Evaluating Explainability of Graph Neural Networks for Network Intrusion Detection with Structural Attacks

Dimitri Galli[1,*,†], Andrea Venturi[1,†], Isabella Marasco[2] and Mirco Marchetti[1]

[1]University of Modena and Reggio Emilia, Department of Engineering "Enzo Ferrari", 41125 Modena, Italy
[2]University of Bologna, Department of Computer Science and Engineering, 40126 Bologna, Italy

Abstract

Among Machine Learning (ML) models, Graph Neural Networks (GNN) have been shown to improve the performance of modern Network Intrusion Detection Systems (NIDS). However, their black-box nature poses a significant challenge to their practical deployment in the real world. In this context, researchers have developed eXplainable Artificial Intelligence (XAI) methods that reveal the inner workings of GNN models. Despite this, determining the most effective explainer is complex because different methods yield different explanations, and there are no standardized strategies. In this paper, we present an innovative approach for evaluating XAI methods in GNN-based NIDS. We evaluate explainers based on their capability to identify key graph components that an attacker can exploit to bypass detection. More accurate XAI algorithms can identify topological vulnerabilities, resulting in more effective attacks. We assess the effectiveness of different explainers by measuring the severity of structural attacks guided by the corresponding explanations. Our case study compares five XAI techniques on two publicly available datasets containing real-world network traffic. Results show that the explainer based on Integrated Gradients (IG) generates the most accurate explanations, allowing attackers to refine their strategies.

Keywords

Explainable Artificial Intelligence, Graph Neural Network, Network Intrusion Detection

1. Introduction

Machine Learning (ML) and Deep Learning (DL) algorithms can enhance the capabilities of modern Network Intrusion Detection Systems (NIDS) [1]. Recent research shows that ML methods improve the classification of cyber attacks, reducing the reliance on manual rule creation. However, traditional ML-based NIDS treat data features as independent variables and data points as individual samples, limiting their effectiveness in capturing the complex dependencies of modern multi-flow attacks in real-world scenarios [2, 3]. To overcome this limitation, Graph Neural Networks (GNN) analyze both individual features and topological structures of network traffic during the training process [4]. Indeed, GNN are a family of neural networks capable of processing network hosts and their communications as nodes and edges within a graph. Since graphs represent the inherent inter-dependencies within network traffic, GNN models can detect malicious patterns exhibited at the topological level [5].

Despite their effectiveness, GNN models operate as *black-boxes*. This lack of transparency hinders their use in practical contexts [6], where security analysts need to understand why a cyber detector flags some flows as malicious [7]. Explainable Artificial Intelligence (XAI) techniques attempt to bridge this gap by defining *explanations* that identify which components in the network graph influence the decision-making of GNN models [8]. In this context, different XAI methods could be used to explain GNN predictions. However, each explainer exploits its mechanisms to identify the most relevant structures. Therefore, different explainers may return different explanations, making it unclear which method should be considered correct. There are no automated frameworks to evaluate the

Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), February 03–8, 2025, Bologna, IT
*Corresponding author.
†These authors contributed equally.
dimitri.galli@unimore.it (D. Galli); andrea.venturi@unimore.it (A. Venturi); isabella.marasco4@unibo.it (I. Marasco); mirco.marchetti@unimore.it (M. Marchetti)
0009-0006-0280-2498 (D. Galli); 0000-0003-3822-968X (A. Venturi); 0009-0006-9074-6886 (I. Marasco); 0000-0002-7408-6906 (M. Marchetti)

quality of explanations in GNN-based NIDS. Furthermore, existing methodologies do not consider the dynamic nature of network traffic. Existing evaluation approaches rely on expensive ground truth explanations [9]. Therefore, these frameworks require importance labels to be available a priori, which limits their use in constantly evolving scenarios. Traditional metrics for evaluating XAI methods examine how model predictions change when relevant structures are isolated or removed from the complete graph. These strategies do not consider any constraints on the network structure and may disrupt the distributed topology of modern cyber attacks [10].

We aim to develop an evaluation framework that satisfies several key properties, such as being (i) *agnostic*, i.e., independent of the specific type of explainers; (ii) *flexible*, i.e., usable without the need for ground truth explanations; (iii) *practical*, i.e., useful in real-world scenarios. In this paper, we present an innovative methodology for comparing and evaluating explainers for GNN-based NIDS that fulfills these requirements. Our approach is based on structural attacks, where attackers manipulate the graph topology to evade detection [11, 12]. Indeed, these perturbations have proven to be effective against GNN-based NIDS [13]. In our proposed strategy, we adversarially change the network topology by injecting the components considered relevant by each explainer into the graph and measuring the severity of structural attacks. We assume that the most effective XAI techniques highlight the explanations that lead to the most impactful attacks.

We apply our approach to an experimental case study to demonstrate how our proposal identifies the most appropriate XAI methods for GNN-based NIDS, even without ground truth explanations. We compare five explainers tailored for graph-based models, considering two public datasets extensively used in network intrusion detection. We design a GNN-based NIDS that achieves good detection performance, allowing practical evaluation of the explainers. Our results reveal an overall increase in attack severity when attackers employ explanations, especially when they exploit Integrated Gradients (IG) to locate graph model topological vulnerabilities.

The paper is structured as follows. Section 2 provides background knowledge on GNN and XAI for NIDS. Section 3 describes the evaluation methodology. Section 4 details the experiments. Section 5 presents the results. Section 6 concludes the paper with final remarks.

2. Background and Related Work

ML-based NIDS use supervised algorithms to classify network traffic and detect malicious patterns [1]. To build such systems, ML algorithms are trained on labeled *netflows*, where each data entry reports a set of metrics and statistics—also referred to as *features*—that summarize the communication between two hosts in the monitored network [14]. Traditional ML-based NIDS analyze data features independently and consider each flow in isolation, enabling near real-time responses [15]. However, these models often struggle to detect modern attacks that rely on complex multi-flow techniques [5].

