. C. Leligou, "Rule-based with machine learning ids for ddos attack detection in cyber-physical production systems (cpps)," *IEEE access*, 2024.
- [26] H. Alqahtani and G. Kumar, "Deep learning-based intrusion detection system for in-vehicle networks with knowledge graph and statistical methods," *International Journal of Machine Learning and Cybernetics*, vol. 16, no. 5, pp. 3539–3555, 2025.
- [27] G. Qian, J. Li, W. He, W. Zhang, and Y. Cao, "An online intrusion detection method for industrial control systems based on extended belief rule base," *International Journal of Information Security*, vol. 23, no. 4, pp. 2491–2514, 2024.
- [28] 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.
- [29] E. Hajiramezani, A. Hasanzadeh, K. Narayanan, N. Duffield, M. Zhou, and X. Qian, "Variational graph recurrent neural networks," *Advances in neural information processing systems*, vol. 32, 2019.
- [30] A. D. Kent, "Cybersecurity Data Sources for Dynamic Network Research," in *Dynamic Networks in Cybersecurity*. Imperial College Press, Jun. 2015.
- [31] R. Arantes, C. Weir, H. Hannon, and M. Kulseng, "Operationally transparent cyber (optc)," 2021. [Online]. Available: <https://dx.doi.org/10.21227/edq8-nk52>
- [32] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSp*, vol. 1, pp. 108–116, 2018.
- [33] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.
- [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 *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–9.
- [35] A. Venturi, M. Ferrari, M. Marchetti, and M. Colajanni, "Arganids: a novel network intrusion detection system based on adversarially regularized graph autoencoder," in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, 2023, pp. 1540–1548.
- [36] M.-L. Messai and H. Seba, "Iot network attack detection: Leveraging graph learning for enhanced security," in *Proceedings of the 18th International Conference on Availability, Reliability and Security*, 2023, pp. 1–7.
- [37] 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 *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 3644–3660.
- [38] M. Wang, N. Yang, and N. Weng, "K-getmid: Knowledge-guided graphs for early and transferable network intrusion detection," *IEEE Transactions on Information Forensics and Security*, 2024.
- [39] X. Wang, X. Wang, M. He, M. Zhang, and Z. Lu, "Spatial-temporal graph model based on attention mechanism for anomalous iot intrusion detection," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 3, pp. 3497–3509, 2023.
- [40] L. Lin, Q. Zhong, J. Qiu, and Z. Liang, "E-gracl: an iot intrusion detection system based on graph neural networks," *The Journal of*

Supercomputing, vol. 81, no. 1, p. 42, 2024.

