Wireless Communication Security

Attack Detection and Mitigation System

Prepared by:
Mohamed Benabdelouahad
Ahmed Djalal Hacini
Hamza Benzaoui
Meriem Mebarek Mansouri

Institution:
The National Higher School of Artificial Intelligence (ENSIA)

April 17, 2025

Abstract

This project explores the application of artificial intelligence for enhancing security in wireless local area networks (WLANs). We propose a real-time intrusion detection and mitigation framework using machine learning models trained on traffic data. The system monitors live network packets, extracts relevant features, classifies them as benign or malicious, and takes appropriate countermeasures when threats are detected. By combining virtualized network environments with adaptive AI techniques, our approach provides a practical solution to secure WLAN infrastructures against a wide range of attacks, including ARP spoofing, denial-of-service (DoS), and rogue access points.

Keywords: WLAN Security, Intrusion Detection, Machine Learning, Wireless Attacks, Real-Time Monitoring, AI in Networking

Contents

1 Introduction **2**
- 1.1 Project Background . . . 2
- 1.2 Project Objectives . . . 2

2 System Design **4**
- 2.1 Overall Architecture . . . 4
- 2.2 Component Breakdown . . . 4

3 Implementation **6**
- 3.1 Dataset . . . 6
  - 3.1.1 IoT-23 Dataset . . . 6
  - 3.1.2 AWID Dataset . . . 7
  - 3.1.3 Manual Data Collection . . . 8
- 3.2 Data Processing Pipeline . . . 8
  - 3.2.1 Preprocessing and Feature Engineering Techniques . . . 8
  - 3.2.2 Feature Relevance to Threat Detection . . . 10
- 3.3 Model Development . . . 11
  - 3.3.1 Architecture Selection . . . 11
  - 3.3.2 Training Process . . . 11

4 Results and Evaluation **13**
- 4.1 Performance Metrics . . . 13
- 4.2 Comparison with Baselines . . . 14
- 4.3 Limitations . . . 16

5 Real Time Data Capturing & Mitigation **17**

6 Conclusion and Future Work **19**
- 6.1 Summary of Contributions . . . 19

Chapter 1

Introduction

1.1 Project Background

Wireless communication systems have become an integral part of modern digital infrastructure, supporting applications ranging from personal connectivity to industrial automation and smart cities. However, the inherent broadcast nature of wireless transmission makes these networks especially susceptible to a wide range of security threats. Traditional WLANs, despite advancements in encryption and authentication protocols, continue to face vulnerabilities such as unauthorized access, spoofing, jamming, and sophisticated denial-of-service (DoS) attacks. The rapid growth of Internet of Things (IoT) devices and the deployment of 5G technologies have further broadened the attack surface, introducing new challenges in traffic monitoring, anomaly detection, and threat mitigation.

In this context, there is a pressing need for more intelligent and adaptive security mechanisms capable of operating in real time. Conventional firewalls and signature-based intrusion detection systems often fall short in detecting novel or evolving attack patterns. As a result, research has increasingly focused on leveraging artificial intelligence (AI) and machine learning (ML) techniques to provide proactive and context-aware wireless network defense.

1.2 Project Objectives

This project proposes an AI-driven framework for enhancing the security of wireless local area networks (WLANs). The primary objective is to design and implement a system capable of monitoring wireless traffic in real time, extracting relevant features using advanced signal and protocol analysis techniques, and classifying network behavior using trained machine learning models.

Beyond detection, the system is designed to respond to malicious activity by automatically executing mitigation actions such as blocking unauthorized IPs, disabling rogue access points, and alerting administrators. The solution is deployed within a virtualized environment comprising dedicated attacker, victim, and monitoring components to simulate realistic WLAN attack scenarios and validate system performance. Through this work, we aim to demonstrate the effectiveness of AI techniques in providing scalable, autonomous, and precise protection for wireless communication infrastructures.

Chapter 2

System Design

2.1 Overall Architecture

The proposed wireless intrusion detection and response system is designed to operate in real time, providing continuous monitoring, intelligent detection, and autonomous mitigation of threats within WLAN environments. The architecture integrates several functional components in a modular pipeline.

