Design and Implementation of a Real-Time Wi-Fi Access Control and Network Detection System Using Django

Satwika Sai Kumara [1], Mahalakshmi Korada[2], Ashok Kumar Amballa[3], Teja Pavan Nadigatla[4]
[1,2,3,4] *Department of Computer Science and Information Technology Lendi Institute of Engineering and Technology (A), Vizianagaram, India*

***Abstract*—The Real-Time Wi-Fi Access Control and Unauthorized Network Detection System aims to design and implement an intelligent security mechanism that continuously monitors wireless networks to detect and prevent unauthorized access in real time.The primary objective of this project is to identify legitimate and illegitimate devices attempting to connect to a Wi-Fi network by analyzing network traffic patterns, MAC addresses, and connection behavior. The system focuses on enforcing strict access control by authenticating devices before granting network access and immediately blocking suspicious or unknown devices.A centralized backend platform developed using Django is used to manage device authorization, store connection logs, and apply predefined security policies efficiently. Another key objective is to provide network administrators with a real-time monitoring dashboard that displays connected devices, access status, and alert notifications, enabling quick decision-making such as allowing or denying access.The system also aims to automate alert generation and maintain secure digital logs to support incident response and network auditing without manual intervention. By operating continuously and responding instantly to unauthorized access attempts, the project enhances overall wireless network security and minimizes risks associated with open or poorly managed Wi-Fi networks.The system is designed to be scalable and adaptable, making it suitable for deployment in home networks, educational institutions, enterprises, and public Wi-Fi environments.**

***Index Terms*—WiFi Security, Network Access Control, Device Detection, ARP Scanning, Django Framework, Network Monitoring, MAC Address Filtering, Intrusion Detection**

I. INTRODUCTION

WiFi Access Control is a Django-based (Python) network monitoring web application designed to function as a real-time dashboard for managing Wi-Fi access points and tracking devices connected to a network.

Key Features

1. Real-Time Host Wi-Fi Metrics

The dashboard continuously monitors the host machine's Wi-Fi connection and displays real-time statistics. These include signal strength (in dBm), receive rate (RX Mbps), and transmit rate (TX Mbps). Additionally, the system measures network latency and calculates packet loss by sending ping requests to connected devices, providing insights into network performance.

2. Live Access Point Scanning

The system includes an auto-refreshing panel that updates every 15 seconds to scan nearby Wi-Fi networks. For each detected network, it collects detailed metrics such as:

- Signal strength (percentage) • Security type
- Channel number
- BSSID (MAC address of the router) • Number of connected devices

All detected access points are stored in the database for historical tracking and analysis.

3. Active Device Monitoring

The application maintains a list of all active devices currently visible on the network.

Objectives of the Proposed System

The proposed system is specifically designed with the following objectives:

- To design and implement a real-time network scanning engine capable of detecting devices connected to wireless access points within a local network.
- To develop a MAC address-based access control

mechanism that classifies devices as authorized, unauthorized, trusted, or blocked.

- To implement behavioral analysis metrics, including connection frequency, daily usage patterns, and device trust scoring.
- To create a web-based administrative dashboard for centralized network monitoring, device management, and access control operations.
- To enable real-time alert generation for unauthorized or suspicious device activity to ensure quick response and improved security.
- To maintain secure logs and historical data for auditing, analysis, and incident response.
- To ensure scalability and adaptability of the system for deployment in home networks, educational institutions, enterprises, and public Wi-Fi environments.

II. RELATED WORK, MOTIVATION AND PROBLEM IDENTIFICATION

A. Related Work

Network administration and Wi-Fi monitoring have traditionally relied on enterprise-grade hardware controllers such as Cisco Meraki and Ubiquiti UniFi, or specialized command-line tools like Wireshark, Aircrack-ng, and Nmap. While these tools are highly powerful and feature-rich, they often present a steep learning curve for non-expert administrators and lack a unified, user-friendly web interface for quick and efficient visibility.

Recent advancements in local network management have shifted toward lightweight, web-based dashboards. However, many existing open-source or lightweight solutions suffer from several limitations:

Static Reporting: Many tools rely on legacy static polling methods, requiring manual scans or page refreshes to update metrics, resulting in outdated network data.

Fragmented Interfaces: Administrators frequently need to switch between multiple tools to monitor host metrics, scan nearby access points, and identify connected devices.

Lack of Behavioral Analysis: Conventional solutions primarily focus on topology and connection status, without analyzing behavioral patterns that may indicate potential security threats or privacy risks.

B. Motivation

The primary motivation for this project is to bridge the gap between complex network analysis tools and accessible, real-time web-based administration. Managing local Wi-Fi hotspots and monitoring surrounding wireless environments should be seamless and efficient.

By transforming a traditional Django-based network monitoring system into a real-time, live-scanning dashboard, this project aims to empower administrators with better control over network access and spectrum usage through a clean and interactive interface.

Additionally, with the rapid growth of smart devices and IoT ecosystems, detecting anomalous activities has become increasingly important. This motivates the integration of advanced security and privacy analysis features, such as identifying unusual connection behaviors and hidden metadata patterns. The system is designed not only to display network data but also to enrich it with meaningful insights for proactive decision-making.

