Graph Neural Network-Based Cyber Attack Classification

Alok Kumar Trivedi
Department of Computer Science
Indian Institute of Technology, Kanpur
alokt21@iitk.ac.in

Priyanka Bagade
Department of Computer Science
Indian Institute of Technology, Kanpur
pbagade@iitk.ac.in

***Abstract*—Intrusion Attacks on IoT devices are ever-increasing. Numerous works have been dedicated to utilizing machine learning or deep learning models for the detecting intrusion attacks within a system.. We have built a framework using Graph Neural Network(GNN) [2] model, which takes the feature set of the dataset and structure of the network into account to detect intrusion attacks on the network. Our GNN [2] model uses four aggregators along with three-degree scalars to aggregate neighborhood embedding, which ensures our model can discriminate between similar node embeddings. We evaluated our model on the publicly available AWID3 dataset, which has 802.11-based and non 802.11-based thirteen kinds of malware attack, and achieved an f1 score of 99.87. The performance of our model was compared with other popular models, and our model performed better than most models.**

***Index Terms*—intrusion detection, GNN, Principal Neighborhood Aggregation, IoT Network**

I. Introduction

As technology advances and internet penetrates even remote parts of the earth, there will be a rise in the quantity of Internet of Things (IoT) devices. According to Statistica Research Department [7], by the year 2025, it is estimated that more than 75 billion IoT devices will be connected to the internet. With the growing proliferation of IoT devices, the potential for malware attacks on those devices will also increase. Most of the IoT edge devices are often small and inexpensive. They also have very few computing capabilities and do not have a firewall installed, which makes them more vulnerable to malware attacks. Between January and June 2022, the number of reported malware attacks exceeded 1.5 billion.
An Intrusion Detection System(IDS) in IoT is a security mechanism that examines data flow to and from the device for any suspicious activity and alerts the system if it finds any threats. IDS system are of two types: Signature based and Anomaly based. Signature based attack detection system works by comparing the pattern of the incoming attack against the database of known malware attack patterns. Signature based attack detection system is very effective at detecting well-known malware attacks but very less effective at detecting unknown or new attacks. An anomaly-based detection system defines a set of rules; if any flow of data does not follow those rules, then they are considered anomalous. Signature based attack detection system works great for detecting known attacks, while Anomaly based system can detect new kinds of attacks with good accuracy. Most Anomaly based attack systems use machine learning or deep learning models to train the detection model. Machine learning or deep learning models are trained on big data sets to learn the rules to detect the anomalous flow of data.
With an increase in the size and complexity of the dataset, it is becoming challenging to extract an appropriate feature set from the data set for intrusion detection. Machine learning or deep learning models only focus on the properties of individual data flow and completely ignore the structural properties of the network. In this work, we have used GNN [2] for intrusion detection, where we treat each networking device as nodes and data flow between two devices as edges. We update the embedding of the graph nodes by not only considering the individual properties of the graph nodes but also taking the relationship between the nodes of the graph into account. We have used the Principal Neighborhood Aggregation(PNA) [9] framework, which uses four aggregators and three-degree scalars for aggregating data from node neighborhoods. Using multiple aggregators helps the model to discriminate between the nodes with similar representation.
The main contribution in this paper are as follows

- We converted the tabular data into graph data by treating each networking device as graph node and drew edge between two nodes if they are communicating with each other. We used PNA [9], which utilizes four aggregators and three degree scalars to update embedding of the graph nodes.
- We used AWID3 dataset [8] to train and evaluate our model. It has 254 data feature and 13 classes of malware attacks. AWID3 dataset has 802.11 based as well as non 802.11 based malicious attacks.

The subsequent sections of our paper are structured in the following manner: Section II explains GNN [2] and its working principle. We also discussed different variants of GNN [2]. The related work is described in section III. A basic IoT system model is reviewed in section IV. In section V, we explained the advantages of using the GNN [2] model over machine learning or deep learning for intrusion detection. We reviewed the AWID3 dataset in section VI and briefly discussed each malicious attack included in the dataset. Section VII describes the Principal Neighborhood Aggregation method and Target encoding method in detail. Section VIII is where we describe the experimental setup. Sections IX, X, and XI discuss our work's result, conclusion, and future prospectus.

II. Graph Neural Network

A Graph Neural Network(GNN) [2]is an artificial neural network designed to operate on Non-Euclidean data. GNN [2] is typically used to learn on and make a prediction about the data organized in a graph-like structure such as a Social Network, Internet of Things(IoT) network, and Biological Complex Network. In Graph neural network, combination of IP address and port address are treated as graph nodes, and the relation between two such address is expressed through edges between two nodes. Unlike traditional machine learning algorithms, GNN [2] leverages the rich relational information between data points along with the features of the data points for computations and predictions. Here, we determine the embedding of each node based on the structure of the graph.

