Investigating Graph Neural Network Approaches for Network Intrusion Detection

Chunlu Wang*
Jesus College
University of Cambridge
cw829@cam.ac.uk

Xiaochen Zhu*
King's College
University of Cambridge
xz479@cam.ac.uk

Abstract

With the popularisation of the Internet and the advancement of informatics technology, network users are confronting more frequent, sophisticated and concealing cyberattacks. Traditional Network Intrusion Detection Systems (NIDS) have become insufficient when handling these scenarios. Machine learning techniques such as Support Vector Machine (SVM), Decision Trees, have been applied for better generalisation in malicious traffic classification. However, these approaches ignore the natural underlying graph structure of networks. To exploit this property, we implement a graph neural network-based Message-Passing Neural Network (MPNN) model for better classification. In addition, to handle real-life dynamic traffic, we also designed recurrent MPNN (RMPNN) to process time series of traffic to detect attacks spanning multiple time steps. We test our models on a partially synthesised dataset and demonstrate their superiority over traditional machine learning algorithms.

Statement of contribution

We, Chunlu Wang and Xiaochen Zhu of the University of Cambridge, jointly declare that our work towards this project has been executed as follows:

- **Chunlu Wang** contributed by designing and implementing the data sampler and preprocessing with configurations, implementing baseline models, running experiments and Kathará realizations , writing the Experiment Details, Evaluation, and Discussion sections, and editing other sections;
- **Xiaochen Zhu** contributed by designing the data sampler and experiments, implementing MPNN and RMPNN models used in this project, drawing Pipeline and RMPNN figures, writing the Abstract, Introduction, Related Work, Task Definition, and Conclusion and Future Work sections, and revising other sections.

We both independently submit identical copies of this paper, certifying this statement to be correct.

GitHub repository with commit log

The companion source code for our project may be found at: https://github.com/RaymondUoE/L45_project.

1 Introduction

In recent years, the dramatic increase in the number of cyberattacks has become a major concern for organisations and governments worldwide [1]. Traditional network intrusion detection systems

*Equal contribution.

Chunlu Wang & Xiaochen Zhu, Investigating Graph Neural Network Approaches for Network Intrusion Detection. *Representation Learning on Graphs and Networks (L45), University of Cambridge*.

(NIDS) with signature-based approaches often require human effort such as carefully engineered firewall rules which are difficult to scale. Such methods have proven to be insufficient in detecting sophisticated and evolving attacks [2, 3]. By contrast, anomaly-based approaches (A-NIDS) analyse the pattern of network traffic. Instead of identifying pre-defined signatures, the system detects anomalous behaviours in the network. Building such systems poses a greater challenge, as the rigid policies (e.g. knowledge-based, statistical-based) introduce false positives, hindering access for benign users [4]. With the development of machine learning, techniques such as bayesian analysis [5], support vector machines (SVM) [6], convolutional neural networks (CNN) [7], are applied to build more robust and scalable A-NIDS and achieved promising results. However, these approaches neglected the underlying graph structure of a network.

A network of computers can be abstracted to a graph, where users (IP addresses) are nodes, and network traffics are edges. Given such a structure, it's natural to introduce graph neural networks (GNN) for intrusion detection. GNNs are a type of deep learning algorithm that processes data with graph structures and have gained recent attention in the field of cybersecurity for their potential in network intrusion detection and security analysis [8, 9].

In this report, we explore the use of GNNs for detecting malicious traffic in a network. We start with node-level classification to detect malicious nodes. Then, we perform graph-level classification asserting the integrity of an entire network. Moreover, we synthesise a toy-problem dataset comprising dynamic network traffic to provide insight into the generalisation of our model to dynamic networks with temporal input.

2 Related Work

Traditional machine learning approach to network security analysis, for example, SVM and decision trees, performs classification task at a node-to-node level. Such methods require the individual traffic between two nodes to be recorded and labelled [10]. Although these algorithms perform well in terms of metrics, they are operating under a simplified scenario which deviates from real-world applications as valuable information is omitted in the datasets used. For instance, multi-flow attacks such as DDoS and port scanning, are often performed in a multi-to-one manner, where a single packet could be benign in isolation but malicious when examined with a larger scope [11, 12]. Moreover, the limitation becomes more evident when traffic with longer time spans or a mixture of benign and malicious packets are involved in such attacks as ARP spoofing and zombie network [13, 14]. These examples indicate that the decent numerical performance of traditional NIDS is rather misleading and urges a rethink in building more sophisticated NIDS with surveillance techniques on a global scale.

