Journal Pre-proof

Software Defined Network and Graph Neural Network-based Anomaly Detection Scheme for High Speed Networks

Archan Dadhania, Poojan Dave, Jitendra Bhatia, Rachana Mehta, Malaram Kumhar, Sudeep Tanwar, Abdulatif Alabdulatif

PII: S2772-9184(24)00045-6
DOI: https://doi.org/10.1016/j.csa.2024.100079
Reference: CSA 100079

To appear in: *Cyber Security and Applications*

Received date: 2 May 2024
Revised date: 10 August 2024
Accepted date: 29 November 2024

Please cite this article as: Archan Dadhania, Poojan Dave, Jitendra Bhatia, Rachana Mehta, Malaram Kumhar, Sudeep Tanwar, Abdulatif Alabdulatif, Software Defined Network and Graph Neural Network-based Anomaly Detection Scheme for High Speed Networks, *Cyber Security and Applications* (2024), doi: https://doi.org/10.1016/j.csa.2024.100079

Software Defined Network and Graph Neural Network-based Anomaly Detection Scheme for High Speed Networks

Archan Dadhania[a], Poojan Dave[b], Jitendra Bhatia*[a], Rachana Mehta[a], Malaram Kumhar[a], Sudeep Tanwar*[a], Abdulatif Alabdulatif[c]

*Corresponding author: Jitendra Bhatia (jitendra.bhatia@nirmauni.ac.in) and Sudeep Tanwar (sudeep.tanwar@nirmauni.ac.in)

[a]Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad, 382481, Gujarat, India
[b]Department of Computer Engineering, Silver Oak University, Ahmedabad, 382481, Gujarat, India
[c]Department of Computer Science, College of Computer, Qassim University, Buraidah, 52571, Saudi Arabia

Abstract

In recent years, the proliferation of Software-Defined Networking (SDN) has revolutionized network management and operation. However, with SDN's increased connectivity and dynamic nature, security threats like Denial-of-Service (DoS) attacks have also evolved, posing significant challenges to network administrators. This research uses the GraphSAGE algorithm to improve DoS attack detection using SDN and Graph Neural Network (GNN) to address the abovementioned problems. The study further explores the effectiveness of four anomaly detection techniques - Histogram-Based Outlier Score (HBOS), Cluster-Based Local Outlier Factor (CBLOF), Isolation Forest (IF), and Principal Component Analysis (PCA) - to identify and mitigate potential DoS attacks accurately. Through extensive experimentation and evaluation, the proposed framework achieves an better accuracy of detecting the anomalies than one without GraphSAGE model underscoring its potential to strengthen the security of SDN architectures against DoS attacks.

Keywords: Software Defined Network, Graph Neural Network, Anomaly Detection, Denial-of-Service, Control plane, Data plane, Network security

1. Introduction

A denial of service (DoS) attack is a simple yet devastating network attack. An attacker often creates susceptible nodes on the internet that are transformed into a botnet and uses these scattered hosts to generate a large number of packets with forged IP addresses to execute an access attack on the victim server. This kind of attack consumes all the resource of the victim's server and crash them so that the victim's server is unable to process the normal request, that is, denial of service [1] [2] [3].

DoS attack is widespread, and SDN is also affected by this kind of network attack [4, 5]. Nowadays, in the open-flow era, DoS attacks on the SDN mostly occur on the control plane. For example, an attacker generates a huge network of attack flows that do not match the flow table rules. Every attack sequence

Email addresses: 21mcei11@nirmauni.ac.in (Archan Dadhania), poojandave1504@gmail.com (Poojan Dave), jitendra.bhatia@nirmauni.ac.in (Jitendra Bhatia*), rachana.mehta@nirmauni.ac.in (Rachana Mehta), malaram.kumhar@nirmauni.ac.in (Malaram Kumhar), sudeep.tanwar@nirmauni.ac.in (Sudeep Tanwar*), ab.alabdulatif@qu.edu.sa (Abdulatif Alabdulatif)

will trigger the switch to transmit a message packet to the controller, potentially blocking the southbound interface and substantial resource consumption on the controller's end [6].

With the extensive use of data plane programmable SDN, data plane attacks have grown more prevalent [7]. For example, an attacker may infiltrate a host within the SDN and transform into another user in the network, wasting resources such as memory and compute power [4, 8]. This form of DoS attack, specifically flooding DoS attacks on the SDN. A similar kind of attack is a serious threat to the SDN network. They can be launched from anywhere worldwide and are difficult to defend against attacks.

Machine learning provides a novel technique for detecting network attacks in this circumstance. To discover the best outcomes, machine learning methods, particularly deep learning-based neural networks, may be applied [9]. Traditional ML techniques (Non-Graph) do not completely use the structure of network topological information in network flow data to identify complex network attacks [10]. We may utilize the GNN since it can identify entire network attacks with interconnected data structures.

Graph-based learning for big graph processing necessitates greater computation time and expenditure; hence, permutation-invariant algorithms are necessary. As a result of using the sampling and aggregation method, Graph Sample and Aggregated (GraphSAGE) may perform a quicker computing process, eliminating the need to rely on the fixed graph structure [11]. GraphSAGE is used for node embedding. However, in article [12] updated it to give an edge embedding solution to the network infiltration problem. The sampling procedures were the subject of GraphSAGE's major concern.

