CHAPTER 4
METHODOLOGY

This chapter outlines the hybrid methodology combining AI-driven techniques and dynamic fuzz testing to address DDoS attacks on IoT networks. It details the process of generating and preprocessing data, training a Graph Neural Network (GNN) model to classify network traffic, and validating its performance in simulated environments. The chapter also describes the mitigation strategy for blocking malicious IPs and refining network performance through iterative evaluations. Overall, it emphasizes a structured approach to enhancing IoT security against emerging threats.

4.1 Addressing IoT Security Using AI-Driven Techniques

The Internet of Things is an explosively growing domain in which smart home appliances, industrial sensors, and much more can interact and be used independently. Although this is very efficient and widely automated, the large-scale deployment of IoT devices triggers fresh network security concerns. Probably the most relevant threat here is DDoS attacks, where malicious actors try to overwhelm IoT networks with excessive traffic generation.

This project will propose a hybrid strategy combining fuzz testing, AI techniques, and network simulation for DDoS attack detection and mitigation. Here, GNN in fuzzy testing is used as the core for traffic pattern analysis as well as the identification of malicious activity in real-time.

4.2 Dynamic Fuzz Testing

4.2.1 Introduction to Dynamic Fuzz Testing

Fuzzing, or fuzz testing, is basically a testing technique that injects malformed or unexpected inputs into a system to expose potential weaknesses. Dynamic fuzz testing is a superior instrument that relates to investigating and finding the vulnerabilities in network security when abnormal traffic patterns emerge during DDoS attacks in IoT. Bugs, misconfigurations, and exploitable weaknesses are successfully detected in fuzz testing in real-world-like scenarios.

The project will simulate a wide array of IoT network traffic behaviors—from the simplest normal IoT communication to highly complex malicious DDoS attempts—by injecting unexpected data into the IoT ecosystem. This will bring to the project's attention vulnerabilities that it would otherwise have never known about had it taken the conventional approach to testing. The anomalies in the network traffic thus serve as a means of training AI models on distinct datasets for the improvement of detection capacity in relation to attack patterns.