The pipeline begins with live traffic capture from the WLAN interface, followed by preprocessing and feature extraction. The formatted data is then passed to a machine learning-based classifier to determine whether a packet or flow is malicious. If an attack is detected, the system automatically executes a mitigation procedure and logs the event. Simultaneously, the system updates a user-facing interface to inform the administrator of the current security status and historical insights.

2.2 Component Breakdown

The system is composed of four major components, each responsible for a specific function in the end-to-end threat detection and response workflow:

1. **Real-time Packet Capture and Preprocessing:** This module continuously captures network traffic using `tshark`, the CLI version of Wireshark. It filters and formats packet data into a structured format compatible with the AI model. The preprocessing script drops irrelevant fields, handles missing values, encodes categorical features, and ensures consistency with the model's expected input schema.

2. **AI Classification Model:** At the core of the system lies the trained machine learning model, which classifies incoming traffic instances into one of several predefined classes, including various attacks and benign

traffic. The model supports both multiclass detection and specialized binary classifiers for classes prone to false positives.

3. **Mitigation Module:** Upon detecting malicious activity, the mitigation module triggers appropriate countermeasures. These include blocking malicious IPs using `iptables`, disabling rogue access points via device APIs, or raising alerts. This automation ensures that threats are addressed swiftly without requiring manual intervention.

4. **User Interface and Dashboard:** The system includes a lightweight and intuitive graphical interface for real-time monitoring. It displays the number and type of detected threats, recent mitigation actions, and overall network health statistics. This dashboard enhances usability and provides transparency into system behavior.

Together, these components form a unified and reactive security framework capable of protecting WLAN environments from a wide range of attacks with minimal overhead and high interpretability.

Chapter 3

Implementation

3.1 Dataset

3.1.1 IoT-23 Dataset

The IoT-23 dataset, curated by the Stratosphere Laboratory, is a comprehensive collection of labeled network traffic traces from Internet of Things (IoT) environments. It contains a diverse range of attack scenarios conducted on actual IoT devices, offering researchers realistic examples of malicious traffic patterns. The dataset includes full PCAP files along with JSON metadata describing the behavior of devices under normal and adversarial conditions.

Notable attacks present in the IoT-23 dataset include:

- Denial-of-Service via SYN flooding (DOS_SYN_Hping)
- ARP Poisoning
- MQTT abuse (e.g., excessive MQTT_Publish)
- NMAP scanning variants such as UDP_SCAN, FIN_SCAN, OS_DETECTION, TCP_SCAN, and XMAS_TREE_SCAN
- Metasploit Brute Force attacks over SSH
- Distributed attacks such as DDOS_Slowloris and botnet control via protocols like ThingSpeak

While this dataset offered a strong foundation for experimentation, we encountered multiple limitations during its integration. Primarily, the dataset is designed for general-purpose IoT threat analysis, and many of the attack scenarios it includes were impractical to reproduce or simulate within the

constraints of our WLAN testbed. Furthermore, the data primarily captured flows and abstracted behavior rather than individual wireless packet-level interactions. This granularity mismatch limited our ability to apply the dataset directly to WLAN-specific intrusion detection. Consequently, we transitioned to a more WLAN-focused dataset for model training and validation.

3.1.2 AWID Dataset

To overcome the limitations of IoT-23, we adopted the AWID3 dataset, which is tailored specifically for WLAN security research. The AWID (Aegean Wireless Intrusion Dataset) collection includes real-world Wi-Fi traces recorded from a testbed that simulates typical 802.11 wireless network environments. Unlike IoT-23, AWID captures packet-level interactions, making it highly suitable for intrusion detection based on wireless protocol anomalies.

