Unveiling the potential of Graph Neural Networks for robust Intrusion Detection

David Pujol-Perich, José Suárez-Varela, Albert Cabellos-Aparicio, Pere Barlet-Ros
Barcelona Neural Networking Center, Universitat Politècnica de Catalunya, Spain
contactus@bnn.upc.edu

ABSTRACT

The last few years have seen an increasing wave of attacks with serious economic and privacy damages, which evinces the need for accurate Network Intrusion Detection Systems (NIDS). Recent works propose the use of Machine Learning (ML) techniques for building such systems (e.g., decision trees, neural networks). However, existing ML-based NIDS are barely robust to common adversarial attacks, which limits their applicability to real networks. A fundamental problem of these solutions is that they treat and classify flows independently. In contrast, in this paper we argue the importance of focusing on the structural patterns of attacks, by capturing not only the individual flow features, but also the relations between different flows (e.g., the source/destination hosts they share). To this end, we use a graph representation that keeps flow records and their relationships, and propose a novel Graph Neural Network (GNN) model tailored to process and learn from such graph-structured information. In our evaluation, we first show that the proposed GNN model achieves state-of-the-art results in the well-known CIC-IDS2017 dataset. Moreover, we assess the robustness of our solution under two common adversarial attacks, that intentionally modify the packet size and inter-arrival times to avoid detection. The results show that our model is able to maintain the same level of accuracy as in previous experiments, while state-of-the-art ML techniques degrade up to 50% their accuracy (F1-score) under these attacks. This unprecedented level of robustness is mainly induced by the capability of our GNN model to learn flow patterns of attacks structured as graphs.

Keywords

Cybersecurity, Network Intrusion Detection, Machine Learning, Graph Neural Networks.

1. INTRODUCTION

Recent years have witnessed a great surge of malicious activities on the Internet, leading to major service disruptions and severe economic and privacy implications. For example, according to [1] the average cost of a data breach in 2020 was $3.86 million, while cyber attacks had an estimated cost to the U.S. economy between $57 billion and $109 billion only during 2016. These figures urge the need for the development of effective Network Intrusion Detection Systems (NIDS) that can detect – and thus prevent – future attacks.

In this context, a recent body of literature proposes the use of Machine Learning (ML) techniques as accurate methods to build NIDS [2, 3]. Indeed, existing solutions often show an accuracy above 98% when evaluated in popular IDS datasets. However, despite their good performance, ML techniques have not yet been widely adopted in commercial NIDS [4]. We argue that an important reason behind this lack of adoption is their insufficient robustness against traffic changes, adversarial attacks [5], and generalization over traffic of other networks, which are crucial factors to achieve practical ML-based solutions applicable to real networks in production.

A main limitation of existing solutions is that most of them treat and classify flows independently, by capturing meaningful flow-level features that correlate with different attacks. This assumption, however, does not properly adapt to numerous real-world attacks that rely on complex multi-flow strategies (e.g., DDoS, port scans). In this paper, we argue that, to effectively detect this type of attacks, it is essential to capture not only the individual features of flows, but also their relationships within the network. Thus, we propose a graph representation we call *host-connection graphs*, which structures flow relationships in a proper way to then capture meaningful information about the structural flow patterns of attacks (e.g., DDoS, port/network scans, brute force attacks). This is mainly supported by the fact that many common attacks can be unambiguously characterized by structural flow patterns that are fixed by the nature of the attack itself.

In this context, we propose a novel Graph Neural Network (GNN) model that uses a non-standard message-passing architecture, especially designed to process and learn from host-connection graphs. GNNs [6] are a novel neural network family that is particularly suitable for processing information inherently represented as graphs (e.g., chemistry, computer networks, physics). As a result, our model shows good capabilities over the graph-structured information within host-connection graphs. We provide an open-source implementation of this model at [7].