GNN are a family of DL algorithms specifically designed to learn from graph-structured data [16]. Indeed, these models can effectively analyze and capture graph patterns by exploiting flow features and topological structures [4]. Since netflows can be tagged—indicating whether they are benign or malicious—and transformed into graph structures, GNN-based NIDS can be used to classify network traffic following a supervised approach [5]. The development of these models typically involves several stages.

At the beginning, we collect the flows and represent them in a specific *graph* structure. A graph G is formally defined as $G = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is a set of *vertices* or *nodes*, and $\boldsymbol{E}$ is a set of *edges* or *links*. The *flow graph* representation is the most common transformation for a computer network because each endpoint in the monitored network represents a node, and each flow corresponds to a link in the graph [17]. However, the majority of GNN perform node classification using a *line graph* representation, where flows are mapped directly to graph vertices, which are connected if they share a host in the respective flows [18]. If a flow graph G has n nodes and m edges, then its line graph transformation $L(G)$ has m nodes and $\frac{1}{2}\sum_{i=1}^{n} d_i^2 - m$ edges, where d_i is the degree of the node i in the flow graph. This conversion is shown in the example in Figure 1.

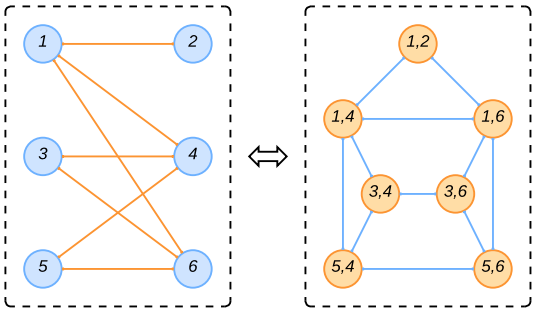

Figure 1: Flow graph and its line graph. The flow graph on the left has six nodes and seven edges. Its line graph on the right has seven nodes and eleven edges. In the line graph, nodes (1,2), (1,4), and (1,6) are connected because they correspond to edges in the flow graph with the same source node 1.

Next, we train the GNN model to transform graph nodes into *embeddings*. These embeddings encapsulate both flow features and structural similarities in high-dimensional vectors. In *transductive* settings, the GNN learns on a single graph [19]. To improve the generalization of the GNN, *inductive* strategies use different input graphs for training and testing the model [10]. Once generated, the embeddings with their respective labels can be used to train a *classifier*. Indeed, any ML model can distinguish between benign and malicious samples.

Despite their high detection performance, GNN-based NIDS are still underutilized in practical contexts [20]. The lack of transparency makes these models opaque to cybersecurity practitioners, who should understand the rationale behind the detector's predictions to validate alerts [8]. XAI algorithms applied to GNN use different approaches to provide insight into which components influence the inner workings of the models. These explainers extend beyond feature importance by defining a *mask* that assigns weights to the components in the graph based on their contribution to predictions [21]. We can have a *hard mask* if the explanation scores take binary values or a *soft mask* if the importance scores take continuous values. This mask may refer to a subgraph whose elements are related to their importance in interpreting model predictions.

Many explainability methods have been proposed for graph learning models. The paper referenced in [22] systemizes explainability methods for GNN based on the explanation target. Explainers working at the *instance-level* extract features relevant to the GNN output. In contrast, *model-level* explainers provide a general understanding of the model. Explanation methods can be further classified based on their integration with the ML model, as reported in [23]. *Post-hoc* methods act as external components dealing with pre-trained models with fixed weights. *Self-interpretable* methods are directly integrated into the neural network. Our analysis considers instance-level and post-hoc explainers, as they are used in real-world scenarios. Due to code availability, we focus on *gradient-based* methods [24, 25], which estimate importance scores by computing the gradient of the GNN, and *perturbation-based* methods [26, 27], which perturb the input graph by removing nodes or rewiring edges to obtain the explanation subgraph.

Compared to evaluating traditional ML models, the evaluation of the quality of different explanations is a complex task. On the one hand, *supervised* approaches [28, 29] compare the explanation with a ground truth importance. These approaches assume that the elements critical to a particular prediction are known, enabling an objective and quantitative evaluation of explanation methods. However, these strategies require human supervision to decide what is important for the model. Consequently, generating ground labels is *time-consuming*, especially when working with real-world datasets [30]. Security analysts should investigate complex dependencies that require significant effort and domain expertise. Our evaluation framework does not rely on ground truth, making it adaptable and flexible to any dataset. On the other hand, *unsupervised* approaches [31, 32] evaluate how explanations extracted by XAI methods influence model predictions, either by isolating or removing the significant components.

These approaches are flexible as they can rely on existing metrics and do not require ground truth explanations. However, explainability methods aim to identify the most relevant factors, generating explanations that should be small and sparse. Therefore, important graphs may be *out-of-distribution*, leading to an incorrect evaluation [33]. Explanatory subgraphs can extract components that do not represent the distributed topology of modern attacks. Instead of removing explanations from the original data distribution, our methodology identifies and injects input key components into the dataset, perturbing the graph topology while preserving the extracted structures.

3. Methodology

As discussed in Section 2, evaluating explainability methods in GNN-based NIDS is a challenging task, as it often depends on expensive ground truth labels or unrealistic explanatory subgraphs. We present an innovative unsupervised evaluation framework that leverages structural adversarial attacks to compare different explainers. More specifically, our strategy is based on the detection accuracy of GNN-based NIDS under realistic structural adversarial attacks [13]. The proposed methodology evaluates the overall quality of the explainability methods by analyzing their contribution to attacking the GNN model.

