Using Graph Neural Networks for the Detection and Explanation of Network Intrusions

Ahmed Rafik El-Mehdi Baahmed[1], Giuseppina Andresini[2], Céline Robardet[3], and Annalisa Appice[2]

[1] Higher School in Computer Science 08 May 1945 of Sidi Bel Abbes, Algeria
[2] Department of Computer Science, University of Bari Aldo Moro, Italy
[3] Univ Lyon, INSA Lyon, LIRIS, UMR5205, F-69621 Villeurbanne, France

Abstract. The ever-increasing evolution of Deep Learning methods has enabled their use in many areas, including cybersecurity. With the exponential growth in the volume of data circulating in computer networks, their security is a paramount necessity. Nowadays, network security is mainly pursued using preventive techniques but also by detecting intrusions as soon as possible, when they occur. Different types of Machine Learning and Deep Leaning models have been recently studied for network intrusion detection, but surprisingly, although Network Intrusion Detection Systems (NIDSs) scrutinize flow data exchanges on a network, graph-based models have been little explored so far. We propose in this article to consider the relevance of Graph Neural Network (GNNs) to detect intrusions and also to explain them. For this purpose, we adapt the GNNExplainer method, that is, the pioneer method for explaining GNN decisions, to edge-level classification models.

Keywords: Network Intrusion Detection Systems, Graph Neural Networks, Explainable AI, Cybersecurity.

1 Introduction

Nowadays, a lot of sensitive data circulates in telecommunications networks and must be protected. This is the role of cybersecurity, which seeks to develop effective safeguard mechanisms, due to the evolution of malicious software and cyber-attacks. These protections are both preventive – built on knowledge of how attacks work – and reactive – linked to early detection of attacks when they occur. These two aspects are essential: the detection allowing us to react quickly in case of intrusions and the understanding making it possible to set up parades upstream in order to avoid types of attacks already encountered.

Early intrusion detection can be performed by Machine Learning (ML) methods. Different types of ML methods have been considered so far, among which Deep Learning (DL) models have been recognized as successful approaches for Network Intrusion Detection (NID). Graph Neural Network methods (GNNs) are an emerging subfield of Deep Learning, which works on graph-based data such as network topology in NIDs. Surprisingly, while NID systems analyse network flow data, graph-based ML models have been little explored so far in this field. A first step in this direction was performed in [15] where a GraphSAGE-based model was proposed for NID problem. In this article, we further evaluate this model by considering another NID benchmark dataset, CICIDS2017 [21], and compare it with other well-established ML models. We further investigate the added value of the graph structure for network intrusion detection. Knowledge of how attacks work can be increased by analyzing ML models. Indeed, if they succeed to detect attacks it is because they have been able to identify mechanisms underlying the attacks. EXplainable Artificial Intelligence (XAI) methods provide insight into how a ML model works, and in the best case, allow the user to understand the model well enough to be able to manually apply it to new data [8]. Different methods have been proposed to explain GNNs. Most of them are instance-level methods that aim to provide input-dependent explanations by identifying important input features on which the model builds its prediction. The gradient/feature-based methods [3] use the gradients or hidden feature map values to compute the importance of the input features. Perturbation-based methods [16, 25] learn a graph mask by studying the prediction changes when perturbing the input graphs. GNNExplainer [25] learns a soft mask by maximizing the mutual information between the original prediction and the predictions of the perturbed graphs. Other techniques, such as PGExplainer [16], PGM-Explainer [23] or GraphSVX [7], use generative approaches to learn surrogate models. These surrogate models can be misleading because the user tends to generalize beyond its neighborhood an explanation related to a local model. To explain the GraphSAGE-based model designed for NID problems formulated by handling attributes on edges and predicting the attack type associated with edges, we adapt the GNNExplainer method. This is the pioneering method for explaining GNNs, to edge-level classification models.

2 Related Work

DL has been acknowledged as an effective approach to NID. Several state-of-the-art approaches have proved the superiority of DL algorithms to recognize attacks [2, 9, 11]. Although DL has been conceded as an effective paradigm for NID, it suffers from some limitations. In particular, the black-box nature of DL models makes their decision challenging to understand. To tackle this issue, several algorithms coupled with XAI have been developed in the last years also in the cybersecurity domain. Several studies have explored post-hoc XAI techniques in NID, such as training a surrogate decision tree model [4] or coupling a surrogate model with a Deep Neural Network [22]. Caforio et al. [5] apply a Grad-CAM XAI technique in order to generate the gradient-based visual explanations for CNN binary classification in intrusion detection. Specifically, Grad-CAM are used to both improve CNN decisions (through a Grad-CAM-based nearest-neighbor classification) and increase the transparency of the CNN black-box decisions. In [19], SHAP (Shapley Additive Explanations) is used to identify the input features that most contributed to binary decisions made by a neural network on NID. SHAP is also employed in [24] to analyze the most relevant features for detecting each category of intrusion. In the study in [1], the analysis of feature relevance is conducted using DALEX to explain how a deep neural model evolves over the stream of network flow in CICIDS2017 dataset to adapt to new attack categories. In addition, recent state-of-the-art works have explored the use of intrinsic XAI techniques (i.e., attention mechanisms) incorporated into DL architectures, instead of post-hoc explanation. In [27], a Temporal Convolutional Network with an attention mechanism is trained for multi-class classification. The work proposed in [2] implements a Convolutional Neural Network with an attention mechanism to enhance both the accuracy and the explanation of the multi-class classification of network traffic data.