Graphs naturally represent the structure of a network and also the inter-dependencies among nodes (IP address) and their edges (network traffic). By processing our data using graph-based neural networks, we can obtain ambient-rich embedding on top of which we build our classifier. Such practice has become a fashion in recent research. Pujol-Perich et al. [12] has applied a simple Graph Neural Network (GNN) and achieved better performance than traditional machine learning techniques in traffic classification as an edge classification task. Lo et al. [9] also proposed a GNN model but with additional topological information to process the network flow data among Internet-of-Things (IoT) devices which provides a well-cluster embedding for different types of network intrusions traffic. Busch et al. [8] also utilises the node information of each user (OS, Software) together with the network traffic for detailed malware detection. Though the previous work illustrates the advantage of graph-based models, they have not fully represented the networks' dynamics. More precisely speaking, the previous models are not expressive enough and do not take the inter-dependency between traffic with different time steps into account. In our work, we will use a more expressive model, the Message Passing Neural Network (MPNN), and investigate its performance on time series of network traffic flow with Recurrent Neural Network (RNN) encapsulations.

3 Task Definition

In our project, we define this intrusion detection task as a supervised learning task. We firstly require labelled network traffic as our data set. The pipeline of our model is the following.

1. Convert network traffic to graph structure with IP addresses as nodes and traffic between nodes as edges.

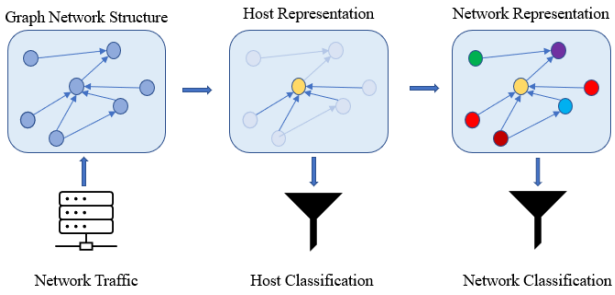

Figure 1: Pipeline for graph neural network based intrusion detection model.

2. Use graph neural network to aggregate traffic of node neighbours and return a node representation.
3. Aggregate and pool nodes representations and return a network representation.

Our graph neural network model is essentially an encoder to acquire better representations of the given network and each individual connected device. The classification model could be built independently using our model's output. A visualization of our pipeline is depicted in Figure 1.

4 Experiment Details

In this section, we describe the data set we used and our approaches for data augmentation. We described the Message Passing Neural network as the graph neural network we chose for the classification task. In addition, we also present the recurrent version of this model for handling time series of network traffic data.

4.1 Dataset, Subsampling, and Preprocessing

We use HIKARI-2021 and HIKARI-2022 [15], datasets containing benign and malicious network traffic in the form of packet traces, generated using real-world benign and multi-type attacking profiles. Each entry represents a network packet, with 26 features, source and target IPs, as well as a label. The combined dataset contains over 700,000 individual entries, forming a giant network traffic graph with IPs representing nodes. The resulting network, therefore, is a directed multigraph. In order to obtain a "snapshot" of the network at any given time, two subsampling procedures are implemented for obtaining static and dynamic graphs respectively.

For static graphs, the sampling procedure obtains a single snapshot of the network by first selecting a random node n_i. Using a configurable hop number k, all k-hops neighbours of node n_i are collected, where the neighbours of n_i are defined by $\mathcal{N}_{n_i} = \{n_j | (n_i, n_j) \in E\}$. Therefore, the resulting graph contains all such nodes that can be reached by a breadth-first search of depth k. In the case where $|\mathcal{N}_{n_i}|$ is impractically large, a cutoff number c is used to randomly select c nodes from $\mathcal{N}_{n_i}$ at each hop. The cutoff number c is chosen to be 20 in our implementation, as we found it best balances the sampled graph structure and the overall number of nodes. After obtaining a subgraph, we assign edge features by randomly sampling the dataset. This process is controlled by two parameters: the probability of positively labelled graphs for graph-level prediction p_g, and the probability of positively labelled edge traffics in such graphs p_e. A positive graph-level label is a binary label, indicating the presence of malicious traffic in the graph. As a result, a sampled graph has a probability of p_g containing only benign traffic. With a probability of $1 - p_g$, a graph may have edge-level features sampled in such a way that each edge has p_e chance of being malicious. Note that overall, The probability of a graph having a negative label is then $(1 - p_g) + (1 - p_e)^{|E|}$, where $|E|$ denotes the number of edges in the sampled graph. We assign non-informative features for all nodes. In addition, we synthesise artificial network graphs using a Barabási–Albert model [16] on the basis that growth and preferential attachment are real-world phenomena observed on computer networks [17]. This improves the variety of our network structures because we find the limited number of "hubs" in the dataset to be sampled disproportionally more likely because they have more neighbours. We generate Barabási–Albert graphs using the implementation of NetworkX [18], with 20 nodes of degree 2. The ratio of sampled and synthetic graphs can be controlled by a parameter, which we choose to be 0.5.

