Foundation University Journal of Engineering and Applied Sciences

FUJEAS
Vol. 6, Issue 1, 2025
DOI:10.33897/fujeas.v6i1.965

Research Article

SDN-based Intrusion Detection and Prevention System Against ARP Spoofing Attacks

Muhammad Junaid Khalid [a], **Syed Mushhad Mustuzhar Gilani** [b, *], **Asif Kabir** [c], **Qamar Nawaz** [b]

[a] UIIT- PMAS, Arid Agriculture University, Rawalpindi, Pakistan.
[b] Department of Computer Science, University of Agriculture, Faisalabad, Pakistan.
[c] Department of CS and IT, University of Kotli, Azad Jammu and Kashmir, Pakistan.

[*] **Corresponding author**: mushhad@uaf.edu.pk

Article Citation:
Khalid et al. (2025). "SDN-based Intrusion Detection and Prevention System Against ARP Spoofing Attacks". *Foundation University Journal of Engineering and Applied Sciences*
DOI:10.33897/fujeas.v6i1.965

Abstract:

Software Defined Networking (SDN) separates the control plane from the data plane, enabling centralized configuration through the SDN controller. While this centralization simplifies management, it also makes the controller's ARP table a critical target, as the stateless nature of ARP allows spoofing attacks. To mitigate this vulnerability, we propose an Intrusion Detection and Prevention System (IDPS) integrated as a controller module. The system monitors ARP and DHCP packets, maintaining a permanent ARP table synchronized with a DHCP table to ensure reliable IP–MAC bindings. The IDPS applies four validation checks in both IP and MAC address scanning modules, ensuring robust detection and prevention of spoofed packets. To achieve scalability, the design employs hashmaps for all lookups, ensuring that each check executes in constant time (O(1)), independent of network size. While this methodology introduces a higher baseline mitigation time (~2.2s) compared to some lightweight approaches, it guarantees predictable performance at scale and comprehensive coverage of spoofing attacks.

Keywords: Software Defined Network; Intrusion Detection; Intrusion Prevention; ARP Spoofing Attacks; Network Security; Cyber Security.

1. Introduction

Software Defined Networking (SDN) is a novel approach to computer networks. SDN separates the control plane from the data plane. This separation means removing the decision power from the networking devices [1], [2], [3] and giving it to the SDN controller. The separation of the control plane also provides the facility for the global configuration of the whole network. This means that there is no need to configure networking devices. Configure the Controller, and the entire network will follow those configurations [4], [5]. These configurations install the flow rules in the networking devices, but if some packet arrives with the data or headers that the flow rules cannot process, then this packet is forwarded to the SDN controller. Then the Controller decides and installs a new rule on the Open-Flow switches [6].

Industrial networks are more susceptible to cyber-attacks than home or personal networks, primarily because of the higher data value they possess. Unlike personal networks, industrial networks store and manage critical information for operational processes and production [7]. This includes proprietary data, intellectual property, and sensitive information related to industrial operations. Consequently, malicious actors target industrial networks to access and exploit this valuable data for financial gain, industrial espionage, or sabotage. The potential impact of such breaches on industrial operations and infrastructure amplifies the attractiveness of these networks as targets for cyber-attacks [8].

SDN also helps in monitoring the network traffic all at once. This helps prevent many attacks but also creates new vulnerabilities for some attacks. One of the most common attacks to intrude on the network is corrupting the Controller's Address Resolution Protocol (ARP) table. This means manipulating the IP address to MAC address and MAC address to IP address conversion. The major weakness of ARP protocols is that this is a stateless protocol, which means that this protocol does not verify the sender's authenticity or where the packet originated [9]. This means that the receiver cannot detect whether the data in the ARP packet is authentic and correct. This vulnerability of ARP is exploited very widely for intrusion into the network. In these types of attacks, the intruder sends the corrupted address(s) in an ARP REQUEST or ARP REPLY [10], [11].

The intrusion can occur through IP address spoofing, MAC address spoofing, or a combination of both. When both are spoofed, the attack becomes particularly dangerous, enabling undetectable man-in-the-middle scenarios where the adversary can intercept and manipulate traffic. In practice, IP spoofing may be carried out by forging IP headers, manipulating ARP/DHCP exchanges (e.g., rogue DHCP servers or forged ARP replies), or, though less common in modern deployments, by corrupting Reverse Address Resolution Protocol (RARP) messages to falsify IP–MAC bindings [12]. MAC spoofing, on the other hand, is typically achieved by altering the NIC's hardware address or injecting malicious ARP packets to poison ARP tables. While RARP is largely obsolete in contemporary networks, a comprehensive security approach must consider even legacy mechanisms as potential attack surfaces.

The Intrusion Detection System (IDS) only indicates an intrusion in the system. Intrusion Prevention Systems (IPS) actively change the network configurations and resources to prevent the network from intrusion. Combining them makes an intrusion detection and prevention system [13]. That will not only detect but also prevent the intrusion from happening.

For this reason, this research is concerned with not only detecting the intruder but also preventing them from doing any damage and ensuring that the intrusion will not happen again. The outcome of this research will be an Intrusion Detection and Prevention System (IDPS) that will filter the malicious ARP packets. The IDPS will be deployed as a module of the SDN controller instead of using it as an application, as shown in Figure 1. This will make the methodology much faster and more efficient. To get a better understanding of the IDPS, we consider different scenarios.

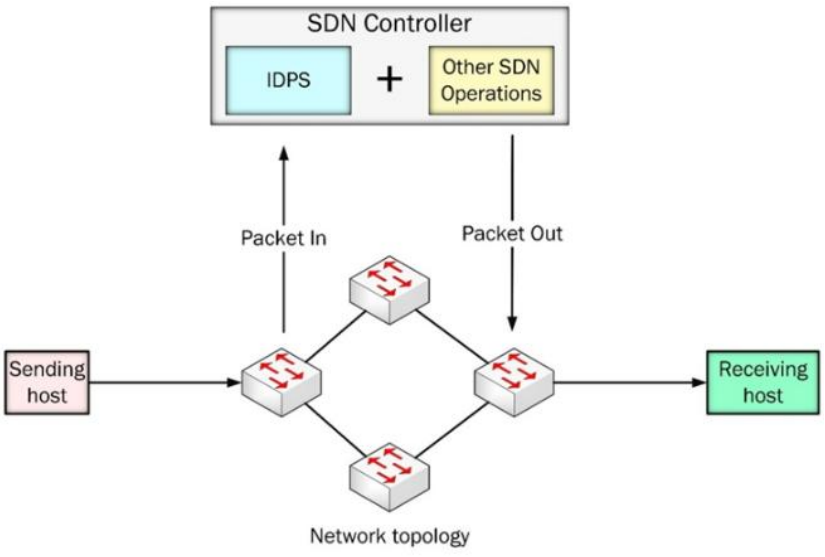

Figure 1: Abstract level diagram of proposed IDPS

1.1. Scenario 1

This scenario demonstrates how the proposed IDPS enforces blacklist rules within the SDN controller. Host 4, whose MAC address is recorded in the controller's blacklist, attempts to inject spoofed ARP packets to manipulate the ARP table. The IDPS intercepts these packets and validates the sender's identity using both the IP and MAC address scanning modules. Because the MAC address matches a blacklisted entry, the controller drops the packets before any ARP table update occurs. This confirms that the blacklist mechanism effectively prevents unauthorized hosts from participating in ARP-based communication and mitigates ARP cache poisoning attempts at the controller level.

