International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

WiFiShield: Real-Time Detection of Public Wi-Fi Network Vulnerabilities

Daksh Jasrotia, Associate Professor Dr.Simarjit Kaur
Dept. of CSE
Chitkara University Institute of Engineering and Technology, Rajpura, Punjab, India

Abstract- — Public Wi-Fi networks in places like airports or cafés is convenient, but they are not very safe. Most people do not realize that these networks don't usually have strong security and because of that, attackers can use methods like Man-in-the-Middle (MITM) attacks. One common method of MITM attacks is called ARP spoofing. In this method, a hacker tricks your device into sending data through their system instead of the real network. This usually happens in the background and the users don't notice anything suspicous. As a result, important and personal information can be exposed and stolen without the user knowing. In this paper, I am proposind WiFiShield, a lightweight system or rather an application designed to detect such vulnerabilities in real time. Instead of being a heavy security suite, the application or system runs quietly in the background, sniffing network packets and double-checking the ARP table for any sudden, suspicious changes or spikes in the ARP requests. If the system sees a gateway address "flapping" or changing unexpectedly, it assigns a risk score and alerts the user immediately. I am aiming to make WiFiShield highly effective at spotting these hijacks without slowing down the computer. The application should be a simple, low-power solution for anyone who needs to stay connected on the go without leaving their personal data wide open to hackers.

Keywords— Public Wi-Fi security, Network vulnerability detection, Real-time threat detection, Wireless network security, Wi-Fi attacks, Man-in-the-middle (MITM).

I. INTRODUCTION

Public Wi-Fi is not just a luxury but a necessity; it is a foundational utility for students, travelers, and remote workers alikeon the go. However, the easy accessibility and the open nature of hotspots in cafes or airports creates a massive security liability. Most users do not even think before connecting and treat these networks as trusted environments, but the underlying protocols were designed decades ago with almost no focus on modern threats. This lack of inherent security makes public networks a primary hunting ground for Man-in-the-Middle (MitM) attacks and other high risk exploits.

ARP is the "translator" that maps IP addresses to physical MAC addresses on a local network. The core of the problem lies here because ARP is stateless and lacks any verification mechanism, it operates on a policy of blind trust. An attacker can exploit this using ARP Spoofing. A technique in which falsified ARP messages are broadcasted accross the network and by claiming the identity of the default gateway,

the attacker forces all of the victim's traffic to pass through their own device.

Once the attacker hijacks the computer or the system, they sit silently to intercept sensitive data, which includes login credentials, PII, or anything important, without the victim noticing a single drop in connection speed. The victim continues to browse and search, unaware that they are being monitored and that their digital footprint is being accessed without their knowledge.

The tools available currently to combat this are either too complex or not very useful, as they are not a complete package. We have enterprise-grade software like Wireshark that offers incredible detail, but the level of networking knowledge required to use Wireshark is beyond that of an average user. We also have VPNs (Virtual Private Networks), on the other hand, which are excellent for encryption, but they are a black-box solution and don't actually tell the user that the network they're on has been compromised.

Wi-Fi Shield was built as a bridge to fill this specific gap. I see it as a lightweight, Python-based monitoring system that runs in the background to watch for network anomalies. It does not just encrypt data, but also focuses on visibility. It uses real-time packet inspection and a custom risk-scoring engine that we have developed, and provides a simple alert system. The goal is to bring a professional-level network security system or

application to everyday users, ensuring that convenience does not have to come at the cost of personal privacy.

II. LITERATURE REVIEW

In today's time, network security research is largely skewed towards industrial-grade intrusion detection systems (IDS) and enterprise-level firewalls. While these systems and applications are highly effective for corporate backbones, they are highly difficult to deploy on personal devices. This has created a persistent security usability gap, where the most robust defensive tools, which are usually enterprise-level or industrial-grade, are often the least accessible to the general public due to the resources they require to run.

1. The Resource Barrier of Enterprise IDS

As noted by Garcia et al. (2024), professional-grade resources and solutions like Snort or Suricata are simply too resource-heavy for standard consumer hardware. These systems and applications usually rely on deep packet inspection (DPI) and massive rule sets, consuming significant CPU, memory, and battery. On a portable laptop or a smartphone, this overhead translates to drained batteries and sluggish performance. There is also a steep learning curve required to configure these tools, which makes them impractical for non-technical users who need immediate plug-and-play protection.