- [41] R. Zhao, M. Shoaib, V. T. Hoang, and W. U. Hassan, "Rethinking tamper-evident logging: A high-performance, co-designed auditing system," in *ACM Conference on Computer and Communications Security (CCS)*, 2025.
- [42] A. Longa, V. Lachi, G. Santin, M. Bianchini, B. Lepri, P. Lio, F. Scarselli, and A. Passerini, "Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities," *arXiv preprint arXiv:2302.01018*, 2023.
- [43] "Arpa transparent computing engagement 3," <https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md>.
- [44] "Arpa transparent computing engagement 5," <https://github.com/darpa-i2o/Transparent-Computing/blob/master/README.md>.
- [45] P. Fang, P. Gao, C. Liu, E. Ayday, K. Jee, T. Wang, Y. F. Ye, Z. Liu, and X. Xiao, "Back-Propagating system dependency impact for attack investigation," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 2461–2478.
- [46] 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 *Proceedings 2024 Network and Distributed System Security Symposium*, ser. NDSS 2024. Internet Society, 2024.
- [47] 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 *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 3533–3551.
- [48] Q. Liu, K. Bao, W. U. Hassan, and V. Hagenmeyer, "Hades: Detecting active directory attacks via whole network provenance analytics," *IEEE Transactions on Dependable and Secure Computing*, 2025.
- [49] M. U. Rehman, H. Ahmadi, and W. U. Hassan, "Flash: A comprehensive approach to intrusion detection via provenance graph representation learning," in *IEEE Symposium on Security and Privacy (S&P)*, 2024.
- [50] Q. Liu, M. Shoaib, M. U. Rehman, K. Bao, V. Hagenmeyer, and W. U. Hassan, "Accurate and scalable detection and investigation of cyber persistence threats," *arXiv preprint arXiv:2407.18832*, 2024.
- [51] M. Shoaib, A. Suh, and W. U. Hassan, "Principled and automated approach for investigating ar/vr attacks," in *USENIX Security Symposium*, 2025.
- [52] J. Gui, X. Xiao, D. Li, C. H. Kim, and H. Chen, "Progressive processing of system-behavioral query," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 378–389.
- [53] J. Gui, D. Li, Z. Chen, J. Rhee, X. Xiao, M. Zhang, K. Jee, Z. Li, and H. Chen, "Aprtrace: A responsive system for agile enterprise level causality analysis," in *2020 IEEE 36th international conference on data engineering (ICDE)*. IEEE, 2020, pp. 1701–1712.
- [54] D. Li, J. Xiao, P. Jiang, J. Gui, D. Song, Y. Ma, G. Huang, and X. Liu, "Provaudit: Enhance high-level privacy inference through system provenance data," *IEEE Transactions on Dependable and Secure Computing*, 2024.
- [55] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: the cids2017 case study," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 7–12.
- [56] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.
- [57] 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 *Proceedings of the 44th IEEE International Conference on Computer Communications (INFOCOM)*, 2025.
- [58] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth IEEE international conference on data mining*. IEEE, 2008, pp. 413–422.
- [59] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.
- [60] 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.
- [61] 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 *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 229–246.
- [62] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.
- [63] M. Zhong, M. Lin, C. Zhang, and Z. Xu, "A survey on graph neural networks for intrusion detection systems: Methods, trends and challenges," *Computers & Security*, p. 103821, 2024.
- [64] G. Apruzzese, P. Laskov, and J. Schneider, "Sok: Pragmatic assessment of machine learning for network intrusion detection," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2023, pp. 592–614.
- [65] J. Aiken and S. Scott-Hayward, "Investigating adversarial attacks against network intrusion detection systems in sdns," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2019, pp. 1–7.
- [66] D. Pujol-Perich, J. Suárez-Varela, A. Cabellos-Aparicio, and P. Barlet-Ros, "Unveiling the potential of graph neural networks for robust intrusion detection," *ACM SIGMETRICS Performance Evaluation Review*, vol. 49, no. 4, pp. 111–117, 2022.
- [67] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu, I. Kevin, and K. Wang, "Hierarchical adversarial attacks against graph-neural-network-based iot network intrusion detection system," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9310–9319, 2021.
- [68] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, and M. Colajanni, "Modeling realistic adversarial attacks against network intrusion detection systems," *Digital Threats: Research and Practice (DTRAP)*, vol. 3, no. 3, pp. 1–19, 2022.
- [69] "Adversary emulation tool," https://github.com/center-for-threat-identified-defense/adversary_emulation_library, 2023.

Appendix A. Simulated Attacks

Table 6 reflects the deliberate design of our simulated attack dataset to mirror real-world threat actor behaviors. The varying event counts, such as OilRig’s high activity (2,532 events) versus Wizard Spider’s sparse patterns (366 events), capture differences in operational tempo observed in actual campaigns. By simulating distinct groups (e.g., state-sponsored Sandworm or cybercriminal Wizard Spider), we enable nuanced analysis of detection methods across diverse attack profiles. This stratification supports targeted research into how defensive systems perform against both high-volume and low-and-slow threats.

TABLE 6: Statistics of simulated attacks.

Attack Name	# Hosts or IPs	# Events
OilRig	5	2,532
Sandworm	5	587
Wizard Spider	4	366

Appendix B. Datasets

We provide a more detailed description of the benign and attack behaviors within our experimental datasets below.

The **LANL** dataset originates from the internal corporate network of the Los Alamos National Laboratory. Its benign activity consists of genuine, human-made internal corporate traffic (e.g., authentication events, internal service communications) collected over 58 days. The red team attacks are also human-conducted, simulating credential dumping, lateral movement, and data exfiltration, providing realistic APT-like scenarios.

The **OpTC** dataset contains logs from approximately 1,000 machines over one week. The benign background is synthetic, generated by the FLARE program to emulate typical behavior in a government agency network. The attacks are also synthetically generated by the FLARE program, simulating a variety of techniques from the MITRE ATT&CK framework.

The **CIC-IDS-2017** dataset is a network traffic dataset spanning five days, created in a lab environment. Its benign traffic comprises synthetic profiles of human users generating HTTP, HTTPS, FTP, SSH, and email protocols. The attacks are synthetically generated using scripts to execute specific attacks like DoS, Brute Force, and Web attacks, with clear start and end times. To address known flaws in the original CIC-IDS-2017 data, we used its improved version as described in [55].