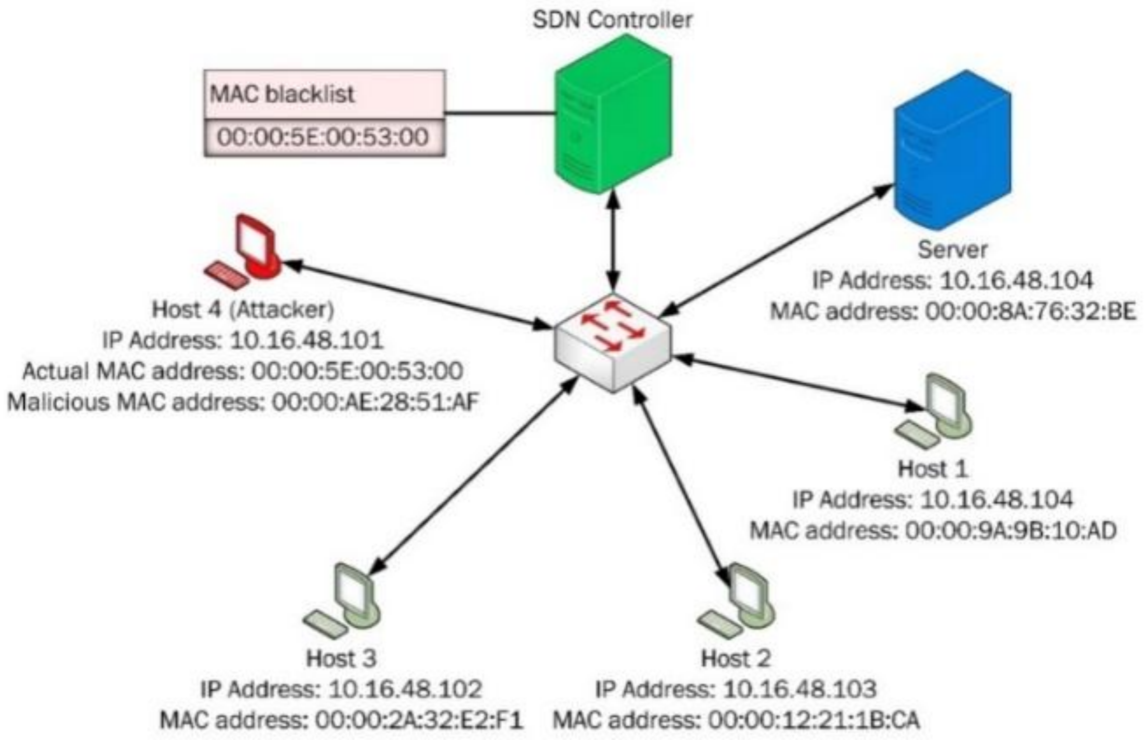

Figure 2: Scenario 1 – where the blacklisted user tries to intrude into the network

1.2. Scenario 2

This scenario evaluates the whitelist verification mechanism of the proposed IDPS. In this case, Host 4, whose MAC address is not included in the controller's whitelist, attempts to inject ARP packets to manipulate the controller's ARP table. The IP and MAC scanning modules validate each packet's source information against the whitelist, immediately discarding packets from unauthorized devices. This test confirms that the whitelist enforcement layer reliably blocks unknown or unregistered hosts at the controller level, preventing unauthorized participation in ARP communication and reinforcing network integrity.

1.3. Scenario 3

This scenario demonstrates the capability of the proposed IDPS to detect and mitigate Man-in-the-Middle (MitM) attacks. In this test, Host 4 impersonates a legitimate host by sending forged ARP replies to both the SDN controller and Host 2, attempting to intercept communication between them. The IDPS analyzes each ARP packet using cross-validation between the Ethernet frame headers, ARP payload fields, and DHCP table entries. Any mismatch in the IP–MAC bindings trigger an alert, and the packet is dropped before it reaches the network. This confirms that the IDPS effectively identifies and prevents MitM attempts by verifying multi-layer consistency within the controller's packet inspection process [14].

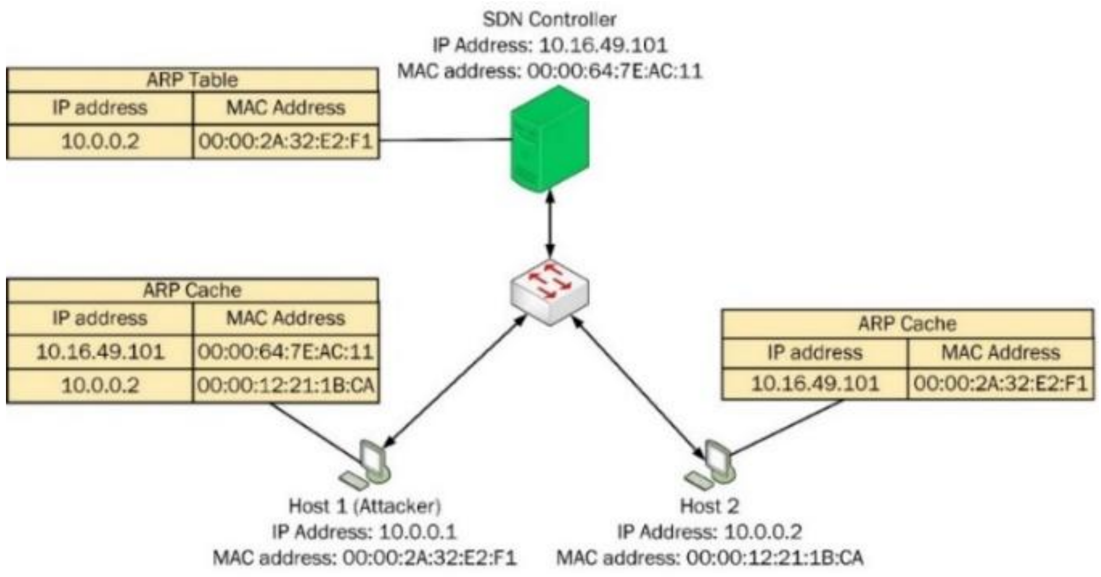

Figure 3: Scenario 2 – where an intruder is trying to intrude on a whitelisted network