In recent years, GNNs have emerged as a sub-field of neural networks applied to graph representation of data. Several studies have explored the use of GNNs in different domains, including cybersecurity applications. The work in [15] proposes a GNN approach for NID for edge classification. The work uses a graph topology of network flows to capture both edge features and topological patterns for IoT attack flow detection. Also the work in [28] adopts a GNN for botnet node detection. The work in [14] implements a Graph Convolutional Network for anomaly and threats detection in network environments to identify both DDos and TOR-nonTOR datasets. Finally, the work in [6] proposes a GNN-based model applied to social networks to detect anomalies (i.e., fraudulent activities and spam). This work uses also some statistical graph properties (e.g., Betweenness centrality, Degree centrality and Closeness centrality) to identify the properties of suspicious nodes.

3 From NID dataset to graph

The main goal of a NID system is to discover any unauthorized access to a computer network. In general, NIDSs rely on traditional (non-graph-based) ML algorithms. Therefore, they neglect topological information naturally available within network flow data. Instead, our approach uses a graph-based representation of network flows, which takes advantage of both the feature representation of network traffic data and the network topological information extracted from flows (i.e., each network flow has a source ip and a destination ip). Our idea is that a graph-based representation of network traffic data can help to disclose and explain potential intrusion patterns. To evaluate the potential of GNNs to detect intrusions and to explain them, we consider a dataset that can be modeled by a graph: CICIDS2017 [21]. It covers a diverse set of attack scenarios created using six attack profiles (Brute Force, Heartbleed, Botnet, DoS, DDoS, Web Attacks and Infiltration) and benign behavior, all created using the B-Profile system [20]. The dataset consists of network traffic analysis results using the CICFlowMeter with labeled flows during five consecutive days (from Monday to Friday) and it is split into 8 CSV datafiles. The files and their attacks are presented in Table 1.

Table 1. CICIDS2017 attack types

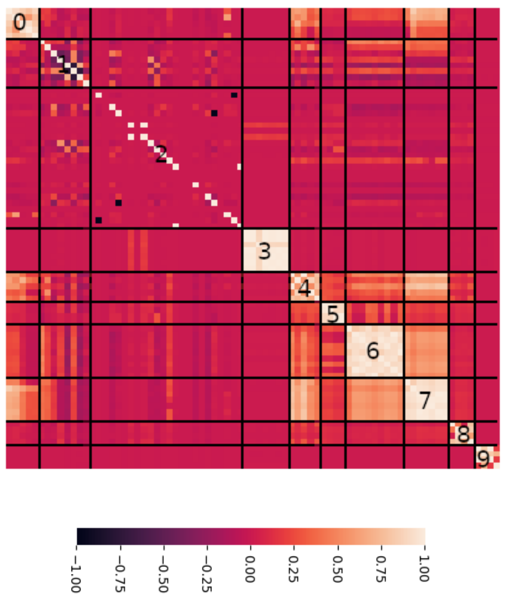
Labels	**Data files**								**All files**	
	Monday	Tuesday	Wednesday	Thursday (WebAttacks)	Thursday (Infiltration)	Friday (Morning)	Friday (PortScan)	Friday (DDos)	#	%
Benign	100%	96.90%	63.52%	98.72%	99.99%	98.97 %	44.52%	43.29%	2273097	80.30%
Bot						1.03%			1966	0.069%
DDoS								56.71%	128027	4.52%
G-Eye			1.48%						10293	0.36%
Dos-H			33.35%						231073	8.16 %
HTTP			0.79%						5499	0.19 %
S-Loris			0.83%						5796	0.20%
FTP-P		1.78%							7938	0.28%
H-Bleed			$\approx 0\%$						11	0%
Infilt.					0.01%				36	0.001%
P-Scan							55.48%		158930	5.61%
SSH-P		1.32%							5897	0.208%
B-Force				0.89%					1507	0.053%
SQLi				0.01%					21	0%
XSS				0.38%					652	0.023%

The network traffic occurs between hosts identified by their IP address and the port number used. The concatenation of both values are used as nodes of the graph. Each edge is attributed with a vector of 76 attributes that describe the traffic between the two incident nodes. This result in a graph[4], that in total, considering the concatenation of the 8 files, is made of 445131 nodes and 2830743 edges. Edges are labeled with one out of 15 attack or non-attack (benign) types. The number of occurrences of the different labels is also given in Table 1. We can see that the labels are clearly imbalanced. The graph is made of 716 connected components whose relative sizes is illustrated in Fig. 1 (left). The largest component has a size of 442782 nodes. 500 connected components have sizes less than 3 vertices. The distribution of node degrees is shown in Fig. 1 (right). We note that a few nodes achieve a very high degree, whereas for the vast majority of nodes, the measured degree is low. Categorical features have been encoded using *target encoder* method that encodes the categories by the posterior probability of the class given the input was the considered category. Once all numeric, the features have been normalized with the StandardScaler method so that they all have a mean of 0 and a standard deviation of 1. As there are 76 features, we grouped them with KMeans methods in 10 clusters based on their correlation coefficients. The correlation matrix reordered by the clusters, as well as the feature clusters are shown in Table 2. In particular, we note, that cluster 6 groups together several features computed on packet size and packet length characteristics, while cluster 7 groups together features computed on Idle tile and flow IAT characteristics. All clusters, except cluster 1 and cluster 2, are of good quality, grouping together highly correlated features. These clusters are used later in Section 5 to explain ML models.

[4] It could potentially be a multi-graph, but in the fact there is no multiple edges.