GNN [2] are often able to extract more information from a set of data points as compared to other machine learning and deep learning algorithms, due to which they are considered suitable for small data sets.

In GNN [2], nodes exchange messages between their neighbors in the form of a vector and update their embedding using a non-linear activation function. The message-passing process between nodes can be explained in two steps, aggregation and updation.

Aggregation- We employ permutation invariant methods like mean, median, sum, etc., to combine the embedding of the neighbors of the node.

Updation- The embedding of a node are updated by combining the present embedding of the node and aggregated neighborhood information.

$$\begin{aligned} h_u^{k+1} &= UPDATE^k\left(h_u^k, AGGREGATE^k(h_v^k, \forall v \rightarrow N(u))\right) \\ &= UPDATE_k\left({}_u^k, m_{N_u}^k\right) \end{aligned}$$

h_u^k is the node embedding for layer l. The state of the node is getting updated by aggregating the neighborhood message and using it to update the node state. Here UPDATE, and AGGREGATE can be both neural networks as well non neural network function, and $m_{N_u}^k$ is the aggregated neighborhood message.

Figure 1 explains a GNN's aggregation and updation process. To update the embedding of node 1, we first aggregate the embedding of the neighbors of node one, and then we use the current embedding of node one and the aggregated embedding to get the embedding for the next layer.

Graph Neural Network can be used for the following tasks.

Node classification- The Embedding of an unlabelled Node is generated using an aggregation and updation process. Then we apply a simple neural network to find the label of that node based on its embedding.

Link Prediction- A GNN [2] can predict or establish the relationship between a pair of graph nodes. Node-based methods and subgraph-based methods are used to predict the missing link between nodes.

In the node-based method, the embeddings of the nodes are used to determine the link between a pair of nodes. The subgraph-based method involves extracting the local subgraph surrounding each link and utilizing the link representation obtained from a Graph Neural Network (GNN) trained on the learned subgraph representation.

Graph Classification- Graph classification involves classifying a graph based on its structures and the properties of its nodes. The aggregated embedding of each node is used to provide a label to an unlabelled graph.

A. *Different variants of GNN*

Graph Convolutional Network [20]: Graph convolutional Network(GCN) [20] is one of the most popular GNN architecture. GCN [20] is to a Non-Euclidean data set what Convolutional Neural Network(CNN) is to a euclidean dataset. Similar to CNN, it uses a single weight matrix per layer for dealing with nodes of various degrees.

$$H^{l+1} = \sigma\left(\hat{D}^{\frac{-1}{2}}\hat{A}\hat{D}^{\frac{-1}{2}}H^l W^l\right) \tag{1}$$

Here H^{l+1} stands for the embedding of graph nodes at layer l+1, W^l is the the weight matrix for l^{th} layer and $\hat{A}$ is self loop Adjacency matrix and $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$.

GCN [20] works great for semi-supervised tasks where labels for some graph nodes are missing, but it has enormous memory requirements as it requires us to store the adjacency matrix of size $n*n$ for n node graph.

Graph Attention Network [15]: Graph Attention Network(GAT) assigns attention weights to every node in the neighborhood of the graph. For a given node in the graph, the attention weights assigned to every other node are proportional to the influence other nodes have on the given node.

$$e_{ij} = a\left(W\vec{h_i}, W\vec{h_j}\right), j \in N_i \tag{2}$$

The attention coefficient, denoted as e_{ij}, signifies the influence of node j on node i.. We calculate the attention coefficient between every two nodes of the graph without considering the structure of the graph. If there does not exist a path between node i and j, then we consider the attention coefficient to be zero.

We normalize the attention coefficient using softmax activation

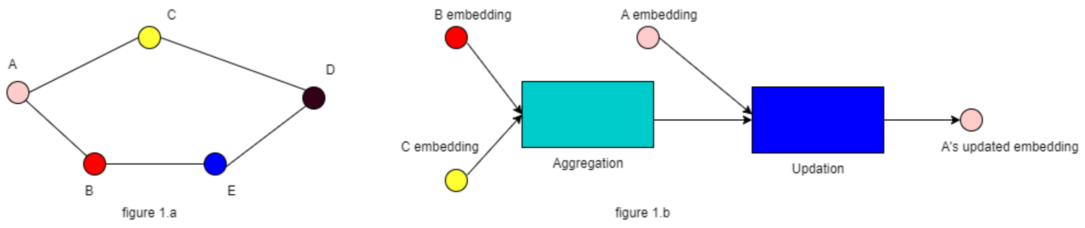

Fig. 1. GNN aggregation and updation process

function to compare the attention coefficient across different nodes.

$$\alpha_{ij} = \frac{exp(e_{ij})}{\Sigma_{k \in N_i} exp(e_{ik})} \tag{3}$$

GAT [15] has fewer memory requirements than GCN as we don't need to store the adjacency matrix. Also, it is very efficient, and the computation of the attention coefficient can be parallelized across all edges.