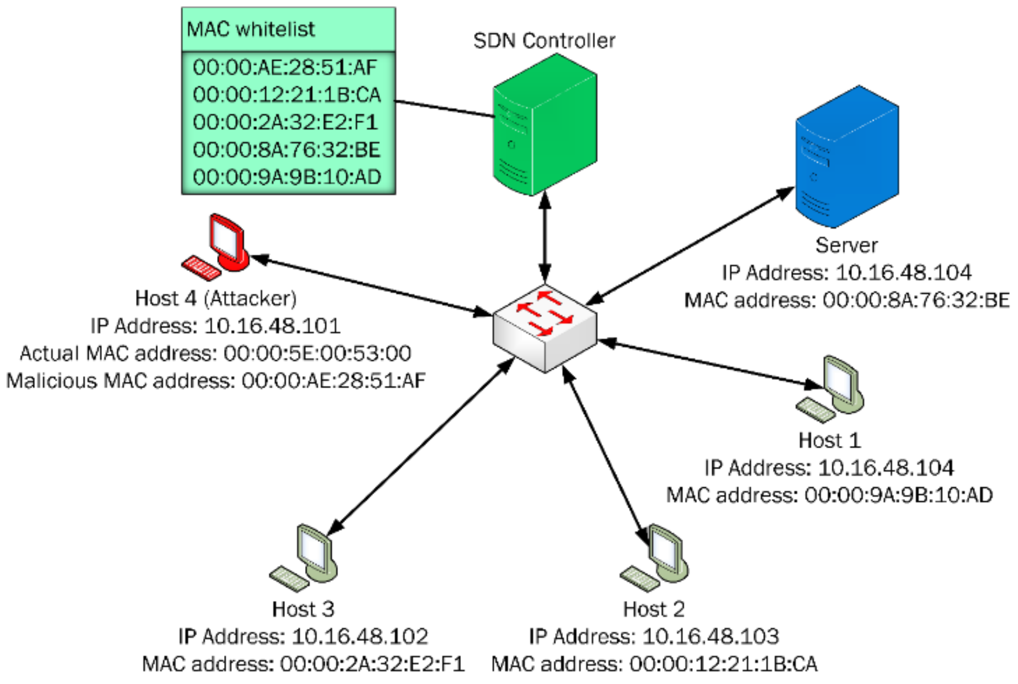

Figure 4: Scenario 3 – Man-in-the-middle attack

1.4. Contributions

These three scenarios are dangerous in industrial networks because the damage caused by intrusions in such environments can be severe and unpredictable [15]. This research focuses on minimizing this threat by proposing new rules and merging previous flow control rules at the SDN controller to filter out malicious ARP packets. While similar spoofing concepts could theoretically be applied using Reverse ARP (RARP) packets to falsify IP–MAC bindings, this paper concentrates specifically on ARP spoofing attacks, as RARP is largely obsolete in modern deployments.

This research proposes an Intrusion Detection and Prevention System (IDPS) that mitigates IP and MAC spoofing by countering the security weaknesses inherent in ARP’s stateless design, providing persistent IP–MAC bindings and verification checks to prevent cache poisoning.

2. Related Work

There is an approach [16] proposing a solution to combat ARP spoofing attacks, prevalent in identity fraud due to Address Resolution Protocol vulnerabilities, by leveraging software-defined networks and game theory principles. ARP spoofing involves inserting fake IP-MAC pairs into victims' ARP caches, potentially leading to severe cyber threats like denial of service or man-in-the-middle attacks. Traditional detection methods based on thresholds may falter against sophisticated attackers blending spoofed ARP packets with legitimate ones. To address this, the abstract suggests a game-theoretic approach where defenders anticipate attackers' moves using Nash equilibrium strategies. In this non-cooperative game, attackers aim to poison ARP caches, while defenders strive to prevent cache poisoning. The proposed method aims to achieve a mixed-strategy Nash equilibrium, identifying the optimal defensive approach. Additionally, it incorporates a player utility-based algorithm to identify and block malicious users' traffic or redirect them to honeypots. Simulation results indicate its superiority in preventing, detecting, and recovering from ARP spoofing attacks compared to existing literature methods, offering a comprehensive strategy to enhance system security against such threats.

Another IDPS proposed [17] against ARP spoofing attack; this IDPS matches the MAC addresses of the ARP packet and Ethernet packet in which the ARP packet was encapsulated. This technique has an excellent accuracy rate, but it can be improved if the current ARP table is also checked to see if the entry or the same IP or MAC address exists in the table.

An ARP spoofing attack mitigation protocol was [18]. They used the Naïve Bayes Algorithm's Efficient Bayes-Based Security Protocol (E2BaSeP). This protocol was designed by changing the formula of the Naïve Bayes algorithm and was used for malicious packet classification. They checked the current ARP table to determine whether the IP or MAC address from the incoming ARP packet already exists. The packet is dropped if an IP or MAC address pair exists in the ARP cache. Although this is an excellent technique, it can be improved by checking the MAC addresses of the incoming ARP packet and the Ethernet packet encapsulating the ARP packet.

A module was proposed to mitigate ARP spoofing attacks [19] that listens to DHCP signals, LINK UP and LINK DOWN signals, and ARP packets to verify the sender's authenticity. Their proposed module saves a copy of the DHCP data and then checks the data from the ARP packet against the DHCP packet. If the data matches the ARP packet data, then the packet is passed. Otherwise, it is dropped; their approach is very effective in detecting and preventing ARP spoofing attacks, but there is a need to check the wrapping Ethernet frame.

A topology listener was proposed [20] that keeps the record of the nodes in the network first check that applies is to check whether the IP addresses of the ARP packet belong to the network. If the packet is dropped, the IP-MAC table is checked for the entry from the ARP packet. If the entry is found first, the packet will be dropped, and then the node that originated the packet will be blacklisted for some fixed amount of time. Their approach is good and has excellent control over the ARP spoofing attack. However, this can be improved if they check the MAC addresses of the incoming ARP packet and the Ethernet frame encapsulating the ARP packet.

An approach for mitigating the ARP poisoning attacks was proposed. This approach works by adding a new module to the POX controller [21]. Their module checks the Ethernet packet wrapping the ARP packet. If the MAC address differs, then the packet is dropped, and the port from the switch is blocked for some time. The switch also saves a flow rule about the packet and drops all ARP packets with similar information. In their approach, no node is allowed to set the IP address manually. Only the DHCP server is used.

Another approach attempted to detect and mitigate ARP spoofing at the data plane [22]. For this purpose, they have used layer three switches using the SDN controller. In this approach, they are utilizing the intelligence capability of the layer three switch. They have also tried to mitigate ARP flood attacks by limiting the count of ARP packets in a constant period. This approach works fine, but there are other perspectives to check as well. For example, it is essential to check the Ethernet address and the MAC address of the ARP packets as well.