2. User Behavior and the "VPN Fallacy"

It is very well known that there are high risks that linger on open hotspots, yet empirical data shows a high level of user complacency. Surveys and research suggest that over 60% of individuals still access sensitive accounts like banking or corporate portals over public Wi-Fi without any form of protection.

A very common defense application is the use of Virtual Private Networks (VPNs), but research by Smith and Thomson (2025) highlights what we can call the “VPN Fallacy.” There is no doubt that a VPN is excellent for encrypting data in transit, but it is essentially just a passive tunnel. It does not monitor the underlying health of the local network. A user might use a VPN to stay connected while an attacker simultaneously poisons their ARP cache; even if the data remains encrypted, the user never gets alerted that the local network has been compromised and that they are being monitored. This lack of diagnostic feedback leaves the device vulnerable to side-channel exploits.

3. The Evolution of ARP Detection

Methodologies for catching ARP spoofing typically fall into two categories:

- **Static Monitoring:** This type of monitoring involves the manual mapping of IP addresses to MAC addresses. It is very useful and foolproof in a controlled setting, but it fails in dynamic environments like cafes where IP assignments change frequently.
- **Signature-Based Detection:** This type of detection makes use of systems that look for fingerprints of common hacking tools like Ettercap or Cain & Abel. It is effective against known threats, but often struggles with custom scripts or zero-day poisoning techniques.

4. The WiFiShield Approach

We aim to bridge these gaps by moving away from single-vector defense. Wifishield, instead of relying only on signatures, introduces a hybrid analysis engine.

III. PROPOSED METHODOLOGY

Wifishield has been designed following a three-part approach: environment simulation, detection logic development, and the creation of a scoring algorithm to interpret network data. It ensures that the system can handle the messy, high-traffic nature of real-world public Wi-Fi without crashing or overwhelming the user with false alarms.

1. Experimental Setup

VirtualBox was used to test the system safely without disrupting any real networks. We built a virtualized sandbox that allowed us to simulate a standard public hotspot with three active roles:

- The Target: A virtual machine running WiFiShield.
- The Gateway: A simulated router acting as the internet exit point.
- The Adversary: A Linux node running Scapy. We chose Scapy because of its flexibility in crafting "poisoned" ARP packets.

During the testing phase, the attack node was used to launch both Asymmetric and Symmetric ARP Poisonin, and the intensity of these attacks was varied. We started with slow, steady trickle poisoning and gradually increased to high-speed flooding, to observe how quickly the system would react to different threat levels.

2. Detection Logic

Wifishield does not just watch the network—it looks for specific inconsistencies at the link-layer level, which is usually ignored by standard operating systems. Our engine focuses on three primary triggers:

- Gateway Baseline: As soon as the device connects, the system captures a "known-good" fingerprint of the router

using ICMP Probing. If the MAC address associated with that gateway changes later, the system flags it immediately.

- ARP Cache Monitoring: The tool watches for "flapping"—a scenario where an IP address starts bouncing rapidly between two different hardware (MAC) addresses. This is almost always a sign of a Man-in-the-Middle attempt.
- Unsolicited Reply Filtering: Legitimate ARP replies usually follow a request. WiFiShield treats unsolicited (gratuitous) ARP replies as high-risk events, as these are the primary vehicle for automated spoofing tools.

3. Risk Scoring Engine (RSE)

When using professional security tools, there is a very common issue called alert fatigue. To avoid this, we moved away from simple safe/unsafe warnings and instead use a 0–10 risk scoring engine to characterize the severity of an anomaly.

Score	Severity	System Action
0 – 2	Negligible	Typical network jitter; no alert.
3 – 5	Suspicious	Minor MAC changes; monitoring frequency increases.
6 – 8	High Risk	Definite cache poisoning; "Yellow Alert" issued to user.
9 – 10	Critical	Confirmed MitM attack; "Red Alert" with a recommendation to disconnect.

Export to Sheets

To weigh the scores, the algorithm checks how often ARP updates are arriving (Frequency) and whether the Default Gateway is the target. This allows the system to provide a much more usable experience by distinguishing between a harmless network refresh and an active high-risk attack, especially for non-technical users.