Graph Sage [22]: Graph Sage is a GNN framework for inductive representation learning on the graph. Most of the GNN [2] frameworks are transductive and cannot be used to generalize on unseen graphs. To generate the embedding of a new graph node, frameworks like GCN [20] have to be retrained. Graph sage can predict the embedding of the new node without retraining as Graph Sage trains its aggregator function to learn the embedding of the graph, so when a new node is added to the graph, it can induce the new node's embedding.

Graph Sage [22] framework is also very efficient as it only considers a sample of neighborhood nodes for the aggregation process. To update the embedding of a node, the Aggregator function combines the sampled neighborhood node's embedding with a weight, either fixed or learned to aggregate the neighborhood embedding. Aggregated neighborhood embedding is concatenated with the current node embedding to generate the subsequent layer node embedding. LSTM cell, Mean function, and a neural network layer with a max-pooling operator are popular aggregator functions.

III. Related Work

There have been many works involving machine learning for the intrusion detection task. Random Forest, Logistic Regression, KNN, and SVM applied by [12] for the intrusion detection task. They performed binary and multiclass classification on two publicly available datasets, UNSW-NB 15 and CICIDS 2017. Among all other ML models, SVM - Medium Gaussian was the best performing model, with a f1 score of .9404 for the binary classification task. In contrast, KNN weighted performed the best for the multiclass classification task with an f1 score of .9988. Similarly, [1], and [4] have applied machine learning models for intrusion detection. [16] used an ensemble model for intrusion detection in the Internet of Medical Things(IoMT) network. They used ensemble learning, employing decision tree, Naïve bays, and Random forest as a first-level learner. The outcome of these classifiers is used by xg Boost at second level for final classification. They evaluated their model on ToN-IoT dataset which contains data collected from heterogeneous IoT networks and got an accuracy of 96.35%.

Deep learning works such as [14], [5], [6] have used AWID2 dataset [5] for the model training.[14] used a Sparse Autoencoder(SAE) to extract features from the dataset, the feature extracted from SAE is then concatenated with the existing dataset. Then SVM is used for feature selection, and ANN is used for the classification process. They reported an f1-score of .992. [6] applied ANN for binary classification task and got an f1-score of .993. [5] developed the AWID2 dataset [5] and also built a NIDS using this dataset. They achieved an f1-score of .948 by using only 20 features in the dataset. [17] combined two deep learning models CNN and LSTM, for malware detection. They evaluated their model on a publicly available IoT Malware dataset. They got an accuracy of 99.83.

Above described papers use Machine learning or deep learning for the intrusion detection task. These models treat each data point independently and do not consider the relationship between the nodes for the model training. Although the above models have achieved good accuracy on their dataset, they might perform poorly if there is any slight change in the dataset, such as a change in the packet length of some of the flow of the test data set.

Some malicious attacks can easily be classified if we feed the structure of the network into the model, such as Denial of service(DoS) attack, as multiple requests to one node can easily be identified, but for the machine learning or deep learning models where each flow are considered independently, it will not be an easy task.

There have been few works where they have used GNN [2] for malware detection. [3] build two GNN model using Graph Attention Network(GAT) and Graph Convolutional Network(GCN) respectively to detect and classify malware attacks in intelligent transportation system(ITS). They collected android apk file from ISCX-AndroidBot-2015 [18] and CICMalDroid [19] dataset and passed them as input to GNN [2] models. They achieved a f1 score of .945 when they use Node2Vec[13] model as a feature generator and passed those features as node embedding to GAT [15] model as input. [3] used GNN [2] for intrusion detection task. They built heterogeneous graph, where each flow from the host to the destination was represented as nodes in the graph.

They used Message-Passing Neural Network (MPNN) [11] framework, where every node receives a message from its neighborhood's node and those massages are used to update the node state. They achieved a f1-score of 0.99.

IV. IoT Network - System Model

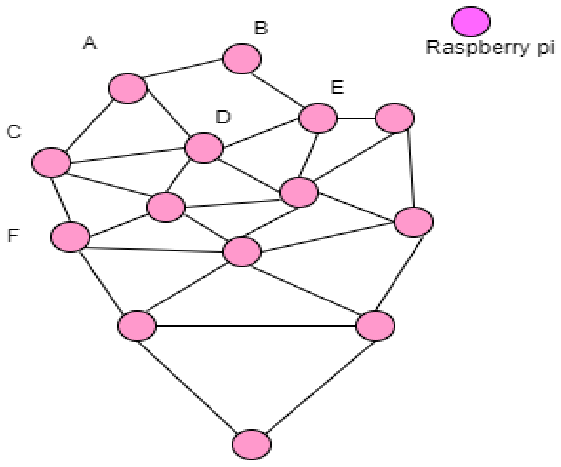

Fig. 2. A IoT network model

[h]
The above figure depicts a typical decentralized IoT network connected with a large number of IoT devices and those devices can communicate with one another directly. The edges between the graph nodes show the wireless communication channel through which different nodes communicate and exchange data. This configuration provides better security as data is not stored in any center node, and it is relatively easier to add or remove a device from the system.
The above IoT system follows a fog computing paradigm where computation and storage capabilities are placed closer to the edge device. Fog computing reduces latency and dependence on central nodes for storage and decision-making. Fog computing paradigm is very beneficial for IoMT, where patients need immediate medical attention in case of an emergency.