The dataset encompasses a rich variety of wireless attacks, such as:

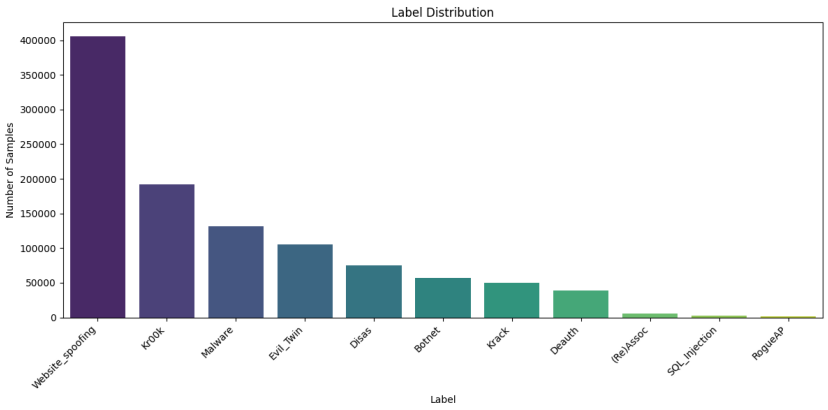

Figure 3.1: Distribution of labels across different WLAN attack types.

- **Kr00k**: A vulnerability exploiting disassociation and encryption flaws in WPA2.
- **Malware**: Malicious payloads delivered over wireless channels.
- **Evil Twin**: Rogue access points impersonating legitimate ones.
- **Disassociation (Disas)**: Forced disconnections of clients from access points.
- **Botnet**: Coordinated device compromise and remote control.

- **Krack**: Exploits against WPA2 4-way handshake.
- **Deauthentication (Deauth)**: Similar to Disas but more aggressive.
- **(Re)Association Attacks**: Exploiting management frame vulnerabilities.
- **SQL Injection**: Application-layer attack vector carried over wireless.
- **Rogue AP**: Unauthorized access points introduced into the network.

This dataset allowed us to effectively train our detection models on realistic wireless attack patterns, covering both management frame abuses and traffic-level anomalies.

3.1.3 Manual Data Collection

To augment the dataset and tailor it to our experimental setup, we performed manual simulations of several attack types, notably Deauthentication and SQL Injection. Using tools such as `aireplay-ng` and controlled application scripts, we launched these attacks in a controlled testbed consisting of virtual machines and real wireless interfaces. Network packets were captured using tools such as Scapy and Wireshark, providing raw data for training and evaluation of the model. This step ensured that our detection system could learn from traffic patterns that accurately reflected the behavior in our deployed WLAN scenario.

3.2 Data Processing Pipeline

3.2.1 Preprocessing and Feature Engineering Techniques

Effective preprocessing is a critical component in building a reliable intrusion detection system. In our pipeline, we began by removing non-informative columns that do not contribute to the classification task. These included indexing features such as `frame.number`, as well as administrative metadata not useful for model learning.

Furthermore, we addressed the issue of missing data, which was prevalent in the raw wireless packet captures. Several features had over 70% null values across the dataset, and such features were excluded to avoid introducing bias or reducing model robustness. This step significantly streamlined the feature space, improving training efficiency and model interpretability.

A central challenge in wireless attack detection lies in the extreme class imbalance. The number of normal traffic instances vastly outweighs that of malicious packets, especially for targeted attacks such as SQL injection or rogue access point insertion. To mitigate this imbalance, we adopted a dual strategy:

1. **Downsampling of the majority class**: We randomly sampled a subset of normal traffic to reduce its dominance while maintaining representative statistical characteristics.

2. **Class weighting during model training**: For algorithms like Random Forest, we enabled the `class_weight='balanced'` option to ensure that the minority classes received proportional emphasis in the loss function.

Categorical variables, particularly those related to frame control fields and WLAN subtypes, were transformed using one-hot encoding to convert them into a numerical format suitable for machine learning models. This process expanded the feature space but improved the model's ability to distinguish subtle patterns across packet types.