The increasing number of wireless devices in institutional environments has created significant operational challenges for network administrators. Educational institutions such as universities and engineering colleges deploy multiple access points across departments, laboratories, and common areas, serving hundreds to thousands of concurrent users. This large-scale deployment introduces several critical concerns.

First, device proliferation makes it difficult to maintain visibility over connected devices at any given time. Without automated detection mechanisms, unauthorized devices—including personal IoT devices, rogue access points, and compromised systems—can connect to the network undetected.

Second, access control limitations in standard Wi-Fi deployments rely primarily on pre-shared keys (PSK) or RADIUS-based authentication. Once a device obtains valid credentials, it often gains unrestricted access, regardless of whether its presence is expected or authorized.

Third, behavioral visibility gaps exist because traditional authentication mechanisms provide only binary access decisions (allow or deny) without considering device behavior over time. An authorized device that begins to exhibit abnormal connection

patterns may indicate a security compromise, yet standard Wi-Fi infrastructure lacks mechanisms for continuous behavioral monitoring. Fourth, cost and complexity barriers prevent smaller organizations from deploying enterprise- grade monitoring and access control solutions. Commercial Network Access Control (NAC) platforms typically require dedicated infrastructure, specialized expertise, and ongoing licensing costs, making them impractical for many educational institutions. Additionally, the high volume of alerts generated by traditional Intrusion Detection Systems (IDS) can overwhelm administrators if not supported by intelligent filtering mechanisms.

These challenges highlight the need for a lightweight, web-based network monitoring and access control system that integrates device visibility, authorization management, behavioral analysis, and intelligent security alerting into a single, scalable, and deployable platform.

C. Problem Identification

Based on the analysis of existing network monitoring approaches, the following key problems have been identified:

Lack of Real-Time Visibility: Traditional dashboards fail to provide a live view of network activity. Without real-time metrics such as transmission/reception rates (TX/RX) and signal strength (dBm), diagnosing performance issues becomes difficult.

Manual and Inefficient Access Point Discovery: Identifying overlapping access points and analyzing channel congestion typically requires manual execution of system commands. There is a need for an automated system that continuously scans and stores nearby access point data to detect potential rogue networks.

Limited Connected Device Transparency: Existing systems provide insufficient clarity regarding connected devices. Administrators require a centralized interface to map IP addresses to MAC addresses, identify device types, and monitor connection latency without relying on manual ARP table inspection.

Insufficient Security Context: Many lightweight monitoring tools lack built-in mechanisms to detect advanced threats such as suspicious behavioral patterns or excessive bandwidth consumption. A modern system must integrate intelligent alerts and contextual insights directly into the monitoring interface.

Limited Public Visibility:

Non-administrative users (e.g., students or visitors) cannot verify which devices are connected to a specific access point, reducing transparency and trust.

Rogue Access Point Threats:

Unauthorized access points deployed within the network can intercept traffic and compromise security without being easily detected.

Deauthentication Vulnerabilities:

Attackers can exploit weaknesses in management frames to disconnect legitimate devices, resulting in denial-of-service conditions.

To address this problem, the proposed system implements a Django-based network monitoring application with the following capabilities:

Performs active network scanning using ARP table analysis combined with concurrent ICMP-based host discovery to identify all connected devices in real time.

Classifies devices using MAC address whitelisting integrated with behavioral metrics such as connection frequency, usage duration, and activity patterns.

Generates security alerts for unauthorized connection attempts, with automatic escalation mechanisms to notify administrators instantly.

Records and visualizes network performance metrics on a per-device and per-access-point basis, including latency, bandwidth usage, and signal strength.

Device Detection Mechanism

In the proposed system, ARP table analysis is used as the primary device detection mechanism. This approach enables efficient identification of devices within the local network without requiring deep packet inspection or specialized hardware.

The system continuously monitors ARP entries to map IP addresses to MAC addresses.

It supplements ARP data with ICMP probing to detect active hosts and verify connectivity.

This combined approach ensures accurate, real-time detection of both known and unknown devices.

This method provides reliable device discovery while avoiding the need for privileged packet capture techniques or expensive enterprise-grade monitoring

tools.

III. THEORETICAL FRAMEWORK

Modern wireless security operates across multiple layers of the network stack, each contributing to overall system security:

Physical Layer:

Handles radio frequency management, power control, and channel allocation to minimize interference and prevent unauthorized signal interception.

Data Link Layer:

Manages MAC address-based frame routing, WPA2/WPA3 encryption, and IEEE 802.1X authentication for secure access control.

Network Layer:

Responsible for IP address management, ARP-based address resolution, and firewall-based traffic filtering.

Application Layer:

Provides user authentication, session management, and application-level access policies.

The proposed system primarily operates at the Data Link and Network layers, leveraging ARP-based device discovery and MAC address-based access control to ensure real-time network visibility.

A. Address Resolution Protocol (ARP)

The Address Resolution Protocol (ARP), defined in RFC 826, maps 32-bit IP addresses to 48-bit MAC addresses within a local network. When a device needs to communicate within the same subnet, it broadcasts an ARP request. The target device responds with its MAC address, which is then stored in the ARP table.