V. GNN for Intrusion Detection

Traditional machine learning and deep learning models treat each flow between server and client independently whereas graph neural network based models extract not only the individual features of flow but also the relation between different flows. This helps the GNN [2] model to extract more information and generate appropriate embedding for every node of graph which explains the relationship between the node and its neighbourhood.
For a given graph in figure 2, G, which has k node n_1, n_2,...., n_k and each node has a feature space X^F. During the Training process, GNN [2] updates the feature space of node A by prioritizing the feature space of those nodes that directly interact or exchange data with node A, such as nodes B and C, over nodes that are not directly communicating with node A. In contrast, machine learning and deep learning models consider each node to have the same impact over every other node present in the graph.
The data set we used for the intrusion detection task had client and server configuration where each client and server is treated as the node of the graph and the flow between a pair of nodes are used to connect the nodes.At every network layer of GNN [2], the nodes of the graph update their state vector and pass the updated state to their neighbors, thus the model was able to capture the structural dependencies and intricacies of the graph.

VI. Dataset [8]

We used the publicly available AWID3 [8] dataset created by the University of Aegean. AWID3 [8] dataset supplements and extends the previously developed AWID2 dataset [5]. AWID2 dataset[5] only had attacks based on 802.11, while AWID3 [8] contains attacks based on both 802.11 and non 802.11.The collected dataset has 254 features for each data point and 13 malicious attacks such as Deauthentication, Disassociation, SSH brute force, Botnet, Malware, [8] etc. They utilized 16 different physical devices and virtual machines to render the packet flow between the clients and the server.
A brief introduction of several malicious attacks present in the dataset is as follows.
Deauthentication [8]: It is a denial of service attack targeting the connection between the user and the wi-fi access point. This attack mainly takes place in 802.11 and can be mounted very easily. This attack involves a station receiving the deauthentication frame with a spoofed mac address from one of the access points.It serves as a preliminary stage for initiating more sophisticated attacks such as the evil twin attack..
Disassociation [8]: It refers to a form of Denial of Service attack wherein the attacker utilizes a disassociation frame to sever the wireless connection between the user and the Wi-Fi access point. It can also be used to launch a more severe attack like the evil twin attack.
Rogue AP [8]: A rogue access point is a device not getting administrated and is present in the network. The rogue access point is set up on the same channel and disguises itself using the identical MAC address and SSID as the target access point. They are deployed in WPA2-PSK authentication mode with PMF disabled. There could be severe repercussions for connecting to the rogue access point, such as data manipulation and information theft. It can also launch man in a middle attack where an attacker can control the communication between two or more clients.
Krack [8]: It is an acronym for Key Reinstallation Attack. It exploits the vulnerabilities of WPA2 (wi-fi Protected Access protocol). It takes advantage of a four-way handshake between the client and the access point. During the third step of the four-way handshake process, the attacker assumes the role of a man in the middle and deceives the victim into reinstalling the encryption key that is already in use. This is accomplished by

preventing message four from reaching the access point. When the encryption key is reinstalled, all the other parameters, like nonce are also set to the there original value. When the above process is continued for a prolonged period, the attacker can hack or hijack multiple devices connected to the wi-fi.

Krook [8]: It exploits the vulnerabilities of WPA2 (wi-fi Protected Access protocol). It impacts the encryption process used for secure data transmission over a wi-fi network. When a station disassociates from an access point, its encryption key resets to all zero. When the attacker forces a client to a disassociation state for a prolonged period, all the frames stored in WNIC get transmitted with all zero key encapsulation to the attacker.

SSH Brute Force attack [8]: SSH, also known as Secure Shell, is a network protocol that provides users a secure way to establish communication between a local machine and a remote host. Attackers carry out SSH brute force attacks by trying a username and password over thousands of servers.

Botnet [8]: A botnet attack involves attacking a group of internet-connected devices with malware, and then those devices become bots controlled by botheads to carry out various kinds of attacks such as DDOS, Brute force attacks.

A botnet attack can steal and manipulate the user and the system data, and monitor the user's activity. It can even install and run any application in the system.

Malware [8]: This attack uses malicious codes to attack specific parts of the operating system or any system app. It may lead to a more severe attack like ransomware, spyware, command and control, etc. There are mainly three types of malware attacks: Trojan horse, virus, and worm.

