ENSIA

Enhancing WLAN Security Using Artificial Intelligence

Wireless Communication Networks and Systems

17 April 2025

Team Members:
Yagoub Douaa Manel
Mers Wafaa
Cheboui Fatma Imene
Ikhelef Lyna
Salamani Amine
Chellal Abdelhak

1. Introduction

Wireless Local Area Networks (WLANs) are a vital component of modern digital infrastructure, enabling seamless connectivity across personal, corporate, and industrial environments. However, the openness and broadcast nature of WLANs make them inherently vulnerable to a wide range of security threats. These include attacks such as ARP spoofing, rogue access points (APs), deauthentication flooding, denial of service (DoS), and WPA2 key cracking.

Traditional WLAN security mechanisms—like WPA2/WPA3 encryption, MAC filtering, and static firewalls—are often limited in their adaptability. They rely on preconfigured rules or cryptographic techniques that fail to dynamically respond to evolving and sophisticated threats.

To overcome these limitations, this project introduces an Artificial Intelligence (AI)-driven security framework designed to enhance WLAN protection through intelligent threat detection and automated mitigation. By leveraging machine learning techniques on captured traffic data, our system can identify abnormal behavior, recognize known attack patterns, and respond in real time, creating a more robust and adaptive wireless security infrastructure.

2. Objectives

The primary objectives of this project are:

1. **Threat Detection:** Design machine learning models capable of identifying malicious behavior in WLAN traffic, including but not limited to ARP spoofing, rogue access points, and DoS attacks.

2. **User Behavior Profiling:** Analyze patterns of legitimate user activity to establish behavioral baselines and detect anomalies indicating compromised devices or unauthorized access.

3. **Automated Response:** Implement intelligent, real-time mitigation strategies that neutralize threats (e.g., blocking rogue devices or modifying access point settings) without disrupting legitimate traffic.

4. **Evaluation and Benchmarking:** Test the system's detection accuracy, false positive rate, and responsiveness in controlled environments and compare its performance to traditional static defenses.

3. Background and Related Work

WLAN security has traditionally relied on cryptographic protocols (e.g., WPA2, WPA3), network segmentation, and rule-based intrusion detection systems (IDS). While effective against basic attacks, these mechanisms struggle to detect zero-day threats, subtle anomalies, or adaptive malicious behavior.

Recent advancements in AI, particularly in machine learning and deep learning, have opened new avenues for network security. AI-based Intrusion Detection Systems (AI-IDS) can analyze high-dimensional data, detect deviations from expected behavior, and adapt

to new threats over time. Models such as Random Forests, Support Vector Machines, Autoencoders, and LSTMs have shown promise in prior work involving network traffic classification, anomaly detection, and cybersecurity event prediction.

Our work builds upon this foundation by applying machine learning techniques to WLAN-specific threats. We take a practical approach, not only by developing detection models, but also by integrating them into a real-world testbed with tools like Scapy, Wireshark, and Zeek, to simulate and respond to actual attacks.

4. System Architecture

The proposed AI-enhanced WLAN security system is organized into four logical layers:

1. Data Capture Layer

This layer is responsible for monitoring wireless traffic. It uses packet sniffing tools such as `Wireshark`, `Tcpdump`, and `Scapy` to collect traffic at both frame and IP levels. Data is stored in structured formats (e.g., CSV) for preprocessing and model input.

2. AI Processing Layer

Captured traffic is fed into machine learning pipelines. This layer hosts trained classifiers (e.g., Random Forests) and anomaly detection models. Features are extracted or approximated depending on available capture tools (e.g., Scapy, Zeek). Traffic is labeled in real-time, enabling live detection of threats.

3. User Interface Layer

This layer provides visualization and management. Logs, alerts, and performance metrics are displayed via dashboards (e.g., `StreamLit`) or CLI tools. It enables administrators to monitor network status and intervention results.

The layered design ensures modularity, allowing each component to be independently tested, updated, or replaced. This architecture not only supports offline analysis but also supports live threat monitoring in practical deployments.

