

Robustness Evaluation of Graph Neural Network-based Network Intrusion Detection Systems against Adversarial Flow Injection

Matthieu Mouzaoui¹^a, Yufei Han¹^b and Gregory Blanc²^c and Gabriel Rilling³^d and Michel Hurfin¹^e

¹Inria, Univ. Rennes

²CEA, LIST, Laboratoire pour l'Instrumentation Intelligente, Distribuée et Embarquée

³SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris

{matthieu.mouzaoui, yufei.han, michel.hurfin}@inria.fr, gregory.blanc@telecom-sudparis.eu, gabriel.rilling@cea.fr

Keywords: Graph Neural Network, Network Intrusion Detection, Adversarial Machine Learning

Abstract: Graph Neural Networks (GNNs) enhance Network Intrusion Detection Systems (NIDS) by leveraging structural relationships in network traffic, but this reliance introduces new vulnerabilities. In this paper, we propose a query-based black-box flow injection attack against GNN-based NIDS, where adversaries inject crafted traffic flows into existing communications to evade detection. Our experiments demonstrate that these constrained attacks are highly effective, frequently succeeding with a minimal budget of injected flows. Analysis reveals that local graph structural properties primarily determine attack success. Additionally, the attack demonstrates strong transferability across various GNN architectures, confirming the viability of a black-box threat. These findings highlight critical vulnerabilities in GNN-based NIDS, even when adversaries are restricted to purely additive flow injections.

1 Introduction

Network Intrusion Detection Systems (NIDS) are central to modern cybersecurity. Traditional signature-based approaches fail against novel threats, driving adoption of Machine Learning (ML)-based NIDS [Sheatsley et al., 2022, Catillo et al., 2024]. Yet, conventional ML systems analyze each flow in isolation, ignoring the relational structure of network traffic. Graph-based representation addresses this by modeling traffic as a graph and leveraging Graph Neural Networks (GNNs) [Zhong et al., 2024]: flow sharing IP addresses become connected nodes or edges, and a GNN classifies each flow by aggregating information from its neighborhood (Fig. 1). Crucially, a flow's predicted label depends not only on its own features but on the structure and labels of neighboring flows at inference time.

Despite their advantages, GNN-based NIDS face significant integrity threats at deployment. Like other

machine learning models, GNNs are vulnerable to adversarial attacks [Bojchevski and Günnemann, 2019]. Because classification depends on graph topology, an attacker who reconstructs the traffic graph can inject crafted flows to manipulate neighborhood aggregation, enabling evasion or false alarms, without modifying existing traffic. This indirect attack surface makes GNN-based NIDS both powerful and inherently exploitable. Prior adversarial attacks on (GNN-based) NIDS often rely on unrealistic assumptions, white-box access, or invalid feature transformations as noted in [Merzouk et al., 2022, Vitorino et al., 2023]. He et al. [He et al., 2023] survey adversarial ML for NIDS and identify feature-level perturbations as impractical, as the reconstructed network packets cannot be transmitted across the network, calling for attacks operating directly in traffic space. Our work addresses this by injecting valid flows within the problem space under realistic reachability constraints. Random or fuzzing-based injection strategies [Venturi et al., 2024] lack domain validity and optimality guarantees. We instead evaluate robustness under a realistic model-agnostic **flow injection attack**. The attacker injects crafted flows without modifying existing traffic, sidestepping both the in-

^a <https://orcid.org/0009-0006-9104-9230>

^b <https://orcid.org/0000-0002-9035-6718>

^c <https://orcid.org/0000-0001-8150-6617>

^d <https://orcid.org/0009-0007-6111-1045>

^e <https://orcid.org/0009-0006-6695-2938>

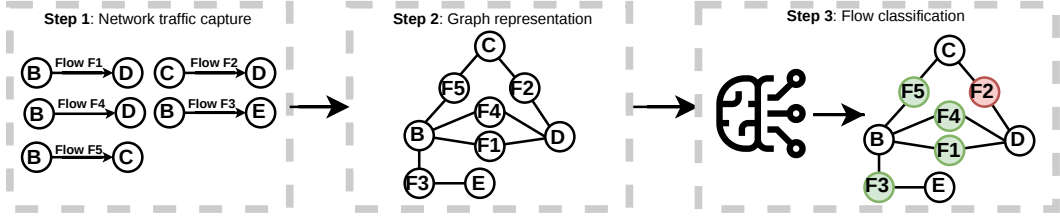


Figure 1: GNN-based NIDS pipeline. Network traffic flows are captured and represented as a graph, which will serve as input to a GNN model. The GNN outputs the predicted label: benign or malicious.

verse feature mapping problem [Pierazzi et al., 2020] and the need for privileged model access. Injecting small, benign-looking flows dilutes the malicious signal during neighborhood aggregation, a technique observed in distributed scanning and C&C communication, and is difficult to distinguish from background noise in model networks. The attacker controls one external endpoint and injects flows toward reachable destinations, with historical traffic available for graph reconstruction. Two challenges arise: (1) the discrete nature of graph-structured representations creates a massive combinatorial search space for injected connections, and (2) reachability constraints limit feasible injection targets to endpoints with active service ports.