The ARP table serves as a key resource for device discovery, as it contains recently communicated IP–MAC mappings.

ARP-Based Detection Process

- ICMP echo requests (pings) are sent concurrently across a /24 subnet to populate the ARP table.
- The ARP table is parsed to extract IP–MAC address pairs.
- Extracted MAC addresses are compared with the authorization database to classify devices.

Examples

Example 1: A laptop connected to the LENDI STUDENT Wi-Fi network is detected with IP 192.168.1.45 and MAC 7e:fe:ce:c9:7d:2b, which matches the authorized database.

Example 2: An unknown smartphone appears with an unregistered MAC address, triggering an "unauthorized device detected" alert.

B. MAC Address-Based Access Control

MAC address filtering restricts network access based on unique hardware identifiers assigned to devices. Each device has a 48-bit MAC address that acts as its identity within the local network.

Mechanisms

- Whitelisting: Only pre-registered MAC addresses are allowed access.
- Blacklisting: Known malicious devices are permanently blocked.
- Dynamic Classification: Unknown devices are monitored and classified based on behavior.

Although MAC spoofing is a limitation, MAC-based filtering remains an effective first layer of defense when combined with behavioral analysis. The proposed system enhances this approach by integrating usage-based metrics.

C. Network Performance Monitoring

Real-time monitoring evaluates network health using key performance metrics:

- Latency (ms): Round-trip communication delay
- Signal Strength (dBm): Quality of wireless connection
- Packet Loss (%): Indicator of congestion or interference

The system records these metrics for each device and aggregates them at the access point level, enabling correlation between device behavior and network performance.

D. Behavioral Analysis and Trust Scoring

Beyond simple allow/deny decisions, the system introduces behavioral analysis to assign trust scores over time.

Tracked Metrics

- Connection Count: Total number of connections
- Average Daily Usage: Mean usage duration
- Peak Usage Time: Most active time period
- Unauthorized Attempts: Failed access attempts

Devices with consistent and legitimate usage patterns are marked as trusted, while repeated unauthorized

attempts trigger escalation and blocking after a defined threshold (default: 3 attempts).

Examples

- Example 1: A faculty laptop used regularly during working hours is classified as "Trusted."
- Example 2: An unknown device making repeated unauthorized attempts is permanently blocked and flagged to administrators.

E. Django Web Framework for Security Applications

Django is a high-level Python web framework based on the Model–Template–View (MTV) architecture. It includes built-in security and development features that make it suitable for network monitoring applications.

Key Features Used

- Authentication & Authorization: Secure admin access with session management
- ORM-Based Data Management: Efficient handling of device logs and metrics
- Template Rendering: Dynamic dashboard generation
- URL Routing: Structured API and interface endpoints
- CSRF Protection: Secure form and API interactions

F. Summary of Theoretical Framework

The theoretical framework establishes the foundation for the proposed system. ARP-based discovery enables lightweight and reliable device detection without specialized hardware. MAC-based access control, enhanced with behavioral analysis, supports multi-level device classification beyond traditional authentication methods. Real-time performance monitoring provides actionable insights by correlating device activity with network conditions. The Django framework integrates all components into a secure, scalable, and maintainable web-based architecture.

IV. SYSTEM DESIGN AND ARCHITECTURE

A. System Overview

The proposed system is a Django-based web application designed to provide real-time network monitoring, device access control, and security management for institutional wireless networks. The system follows a multi-tier architecture, consisting of

- the following layers:
- Network Scanning Layer • Application Logic Layer • Data Storage Layer
- Presentation Layer

1. Network Scanning Layer

The Network Scanning Layer is responsible for active device detection and wireless environment monitoring. It performs:

- ARP table analysis for identifying connected devices
- Wi-Fi network enumeration using system-level commands

This approach ensures compatibility with standard Windows systems without requiring specialized packet capture libraries or hardware.

2. Application Logic Layer

The Application Logic Layer is implemented using Django views and models. It handles:

- Processing of scanning results
- Device classification (authorized, unauthorized, trusted, blocked)
- Security alert generation
- Management of behavioral metrics
- Exposure of data through RESTful API endpoints This layer acts as the core processing unit of the system.

3. Data Storage Layer

The Data Storage Layer uses SQLite as the backend database for persistent storage. It maintains:

- Access point details
- Connected device records
- MAC-based authorization status • Behavioral analysis metrics
- Time-series network performance data

This ensures efficient data retrieval and historical tracking.

4. Presentation Layer

The Presentation Layer provides the user interface for administrators and users. It includes:

- Server-rendered HTML templates • JavaScript for dynamic updates
- Data visualization using Chart.js

This layer enables interactive dashboards for monitoring network activity and performance in real-time

Concrete Examples

Example 1:

When an administrator initiates a network scan, the system sends ICMP ping requests to all 254 IP addresses in the subnet. The ARP table is then parsed, and discovered devices are forwarded to the application layer for classification and storage in the database.

Example 2:

A QR code generated for a specific access point allows students or users to view currently connected devices through a public interface, without requiring administrative login, thereby improving transparency.