5. Dataset and Preprocessing

To enable effective machine learning-based detection of WLAN attacks, we employed a multi-dataset strategy. Each dataset addresses a specific attack vector or system requirement (e.g., real-time compatibility). Extensive preprocessing was performed to clean, normalize, and select features suitable for the task. The following subsections outline the rationale, methodology, and outcomes for each dataset used.

Repository Link: All source code, preprocessing scripts, and model notebooks can be found in the following GitHub repository: ***https://github.com/Manelygb/Network-Intrusion-Attacks-Classification.git***

1. AWID Dataset – Comprehensive Multiclass Model

- **Source:** Original AWID website
- **Purpose:** This was the primary dataset for training a multiclass classifier to distinguish between normal traffic and multiple WLAN attacks (Deauthentication, ARP Spoofing, and Rogue AP).
- **Challenge:** The dataset was requested from this original website, which took much time to be obtained. The size of the dataset is really huge. Highly imbalanced — the vast majority of records represented normal traffic.
- **Initial Shape:** Over 1.2 million rows and 254 features, many with over 10% missing data, and some with over 90

Preprocessing Steps:

- Removed all features with more than 10% NaN values.
- Dropped constant-value columns like MAC-specific identifiers (`wlan.ra`) to avoid overfitting.
- Ranked features by importance using Random Forest feature importances in order to eliminate more features, and make the model lighter.
- **Retained only 23 features that contributed meaningfully to classification.**

Result: This cleaned version served as the foundation for a highly accurate multiclass classifier capable of distinguishing among four traffic types.

2. AWID Dataset – Scapy-Compatible Model

- **Purpose:** In real-time system deployment using Scapy, most features retained in the previous step were not available through standard capture. Thus, a new model was trained on a subset of features that Scapy could reliably extract.
- **Selected Features:** `frame.len`, `tcp.srcport`, `tcp.dstport`, `ip.ttl`, `tcp.flags.syn`, `tcp.flags.ack`, `tcp.flags.fin`, `tcp.flags.reset`, `tcp.time_relative`
- **Tradeoff:** Sacrificed slight accuracy for practicality.

Preprocessing:

- Retrained the model with the limited feature set. Ignored relevant columns that contained more than 80
- Removed rows with heavy missing values.
- Applied imputation where appropriate for numeric features.

Outcome: Supported live deployment, making it the preferred model for Scapy-based real-time detection.

3. Kaggle ARP Spoofing Dataset – Lightweight Binary Classifier

- **Source:** Kaggle ARP Spoofing Based MITM Attack Dataset
- **Purpose:** Complementary model for binary classification (normal vs ARP spoofing). Selected because the dataset was smaller, balanced, and easier to preprocess.
- **Shape:** 74,343 rows and 79 features.

Why we used it:

- Easy to experiment with model performance on focused attack type.
- Balanced class distribution allowed clean evaluation of precision, recall, and F1.
- Model trained quickly and gave useful insights for ARP detection via feature correlation.

Preprocessing: Standard numeric feature cleaning, label encoding.

4. CSE-CIC-IDS2018 – WPA2 Key Cracking Proxy Detection

- **Source:** CSE-CIC-IDS 2018
- **Motivation:** No public dataset for WPA2 key cracking was available. We used brute-force login attacks (FTP, SSH) as a behavioral proxy to model similar attack dynamics.
- **Attack Logic:** WPA2 cracking involves repeated handshake attempts and dictionary-based replay, resembling brute-force behavior in traffic frequency and packet timing.

Selected Features:

- `Tot Fwd Pkts`, `TotLen Fwd Pkts`, `Fwd Pkt Len Max/Min/Mean`, `Pkt Len Var`, `Flow IAT Std/Min`, `Fwd Seg Size Min`, `Fwd Act Data Pkts`

Approximation Mapping: Captured traffic during simulated WPA2 cracking (EAPOL handshakes, repeated deauths) was mapped to the above features to mimic brute-force flows. Details of the map are explained in Table 1.