A malware attack can steal a user's information and credentials. It can corrupt critical OS files. A Malware attack may also lead to large-scale distributed denial of service (DDOS) attacks.

SQL Injection [8]: In this attack, a harmful SQL Query is inserted into input data to retrieve unnecessary information or tamper with the database. A SQL injection attack can access confidential data from the database, alter its contents, perform administrative tasks within the database, and occasionally even execute commands on the operating system.

SSDP amplification [8]: Simple Service Discovery Protocol (SSDP) is used by Plug Play (UPnP) devices to advertise their existence to another device on the network. SSDP amplification attack is a form of amplified DDOS (Distributed Denial of service) attack SSDP to carry out the attacks.

This attack entails utilizing WLAN stations to transmit a high volume of SSDP packets to each SSDP-enabled device, with the victim's IP address listed as the source IP address. As a result, every device responds to the victim with substantial packets, leading to a denial of service attack.

Evil Twin [8]: It is a man in the middle attack where the attacker sets up a wifi access point to steal information. The same channel is utilized by the rogue WiFi access point, and it disguises itself under the same MAC address and SSID as the target one. The attacker moves his wifi device closer to the victim and floods the original wifi device with a denial of service attack so that the victim connects to his device. The victim is redirected to a web page that imitates a WiFi login portal's design when they connect to the rogue AP. This attack is used to steal login credentials, and they can further attack the device to launch a ransomware attack, where the attacker can remotely operate the device.

Website Spoofing [8]: Similar to the Evil Twin attack, this redirects users to a web page designed identically to popular websites like Facebook, Instagram, etc., using ARP and DNS spoofing to steal their information.

VII. Methodology

A. Principal Neighbourhood Aggregation [9]

Principal neighborhood aggregation works on the principle that to discriminate between a multi-set of size n, we need at least n aggregators. It uses four permutation invariant aggregators: mean, max, median, and standard deviation, along with degree scalars, to determine the node embeddings of the graph. PNA algorithm also uses higher moments if the degree of a node is high and four aggregators cannot describe the neighborhood correctly. Other GNN architectures, such as GCN and GAT, use only one aggregator, which may not be enough to extract the valuable node information and may limit the learning capabilities of the GNN model.

The common aggregators used in PNA architecture are described briefly

Mean Aggregator- The mean aggregator calculates the average of all the incoming messages for each node of the graph.

$$\mu_i(X^l) = \frac{1}{d_i} * \sum_{j \in N(i)} X_j^l \tag{4}$$

Here u_i refers to the mean value for node i, X^l is the node feature for the layer l, d_i is the number of neighbors for node i, which remains unchanged in the case of a static graph and N_i to the set of neighbors for the node i.

Maximum and Minimum Aggregator- The max and min aggregators are similar to the max pooling and min pooling process used in Convolution Neural Network(CNN). It only considers the maximum or minimum value neighbor of the node for state updation.

$$\mathrm{max_i}(X^l) = max_{j \in N(i)} X_j^l \tag{5}$$

$$\mathrm{min_i}(X^l) = min_{j \in N(i)} X_j^l \tag{6}$$

Standard deviation Aggregator- The standard deviation aggregator is used to describe the spread of the neighborhood values for a node.

$$\sigma_i(X^l) = \sqrt{\mathrm{ReLu}(\mu_i(X_l^2) - (\mu_i(X_l))^2) + \epsilon} \tag{7}$$

σ_i refers to the standard deviation value for node i, and X^l represents the node feature for layer l. The Relu activation function filters out the negative values, and a small value is added so that the function stays differentiable.

Normalized moments aggregation - When the node's degree is high, four aggregators might not be enough to extract useful information from the neighboring nodes. In that case, PNA uses high moments to aggregate information.

$$M_n(X) = \sqrt[n]{E[X-u]^n} \tag{8}$$

In some cases where two nodes have similar neighborhood feature vectors but different node degrees, the aggregators discussed above are failed to distinguish between two such nodes. To solve such a problem, PNA uses the logarithm of degree-based scalars.

$$S_{amp}(d) = \frac{\log(d+1)}{\lambda} \tag{9}$$

$$\lambda = \frac{1}{|train|} \sum_{i \to train} \log(d_i + 1) \tag{10}$$

Here λ is the normalization parameter, and d is the degree of nodes. Degree-based scalar is further generalized by adding the variable α, which is positive for amplification, negative for attenuation, and zero for no scaling. The final expression for degree-based scalar is

$$S(d, a) = \left(\frac{\log(d+1)}{\lambda}\right)^{\alpha} \tag{11}$$

Here, value of α can vary from -1 to +1.
The three types of degree scalar and four aggregators are combined, resulting in 12 message vectors from each neighboring node.
The expression below explains the message passing and updation process for a layer of gnn using the Principal Neighborhood Aggregation(PNA) framework.

$$X_i^{t+1} = U\left(X_i^t, \oplus_{(i,j)\in E} M(X_i^t, E_{j\to i}, X_j^t)\right) \tag{12}$$

The $E_{j\to i}$ is the feature of the edge connecting node j to i. M and U are neural networks responsible for the aggregation and updation process.