For dynamic graphs, we sample subgraphs from the dataset with the same parameters and synthesise Barabási–Albert graphs. Then, for each graph, we simulate new and disappearing network connections between nodes over discrete time steps T by randomly adding and removing edges. The probabilities of a directed edge to be added and an existing one to be removed are configurable. Here, we choose $p_{add} = 0.1$ and $p_{remove} = 0.05$. To serve as a starting point for our toy problem of detecting ARP spoofing on a dynamic network, we assign an integer categorical feature for each edge at each time step, indicating the type of traffic. Malicious traffic in this case is determined by a sequence of types, which is configurable. The sequence can have up to T elements. Then, at any time step t, a positive label is assigned to an edge $(n_i, n_j)_t \in E_t$ if the l-step-back type history of that edge matches any specified sequence of length l. For instance, an edge will be marked as positive at $t = 2$ if it has type 0, 1, and 2 at $t = 0, 1, 2$, and the sequence (1, 2) is specified to be an attack. The attack sequence can also be of length 1, where the attack is immediately detectable. This formulation is to simulate a type of man-in-the-middle ARP spoofing attack where the attack is not detectable until a sequence of traffic patterns is observed. The categorical type feature in this case can represent the result of a clustering model or a multi-class prediction model. However, it is left to be a randomly assigned integer in our proof-of-concept for simplicity's sake.

Overall, we obtained 10000 static graphs and 10000 dynamic graphs with each dynamic graph having five time steps. For each set of graphs, half of the total graphs are subsampled from the dataset and the other half are synthesised using a Barabási–Albert model. Figure 2 shows two sampled graphs as examples. They have "hubs"—nodes with a higher degree than others—which is natural in network graphs.

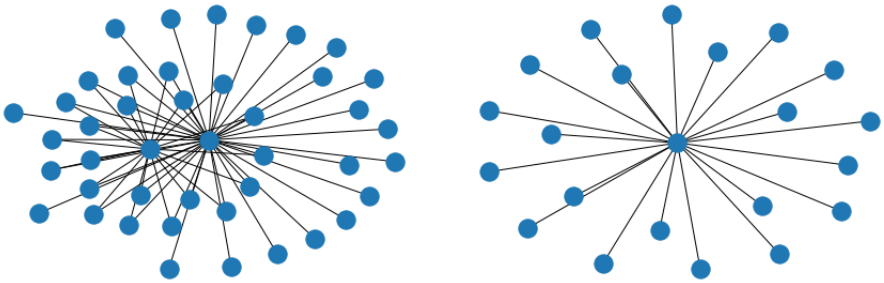

Figure 2: Two sampled subgraphs with two hubs (left) and one hub (right).

We sample the features and labels for static graphs from HIKARI-2021 and HIKARI-2022 [15]. We keep the original taxonomy of 6 traffic categories in the combined dataset as labels: benign (non-malicious traffic), background (non-malicious background communication), bruteforce (browser-based bruteforce attack against login systems), bruteforce-XML (bruteforce attack against login systems using XML-RPC protocol), probing (vulnerability probing that scans the system for vulnerabilities), and XMRIGCC CryptoMiner (malicious crypto-mining attack). For dynamic graphs, we manually configure multiple series of (randomly assigned) traffic types that are malicious. We report the proportion of positive nodes and graphs for static and dynamic graphs in Table 1. We kept the proportion of positive static graphs at a slight minority, but the proportion of positive nodes is much lower due to the large number of negative nodes. For dynamic graphs, however, nearly all graphs are positive because a graph almost always has at least one malicious node, given the number of time steps and the total number of nodes, despite a low probability of each node being malicious. To sample data that are closer to reality, we also used the Kathará emulator [2] to realise some network graphs.