5. CIC-IDS2017 – DDoS Detection Using Zeek Logs

- **Source:** CSE-CIC-IDS 2017
- **Motivation:** To detect network-layer DDoS attacks by analyzing flow features extracted via Zeek logs in real time. This would serve as a generalization for the DDoS attack detection.
- **Layer:** Focused on layer 3/4 detection.
- **Tools:** Zeek for live flow feature capture.

Selected Features:

- `Flow Duration`, `Total Fwd Packets`, `Total Bwd Packets`, `Total Length of Fwd/Bwd Packets`, `Destination Port`

Model Feature	How to Extract or Approximate in WPA2 Cracking
`Fwd Seg Size Min`	Use EAPOL packet sizes (usually fixed size) or minimum size of outbound handshake packets.
`Tot Fwd Pkts`	Count packets from attacker (to AP or victim) during the attack.
`Fwd Act Data Pkts`	Count EAPOL packets or specific replay attempts from PCAP.
`TotLen Fwd Pkts`	Sum of sizes of forward packets (e.g., handshake + deauth).
`Flow IAT Min/Std`	Inter-arrival time between handshake attempts (replays).
`Fwd Pkt Len Max`	Max packet size sent by attacker during cracking.
`Pkt Len Var`	Variance in packet sizes (if there is randomness in replay).
`Fwd Pkt Len Mean`	Mean of packet lengths (probably low for EAPOL frames).

Table 1: Approximation of model features for WPA2 key cracking using traffic analysis

6. AI Modeling and Implementation

The AI-based detection system was implemented through a series of supervised machine learning models, each designed to target a specific WLAN attack or use-case constraint (e.g., real-time traffic capture). Random Forest classifiers were chosen across all models due to their excellent performance, robustness to overfitting, interpretability, and low computational cost—making them well-suited for real-time security systems.

The following subsections detail each model's motivation, training process, and final performance.

1. Full AWID-Based Multiclass Classifier

Goal: Create a high-accuracy, multiclass classifier to detect several WLAN attacks: Deauthentication, Rogue AP, and ARP Spoofing.

Modeling Details:

- Dataset: Preprocessed AWID dataset with 23 selected features.
- Target classes: `Normal`, `Deauth_Attack`, `Rogue_AP_Attack`, `Website_Spoofing_Attack`.
- Final features included temporal (e.g., `frame.time_epoch`), physical layer (e.g., `wlan_radio.signal_dbm`), and control information (e.g., `wlan.fc.subtype`).

Performance Metrics:

- **Train Accuracy:** 99.97%
- **Test Accuracy:** 99.87%

- **Macro F1-Score:** ≈ 1.00

Deployment Role: Serves as the most precise model when complete packet details are available. Ideal for offline or high-fidelity environments.

2. AWID Scapy-Compatible Model (Lightweight Deployment)

Goal: Enable live packet classification using only features accessible from Scapy in real time.

Why This Model:

- Real-time compatibility: Limited to features extractable using standard socket-level packet capture.
- Maintains high accuracy while using significantly fewer features.

Modeling Notes:

- Extensive preprocessing and imputation due to column variability and NaNs.
- Performance slightly lower than the full model but extremely efficient.

Performance Metrics:

- **Accuracy:** 99%
- **Macro F1-Score:** 0.9835

Deployment Role: This is the primary model embedded in the live WLAN monitoring system.

3. ARP Spoofing Detection (Kaggle Dataset)

Goal: Investigate a minimal dataset focused on ARP spoofing with balanced classes and fast training times.

Why This Model:

- The dataset is lightweight and class-balanced, making it ideal for controlled experimentation.
- Binary classification allowed precise tuning of detection sensitivity.

Modeling Details:

- Model: Random Forest with standard hyperparameters.
- Train-test split with stratified sampling to maintain class proportions.

Performance Metrics:

- **Accuracy:** 94.7%
- **Macro F1-Score:** 0.95

Deployment Role: Serves as a reference or fallback model in case of high system load or for focused ARP spoofing detection modules.