2.1. Findings From Literature Review

After studying all these studies closely, it can be observed that a minimal number of checks are applied. These leave at least one way behind to perform a successful ARP attack. The Functionality of IDPS can be extended to include more checks to cover the attack extensively.

The proposed IDPS will have all the missing checks and will check the impact on the mitigation time. The primary goal of the proposed IDPS is to increase the attack coverage and minimize the success chances of the IP or MAC address spoofing using the ARP spoofing methods. The proposed IDPS is also focused on minimizing the mitigation time.

3. Materials and Methods

3.1. Proposed IDPS

The proposed IDPS has two modules, one for checking IP addresses and the second for checking MAC addresses. Alongside these modules, the ARP cache at the SDN controller is turned into a permanent ARP table because cache memory is volatile and gets erased over time. The SDN controller will also behave as a DHCP controller, and only the DHCP IP will be able to communicate. For this purpose, a DHCP table is also maintained at the SDN controller. The flow rules for ARP and packets will never be installed at the SDN controller. To make sure that the ARP packets are redirected to the SDN controller. As soon as the new IP address is added ARP REQUEST packet will be sent to the new IP address. Then both IP and MAC address scanning Modules will scan the ARP REPLY packet. Similarly, the ARP entry will be deleted if an IP address gets disconnected, as shown in Figure 5.

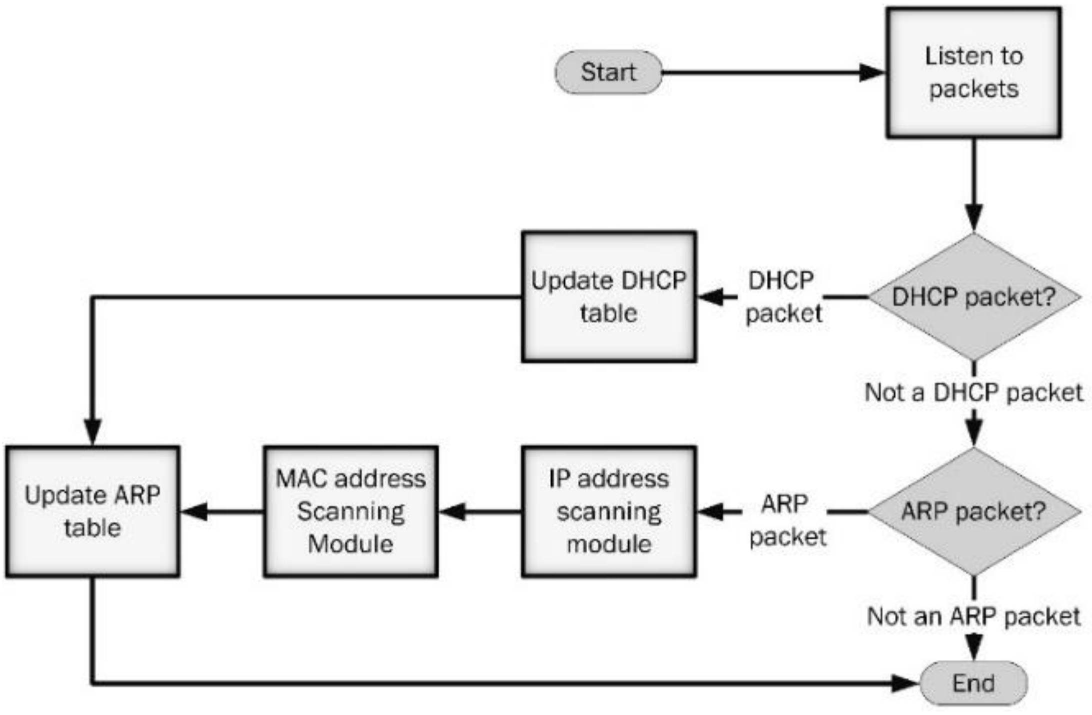

Figure 5: Architecture of the proposed Intrusion Detection and Prevention System

The IP and MAC address scanning modules work one after another and check the packet for possible malicious information from the ARP packets. If any of the modules drop the packet, the packet will be dropped by the SDN controller.

Three types of ARP spoofing attacks are mitigated using the methodology: ARP RE-QUEST attack, ARP REPLY attack, and ARP REPLY DESTINATION attack. ARP RE-QUEST attack means when the ARP REQUEST packet is corrupted, ARP REPLY attack is when the ARP REPLY has malicious information, and ARP REPLY DESTINATION attack is when the destination addresses in the ARP REPLY packet are corrupted.

3.1.1. IP Address Scan Module

This module will scan the IP address from the ARP packet using the following checks.

a) If the sender and receiver IP addresses on the ARP packet do not exist in
b) The DHCP table, then the packet will be dropped, but if it does exist, it will be forwarded
c) If the sender and receiver IP addresses exist with any other MAC addresses in the ARP table, then the packet will be dropped. Otherwise, it will be forwarded.
d) If the sender and receiver IP addresses are in the IP whitelist, then the packet will be processed; if it is not, it will be dropped.
e) The packet will be dropped if the sender and receiver IP addresses are on the IP blacklist. If it does not exist, then the packet will be forwarded.

The checks are also demonstrated in Figure 6. These checks will minimize the chances of IP address spoofing or the man-in-the-middle attack. IP and MAC spoofing create a danger that the attacker gets a complete IP-MAC disguise and becomes invisible. This invisibility can result in a considerable loss to the network owners.

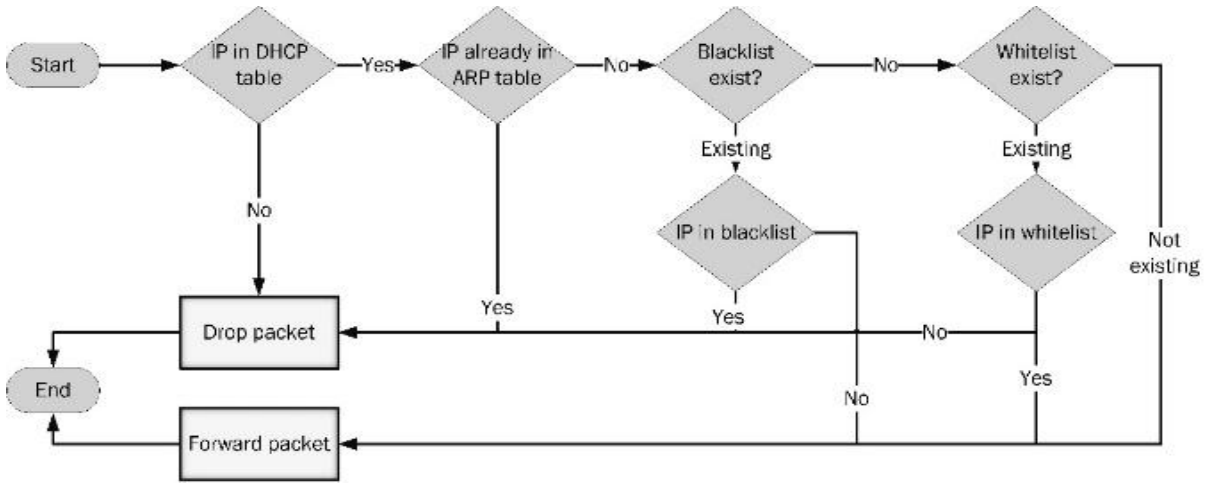

Figure 6: IP address scan module

3.1.2. MAC Address Scan Module

Similar to IP address scanning, this module will work for scanning the MAC addresses of the ARP packet and the Ethernet frame encapsulating it. The following are the checks.