Table 2. Clusters of features obtained with KMeans based on their correlation coefficient (left). Heatmap of the correlation matrix reordered by clusters (right)

0	Fwd IAT Min, Bwd IAT Min, Bwd IAT Mean, Flow IAT Mean, Fwd IAT Mean
1	Bwd Packet Length Min, Protocol, Min Packet Length, PSH Flag Count, ACK Flag Count, FIN Flag Count
2	ECE Flag Count, CWE Flag Count, Flow IAT Min, Fwd Avg Packets/Bulk, Fwd PSH Flags, Fwd Packets/s, Bwd Avg Bytes/Bulk, Bwd Packets/s, Fwd Avg Bulk Rate, Fwd Avg Bytes/Bulk, RST Flag Count, Idle Std, Init_Win_bytes_bwd, Fwd Packet Length Min, Init_Win_bytes_fwd, Subflow Fwd Bytes, Bwd Avg Bulk Rate, Total Length of Fwd Packets, Bwd PSH Flags, SYN Flag Count, URG Flag Count, Bwd URG Flags, Fwd URG Flags, Bwd Avg Packets/Bulk, Down/Up Ratio
3	Subflow Bwd Bytes, Subflow Fwd Packets, Total Length of Bwd Packets, Total Bwd Packets, act_data_pkt_fwd, Total Fwd Packets, Subflow Bwd Packets
4	Bwd IAT Total, Bwd IAT Std, Flow Duration, Bwd IAT Max, Fwd IAT Total
5	Avg Fwd Segment Size, Fwd Packet Length Mean, Fwd Packet Length Std, Fwd Packet Length Max
6	Average Packet Size, Avg Bwd Segment Size, Max Packet Length, Bwd Packet Length Std, Packet Length Mean, Packet Length Std, Bwd Packet Length Max, Bwd Packet Length Mean, Packet Length Variance
7	Idle Max, Fwd IAT Max, Idle Min, Idle Mean, Flow IAT Max, Flow IAT Std, Fwd IAT Std
8	Active Max, Active Min, Active Mean, Active Std
9	Bwd Header Length, Fwd Header Length.1, Min Seg Size Fwd, Fwd Header Length



4 Methods

We present here the model of GNN that we use to detect intrusions in networks, as well as the method that we use to explain this model.[5]

4.1 E-GraphSAGE model

GNNs are an adaptation of Neural Network technologies to construct graph embeddings in Euclidean space and use them for classification tasks. The basic operation consists to learn a mapping of graph nodes to a k-dimensional embedding space in such a way that similar nodes – those with similar features and similar neighborhoods – are closed to each other in the embedding space. To that end, it progressively aggregates node neighboring information at each layer of the GNN. Starting with a vector associated to each node, that represents the features associated to the node or be a vector full of ones, the GNN update the vector of a node by aggregating the vectors of its direct neighbors multiplied by a learnable weight. The seminal GCN method [12] has been supplanted by GraphSage method [10] as it overcomes two main limitations of the previous method: the fact that GCN requires the whole graph structure during learning and its computational cost that is high especially for hub nodes. GraphSage

[5] The source code is available at `https://github.com/EagleEye1107/E-GNNExplainer`

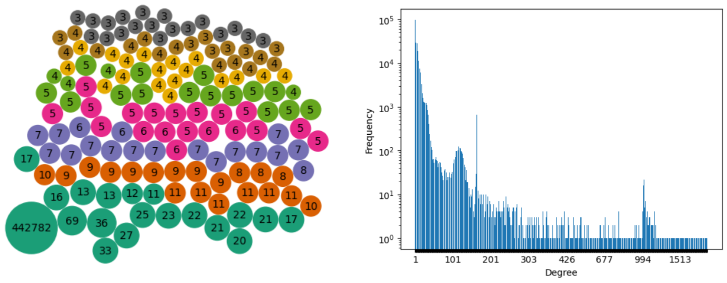


Fig. 1. Representation of the 150 largest connected components – log scale (left). Distribution of degree of nodes (right)

samples the set of neighbors during embedding computation. This is similar to mini-batching, a commonly used technique in machine learning. It consists in breaking down a dataset into smaller batches with the benefit of training models more effectively, that is to say it generally improves accuracy, increases speed and scalability. Neighbor sampling considers only a fixed number of neighbors picked up randomly. However, since creating a subgraph for each node is not efficient, GraphSage processes a set of nodes at a time by considering a subgraph shared by several nodes. GNNs are generally composed of several Convolutional layers whose results are transformed by a non-linear activation function.

Recently, a new method, E-GraphSAGE [15] has been proposed that extends GraphSage algorithm for edge classification when attributes are on edges. It samples and aggregates the edge information of the graph at each layer. Considering a graph $G = (V, E)$ and $\mathbf{e}_{u,v}$ the feature vector associated to edge $uv \in E$, E-GraphSAGE computes node embedding $\mathbf{h}_v^k$ for each node $v \in V$ and each GNN layer k. Considering $\mathbf{h}_v^0 = (1, \cdots, 1)$, it computes $\mathbf{h}_v^k$ for $k > 0$ by first evaluating $\mathbf{h}_{\mathcal{N}(v)}^k = AGG\big(\{\mathbf{e}_{uv},\ u \in \mathcal{N}(v)\}\big)$ where $\mathcal{N}(v)$ is a sample of the neighbors of v and AGG is an aggregation function such as mean, and then concatenating the previous vector with the $\mathbf{h}_v^{k-1}$: $\mathbf{h}_v^k = \sigma\Big(W_k \cdot CONCAT\big(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k\big)\Big)$ with W_k the learnable weights and σ a non linear activation function such as ReLU. $\mathbf{h}_v^K$ is the embedding of node v at the last layer and E-GraphSAGE outputs for each vertex $uv \in E$, $CONCAT\Big(\mathbf{h}_u^k, \mathbf{h}_v^k\Big)$. In the experiment section, we study the interest of E-GraphSAGE for network intrusion detection, both from the point of view of prediction performance and from the interpretation that can be drawn from the model.