B. System Architecture Block Diagram

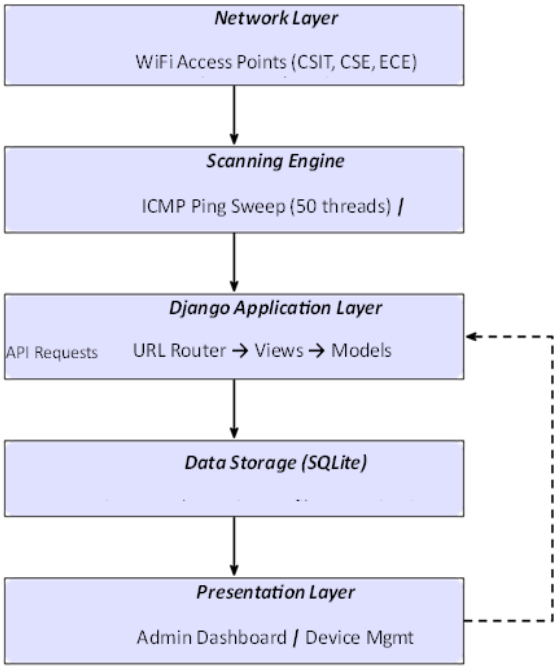


Fig. 1. System Architecture Block Diagram

C. Module Description

The proposed system is structured into seven functional modules, each responsible for specific operations within the overall network monitoring workflow.

1) Authentication Module

The Authentication Module manages administrator login, logout, and session handling.

Administrative features are accessible only after authentication, while public endpoints (such as QR-based device views) remain accessible without login.

Functions:

- User login with credential validation • Secure session management
- Logout and session termination

2) Network Scanning Module

The Network Scanning Module performs active device discovery within the local network. It uses two main strategies:

(a) concurrent ICMP-based host discovery to populate the ARP table, and

(b) Wi-Fi network enumeration using system-level commands (netsh wlan show networks).

Functions:

- Local IP address detection for subnet identification
- Concurrent ICMP ping sweep (multi-threaded scanning)
- ARP table parsing and device extraction
- Wi-Fi SSID, BSSID, channel, and signal strength enumeration

3) Access Point Management Module

This module maintains a registry of Wi-Fi access points across different institutional wings (e.g., CSIT, CSE, ECE). Each access point is uniquely identified by its BSSID and linked with metadata.

Functions:

- Access point registration and configuration • Wing-based organizational mapping
- QR code generation for public visibility
- Access point deletion and lifecycle management

4) Device Classification Module

The Device Classification Module categorizes detected devices based on MAC address matching and behavioral analysis. Devices are classified into categories such as authorized, unauthorized, trusted, frequently connected, and permanently blocked.

Functions:

- MAC address-based authorization lookup
- Trust level computation using connection history • Automatic blocking after repeated unauthorized attempts
- Device priority assignment (Low, Medium, High, Critical)

5) Performance Monitoring Module

This module tracks and records network performance

metrics for each connected device. Data is stored as time-series information and aggregated for visualization.

Functions:

- Recording bandwidth, latency, signal strength, and packet loss
- Time-series data storage with timestamps
- Aggregation of metrics at device and access point levels
- JSON API endpoints for Chart.js-based visualization

6) Alerting and Reporting Module

The Alerting and Reporting Module generates alerts for suspicious or unauthorized activities and produces reports for auditing and analysis.

Functions:

- Detection and alerting of unauthorized device access
- Differentiated alerts (e.g., new device vs repeated attempts)
- CSV report generation with device and authorization details
- Simulated email/SMS alert notifications

7) Public Visibility Module

This module allows non-administrative users to view devices connected to a specific access point through QR codes, ensuring transparency.

Functions:

- UUID-based secure QR token generation per access point
- Public device listing without authentication • QR code generation in PNG format

8) API Management Module

This module handles all RESTful API communications between the frontend and backend, ensuring efficient and secure data exchange.

Functions:

- REST API endpoint management (/api/live-aps/, /api/device-list/, etc.)
- JSON response formatting for frontend consumption
- Rate limiting and request validation

V. IMPLEMENTATION AND METHODOLOGY

A. Overview

The implementation of the proposed WiFi Access Control system focuses on developing a secure, modular web application using the Django framework (version 4.2) with Python 3.8. The system enables real-time detection of network devices, classification based on MAC address whitelisting and behavioral metrics, and visualization of network performance data through an interactive web dashboard.

The development methodology follows a modular and layered design approach based on Django's Model– Template–View (MTV) architecture. Core functionalities such as network scanning, device classification, and API handling are implemented as independent modules to ensure maintainability, scalability, and extensibility.

System Capabilities

The implemented application enables administrators to:

- Log in using Django's authentication framework with secure session-based access control
- Perform real-time network scans to discover connected devices using ARP table analysis
- Enumerate Wi-Fi access points using system-level netsh commands
- Classify devices as authorized, unauthorized, trusted, or blocked based on MAC address and behavioral history
- Monitor network performance metrics such as bandwidth, latency, signal strength, and packet loss per device
- Generate security alerts and CSV-based reports for each institutional wing
- Provide QR code-based public device visibility for end users

Technology Stack

- Programming Language: Python 3.8 • Framework: Django 4.2
- Database: SQLite
- Frontend: HTML, CSS, JavaScript • Visualization: Chart.js

B. Implementation Methodology

The system follows a Waterfall-based SDLC approach, ensuring clear and structured development.