After extensive evaluation, we selected a refined set of features that contributed most significantly to classification performance. These included both temporal metrics and protocol-level details. The final set of features used in model training was as follows:

```
attacks_cols = [
    'frame.len', 'frame.time_delta', 'frame.time_delta_displayed',
    'frame.time_epoch', 'frame.time_relative',
    'radiotap.length', 'radiotap.timestamp.ts',
    'wlan.duration', 'wlan.fc.frag', 'wlan.fc.order', 'wlan.fc.moredata',
    'wlan.fc.protected', 'wlan.fc.pwrmgt', 'wlan.fc.type', 'wlan.fc.retry',
    'wlan.fc.subtype', 'wlan_radio.duration',
    'wlan_radio.data_rate', 'wlan_radio.signal_dbm', 'wlan.seq',
]
```

These features capture key aspects of wireless traffic such as timing behavior, frame structure, control flags, and physical-layer signal strength, enabling our model to distinguish malicious actions from benign traffic with improved accuracy.

3.2.2 Feature Relevance to Threat Detection

The selection of features plays a crucial role in the effectiveness of any intrusion detection model. The features used in our final model, as listed in the `attacks_cols` set, were chosen based on their ability to reveal structural, temporal, and physical-layer anomalies in WLAN traffic. These anomalies often serve as indicators of specific attack patterns.

- **Frame-level Timing and Size Metrics**: Features such as `frame.len`, `frame.time_delta`, `frame.time_delta_displayed`, `frame.time_epoch`, and `frame.time_relative` help detect anomalies in packet timing and transmission frequency. For example, attacks like **Disassociation (Disas)**, **Deauthentication (Deauth)**, or **SQL Injection** often introduce irregular timing patterns due to rapid packet bursts or response delays.

- **Radiotap Layer Features**: Attributes such as `radiotap.length`, `radiotap.timestamp.ts`, `wlan_radio.duration`, `wlan_radio.data_rate`, and `wlan_radio.signal_dbm` are essential for capturing the physical characteristics of transmitted frames. These are particularly useful for detecting attacks like **Kr00k** and **Krack**, which exploit vulnerabilities at the physical or encryption layer, often accompanied by specific signal strength or duration signatures.

- **Frame Control Flags**: Fields such as `wlan.fc.frag`, `wlan.fc.order`, `wlan.fc.moredata`, `wlan.fc.protected`, `wlan.fc.pwrmgt`, `wlan.fc.type`, `wlan.fc.retry`, and `wlan.fc.subtype` provide low-level insight into the 802.11 frame's purpose and behavior. Abnormal use or manipulation of these flags is indicative of attacks like **Evil Twin**, **(Re)Association**, and **Rogue Access Point (RogueAP)**, which often involve unauthorized frame injection or spoofing.

- **Sequence Numbers and Duration**: The feature `wlan.seq` can reveal frame spoofing or replay attacks by identifying discontinuities or repetitions in packet sequence numbers. Combined with `wlan.duration`, these features assist in identifying stealthy attacks such as **Botnet** communication or **Website Spoofing**, which might otherwise mimic normal traffic structure.

In summary, these features collectively enable the detection of a wide variety of wireless attacks by highlighting discrepancies in how frames are constructed, transmitted, and acknowledged. They allow the model to distinguish between benign and malicious behavior at multiple protocol layers,

from frame construction to temporal dynamics and physical transmission characteristics.

3.3 Model Development

3.3.1 Architecture Selection

Choosing the right machine learning architecture is essential for designing a reliable and accurate intrusion detection system. In this project, we experimented with three widely used models for tabular data: Random Forest, a simple feedforward Artificial Neural Network (ANN), and XGBoost, a gradient boosting algorithm known for its performance on structured data. Each model was evaluated using accuracy, precision, recall, and F1-score on both multiclass and binary attack detection tasks.

Among these, Random Forest consistently yielded the best performance, achieving over 98% accuracy on training, validation, and test splits. Its ensemble nature, robustness to overfitting, and ability to handle imbalanced datasets with internal class weighting made it an ideal fit for our problem. Additionally, Random Forest provided interpretable feature importance metrics, helping validate our feature engineering process. While ANN and XGBoost showed decent results, they required extensive hyperparameter tuning and still fell short of the accuracy and stability achieved by Random Forest, particularly on rare attack classes.