In the evaluation, we reveal the potential of the proposed GNN model to achieve robust NIDS solutions. First, we evaluate our model in the well-known CIC-IDS2017 dataset [8]. Our results show that the proposed model is able to accurately detect a wide variety of up-to-date attacks, achieving similar accuracy to state-of-the-art ML techniques widely used for NIDS (0.99 of weighted F1-score). Then, we test the robustness of our solution under different com-

Workshop on AI in Networks and Distributed Systems (WAIN) 2021
Milan,Italy

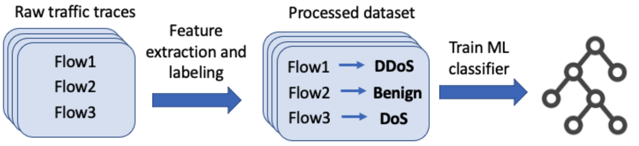

Figure 1: Scheme of traditional ML-based NIDS with flow-based operation.

mon adversarial attacks [5], which are focused on modifying specific flow features, such as the packet size and the inter-arrival times. The results show that our GNN-based NIDS is robust to this type of detection prevention methods commonly used by attackers. In contrast, the state-of-the-art ML benchmarks evaluated significantly decrease their performance, observing degradations of accuracy (weighted F1-score) up to 50% in our experiments. These results suggest that the proposed GNN model is able to capture meaningful patterns from flow relationships, that are more robust to the adversarial attacks analyzed in this paper.

2. WHY GRAPH-BASED NIDS?

Traditionally, ML-based NIDS leverage supervised-learning algorithms, such as Decision Trees, Random Forest, or Support Vector Machines (SVM) to classify the traffic. To train such systems (see Figure 1), first individual flow records are typically built from traffic captures, including some features that can be relevant to then classify flows (e.g., packet lengths, inter-arrival times, duration). Afterward, each flow record is labeled according to the attack it represents. Then, a ML model is trained to classify flows individually, based on the information contained in their records.

While this type of models often achieve good accuracy when trained and evaluated with traffic of the same network, they are especially vulnerable to adversarial attacks, which often vary flow features along time to avoid detection. This limitation becomes particularly evident from the optic of multi-flow attacks, where it is typically needed to analyze and relate a set of flows before detecting the malicious action (e.g., port scans, network scans, DDoS, brute force attacks).

Instead, we argue the importance of capturing and modeling the inter-dependencies between different flows traversing the network, which can be naturally represented in the form of graphs. As an example, Figure 2 shows graph representations of common multi-flow attacks. As we can observe, these attacks present inherent flow patterns that make them easily identifiable. For instance, DDoS attacks are distributed by definition, which means that we can expect a massive number of connections f_x from different hosts a_x to the same target v. Another classic example are port scans, which involve numerous connections f_x from the same host a to different ports of a same destination host v. Or network scans, which often involve multiple connections f_x to hosts of the same network v_x from a single source a. In these cases, inspecting flows individually – as most traditional ML-based NIDS do (Fig. 1) – reasonably hinders the possibility to discriminate such attacks from benign traffic. In practice, some traditional ML-based NIDS have shown high accuracy levels on these attacks, however this can be arguably explained by a high degree of over-fitting on the training and validation datasets, as ML models can eventually learn specific flow-level features (e.g., average packet size) that are highly correlated to some attacks. Nevertheless, this makes them strongly vulnerable to simple variations on malicious flows (e.g., packet lengths, inter-arrival times, ports), which is a common practice among attackers.

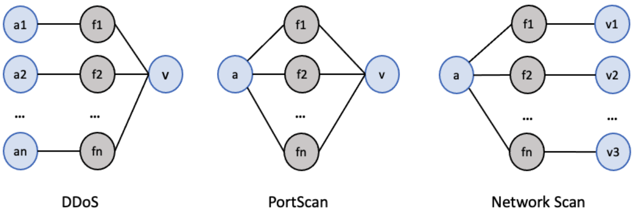

Figure 2: Graph-based representation of well-known attacks. a_x nodes refer to the attackers, v_x nodes represent the targets, and f_x nodes represent different flows.

In light of the above, we claim that learning the underlying structural flow patterns of attacks is essential to achieve a deeper knowledge and characterization of them, especially for attacks involving multiple flows. More importantly, representing flows and their relations as graphs – as those of Figure 2 – enables to capture more robust patterns against potential adversarial attacks, which typically keep the same flow structure, as it is fixed by the nature of the attack itself.