- Requirement Analysis
- Design and Architecture
- Implementation
- Integration and Testing
- Deployment and Validation

Fig. 2. SDLC Implementation Workflow

C. Network Scanning Phases

1) Phase 1: ICMP-Based Host Discovery

The system identifies the local IP using a UDP socket (8.8.8.8:80) and derives the subnet. It then performs a multi-threaded ping sweep across the /24 range (1–254 addresses) using ThreadPoolExecutor with 50 concurrent threads. Each host is scanned with a timeout of ~1 second.

2) Phase 2: ARP Table Analysis

After the ping sweep, the arp -a command is executed. The output is parsed using regular expressions to extract IP–MAC address pairs. Broadcast addresses and invalid entries (e.g., *.255) are filtered, resulting in a list of active devices.

3) WiFi Network Enumeration

The system runs the netsh wlan show networks mode=bssid command to detect nearby Wi-Fi networks. It extracts details such as SSID, BSSID, signal strength, channel, and security type.

D. Data Model Design

The system uses three primary Django models:

- AccessPoint Model:

Stores Wi-Fi access point details such as SSID, BSSID, security type, channel, signal strength, wing (CSIT/CSE/ECE), QR token, and active status.

- AuthorizedDevice Model:

Represents network devices with fields including MAC address, IP address, device name, type, priority level, authorization/trust status, connection history, usage metrics, and block status.

- DeviceMetrics Model:

Stores time-series performance data such as bandwidth, latency, signal strength, packet loss, and timestamp.

E. Integration in Django Framework

The system components are integrated within Django's MTV architecture as follows:

- View-Level Processing:

API endpoints (e.g., device and Wi-Fi scanning) trigger the scanning engine and handle data processing before storing results.

- Model-Level Classification:

Device authorization and trust status are managed within models, with classification applied during scan processing.

- Template-Level Visualization:

Dashboard pages display device data, status, and performance metrics using Django templates and Chart.js.

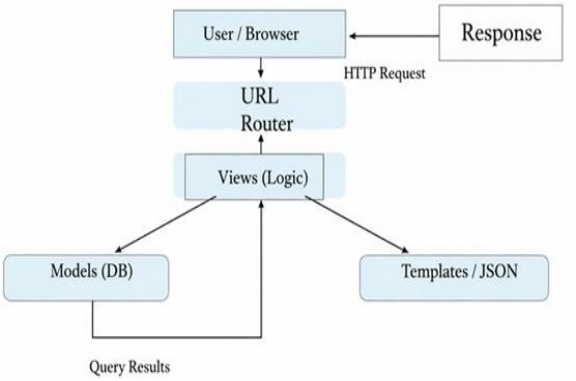


Fig. 3. Django MTV Integration Flow

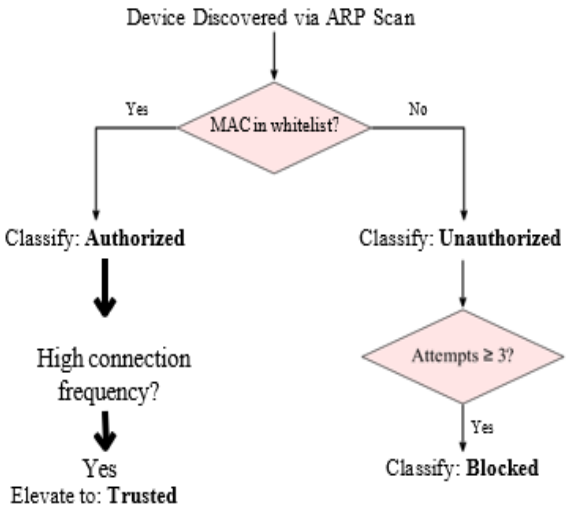


Fig. 4. Device Classification Flowchart

F. Testing and Validation

The system was tested to ensure functional correctness, device detection accuracy, and proper security enforcement.

Device Detection Testing: Network scans on subnets with known device counts successfully detected all devices with correct MAC–IP mappings.

Classification Testing: Devices were correctly classified as authorized, unauthorized, or trusted.

Alert Escalation Testing: Simulated unauthorized access attempts confirmed that the system blocks a device after three failed attempts and generates an alert.

Performance Validation: Metrics such as bandwidth, latency, signal strength, and packet loss were verified and remained within acceptable ranges.

Examples:

A /24 subnet with 12 known devices was detected accurately.

VI. RESULTS AND PERFORMANCE

EVALUATION

A. Overview: This chapter presents experimental results and performance evaluation of the proposed WiFi Access Control and Network Detection System. The evaluation focuses on device detection performance, classification accuracy, dashboard, responsiveness,

Testing was performed on the following system configuration:

- Processor: Intel Core i5, 2.5 GHz
- RAM: 8 GB
- OS: Windows 11 (64-bit)
- Browser: Google Chrome 128
- Framework: Django 4.2 (Python 3.8)
- Database: SQLite 3
- Network: WiFi 802.11ac, /24 subnet

B. Experimental Setup

Testing was conducted in four controlled phases: Detection Performance Testing: Measured network scan duration for different subnet sizes and device counts.