a) If the sender and receiver MAC addresses on the ARP packet do not exist in the Ethernet frame, then the packet will be dropped, but if it do exist, it will be forwarded.
b) The packet will have dropped if the sender and receiver MAC addresses exist with any other IP addresses in the ARP table. Otherwise, it will be forwarded.
c) If the sender and receiver MAC addresses are in the MAC whitelist, then the packet will be processed; if it is not, it will be dropped.
d) The packet will be dropped if the sender and receiver MAC addresses are on the MAC blacklist. If it does not exist, then the packet will be forwarded.

The checks are also demonstrated in Figure 7. These checks will minimize the chances of MAC address spoofing and man-in-the-middle attacks, as discussed in scenarios 1, 2, and 3. Both modules work together to minimize the chances of intrusion into the network, making it almost impossible to intrude into the network using ARP spoofing and poisoning attacks.

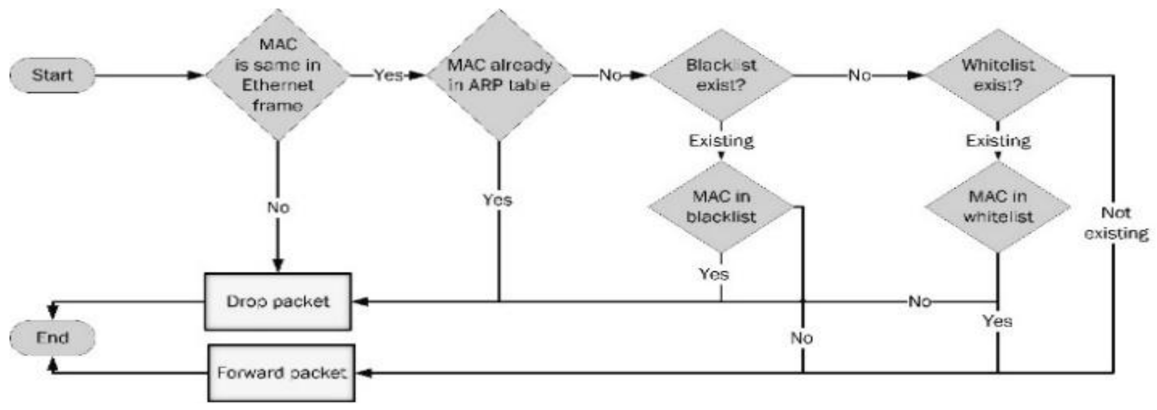

Figure 7: MAC address scan module

4. Results

4.1. Simulation Environment

The experiments were executed on a physical testbed using the POX SDN controller (Github commit ID: 5f82461) and the Mininet network emulator. Hardware and software details are listed to support reproducibility: an Intel Core i5-2520M CPU with 12 GB RAM running Ubuntu 20.04 LTS, Mininet version 2.2, and POX controller. Unlike Mininet's default in-process controller, we used a custom POX implementation deployed as a remote controller. Although both the Mininet simulator and the POX controller ran on the same physical machine, the controller was launched as a separate process and connected over the loopback interface (127.0.0.1:6633). This setup reflects a deployment model closer to production SDN environments, where the controller is logically external to the data plane.

For initial validation, we adopted a 5-host topology (Figure 8) to demonstrate the functionality of the proposed IDPS. Host roles are defined in Table 1: H1 (attacker), H2–H4 (clients), and H5 (server). The attacker (H1) was equipped with the arpspoof tool from the dsniff suite and driftnet for packet interception. Background "normal" traffic was generated using iperf3 TCP flows and periodic HTTP requests to H5. Flow rules for ARP packets were not installed directly in switches, ensuring that all ARP traffic was redirected to the controller for validation.

To assess scalability, the topology was subsequently expanded in increments of five hosts per iteration, reaching up to 100 nodes. Across these larger topologies, the observed mitigation time did not increase

Table 1: IP configuration

Host	**IP address**
H1 (Attacker)	10.0.0.1
H2	10.0.0.2
H3	10.0.0.3
H4	10.0.0.4
H5 (Server)	10.0.0.5
SDN controller (Remote)	10.16.19.101

significantly, confirming that the O(1) hashmap-based lookups provide predictable performance regardless of network size.

The implementation is done by monitoring the PacketIn events. The PacketIn event occurs when the packet has no flow control entry, and the packet id is forwarded to the SDN controller. The first rule embedded on the switch is the no-flow rule for ARP and the DHCP packets. This will ensure that all ARP and DHCP packets are forwarded to the Controller. This makes the Controller the data-gathering point. After the packet is received, its type will be checked: if it is an ARP REPLY packet, then the checks discussed above will be applied, and the packet's fate will be decided. If it is an ARP REQUEST packet, the checks will be applied to the packet, and the ARP REPLY packet will be generated on the SDN controller using the Scapy Python Library (version 2.4.5). If it is a DHCP LINK UP or DHCP LINK DOWN packet, then the DHCP and ARP tables will be updated.

The IDPS applies a fixed, two-stage verification pipeline to every ARP Packet-In event to ensure consistent and predictable processing latency. When an ARP packet is received by the controller, the packet type is first identified (ARP REQUEST or ARP REPLY). In both cases, the packet then proceeds through the IP Address Scan Module, followed immediately by the MAC Address Scan Module (i.e., the modules execute sequentially). For an ARP REQUEST, the controller performs the two-stage validation on the incoming request and, if the checks pass, generates and sends an ARP REPLY on behalf of the legitimate host; for an ARP REPLY, the controller validates the reply using the same sequential modules and installs or updates the permanent ARP table only if all checks succeed. This strict two-stage ordering (IP → MAC) ensures a fixed number of O(1) lookups per packet and makes the mitigation time largely independent of network size; small timing variations observed in experiments are attributable to background bandwidth usage rather than to increased lookup complexity.

4.1.1. ARP REQUEST Attack

For an ARP REQUEST attack, the IP addresses of both the sender and receiver and the MAC address of the sender are validated using the Ethernet frame, blacklist/whitelist, ARP table, and DHCP table, since an ARP REQUEST does not contain the receiver's MAC information. The results are presented in Figure 8. While the mitigation time averages around 2.18 s on our testbed, the key observation is that the time remains consistent across different network sizes, with only minor fluctuations attributable to background bandwidth usage.