4.2 GNNExplainer

From a powerful ML model for the detection of intrusion in networks, it is possible to increase the knowledge on the mechanisms of intrusion detection from the explanation of the model. If this model exploits the topological relations between the nodes, as GNNs do – relations by which the intrusions are propagated within the network – then, the knowledge which one draws from the model can be rich. We propose to use a XAI approach, based on perturbations, that consists in the creation of different input disturbances, whether on the graph structure, or on the graph features, to study their effect on the output. GNNExplainer [25] is the seminal perturbation-based method for GNNs and we adapt it to explain E-GraphSAGE. GNNExplainer generates two soft masks of continuous values in [0, 1], one called $feature\_mask$, whose values are weights directly applied on node features using an element-wise (Hadamard) product, and another one, called *edge_mask*, which, after be transformed by the sigmoid function to get values in $\{0, 1\}$, is used to extract the most important edges using the same product. GNNExplainer is a learning-based explanation method [26], that is to say, masks are generated thanks to a learning approach that aims to maximize the mutual information between the original prediction and the one obtained with the simplified graph as input. GNNExplainer represents a powerful approach to explain GNNs. However, the method is not adapted to edge-classification GNNs. We adapt GNNExplainer to the edge classification task where the input features and the target are on the edges and not on the nodes. It learns and applies a $feature\_mask$ on the edge features by computing significant importance weights for each feature of each data flows represented by the edges of the original graph, and an *edge_mask* on the edges. However, the two soft masks can affect each other since we are targeting the same data. To avoid affecting the $feature\_mask$ by the *edge_mask*, we separate the two explanations. Algorithm 1 sketches the two methods that work similarly. To explain the edge (u, v), explain_edge (resp. explain_features) first extracts the K-hop neighborhood from u (line 1) where K is number of layers of E-GraphSAGE. Then E-GraphSAGE is called (line 2) and the mask is initialized (line 3). In the following loop, the mask generator is learnt. First the mask is applied on the subgraph using Hadamard product (line 5), and the GNN model is called (line 6). After that, the mutual information between the prediction of the original subgraph and the simplified subgraph is used to update the mask generator (line 7) by back-propagating the error on the inner weights of the generator.

5 Experiments

5.1 E-GraphSAGE model evaluation

We evaluate the effectiveness of GNNs trained to detect network intrusions. The main questions that we aim to answer are: *Is E-GraphSAGE able to detect intrusions?* and *Does the graph structure improve the prediction accuracy?* To evaluate the accuracy performance of E-GraphSAGE, we compare its performance results with the classical classification methods: Random Forest and XGBoost [18]. In all the experiments, we used 70% of the data to train the models and we tested on the 30% remaining data. As CICIDS2017 is imbalanced (see Table 1), we evaluated the ML models using the accuracy score, but more importantly the F1 score, which balances between precision and recall by computing their harmonic mean.

Algorithm 1: E-GNNExplainer

```
  Input: GNN model E-GraphSAGE, edge to explain $(u, v)$, number of epochs
         $nb\_epochs$, depth $K$, Graph $G = (V, E)$
  Output: Mask on edges or features: $mask$
  /* $K$-hop subgraph                                                    */
1 $sub\_G \leftarrow subgraph(G, u, K)$
2 $pred\_original \leftarrow E - GraphSAGE(sub\_G, (u, v))$
3 $mask \leftarrow init\_mask(sub\_G)$
4 for $epoch = 1$ to $nb\_epochs$ do
5     $sub\_G\_masked \leftarrow apply\_mask(sub\_G, mask)$
6     $pred \leftarrow E - GraphSAGE(sub\_G\_masked, (u, v))$
7     $mask \leftarrow update\_mask(mask, pred, pred\_original)$
8 Return $sub\_G\_masked$
```

Comparison with Random Forest and XGBoost. We learn the models in a binary setting, predicting all attacks versus Benign. We first trained and tested the models files by files (i.e., each file was divided in training set and testing set). Results are shown in Table 3. Monday file results are not reported as it only contains Benign instances. The results obtained by E-GraphSAGE are comparable

Table 3. Model accuracy performance measured on each file and across files (in the binary classification setting)

	E-GraphSAGE		RF		XGboost	
Data file	Acc	F1	Acc	F1	Acc	F1
Tuesday	99.42%	91.58%	99.98%	99.78%	**99.99%**	**99.86%**
Wednesday	**99.99%**	**99.99%**	99.90%	99.87%	99.96%	99.95%
Thursday (WebAttacks)	99.98%	99.24%	99.96%	98.44%	**99.98%**	**99.46%**
Thursday (Infiltration)	99.98%	64.70%	**99.99%**	**77.77%**	99.99%	45.45%
Friday (Morning)	**99.99%**	**99.74%**	99.92%	96.41%	99.97%	98.71%
Friday (PortScan)	99.80%	99.82%	**99.99%**	**99.99%**	99.98%	99.99%
Friday (DDos)	**99.99%**	**99.99%**	99.97%	99.98%	99.98%	99.98%
All files	99.54%	99.06%	99.84%	99.68%	**99.89%**	**99.78%**

to the ones attained by Random Forest and XGboost methods. When the attacks are not too rare, it even has better scores than competing methods. This is all the more remarkable since RF and XGboost are ensemble methods, which is not the case with E-GraphSAGE that relies on a single model. Due to the lack of attack instances in the Tuesday, Thursday (both files) and Friday Morning data files (as shown in Table 1, with less than 3% of attack instances in each file), E-GraphSAGE obtains its lower F1 scores in the experiments performed on these days.