3. BACKGROUND ON GNN

Graph Neural Networks (GNN) [6] are a recent neural network family specifically designed to learn and generalize over graph-structured data, by capturing and modeling the inherent patterns in graphs. This has resulted in an unprecedented predictive power in many applications where data is structured as graphs [9]. This section describes the basic architecture of *Message-Passing Neural Network (MPNN)* [10], which represents a general framework covering most of the existing state-of-the-art GNN models [9].

MPNN operates over a graph $G=(V,E)$, where every node $v \in V$ is characterized with an initial set of features X_v, used to encode its initial hidden-state h_v^0 (which is represented as a n-element vector). The MPNN then proceeds with the message-passing phase, which is repeated a given number of iterations T. In each message-passing iteration t, every node v receives a *message* $m_{v,j}$ from each of its neighbors $j \in N(v)$. Particularly, messages are the result of combining the hidden states of connected nodes (h_v, h_j) with a message function $m(\cdot)$, which is typically approximated by a neural network and is uniformly applied over all node pairs in the graph. Then, all the messages received in a node are combined with an aggregation function $a(\cdot)$, producing a fixed-size output independently of the number of messages received (i.e., the number of nodes connected). This aggregation function is often implemented as an element-wise summation.

Lastly, each node updates its hidden state (h_v) based on the aggregated messages received from its neighbors, using an update function $u(\cdot)$ also approximated by a neural network.

Formally, the message passing at a given iteration t is defined as follows:

$$m_{v,j} = m(h_v^t,\ h_j^t,\ e_{v,j}) \quad (1)$$

$$M_v^{t+1} = a(\{m_{v,j} \mid j \in N(v)\}) \quad (2)$$

$$h_v^{t+1} = u(h_v^t,\ M_v^{t+1}) \quad (3)$$

Given the final hidden states obtained after T message-passing iterations, the GNN executes a readout phase. In this context, a subset of hidden-states – which depends on the specific GNN model – is passed through a learnable readout function $r(\cdot)$ that produces the output of the GNN model. Thus, $r(\cdot)$ is mainly intended to map the final nodes' hidden-state embeddings (h_v^T) to the output labels of the model ($\hat{y}$):

$$\hat{y} = r(h_v^T \mid v \in V) \tag{4}$$

As a result, the novel message-passing architecture of GNNs endows these models with an unprecedented generalization power over graphs of different size and structure. We refer interested readers to [9] for further details regarding GNNs.

4. PROPOSED GNN-BASED NIDS

This section describes the proposed GNN-based NIDS. We first present a host-connection graph we use to represent the traffic, which has enough expressiveness to represent flow patterns of attacks, such as those depicted in Figure 2. Then, we describe a novel GNN architecture tailored to operate over the previous graph. This new GNN model comprises a non-standard message-passing architecture that deals with the heterogeneous elements and the particularities of the network intrusion detection problem.

4.1 Host-Connection Graph Representation

Given a set of flows $\mathcal{F}$, we build a host-connection graph $G_{\mathcal{F}}$, that includes a node for each distinct host involved, either sending or receiving traffic. Moreover, each flow is represented as a node of this graph. Thus, given a flow $f \in \mathcal{F}$, with a source host S, and a destination host D, we create two undirected edges: one from the source host to the flow ($S \rightarrow f$), and another from the flow node to the destination host ($f \rightarrow D$).

This representation provides enough expressiveness to properly capture flow patterns of attacks. Particularly, the host-connection graph comprises relevant aspects of flows, with focus on their structural features. First, it enables to differentiate and relate features for the upstream and downstream traffic of the flow, and second – and more important – the graph explicitly represents the relations between different flows, which are connected to the same source/destination hosts.

Note that a more straightforward representation would be to consider only hosts as graph nodes, and flows as graph edges connecting the src/dst hosts. However, the decision to add specific nodes representing each flow was driven by the way GNN models operate. Note that GNNs consider only as learnable objects the hidden states of nodes in input graphs (as described in Sec. 3). As a result, to properly learn embeddings on flows, it is needed to represent them as nodes of the graph. Note that this graph representation includes heterogeneous elements (i.e., hosts and flows), which is not well supported by standard GNN models. This call us to devise a new message-passing architecture specifically adapted to process and learn the host-connection graph described in this section.