The model proposed in [13] use the edge features to improve GraphSAGE performance for network intrusion detection. In this approach, important information is embedded as edge features while the rest as node features. The model shows a good result for multiclass classification problem. However, the SDN framework and its underlying controller and data plane features are not explored to a significant extent. In model [14], GraphSAGE is learn the features of switches through host embedding, then the information learned is used for weight induced limiting scheme for penalizing misbehaving hosts. The GraphSAGE is used here at the controller and switch level for DDos attack consideration. The model proposed in [15], rearranges the node features to embed it into the graph learning for multi class flow detection, including the DOS Attack. The host information are infused into the node embedding and then fed to GraphSAGE for identifying intrusion detection. Here, Graph learning is constrained to the embedding learning of host and their interconnected edges only. While, the SDN specification are not exploited in full nature. The approach proposed in [16], authors have used GNN ensemble learning. GNN is a type of ML method to which uses various GNN models to enhance the detection accuracy. Recent solutions have used data-driven learning [17] Graph Convolution Network (GCN) [18] and Graph Attention Network (GAT) [19] to improve classification performance over GraphSAGE, but require a long computational time [20]. To achieve the optimal outcome, a causal weight was calculated between the neighbour target node and the label and then added to the original GraphSAGE sample. This causal sampling approach provides the highest performance increases and requires the least processing effort. Most of the existing approaches focuses on improving the node features and edge features of the graph to improve the intrusion detection. Also, the significant work focuses on the control plane of SDN. However, they don't exploit the intrinsic characteristic of underlying data plane of SDN. In our proposed work, we integrate SDN with the network benefit of GraphSAGE model for detecting the DoS attack. Our work signifies the importance of data plane in anomaly detection.

1.1. Research Contribution

The main research contribution of this paper is given below:

- We proposed a GNN-based Anomaly detection technique for network intrusion detection. The approach detects both edge features and topological patterns in the absence of label data.

- We uses the GraphSAGE model, redesigned DGI and applied four alternative anomaly detection methods. The techniques used for network intrusion detection are PCA based anomaly detection, IF, CBLOF based anomaly detection, and HBOS based anomaly detection.
- We applied the GraphSAGE model to Network Intrusion Detection System (NIDS) datasets to discover network anomalies. Through experimental examination, a considerable improvement over raw characteristics reveals its potential.
- We developed and configured the network using SDN and then convert it to a three-dimensional network state tensor that reflects the properties of all the switches at various moments. The GraphSAGE model is used to identify DoS attacks across the data plan. The data plan programmable SDN identifies DoS attacks and traces their movements.

The rest of the paper is organized as follows: Section 2 discusses the background of the proposed work. Section 3 presents the problem formulation and mathematical modeling for the proposed work. Section 4 describes the proposed framework. Section 5 presents experimentation, results, and analysis. Finally, Section 6 concludes the paper with future research directions.

2. Related Work

Anomaly detection is a critical aspect of network security. The major task of anomaly detection is to detect the abnormalities and deviations of a network from its regular behavior. There have been several anomaly detection approaches, like traditional network approaches, SDN-based approaches, ML-based approaches, and Hybrid approaches. However, dealing with complex and large data using these basic approaches can take time and effort. It demands the need for advanced and sophisticated approaches to detect anomalies. Deep learning-based techniques work well in this direction and help achieve better anomaly detection performance in SDN. Here, our major work is driven toward detecting DoS in SDN using the advanced DL-based approach. SDN has numerous applications in the data center scenario, offering benefits such as network programmability, flexibility, and centralized management. One crucial area where SDN can be particularly useful is detecting DoS attacks. DoS attacks involve many compromised devices flooding a target network or service with overwhelming traffic. Detecting and mitigating such attacks requires a comprehensive understanding of the network's traffic patterns and the ability to differentiate legitimate traffic from malicious traffic.

Denial of Service attacks are pervasive and heavily impact SDNs [4] [5]. The DoS attacks mostly target the control plane in SDN. For instance, many attack flows that don't match the flow table rules are generated, blocking the SDN framework's southbound interface and eventually affecting the resource consumption on the controller's side [6]. The prevalence of programmable SDN has also led to increased attacks on the data plane. For instance, one common attack is compromising the SDN host and using it as a bot to launch attacks against other users in the network. These attacks aim to exhaust resources like memory and computational power [4] [8]. These attacks pose a significant threat to SDN networks as they can originate from anywhere, globally, and are challenging to defend against.

Deep learning-based approaches are promising solutions for detecting DoS attacks in the current technological trends. Deep learning methods include Auto Encoder, Long Short Term Memory, Recurrent Neural Networks, Generative Adversarial Networks, and many others. These deep learning approaches fail to fully exploit the network topological structure and its information inherent in the network flow data [10]. To better analyze such networks and their inherent structure, Graph-based Neural Networks are preferable [9]. The GNNs have the strength to detect complete network attacks with interrelated data structures. GNN

also provides an advantage of peer learning to the network nodes.There have been several work carried out in the direction of incorporating graph knowledge for anomaly detection in IDS. A detailed overview of graph based anomaly detection is presented in [21] by the researchers. It is based on taxonomy of anomalous component present in the graph, the category of the graph, i.e. static or dynamic, types of graph techniques, type of anomaly and type of applications. The article [22], explores an application oriented graph representation learning for detection of network intrusion and host intrusion through usage of GNN based techniques. It also presents a graph structured data and its characteristics required for the same. It signifies the impact graph learning leaves without the need for external knowledge of domain. In article [23], the utility of anomaly detection based machine learning and deep learning model is presented with a view on smart surveillance, sensor networks and local area networks. Further it presents the graph based applications and use case in anomaly detection based on federated learning, auto encoders, embeddings of graph, transformers, signal processing and contrastive learning. The research work [24], presents a graph based anomaly detection and analytics techniques. It presents graph techniques based on convolution, auto encoders, attention mechanism and other approaches. These recent surveys, validates the significance of usage of graph learning in Intrusion detection and paves us the path for future research.