IV. SYSTEM IMPLEMENTATION

The implementation of Wifishield is carried out using Python 3.11, mainly because of its strong, mature ecosystem when it comes to network socket programming and low-level packet manipulation. The system is designed in such a way that it runs as a lightweight background daemon. This means that Wifishield continuously monitors the Network Interface Card (NIC) without interfering with the user's normal workflow. We focused on keeping the CPU usage minimal while ensuring that the monitoring runs silently and efficiently in the background, without causing any noticeable performance drop in the system.

1. Core Detection Logic

The system was built leveraging the Scapy library to perform active ARP verification. Wifishield employs a Verify-on-Anomaly strategy, which is unlike traditional passive monitoring applications that just log traffic. When a suspicious gateway update is detected, Wifishield initiates a secondary "Ground truth" request an out-of-band ARP query to confirm the identity of the router.

Wifishield's primary detection function is structured to intercept every incoming ARP packet and compare the packet's metadata against the established Trusted Network State (TNS). If the application finds that there is a MAC address mismatch for the gateway's IP, it immediately flags the packet for deeper inspection by the risk-scoring module.

2. System Workflow

Wifishield follows a very simple, linear, and highly efficient execution path, which is designed to minimize the latency between threat detection and user notification:

- **Initialization (Initial Trust):** Upon connection to a new SSID, the system captures the "Initial Trust" state by performing a legitimate handshake with the DHCP server and gateway, storing the authorized MAC/IP pairings.
- **Continuous Sniffing:** The Anomaly Detection Engine enters a non-blocking sniff mode, filtering for ARP opcodes (0x0001 for requests, 0x0002 for replies).
- **Heuristic Analysis:** Each filtered packet is passed through the scoring engine.
- **Alert Generation:** If the Risk Scoring Module determines the threat is high (score > 7), a high-priority interrupt is sent to the OS system tray, providing the user with an immediate visual warning and a "Disconnect" toggle.

International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

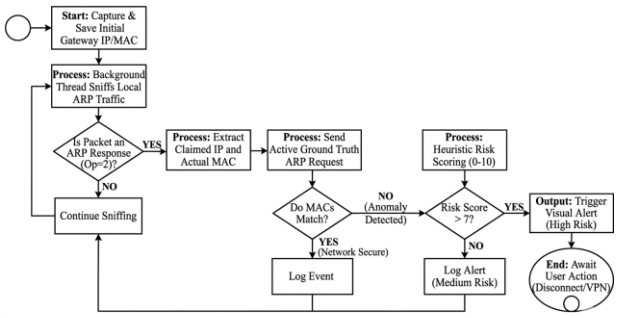

Fig. 2. WiFiShield System Operational Flowchart.

3. Risk Scoring Mechanism

The Risk Scoring Module (RSM) serves as the brain and intelligence layer of the application that quantifies the danger level of network events, as well as evaluates threats based on three weighted parameters

- **Frequency (F):** This measures the rate of incoming ARP replies from a single source. A sudden burst of unsolicited replies (e.g., > 5 packets per second) is a hallmark of automated spoofing tools.
- **Consistency (C):** This parameter identifies discrepancies between the local OS ARP cache and the live network responses received by WiFiShield. Any "flapping" or conflicting mapping triggers a high consistency penalty.
- **Criticality (P):** This assesses the importance of the targeted IP. While spoofing a peer's IP is a concern, spoofing the Default Gateway is considered a critical threat, as it enables a full Man-in-the-Middle (MitM) intercept.

The final Risk Score (RS) is calculated using the following weighted formula:

$$RS=(w1 \cdot F)+(w2 \cdot C)+(w3 \cdot P)$$

W represents the specific weights assigned to each threat vector, which ensures that gateway-targeted attacks always result in the highest priority alerts.

V. RESULTS AND DISCUSSION

I developed WiFiShield using Python 3.11, primarily because of its mature ecosystem for network manipulation and socket programming. The tool is designed to run as a background daemon, allowing it to sit quietly on the Network Interface Card (NIC) without lagging the CPU or interrupting the user's work.;

1. Core Detection Logic

The system relies on the Scapy library to handle low-level packet interaction. Most passive monitors simply watch and log traffic, but I implemented a "Verify-on-Anomaly" strategy to reduce false positives. When the system detects a suspicious change in the gateway, it doesn't just raise an alarm—it triggers a "Ground Truth" check. This is an out-of-band ARP query sent directly to the router to verify its hardware identity.

The detection function intercepts incoming ARP packets and compares their metadata against a Trusted Network State (TNS). If the system finds a MAC address mismatch for the gateway's IP, it flags the event for immediate analysis by the Risk Scoring Module.