4.2 GNN model description

This section describes the proposed GNN model, which comprises a non-standard message-passing algorithm that adapts to the needs of the network intrusion detection problem, considering as input the host-connection graph representation described in Section 4.1.

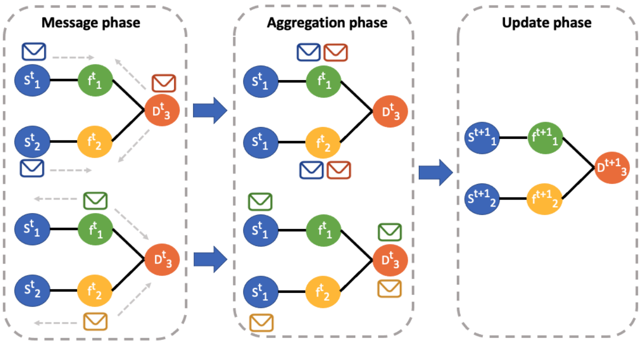

Figure 3: Illustration of the message-passing phase of the proposed GNN-based NIDS.

Let us define h_i^t as the hidden state of node i during iteration t, and X_i as the initial features for node hidden states. In the host-connection graph, nodes can represent a host, or a flow, so the hidden states of these nodes will be typically initialized with features of different nature. These features will depend on the monitoring data accessible in the network. Without loss of generality, let us assume that initial features are $X_i = [x_0, ..., x_k]$. In this case, we form the initial hidden state of flow i as follows:

$$h_i^0 = [x_0, ..., x_k, 0, 0, 0..., 0] \tag{5}$$

Note that hidden state vectors have a pre-defined length typically larger than the number of elements in the initial feature vector. Thus, hidden states are zero-padded.

Alternatively, if node i represents a host, we simply encode all ones in the initial hidden-state h_i^0. In this context, it is important to avoid identifiers on nodes (e.g., avoid IP addresses or prefixes to initialize hosts). This would not be desirable for the model, as the objective is to focus on the structural flow patterns, thus achieving a more general and robust characterization of attacks.

We first describe the message-passing phase of the GNN, which naturally considers the heterogeneity of the graph. Figure 3 illustrates the message-passing process. Formally, we apply the following operation in each message-passing iteration $t \in [T]$:

$$a_i^t = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \sigma_{type}(h_i^t \mid\mid h_j^t) \tag{6}$$

$$h_i^{t+1} = \delta_{type}(h_i^t \mid\mid a_i^t) \tag{7}$$

First, the model applies a learnable message function σ_{type} given the concatenation of the hidden states of two connected nodes –i.e., an edge in the input graph of the GNN. Here, σ_{type} implicitly comprises two possible functions, which respectively depend on the type of edge where they are applied: σ_{sf} for edges ($S \rightarrow f$), and σ_{fd} for edges ($f \rightarrow D$), according to the description of the host-connection graph in Section 4.1.

Afterward, an aggregation function is applied to the messages computed on each node. For this, we apply an element-wise mean over messages. In our case, using this function helps better normalize data across the multiple message-passing iterations, rather than using a standard element-wise summation.

Finally, the hidden states are updated considering the information collected in the new aggregated message. This is done by applying the update function δ_{type} to the aggregated message and the current hidden state of the node. Similarly to the message function, δ_{type} comprises two different learnable functions (δ_h and δ_f) respectively applied to update the hosts' and the flows' hidden states.

As a result, the σ_{sf}, σ_{fd}, δ_h and δ_f functions are all learnable functions than can be approximated by neural networks during training. Particularly, we implement σ_{sf} and σ_{fd} as 2-layer fully-connected NNs, while δ_h and δ_f are modeled as Gated Recurrent Units (GRUs [11]).

Finally we define the readout function $r(\cdot)$ as follows:

$$y_i = r(h_i^T) \tag{8}$$

The function $r(\cdot)$ takes as input the final hidden states of each flow, and outputs the predicted class for the flow (either a specific attack, or benign traffic). This function is implemented with a 3-layer fully-connected NN, where the output classes are represented via a one-hot encoding.