Incorporating graph-based neural network models demands large processing capabilities and computational time. It requires the operations to be permutation invariant. GraphSAGE is a GNN-based model that works in the same direction and provides a faster computational process through sampling and aggregation. It eliminates the need for fixed graph structure [11]. Here, we use the GraphSAGE-based model alongside SDN for anomaly detection. The general application of GraphSAGE is for node embedding, [12] modified it to edge embedding to solve the network intrusion problem. Several other GNN-based approaches have been validated for anomaly detection like data-driven learning based [17], GCN [18] and GAT [19] to improve classification performance above the GraphSAGE model. However, they are time-intensive. In article [20], a causal inference calculates the casual weight between the neighbour target node and the label. Then, those weights are used to the original GraphSAGE sampling to produce the optimal result. This causal sampling approach provides the highest performance increases and requires the least processing effort.

Graph-based learning for big graph processing necessitates greater computation time and expenditure; hence, permutation-invariant algorithms are necessary. By utilizing the GraphSAGE model in a DoS attack setup, we can create a graph representation of the network, where each node represents a device or network entity, and the edges represent connections or communication links [25]. This graph can capture the inherent relationships and dependencies within the network. The model can then be trained on this graph to learn representations of the network entities. These representations encode valuable information about the characteristics and behaviors of the devices and their relationships. The model can detect anomalies and identify potential DDoS attacks by analyzing the node representations and patterns. The generalized system model using GraphSAGE with SDN is represented in Figure 1.

3. Problem Formulation

The data plane in SDN networks is also vulnerable to DoS attacks. The data center is presently the most extensively utilized component in SDN, and DoS attacks within the data center are becoming increasingly widespread. In the current experimental scenario, the prime focus is on the data center in SDN architecture; the bot compromised attacker and victims reside in the same network. The attacker needs to learn about the network topology. It compromises data center hosts to launch a DoS attack. The attacker aims to deplete a server's resources or block a link in the data center, disrupting regular traffic and preventing the network request from being fulfilled within the time limit. When the attacker launches an attack on multiple hosts simultaneously, the attack flow enters the network from the edges and switches, eventually disturbing the network and overlapping the attack paths.

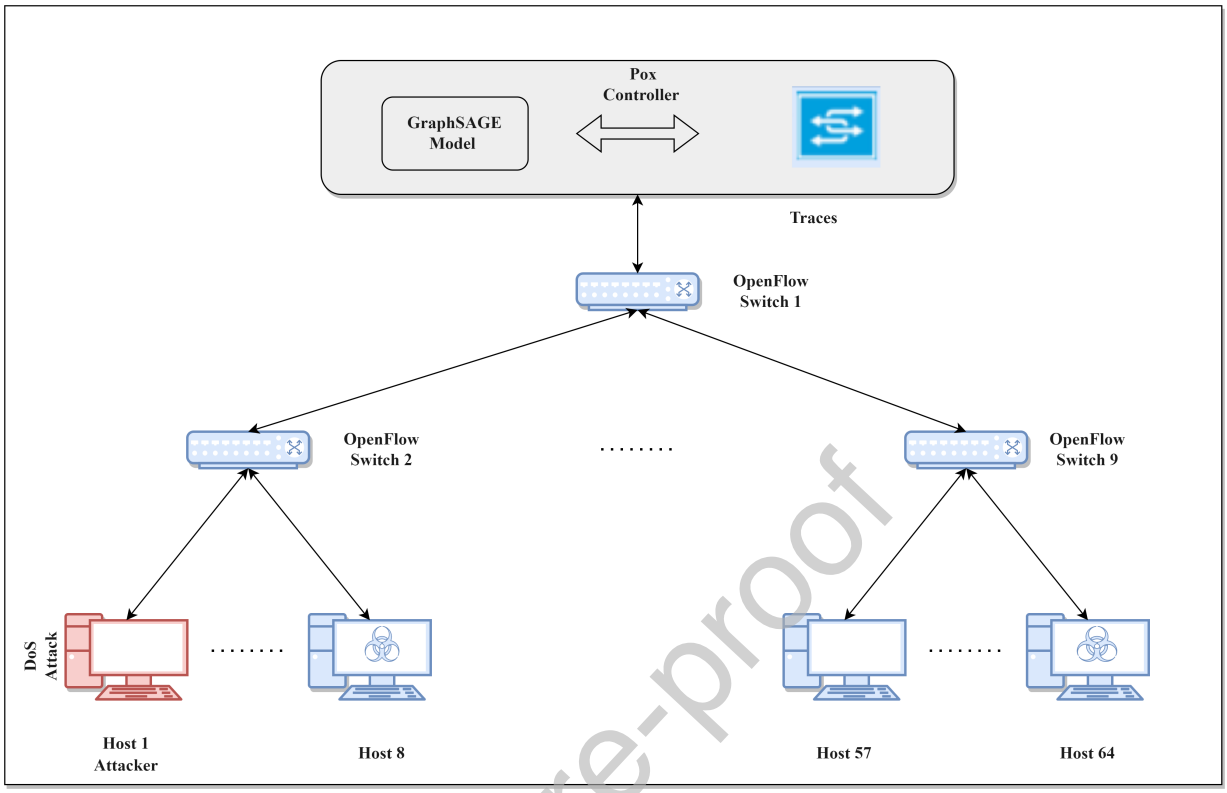


Figure 1: SDN-based GraphSAGE System Model

The SDN in the current model is represented using the graph. In graph $G = (V, E)$, the V represents the network nodes and devices, and E represents the edges between the controller, switch, and host device. The feature matrix linked with the network nodes is denoted by X, and each row X_i relates to the node's features $v_i \in V$. Let A be the adjacency matrix of the network graph G, with $A_{ij} = 1$ if there is a direct link between node v_i and v_j and $A_{ij} = 0$ otherwise. The ground truth labels Y indicate whether each node is anomalous (label 1) or normal (label 0). Network anomaly detection aims to develop a mapping $f : X \rightarrow Y$ that can properly categorize each network node as normal or anomalous based on its attributes and the structural information of the network.