Table 4. Comparison between E-GraphSAGE and E-GSAGE-Abl on each file and across files (in the binary classification setting)

	E-GraphSAGE		E-GSAGE-Abl	
Data file	Acc	F1	Acc	F1
Tuesday	99.42%	91.58%	96.27%	62.21%
Wednesday	99.99%	99.99%	98.78%	98.35%
Thursday (WebAttacks)	99.98%	99.24%	98.34%	59.65%
Thursday (Infilteration)	99.98%	64.70%	99.91%	20.83%
Friday (Morning)	99.99%	99.74%	97.47%	44.05%
Friday (PortScan)	99.80%	99.82%	99.97%	99.97%
Friday (DDos)	99.99%	99.99%	99.90%	99.91%
All files	99.54%	99.06%	98.33%	96.63%

Ablation study. To evaluate the impact of the network structure on model E-GraphSAGE, we perform an ablation study using E-GSAGE-Abl. In this model, the graph structure is modified such that the edges of the graph are not connected to each other. This is achieved by renumbering the nodes in such a way that any two different edges have different incident nodes. Essentially, this means that there are no common nodes between different edges in the modified graph structure. By considering this modification, we can evaluate the impact of the graph structure on the performance of the E-GraphSAGE model. Specifically, we can analyze how the absence of interconnectivity between edges affects the model's ability to capture and propagate information through the graph. E-GSAGE-Abl model provides a way to assess the importance of graph structure and interconnected edges in E-GraphSAGE performance. By comparing its performance with that of E-GraphSAGE model, we can gain insights into the impact of graph connectivity on the model's ability to learn and generalize from graph-structured data. Results are presented in Table 4. On Wednesday, Friday (DDos) and Friday (PortScan) data, the detection is clearly done on the basis of the features. In this case, E-GSAGE-Abl shows high results despite ablation. On the other hand, on these same datasets, the results of E-GraphSAGE are equivalent to those obtained by RF and XGboost. Results on Friday (Morning) are particularly interesting as, on this dataset, E-GraphSAGE outperforms clearly RF and XGboost, while E-GSAGE-Abl's performance drops significantly. Last line in Table 4 shows the accuracy performance results obtained when considering all the files at once. We can observe that E-GSAGE-Abl perform worst than E-GraphSAGE, showing that the graph structure brings information to the E-GraphSAGE and increases the quality of its predictions.

5.2 Explaining intrusion detection models

Our goal is to study the impact of the graph structure and the graph information on the performance of NIDSs, by focusing our study on understanding which graph substructures and features are used by the E-GraphSAGE approach to predict attacks. To that end, we use E-GNNExplainer. E-GNNExplainer can be applied for both Local and Global explanations, but, in this work, we focus on Local explanations only, to study in details the behavior of E-GraphSAGE. The two main questions we aim to answer are: *What is the most influential subgraph impacting the prediction of an edge of a given type?* and *What are the edge features in this subgraph that have the most impact on the prediction of an edge of a given type?* To answer these questions, we focus on the impact of the graph structure and the message passing aspect of the E-GraphSAGE on its predictions. For that, we consider the edges whose prediction change when the graph structure is not taken into account. That is to say, we select the wrongly predicted edges by E-GSAGE-Abl that are correctly predicted by E-GraphSAGE. These edges are the source of the 3% difference in the F1 score between the two models observed in Table 4. This allows us to examine in depth the importance of the structure of the graph on the predictions. This set of edges is denoted $Impact\_edgse$ in the following.

Graph structure analysis. We analyse the graph structure that impacts predictions of E-GraphSAGE. For each attack class, we applied E-GNNExplainer on the edges of this type belonging to $Impact\_edges$ and kept the edge for which the mask produced by E-GNNExplainer has the maximum mutual information value. To analyse the subgraph found by E-GNNExplainer, we consider in Fig 2 the distribution of the edge types within the subgraph found for each given class. We note that Benign, Infiltration and Bot edges dominate subgraphs found for these three classes, respectively. Benign edges are largely present in subgraphs of several attacks (e.g., Heartbleed, DDos, DoS, SQL Injection, Port Scans and SSH-Patator). This is not surprising due to the imbalanced condition of this dataset. Finally, we note that two sub-categories of DoS attack edges (i.e., DoS Hulk and DoS Slowhttptest) and PortScan edges appear simultaneously in the intrusion subgraphs of several attack categories. This suggests that the presence of these edges into a subgraph can be considered as a relevant sign of malicious network traffic, although these edges may not provide sufficient information to identify the exact category of the malicious behaviour. We can notice that Web Attacks (SQL Injection, XSS, Brute Force, DDoS, DoS Hulk and PortScan) have similar attack distribution in their neighborhood showing many Benign, Dos Hulk, DDOs and PortScan edges. These different attacks are the most frequent ones in the data and it is for this reason that they stand out. Bot, Heartbleed and Infiltration attacks, on the other hand, show the majority of the edges of the same type in their vicinity. Indeed, these attacks are known to propagate through the network, and rely on edges that are similar to themselves, as correctly identified by E-GNNExplainer. To go into more details in the study of the most influential subgraphs for E-GraphSAGE predictions, we visualize the subgraphs provided by E-GNNExplainer for the two edges that maximize the mutual information. The first subgraph corresponds to the explanation of a Bot edge and is shown in Fig. 3. The Bot attack edge is colored in red on the graph. On this subgraph, we note that the Bot attack is performed from node 56 to