We use *ReLU* activation functions on all the layers of the different NNs mentioned above. Except for the last layer of the $r(\cdot)$ function, which uses a *softmax* activation. As we apply this GNN model for multi-class classification, we use a *categorical cross-entropy* loss function for training. However, the model could also be directly used for binary classification (e.g., classify flows on malicious or benign traffic) using a *binary cross-entropy* loss function instead. We set the number of message-passing iterations to $T = 8$, and the size of the hidden states to 128 elements. For additional details, we refer the interested reader to our publicly available implementation [7] using the IGNNITION framework [12].

5. EVALUATION

This section presents an evaluation of the proposed GNN-based NIDS, following two main directions. First, we evaluate the accuracy of the system compared to other state-of-the-art ML-based NIDS, using the well-known CIC-IDS2017 dataset [8]. Then, we artificially generate some common adversarial attacks in the previous dataset, to analyze the robustness of our GNN model compared to the other ML-based benchmarks.

5.1 Dataset

To evaluate the proposed GNN-based NIDS – described in Sec. 4– we use the well-known CIC-IDS2017 [8], which contains a representative collection of up-to-date attacks well mixed with real-world traffic. More in detail, malicious traffic is classified in 7 broad classes of attacks: Brute Force, Heartbleed, Botnets, DoS, DDoS, Infiltrations and Web attacks. In total, there are 15 different sub-classes of attacks. Likewise, each flow record aggregates a total of 80 features. We select a subset of them as input features of the evaluated models according to feature selection performed in [8].

We evaluate our model with a training and validation dataset, generated through a random split of 80% and 20% of graph samples respectively – totaling 895,400 flows for training, and 223,850 flows for validation. In our experiments, we show the results averaging over 5 cross-validations following the aforementioned splitting methodology.

Table 1: Weighted F1-Score of different ML-based NIDS over the CIC-IDS2017 dataset (11 attack classes + Benign traffic).

Class label	**MLP**	**AdaBoost**	**RF**	**ID3**	**Our proposal**
Benign	0.67	0.68	**0.99**	**0.99**	**0.99**
SSH-Patator	0.0	0.0	**0.99**	**0.99**	0.98
FTP-Patator	0.0	0.0	**0.99**	**0.99**	**0.99**
DoS GoldenEye	0.12	0.0	0.97	0.96	**0.99**
DosHulk	0.63	0.63	**0.99**	**0.99**	**0.99**
DoS slowloris	0.02	0.0	**0.99**	**0.99**	0.98
DoS Slowhttptest	0.01	0.0	**0.98**	**0.98**	0.97
DDoS	0.51	0.0	**0.99**	**0.99**	**0.99**
Web-Brute Force	0.0	0.0	**0.82**	0.76	0.73
Web-XSS	0.0	0.0	0.69	0.65	**0.83**
Bot	0.0	0.0	**0.98**	**0.98**	**0.98**
Port Scan	0.78	0.0	**0.99**	**0.99**	**0.99**

5.2 Experimental results of the NIDS

This section evaluates the accuracy of the proposed GNN-based NIDS over the CIC-IDS2017 dataset (Sec. 5.1).

A main difficulty when training ML-based NIDS is that datasets are inherently imbalanced, having a great bulk of benign traffic and a small portion of traffic related to attacks. For instance, in the CIC-IDS2017 dataset, malicious traffic represents only ≈12% of the flows. To address this, we first apply a pre-processing stage on both the training and validation datasets. Particularly, we randomly drop 90% of the graphs containing only normal traffic (Benign class), thus over-representing traffic belonging to attacks. For the evaluation, we consider only classes with more than 100 flow samples, resulting in 12 different sub-classes. We tested the use of loss functions specifically designed for imbalanced datasets (e.g., Focal loss [13]), however we did not find a significant improvement, finally using a common categorical cross-entropy loss function.

Table 1 summarizes the accuracy achieved by our GNN-based NIDS with respect to a collection of ML methods commonly used in state-of-the-art NIDS. In particular, we benchmark our solution against a 3-layer Multilayer perceptron (MLP) [14], Ada-boost [15], Random Forest (RF) [16] and ID3 [17]. We use a standard weighted F1-score to measure the per-class accuracy, which unifies in a single metric the precision and recall of solutions. From these results, we can observe that the proposed model achieves a level of accuracy comparable to state-of-the-art ML methods, obtaining a weighted F1-score of 0.99 over all traffic flows.