We use a GNN-based model to capture the structural relationships in the network graph. To generate node embeddings, the GNN uses node characteristics and graph topology. SDN is used to regulate and monitor the network's behaviour in real time. SDN allows us to change network flows, redirect traffic, and respond to new anomalies. The SDN controller's actions are defined as a series of rules R that determine network policies depending on observed anomalies. The identification of anomalies is formalized as a binary classification job. To produce predictions, we integrate the learned GNN embeddings with the SDN rules R. This model combines GNN and SDN capabilities to give a comprehensive approach to anomaly detection, combining structural information and dynamic network control to improve accuracy and flexibility.

Architechture.jpg

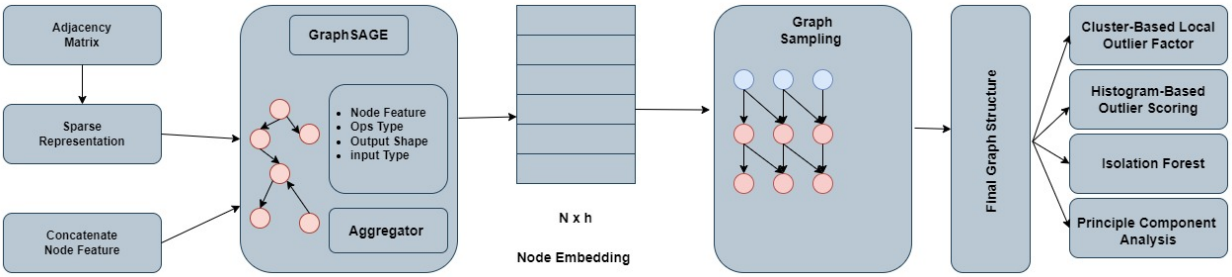


Figure 2: GraphSAGE Architecture

4. Proposed SDN based Framework

Our major focus is to detect the DoS attacks on the data plane, rather than the control plane in the usual scenario, using the SDN. It detects the attack's source and attack path to formalize a security strategy for the future occurrence of such malicious activities. The programmable switches are used with a fine-grained network state acquisition technique. This network state is used for developing a DoS detection model on top of GraphSAGE at the controller level. Figure 2 shows the architecture of GraphSAGE. A focused security strategy is built using the detected data and network traffic; this helps in transmitting correct information to the controller for further consideration. Network state awareness model, DoS attack detection model, and the implementation of a targeted defense strategy are shown in Figure 3. The subsequent part of this section explains the proposed model's working and operational algorithm.

The control level works as a binding entity for the entire architecture. Its flow table has access to the network state data and shared store data of each switch. The controller in our architecture is modified to incorporate the GraphSAGE model. In the context of a SDN, a graph is constructed with nodes representing entities like switches, devices, and subnets and edges representing the relationships or connections between them. Pertaining to the context shown in the Figure 1, the 9 OpenFlow switches and 64 Host System will work as nodes and edges will represent the connection and communication among them. The nodes in the graph represents the raw embeddings of the switch or host systems like node features, payload information, header, output, metadata and others. Using the GraphSAGE model the user neighbor selection is carried out which will enhance the user node feature through graph aggregation and update mechanism in iterative process. The learned embeddings are then fed to anomaly detection module for finding anomalous nodes. The model learning are then back propagated to host for model learning. The interception through node embeddings learned in GraphSAGE will allow us to find anomalous traffic in data plane directly through the neighbor learning process. This will also help to stop the incoming malicious traffic from network switch itself. The GraphSAGE model is trained using the adaptive interval sampling strategy to minimize additional CPU resources and bandwidth usage. The detection model and packet header are processed at the controller to give fine-grained network status. The network is modeled as a spatial-temporal graph on the controller side to maintain network state change. The collected data is translated into a two-dimensional tensor at the controller, comprising each switch's feature vectors at various instances. The obtained two-dimensional tensor and the network architecture adjacency matrix are inputs for the DoS attack detection model. Convolutions are applied to the tensor by the detection model to extract temporal features, capturing the temporal properties of the data. The adjacency matrix is also used to apply convolutions to the tensor to extract spatial features and investigate how the states of neighboring nodes affect one another spatially. The model locates switches that host DoS attack flows and establishes the path through which these attack

flows are propagated inside the network by analyzing and learning temporal and geographical aspects.