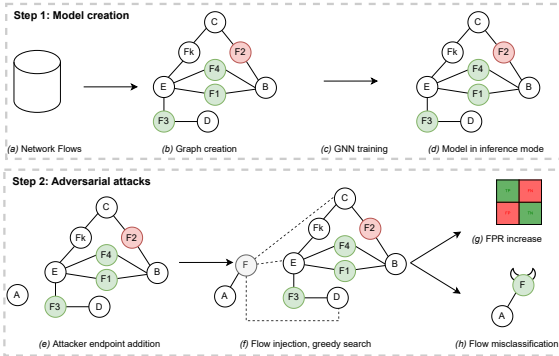


Figure 2: Overview of the adversarial attack. The network flows (a) are converted to a graph representation (b), a GNN is trained on available labels (c), and its parameters are fixed (d). The attacker adds its attacker-controlled endpoint, namely node A (e), and starts a greedy search-based flow injection (f) to maximize its objective (g, h). Here, nodes C, D and E are candidates to become the target endpoint of the added flow F.

Our central position is that the neighborhood aggregation mechanism making GNNs effective for intrusion detection is simultaneously their primary structural vulnerability. Purely additive flow injection is sufficient to exploit it systematically under realistic operational constraints. Our study is designed to answer the following Research Questions:

RQ1: Attack feasibility with injecting flows.

Can a flow injection-based adversarial attack succeed under the defined reachability constraints and black-box attack setting?

RQ2: Structural properties influencing attack difficulty. Which structural properties of victim endpoints determine the required number of injected flows? The investigation examines the influence of the endpoint degree (number of connected flows) and purity (ratio of benign versus malicious flow neighbors).

RQ3: Side effects. Does the designed attack have undesirable or unexpected effects on non-targeted network flows or does it remain stealthy and focused on the attacker’s objective?

RQ4: Attack transferability. Can attacks designed against a specific model be effective against different model architectures and other graph representations? This would enable an attacker to design attacks against an unknown model by optimizing it against a surrogate model.

To address these questions, we present the design of a flow-injection driving adversarial attack against GNN-based NIDS (see Fig. 2). It is formalized as a combinatorial optimization problem, adding new communications incrementally to the network rather than modifying flow features. This formulation is identified as a weakly submodular optimization problem. A greedy search strategy can thus efficiently craft adversarial attacks [Wang et al., 2020]. Experiments were performed on a widely used NIDS dataset using different model architectures.

2 Background

Graph-based representations of network traffic enable detection models that exploit structural relationships amongst flows. Three main representations exist: **flow graphs** [Zhong et al., 2024, Lin et al., 2024, Deng and Huang, 2025, Chang and Branco, 2021] model endpoints as nodes and flows as edges (edge classification); **line graphs** [Friji et al., 2023] transform flows into nodes (node classifica-

tion) at the cost of high density; and **heterogeneous graphs** [Pujol-Perich et al., 2022] combine both with typed nodes, meaning that structural attacks designed for it remain effective across representation, as confirmed empirically in Sect. 6.5. This paper adopts the latter approach, using the *flow* and *endpoint* node types. GNNs have been applied to network intrusion detection to exploit network traffic structure by aggregating features from local neighbourhoods and capturing multi-hop dependencies. Standard architectures (GCN [Kipf and Welling, 2017], GraphSAGE [Hamilton et al., 2017], GAT [Veličković et al., 2018]) are used for line graphs; E-GraphSAGE [Lo et al., 2022] extends GraphSAGE to edge features for flow graphs; and heterogeneous message-passing [Pujol-Perich et al., 2022] handles typed-node graphs.