[h] The figure 3 depicts the aggregation and scaling process of PNA. Node B is the only neighbor of Node A. we use four aggregators and three scalars, as explained above. We pass the aggregated message to the MLP model to get the aggregated neighborhood message vector.

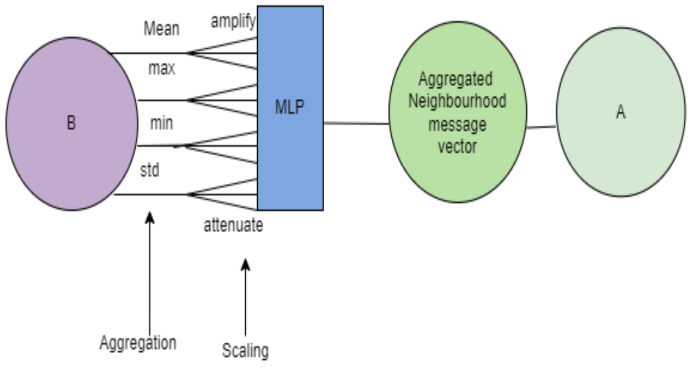

Fig. 3. Principal Neighbourhood aggregation

B. Target Encoding [10]

Target encoding is an appropriate alternative to one hot encoding. It controls the dimensionality of the dataset by reducing the number of columns as one hot encoding creates many sparse columns, which unnecessarily increases the size of the dataset. This problem is amplified if a column has too many categories and all of them are not relevant to the prediction task.
Our dataset had Many columns with categorical data type, and since the cardinality of many columns was more than 100, One hot encoding of categorical columns would have been impractical.
Target encoding is an encoding technique that replaces the categorical data with a value proportional to the effect that category has on the target. For a binary classifier, we replace the categorical data with the posterior probability of the target value equal to 1 when that category is present in the input data.

$$encoding = p(target = 1|x = c_i) \tag{13}$$

One of the problems with this approach is that by using the posterior probability of the target for encoding the categorical data, we are giving the information of target value distribution to the input data before the training process. This may lead to overfitting. This problem is called Target leakage.
One of the ways to prevent target leakage is to use Prior smoothing. Here we replace the categorical data with the combination of the posterior probability of target value equal to one when that category is present in the input data and the probability of target value equal to one for the whole data.

$$encoding = \alpha * p(target = 1|x = c_i) + (1-\alpha) * p(target = 1) \tag{14}$$

where α is the balancing factor. It is calculated by

$$\alpha = \frac{1}{1 + \exp - \frac{n-k}{f}} \tag{15}$$

Here f is the smoothing factor and k is the min_samples_leaf.
Target encoding for multiclass classification is similar to what we do for binary classification. Here, we encode the categorical data for each class independently. For example, for a ten-class classification task, we will have ten encoded columns for each categorical column in the dataset.

VIII. Experimental setup

A. Dataset collection

[h] The main protocols used in this dataset were IEEE 802.1X EAP, 802.11w, and IEEE 802.11ac-2013. As depicted in figure 4, they created a physical lab to render the flow of packets between servers and clients. They utilized sixteen

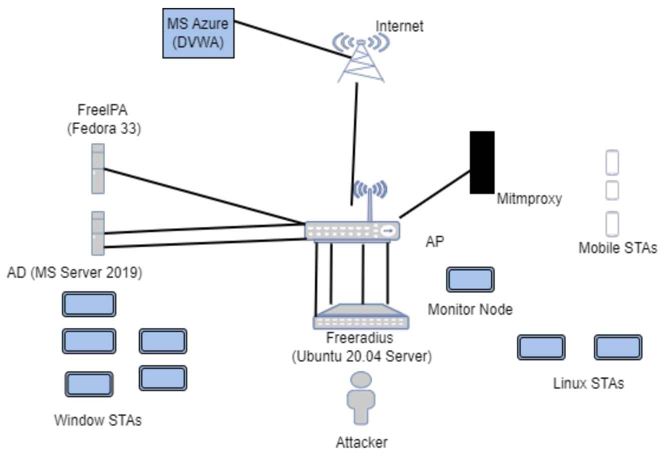

Fig. 4. Experimental Setup

different physical devices and virtual machines. Out of the sixteen devices, the attacker operating one of them as a laptop STA and ten were utilized as client STAs. Additionally, three devices served as servers, while the remaining three were assigned as Access Points (APs). For a more detailed description of the data collection process, we refer to [8].