Architecture Implemented.png

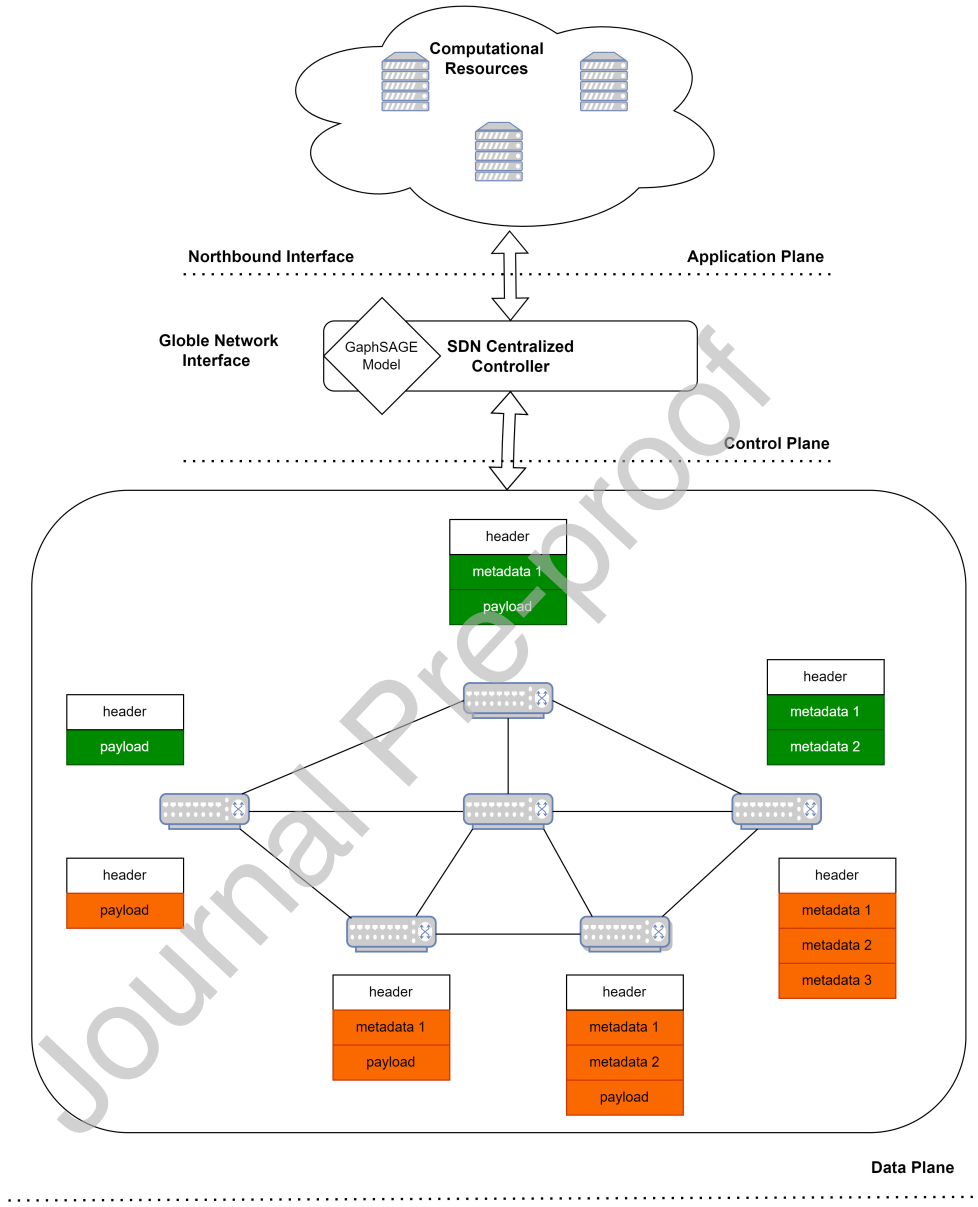


Figure 3: Proposed SDN Architecture

The network model used for development incorporates the Mininet network emulator along with an SDN controller. This combination provides a comprehensive overview of the network environment. Mininet is utilized as a network emulator, enabling the creation of a virtual network topology with virtual hosts, switches, and links. The SDN controller acts as the central management entity responsible for controlling the behavior of the virtual switches and facilitating communication between the emulated network components. Mininet and the SDN controller form a robust framework for simulating the attack detection model.

After training, the tuned GraphSAGE model was used to create edge embeddings for the training and testing graphs. Four methods were used to detect anomalies: PCA, IF, CBLOF, and HBOS. Each algorithm was trained unsupervised using the edge embeddings provided by the graph. The embeddings are sent into the anomaly detection model as input. Both infected and non-contaminated training samples were used in the studies.

Grid searches were performed for each detection method in each trial to guarantee optimal parameter optimization. Grid searches entailed experimenting with different values for the contamination parameter, the number of PCA components, the number of estimators for IF, the number of clusters for CBLOF, and the number of bins for HBOS. After the training phase, the edge embeddings from the test set were used for unsupervised model evaluation. The PCA algorithm used for finding anomalies in this study [26] is an improved variant of the traditional PCA algorithm, specifically designed for anomaly detection rather than dimensionality reduction. This method begins with a PCA reduction on a set of "normal" samples to generate a correlation matrix. This "normal" correlation matrix is a benchmark for the algorithm's following outlier detection phase. In other terms, an outlier is a sample that deviates considerably from the typical correlation matrix.

The IF anomaly detection technique [27] employs tree structures to isolate and detect anomalous samples. Samples isolated closer to the tree's root are identified as anomalies, while samples located deeper in the tree are considered "normal." An isolation forest is constructed to facilitate this process by utilizing an ensemble of trees. The IF algorithm is widely recognized for its effective and accurate identification of anomalies, achieving efficient performance.

The CBLOF algorithm [28] handles anomaly detection as a clustering-based issue. It groups samples and assigns an outlier factor based on the size of the cluster to which the sample belongs and the distance between the sample and the nearest cluster. This outlier factor measures the amount of deviation displayed by the sample, which is then used to determine whether or not it is an anomaly.

In contrast to the cluster-based approach of CBLOF, the HBOS algorithm [29] adopts a histogram-based methodology for anomaly detection. This technique involves constructing a univariate histogram for each feature, with each histogram consisting of multiple bins. Bin densities are derived by counting the occurrence of samples within each bin across all histograms. The calculated density values for each histogram are then utilized to determine the HBOS score, ultimately determining whether a sample is considered an outlier.

Algorithm 1 represents the proposed real-time anomaly detection model using the concept of GraphSAGE. Here, the SDN controller plane and data plane devices are already fixed, and the network data is gathered from those planes. The essential features of the preprocessed network data are then fed to the graph for representation purposes. The entire graph $G(N, E)$ represents the SDN using N network nodes and E edges. X represents the feature matrix of network nodes, where each signifies the feature of a single node. The feature matrix is populated using the information obtained from the dataset. Equivalent to a feature matrix is a Y label vector; it indicates where a specific node is anomalous or normal. All the information are then fed to the GraphSAGE model for training purpose. GraphSAGE generates the network node embeddings in the training phase, Z. An anomaly detection threshold, T, is defined using the normal distribution of embeddings with μ mean and σ standard deviation. The parameter α is the number of standard deviations selected for finding anomaly. It indicates the sensitivity of the anomaly detection carried out. If a value lies outside the range, then it is considered anomaly. Anomaly threshold T will work for defining whether a node is anomalous or not. After the model is trained, it is used for real-time network monitoring. The Graph Inference model activates and updates the graph representation during monitoring. The newly learned graph representation G_{new} and node features X_{new} are used to generate the updated and