By design, different explainability methods highlight the graph components that are important for the model predictions. The general idea of our methodology is to evaluate explainers by observing how well structural adversarial attacks driven by explanations change the graph structure and thwart the GNN-based NIDS. To evaluate the quality of each method, we calculate the *attack severity (AS)* [34], which measures the degradation of the detector performance when key graph components perturb the input topology. Our hypothesis is straightforward: *accurate XAI algorithms identify the graph elements most critical for evasion, leading to higher AS when exploited.* Our framework identifies the most accurate XAI method and shows how attackers can exploit these insights to refine their strategies. Our strategy can further help security practitioners strengthen their defenses against such threats.

An example of the proposed methodology is shown in Figure 2. To improve readability, we model network traffic using a flow graph. In this model, nodes and edges represent hosts and communications. The compromised host c_1 is controlled by the adversary and coordinates the botnet with nodes c_2, c_3, c_4 to flood the victim node u_3 with a massive amount of packets. A GNN-based NIDS detects DDoS attacks where different hosts send packets to a single target node. In particular, the GNN is trained to detect malicious patterns by flagging the edges associated with them as suspect. The explainer extracts a mask containing the most important flow records for the model. These samples correspond to edges within the flow graph critical to detecting cyber attacks. These elements enhance structural attacks to evade the GNN-based NIDS. The XAI method that identifies the samples leading to more misclassifications is considered the most precise.

Below we define the deployment scenario and describe the two phases of the proposed evaluation strategy.

3.1. Deployment Scenario

We consider the same deployment scenario as previously proposed in [18]. The corporate network includes multiple devices and a single border router that facilitates communication for all the hosts. We also assume that a remote attacker has built his own C&C infrastructure by compromising one or more devices to perform malicious operations. We suppose that a GNN-based NIDS monitors the internal network by analyzing the graph representation of network traffic.

Network packets passing through the border router are captured by a *flow exporter*, which extracts the corresponding *flow records* $\boldsymbol{D}$. These samples, collected during a specific time window, are processed by a *graph generator* to produce the *graph* G. In this evaluation methodology, we consider a *flow graph* G, where the hosts in the internal network are associated with nodes, while the communications are associated with edges. In the experiments, we translate this graph representation into a *line graph* $L(G)$ by considering the edges in G as nodes in $L(G)$ and linking together two nodes in $L(G)$ if the corresponding edges in G share a vertex.

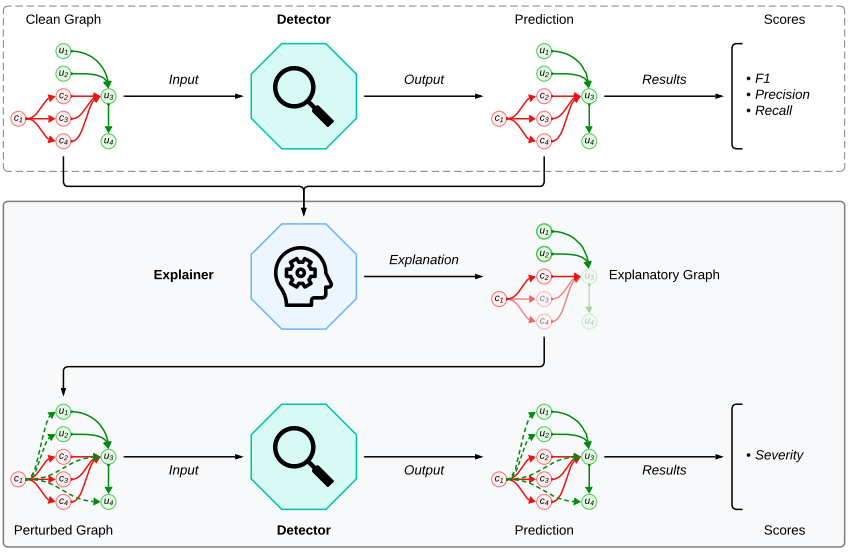

Figure 2: Evaluation framework. Green nodes and arrows denote uncompromised hosts u_i and benign flows $\mathscr{B}$, while red ones represent compromised hosts c_i and malicious netflows $\mathscr{M}$. In the upper box, the cyber detector learns to predict attacks on the clean graph G. In the lower box, the explainer generates the explanatory graph $G^\star$, whose benign edges are injected into the graph G to obtain the perturbed graph $\hat{G}$, which is submitted to the trained detector to evaluate the XAI method.

Once generated, the graph is processed by the GNN-based NIDS to produce the *embeddings* $\boldsymbol{Z} \in \mathbb{R}^{n \times d}$, where n is the number of flows (i.e., edges in the flow graph or equivalently nodes in the line graph), while d is the dimension of the latent space [35]. The embeddings allow the GNN model to perform classification as they encapsulate the f-dimensional features $\boldsymbol{X} \in \mathbb{R}^{n \times f}$ and binary labels $\boldsymbol{y} \in \{0,1\}^n$ of the corresponding flow records, but also represent the most significant structures in the network topology [36]. The final layer of the detector consists of a binary classifier that predicts whether the embeddings are legitimate or malicious. However, while the GNN-based NIDS performs the classification, the rationale behind the predictions remains unclear to security analysts, highlighting the need for XAI techniques.

3.2. Explaining

The first phase of our proposed framework uses explainability methods to identify the influential components in the input graph and provide insights into the GNN-based NIDS. In this phase, we aim to compute explanations that an attacker can exploit to manipulate the input graph structure.

As discussed in Section 2, we focus on *instance-based* explainers that are capable of highlighting specific graph components, omitting *model-based* explainers due to their heterogeneity. In this context, we follow a *post-hoc* approach, examining pre-trained models and excluding *self-explainable* strategies due to their infeasibility in the real world.

Given the flow graph G and the GNN predictions $\hat{\boldsymbol{y}}$, an explainer defines an *explanation mask*, i.e., a graph $G^\star = (\boldsymbol{V}^\star, \boldsymbol{E}^\star)$ where each node $v_i^\star \in \boldsymbol{V}^\star$ and edge $e_i^\star \in \boldsymbol{E}^\star$ has an importance value $h_i \in [0,1]$ representing its contribution to the intrusion detection. Since we want to compute a relevance value for each record in the dataset using a flow graph representation, we consider the edge mask of the *explanatory graph* $G^\star$, leaving out the node weights.