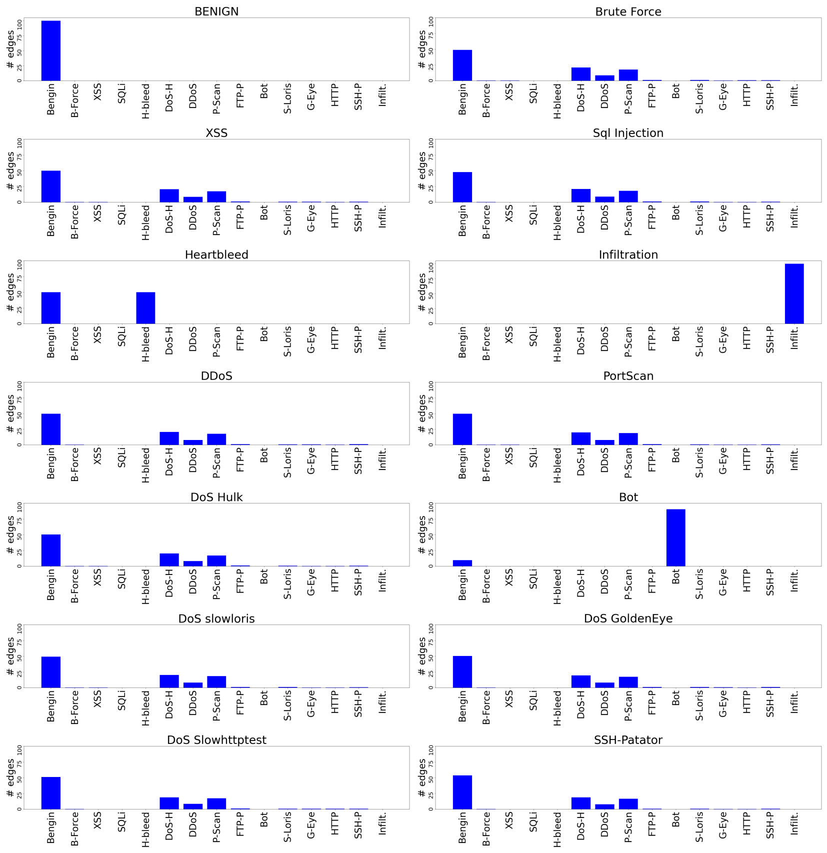


Fig. 2. Distribution of edges according to their type within the edge explanation maximizing mutual information for each attack class

node 47. We can notice the dependency of the explained attack with the other attacks from the same type, performed by the same source node. Interestingly, this dependency includes the edge $(40, 56)$, also associated to a Bot attack targeting the node 56. We recall that Botnet attacks are performed with the goal to create a network of devices controlled by attackers remotely. Then, the infected devices are used to carry out attacks. In fact, the graph dependency reported in Fig. 3 highlights the propagation of Botnet attack from node 40 to node 47, via node 56. Fig. 4 shows the most influential subgraph associated to the prediction of a Benign edge (in red on the subgraph). One can notice the presence of only benign edges within the subgraph.

Edge feature analysis. To understand in depth the contribution of the edge features on the predictions of E-GraphSAGE, we used E-GNNExplainer to ob-

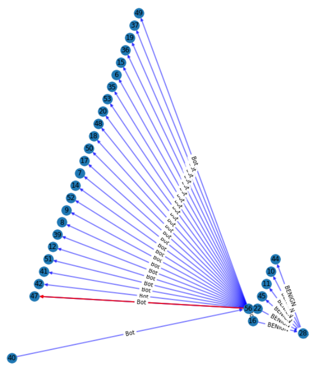


Fig. 3. The most influential subgraph obtained by GNNExplainer to explain Bot attack edge (56,47)

tain the feature mask that represents the feature importance, while explaining the edges from Fig. 2. To analyze the masks, we used the clustering presented in Table 2 to group the 76 features into 10 clusters. Fig 5 presents the number of occurrences in each cluster of the 10 most important features. The plot suggests that the features in cluster 9 are involved in all attacks except Bot and Benign. We point out that cluster 9 contains features as *Fwd Header Lenght1.* and *Min Seg Size Fwd* that are recognized as important to distinguish attacks from benign also in [29]. In addition, we can observe that cluster 3, involving features counting the number of packets in both backward and forward directions, achieves a high frequency in all classes, except Bot attacks. On the other hand, features in cluster 1 are recognised as the most important by GNNExplainer to identify Bot attacks. The cluster contains features counting the number of flags sent by a host as *PSH Flag Count*, *ACK Flag Count* and *FIN Flag Count*. In normal traffic flow, the number of these flags is random, but in Bot traffic, a pattern makes the number of these flags exchanged in the network high and almost fixed [13]. In addition, cluster 1 contains the feature *Protocol*. We can also observe that *Protocol* is present in Table 5 (left) reporting the top-10 most important features for the prediction of the Bot attack edge (56,47) using GNNExplainer. This is expected since HTTP protocol is one of the more commonly used protocols to propagate Bot attacks [17]. We also note that the top-10 most important features for the prediction of the Benign edge (57,18) comprise *Bwd Packet Length Min*, *Packet Length Variance* and *Active Std* which are also relevant for recognizing the Bot attack edge (56,47). The remaining features are specific for the Benign profile.