The **CTU-13** dataset is a botnet traffic dataset developed by Czech Technical University (CTU) in a lab environment. Its benign traffic comprises synthetic normal communications emulating a small-to-medium enterprise (SME) network, including HTTP/HTTPS, email, file transfers, and basic P2P traffic. The attacks focus on botnet behavior and data collection, with clear labeling of infected hosts and timelines. Following [35], we selected the two Neris attacks in CTU-13 that have the longest duration and satisfy the condition of having no mixed attacks and normal behaviors based on timestamps. Since the number of nodes in CTU-13 reaches the million scale—far exceeding the processing limit of GIDS—we sampled the background traffic to keep the number of nodes at 20K.

Furthermore, to assess the generalizability of these GIDS on our **newly collected dataset**—which contains over 10 billion events collected from a real-world enterprise network over 101 days—we first simulated three types of attacks and collected attack traffic. Then, we adopted the same approach as in [57] to merge attack traffic into the new dataset. The benign activity in this dataset is entirely human-made, representing the authentic, diverse, and noisy traffic

typical of a large organization. This includes web browsing, cloud service access, internal authentication (e.g., Kerberos, NTLM), database queries, and remote desktop connections.

Table 6 in the appendix shows the statistical information of the simulated attacks. We configured the victim enterprise’s network environment using virtual machines and simulated the attacks following guidelines from the MITRE adversary emulation library [69]. This environment comprises a Windows domain with a Domain Controller (running Windows Server 2019) and multiple domain-joined hosts running Windows and Linux systems. We chose three representative APTs: Sandworm, Wizard Spider, and OilRig, known for their comprehensive attack chains and significant disruptive potential. The goal of these attacks was to penetrate the domain environment and gain control over the Domain Controller, thereby dominating the entire domain. Throughout the attacks, we utilized Zeek to capture attack traffic flows as attack events.

Appendix C. Example Event Record

The term “event” in our work primarily refers to a summarized network flow or connection record. While the exact schema can vary slightly between datasets (LANL uses primarily authentication logs, OpTC, CIC-IDS-2017 and CTU-13 use NetFlow-like data, our enterprise dataset uses Zeek logs), a typical event record contains fields similar to the following example from our Zeek-based enterprise dataset:

```
{
  "ts": 1682870388.686172, // Timestamp
  "uid": "CrqQV2zM8S1v06T9l", // Unique connection ID
  "id.orig_h": "10.157.132.0", // Originator IP
  "id.orig_p": 53213, // Originator port
  "id.resp_h": "202.120.223.6", // Responder IP
  "id.resp_p": 53, // Responder port
  "proto": "udp", // Protocol
  "service": "dns", // Detected application protocol
  "duration": 0.000366, // Connection duration
  "orig_bytes": 40, // Originator bytes sent
  "resp_bytes": 56, // Responder bytes sent
  "orig_pkts": 1, // Originator packets sent
  "resp_pkts": 1, // Responder packets sent
  "conn_state": "SF", // Connection state (e.g., REJ)
  "label": "normal" // Label (normal/malicious)
}
```

This structure represents a bidirectional flow summary. Other datasets might use different field sets (e.g., LANL events focus on user, source computer, destination computer, authentication type, logon type).

Appendix D. Model Parameters

In this section, we will provide a detailed description of the model parameters used in the evaluation experiments. In the replication experiments of RQ2, we adjusted the model parameters on the public datasets LANL, OpTC,

CIC-IDS-2017 and CTU-13 to achieve optimal results. In the experiments of RQ3, we conducted parameter adjustments on the three selected attack datasets. Table 8 shows hyperparameters used for each system across datasets. Specifically, for Anomal-E, we adjusted the embedding dimension and detection model. For VGRNN, we adjusted the number of layers in the GNN model, the time-window size of snapshot, learning rate, threshold weight, and patience. For PIKACHU, we adjusted the embedding dimension, snapshot time window size, learning rate, and the number of sampled neighbors. For EULER, we tried various combinations of GNN and RNN, and adjusted the number of layers in the GNN model, the time-window size of snapshot, learning rate, threshold weight, and patience. For ARGUS, we tried various combinations of RNN, and adjusted the number of layers in the GNN model, the time-window size of snapshot, learning rate, threshold weight, and patience. Additionally, we adjusted the margin parameter used in calculating the average precision loss. For Isolation Forest (IF), we adjusted the number of estimators, maximum samples, contamination rate, maximum features, and bootstrap. For Local Outlier Factor (LOF), we adjusted the number of neighbors, contamination, algorithm, leaf size, and pairwise distances. For OC-SVM, we experimented with different kernel functions and adjusted the value of ν —which represents an upper bound on the fraction of training errors and a lower bound on the fraction of support vectors—as well as the kernel coefficient.