new graph embeddings Z_{new}. The new node embeddings are compared with the learned threshold, T. All the nodes for which the anomaly is detected are stored in A. If the embeddings exceed the threshold, their labels, $Y[i].label$, are marked as anomalous. The updated network information of anomalous nodes is returned to the administrator for further consideration and action. The network monitoring is continuous and will continue until the network is active. The time complexity of GraphSAGE is mainly dependent on the number of nodes, N, number of edges, E, neighbors of the nodes, S and layer L. The neighbor sampling takes $O(N \times E \times L)$ time and aggregation of features takes $O(N \times E \times L \times S)$ time. So, the time complexity of training phase is $O(N \times E \times L \times S)$. The computational complexity of our algorithm is of polynomial order that will not hinder the scalability of the proposed system. Further, the complexity multiplies depending on the hidden features taken into consideration. The time for inference is linear to the number of nodes present in the environment as it involves the validation of whether a node is anomalous or not. The intrinsic nature and sampling strategies of GraphSAGE algorithm make it possible to work with large graphs.

Algorithm 1 Real-time Anomaly Detection using GraphSAGE in SDN

Require: Graph $G(N, E)$; Node Feature Matrix X; Anomalous Label Y

GraphSAGE Training and Anomaly Threshold:

$Z \leftarrow \text{GraphSAGE\_Train}(G, X, Y)$

$\mu \leftarrow \frac{1}{N} \sum_{i=1}^{N} Z_i$

$\sigma \leftarrow \sqrt{\frac{1}{N} \sum_{i=1}^{N} (Z_i - \mu)^2}$

$T \leftarrow \mu + \alpha \cdot \sigma$ ▷ α is a tunable parameter.

Real-time Monitoring:

while Operational(Network) **do**

$Z_{\text{new}} \leftarrow \text{GraphSAGE\_Inference}(G_{\text{new}}, X_{\text{new}})$

$A \leftarrow \{i \mid Z_{\text{new}_i} > T\}$

for i in A **do**

$G_{\text{new}}.Y[i].\text{label} \leftarrow$ 'Anomalous'

end for

end while

5. Result and Analysis

SDN and GNN have a substantial impact on bringing transformation capabilities into network anomaly detection. By offering centralized control and programmability, the model transforms how network management can be carried out. The proposed model shows the same capability for network intrusion detection (IDS). Two models have been compared here, one with and one without GraphSAGE, to show the impact of GNN in the IDS task. For the intrusion detection, we have used four techniques, PCA, IF, CBLOF and HBOS. This section provides the details of the experimental analysis carried out.

5.1. Anomaly Detection Techniques

This section discusses the techniques used to detect anomalies in the proposed model.

5.1.1. Principal Component Analysis

Principal component analysis technique is a extensively used in data analysis and ML for feature extraction and dimensionality reduction. It convert the high dimensional data into low dimensional data preserving

the variance in the original data. PCA is useful to balance the dataset when it is unbalanced in nature. PCA is widely used for detecting anomalies in the given datasets and helps to understand and visualize the existing representation of the data.

5.1.2. Isolation Forest

Isolation forest is a ML based anomaly detection technique, which efficiently identifies the anomalies from the dataset. Isolation forests are built using decision trees, like a random forest. It uses binary partitioning to quickly remove the anomalies from a dataset. It takes very less time to identifies outliers, making it suitable to detect anomalies in various applications. The advantages of IF includes its scalability, tolerance for outliers, easy to implement and handling high-dimensional data. It is used in many applications such as cybersecurity, finance and healthcare.

5.1.3. Histogram-Based Outlier Score

Histogram-Based Outlier Score is an unsupervised anomaly detection technique that uses histograms to find the outliers in the given dataset. It require complex computation to detect anomalies. It is very efficient and speedily find the outliers that makes it suitable for enormous datasets. HBOS create histograms for the feature distribution and then compute an outlier score using these histograms.

5.1.4. Cluster-Based Local Outlier Factor

Cluster-Based Local Outlier Factor identify the outliers using the combination of local distances to nearby clusters and the size of the data points clusters. It scales distances between observations and cluster centers by cluster sizes, allowing for flexibility in choosing clustering methods

5.2. Dataset

The dataset used for the experiment is NF-CSE-CIC-IDS2018-v2. This benchmark dataset is specially tailored for research in network intrusion detection, curated by the Canadian Institute for Cybersecurity (CIC). The dataset comprises network traffic data. It includes details of network flow, like source and destination IP addresses, port numbers, protocol types, packet sizes, duration, and labels. The labels present are ”normal” and ”intrusive”. It indicates whether a traffic flow is malicious or not. The parameters that we included for usage in models includes address of the source, port number, address of the destination, port number, packet size, and label indicating whether a message or payload is of DoS attack or not.

5.3. Tools & Technology

For the implementation of SDN and GNN, a wide range of tools and technologies are available, including POX, NOX, Ryu, OpenDaylight, Floodlight, PyTorch Geometric, Graph Nets, Deep Graph Library (DGL), and Spektral. We used POX, PyTorch Geometric, DGL, and Graph Nets. POX SDN controllers are open-source SDN controllers with a Python-based programming interface. PyTorch Geometric, DGL, and Graph Nets are GNN development libraries. PyTorch Geometric allows for large-scale graph processing flexibility. The details of these tools & technologies are mentioned in Table 1.