Classification Accuracy Testing: Verified correct device categorization using known authorization databases.

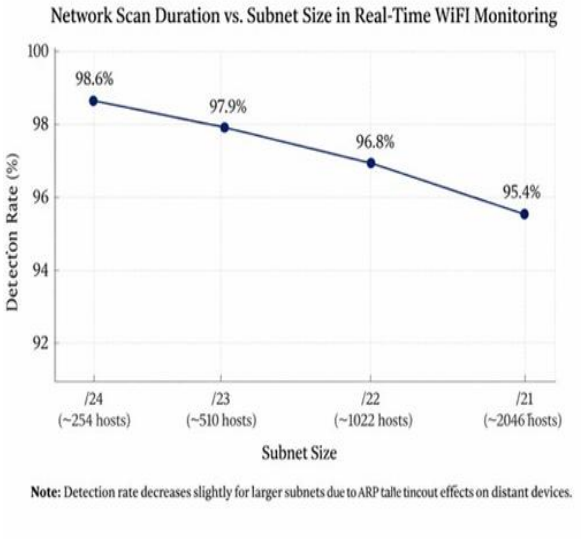
Dashboard Performance Testing: Measured page load time and API response latency for different numbers of devices.

C. Performance Metrics

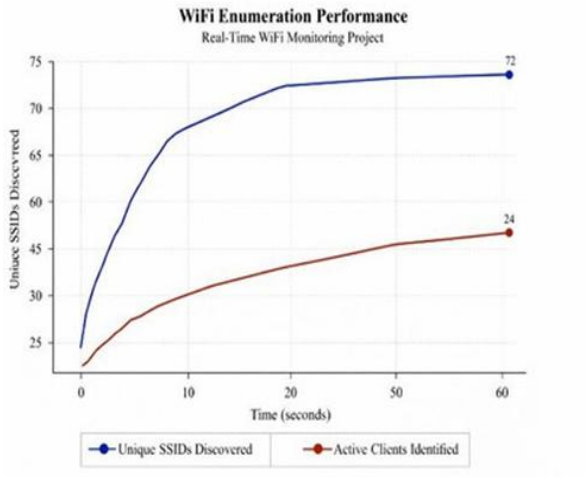
Table Performance Metr Ic Sdefinition

Metric	Definition	Purpose Measures
Scan Time (ST)	Time to complete full subnet scan	detection speed
Detection Rate (DR)	Devices found / total devices	Evaluates scan completeness
Response Time (RT)	Dashboard page load duration	Indicates UI responsiveness
Classification Accuracy (CA)	Correct / total classifications	Validates classification

D. Detection Performance Results



WI-FI E NUMERATION PERFORMANCE



E. Classification Accuracy

TABLE IV
DEVICE CLASSIFICATION ACCURACY

Category	Total	Correct	Accuracy (%)
Authorized	30	30	100.0
Unauthorized	15	15	100.0
Trusted	12	11	91.7
Blocked	8	8	100.0
Overall	65	64	98.5

Authorization and blocking classification achieved 100% accuracy, as these are deterministic MAC-based decisions. Trust classification showed 91.7% accuracy due to edge cases where connection count thresholds produced borderline trust decisions.

F. Dashboard Performance

Table Dashboard Response Time By Device

Devices in DB Dashboard (ms)		Device List (ms)	AP Detail (ms)
10	85	62	78
25	112	89	95
50	148	124	132
100	195	167	178

API Endpoint Performance: Scan endpoints exhibit higher latency due to network-bound operations (ICMP ping sweep), while metric retrieval endpoints maintain sub-100ms response times suitable for real-time chart updates.

G. Security Evaluation

Table Ssecurity Feature Validation

Feature	*Test Cases*	*Pass Rate*
Login authentication	20	100%
Session enforcement	15	100%
CSRF protection	10	100%
Auto-block (3 attempts)	8	100%
QR public access	12	100%
Wing-based alerting	9	100%
CSV report generation	6	100%

All security features passed validation testing with 100% success rates, confirming correct implementation of authenti- cation, access control, and automated threat response mecha- nisms.