TABLE 7: Parameter descriptions and their use in each GIDS.

Parameter	GIDS	Description
Snapshot Duration	ARGUS, EULER, VGRNN, PIKACHU	Time window of each snapshot.
Learning Rate	ARGUS, EULER, VGRNN, PIKACHU	Learning rate used for training the model.
# Layers	ARGUS, EULER, VGRNN	Number of layers in the GNN model.
# Neighbors	PIKACHU	Number of sampled neighbors used in each iteration.
Embedding Dim	PIKACHU, Anomal-E	Dimensionality of the learned embeddings.
RNN Model	ARGUS, EULER	Type of RNN used for modeling temporal dependencies.
Margin	ARGUS	Margin value used in calculating the average precision loss.
Detection Models	Anomal-E	Classical models used for anomaly detection.

The impact of the implementation parameters on AUC scores is shown in Figure 6. It can be seen that with different parameter settings, the models’ AUC scores remain stable. Table 7 explains each of the parameters on the X axis in Figures 3 and 6.

Appendix E. Comparison of EULER’s Reproduction Results

For the EULER model, we also evaluated the EULER-SM LSTM version from the paper [17], which demonstrated

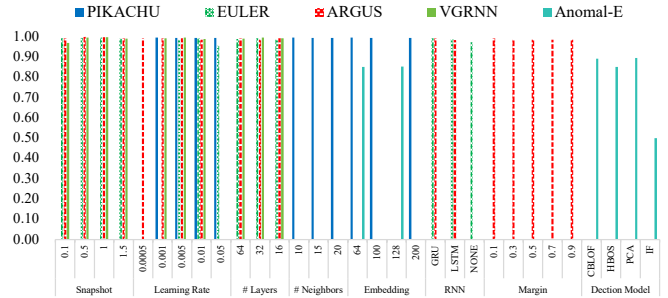


Figure 6: Impact of key implementation parameters on AUC.

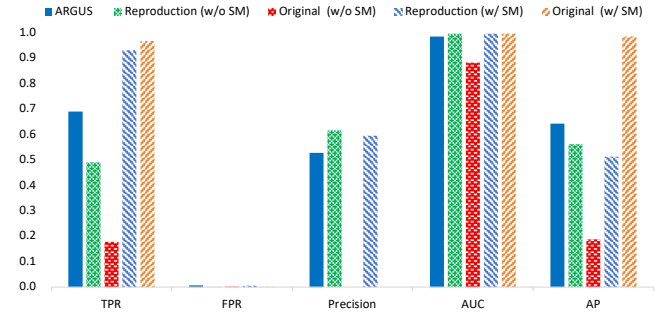


Figure 7: Comparison of EULER results on the DARPA OpTC dataset. “ARGUS” represents the reproduction result of EULER in the ARGUS paper. “Reproduction” and “Original” represent the results of our reproduction experiments and original paper, respectively. Original precision scores are not included in the figure since they are not provided in the original paper.

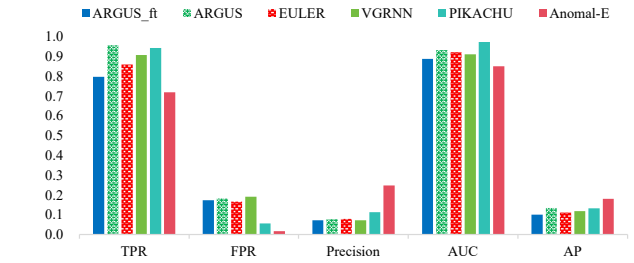


Figure 8: Evaluation on the small-scale dataset

the best detection performance. This model uses an LSTM network to learn the temporal features between dynamic temporal graphs and includes an additional softmax layer (SM) at the end to aggregate the embeddings of neighbor nodes. However, when we reproduced this evaluation experiment as described in the paper, the AP score of the EULER model, whether using SM or not, could not reach the original paper’s results. Furthermore, we compared it with the reproduction results of the EULER model provided in the ARGUS paper. As shown in Figure 7, these results are close to those of our reproduction experiment.