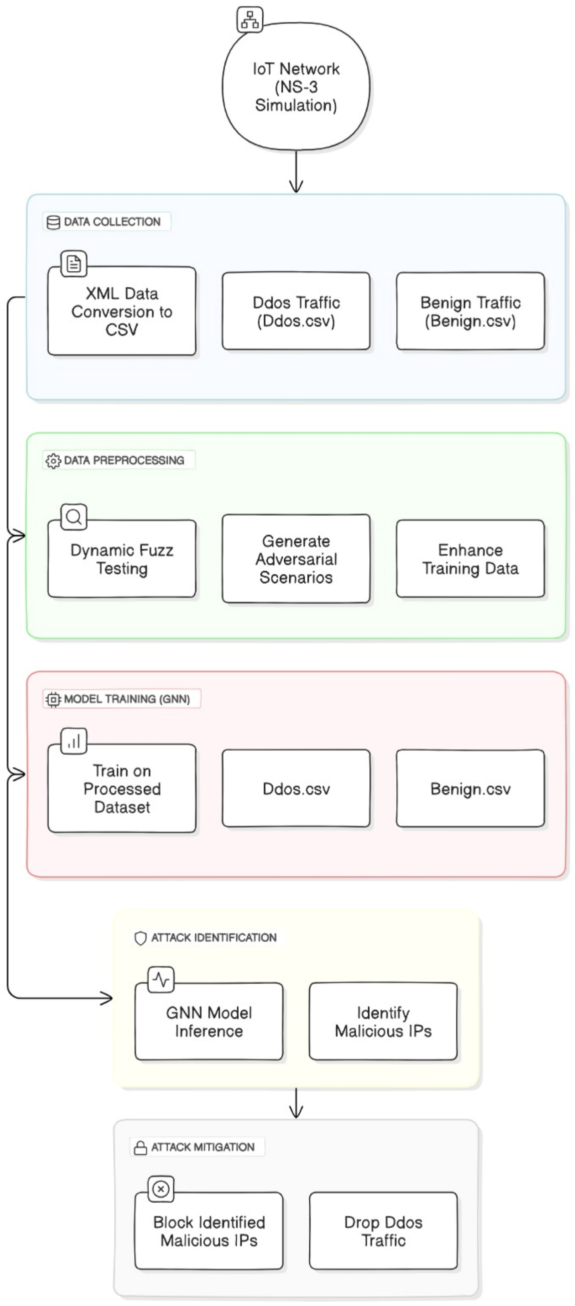

Fig. 4.1 Architecture Diagram

4.2.2 Data Generation Through Fuzz Testing

Data generation based on fuzz testing is important to give a precise representation of normal and malicious traffic. This process involves generating or synthesizing traffic through the help of fuzz testing tools and then configuring them in a manner that simulates potential attack vectors. Thus, the flow of packets generated through such a method brings out the closest imitation of common DDoS attacks, like SYN floods, UDP amplification, or botnet-induced flooding.

The simulation environment is set up to mirror a wide variety of network traffic behaviors. The resulting dataset captures multiple attack vectors, thereby subjecting the AI models that are trained on this data to various patterns. Traffic is labeled to distinguish between benign and malicious activity; thus, it creates a foundation for training machine learning models. Through such a well-balanced dataset, the system can generalize more effectively when detecting real-time attacks.

4.3 Dataset Preprocessing

4.3.1 XML to CSV from Simulation Data

Realistic IoT Network Simulations through NS3 feed the AI model with relevant data. The raw simulation data is produced in XML format, recording timestamps, source and destination IPs, packet size, etc. Though well-structured, XML isn't a good data source to be fed directly to train machine learning models. This custom script transforms the XML data into a simplified and more machine learning-friendly CSV format.

This step has the added advantage of handling large sets of data using frameworks like TensorFlow or PyTorch, which ensure the integrity of the data being processed. This allows easy manipulation of that data, feature extraction, and preprocessing for easy analysis.

4.3.2 Data Labeling and Balancing

Besides, the dataset needs to be properly labeled and balanced. That is the classification between benign traffic and attacks—a step necessary to prevent training a biased model towards one class. Moreover, having a balanced dataset avoids favoring benign or malicious traffic within an AI model, making it more robust to new data.

On the labeled dataset, further preprocessing is carried out in normalization of packet size, traffic frequency, and source/destination IP addresses. In case there is class imbalance, other methods like SMOTE (Synthetic Minority Over-sampling Technique) could be used. This way, both benign and attacking traffics are distributed equally in the training dataset.

4.4 Training AI Model

It is an AI project with a Graph Neural Network (GNN) as the type of training it does, which is supposed to classify IoT network traffic as being either benign or malicious. It is best for this task because it can consider relationships between nodes—take, for instance, the IoT devices—and learn to pick up complex traffic patterns that are hard to capture using traditional machine learning models.

4.4.1 Model Architecture

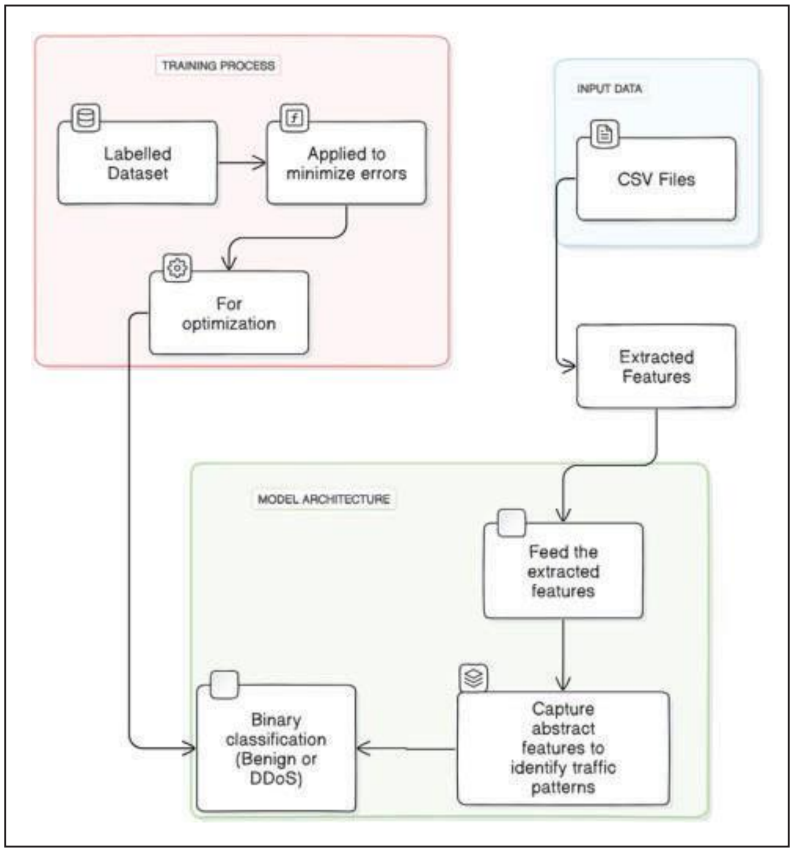
TRAINING PROCESS
Labelled Dataset
Applied to minimize errors
For optimization
INPUT DATA
CSV Files
Extracted Features
MODEL ARCHITECTURE
Feed the extracted features
Capture abstract features to identify traffic patterns
Binary classification (Benign or DDoS)