Table 2 and Figure 8 compare these results with prior approaches. Although some works, such as Darwesh et al., report substantially lower mitigation times (150–170 ms), these methods typically implement only two verification checks, whereas our proposed IDPS performs four sequential checks for every ARP packet. This broader coverage necessarily incurs higher baseline latency but eliminates the bypass opportunities left open in lighter approaches.

Table 2: Mitigation time and cost of ARP REQUEST attack

Approach/References	Average Mitigation time	Whitelist or Blacklist check	Ethernet frame and ARP packet check	ARP table check	DHCP table check
Mvah et al. [16]	1.961 s	No	No	Yes	No
Girdler & Vassilakis [17]	2.204 s	Yes	Yes	No	No
Tchendi et al. [18]	1.739 s	No	No	Yes	No
Rangisetti et al. [19]	1.812 s	No	No	No	Yes
Aldabbas & Amin [20]	2.264 s	No	No	Yes	Yes
Darwesh et al. [23]	150 ms	No	Yes	No	Yes
Proposed approach	**2.183 s**	**Yes**	**Yes**	**Yes**	**Yes**

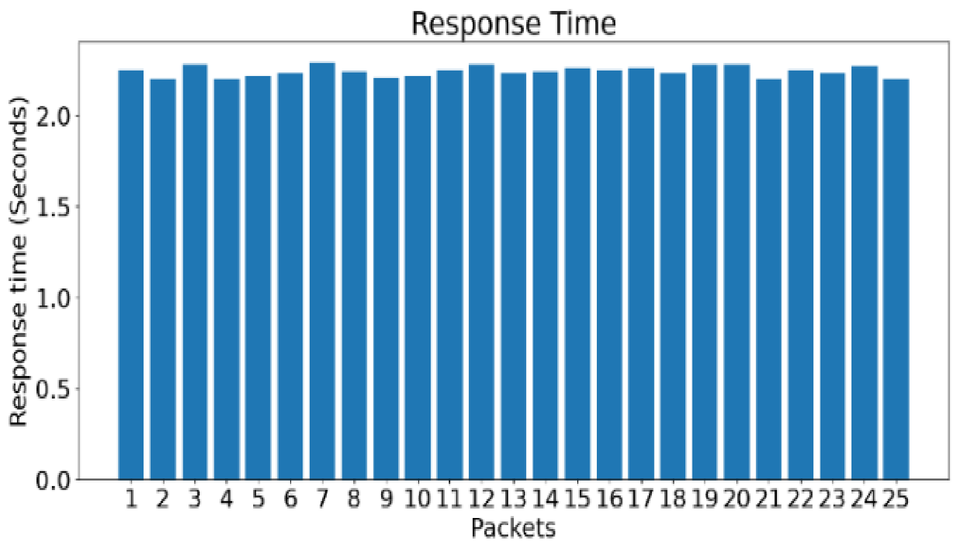

Figure 8: Mitigation time of ARP REQUEST attack for up to 25 ARP packets

4.1.2. ARP REPLY and ARP REPLY DESTINATION Attack

In the case of ARP REPLY attacks, a malicious host may attempt to inject forged replies either in response to legitimate ARP REQUESTs or as unsolicited packets. In our implementation, the controller itself also generates ARP REQUESTs (using the Scapy library in Python) when a new IP is assigned, and malicious ARP REPLYs can attempt to exploit this exchange as well. To defend against such attacks, the proposed IDPS validates every incoming ARP REPLY packet through the following checks:

a) DHCP table verification: confirm that the claimed IP–MAC binding exists in the DHCP lease table.

b) ARP table consistency: if an entry already exists, compare the incoming IP–MAC pair with the stored binding and drop the packet if a mismatch is detected (possible spoofing).

c) Ethernet–ARP MAC consistency: compare the MAC address in the Ethernet frame with the MAC in the ARP payload.

Only if all checks pass is the ARP REPLY accepted and used to update the permanent ARP table. This ensures robustness against both unsolicited ARP REPLY injections and forged responses to controller- or host-initiated ARP REQUESTs. Results for these experiments are shown in Figure 9.

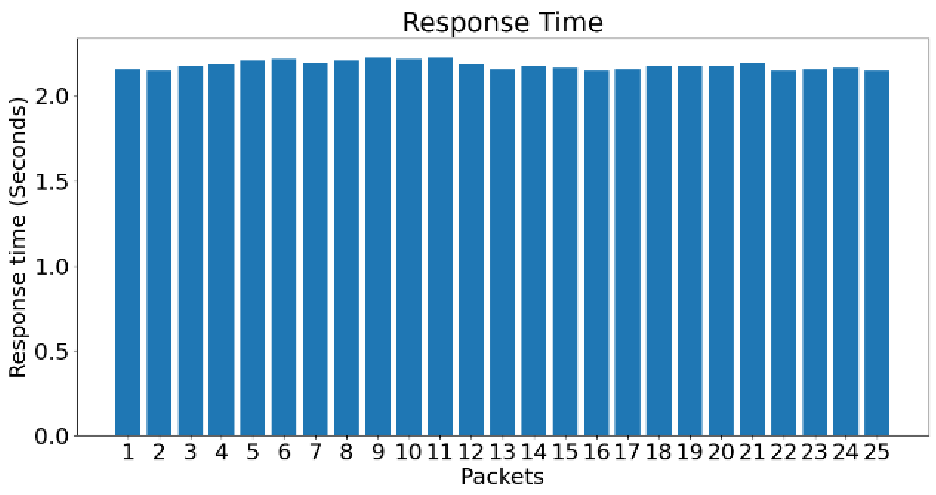

Figure 9: ARP REPLY and ARP REPLY DESTINATION attack mitigation time up to 25 ARP packets

Like the ARP REQUEST attack, ARP REPLY and ARP REPLY DESTINATION attacks also have a stable response time. The average mitigation time and its comparison with previous approaches are given in Table 3.

Table 3: Mitigation time and cost for ARP REPLY and ARP REPLY DESTINATION attack

Approach/References	Average Mitigation Time	Whitelist or Blacklist Check	Ethernet Frame and ARP Packet Check	ARP Table Check	DHCP Table Check
Girdler & Vassilakis [17]	2.202 s	Yes	Yes	No	No
Mvah et al. [16]	1.961 s	No	No	Yes	No
Tchendi et al. [18]	2.071 s	No	No	Yes	No
Rangisetti et al. [19]	1.812 s	No	No	No	Yes
Aldabbas & Amin [20]	2.264 s	No	No	Yes	Yes
Darwesh et al.[23]	170 ms	No	Yes	No	Yes
Proposed approach	**2.242 s**	**Yes**	**Yes**	**Yes**	**Yes**

The mitigation time, as shown in Table 4, averages around 2.2 seconds for the proposed IDPS. While this is slower than lightweight approaches such as Darwesh et al. (150–170 ms), the difference arises from our use of four sequential verification checks instead of only two. The key contribution is not raw mitigation speed but the predictable performance and stability of our method: the number of checks remains constant regardless of network size, and efficient use of hashmaps ensures O(1) lookups even as the number of nodes scales up. This design guarantees that the mitigation time does not grow with the size of the network, making the approach practical for large-scale deployments.