4.2 Node- and Graph-Level Classification for Static Graphs

We use a message-passing neural network (MPNN) for node- and graph-level classification tasks, where the node-level objective is to predict whether or not a node is sending out a malicious message, and the graph-level objective is to predict the presence of such nodes. The MPNN layer updates node

[2]https://www.kathara.org/

	Positive Nodes	Positive Graphs
Static Graphs	2.31%	37.56%
Dynamic Graphs	1.53%	99.7%

Table 1: Proportions of positive labels for each set of graphs w.r.t node and graph levels. Positive nodes are all malicious nodes, no matter the category; Positive graphs are graphs that have at least one positive node (for static graphs) and at least one positive node at any time (for dynamic graphs).

representations h_i^ℓ as follows:

$$h_i^{\ell+1} = \phi\left(\left[h_i^\ell, \sum_{j \in \mathcal{N}_i} \psi\left([h_i^\ell, h_j^\ell, e_{ij}]\right)\right]\right), \tag{1}$$

where ϕ and ψ are update- and message-MLPs (multilayer perceptrons), $\mathcal{N}_i$ the set of neighbours of node n_i, and e_{ij} the edge-level features for edge $(n_i, n_j) \in E$. Then, a global mean pooling is used to obtain the graph-level representation after all layers:

$$h_G = \frac{1}{|V|} \sum_{i \in V} h_i^{\ell=L}. \tag{2}$$

Final classification results are derived using two classification heads, attached to (each of) the node representations at layer L: $h_i^{\ell=L}$ and to the pooled graph representation h_G.

4.3 Node- and Graph-Level Classification for Dynamic Graphs

We define dynamic networks as graphs that may have edges added and/or removed at each step over some discrete time steps. As previously discussed, dynamic networks are used to simulate real-world scenarios where new connections between nodes are established and existing ones expire. We take inspiration from recurrent neural networks (RNN) and implement a recurrent MPNN (RMPNN) to handle a graph's changes and malicious traffics that can only be detected upon receiving the last packet. RMPNN encodes the history of node embeddings by taking the node embeddings at time $t-1$ as an additional input at time t. As a result, RMPNN encodes the representation of a node i at time t using an MPNN with a residual connection between layers:

$$h_i^{t,\ell+1} = h_i^{t,\ell} + \phi\left(\left[h_i^{t,\ell}, \sum_{j \in \mathcal{N}_i} \psi\left([h_i^{t,\ell}, h_j^{t,\ell}, e_{ij}^t]\right)\right]\right), (\forall \ell \in \{0, 1, ..., L\}). \tag{3}$$

It encodes the history at $\ell = 0$ by:

$$h_i^{t,\ell=0} = \sigma\Big(MLP_{feat}(x_i^t) + MLP_{hist}(h_i^{t-1,\ell=L})\Big), \tag{4}$$

where σ is a non-linear activation function, which we choose to be ReLU [19], MLP_{feat} is a feature projection MLP that takes the feature vector of node i at time t as input, and MLP_{hist} is an MLP taking as input the previous final output of MPNN at time $t-1$. Figure 3 shows the structure of RMPNN. Similar to MPNN for static graphs, we derive node- and graph-level classification using two respective classification heads.

For static graphs, the classification task is formulated as a multi-class classification, with 5 classes indicating the presence and types of attack. They correspond to the different traffic types in the dataset [15]: benign, bruteforce, bruteforce-XML, probing, and XMRIGCC CryptoMiner. This taxonomy is relaxed for graph-level prediction, which outputs a binary label indicating whether an attacker is in the network. The classification problem is treated as a binary classification for the proof-of-concept toy problem for dynamic networks. The experiments were conducted with early stopping w.r.t cross-entropy loss on the validation set. The model is validated after each epoch and training is stopped if no improvements are observed for five epochs.