5.3 Robustness Against Adversarial Attacks

Section 5.2 shows that the proposed GNN-based NIDS achieves similar accuracy to state-of-the-art ML-based NIDS over the CIC-IDS2017 dataset. Nevertheless, in Section 2 we discussed the limitations of traditional flow-based ML-based methods, which can be highly vulnerable to variations in individual flow features. This section aims to assess the robustness of the proposed NIDS when facing common adversarial attacks [5].

Typically, the packet size is a highly discriminative feature for detecting many attacks at the level of individual flows (e.g., DDoS). In our first experiment, we artificially increment the packet size of attack-related flows, in order to test the robustness of the proposed NIDS against this potential detection prevention method. Figure 4 (top) shows the evolution of the accuracy (weighted F1-score) as the packet size of attack-related flows is incremented ([0, 200] bytes). As we can observe, the proposed GNN model is able to keep

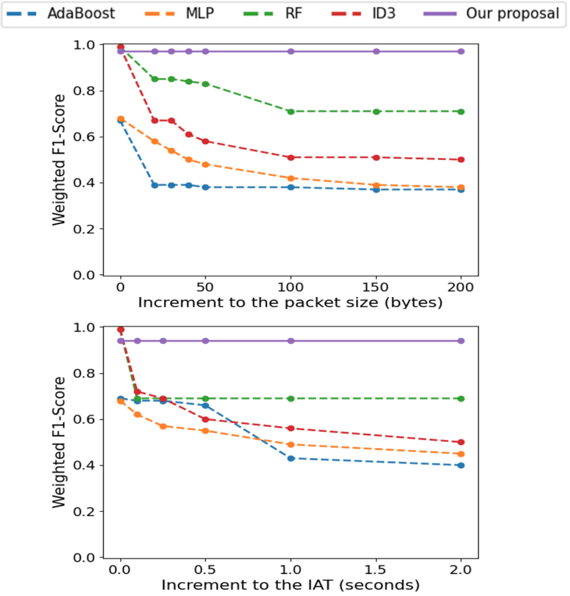

Figure 4: Weighted F1-Score of ML-based NIDS under potential adversarial attacks: variations on the packet size (top), and inter-arrival times (bottom).

the same level of accuracy as in the experiments of the previous section, showing robustness to this adversarial attack. In contrast, the other ML benchmarks suffer a significant degradation on their accuracy as the packet size increases.

In a second experiment, we make variations on the throughput of attack-related flows. For this, we artificially increase the packet inter-arrival times on these flows, so that they serve traffic at lower rates. Figure 4 (bottom) shows the results after applying increments to inter-arrival times ([0, 2] seconds), showing again that the proposed model maintains its base level of accuracy; while the other ML benchmarks considerably decrease their weighted F1-score (up to 50%).

Overall, we can observe in Figure 4 that all the baseline ML models evaluated exhibit strong vulnerability against common adversarial attacks that modify flow-level features. In contrast, the proposed GNN model maintains the same level of accuracy, being completely robust to these detection prevention methods. This is mainly due to its ability to capture the structural flow patterns of attacks, which remain unchanged even after varying flow features.

6. RELATED WORK

The application of ML for intrusion and anomaly detection has been largely investigated by the research community. First, ML-based solutions proposed the use of traditional ML algorithms, such as K-nearest neighbors (KNN), Support Vector Machines (SVM), Random Forests (RF), or a combination of them, to classify network attacks. There are numerous surveys that cover the vast related work in this area (e.g., [2, 3, 18, 19]).

More recent works propose also the use of Deep Learning techniques (i.e., neural networks). For instance, [20] proposes a NIDS that combines Deep Autoencoders and Long Short-Term Memory (LSTM) cells. In this system, the autoencoder learns relevant flow embeddings, which are then fed to a LSTM model to classify the attacks on flows, by opportunistically exploiting the temporal dependencies in the data. Other works such as [21] propose the use of Convolutional Neural Networks (CNN) to extract meaningful information from NIDS data.