B. Data Pre-Processing

The dataset we received was divided into 14 different classes. It had more than 50 million rows and 256 columns. As the dataset size was huge, we took a fraction of the dataset from each category so that every class had a significant representation. We removed columns where more than 40 % of values were Null or irrelevant for the intrusion detection task, and then we removed highly correlated columns. We converted some Numerical data type columns to the categorical column if there were very few unique real numbers getting repeated in that column.
We examined every categorical column, and if there were groups with less than 200 cardinalities in the column, we removed those groups from the dataset. Mac address column had values in hexadecimal format, we changed it to decimal values. We removed column with zero variance and used Target Encoding to convert categorical columns to numerical columns and MinMaxScaler to standardize the dataset. We further split the dataset into 70% train, 10% validation, and 20% test set.

$$data_{scaled} = \frac{data - data_{min}}{data_{max} - data_{min}} \quad (16)$$

C. Graph Building

We build a homogeneous graph by combining the ip address and port address. If node A is receiving or delivering data to node B, then node A and B are considered neighbors.

D. Training

Our model consists of a combination of the GNN [2] model and the MLP model. For the GNN model, we used PNA framework to determine the graph's embedding of every node. Later we passed those nodes embedding to a single-layer MLP model to classify the flow from the source to the destination into fourteen classes (Thirteen kinds of malicious attack + one Normal class).
We used three layers of the PNA framework to update the graph node embeddings. For the first layer operation, for any node, the embeddings of its neighbors are aggregated and concatenated with the current embedding of that node to get its first layer embedding. The first layer node embeddings of the graph indicate the relationship between the node and its one-hop neighbors.
The second layer node embeddings of the graph indicate the relationship between the node and its two-hope neighbors by transforming the first layer node embeddings using PNA process as explained in the last paragraph. We carried out this process for three layers as given our average graph diameter, building a PNA model with more than three layers would have resulted in overfitting.

IX. Result

A. Evaluation Metric

To evaluate our model's performance on the dataset [8] and compare it against the benchmark results, We used Precision, Recall, and F1-score as the evaluation metric. Precision determines the count of the attacks correctly classified among the total number of flows detected as an attack. Recall determines the count of the attacks detected among the total number of flows classified as an attack in the test set. F1 score is the harmonic mean of Precision and Recall.

$$Precision = \frac{TP}{TP + FP} \quad (17)$$

$$Recall = \frac{TP}{TP + FN} \quad (18)$$

$$f_1score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (19)$$

B. Experimental Result

Intrusion Attack	Precision	Recall	f1-score
Normal	1.00	1.00	1.00
Deauth	1.00	1.00	1.00
Disas	1.00	1.00	1.00
(Re)Assoc	1.00	0.98	0.99
RogueAP	1.00	1.00	1.00
Krack	1.00	0.99	1.00
Krook	1.00	1.00	1.00
SSH	0.99	0.99	0.99
Botnet	0.99	0.97	0.98
Malware	0.98	1.00	0.99
SQL_injection	1.00	0.86	0.92
SSDP	1.00	1.00	1.00
Evil_Twin	1.00	1.00	1.00
Website_spoofing	1.00	1.00	1.00

TABLE I
Class Wise Precision, Recall and F1 score

Model	Dataset	f1-score
PNA	AWID3 [8]	.9987
D-FES SVM [14]	AWID2 [5]	0.9992
Random Forest [5]	AWID2 [5]	0.944
ANN [6]	AWID2 [5]	0.993
Hybrid CNN-LSTM [17]	Internet of Things dataset	0.9983
Ensembel Model [16]	ToN_IoT dataset [21]	0.9635
Ensembel Model [16]	ToN_IoT dataset [21]	0.9635

TABLE II
CLASS WISE PRECISION, RECALL AND F1 SCORE

We trained our model for maximum 50 epoch and used an early stopping at 20 epoch. Our dataset has class imbalance problem where Normal and SSDP attack class were over represented. To solve that problem we used weighted precision, recall and F1-score. We evaluated our model's performance on AWID3 dataset [8], which supplements and extends the AWID2 dataset [5].
Table I shows the class wise performance of the model. Barring the SQL_injection class,our model has performed great. SQL_injection has been one of the most difficult malicious attack to detect, even the most sophisticated web application firewall cannot detect all the SQL_injection attacks. Most SQL_injection codes look like regular SQL query and get through the detection model.
[14], [5] ,and [6] evaluated their model's performance on AWID2 dataset [5], developed by University of Aegean. It is evident from table II that simple machine models [5], [16] were not able to perform as good as deep learning and GNN models. Deep learning and GNN models are better equipped to extract valuable feature information than machine learning models. [14] used a Sparse Autoencoder(SAE) to extract features from the dataset, the feature extracted from SAE is then concatenated with the existing dataset. Then SVM is used for feature selection, and ANN is used for the classification process. [6] trained a ANN model on the AWID2 [5] dataset, and the model performed better than machine learning models, similarly [17] used CNN-LSTM model where CNN extracted the feature from the dataset and LSTM model was used for classification. We used PNA model for the feature extraction process and MLP for the classification task.