When the explanation is extracted, we select the most significant benign flows $\mathscr{B}^\star$. We focus on benign netflows $\mathscr{B}$ rather than malicious ones $\mathscr{M}$ because the structural adversarial attacks considered in our evaluation exploit legitimate records to manipulate the graph structure. In addition, benign communications dominate the network traffic, making them more accessible for analysis. In contrast, malicious transmissions are often rare in the real world, making it difficult to establish a consistent

baseline for evaluation. Adversaries who use benign flows to hide their malicious operations also pose a significant challenge to detection systems. Indeed, attackers can inject benign edges into the neighborhood of compromised nodes within the graph to effectively evade detection of malicious links.

To prioritize which benign flows have a major impact on the detector performance, we rank and select the top K important legitimate netflows $\mathscr{B}^\star$ based on the computed relevance values h_i. In the next step, we evaluate which explainer really captures the influential links, as different explainers may return different explanations.

3.3. Evaluation

The actual evaluation of the explainability algorithm happens in this phase. Our approach leverages structural attacks, where attackers manipulate the graph structure to evade the GNN-based NIDS. As explainability methods highlight important components within the graph, attacks based on these explanations reflect which method most correctly identifies the flows critical to the detection. In our assessment, the most effective explainer is the one that identifies netflows leading to the highest number of misclassified samples when submitted to the GNN-based NIDS.

At this stage, we consider a realistic *gray-box* attack scenario [37], where the attacker has partial knowledge of the defensive system. The attacker knows that a GNN-based NIDS monitors the network, but he has no insight into its parameters. We also assume that the attacker has a limited quantity of flow records available to define explanations for the input graph components. Hence, the attacker can compute an explanatory mask to identify the important benign netflows and perturb the input graph structure.

To evaluate the accuracy of different explainability algorithms, we focus on $\textbf{C2x}_{\mathscr{B}}$ attacks [13]. Here, attackers initiate new benign communications $\mathscr{B}$ from compromised hosts c_i to random targets u_i to manipulate the graph and thwart the GNN. In other words, attackers change the structural patterns of their attacks by perturbing the graph topology. Indeed, these perturbations generate embeddings that evade detection by tricking the GNN into making misclassifications. This approach is feasible in the real world because the attacker can execute any strategy on the compromised hosts c_i once he has complete control over them. Consequently, evaluating GNN explanations using $\textbf{C2x}_{\mathscr{B}}$ attacks constitutes a practical strategy. Although these attacks succeed in evading the GNN, the contribution of explanations is not considered in the threat model. Indeed, the attacks proposed in the original work [13] randomly select legitimate netflows without considering which ones might be the most relevant for the detection task.

In our framework, we do not randomly collect the benign flows to manipulate the graph. Instead, we rank and select the top K legitimate netflows based on their relevance scores h_i. We then use these important samples to evade the detector and evaluate the explainer. More specifically:

1. From the explanatory graph $G^\star$, we identify the most important benign flows $\mathscr{B}^\star$ and inject them into the original dataset $\boldsymbol{D}$, resulting in an augmented dataset $\hat{\boldsymbol{D}}$.
2. Given the perturbed dataset $\hat{\boldsymbol{D}}$, we build the corresponding flow graph $\hat{G}$, where important legitimate samples $\mathscr{B}^\star$ are associated with new edges.
3. The manipulated graph $\hat{G}$ leads to misclassifications when fed to the GNN-based NIDS, since the neighborhood of compromised nodes includes the benign edges $\mathscr{B}^\star$.
4. The effectiveness of the XAI technique is evaluated by measuring the impact of the injected legitimate transmissions $\mathscr{B}^\star$ into the graph on the GNN-based NIDS predictions.

4. Case Study

In Section 3, we described our evaluation framework to compare XAI methods in GNN-based NIDS. We now present an experimental case study to validate our proposed methodology. In particular, we present the datasets used, the GNN-based NIDS targeted, the different explainers tested, and the framework

Table 1
Data distribution for training and testing phases.

(a) CTU-13.

Botnet	Training		Testing	
	(#Ben)	(#Mal)	(#Ben)	(#Mal)
Neris	640821	64015	160129	16080
Rbot	220138	21941	54952	5568
Virut	258781	25846	64659	6498
Menti	22586	2274	5664	551
Murlo	8841	891	2219	215

(b) ToN-IoT.

Attack	Training		Testing	
	(#Ben)	(#Mal)	(#Ben)	(#Mal)
Backdoor	105235	10513	26297	2640
DDoS	105219	10529	26313	2624
DoS	105182	10566	26350	2587
Injection	105303	10445	26229	2708
Password	105252	10496	26280	2657
Ransomware	105233	10515	26299	2638
Scanning	105157	10591	26375	2562
XSS	105222	10526	26310	2627

implementation, which leverages *DGL*[1] and *PyTorch Geometric*[2] libraries to provide reliable graph processing and model training. To ensure the reproducibility of our results, the source code of the following experiments is freely available at https://github.com/dimgalli/evaluating-xai.git.

4.1. Datasets

We base the case study on two publicly available datasets: *CTU-13* [38] and *ToN-IoT* [39]. These two datasets consist of labeled netflows and are widely referenced in the literature, making them reliable benchmarks for this study [35]. CTU-13 contains network traces that combine benign and malicious traffic from several real-world botnet variants. These botnets exhibit specific structural behaviors that align well with the capabilities of GNN-based NIDS. ToN-IoT contains heterogeneous types of IoT data. We consider the benign flow records and the malicious netflows of the cyber threats. The attacks in the two datasets exhibit complex malicious patterns, relying on different multi-flow strategies. Therefore, they are excellent candidates for evaluating GNN explainability methods.