Adversarial Attacks on GNN-based NIDS. Adversarial attacks against GNNs are categorized as **feature manipulation** [Zügner et al., 2018] or **structural** [Bojchevski and Günnemann, 2019]. For structural attacks, since graphs are discrete, gradient descent is not applicable; a greedy search selecting the perturbation with the highest marginal gain is standard. In the NIDS context, domain-compliance constraint [Pierazzi et al., 2020] further restricts feasible perturbations: existing flows cannot be removed or modified. Merzouk et al. [Merzouk et al., 2022] enforce validity constraint; Venturi et al. [Venturi et al., 2024] explore flow injection but ignore port reachability; Galli et al. [Galli et al., 2025] show explainability methods expose exploitable structure. These works highlight how structural information enabling efficient classification simultaneously guides attack crafting.

3 Heterogeneous Graph Representation of Network Traffic

We adopt a heterogeneous graph representation similar to [Pujol-Perich et al., 2022] with two disjoint node types: **flow** nodes V_f representing network flows with feature vectors (e.g., packet counts, byte counts); and **endpoint** nodes V_{ep} ($IP:Port$ pairs with randomly initialized features). Each flow node connects to exactly two endpoint nodes (source and destination), forming a bipartite structure, e.g. flow F from $IP_A:P_1$ to $IP_B:P_2$ creates nodes $IP_A:P_1$ (source) and $IP_B:P_2$ (destination) linked to flow node F . A GNN is trained on a node subset to learn its parameters. For the two relations $ep \rightarrow fl$ and $fl \rightarrow ep$, message-passing

for relation $\beta \rightarrow \alpha$ is:

$$m_{\mathcal{N}(a)}^{(l+1)} = \text{AGGR} \left\{ \mathbf{h}_b^{(l)}, \forall b \in \mathcal{N}(a) \right\} \quad (1)$$

$$\mathbf{h}_a^{(l+1)} = \sigma \left(\mathbf{W}_{\beta \rightarrow \alpha}^{(l)} \text{CONCAT} \left(\mathbf{h}_a^{(l)}, m_{\mathcal{N}(a)}^{(l+1)} \right) \right) \quad (2)$$

where a denotes a node of type α , b a node of type β and $\mathcal{N}(a)$ the neighbouring nodes of node a which are of type β , σ denoting a non-linear activation function and AGGR an aggregation function (mean-pooling for SAGE [Hamilton et al., 2017], weighted attentions for GAT [Veličković et al., 2018] or weighted summation for GCN [Kipf and Welling, 2017]). After L layers, each flow node’s embedding is passed to an MLP classifier returning logits $\mathbf{z}$ with predicted label $\hat{y}_i = \arg \max_c \mathbf{z}[c]$, where c indexes the output classes. For binary classification, the **prediction margin** is defined as:

$$\delta_i = \mathbf{z}_i[0] - \mathbf{z}_i[1] \quad (3)$$

where 0 denotes benign, and 1 denotes malicious. A positive margin $\delta_i > 0$ indicates a benign class prediction, while $\delta_i < 0$ indicates a malicious class prediction. Higher absolute margins indicate greater confidence in a benign classification.

4 Threat Model

4.1 Attacker’s Capabilities.

The attacker controls a single external endpoint $\mathbf{v}_a \in V_{ep}$ and injects new flow nodes toward reachable destinations within the monitored network. To ensure attack realism, the attacker complies with two constraints:

Constraint 1 (C1): Additive-only modification. The attacker cannot modify or remove existing communications observed in the network graph to avoid temporal inconsistencies.

Constraint 2 (C2): Endpoint reachability. The attacker can inject flows only from its controlled endpoint $\mathbf{v}_a$ to destination endpoints within the targeted network that have reachable service ports. This ensures the attack complies with the network boundary monitored by the NIDS and service availability.

4.2 Attacker’s Knowledge

The attacker operates in a black-box setting: no model parameters or gradients are known; only soft-label (logit) query outputs are available.

Snapshot-based graph knowledge: the attacker reconstructs the traffic graph from a historical snapshot (e.g., days or weeks of observed flows). In practice, minor discrepancies with the NIDS’s internal

graph may arise from preprocessing [Merzouk et al., 2022]; this imperfect knowledge is out of scope (see Sect. 7). Statistical patterns are available via public datasets or passive sniffing.

Destination endpoint identification: endpoints previously observed as destinations are assumed reachable.

4.3 Attack Objectives

We consider two attack objectives in the study. **Evasion:** the attacker initiates a malicious flow toward a victim endpoint. The objective is its misclassification as benign by the NIDS. **Denial of Service:** the attacker aims to increase the model’s False Positive Rate (FPR) by injecting flows that alter the graph structure, causing neighboring flows with benign features to be misclassified as malicious. This induces excessive false alarms, degrading NIDS reliability through alert fatigue.