3.3.2 Training Process

Our initial training approach relied on a unified multiclass classification model trained on the full dataset. This model successfully learned to detect most attack classes with high accuracy. However, two classes—**Botnet** and **SQL Injection**—consistently exhibited elevated false positive rates, with many of their packets misclassified as normal. This behavior was likely due to the severe class imbalance, as normal packets vastly outnumbered these rarer attacks.

To address this, we introduced specialized models targeting the problematic classes. We downsampled the majority class (normal traffic) so that it was approximately three times the size of the minority classes (Botnet and SQL Injection). This balancing along with using specialized model strategy drastically improved classification performance: false positives for Botnet were reduced from 1,735 to 198, and for SQL Injection from 159 to just 32.

A noteworthy observation was the performance on the **RogueAP** class. Although not linearly separable, the class was fully distinguishable using Random Forest without signs of overfitting. This is likely due to the presence of strong, distinct patterns in the captured features—such as specific frame subtypes, transmission durations, or abnormal radiotap-level signal indicators—that set RogueAP traffic apart from other classes.

In summary, Random Forest emerged as the most effective model due to its generalization ability, resilience to imbalanced data, and strong out-of-the-box performance. Our training strategy, combining multiclass modeling with specialized remediation for hard-to-classify attack types

Chapter 4

Results and Evaluation

4.1 Performance Metrics

The primary model—Random Forest trained on the full multi-class dataset—demonstrated strong performance across most classes, especially in distinguishing common attack types and normal traffic. Below is the classification report for the test set:

Table 4.1: Classification Report for Multiclass Random Forest Model (Test Set)

Class	**Precision**	**Recall**	**F1-score**	**Support**
(Re)Assoc	0.98	0.98	0.98	1100
Botnet	0.92	0.85	0.88	11378
Deauth	0.98	0.98	0.98	7788
Disas	0.98	0.98	0.98	15026
Evil_Twin	0.98	0.97	0.98	20966
Kr00k	0.98	0.98	0.98	38361
Krack	0.98	0.97	0.98	9998
Malware	0.99	0.99	0.99	26322
Normal	0.99	0.99	0.99	5063368
RogueAP	0.99	0.99	0.99	262
SQL_Injection	0.95	0.70	0.80	526
Website_spoofing	0.99	0.99	0.99	81024
Accuracy			**0.99**	5276119
Macro Avg	0.98	0.94	0.96	5276119
Weighted Avg	0.99	0.99	0.99	5276119

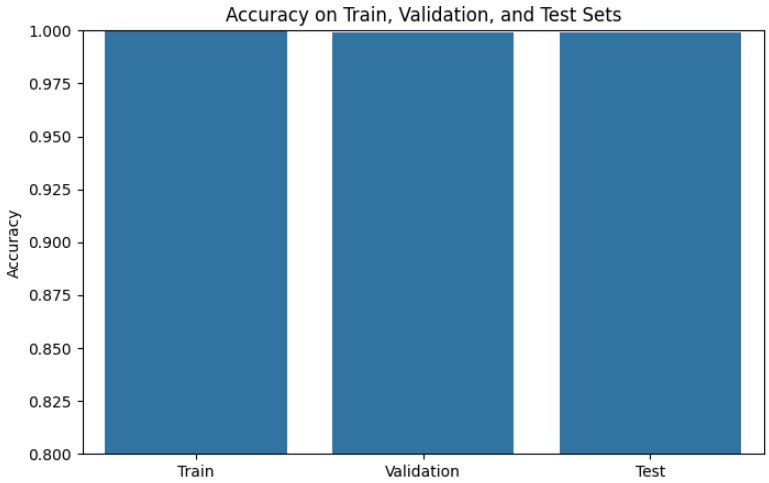

Figure 4.1: Training and validation accuracy for the multiclass model.

4.2 Comparison with Baselines

While baseline models such as XGBoost and ANN achieved acceptable performance, they were either less accurate or more prone to false positives, particularly on minority classes. Random Forest clearly outperformed them in generalization and stability. Additionally, when specialized models were trained for challenging classes like Botnet and SQL Injection, we saw a drastic improvement in precision and a significant reduction in false positives.