We apply the same preprocessing steps described in [40], which include discarding non-TCP traffic and filtering out outliers. We also eliminate explicit IP addresses and port numbers to avoid separating flows by spurious correlations [41]. The final feature set includes more than 30 attributes and is consistent with previous work [18]. In particular, we build separate collections for each cyber threat in the two datasets, excluding the sets with insufficient malicious netflows that would yield underperforming detectors. We combine the benign data and the malicious samples for each specific threat in a benign:malicious ratio of 10:1. This distribution ratio reflects real-world scenarios [41]. Each collection is then divided into training and test subsets using an 80:20 ratio. We report the resulting number of benign and malicious flows of the two datasets used to train and test the models in Table 1, where each row indicates the botnet variant and attack type considered in the experiments.

4.2. Detectors

Our evaluation leverages GNN's unique ability to capture relational dependencies between netflows that classical ML techniques fail to achieve. This feature makes GNN models strong candidates for evaluating explanations since the perturbation of key graph components affects detection performance [42]. To guarantee a fair and meaningful comparison between different XAI methods, we focus on GNN that perform node-level classification. Indeed, these models generate more informative embeddings and exhibit greater robustness to adversarial perturbations [43].

Our GNN-based NIDS is built on the *GraphSAGE* model [44], which aggregates features to generate node embeddings. In the context of this paper, it is referred to as *LineGraphSAGE* [45] because it allows node classification directly on line graphs. As discussed in Section 2, flow graphs can be converted to line graphs using a linearization procedure. Attributes of the original netflows are transferred to the nodes

[1] https://docs.dgl.ai
[2] https://pytorch-geometric.readthedocs.io/

of the line graph, so GraphSAGE can be used to learn node embeddings without losing information from the netflow features. In addition, the inductive nature of this algorithm allows an effective embedding generation for nodes in unseen test graphs. Therefore, this approach makes GraphSAGE more suitable for network intrusion detection in the real world.

For each attack in the two datasets, we train a specific instance of the binary classifier to detect the malicious netflows (i.e., malicious nodes in the line graph), as suggested by prior research [46, 47]. We implement the detectors using the same hyperparameters outlined in [13].

4.3. Explainers

For our case study, we evaluate five different XAI algorithms tailored for GNN models. As discussed in Section 2, GNN-based NIDS learn attack patterns simultaneously from flow features and topological structures. However, it is unclear whether GNN relies specifically on feature-level information or graph structure when making its predictions. For this reason, we consider in the experiments two methods that highlight flow-level features [24, 25] and two strategies that provide explanations representative of the underlying topology [26, 27]. We recall that we use post-hoc XAI strategies to identify the structural vulnerabilities of the cyber detector, focusing on improving the overall security rather than the interpretability.

The XAI algorithms evaluated in the experiments are: *Dummy Explainer (DE)*, *Integrated Gradients (IG)* [24], *Saliency (SA)* [25], *GNNExplainer (GE)* [26], and *GraphMask (GM)* [27]. DE assigns random explanation scores as a baseline for comparison. IG calculates feature importance by integrating the gradients of model output relative to input along a baseline-to-input path. SA computes node importance by measuring gradients of model output with respect to input features. GE defines critical graph substructures by estimating the mutual information. GM generates masked versions of the graph by dropping edges and observing their effect on model predictions.

4.4. Framework Implementation

We now describe the implementation of our evaluation methodology, which is detailed in Section 3.

As mentioned in Section 4.1, each collection $\boldsymbol{D}_\alpha$ containing benign and malicious samples of a particular cyber threat α is divided into training and test subsets, preserving the 10:1 benign:malicious ratio typical of real-world settings [41]. For each collection obtained, we generate the flow graph G_α and then the line graph $L(G_\alpha)$ representation of the network traffic. Then, line graphs obtained from training sets are used to train an ensemble of LineGraphSAGE classifiers to detect specific malicious variants, as outlined in Section 4.2. Instead, line graphs obtained from test sets are used to evaluate detectors and explainers.

Each of the explainers presented in Section 4.3 is then applied to each instance of LineGraphSAGE to compute the corresponding explanations. Each explainer takes the clean line graph $L(G_\alpha)$ containing the network traffic of particular threat α and the corresponding model predictions $\hat{\boldsymbol{y}}$ to define its explanatory graph $L(G_\alpha^\star)$, where each node $v_i^\star$ has an importance score h_i indicating its influence in the detection process. On the one hand, gradient-based algorithms (i.e., IG and SA) assign a score to each attribute, so we calculate the average over the features to derive a single value for each node in the graph. On the other hand, perturbation-based methods (i.e., GE and GM) directly return a mask containing a score for each vertex. Since our strategy relies on $\mathbf{C2x}_{\mathscr{B}}$ structural attacks performed in the problem space, we tag each vertex $v_i^\star$ of the graph $L(G_\alpha^\star)$ with an identifier i to retrieve the original netflows in the dataset $\boldsymbol{D}_\alpha$. We then rank all flow records based on their importance score h_i. Next, we consider only the benign flows $\mathscr{B}$ and discard the malicious samples $\mathscr{M}$. To ensure a fair comparison between different attacks, we select the top 10% of legitimate examples $\mathscr{B}$ previously sorted by relevance h_i.

To evaluate the explainer's performance, we test the robustness of LineGraphSAGE against $\mathbf{C2x}_{\mathscr{B}}$ attacks [13]. In this scenario, attackers set up new benign communications from their compromised nodes to random destinations in the network. In other words, we inject new data samples into the test set, introducing new components into the resulting graph—edges in the flow graph and nodes in the

line graph. This perturbation is practically feasible because the generation of the graph representation occurs once a sufficient number of flows have been collected [20]. The original attacks involve sending β benign netflows $\mathscr{B}$ from each compromised host c_i to random targets u_i. Consequently, this process injects $\beta * |\mathbf{C}|$ new benign nodes into the resulting line graph, where $|\mathbf{C}|$ is the number of controlled hosts. However, instead of randomly sampling benign flows from the available pool, we select the relevant legitimate netflows to perturb the graph structure. This strategy enables us to evaluate the contribution of different explainers in refining the structural violations. To ensure that new communications originate from compromised nodes in the graph, we replace source IP addresses and port numbers included in the benign flows with those of the controlled hosts. We increase the number of legitimate samples injected into the test set by considering the following step values for β: 1, 2, 5, 10, and 20.