Many ML-based NIDS treat and classify traffic flows independently, which is a limiting factor for the detection of common multi-flow attacks (e.g., DDoS, network scans) [2]. In this context, some works have explored the representation of traffic into clusters [22] or graphs [23, 24]. Likewise, some recent works such as [25, 26] propose the use of graph-based deep learning to exploit the relationship among network connections, showing significant improvement for malware detection in mobile applications. Similarly, [27] approaches the problem of Botnet detection assuming visibility of the full botnet topology. However, none of these previous works demonstrate robustness against adversarial attacks that produce variations on flow features to evade detection, such as the packet size, or throughput (e.g., inter-arrival times).

Recently, some pioneering works started to unveil the potential of GNNs for other network-related problems, such as network modeling [28, 29], network optimization [30], network planning [31] or network troubleshooting [32, 33]. In this work, we show that GNNs can also represent a breakthrough in the field of network intrusion detection.

7. CONCLUSIONS

In this paper we motivate the use of Graph Neural Networks (GNNs) to develop accurate and robust NIDS. We argue that, to achieve effective ML-based NIDS, it is essential not only to collect relevant patterns on individual flow features, but also to capture meaningful structural flow patterns that characterize different attacks (e.g., DDoS, port/network scans, brute force attacks). To this end, we first present a graph representation that properly structures the properties of flows and their relationships in the network. Then, we present a novel GNN architecture specifically designed to learn and generalize over the previous graph-structured information.

First, we have tested the accuracy of our model, showing comparable results to state-of-the-art ML-based NIDS in the well-known CIC-IDS2017 dataset. Then, we have tested the solution against two common adversarial attacks that intentionally modify relevant flow features on attack-related flows (packet size and inter-arrival times) to evade detection. The results show that, while the proposed GNN model is completely robust to these attacks, state-of-the-art ML models for NIDS degrade their accuracy up to 50% (weighted F1-score). This is mainly thanks to the capability of the proposed GNN model to *learn* the inherent structural flow patterns that compose different attacks. These structural patterns represent a deeper and more robust knowledge about attacks, as they typically remain unchanged over time, and across different networks.

Acknowledgements

This publication is part of the Spanish I+D+i project TRAINER-A (ref. PID2020-118011GB-C21), funded by MCIN/AEI/10.13039/501100011033. This work is also partially funded by the Catalan Institution for Research and Advanced Studies (ICREA), and by the European Union's Horizon 2020 research and innovation programme within the framework of the NGI-POINTER Project, funded under grant agreement No. 871528. This article reflects only the authors' view; the European Commission is not responsible for any use that may be made of the information it contains.

References

[1] IBM and the Ponemon Institute. Cost of a Data Breach Report 2020. `https://www.ibm.com/security/data-breach`

[2] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. 2019. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity* 2, 1 (2019), 1–22.

[3] Paulo Angelo Alves Resende and André Costa Drummond. 2018. A survey of random forest based methods for intrusion detection systems. *ACM Computing Surveys (CSUR)* 51, 3 (2018), 1–36.

[4] Robin Sommer and Vern Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE symposium on security and privacy.* 305–316.

[5] Igino Corona, Giorgio Giacinto, and Fabio Roli. 2013. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences* 239 (2013).

[6] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

[7] Barcelona Neural Networking center. GNN-NIDS. `https://github.com/BNN-UPC/GNN-NIDS`. Accessed Oct. 20, 2021.

[8] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSP* 1 (2018), 108–116.

[9] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[10] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of the International Conference on Machine Learning (ICML)*, Vol. 70. 1263–1272.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[12] David Pujol-Perich, José Suárez-Varela, Albert Cabellos-Aparicio, and Pere Barlet-Ros. 2021. IGNNITION: Bridging the Gap Between Graph Neural Networks and Networking Systems. *IEEE Network (in press)* (2021).

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision.* 2980–2988.

[14] Matt W Gardner and SR Dorling. 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment* 32, 14-15 (1998), 2627–2636.

[15] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. 2009. Multi-class adaboost. *Statistics and its Interface* 2, 3 (2009), 349–360.