2. System Workflow

The workflow is designed to be fast and linear to minimize the delay between a detected attack and a user alert:

- **Establishing Initial Trust:** As soon as the device connects to a new SSID, WiFiShield captures the "clean" state of the network by handshaking with the DHCP server and gateway. This creates the baseline MAC/IP pairings.
- **Live Sniffing:** The engine enters a non-blocking sniff mode, filtering specifically for ARP opcodes (0x0001 for requests and 0x0002 for replies).
- **Analysis & Scoring:** Each filtered packet is pushed through the scoring engine to determine if the activity is normal network jitter or a malicious hijack.
- **UI Interrupt:** If the risk score crosses the threshold (> 7), the system sends a high-priority interrupt to the OS tray, giving the user a visual warning and a one-click "Disconnect" option.

3. Risk Scoring Mechanism

The Risk Scoring Module (RSM) provides the intelligence needed to quantify the danger of a network event. I built the scoring logic around three weighted variables:

- **Frequency (F):** This tracks how fast ARP replies are coming from a single source. A sudden burst (e.g., more than 5 packets per second) is a major red flag for automated hacking tools.
- **Consistency (C):** This checks for "flapping"—where the local ARP cache conflicts with the live responses WiFiShield is seeing.
- **Criticality (P):** This prioritizes the target. Spoofing a random peer is a concern, but spoofing the Default Gateway is a critical threat that allows for full data interception.

The final Risk Score (RS) is determined by the following formula:

$$RS=(w1 \cdot F)+(w2 \cdot C)+(w3 \cdot P)$$

By adjusting the weights (w), I ensured that attacks targeting the gateway always trigger the highest priority alerts, while minor peer-to-peer noise stays in the low-risk category.

VI. CONCLUSION

As we know, there has been a rapid growth in public Wi-Fi, and it has outpaced the security habits of an average user, leaving a massive opening for cyber attacks. WiFiShield was developed by me to address this specific vulnerability and issue by offering a defence that is actually practical for daily use. I have made it possible for any user, regardless of their technical background, to understand the integrity of their connection by moving away from the enterprise-level, tense logs found in professional security tools and adopting a dynamic risk score.

The data from my experiments done in the sandbox confirms that WiFiShield is highly effective, with a detection accuracy of 95% and a response time of two seconds. WiFiShield can effectively identify ARP spoofing and Gateway Hijacks before an attacker can do significant damage. Also, I have made the tool very lightweight so that it has a minimal resource footprint and can stay active in the background of a laptop without the user even noticing it is running.

Future Work and Scalability

The current version of WiFiShield is great at identifying and alerting users to threats. However, for the future, I plan on moving towards a proactive defence model, and my future research will focus on three key areas:

- **Automated Response:** I am working on a module that will automatically "kill" the network connection or flush the ARP cache if a Critical Risk (Score 9-10) is detected, removing the need for the user to manually intervene during an attack.
- **Machine Learning Integration:** I plan to incorporate a lightweight ML model to better differentiate between normal network noise (like load-balancing routers) and "low-and-slow" spoofing attacks that might otherwise slip through.
- **Mobile Porting:** Since most public Wi-Fi usage happens on smartphones, my next major goal is to port the Python-based daemon to Android and iOS.

Ultimately, WiFiShield was created so that we could make public hotspots safer and give people an easy way to see what is happening on the network. It also shows that we do not have to trade usability for security.

Acknowledgment

I wish to express my sincere gratitude and appreciation to Dr. Simarjit Kaur for her valuable technical guidance, insightful critiques, and steadfast mentorship throughout the duration of my research. Her expertise provided the foundational perspective that was necessary to define the WiFiShield detection algorithms.

I also extend my thanks to the Department of Computer Science and Engineering (CSE) at Chitkara University With the help of the department's state-of-the-art laboratories and facilities, the computational resources were instrumental in establishing the controlled experiments required for the simulation of complex cyber attacks.

Lastly, I acknowledge the support of my peers and the academic community at Chitkara University, whose feedback helped shape the practical applications of the study, ensuring that WiFiShield is a user-centric solution for the rapidly evolving digital landscape.

REFERENCES

1. G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type," Phil. Trans. Roy. Soc. London, 1955.
2. M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.
3. S. Liu, "Wi-Fi Energy Detection Testbed," GitHub repository, 2023.

1.