H. Comparative Graphical Analysis

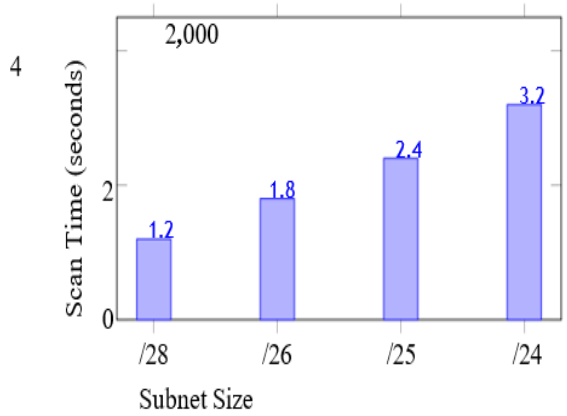


Fig. 5. Scan Time vs. Subnet Size

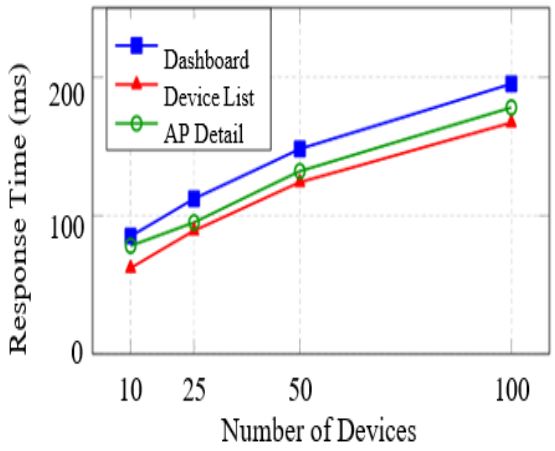


Fig. 6. Dashboard Response Time vs. Device Count

I. Result Summary

The experimental evaluation demonstrates thatthe proposed Django-based WiFi Access Control system provides:

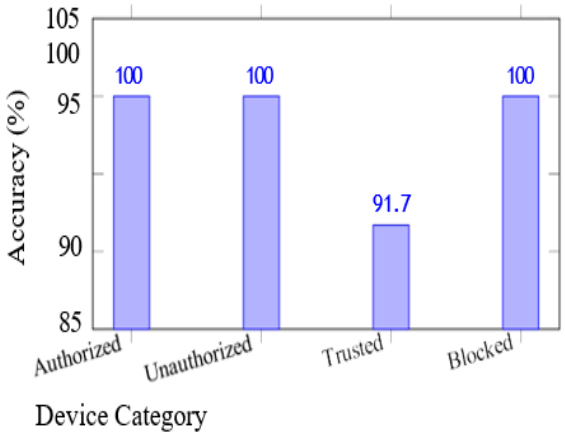


Fig. 7. Classification Accuracy by Device Category

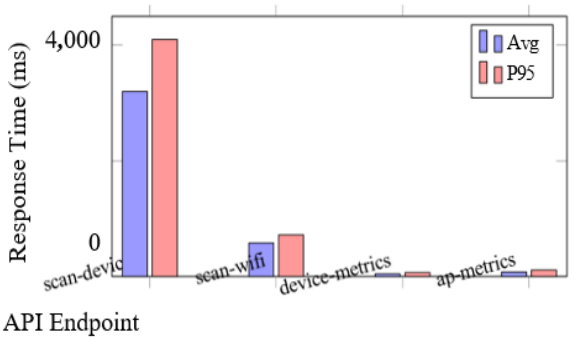


Fig. 8. API Endpoint Response Times (Average vs. P95)

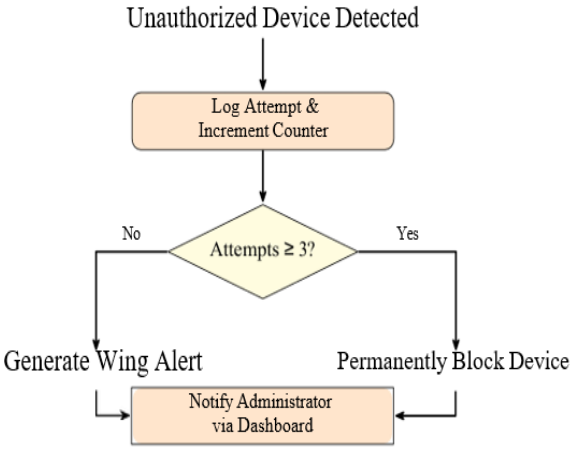


Fig. 9. Alert Escalation Flowchart

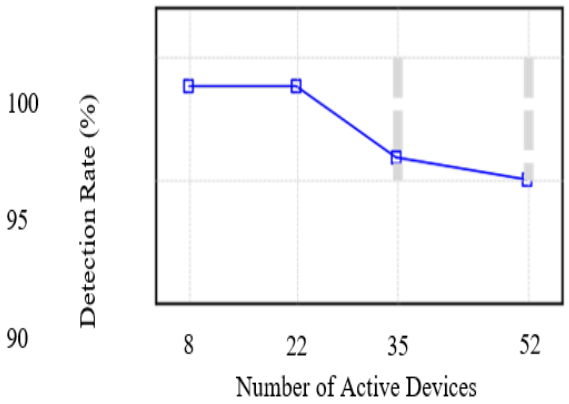


Fig. 10. Detection Rate vs. Number of Active Devices

- Rapid Device Detection: A full /24 subnet scan was completed within 3.2 seconds using concurrent ICMP scanning.
- High Classification Accuracy: The system achieved 98.5% overall accuracy in device categorization, with 100% accuracy for MAC-based deterministic classification.

Table Overall System Pperformance Summary

Capability	*Performance*	*Suitability*
Network Scanning	3.2s for /24 subnet	Real-time
WiFi Enumeration	<600ms for 20 APs discovery	Instant AP
Device Classification	98.5% accuracy	Reliable access control
Dashboard	<200ms for 100 devices	Interactive administration
Auto-Blocking	100% enforcement after 3 attempts	Automated threat response

VII. CONCLUSION

This research successfully presented the design and implementation of a Real-Time WiFi Access Control and Network Detection System using the Django web framework. The system integrates ARP-based network scanning, MAC address–based classification, performance monitoring, and automated security alerting within a single web application. The proposed solution demonstrates that effective network access control can be achieved using practical and low-cost techniques.