4. WPA2 Key Cracking Proxy Model (CSE-CIC-IDS2018)

Goal: Approximate WPA2 cracking behavior using a proxy model trained on brute-force login attacks (FTP/SSH).

Justification:

- No labeled dataset exists for WPA2 cracking.
- Brute-force attacks on login protocols mimic similar traits: repeated attempts, time-based anomalies, consistent traffic signatures.

Modeling Strategy:

- Used CSE-CIC-IDS2018 dataset's Brute Force attack labels.
- Mapped WPA2 traffic characteristics (e.g., EAPOL handshakes) to features like `Tot Fwd Pkts`, `Fwd Pkt Len Mean`, `Flow IAT Std`.

Performance Metrics:

- **Accuracy:** 99.99%
- **Confusion Matrix:** Excellent class separation

Deployment Role: Active when WPA2 cracking signatures are detected. Anomaly-based pre-filtering can trigger this model for deep analysis.

5. DDoS Detection Model (CIC-IDS2017 + Zeek)

Goal: Detect layer 3/4 DDoS attacks based on flow statistics extracted from Zeek logs.

Why Zeek:

- Lightweight and highly scalable flow-level logging engine.
- Compatible with network-wide monitoring.

Modeling Approach:

- Random Forest trained to classify benign vs DDoS.
- Features directly extracted from Zeek '.log' files and normalized.

Performance Metrics:

- **Accuracy:** 100%
- **Macro F1-Score:** 0.9996

Deployment Role: Passive flow-level detector running alongside the real-time packet inspection layer for broader monitoring.

7. Testbed and Environment Setup

8. Attack Simulation Scenarios

1. ARP Spoofing Attack: Man-in-the-Middle via ARP Table Poisoning

In this simulation, we demonstrated a classic ARP spoofing (Man-in-the-Middle) attack using two virtual machines—Kali Linux as the attacker and Ubuntu as the victim—within a controlled NAT network setup. This configuration ensured traffic isolation and repeatable testing conditions. The attack was executed using the 'arpspoof' tool from the 'dsniff' suite, which enables the attacker to poison ARP tables by sending forged ARP replies on the network.

After verifying network connectivity between both machines and enabling IP forwarding on Kali to allow traffic relaying, the spoofing was launched in two terminals:

- In Terminal 1, the attacker spoofed the victim into believing it was the gateway: `arpspoof -i eth0 -t <victim-ip> <gateway-ip>`
- In Terminal 2, the attacker spoofed the gateway into believing it was the victim: `arpspoof -i eth0 -t <gateway-ip> <victim-ip>`

This bi-directional spoofing successfully inserted the attacker into the communication path, allowing full interception and forwarding of packets between the two endpoints without disrupting connectivity. The success of the attack was confirmed by inspecting the victim's ARP table via the `arp -a` command, where the legitimate MAC address of the gateway was replaced by that of the Kali machine.

During the attack, 'Scapy' was used to capture live traffic on the attacker's side. The system's AI-enhanced detection pipeline extracted key flow-level features from the captured packets, such as frame length, TCP port numbers, TTL values, and TCP flag statuses. These features were then passed through a preprocessing module that performed feature engineering and formatting before routing the data to the ARP spoofing detection model. This model, embedded in the system architecture, analyzed the traffic in real time and triggered appropriate responses based on the detection logic.

This scenario validated the effectiveness of the system's architecture in handling real-world threats—seamlessly moving from traffic capture to live inference under active ARP spoofing conditions.

2. Deauthentication Flood Attack: Simulated Packet Injection and Flow Monitoring

In this scenario, we explored the simulation of a Deauthentication (Deauth) flood attack—a well-known Denial of Service (DoS) technique targeting WLAN environments. These attacks aim to forcibly disconnect legitimate clients from their access points by exploiting the unprotected nature of 802.11 management frames. Under normal circumstances, such attacks are carried out using specialized tools like aireplay-ng or mdk3, which rely on a wireless interface operating in monitor mode.