Using the notation from Sect. 3, the original traffic graph is denoted as $\mathbf{G} = (\mathbf{V}_{fl} \cup \mathbf{V}_{ep}, \mathbf{E})$. The attacker constructs a modified graph $\mathbf{G}'$ by injecting a small number of flows from its controlled endpoint $\mathbf{v}_a$ to selected destination endpoints, identified through queries. In practice, legitimate flows may interleave with injected ones; their impact is discussed in Sect. 7. The attack is formulated as:

$$\mathbf{G}^* = \arg \max_{\mathbf{G}' \text{ satisfies C1, C2}} \ell_{\text{atk}}(\mathbf{G}') \quad (4)$$

where the objective function ℓ_{atk} , defined in Sect. 5, depends on the attack mode. The problem is NP-hard due to its combinatorial nature, requiring an efficient approximation algorithm (Algorithm 1).

5 Attack Methods

We first define the validity of the flow injection attack protocol. Then, we specify attack objectives $\ell_{\text{atk}}(\mathbf{G}')$ for two attack scenarios under constraints **C1** and **C2** and present a unified, greedy, search-based flow-injection method.

Protocol validity. Injected flows must have protocol-consistent features (packet counts, flags, size distributions). We assume injected flows use feature centroids of benign or malicious traffic ($\mathbf{x}_{\text{benign}}$, $\mathbf{x}_{\text{malicious}}$), obtainable from packet statistics. The feasibility of this assumption is discussed in Sect. 7.

Evasion. The goal is to construct $\mathbf{G}'$ such that a malicious flow $\mathbf{v}_{ab}$ in the graph is misclassified as benign. The attacker injects flows with features $\mathbf{x}_{\text{benign}}$

Algorithm 1: Greedy Flow Injection Attack

Input : Graph $\mathbf{G} = (\mathbf{V}_{fl} \cup \mathbf{V}_{ep}, \mathbf{E})$, model f_θ , budget K , attack scenario; for evasion: victim endpoint $\mathbf{v}_b \in \mathbf{V}_{ep.dst}$

Output: Modified graph $\mathbf{G}'$ satisfying C1–C2

Initialise $\mathbf{G}' \leftarrow \mathbf{G}$;
 Add attacker endpoint $\mathbf{v}_a$ to $\mathbf{G}'$;
if *attack scenario* = *evasion* **then**
 Inject attack flow f_{ab} : $(\mathbf{v}_a \rightarrow \mathbf{v}_b)$ with features $\mathbf{x}_{\text{attack}}$ and add it to $\mathbf{G}'$;
 Set objective $\ell_{\text{atk}} \leftarrow \ell_{\text{evasion}}$ (Eq. 5);
 Set $\mathbf{x}_{\text{scenario}} = \mathbf{x}_{\text{benign}}$;
else
 Set objective $\ell_{\text{atk}} \leftarrow \ell_{\text{DoS}}$ (Eq. 6);
 Set $\mathbf{x}_{\text{scenario}} = \mathbf{x}_{\text{malicious}}$;
for $t \leftarrow 1$ **to** K **do**
 $\ell_{\text{current}} \leftarrow \ell_{\text{atk}}(\mathbf{G}')$;
 $\ell_{\text{max}} \leftarrow -\infty$;
 $\mathbf{G}_{\text{best}} \leftarrow \mathbf{G}'$;
 for each $\mathbf{v} \in \mathbf{V}_{ep.dst}$ **do**
 Create candidate flow f_v : $(\mathbf{v}_a \rightarrow \mathbf{v})$ with features $\mathbf{x}_{\text{scenario}}$;
 Construct temporary graph $\mathbf{G}_{\text{temp}} \leftarrow \mathbf{G}' \cup \{f_v\}$ with edges $(\mathbf{v}_a, f_v)$ and $(f_v, \mathbf{v})$;
 Compute $f_\theta(\mathbf{G}_{\text{temp}})$;
 $\ell_{\text{temp}} \leftarrow \ell_{\text{atk}}(\mathbf{G}_{\text{temp}})$;
 $\Delta\ell \leftarrow \ell_{\text{temp}} - \ell_{\text{current}}$;
 if $\Delta\ell > \ell_{\text{max}}$ **then**
 $\ell_{\text{max}} \leftarrow \Delta\ell$;
 $\mathbf{G}_{\text{best}} \leftarrow \mathbf{G}_{\text{temp}}$;
 $\mathbf{G}' \leftarrow \mathbf{G}_{\text{best}}$;
 if *attack scenario* = *evasion* **and** $\delta_{ab}(\mathbf{G}') > 0$ **then**
 break ; // Attack succeeded
return $\mathbf{G}'$;