5.4. Parametric Setting & Evaluation Metrics

Table 2 show the parameters and hardware used in the SDN framework development. Here, we are using Hping3 for traffic generation and considering a DoS attack. In CBLOF, we carried out evaluation of cluster size of 2,3,5,7,9, and 10. For PCA and HBOS, the estimator parameter considered are 5,10,15,20,25 and 30. The contamination value is evaluated at 0.001, 0.01, 0.04, 0.05, 0.1, and 0.2 for PCA, HBOS and IF

Table 1: Tools and Technologies

Sr No.	Tool	Programming Language	Features
1	POX	Python	Highly customized, Easy to use
2	PyTorch Geometric	Python, PyTorch	Provide wide range of tools & functionality
3	Deep Graph Library	Python, PyTorch, TensorFlow, MxNet	Supports multiple backends, Provides wide range of tools and functionality
4	Graph Nets	Python, PyTorch	Easy to use, Simple interface

Table 2: Experimental Setup

Parameter	Value
Platform	Intel Core i3, 8 GB RAM
Operating System	Ubuntu 20.04 LTS
Type Of Traffic	TCP, UDP, ICMP
Traffic Generator	Hping3
Type Of Attack	DoS
Packet used	200, 600, 1000
Controller	Floodlight
Switch	OpenFlow Switch
Network Emulator	Mininet
Tool	PyTorch

model. The estimator for IF is kept at 20, 50, 100, and 150. Here, the contamination provides the proportion of outliers considered.

To evaluate anomaly detection models, there are several available metrics like Confusion Matrix, Receiver Operating Characteristic (ROC) Curve, Mean Square Error, Threshold based methods and statistical methods. Among which the accuracy and detection rate are significantly used in literature. So, to validate the proposed model, we have considered accuracy and detection rate.

5.4.1. Accuracy

Accuracy defines how accurately our model can detect network attacks. The equation 1 represents the formula for accuracy measurement.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

TP represents true positives, which refers to the number of switches correctly identified as containing attack traffic and containing attack traffic. FP stands for false positives, indicating the number of switches mistakenly identified as containing attack traffic but do not have any. TN represents true negatives, denoting the number of switches correctly identified as not containing attack traffic and, indeed, do not have any.

Finally, *FN* stands for false negatives, representing the number of switches mistakenly identified as not containing attack traffic but having attack traffic.

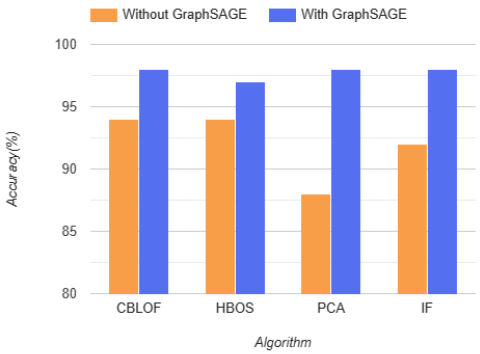
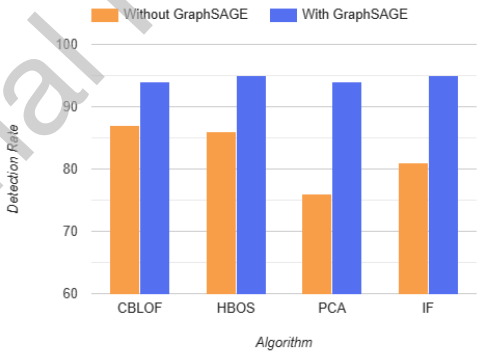


Figure 4: Accuracy Comparison

5.4.2. Detection Rate

The detection rate of a Denial-of-Service attack using the GraphSAGE Model is calculated using the equation 2.

$$\text{Detection Rate} = \frac{\text{Number of correctly detected DoS attacks}}{\text{Total number of DoS attacks}} \tag{2}$$



Rate.png

Figure 5: Detection Rate Comparison

Figure 4 and Figure 5 represent the result of accuracy and detection rate for four different variants of the SDN model with and without GraphSAGE. Here, the four variants are CBLOF, HBOS, PCA, and IF. The results are validated on the CICIIDS-2018 dataset with contamination. The models using GraphSAGE have better accuracy than the one without GraphSAGE; the accuracy obtained is 98%, 97%, 98% and 98% as compared to 94%, 94%, 88%, and 92% respectively for CBLOF, HBOS, PCA, and IF with GraphSAGE and without GraphSAGE. The anomaly detection rate shows a similar behavior; the detection rate obtained is 94%, 95%, 94%, and 95% as compared to 87%, 86%, 76%, and 81% respectively, for CBLOF, HBOS, PCA, and IF with GraphSAGE and without GraphSAGE. The overall accuracy of detection of anomalies with the GraphSAGE model and without GraphSAGE is 97% and 94%, respectively. Algorithms incorporating the

GraphSAGE model achieve better performance in terms of accuracy and anomaly detection. It is evident that including graph learning in intrusion detection task can boost performance significantly.

The data at the SDN controller is received from the data from the data plane. The GraphSAGE algorithm detects anomalies and protects against DoS attacks with the help of GNN and SDN. If the data received is manipulated or corrupted, it will significantly impact the performance and security of the network. The corrupted data can lead to wrong route selection, ultimately increasing latency. The corrupted data may affect the configuration and allow unauthorized access to the network.

6. Conclusion

In this research work, we embarked on a comprehensive exploration of enhancing Denial-of-Service attack detection within SDN environments. By harnessing the power of the GraphSAGE algorithm, we effectively captured the intricate relationships between network entities, enabling us to gain deeper insights into potential malicious activities. Moreover, we evaluated and compared the performances of three well-established anomaly detection methods - CBLOF, HBOS, PCA, and IF in identifying DoS attack patterns.

The experimental results validate the efficacy of our proposed framework, showcasing an impressive 94% detection rate and 97% accuracy. These outcomes underscore the significance of leveraging advanced machine learning techniques to fortify SDN architectures against the ever-evolving landscape of cyber threats. By achieving such high levels of accuracy and detection, we have made a substantial stride in improving the security posture of SDN networks.