5 Evaluation

Our models are evaluated against a decision tree, an SVM [20], and a two-layer MLP on the static node- and graph-level classification tasks. It should be mentioned that due to the extreme imbalance

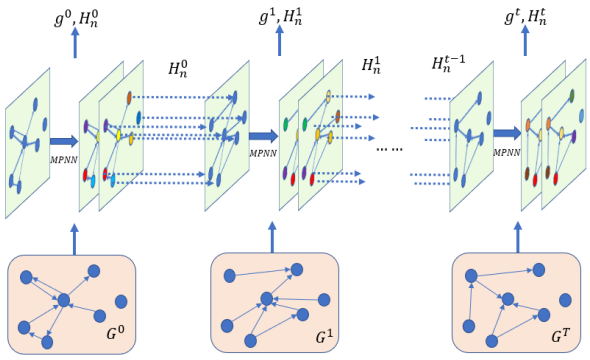

Figure 3: The architecture of RMPNN. At each time step t, a graph G^t is given to the network as input. The MPNN will output a graph representation g and hidden states of all nodes H^n after n layers of propagation. H_n^t will be add to the initial projection H_0^{t+1} of the nodes in the graph G^{t+1} at next time step for propagation.

between benign and malicious actors in the dataset and in reality, it is non-trivial to outperform the most-frequent baseline for node-level classification, which outputs non-malicious for all nodes. Table 1 shows the proportion of positive labels, from which the high accuracy of a most-frequent baseline can be induced. Despite the fact that accuracy can be misleading in this scenario, the accuracy measures are nonetheless reported for reference. We report the false negative rate (FNR) in addition to accuracy, on the basis that false negative classifications are much more costly in the domain of network security analysis. FNR is defined as:

$$FNR = \frac{FN}{FN + TP}, \tag{5}$$

which calculates the proportion of false negatives among all positive samples. A lower FNR indicates that fewer malicious actors are undetected. For graph-level classification, a lower FNR indicates fewer compromised networks are undetected. Table 2 shows the performance metrics of baseline models and MPNN in node and graph classification. We derive graph-level predictions by determining the existence of malicious nodes in the graph.

	Node Classification		**Graph Classification**	
Model	**ACC** ↑	**FNR** ↓	**ACC** ↑	**FNR** ↓
Decision Tree	98.58	39.22	67.52	33.12
SVM	6.44	7.47	37.44	0.0
MLP	97.67	99.49	73.00	65.60
MPNN	91.12	97.89	88.01	15.71

Table 2: Performance metrics of baseline models and MPNN in node and graph classification, where ↑ denotes higher is better and ↓ lower is better.

For dynamic graphs, we flatten the graphs over time t to measure baseline performance for node classification. This by default causes the model unable to distinguish such attacks that are undetectable until the last packet. For our toy problem, we highlight this difference by setting all attacks to be sequential. That is, all positive nodes are dependent on their history. We define a graph to be compromised if a malicious node is detected at any time step. Table 3 shows the performance of baseline models and RMPNN.

6 Discussion

It is immediately noticeable that the performance of all models is not ideal. This can be attributed to several factors. Firstly, our dataset is extremely unbalanced. This reflects real-world scenarios

Model	Node Classification ACC↑	Node Classification FNR↓	Graph Classification ACC↑	Graph Classification FNR↓
Decision Tree	98.46	100.0	0.28	100.0
SVM	76.17	11.19	99.72	0.0
MLP	98.46	100.0	0.28	100.0
RMPNN	94.42	99.38	40.98	80.64

Table 3: Performance metrics of baseline models and Recurrent MPNN in node and graph classification for dynamic graphs.

to a certain extent, where malicious actors and traffic are extremely rare, but this makes it non-trivial for models to outperform a most-frequent baseline. Even though we manually synthesised our dataset w.r.t graph labels, the node labels within the graph are still imbalanced. Additionally, this leads to misleading accuracy scores, where a high score does not necessarily reflect a better performance in detecting malicious actors and compromised networks. Secondly, there is room for improvements in dataset quality. Although we are using packet-specific features which are generated using real-life profiles in our experiments, we do not have access to node- and actor-specific features. We believe those features, such as the programmes running on a machine, would be beneficial to the task. However, obtaining such features would require expert domain knowledge which we do not have access to. Nevertheless, our model fully supports the use of node features, which are non-informative in our experiments. With respect to static graphs, the decision tree performs better than MPNN in node-level classification, but worse in graph classification because it only has access to the edge-level features without graph structure. The SVM fails because it overwhelmingly classifies nodes to be positive, which resulted in no negative labels being predicted for graph classification. The MLP performance is better than other baseline models in terms of graph classification but still underperforms compared to MPNN. In terms of dynamic graphs, it is clear that all three baseline models fail completely in determining if a graph is compromised. The SVM predicts every graph to be positive, while the other two models predict every graph to be negative. A similar trend can be observed for node classification, as the decision tree and MLP classifies every node to be negative. In contrast, our recurrent MPNN model achieves this with some degree of success.