and maximises the classification loss over the malicious flow $\mathbf{v}_{ab}$:

$$\ell_{\text{evasion}}(\mathbf{G}') = \delta_{\mathbf{v}_{ab}} = z_{\mathbf{v}_{ab}}[0] - z_{\mathbf{v}_{ab}}[1], \quad (5)$$

where $z_{\mathbf{v}_{ab}}$ are the logits and $\delta_{\mathbf{v}_{ab}}$ is the margin (Eq. 3). Maximising this objective drives $\delta_{\mathbf{v}_{ab}} > 0$, leading to misclassification.

Denial of Service. The goal is to increase the misclassification risk over all the benign traffic flows in the graph as much as possible. For $B_{\text{test}} \subset \mathbf{V}_{fl}$, the set of initially correctly classified benign flows, the attacker injects flows with features $\mathbf{x}_{\text{malicious}}$ and maximises

$$\ell_{\text{DoS}}(\mathbf{G}') = - \sum_{j \in B_{\text{test}}} \delta_i = \sum_{j \in B_{\text{test}}} (z_j[1] - z_j[0]). \quad (6)$$

This reduces the margins δ_i (Eq. 3), causing misclassification when they become negative.

The greedy algorithm 1 iteratively adds up to K flow nodes, selecting at each step the destination endpoint that maximises the marginal gain in ℓ_{atk} . At iteration $t \in 1, \dots, K$, each candidate destination v is evaluated by temporarily adding a flow from v_a to v , computing the objective, and choosing the best improvement. The NP-hard optimisation problem in Eq. 4 is thus approximated via a greedy strategy with guarantees for weakly submodular maximization problems [Wang et al., 2020].

6 Experimental Evaluation

6.1 Experimental Setup

NIDS Dataset. We conduct preliminary experiments using the X-IIoTID dataset [Al-Hawawreh et al., 2022], preprocessed to keep only network-related features.

GNN model. Four GNN architectures are evaluated, including GraphSAGE [Hamilton et al., 2017], GCN [Kipf and Welling, 2017], GAT [Velićković et al., 2018], and GIN [Xu et al., 2019], each with 2 and 4 layers. We use the Adam optimizer, with a learning rate of 0.001 and weight decay of 10^{-4} , training for 500 epochs to ensure convergence. A temporal 60/20/20 train/validation/test split prevents data leakage [Arp et al., 2022]; all models exceed 0.97 F1 on clean data. The flow injection attack is then performed on the testing data.

Metrics. Attack performance is measured by the **Attack Success Rate (ASR)** for evasion (fraction of targeted flows misclassified as benign) and the increase in **False Positive Rate (FPR)** for DoS. We also report the number of injected flows required, used as a proxy for attack cost.

6.2 RQ1: Attack Feasibility Evaluation

Targeted Evasion Attack. The attack achieves an ASR of 84.5% on a 2-layer GraphSAGE. The injection budget distribution is highly skewed with a median of 6 and a 90% quantile of 37 flows. This reflects two endpoint categories: easy targets requiring fewer than 6 flows, and resistant ones needing up to five times more, consistent with the structural analysis in Section 6.3.

Diminishing marginal returns. Figure 3 shows the prediction margin exhibiting diminishing marginal returns for the evasion attack: rapid initial growth followed by a plateau. If early injections fail to shift

the margin significantly, further additions are unlikely to cross the decision boundary, consistent with the weakly submodular attack objective [Wang et al., 2020].

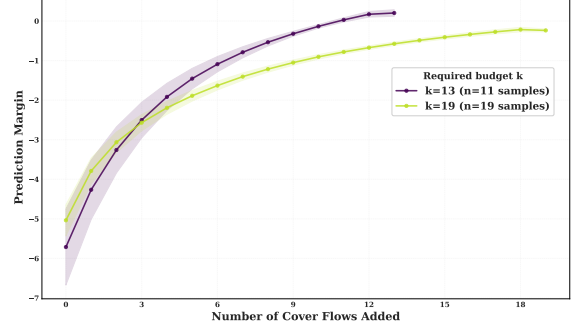


Figure 3: Trajectory of margin changes at each attack step, for evasion attack instances, on the X-IIoTID dataset against a 2-layer GraphSAGE GNN. Two groups of attack instances are representing with k , the minimum number of injected flows required to achieve misclassification, being respectively 13 and 19.