Appendix F. Evaluation on the Small-scale New Dataset

Figure 8 shows the results of GIDS on the small-scale dataset based on real-world enterprise traffic.

TABLE 8: Hyperparameters used for each system across datasets.

IF						
Dataset	Estimators	Max Samples	Contamination	Max Features	Bootstrap	
CIC-IDS-2017	50	auto	auto	1.0	True	
CTU-13 (neris_1)	100	0.7	0.01	0.5	True	
CTU-13 (neris_2)	100	0.7	0.01	0.5	True	
LOF						
Dataset	Neighbors	Contamination	Algorithm	Leaf Size	Pairwise Distances	
CIC-IDS-2017	2	0.05	ball_tree	30	2	
CTU-13 (neris_1)	5	0.01	ball_tree	30	2	
CTU-13 (neris_2)	5	0.01	auto	30	2	
OC-SVM						
Dataset	Kernel	Nu	Gamma			
CIC-IDS-2017	linear	0.01	0.5			
CTU-13 (neris_1)	rbf	0.05	scale			
CTU-13 (neris_2)	rbf	0.05	auto			
Anomal-E						
Dataset	Embedding Dim	Detection Model				
CIC-IDS-2017	64	HBOS				
CTU-13 (neris_1)	64	HBOS				
CTU-13 (neris_2)	64	PCA				
OilRig	128	CBLOF				
Sandworm	64	HBOS				
WizardSpider	64	CBLOF				
VGRNN						
Dataset	# Layers	Snapshot Dur. (s)	Learning Rate	Threshold Weight	Patience	
LANL	32	5400	0.01	0.50	10	
OpTC	32	1800	0.005	0.50	10	
CIC-IDS-2017	32	1200	0.05	0.5	10	
CTU-13 (neris_1)	16	150	0.01	0.49	10	
CTU-13 (neris_2)	16	600	0.01	0.49	10	
OilRig	32	150	0.01	0.48	10	
Sandworm	32	150	0.001	0.43	10	
WizardSpider	32	150	0.005	0.48	10	
PIKACHU						
Dataset	Embedding Dim	Snapshot Dur. (s)	Learning Rate	# Neighbors		
LANL	200	3600	0.001	10		
OpTC	64	360	0.001	10		
CIC-IDS-2017	100	600	0.001	15		
CTU-13 (neris_1)	100	300	0.001	15		
CTU-13 (neris_2)	100	600	0.001	5		
OilRig	100	300	0.001	10		
Sandworm	100	300	0.001	10		
WizardSpider	100	300	0.001	10		
EULER						
Dataset	GNN Model	RNN Model	# Layers	Snapshot Dur. (s)	Learning Rate	Threshold Weight
LANL	GCN	None	64	10800	0.0005	0.6
OpTC	GCN	GRU	32	360	0.01	0.6
CIC-IDS-2017	GCN	GRU	32	300	0.005	0.45
CTU-13 (neris_1)	GCN	GRU	32	600	0.05	0.5
CTU-13 (neris_2)	GCN	GRU	32	600	0.01	0.5
OilRig	GCN	GRU	32	150	0.005	0.48
Sandworm	GCN	GRU	32	150	0.005	0.42
WizardSpider	GCN	GRU	32	150	0.005	0.47
ARGUS						
Dataset	Edge Feature	RNN Model	# Layers	Snapshot Dur. (s)	Learning Rate	Threshold Weight
LANL	Yes	GRU	32	3600	0.005	0.6
OpTC	No	GRU	64	360	0.005	0.55
CIC-IDS-2017	Yes	GRU	16	900	0.0001	0.4
CIC-IDS-2017	No	GRU	16	900	0.05	0.3
CTU-13 (neris_1)	Yes	GRU	16	450	0.0005	0.48
CTU-13 (neris_1)	No	GRU	16	300	0.0005	0.5
CTU-13 (neris_2)	Yes	GRU	16	150	0.001	0.47
CTU-13 (neris_2)	No	GRU	16	600	0.005	0.4
OilRig	Yes	GRU	32	150	0.0001	0.46
OilRig	No	GRU	16	450	0.01	0.48
Sandworm	Yes	GRU	32	450	0.0001	0.45
Sandworm	No	GRU	16	150	0.0001	0.43
WizardSpider	Yes	GRU	32	150	0.0001	0.46
WizardSpider	No	GRU	32	150	0.001	0.48

* Snapshot Dur. (s) = Snapshot Duration in seconds. "# Layers" denotes number of GNN embedding layers.