5. Results

We now present the results of our case study, where we apply the proposed evaluation strategy to a state-of-the-art GNN-based NIDS.

First, we evaluate LineGraphSAGE classifiers on the clean line graphs generated from the respective unperturbed validation sets to confirm the suitability of the considered GNN-based NIDS as a target for structural attacks. For this evaluation, we leverage measures commonly used in computer security [34], namely *F1-score*, *precision*, and *recall*. Considering a malicious network flow as a positive sample, these three evaluation metrics can be summarized as follows:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

where *TP*, *FP*, and *FN* are the number of true positives, false positives, and false negatives, respectively. These metrics range between 0 and 1, with higher scores indicating better performance. We recall that we do not rely on *accuracy* scores since the datasets employed in network intrusion detection are typically highly unbalanced.

We then evaluate explainers by observing how the benign flows they identify as important influence LineGraphSAGE performance once explanations are injected into the graph to perturb its overall structure. For this purpose, we rely on the *attack severity (AS)* measure [34]. This metric is defined as follows:

$$AS = 1 - \frac{Recall(\text{after the attack})}{Recall(\text{before the attack})}$$

where the numerator indicates the recall score on the perturbed graph, while the denominator is the recall value on the clean graph. This metric ranges between 0 and 1, with highly accurate attacks getting *AS* values closer to 1. Therefore, the most effective explainers are those achieving *AS* values close to 1, resulting in the most severe attacks.

5.1. Detectors Performance

Our evaluation campaign begins by testing the GNN-based NIDS presented in Section 4.2 on clean network traffic. In particular, we create a line graph representation for each test set and evaluate the specific instance of the GNN-based NIDS on it. The performance of the LineGraphSAGE classifiers on the two considered datasets is shown in Table 2. Each row refers to a threat and reports the detection results of the specific GNN classifier on it, with the last row summarizing the mean μ and the standard deviation σ values across the different instances.

The GNN-based NIDS achieves high performance across all evaluation metrics, with an average *F1-score* of 0.935 and 0.995 on CTU-13 and ToN-IoT datasets, respectively. From Table 2, we observe that our LineGraphSAGE models obtain scores that are in line with those of the state-of-the-art [13]. All classifiers exceed 0.9 in performance on the CTU-13 dataset, except on *Neris* traffic, where the GNN-based detector struggles due to the heterogeneity of malicious netflows, with a *F1-score* of 0.846 and a *recall* of 0.767. By contrast, each instance of LineGraphSAGE obtains near-perfect performance

Table 2
F1-score, *precision*, and *recall* values of the GNN classifiers on clean graphs.

(a) CTU-13.

Botnet	*F1-score*	*Precision*	*Recall*
Neris	0.846	0.944	0.767
Rbot	0.989	0.985	0.992
Virut	0.943	0.981	0.907
Menti	0.953	0.983	0.926
Murlo	0.946	0.934	0.958
μ	0.935	0.965	0.910
(σ)	(0.048)	(0.022)	(0.077)

(b) ToN-IoT.

Attack	*F1-score*	*Precision*	*Recall*
Backdoor	0.999	0.999	0.999
DDoS	0.995	0.995	0.995
DoS	0.994	1.000	0.989
Injection	0.991	0.996	0.986
Password	0.998	0.999	0.996
Ransomware	0.995	0.995	0.995
Scanning	0.994	0.993	0.995
XSS	0.995	0.993	0.997
μ	0.995	0.996	0.994
(σ)	(0.002)	(0.003)	(0.004)

scores on the ToN-IoT dataset, reflecting its reliability in detecting multi-flow cyber attacks with different structural topologies. The performance results prove that our GNN-based NIDS is effective in deployment scenarios that do not involve manipulated network traffic. Therefore, the considered detection model represents a solid target for the structural attacks through which we evaluate the different explanations.

5.2. Explainers Performance

We now evaluate the quality of the explanations extracted by the five XAI methods presented in Section 4.3. For each considered explainer, we generate an explanatory graph and inject its benign flow records into the corresponding test set, manipulating the resulting line graph with new nodes. As discussed in Section 4.4, we gradually increase the number β of key benign samples $\mathscr{B}^\star$ to estimate the severity of structural attacks at distinct perturbation levels. To assess the effectiveness of the explanation-based attacks, we feed the perturbed graphs into the GNN-based NIDS and measure how much the overall detection capability drops.

We compare the effectiveness of explainability methods in identifying important flow records and perturbing the resulting graph structure in Table 3, where each cell corresponds to the mean *AS* value obtained by the explainer in the column on the network traffic in the row. Each *AS* value is averaged over the perturbation steps β. For each row, we mark the best *AS* value in bold. The rows at the bottom in light gray summarize the *AS* values obtained by each explainer with the mean value μ computed over the different cyber attacks.

We immediately observe that AS values are not zero. The results are consistent with those obtained in the original work [13] and validate the vulnerability of GNN-based NIDS to structural attacks. The *IG* method achieves the highest *AS* values across all attacks, with an average value of 0.422 and 0.195 on CTU-13 and ToN-IoT datasets, respectively. This result shows that the features of flows are important for the detection model and support more effective perturbations. We observe that *SA* slightly outperforms *IG* over *Rbot* traffic, with a marginal difference of 0.004 in the mean values (0.237 for *SA* and 0.233 for *IG*). Similarly, *GraphMask* outperforms *IG* over *DDoS* and *DoS* attacks with average values of 0.273 and 0.023, compared to 0.187 and 0.018 of *IG*. The *AS* of *IG* measured on *Menti* and *Murlo* are the most significant, with average values equal to 0.728 and 0.900, respectively. This result demonstrates the effectiveness of explanations in perturbing graphs formed by a limited number of components. By contrast, the detector trained on *DoS* traffic is the most robust, with *AS* values not exceeding 0.1. Indeed, the limited number of controlled hosts reduces the attacker's potential threat. Our findings demonstrate that only attacks based on *IG* are effective than those that leverage random benign records. Hence, *IG*-generated explanations allow attackers to determine the critical features of network traffic and the structural vulnerabilities of the GNN-based NIDS.