The proposed IDPS is particularly suited for industrial networks where security is prioritized over minimal latency. For real-world adoption, the system can be deployed as a module of the SDN controller, requiring only integration with DHCP services and standard ARP handling rules. Deployment at scale would involve provisioning sufficient controller resources, validating the system under representative industrial traffic patterns, and integrating with existing monitoring and logging frameworks. This ensures not only detection and mitigation of ARP spoofing but also operational feasibility in industrial network environments where reliability and predictability are paramount.

4.2. Performance Analysis

4.2.1. Implementation Performance Analysis

The implementation of the proposed approach is performed using HashMaps instead of using arrays and lists. The reason behind using HashMaps is that they have O(1) read and write time complexity. We have used three hashmaps.

First, to store IPs, have a structure of {"IP", Boolean} while adding the IP in the table, the value against it is set to true. And when querying the hashmaps, if the IP does not exist, the false value will be returned by default.

The second one is to store MACs against the IP addresses, which means the structure of this hashmap is {"IP","MAC"}. When an ARP REPLY packet is received at the controller, and all the checks have been applied, and it is sure that the packet is not malicious, then the IP, MAC pair will be added. If there comes an IP that does not exist in the ARP table, then there will be a null value indicating that the pair is malicious.

The third one is to store IPs against MAC addresses, which means the structure of this hashmap is {"MAC","IP"}. When an ARP REPLY packet is received at the controller, and all the checks have been applied, and it is sure that the packet is not malicious, then the MAC, IP pair will be added. If there comes an MAC that does not exist in the ARP table, then there will be a null value indicating that the pair is malicious.

4.2.2. Network Performance Analysis

The network performance was not affected, as only ARP packets are checked and all other packets are forwarded as usual. Secondly, as the implementation is based on hashmaps, the network size does affect the performance of the mitigation strategy. This approach also does not affect the throughput of the network. To verify this, we have performed an extensive experiment. For the extensive experiment, we extended our topology up to 100 nodes by incrementing the 5 nodes in each iteration. In every iteration, one attacker was being added, i.e., in 5 nodes there is 1 attacker, in 10 nodes there are 2 attackers, in 15 nodes there are 3 attackers, and so on up to 100 nodes with 20 attackers. We then monitored the performance, and no significant difference was found in attack mitigation time, as shown in Figures 10 and 11.

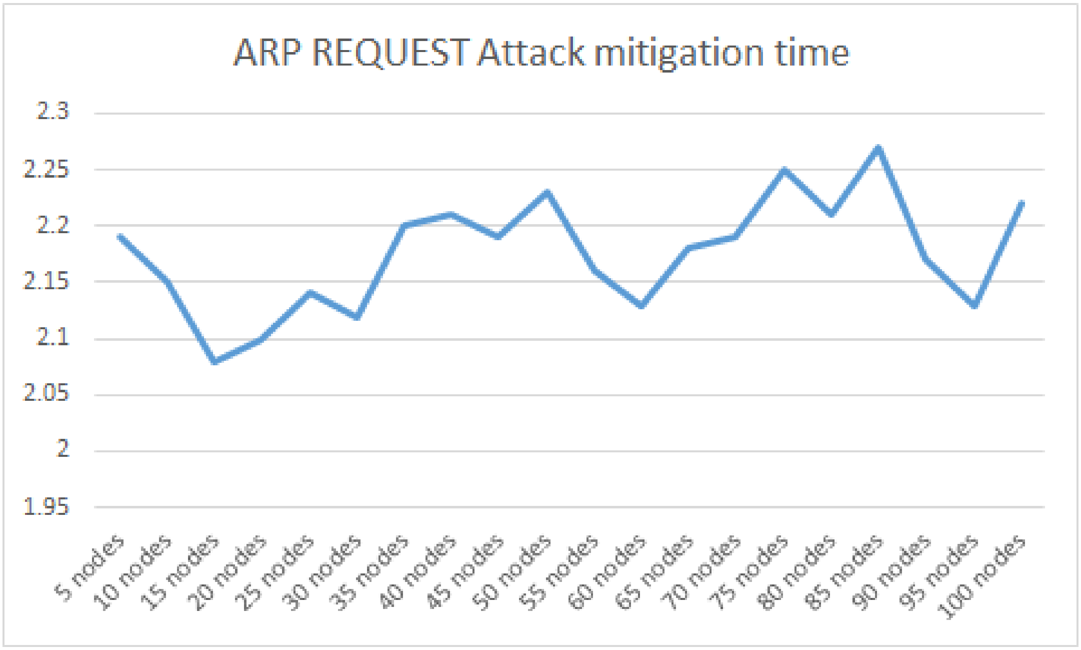

Figure 10*:* ARP REQUEST attack mitigation time with up to 100 nodes in the network

If we see the graphs shown in Figures 10 and 11, we can see that the values lie between 2.0 and 2.3 seconds. Thus, it reveals that there is no significant difference that appeared with the increase in the number of nodes in the network.

4.3. Robustness Analysis

The robustness of the proposed Intrusion Detection and Prevention System (IDPS) was evaluated by monitoring its behavior in a network environment containing both normal traffic and malicious ARP packets. The following subsections outline the experimental setup, observations, and significance of the results.

4.3.1. Experimental Test

The testing environment comprised a simulated SDN network with 5 hosts, configured as described in Section 4.1. Normal traffic, including legitimate ARP packets, coexisted with malicious ARP packets