However, the wireless adapters we tested were not compatible with our system's operating environment and were difficult to integrate properly, preventing us from executing the attack over the air. To overcome this, we used Scapy to craft and simulate deauth traffic programmatically. The generated packets mimicked real deauthentication frames by setting the appropriate fields in the 802.11 protocol headers. Specifically, we forged packets with the attacker posing as the access point, repeatedly sending deauthentication messages to a targeted client MAC address. These packets used the typical "reason code 7" to indicate a disconnection due to a non-associated station—commonly seen in malicious disconnection attempts.

While the packets could not be broadcast on the wireless channel, we injected them into the virtual network environment to simulate the high-frequency nature and behavioral footprint of a real attack.

To monitor and analyze this synthetic attack, we deployed Zeek as a flow-level monitoring solution. Zeek captured statistical features such as packet frequency, inter-arrival times, and destination address patterns, logging them in structured formats. These logs were then processed by the system's AI pipeline. After undergoing preprocessing and feature engineering, the data was passed to the DDoS detection model—originally trained on CIC-IDS2017 traffic. Although the source of the traffic was synthetic, its anomalous signature closely resembled that of a real deauth flood or low-rate DoS attack, making it suitable for evaluation within our detection framework.

This setup demonstrated the system's flexibility in recognizing and processing malicious behavior, even in constrained environments where hardware-based simulations were not feasible.

3. Simulating a WPA EAPOL-Key Injection over Ethernet

In this experiment, we simulated the injection of a fake WPA EAPOL-Key message over a virtual Ethernet link. The objective was to reproduce the behavior of a handshake or key exchange packet as observed in WPA/WPA2 networks—without needing a wireless adapter operating in monitor mode.

To achieve this, we developed a Python script using the Scapy library. The script defined a custom EAPOL-Key layer, manually specifying the structure and fields of a WPA key frame. This included important elements such as the replay counter, nonce, MIC, and descriptor type—all modeled to mimic Message 1 of the WPA 4-way handshake.

Since we were sending packets over Ethernet (not Wi-Fi), the script first attempted to resolve the MAC address of the target IP address using ARP. If successful, the EAPOL-Key packet was directed to the discovered MAC. Otherwise, it defaulted to sending the frame to the broadcast address `ff:ff:ff:ff:ff:ff`.

Once the packet was crafted and addressed, it was injected into the network using Scapy's `send()` function on a VirtualBox bridged interface. The target machine—in this case, a macOS host—used `scapy` to capture and inspect the frame. The EAPOL-Key message appeared as expected at Layer 2, complete with the custom-defined fields and appropriate EtherType (0x888e).

This approach allowed us to simulate part of the WPA handshake in a fully virtualized, hardware-independent setup. It also provided useful data for flow-based monitoring and potential anomaly detection models.

Key points:

- No monitor-mode adapter or wireless driver was required.
- ARP was used to dynamically resolve the MAC of the target IP.
- The EAPOL-Key frame was constructed and sent using pure Python/Scapy logic.

4. Simulating a Rogue Access Point (Beacon Flood) over Ethernet

In this scenario, we simulated a Rogue Access Point (Rogue-AP) beacon flood—a type of wireless disruption attack where fake beacon frames are broadcast to confuse or overwhelm client devices. Traditionally, such attacks require wireless adapters in monitor mode that can inject raw 802.11 management frames into the air.

However, since our setup did not include compatible wireless hardware, we opted to simulate this attack entirely over Ethernet using a virtualized lab environment. The goal was to replicate the frame structure and timing characteristics of a beacon flood without transmitting anything over real radio channels.

Using Scapy, we crafted a fake 802.11 beacon frame by building a custom Dot11 packet that included an SSID element (set to "RogueAP"), capability fields ('ESS+privacy'), and supported rates. This frame was encapsulated inside an Ethernet packet using a private EtherType (0x88b5), allowing us to tunnel 802.11-like frames across a wired virtual interface.