Table 3
Attack severity (AS) values of the GNN classifiers on graphs perturbed with each explanation.

(a) CTU-13.

Botnet	*DE*	*IG*	*SA*	*GE*	*GM*
Neris	0.071	**0.104**	0.058	0.058	0.056
Rbot	0.181	0.233	**0.237**	0.169	0.194
Virut	0.101	**0.143**	0.103	0.059	0.103
Menti	0.457	**0.728**	0.538	0.402	0.529
Murlo	0.668	**0.900**	0.798	0.675	0.478
μ	0.296	**0.422**	0.347	0.273	0.272

(b) ToN-IoT.

Attack	*DE*	*IG*	*SA*	*GE*	*GM*
Backdoor	0.151	**0.203**	0.035	0.135	0.164
DDoS	0.108	0.187	0.189	0.102	**0.273**
DoS	0.012	0.018	0.011	0.012	**0.023**
Injection	0.145	**0.226**	0.075	0.123	0.208
Password	0.104	**0.223**	0.026	0.103	0.173
Ransomware	0.183	**0.247**	0.233	0.186	0.167
Scanning	0.114	**0.184**	0.097	0.112	0.101
XSS	0.190	**0.274**	0.187	0.177	0.222
μ	0.126	**0.195**	0.107	0.119	0.166

6. Conclusions

To address the black-box nature of GNN, XAI algorithms have been developed by practitioners and researchers. However, evaluating different XAI methods is challenging because traditional approaches often rely on expensive ground truth explanations or oversimplified metrics. In this paper, we present a novel framework for evaluating GNN explainers based on their ability to identify the most relevant graph components and improve the severity of structural attacks against GNN-based NIDS. Our experimental campaign, conducted on two public datasets, shows that the *IG* explainer consistently outperforms other methods in determining the most accurate explanations and guiding powerful attacks. More specifically, *IG* achieves the highest *AS* values in most cases, while *GraphMask* excels in *DDoS* and *DoS* scenarios due to the characteristic topological patterns associated with these threats. These results confirm the effectiveness of adversarial perturbations against GNN-based NIDS, especially when explanations generate structural attacks. Our proposal represents a significant contribution to the evaluation of XAI methods in GNN-based NIDS. Future research could expand the proposed work to other areas within cybersecurity, exploring new attack methods and defense strategies to improve the resilience of GNN-based NIDS.

Acknowledgments

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] R. Sommer, V. Paxson, Outside the closed world: On using machine learning for network intrusion detection, in: 2010 IEEE symposium on security and privacy, IEEE, 2010, pp. 305–316.
[2] F. Manganiello, M. Marchetti, M. Colajanni, Multistep attack detection and alert correlation in intrusion detection systems, in: Information Security and Assurance: International Conference, ISA 2011, Brno, Czech Republic, August 15-17, 2011. Proceedings, Springer, 2011, pp. 101–110.
[3] F. Pierazzi, S. Casolari, M. Colajanni, M. Marchetti, Exploratory security analytics for anomaly detection, computers & security 56 (2016) 28–49.
[4] W. Jiang, Graph-based deep learning for communication networks: A survey, Computer Communications 185 (2022) 40–54.

[5] D. Pujol-Perich, J. Suárez-Varela, A. Cabellos-Aparicio, P. Barlet-Ros, Unveiling the potential of graph neural networks for robust intrusion detection, ACM SIGMETRICS Performance Evaluation Review 49 (2022) 111–117.
[6] A. Warnecke, D. Arp, C. Wressnegger, K. Rieck, Evaluating explanation methods for deep learning in security, in: 2020 IEEE european symposium on security and privacy (EuroS&P), IEEE, 2020, pp. 158–174.
[7] N. Moustafa, N. Koroniotis, M. Keshk, A. Y. Zomaya, Z. Tari, Explainable intrusion detection for cyber defences in the internet of things: Opportunities and solutions, IEEE Communications Surveys & Tutorials 25 (2023) 1775–1807.
[8] A. Nadeem, D. Vos, C. Cao, L. Pajola, S. Dieck, R. Baumgartner, S. Verwer, Sok: Explainable machine learning for computer security applications, in: 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P), IEEE, 2023, pp. 221–240.
[9] G. Apruzzese, P. Laskov, A. Tastemirova, Sok: The impact of unlabelled data in cyberthreat detection, in: 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P), IEEE, 2022, pp. 20–42.
[10] H. Zhu, J. Lu, Graph-based intrusion detection system using general behavior learning, in: GLOBECOM 2022-2022 IEEE Global Communications Conference, IEEE, 2022, pp. 2621–2626.
[11] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, F. Roli, Evasion attacks against machine learning at test time, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13, Springer, 2013, pp. 387–402.
[12] L. Sun, Y. Dou, C. Yang, K. Zhang, J. Wang, S. Y. Philip, L. He, B. Li, Adversarial attack and defense on graph data: A survey, IEEE Transactions on Knowledge and Data Engineering 35 (2022) 7693–7711.
[13] A. Venturi, D. Stabili, M. Marchetti, Problem space structural adversarial attacks for network intrusion detection systems based on graph neural networks, arXiv preprint arXiv:2403.11830 (2024).
[14] G. Vormayr, J. Fabini, T. Zseby, Why are my flows different? a tutorial on flow exporters, IEEE Communications Surveys & Tutorials 22 (2020) 2064–2103.
[15] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, F. Ahmad, Network intrusion detection system: A systematic study of machine learning and deep learning approaches, Transactions on Emerging Telecommunications Technologies 32 (2021) e4150.
[16] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE transactions on neural networks 20 (2008) 61–80.
[17] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, 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.
[18] A. Venturi, M. Ferrari, M. Marchetti, 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.
[19] J. Zhou, Z. Xu, A. M. Rush, M. Yu, Automating botnet detection with graph neural networks, arXiv preprint arXiv:2003.06344 (2020).
[20] A. Venturi, D. Pellegrini, M. Andreolini, L. Ferretti, M. Marchetti, M. Colajanni, et al., Practical evaluation of graph neural networks in network intrusion detection, in: CEUR Workshop Proceedings, volume 3488, CEUR-WS, 2023.
[21] A. Longa, S. Azzolin, G. Santin, G. Cencetti, P. Liò, B. Lepri, A. Passerini, Explaining the explainers in graph neural networks: a comparative study, ACM Computing Surveys (2024).
[22] H. Yuan, H. Yu, S. Gui, S. Ji, Explainability in graph neural networks: A taxonomic survey, IEEE transactions on pattern analysis and machine intelligence 45 (2022) 5782–5799.
[23] J. Kakkad, J. Jannu, K. Sharma, C. Aggarwal, S. Medya, A survey on explainability of graph neural networks, arXiv preprint arXiv:2306.01958 (2023).
[24] M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks, in: International