The confusion matrices in Figures 4.2 and 4.3 illustrate the substantial improvements achieved through targeted preprocessing and specialized modeling. Initially, both **Botnet** and **SQL Injection** classes suffered from high false positive rates when using a generic multiclass model. To address this, we implemented a multi-faceted approach:

- **Data Balancing:** We reduced the dominance of the normal class by downsampling its instances to approximately three times the number of Botnet and SQL Injection samples. This allowed the classifier to better focus on minority-class boundaries.
- **Class Weighting:** During training, the `class_weight='balanced'` parameter was used in Random Forest to emphasize the underrepresented classes in the loss function. This helped counteract the skewed class distribution without discarding valuable normal instances entirely.
- **Specialized Binary Models:** Rather than relying solely on the multiclass classifier, we trained dedicated binary classifiers specifically for

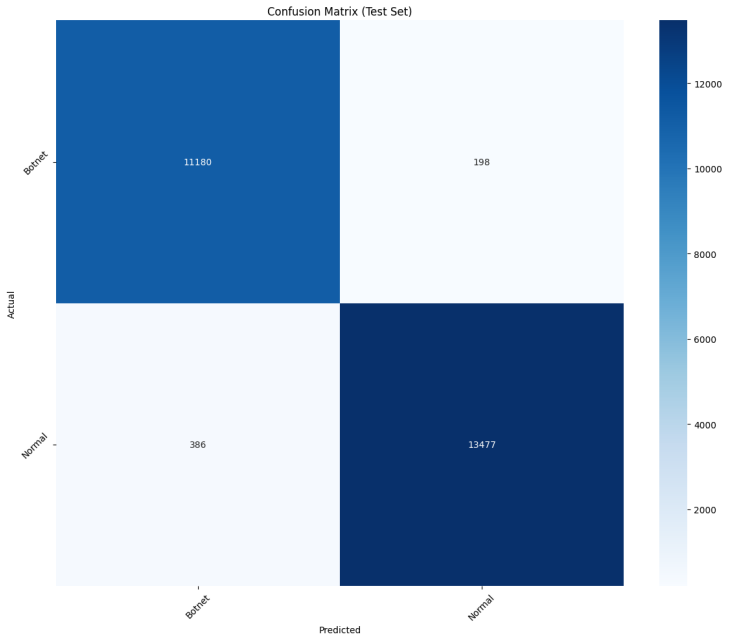

Figure 4.2: Confusion Matrix – Specialized Botnet Detection Model

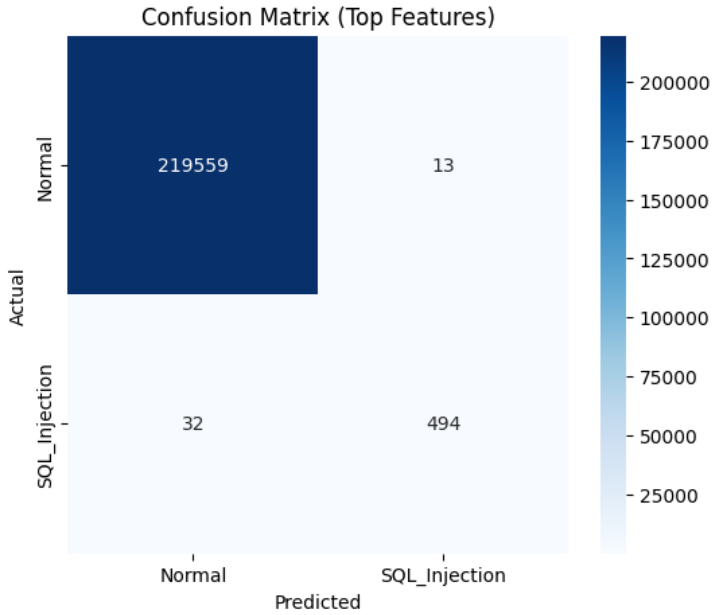

Figure 4.3: Confusion Matrix – Specialized SQL Injection Detection Model

Botnet and SQL Injection detection. These models had a simplified decision space and could focus on subtle distinctions between normal

and malicious traffic.