Overall, using GNNs reduces false negative rates in node-level classification w.r.t static graphs, albeit by a slight amount compared to MLP. MPNN also is superior in graph classification on static graphs. Additionally, MPNN and RMPNN do not experience catastrophic failures that are observed to be common for other baseline models. This shows GNNs are more robust in network security analysis and can reduce the number of false negatives, which is crucial in this domain because of their high cost. We believe that the performance can be further improved with the integration of node-level features.

7 Conclusion & Future Work

In this report, we have proposed a MPNN graph neural network-based network intrusion detection model. We've conducted experiments for malicious traffic detection on the node, and graph levels. In addition, considering dynamic traffic in networks, we also implemented the recurrent neural network based RMPNN model to handle time series of data. Our model has demonstrated its superiority by using the graph structure of data and out-performs the traditional machine learning algorithms such as simple MLP and SVMs. In addition, we also used Kathará emulated data to illustrate the plausibility of our model's deployment in real-life scenarios.

Due to the limited scope of this mini-project, many valuable experiments are not included in this report. A very straightforward supplement could be conducted by collecting real-life data from a local Autonomous System Network (ASN) group and repeating the previous experiments with detailed analysis. It's also promising to generalise our methods for sub-graph classifications such as identifying a group of Zombie Networks within the traffic of a large Internet Service Provider (ISP). Moreover, it's also beneficial to investigate the effect of applying different graph neural network models such as graph isomorphism networks (GIN) [21]. More sophisticated data pre-processing, such as graph rewiring, could also be considered in order to alleviate the problem of over-squashing or under-reaching, which is common between traffic among multiple ASN groups especially when

the scale of the network increases [22–24]. As such, our project provides insight into using graph neural network approaches for better NIDS by exploiting the graph structure of the network.

References

[1] Wenli Duo, MengChu Zhou, and Abdullah Abusorrah. A survey of cyber attacks on cyber physical systems: Recent advances and challenges. *IEEE/CAA Journal of Automatica Sinica*, 9 (5):784–800, 2022. 1

[2] Sailesh Kumar. Survey of current network intrusion detection techniques. *Washington Univ. in St. Louis*, pages 1–18, 2007. 2

[3] Stephen Northcutt and Judy Novak. *Network intrusion detection*. Sams Publishing, 2002. 2

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

[5] Mrutyunjaya Panda, Ajith Abraham, and Manas Ranjan Patra. Discriminative multinomial naive bayes for network intrusion detection. In *2010 Sixth International Conference on Information Assurance and Security*, pages 5–10. IEEE, 2010. 2

[6] Iftikhar Ahmad, Mohammad Basheri, Muhammad Javed Iqbal, and Aneel Rahim. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE access*, 6:33789–33795, 2018. 2

[7] Onur Barut, Matthew Grohotolski, Connor DiLeo, Yan Luo, Peilong Li, and Tong Zhang. Machine learning based malware detection on encrypted traffic: A comprehensive performance study. In *Proceedings of the 7th International Conference on Networking, Systems and Security*, pages 45–55, 2020. 2

[8] Julian Busch, Anton Kocheturov, Volker Tresp, and Thomas Seidl. Nf-gnn: Network flow graph neural networks for malware detection and classification. In *33rd International Conference on Scientific and Statistical Database Management*, pages 121–132, 2021. 2

[9] Wai Weng Lo, Siamak Layeghy, Mohanad Sarhan, Marcus Gallagher, and Marius Portmann. E-graphsage: A graph neural network based intrusion detection system for iot. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2022. 2

[10] Mohammadreza Ektefa, Sara Memar, Fatimah Sidi, and Lilly Suriani Affendey. Intrusion detection using data mining techniques. In *2010 International Conference on Information Retrieval & Knowledge Management (CAMP)*, pages 200–203. IEEE, 2010. 2

[11] Adeilson Marques da Silva Cardoso, Rafael Fernandes Lopes, Ariel Soares Teles, and Fernando Benedito Veras Magalhães. Real-time ddos detection based on complex event processing for iot. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 273–274. IEEE, 2018. 2

[12] David Pujol-Perich, José Suárez-Varela, Albert Cabellos-Aparicio, and Pere Barlet-Ros. Unveiling the potential of graph neural networks for robust intrusion detection. *arXiv preprint arXiv:2107.14756*, 2021. 2