Before transmission, we used ARP to resolve the MAC address of the target machine (in this case, the host macOS system). If no response was received, we defaulted to the Ethernet broadcast address.

Once prepared, the custom frame was transmitted repeatedly using a Scapy loop, with a 100 ms delay between packets. On the macOS host, the packets were captured using Scapy too, confirming that the frames arrived intact and with the expected structure.

This method allowed us to simulate a high-frequency beacon spam pattern in a safe, closed environment, suitable for feeding into traffic analysis pipelines or IDS training without requiring real Wi-Fi emissions.

Key points:

- Beacon frames were generated with spoofed MAC and SSID ("RogueAP").
- Frames were tunneled over Ethernet using a custom EtherType (0x88b5).
- The Mac target passively received the traffic and captured it .

9. System Evaluation and Performance Metrics

10. Limitations and Challenges

While the AI component of the system demonstrated strong classification performance in offline evaluations, several limitations were encountered throughout the project:

- **Feature Availability in Real-Time Capture:** The full AWID model relies on low-level wireless features not directly accessible via Scapy or basic packet capture. Also, using Wireshark with the available hardware resources we have resulted in capturing most important data with nan values. As a result, we had to retrain a separate, reduced-feature model specifically for live use cases, slightly sacrificing accuracy.
- **Data Imbalance and Quality:** The AWID dataset was heavily imbalanced, with most samples belonging to normal traffic. This required careful feature selection and resampling strategies to ensure that attack detection was not biased or under-represented.
- **Absence of WPA2 Key Cracking Datasets:** No public datasets contain labeled examples of WPA2 dictionary attacks. We used a proxy strategy based on brute-force login behavior, which, while innovative, may not fully capture all nuances of actual handshake cracking.
- **Limitations in Attack Simulation:** Some attacks, such as DDoS and WPA2 cracking, required complex network setups and could not be reliably reproduced at scale. As such, certain simulations were limited in scope or duration.
- **System Integration Challenges:** Interfacing real-time detection with mitigation tools introduced a great challenge in compatibility and delays requiring fine-tuning to avoid false positives disrupting legitimate users.

Despite these challenges, the modularity of the system allowed us to isolate and address most technical bottlenecks without compromising the broader project goals.

11. Conclusion

This project demonstrated the feasibility and effectiveness of enhancing WLAN security using AI-based traffic analysis. Through the use of multiple datasets and targeted models, we successfully trained classifiers capable of identifying and responding to key WLAN attacks, including:

- Deauthentication attacks
- ARP spoofing
- Rogue access points
- WPA2 key cracking (approximated)

- DDoS attacks via flow monitoring

The Random Forest models used throughout the project provided strong performance and interpretability, making them suitable for both offline analysis and real-time deployment.

One of the key strengths of this project lies in its layered architecture—allowing high-fidelity analysis in offline environments and lightweight detection in live systems. Moreover, by combining different datasets and tools (Scapy, Zeek, Wireshark), we established a flexible foundation for further experimentation and integration.

12. Future Work

Several avenues can be explored to enhance and extend the current system:

- **Expand Attack Coverage:** Extend the model to detect other advanced threats such as Evil Twin attacks, DNS spoofing, and beacon flooding.
- **Real-Time Dashboards:** Integrate Grafana or a custom web dashboard for live visualization of threats and network status.
- **Deep Learning Models:** Explore the use of LSTM or CNN-based models to detect sequential or time-dependent attack patterns, especially for stealthy or low-rate attacks, and especially with the expansion of training data.
- **Hardware Deployment:** Port the lightweight version of the system onto Raspberry Pi or low-power access points for distributed, edge-based WLAN security.
- **Improved Dataset Collection:** Build a custom dataset for WPA2 handshake cracking by simulating dictionary attacks and logging features in a controlled lab environment.
- **Federated or Online Learning:** Implement a feedback loop to allow the model to update over time based on new patterns or administrator feedback.

The current system serves as a solid prototype, and with further development, it has the potential to become a production-grade solution for AI-enhanced WLAN security.