Denial-of-Service Attack. In this scenario, the attacker iteratively adds malicious flows to increase the FPR of the NIDS. The relative increase in the FPR is reported as a function of the number of added flows. The experiments, using a GraphSAGE model as the victim, show limited results. Indeed, the increase in FPR is minimal (from 0.0420 to 0.0432) despite injecting 3,000 flows. The few affected flows are connected to low-degree endpoints. Moreover, benign flows form homophilic communities, giving the GNN a strong prior for benign classification that the attack’s dispersed injections cannot overcome. On the contrary, in the evasion attack, the injected flows are concentrated around the victim endpoint, maximizing local impact on the aggregation. This suggests that GNN-based NIDS are vulnerable to targeted evasion attacks, whereas Denial-of-Service attacks are more difficult to execute.

6.3 RQ2: Influence of Structural Properties

Detection output depends on two local structural properties: the *endpoint degree* (number of connected flows) and *neighborhood label purity* (fraction of malicious flow neighbors). We examine their impact on both attack modes.

6.3.1 Targeted Evasion Attack

Degree distribution. In the evasion setting, attacks predominantly (91.2%) target the endpoint directly

connected to the victim flow. Table 1 presents the number of required cover flow nodes, stratified by the degree of the endpoint connected to the targeted flow node. Low-degree endpoints ($\text{deg}=1$, $\text{deg}=2$) require significantly fewer cover flows than moderate degree nodes ($\text{deg}=6-50$), exhibiting a positive relationship between degree and required attack budget. For very high-degree endpoints ($\text{deg}>50$), the limited sample size ($n=15$) and high variance ($\sigma=34.20$) prevent definitive conclusions about the relationship between degree and attack difficulty.

Table 1: Number of required cover flow nodes by the degree of endpoint connected to the targeted flow node, on X-IIoTID dataset against GraphSAGE with 2 layers.

Degree Group	n	Mean (μ)	Median	Std Dev (σ)
$\text{deg} = 1$	54	5.37	4.0	6.24
$\text{deg} = 2$	118	7.03	5.0	6.41
$\text{deg} = 3-5$	96	11.89	4.0	13.15
$\text{deg} = 6-50$	45	45.04	45.0	31.12
$\text{deg} > 50$	15	25.20	11.0	34.20

Label purity. Malicious neighborhood purity (fraction of malicious flow neighbors) strongly predicts attack cost (Fig. 4): purity above 50% requires, on average, 16 injected flows, twice that of lower purity neighborhoods, as message passing creates a strong prior for malicious classification.

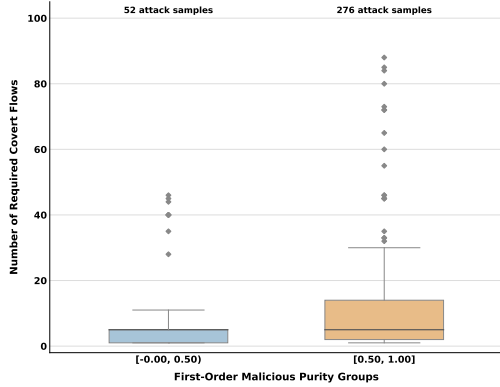


Figure 4: Boxplot of the number of injected cover flow nodes for successful evasion attacks, per malicious purity group, on X-IIoTID against a 2-layer SAGE GNN.

6.3.2 Denial-of-Service Attack

Degree distribution. The destination endpoints targeted by injected flows have a mean degree of 40377 and a median of 3937, compared to an average degree of 4.52 over all endpoint nodes, corresponding to high-degree servers that influence many benign flows. From the attacker’s perspective, the goal is to misclassify as many benign flows as possible with the fewest

Table 2: Transferability attack success rate (ASR) for attacks designed against GraphSAGE ($L = 2$) on X-IIoTID

Target	L	K	$K + 1$	$K + 2$	TPR@FPR
GAT	2	0.113	0.113	0.119	0.963
GAT	4	0.357	0.479	0.642	0.601
GCN	2	0.872	0.930	0.948	0.953
GCN	4	0.384	0.479	0.633	0.956
GIN	2	0.143	0.216	0.275	0.983
GIN	4	0.317	0.463	0.581	0.984
GraphSAGE	4	0.905	0.942	0.942	0.972

additional flows. Thus, the attacker preferably targets high-degree endpoint nodes, as they impact a larger number of benign flows.

Label purity. Injected flows target benign-dominated endpoints to maximize false alarms, but dispersing injections across the graph conflicts with maximizing local impact, explaining the DoS strategy’s ineffectiveness.