6 Conclusion and future work

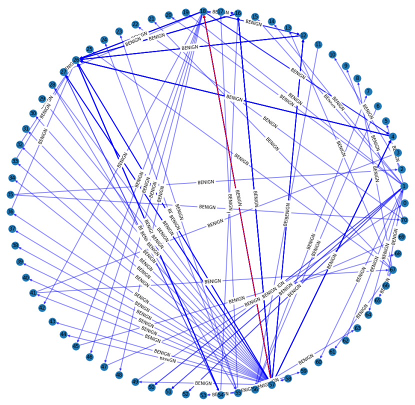


Fig. 4. The most influential subgraph obtained by GNNExplainer to explain Benign edge (57,18)

Table 5. The top-10 most important features found by GNNExplainer to explain the Bot attack edge (56,47) (left) the Benign edge (57,18) (right) classified by E-GraphSAGE model

Rank	Features	Importance weight
1	Protocol	0.969751
2	Fwd Packet Length Min	0.963157
3	Max Packet Length	0.960226
4	Bwd Packet Length Min	0.954250
5	Min Packet Length Std	0.953229
6	PSH Flag Count	0.944505
7	ECE Flag Count	0.935783
8	RST Flag Count Std	0.909857
9	Active Std	0.904808
10	Packet Length Variance	0.878375

Rank	Features	Importance weight
1	Subflow Bwd Bytes	0.576703
2	Bwd Packet Length Min	0.562913
3	Idle Mean	0.548867
4	Avg Bwd Segment Size	0.541600
5	Fwd Packet Length Max	0.535543
6	Packet Length Variance	0.534765
7	Active Std	0.533494
8	Bwd Avg Bytes Bulk	0.531683
9	Subflow Bwd Packets	0.531668
10	Average Packet Size	0.531412

This article explores the effectiveness of GNNs trained for network intrusion detection problems. The analysis is performed by considering a benchmark cybersecurity dataset, namely CICIDS17, that contains benign network flows, as well as malicious network flows belonging to multiple intrusion categories. Specifically, CICIDS17 network flow data, that occur between hosts identified by the host IP and the used port, are represented as edges of a graph structure, so that the network intrusion detection problem can be formulated as an edge classification task. The experimental study, described in the article, explored the accuracy of the GNN model compared to that of traditional ML models neglecting the graph structure of data. In addition, the described experimental study illustrated an ablation study to show the gain in accuracy achieved accounting for the graph structure information training a GNN model. As a further contribution, in this article, we illustrate an extension of GNNExplainer formulated to explain GNN decisions on network flow connections at edge level. The GNNExplainer allows us to disclose useful insights into the network structure of attack

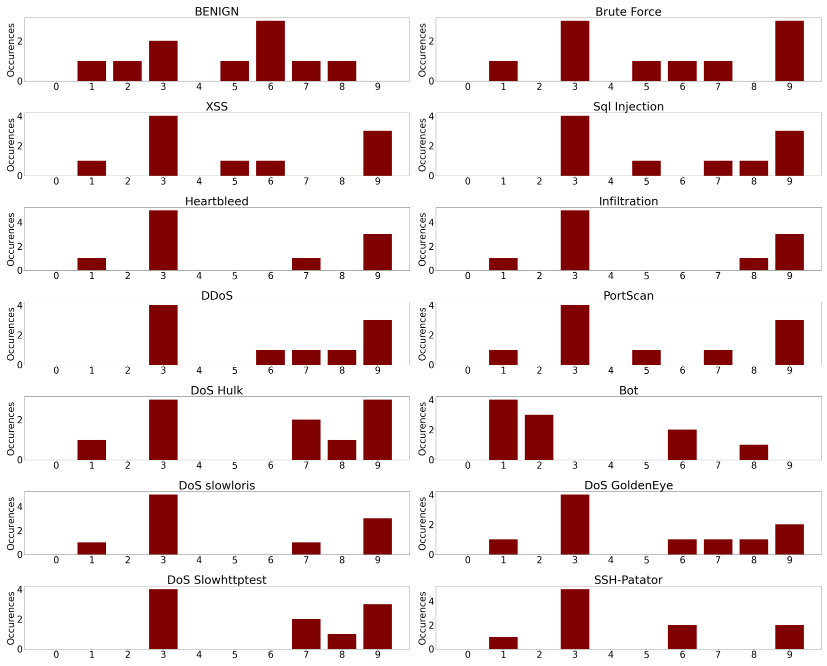


Fig. 5. Feature importance, grouped by KMeans clusters, obtained by GNNExplainer to explain attack edges

signature, as well as into the main characteristics of the intrusions. To the best of our knowledge this is one of the first studies exploring both the accuracy and explainability of GNN methods in cybersecurity problems. This study paves the way for future developments in this direction. One limitation of the proposed approach is that it neglects the streaming nature of the network traffic. Signatures of intrusions may change over time also due to adversarial attacks. As future works, we plan to explore how changes in both the graph structure explanations and the network traffic characteristic explanations can help in detecting concept drifts by triggering appropriate adaptations of the trained GNN model to the drifted data. In addition, we plan to explore possible graph representations of malware detection problems, in order to explore the advantages of both GNNs and GNNExplaner also in the field of malware detection.