Fig. 4.2 GNN Model Architecture

The architecture of the GNN captures spatial and temporal dependencies in the traffic dataset. It accounts for interconnectivity as well as communication between different IoT devices, which are termed nodes. It captures how the patterns within a network change with time. In such a manner, through amalgamation of aspects of both spatial and temporal, GNN traces the path of the traffic flow and identifies anomalies that indicate DDoS attacks.

The architecture design is multi-layered graph convolutional networks (GCNs) that process traffic data and analyze the underlying patterns. One could include attention mechanisms that focus on relevant features in the dataset to make it even better at DDoS attack recognition.

4.4.2 Training Process and Hyperparameter Tuning

It performs training of the GNN model in iterations with the labeled dataset. Multiple epochs of training analyze the performance of the model in terms of accuracy, precision, recall, and F1-score. Through the process, hyperparameter tuning of learning rates, batch sizes, and the number of graph convolutional layers has been carried out. Cross-validation techniques are applied in the process of training to ensure the model generalizes well to unseen data. It is the most critical step to avoid overfitting and ensure that the model can pick up attacks or otherwise in a real scenario.

4.4.3 Validation

The trained model is then validated on freshly simulated environment traffic data. During the validation process, the model is tested against diverse network configurations, and its capability to detect DDoS attacks is examined in different network setups, thus generalizing well to new unseen data. The results are analyzed to improve the model, if needed, either by adjusting hyperparameters or the architecture.

4.5 Simulation Environment

4.5.1 Network Setup

The NS3 simulator offers a realistic testbed for the IoT network, which contains different IoT devices that can connect to a server via a central router. This simulation environment further generates both legitimate and malicious traffic, while the latter simulates a DDoS attack launched by bot nodes. The simulation ensures a realistic representation of IoT traffic, which allows the GNN model to learn from authentic traffic patterns.

4.5.2 Traffic Simulation

The capabilities of NS3's traffic generation capability are used to simulate TCP and UDP flows like real-world network traffic. On-Off traffic generators will simulate attacks such as DDoS, while BulkSend applications simulate normal flows in a network. The captured traffic is for a period and then written to files so that it can be used both to train and test the AI model.

4.5.3 Visualization with NetAnim

To further visualize the network behavior, NetAnim has been used by graphing the traffic flow and interaction between IoT devices. From this graphical output, a better view is obtained on the disturbance caused to normal traffic patterns due to DDoS attacks and how mitigating strategies have been designed to work in real-time. This validation is of paramount importance while testing the mitigation mechanisms in the system.

4.6 Mitigation Strategy

4.6.1 Blocking of Malicious IPs

The heart of the mitigation strategy is the dynamic blocking of identified malicious IP addresses due to the GNN model's detection. Upon identification of an attack, the NS3 simulation environment imposes router- level actions that block packets from such malicious IP addresses and prevent further disruption to the traffic. This packet filtering mechanism ensures that even during attacks, the network remains operational with minimal effects on legitimate traffic.

4.6.2 Evaluation and Refinement

The mitigation strategy is evaluated in terms of network performance before and after the introduction of blocking mechanisms. Metrics such as delivery ratio, latency, and throughput are analyzed to assess the effectiveness of the mitigation. Feedback from the simulation executions is used to iteratively refine the strategy so that it can handle a variety of attack scenarios.