6.4 RQ3: Side Effects

We assess whether injections targeting a single flow cause unintended misclassification elsewhere. **Misclassification:** no non-targeted node changed label, confirming high locality and low defender-alerting collateral. **Margin degradation:** measurable maximum softmax probability reduction appears in approximately 0.88% of flows in the graph, more pronounced on malicious flows (-1.36 percentage points) than benign (-0.17 pp), but without label flips. **Spatial locality:** affected nodes are within 1-2 hops of the target (mean hop distance: 1.19), consistent with GNN message-passing range.

6.5 RQ4: Attack Transferability

We evaluate **cross-architecture transferability**. For each targeted flow, we define K as the minimum number of injected cover flows required to achieve misclassification on the surrogate model, which varies across samples. We then evaluate transferability under three scenarios with K , $K + 1$, and $K + 2$ injections. The attack transfers well to GCN with 2 layers and GraphSAGE with 4 layers, which share similar mean-aggregation boundaries. Moderate transferability to deeper GCN, GAT, and GIN improves with extra budget. GAT and GIN, with 2 layers, show limited transfer: GAT’s attention mechanism induces distinct decision boundaries, reducing susceptibility to mean-aggregation-optimized attacks.

Across graph representations. Attacks also transfer well across graph representations, from hetero-

geneous graph to flow-graph (E-GraphSAGE in Table 3), consistent with the heterogeneous structure generalizing the flow-graph representation.

Table 3: ASR for E-GraphSAGE models [Lo et al., 2022] with 2 and 4 layers, at different budgets of injected cover flows, from the required budget K to $K + 2$ to successfully attack SAGE with 2 layers on a heterogeneous graph representation.

Target Model	$K +$	ASR (%)
E-GraphSAGE (2 layers)	0	86.6
E-GraphSAGE (2 layers)	1	90.2
E-GraphSAGE (2 layers)	2	90.2
E-GraphSAGE (4 layers)	0	87.2
E-GraphSAGE (4 layers)	1	91.2
E-GraphSAGE (4 layers)	2	91.1

7 Open Problems and Research Agenda

Our results surface four open problems that define the agenda for this line of work.

Hard-label constraints. The current evaluation assumes soft-label access. In the harder, more common hard-label setting, marginal gain estimates are unavailable, requiring score estimation or fundamentally different search strategies. Characterizing when attacks remain feasible — likely for low-degree endpoints but not high-purity ones — is an open problem.

Joint structural and feature optimisation. Fixing injected features to class centroids sidesteps the inverse-feature mapping problem [Pierazzi et al., 2020] but leaves attack power on the table. Joint optimisation over injection topology and flow features would likely reduce the required budget, but require navigating a mixed discrete-continuous search space for which no principled, tractable method currently exists in the NIDS setting.

Temporal graph dynamics. Our static snapshot evaluation cannot capture how the continuous graph evolution, like endpoints appearing and disappearing, neighbourhoods shifting, affects attack stability. Whether the local nature of structural attacks provides inherent robustness to moderate graph drift, or whether an online attack strategy is required, can only be answered in dynamic simulators or live virtualised environments.

Robustness by design. The neighbourhood aggregation that makes GNNs effective for NIDS is precisely what makes them structurally exploitable. Post-hoc defences are difficult to apply given the discrete, dynamic nature of network graphs. We argue that adversarial robustness must be a first-class design

constraint — for instance through architectures that bound individual node influence or that flag anomalous injection patterns directly. Developing such intrinsically robust architectures is, in our view, the most critical open problem raised by this work.

8 Conclusion

Our results show that GNN-based NIDS remain vulnerable to problem-space adversarial attacks, even when attackers are limited to injecting valid flows from a single endpoint. Although these models resist untargeted Denial-of-Service attempts, targeted evasion attacks exploit local structural properties — rather than global statistics or specific features — to misclassify flows with high stealth and cross-architecture transferability. This weakness reflects a core paradox: the same message-passing mechanism that captures inter-flow dependencies also enables attackers to influence a flow’s classification without modifying it. As a result, connectivity itself becomes an exploitable vector, highlighting the need to embed adversarial robustness into GNN-based security design. The open problems identified in Sect. 7 chart the path toward more resilient architectures and a broader understanding of the boundaries of GNN-based NIDS under realistic adversarial conditions.

Acknowledgment

We used DeepL and Gemini to polish the paper’s text.

This work has been partially supported by the French National Research Agency (ANR) under the France 2030 label (SuperViz ANR-22-PECY-0008).