[13] Paulo Salvador, António Nogueira, Ulisses Franca, and Rui Valadas. Framework for zombie detection using neural networks. In *2009 Fourth International Conference on Internet Monitoring and Protection*, pages 14–20. IEEE, 2009. 2

[14] Sean Whalen. An introduction to arp spoofing. *Node99 [Online Document]*, 2001. 2

[15] Andrey Ferriyan, Achmad Husni Thamrin, Keiji Takeda, and Jun Murai. Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic. *Applied Sciences*, 11(17), 2021. ISSN 2076-3417. doi: 10.3390/app11177868. URL https://www.mdpi.com/2076-3417/11/17/7868. 3, 4, 5

[16] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002. doi: 10.1103/RevModPhys.74.47. URL https://link.aps.org/doi/10.1103/RevModPhys.74.47. 3

[17] Albert-László Barabási. *Network Science*. Cambridge University Press, Cambridge, 2015. 3

[18] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008. 3

[19] Abien Fred Agarap. Deep learning using rectified linear units (relu), 2019. 5

[20] Nello Cristianini and Elisa Ricci. *Support Vector Machines*, pages 928–932. Springer US, Boston, MA, 2008. ISBN 978-0-387-30162-4. doi: 10.1007/978-0-387-30162-4_415. URL https://doi.org/10.1007/978-0-387-30162-4_415. 5

[21] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 7

[22] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021. 8

[23] Anukool Lakhina, Konstantina Papagiannaki, Mark Crovella, Christophe Diot, Eric D Kolaczyk, and Nina Taft. Structural analysis of network traffic flows. In *Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 61–72, 2004.

[24] Theophilus Benson, Aditya Akella, and David A Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 267–280, 2010. 8

A Appendix

Parameter	Value
Hidden Dimension	128
Learning Rate	0.001
Batch Size	128
Number of MPNN Layers	4
Early Stopping Patience	5
Number of Epochs	50

Table 4: Hyperparameter Setup For MPNN and RMPNN Models

For Static Graphs		For Dynamic Graphs	
Parameter	**Value**	**Parameter**	**Value**
Number of Samples	10000	Number of Samples	10000
Proportion of Subsampled Graphs	0.5	Proportion of Subsampled Graphs	0.5
Number of hops	3	Number of hops	3
Probability of a graph to be sampled only from benign data	0.5	Time Steps t	5
Probability of a node sampled from malicious data, given it has access	0.2	Number of Types	4
		Probability of a non-existing edge to be added at each t	0.1
		Probability of an existing edge to be dropped at each t	0.05

Table 5: Hyperparameter Setup For Samplers

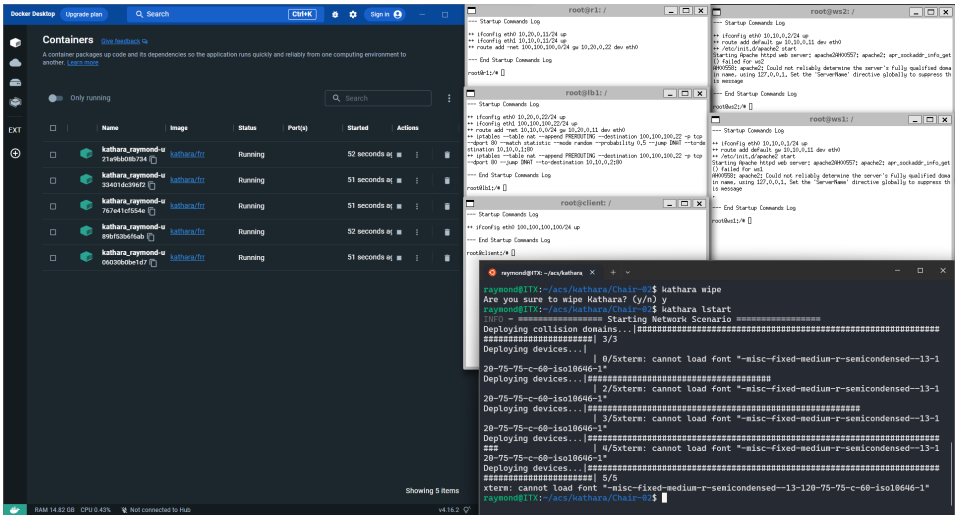

Figure 4: An example of Kathará emulated network