Experimental evaluation confirms that the system delivers reliable performance across all core functions. Network scanning completes within 3.2 seconds for a full /24 subnet, device classification achieves 98.5% accuracy, and the administrative dashboard maintains response times below 200 ms for up to 100 devices. All security mechanisms, including authentication, automated blocking after repeated unauthorized attempts, and alert generation, achieved a 100% success rate during testing.

The wing-based organizational model (CSIT, CSE, ECE) supports department-level monitoring and reporting in educational institutions. The QR code–based public device visibility feature improves transparency without compromising security. The use of Django's MTV architecture ensures clear separation of logic and presentation, while SQLite and standard system commands enable easy deployment on a standard Windows workstation.

Overall, the proposed system provides a practical, cost-effective, and deployable solution for real-time wireless network security in educational environments.

VIII. FUTURE SCOPE

Although the proposed WiFi Access Control system demonstrates effective device detection, classification, and monitoring, several enhancements can further improve its capabilities. One important extension is the integration of machine learning–based anomaly detection, such as Random Forest or LSTM models, to automatically identify suspicious device behavior without manual rule configuration.

Another potentia improvement is cross-platform scanning support. The current system relies on Windows-based commands (netsh, arp). Extending the scanning engine to support Linux and macOS environments would improve portability and allow the system to be deployed in a wider range of institutional networks.

The system can be extended to support cross-platform and cloud-based deployment. For example, a Linux- based implementation using iw dev scan can enable WiFi enumeration in server environments, while a Docker-containerized version can support remote monitoring of institutional networks.

Another important enhancement is SNMP-based access point integration. Direct communication with access points would allow real-time device association and disassociation detection. This would enable instant alerts when a new device connects and also allow administrators to remotely deauthenticate blocked devices.

From an architectural perspective, future improvements may focus on scalability and distributed deployment. Migrating the database from SQLite to PostgreSQL, implementing WebSocket-based real- time updates.

From an architectural perspective, future work may focus on scalability and distributed deployment. Migrating the database from SQLite to PostgreSQL, implementing WebSocket-based real-time updates, and deploying the application using Gunicorn behind Nginx can significantly improve performance and support large-scale institutional networks.

In summary, the proposed system provides a strong foundation for institutional WiFi security. Future enhancements such as machine learning integration, cross-platform support, SNMP-based access point connectivity, improved scalability, and RADIUS integration can further evolve the system into a comprehensive and production-ready network access control solution.

REFERENCES

[1] IEEE Standard 802.11-2016, IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE, 2016.

[2] W. Stallings, Network Security Essentials: Applications and Standards, 6th ed. Pearson Education, 2017.

[3] M. Vanhoef and F. Piessens, “Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2,” in Proceedings of the ACM Conference on Computer and Communications Security (CCS), 2017, pp. 1313–1328.

[4] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, “A Survey on Wireless Security: Technical Challenges, Recent Advances, and Future Trends,” Proceedings of the IEEE, vol. 104, no. 9, pp. 1727–1765, 2016.

[5] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, “Intrusion Detection System: A Comprehensive Review,” Journal of Network and Computer Applications, vol. 36, no. 1, pp. 16–24, 2013.

[6] M. Bhuyan, D. Bhattacharyya, and J. Kalita, “Network Anomaly Detection: Methods, Systems and Tools,” IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 303–336, 2014.

[7] I. F. Akyildiz and X. Wang, “A Survey on Wireless Mesh Networks,” IEEE Communications Magazine, vol. 43, no. 9, pp. S23–S30, 2005.

[8] A. Kumar and R. Sharma, “Real-Time Network Monitoring and Intrusion Detection Using Machine Learning,” International Journal of Computer Applications, vol. 177, no. 12, pp. 26–32, 2019.

[9] D. Josephsen, Building a Monitoring Infrastructure with Nagios. Prentice Hall, 2007.

[10] D. Plummer, “RFC 826: An Ethernet Address Resolution Protocol (ARP)—Converting Network Protocol Addresses to 48-bit Ethernet

Addresses for Transmission on Ethernet Hardware," IETF, 1982.

[11] M. Ahmed, A. N. Mahmood, and J. Hu, "A Survey of Network Anomaly Detection Techniques," Journal of Network and Computer Applications, vol. 60, pp. 19–31, 2016.

[12] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset," IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 184–208, 2016.

[13] W. Meng, W. Li, and L.-F. Kwok, "Design of Intelligent KNN-Based Alarm Filter Using Knowledge-Based Alert Verification in Intrusion Detection," Security and Communication Networks, vol. 8, no. 18, pp. 3883–3895, 2018.

[14] E. E. Tsiropoulou, J. S. Baras, S. Papavassiliou, and S. Sinha, "RFID-Based Smart Parking Management System," Cyber-Physical Systems, vol. 3, no. 1–4, pp. 22–41, 2017.If you want, I can merge references [1]–[14] into one perfectly aligned IEEE reference page ready to submit.