Experiments presented in this paper were carried out using the Grid’5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see <https://www.grid5000.fr>).

REFERENCES

- Al-Hawawreh, M., Sitnikova, E., and Aboutorab, N. (2022). X-iiotid: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things. *IEEE Internet of Things Journal*, 9(5):3962–3977.
- Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., and Rieck, K. (2022). Dos and don’ts of ma-

- chine learning in computer security. In *Proc. of USENIX Security Symposium*.
- Bojchevski, A. and Günnemann, S. (2019). Adversarial attacks on node embeddings via graph poisoning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 695–704. PMLR.
- Catillo, M., Repola, A., Pecchia, A., and Villano, U. (2024). Towards realistic problem-space adversarial attacks against machine learning in network intrusion detection. In *Proceedings of the 19th International Conference on Availability, Reliability and Security, ARES '24*. Association for Computing Machinery.
- Chang, L. and Branco, P. (2021). Graph-based solutions with residuals for intrusion detection: The modified e-GraphSAGE and e-ResGAT algorithms. *ArXiv*, abs/2111.13597.
- Deng, P. and Huang, Y. (2025). Edge-featured multi-hop attention graph neural network for intrusion detection system. *Computers & Security*, 148:104132.
- Friji, H., Olivereau, A., and Sarkiss, M. (2023). Efficient Network Representation for GNN-Based Intrusion Detection. In *Applied Cryptography and Network Security: 21st International Conference, ACNS 2023, Kyoto, Japan, June 19–22, 2023, Proceedings, Part I*, pages 532–554. Berlin, Heidelberg. Springer-Verlag.
- Galli, D., Venturi, A., Marasco, I., and Marchetti, M. (2025). Evaluating Explainability of Graph Neural Networks for Network Intrusion Detection with Structural Attacks. In *CEUR WORKSHOP PROCEEDINGS*, volume 3962. CEUR-WS.
- Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. *Proceedings of NeurIPS*, 30.
- He, K., Kim, D. D., and Asghar, M. R. (2023). Adversarial machine learning for network intrusion detection systems: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 25(1):538–566.
- Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR 2017*. OpenReview.net.
- Lin, L., Zhong, Q., Qiu, J., and Liang, Z. (2024). E-GRACL: An IoT intrusion detection system based on graph neural networks. *The Journal of Supercomputing*, 81(1):42.
- Lo, W. W., Layeghy, S., Sarhan, M., Gallagher, M., and Portmann, M. (2022). E-GraphSAGE: A Graph Neural Network based Intrusion Detection System for IoT. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9.
- Merzouk, M. A., Cuppens, F., Boulahia-Cuppens, N., and Yaich, R. (2022). Investigating the practicality of adversarial evasion attacks on network intrusion detection. *Annals of Telecommunications*, 77(11-12):763–775.
- Pierazzi, F., Pendlebury, F., Cortellazzi, J., and Cavallaro, L. (2020). Intriguing properties of adversarial ML attacks in the problem space. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1332–1349.
- Pujol-Perich, D., Suarez-Varela, J., Cabellos-Aparicio, A., and Barlet-Ros, P. (2022). Unveiling the potential of Graph Neural Networks for robust Intrusion Detection. *ACM SIGMETRICS Performance Evaluation Review*, 49(4):111–117.
- Sheatsley, R., Papernot, N., Weisman, M. J., Verma, G., and McDaniel, P. (2022). Adversarial examples for network intrusion detection systems. *Journal of Computer Security*, 30(5):727–752.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *ICLR 2018*.
- Venturi, A., Stabili, D., and Marchetti, M. (2024). Problem space structural adversarial attacks for Network Intrusion Detection Systems based on Graph Neural Networks.
- Vitorino, J., Praça, I., and Maia, E. (2023). Towards adversarial realism and robust learning for iot intrusion detection and classification. *Annals of Telecommunications*, 78(7–8):401–412.
- Wang, Y., Han, Y., Bao, H., Shen, Y., Ma, F., Li, J., and Zhang, X. (2020). Attackability Characterization of Adversarial Evasion Attack on Discrete Data. In *Proceedings of the 26th ACM KDD, KDD '20*, pages 1415–1425.
- Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *ICLR*.
- Zhong, M., Lin, M., Zhang, C., and Xu, Z. (2024). A survey on graph neural networks for intrusion detection systems: Methods, trends and challenges. *Computers & Security*, 141:103821.
- Zügner, D., Akbarnejad, A., and Günnemann, S. (2018). Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of the 24th ACM KDD*, pages 2847–2856.