conference on machine learning, PMLR, 2017, pp. 3319–3328.
[25] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, H. Hoffmann, Explainability methods for graph convolutional neural networks, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 10772–10781.
[26] Z. Ying, D. Bourgeois, J. You, M. Zitnik, J. Leskovec, Gnnexplainer: Generating explanations for graph neural networks, Advances in neural information processing systems 32 (2019).
[27] M. S. Schlichtkrull, N. De Cao, I. Titov, Interpreting graph neural networks for nlp with differentiable edge masking, arXiv preprint arXiv:2010.00577 (2020).
[28] B. Sanchez-Lengeling, J. Wei, B. Lee, E. Reif, P. Wang, W. Qian, K. McCloskey, L. Colwell, A. Wiltschko, Evaluating attribution for graph neural networks, Advances in neural information processing systems 33 (2020) 5898–5910.
[29] T. Funke, M. Khosla, M. Rathee, A. Anand, Zorro: Valid, sparse, and stable explanations in graph neural networks, IEEE Transactions on Knowledge and Data Engineering 35 (2022) 8687–8698.
[30] P. Li, Y. Yang, M. Pagnucco, Y. Song, Explainability in graph neural networks: An experimental survey, arXiv preprint arXiv:2203.09258 (2022).
[31] C. Agarwal, M. Zitnik, H. Lakkaraju, Probing gnn explainers: A rigorous theoretical and empirical analysis of gnn explanation methods, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2022, pp. 8969–8996.
[32] K. Amara, R. Ying, Z. Zhang, Z. Han, Y. Shan, U. Brandes, S. Schemm, C. Zhang, Graphframex: Towards systematic evaluation of explainability methods for graph neural networks, arXiv preprint arXiv:2206.09677 (2022).
[33] X. Zheng, F. Shirani, T. Wang, W. Cheng, Z. Chen, H. Chen, H. Wei, D. Luo, Towards robust fidelity for evaluating explainability of graph neural networks, arXiv preprint arXiv:2310.01820 (2023).
[34] G. Apruzzese, M. Colajanni, L. Ferretti, M. Marchetti, Addressing adversarial attacks against security systems based on machine learning, in: 2019 11th international conference on cyber conflict (CyCon), volume 900, IEEE, 2019, pp. 1–18.
[35] T. Bilot, N. El Madhoun, K. Al Agha, A. Zouaoui, Graph neural networks for intrusion detection: A survey, IEEE Access 11 (2023) 49114–49139.
[36] A. Venturi, D. Galli, D. Stabili, M. Marchetti, et al., Hardening machine learning based network intrusion detection systems with synthetic netflows, in: CEUR WORKSHOP PROCEEDINGS, volume 3731, CEUR-WS, 2024.
[37] A. Kuppa, N.-A. Le-Khac, Adversarial xai methods in cybersecurity, IEEE transactions on information forensics and security 16 (2021) 4924–4938.
[38] S. Garcia, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods, computers & security 45 (2014) 100–123.
[39] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, A. Anwar, Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems, Ieee Access 8 (2020) 165130–165150.
[40] A. Venturi, G. Apruzzese, M. Andreolini, M. Colajanni, M. Marchetti, Drelab-deep reinforcement learning adversarial botnet: A benchmark dataset for adversarial attacks against botnet intrusion detection systems, Data in Brief 34 (2021) 106631.
[41] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, K. Rieck, Dos and don'ts of machine learning in computer security, in: 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 3971–3988.
[42] D. Zügner, A. Akbarnejad, S. Günnemann, Adversarial attacks on neural networks for graph data, in: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 2847–2856.
[43] S. Zhang, H. Tong, J. Xu, R. Maciejewski, Graph convolutional networks: a comprehensive review, Computational Social Networks 6 (2019) 1–23.
[44] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems 30 (2017).
[45] X. Hu, W. Gao, G. Cheng, R. Li, Y. Zhou, H. Wu, Towards early and accurate network intrusion

detection using graph embedding, IEEE Transactions on Information Forensics and Security (2023).
[46] M. Stevanovic, J. M. Pedersen, An analysis of network traffic classification for botnet detection, in: 2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), IEEE, 2015, pp. 1–8.
[47] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, M. Marchetti, On the effectiveness of machine and deep learning for cyber security, in: 2018 10th international conference on cyber Conflict (CyCon), IEEE, 2018, pp. 371–390.