In the proposed model we have used centralized SDN controller to manage the resources centrally. Though centralized controlling offers good network resources management, it also posses limitations in case of failure of SDN controller. To overcome this problem, multiple SDN controllers can be used to handle the issues may arise due to the failure of single SDN controller. In case of failure of one controller load of the failed controller can be transferred another controller. This will increase the scalability and the reliability of the proposed model. We need to balance between the overhead of managing the multiple SDN controllers and accuracy & detection rate to identify the DoS attacks.

However, we acknowledge that the threat landscape continues to evolve, and future work could delve into the adaptability of our approach to emerging attack vectors and the potential integration of real-time response mechanisms. Additionally, the scalability of the proposed solution across large-scale SDN deployments demands further investigation. The proposed work can be extended in future to solve the problem of single SDN controller failure issue by using multiple SDN controllers. Also other types of attacks can also be considered while implementing the proposed system using GNN.

References

[1] W. Dou, Q. Chen, and J. Chen, "A confidence-based filtering method for ddos attack defense in cloud environment," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1838–1850, 2013.

[2] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic DDoS defense," in *24th USENIX Security Symposium (USENIX Security 15)*, (Washington, D.C.), pp. 817–832, USENIX Association, 2015.

[3] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[4] J. Wang, R. Wen, J. Li, F. Yan, B. Zhao, and F. Yu, "Detecting and mitigating target link-flooding attacks using sdn," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 6, pp. 944–956, 2018.

[5] N. Arboleda, "AWS hit by DDoS attack dragging half of web down." `https://www.crn.com.au/news/aws-hit-by-ddos-attack-dragging-half-of-web-down-532842/`, 2020. [Online; accessed 02-June-2024].

[6] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE communications surveys & tutorials*, vol. 18, no. 1, pp. 602–622, 2015.

[7] M. Kumhar and J. Bhatia, "Software-defined networks-enabled fog computing for iot-based healthcare: Security, challenges and opportunities," *Security and Privacy*, vol. 6, no. 5, p. e291, 2023.

[8] T. Sasaki, C. Pappas, T. Lee, T. Hoefler, and A. Perrig, "Sdnsec: Forwarding accountability for the sdn data plane," in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–10, IEEE, 2016.

[9] W. Jiang, "Graph-based deep learning for communication networks: A survey," *Computer Communications*, vol. 185, pp. 40–54, 2022.

[10] J. Suárez-Varela, P. Almasan, M. Ferriol-Galmés, K. Rusek, F. Geyer, X. Cheng, X. Shi, S. Xiao, F. Scarselli, A. Cabellos-Aparicio, *et al.*, "Graph neural networks for communication networks: Context, use cases and opportunities," *IEEE Network*, 2022.

[11] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[12] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-graphsage: A graph neural network based intrusion detection system for iot," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, IEEE, 2022.

[13] H.-D. Le and M. Park, "Enhanced graphsage for multi-class intrusion detection," pp. 39–41, 01 2024.

[14] A. El Kamel, "A gnn-based rate limiting framework for ddos attack mitigation in multi-controller sdn," in *2023 IEEE Symposium on Computers and Communications (ISCC)*, pp. 893–896, IEEE, 2023.

[15] H.-D. Le and M. Park, "Enhancing multi-class attack detection in graph neural network through feature rearrangement," *Electronics*, vol. 13, no. 12, p. 2404, 2024.

[16] R. Abu Bakar, L. De Marinis, F. Cugini, and F. Paolucci, "Ftg-net-e: A hierarchical ensemble graph neural network for ddos attack detection," *Computer Networks*, vol. 250, p. 110508, 2024.

[17] J. Oh, K. Cho, and J. Bruna, "Advancing graphsage with a data-driven node sampling," *arXiv preprint arXiv:1904.12935*, 2019.

[18] S. Abu-El-Haija, A. Kapoor, B. Perozzi, and J. Lee, "N-gcn: Multi-scale graph convolution for semi-supervised node classification," in *uncertainty in artificial intelligence*, pp. 841–851, PMLR, 2020.

[19] P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Li, and Y. Bengio, "Graph attention networks," 2018.

[20] T. Zhang, H.-R. Shan, and M. A. Little, "Causal graphsage: A robust graph method for classification based on causal sampling," *Pattern Recognition*, vol. 128, p. 108696, 2022.

[21] P. B. Lamichhane and W. Eberle, "Anomaly detection in graph structured data: A survey," *arXiv preprint arXiv:2405.06172*, 2024.

[22] T. Bilot, N. El Madhoun, K. Al Agha, and A. Zouaoui, "Graph neural networks for intrusion detection: A survey," *IEEE Access*, vol. 11, pp. 49114–49139, 2023.

[23] A. D. Pazho, G. A. Noghre, A. A. Purkayastha, J. Vempati, O. Martin, and H. Tabkhi, "A survey of graph-based deep learning for anomaly detection in distributed systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 1, pp. 1–20, 2023.

[24] J. Ren, F. Xia, I. Lee, A. Noori Hoshyar, and C. Aggarwal, "Graph learning for anomaly analytics: Algorithms, applications, and challenges," *ACM Transactions on Intelligent Systems and Technology*, vol. 14, no. 2, pp. 1–29, 2023.

[25] A. Dhadhania, J. Bhatia, R. Mehta, S. Tanwar, R. Sharma, and A. Verma, "Unleashing the power of sdn and gnn for network anomaly detection: State-of-the-art, challenges, and future directions," *Security and Privacy*, vol. 7, no. 1, p. e337, 2024.

[26] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," tech. rep., Miami Univ Coral Gables Fl Dept of Electrical and Computer Engineering, 2003.

[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," pp. 413–422, 2008.

[28] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern recognition letters*, vol. 24, no. 9-10, pp. 1641–1650, 2003.

[29] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," *KI-2012: poster and demo track*, vol. 1, pp. 59–63, 2012.

There is no Conflict of Interest