This strategy proved highly effective: false positives for Botnet were reduced from 1,735 to 198, while those for SQL Injection dropped from 159 to 32. These results validate the importance of tailoring the learning strategy to the characteristics of specific attack classes, especially in scenarios where attack behavior closely mimics benign traffic.

4.3 Limitations

Despite the high performance, several limitations remain. First, class imbalance still poses challenges for extremely rare attacks, requiring manual tuning or downsampling techniques that may not generalize well in a live environment. Second, the system's reliance on offline training prevents it from adapting quickly to novel attack vectors unless retrained. Finally, while Random Forest performed best overall, its inference time and memory footprint may be suboptimal for edge-based or low-resource environments, suggesting the need for future exploration of lightweight or online-learning models.

Chapter 5

Real Time Data Capturing & Mitigation

The real-time data acquisition process is handled by `tshark`, which continuously captures packets from the wireless interface. These packets are immediately passed through a lightweight preprocessing layer that reformats the data, drops irrelevant fields, encodes categorical features, and aligns the input with the trained AI model's expected structure. This enables low-latency threat classification at the packet or flow level.

Once a threat is detected by the AI model, the mitigation module initiates a predefined set of responses tailored to the detected attack type. This component operates autonomously and supports a wide range of countermeasures, including:

- **BLOCK / BLOCK_IP**: Prevents malicious devices from further communication by using `iptables` to block traffic from specific IP or MAC addresses.
- **RATE_LIMIT**: Throttles traffic from suspicious sources using `tc` to counteract flooding or DDoS attacks.
- **FLUSH_ARP / RESET_ARP**: Resets ARP cache to prevent ARP spoofing and man-in-the-middle attempts.
- **BLOCK_DEAUTH**: Uses tools such as `mdk3` or `iptables` to block deauthentication frames that cause wireless disconnections.
- **PATCH_SQL**: Applies input sanitization patches to backend services vulnerable to SQL injection.
- **HOSTAPD_CONFIG_PROTECTION**: Reconfigures `hostapd` to mitigate reassociation and rogue AP attacks.

These mitigation actions are linked to specific types of wireless attacks. For example, rogue access points are neutralized through MAC-based blocking and ARP flushing, while SQL injection vulnerabilities are addressed via backend patching. Deauthentication attacks are suppressed by blocking disconnection frames, ensuring uninterrupted client connectivity. DDoS and SYN flood attacks are mitigated through rate limiting and traffic control, preserving availability of services.

This layered, responsive mitigation strategy ensures the system not only detects but actively responds to threats in real time, significantly enhancing the resilience of WLAN infrastructures.

Chapter 6

Conclusion and Future Work

6.1 Summary of Contributions

This project presented a comprehensive AI-driven framework for real-time detection and mitigation of wireless network attacks. Our key contributions include:

- **End-to-End System Design:** We designed and implemented a complete wireless intrusion detection and response system integrating packet capture, preprocessing, machine learning-based classification, and automated mitigation.
- **Dataset Engineering and Modeling:** We utilized multiple datasets including IoT-23 and AWID, and supplemented them with manually captured traffic for underrepresented attack types. A hybrid modeling strategy combining multiclass and specialized binary classifiers significantly improved detection performance, especially for challenging classes like Botnet and SQL Injection.
- **High Accuracy and Low False Positives:** Through careful preprocessing, balancing, and class-specific models, we achieved over 98% accuracy across most classes with a substantial reduction in false positive rates.
- **Real-Time Mitigation Module:** We implemented a robust mitigation engine capable of executing targeted responses using tools like `iptables`, `tc`, `hostapd`, and `mdk3`. This enabled the system to automatically neutralize threats such as deauthentication, rogue APs, DDoS floods, and SQL injection attempts.
- **Interactive Monitoring Interface:** A lightweight dashboard was developed to display threat statistics, logs, and system actions in real

time, improving usability and administrative visibility.

These contributions demonstrate the viability of using AI to enhance WLAN security, offering a scalable and adaptive solution to a growing class of network threats.