X. FUTURE WORK

We successfully trained a GNN [2] model on intrusion detection dataset, however we considered the graph structure and node and edge features as static. We removed the time series column from the dataset.
We would like to incorporate time series analysis in graph neural network where the edge and node properties will be dynamic and we can effectively analyze the relationships between devices in a network and detect patterns that may indicate the presence of an intrusion.

XI. CONCLUSION

In this study, we developed a Graph Neural Network model using PNA [9] method to detect the malicious attack in a wi-fi network. We chose PNA [9] over other GNN frameworks, such as GCN [20] and GAT [15], as PNA [9] utilizes multiple aggregators to aggregate neighborhood message vectors. We used four permutation invariant aggregators and three degree scalars to aggregate the neighborhood message vectors and updated the node embedding of the graph. We trained the GNN [2] model to classify network flow in fourteen different classes on AWID3 dataset [8] and achieved an f1-score of 0.9987. Our model achieved an f1-score of 1 for 9 out of 14 classes in the case of class-wise performance.

References

[1] Arshid Ali, Shahtaj Shaukat, Muhammad Tayyab, Muazzam A Khan, Jan Sher Khan, Arshadk, Jawad Ahmad. “”Network Intrusion Detection Leveraging Machine Learning and Feature Selection””. In: ().

[2] Adam Pearce Benjamin Sanchez-Lengeling Emily Reif. *Graph neural network*. https://distill.pub/2021/gnn-intro///. [Online; accessed 5-Feb-2023]. 2021.

[3] Alper Ozcan Cagatay Catal Hakan Gunduz. “Malware Detection Based on Graph Attention Networks for Intelligent Transportation Systems”. In: *Results in Engineering* (2021).

[4] Chris Sinclair, Lyn Pierce, Sara Matzner. “”An Application of Machine Learning to Network Intrusion Detection””. In: ().

[5] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, Stefanos Gritzalis. “”Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset””. In: ().

[6] Danish Kaleem, K. Ferens. “”A Cognitive Multi-agent Model to Detect Malicious Threats””. In: ().

[7] Statista Research Department. *Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025*. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/. [Online; accessed 11-January-2023]. 2016.

[8] Efstratios Chatzoglou, Georgios Kambourakis, Constantinos Kolias. “”Empirical Evaluation of Attacks Against IEEE 802.11 Enterprise Networks: The AWID3 Dataset””. In: ().

[9] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, Petar Veličković. “Principal Neighbourhood Aggregation for Graph Nets”. In: ().

[10] H20.ai. *Target Encoding*. https://towardsdatascience.com/dealing-with-categorical-variables-by-using-target-encoder-a0f1733a4c69//. [Online; accessed 11-Febuary-2023]. 2021.

[11] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, George E. Dahl. “Neural Message Passing for Quantum Chemistry”. In: ().

[12] Kathryn Ann Tait, Jan Sher Khan , Fehaid Alqahtani, Awais Aziz Shah, Fadia Ali Khan , Mujeeb Ur Rehman. “Intrusion Detection using Machine Learning Techniques An Experimental Comparison”. In: ().

[13] Jure Leskovec. *node2vec: Scalable Feature Learning for Networks*. https://www.unb.ca/cic/datasets/android-botnet.html. [Online; accessed 19-Feb-2023].

[14] Muhamad Erza Aminanto, Rakyong Choi, Harry Chandra Tanuwidjaja, Paul D. Yoo, Kwangjo Kim, Member. “”Deep Abstraction and Weighted Feature Selection for Wi-Fi Impersonation Detection””. In: ().

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio. “Graph Attention Networks”. In: ().

[16] Prabhat Kumar, Govind P. Gupta, Rakesh Tripathi. “”An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for IoMT networks””. In: ().

[17] Soneila Khan, Adnan Akhunzada. “ ”A hybrid DL-driven intelligent SDN-enabled malware detection framework for Internet of Medical Things (IoMT)””. In: ().

[18] Canadian institute of Technology. *Android Botnet dataset*. https://www.unb.ca/cic/datasets/android-botnet.html. [Online; accessed 19-Feb-2023]. 2015.

[19] Canadian institute of Technology. *CICMalDroid 2020*. https://www.unb.ca/cic/datasets/maldroid-2020.html. [Online; accessed 19-Feb-2023]. 2020.

[20] Thomas N. Kipf, Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: ().

[21] Tim M. Booij, Irina Chiscop, Erik Meeuwissen, Nour Moustafa, Frank T. H. den Hartog. “”ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets””. In: ().

[22] William L. Hamilton, Rex Ying, Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: ().