[16] Gérard Biau and Erwan Scornet. 2016. A random forest guided tour. *Test* 25, 2 (2016), 197–227.

[17] Badr Hssina, Abdelkarim Merbouha, Hanane Ezzikouri, and Mohammed Erritali. 2014. A comparative study of decision tree ID3 and C4. 5. *International Journal of Advanced Computer Science and Applications* 4, 2 (2014), 13–19.

[18] Ali A Ghorbani, Wei Lu, and Mahbod Tavallaee. 2009. *Network intrusion detection and prevention: concepts and techniques.* Vol. 47. Springer Science & Business Media.

[19] Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security* 28, 1-2 (2009), 18–28.

[20] Haitao He, Xiaobing Sun, Hongdou He, Guyu Zhao, Ligang He, and Jiadong Ren. 2019. A novel multimodal-sequential approach based on multi-view features for network intrusion detection. *IEEE Access* 7 (2019), 183207–183221.

[21] Mohammad Mehedi Hassan, Abdu Gumaei, Ahmed Alsanad, Majed Alrubaian, and Giancarlo Fortino. 2020. A hybrid deep learning model for efficient intrusion detection in big data environment. *Information Sciences* 513 (2020), 386–396.

[22] Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. 2014. An empirical comparison of botnet detection methods. *computers & security* 45 (2014), 100–123.

[23] Siddharth Bhatia, Bryan Hooi, Minji Yoon, Kijung Shin, and Christos Faloutsos. 2020. Midas: Microcluster-based detector of anomalies in edge streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3242–3249.

[24] Siddharth Bhatia, Arjit Jain, Pan Li, Ritesh Kumar, and Bryan Hooi. 2021. MStream: Fast Anomaly Detection in Multi-Aspect Streams. In *Proceedings of the Web Conference 2021.* 3371–3382.

[25] Peng Xu, Claudia Eckert, and Apostolis Zarras. 2021. Detecting and categorizing Android malware with graph neural networks. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing.* 409–412.

[26] Julian Busch, Anton Kocheturov, Volker Tresp, and Thomas Seidl. 2021. NF-GNN: Network Flow Graph Neural Networks for Malware Detection and Classification. *arXiv preprint arXiv:2103.03939* (2021).

[27] Jiawei Zhou, Zhiying Xu, Alexander M Rush, and Minlan Yu. 2020. Automating Botnet Detection with Graph Neural Networks. *arXiv preprint arXiv:2003.06344* (2020).

[28] Krzysztof Rusek, José Suárez-Varela, Albert Mestres, Pere Barlet-Ros, and Albert Cabellos-Aparicio. 2019. Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN. In *Proceedings of the 2019 ACM Symposium on SDN Research*. 140–151.

[29] Krzysztof Rusek, José Suárez-Varela, Paul Almasan, Pere Barlet-Ros, and Albert Cabellos-Aparicio. 2020. RouteNet: Leveraging Graph Neural Networks for network modeling and optimization in SDN. *IEEE Journal on Selected Areas in Communications* 38, 10 (2020), 2260–2270.

[30] Paul Almasan, José Suárez-Varela, Arnau Badia-Sampera, Krzysztof Rusek, Pere Barlet-Ros, and Albert Cabellos-Aparicio. 2019. Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *arXiv preprint arXiv:1910.07421* (2019).

[31] Hang Zhu, Varun Gupta, Satyajeet Singh Ahuja, Yuandong Tian, Ying Zhang, and Xin Jin. 2021. Network Planning with Deep Reinforcement Learning. In *Proceedings of ACM SIGCOMM 2021*.

[32] Zili Meng, Minhu Wang, Jiasong Bai, Mingwei Xu, Hongzi Mao, and Hongxin Hu. 2020. Interpreting deep learning-based networking systems. In *Proceedings of ACM SIGCOMM 2020*. 154–171.

[33] David Pujol-Perich, José Suárez-Varela, Shihan Xiao, Bo Wu, Albert Cabellos-Aparicio, and Pere Barlet-Ros. 2021. NetXplain: Real-time explainability of Graph Neural Networks applied to networking. *ITU Journal on Future and Evolving Technologies* (2021).