References

1. Andresini, G., Pendlebury, F., Pierazzi, F., Loglisci, C., Appice, A., Cavallaro, L.: Insomnia: Towards concept-drift robustness in network intrusion detection. In: 14th ACM Workshop on AI and Security. p. 111–122. AISec'21 (2021)

2. Andresini, G., Appice, A., Caforio, F.P., Malerba, D., Vessio, G.: ROULETTE: A neural attention multi-output model for explainable network intrusion detection. Expert Syst. Appl. **201**, 117144 (2022)
3. Baldassarre, F., Azizpour, H.: Explainability for GCNs. arXiv:1905.13686 (2019)
4. Burkart, N., Franz, M., Huber, M.F.: Explanation framework for intrusion detection. In: Beyerer, J., Maier, A., Niggemann, O. (eds.) Machine Learning for Cyber Physical Systems. pp. 83–91. Springer Berlin Heidelberg, Berlin, Heidelberg (2021)
5. Caforio, F.P., Andresini, G., Vessio, G., Appice, A., Malerba, D.: Leveraging grad-cam to improve the accuracy of network intrusion detection systems. In: DS 2021, Proceedings. vol. 12986, pp. 385–400. Springer (2021)
6. Chaudhary, A., Mittal, H., Arora, A.: Anomaly detection using graph neural networks. In: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). pp. 346–350 (2019)
7. Duval, A., Malliaros, F.D.: Graphsvx: Shapley value explanations for graph neural networks. In: ECMLPKDD'21. pp. 302–318 (2021)
8. Fürnkranz, J., Kliegr, T., Paulheim, H.: On cognitive preferences and the plausibility of rule-based models. Machine Learning **109**(4), 853–898 (dec 2019)
9. Gao, M., Ma, L., Liu, H., Zhang, Z., Ning, Z., Xu, J.: Malicious network traffic detection based on deep neural networks and association analysis. Sensors **20**(5), 1452 (2020)
10. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in Neural Information Processing Systems pp. 1024–1034 (2017)
11. Kasongo, S., Sun, Y.: Performance analysis of intrusion detection systems using a feature selection method on the unsw-nb15 dataset. J Big Data **7**(105), 1–20 (2020)
12. Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. In: Proceedings of the 5th International Conference on Learning Representations. ICLR '17 (2017), `https://openreview.net/forum?id=SJU4ayYgl`
13. Kirubavathi Venkatesh, G., Anitha Nadarajan, R.: Http botnet detection using adaptive learning rate multilayer feed-forward neural network. In: Information Security Theory and Practice. Security, Privacy and Trust in Computing Systems and Ambient Intelligent Ecosystems. pp. 38–48 (2012)
14. Kisanga, P., Woungang, I., Traore, I., Carvalho, G.H.S.: Network anomaly detection using a graph neural network. In: 2023 International Conference on Computing, Networking and Communications (ICNC). pp. 61–65 (2023)
15. Lo, W.W., Layeghy, S., Sarhan, M., Gallagher, M., Portmann, M.: E-graphsage: A graph neural network based intrusion detection system for iot (2021)
16. Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., Zhang, X.: Parameterized explainer for GNN. In: NeurIPS 2020 (2020)
17. Owen, H., Zarrin, J., Pour, S.: A survey on botnets, issues, threats, methods, detection and prevention. Journal of Cybersecurity and Privacy **2**(1), 74–88 (2022)
18. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
19. Sarhan, M., Layeghy, S., Portmann, M.: An explainable machine learning-based network intrusion detection system for enabling generalisability in securing iot networks. CoRR **abs/2104.07183** (2021)
20. Sharafaldin, I., Gharib, A., Habibi Lashkari, A., Ghorbani, A.: Towards a reliable intrusion detection benchmark dataset. Software Networking **2017**, 177–200 (01 2017)

21. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: International Conference on Information Systems Security and Privacy (2018)
22. Szczepański, M., Choraś, M., Pawlicki, M., Kozik, R.: Achieving explainability of intrusion detection system by hybrid oracle-explainer approach. In: IJCNN 2020. pp. 1–8 (2020)
23. Vu, M.N., Thai, M.T.: PGM-Explainer: Probabilistic Graphical Model Explanations for Graph Neural Networks . In: NeurIPS 2020 (2020)
24. Wang, M., Zheng, K., Yang, Y., Wang, X.: An explainable machine learning framework for intrusion detection systems. IEEE Access **8**, 73127–73141 (2020)
25. Ying, R., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnnexplainer: Generating explanations for graph neural networks. Advances in neural information processing systems **32**, 9240–9251 (2019)
26. Yuan, H., Yu, H., Gui, S., Ji, S.: Explainability in graph neural networks: A taxonomic survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(5), 5782–5799 (2023)
27. Zhao, P., Fan, Z., Cao, Z., Li, X.: Intrusion detection model using temporal convolutional network blend into attention mechanism. International Journal of Information Security and Privacy **16**(1), 1–20 (2022)
28. Zhou, J., Xu, Z., Rush, A.M., Yu, M.: Automating botnet detection with graph neural networks. CoRR **abs/2003.06344** (2020)
29. Šarčević, A., Pintar, D., Vranić, M., Krajna, A.: Cybersecurity knowledge extraction using xai. Applied Sciences **12**(17) (2022)

Acknowledgment

The work of Ahmed Rafik El-Mehdi Baahmed has benefited from state aid managed by the National Research Agency under France 2030 bearing the reference "ANR-22-PEAE-0008". The work of Giuseppina Andresini is partially supported by the research project FAIR - Future AI Research (PE00000013), Spoke 6 - Symbiotic AI (CUP H97G22000210007) under the NRRP MUR program funded by the NextGenerationEU. The work of Annalisa Appice is partially supported by the research project SERICS (PE00000014) under the NRRP MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.