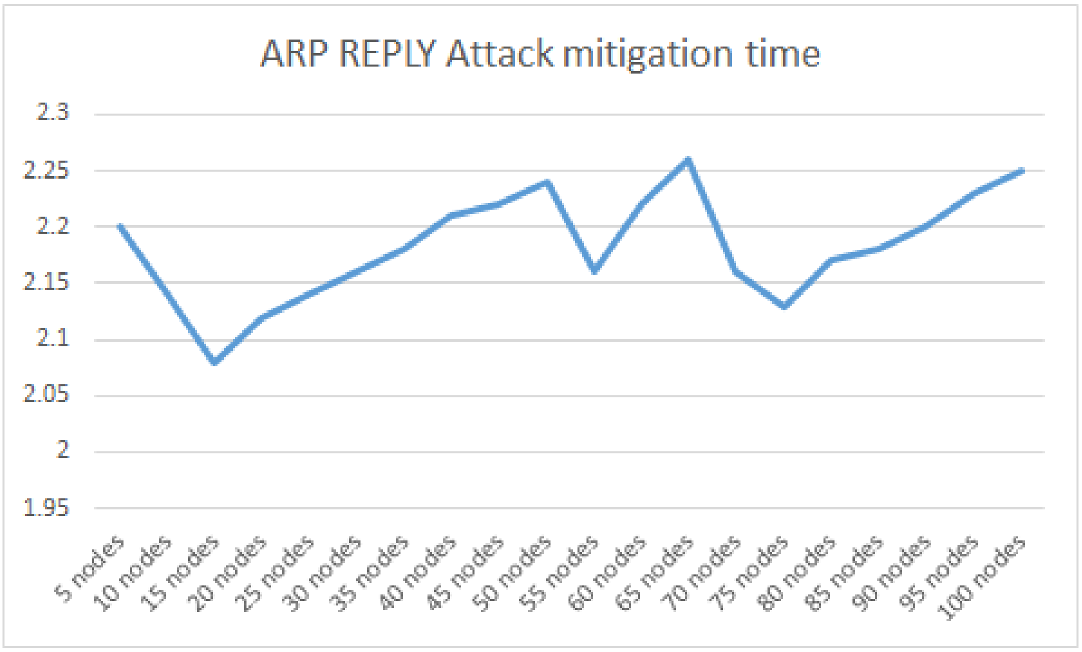

Figure 11: ARP REPLY attack mitigation time with up to 100 nodes in the network

generated by an attacker using the arpspoof tool. The legitimate traffic adhered to standard communication protocols, while the attacker attempted to execute ARP spoofing attacks, including ARP REQUEST and ARP REPLY DESTINATION attacks.

To ensure a realistic evaluation, normal traffic was maintained at typical levels, with periodic bursts of ARP communication. Legitimate ARP packets passed through all the checks implemented in the IP and MAC address scanning modules, while malicious packets were crafted to exploit known ARP vulnerabilities.

4.3.2. Observations

During the testing phase, the Intrusion Detection and Prevention System (IDPS) demonstrated its effectiveness in handling legitimate network traffic without any disruptions. No legitimate ARP packets were dropped, ensuring that all such packets traversed the network seamlessly, with no interruptions or degradation in communication quality.

In terms of malicious activity, the IDPS excelled in detecting and mitigating attacks. Every malicious ARP packet introduced into the network was promptly detected and dropped by the system. Remarkably, no false negatives were observed, guaranteeing that all attack attempts were thwarted effectively and preventing any harm to the network.

The IDPS's checks were specifically focused on DHCP and ARP packets, meaning that general network traffic, including non-ARP communication, was left unaffected. This isolation of traffic types ensured that the overall throughput and latency of the network remained at optimal levels, preserving the performance of the system while maintaining robust security measures. The performance of the IDPS in handling legitimate and malicious ARP packets is summarized in Table 4.

The results demonstrate the robustness of the proposed IDPS in distinguishing between legitimate and malicious ARP packets with 100% accuracy. The deterministic checks ensure that the system operates without false positives or false negatives under the tested conditions.

Moreover, the isolation of these checks to DHCP and ARP packets minimizes the impact on general network traffic, making the IDPS a practical and scalable solution for industrial environments. This level of robustness is critical in ensuring uninterrupted operations in networks vulnerable to ARP spoofing

Table 4: ARP packet handling and detection metrics

Metric	Observed Value
Total ARP Packets	40
Malicious ARP Packets	25
Legitimate ARP packets	15
Legitimate packets dropped	0
Malicious packets dropped	25

attacks.

5. Conclusions and Future Direction

Intrusions in industrial networks are highly disruptive, with consequences ranging from operational shutdowns to severe data breaches. While previous ARP spoofing defenses show good performance, their reliance on minimal checks leaves exploitable gaps. The proposed IDPS addresses this limitation by introducing a comprehensive verification process across DHCP tables, ARP tables, Ethernet frames, and whitelist/blacklist mechanisms, thereby closing common bypass paths. Although the average mitigation time of ~2.2 seconds is higher than lightweight methods, such as Darwesh et al., it remains consistent across topologies scaling up to 100 nodes. This stability, combined with broader attack coverage, makes the system a robust and scalable defense mechanism for industrial network environments where reliability and resilience are paramount.

It is acknowledged that legitimate ARP packets must undergo all implemented checks, which introduces a slight latency overhead for these packets. However, this overhead does not affect general network traffic, as the checks are applied only to ARP and DHCP packets. A more detailed evaluation under varied network loads will be conducted in future work to validate that this remains negligible under real-world conditions. Future enhancements will focus on reducing the latency associated with legitimate packet handling while maintaining detection accuracy. Potential directions include optimizing the order of checks to eliminate redundancy, enabling parallel execution of the IP and MAC verification modules within the SDN controller, and exploring hardware-assisted acceleration, such as GPU offloading in high-throughput environments. These approaches will be evaluated with consideration of their resource requirements and architectural implications. Rather than seeking to make ARP spoofing entirely impossible, the objective is to further reduce the likelihood and impact of successful attacks, thereby strengthening the resilience of the proposed IDPS in dynamic industrial network environments.

Acknowledgment

We are thankful to the research reviewers who helped in writing and improving these guidelines.

Conflict of Interests

Publication of this research article has no conflict of interest.

Funding Statement

This research article is not a product of a funded research project.

6. References

[1] S. H. Haji *et al.*, "Comparison of Software Defined Networking with Traditional Networking," *Asian J. Res. Comput. Sci.*, pp. 1–18, May 2021, doi: 10.9734/ajrcos/2021/v9i230216.

[2] R. Fazal, S. M. M. Gilani, and M. J. Khalid, "SD-ALB: Software Defined Adaptive load balancing in Data Center Network," *KIET J. Comput. Inf. Sci.*, vol. 6, no. 2, pp. 51–67, Aug. 2023, doi: 10.51153/kjcis.v6i2.181.

[3] S. M. M. Gilani, T. Hong, W. Jin, G. Zhao, H. M. Heang, and C. Xu, "Mobility management in IEEE 802.11 WLAN using SDN/NFV technologies," *EURASIP J. Wirel. Commun. Netw.*, vol. 2017, no. 1, p. 67, Apr. 2017, doi: 10.1186/s13638-017-0856-9.

[4] P. Manso, J. Moura, and C. Serrão, "SDN-Based Intrusion Detection System for Early Detection and Mitigation of DDoS Attacks," *Information*, vol. 10, no. 3, Art. no. 3, Mar. 2019, doi: 10.3390/info10030106.

[5] H. MengHeang, S. M. Gilani, T. Hong, G. Zhao, and H. B. Abdalla, "Load Balancing in Wireless Networks using SDN-enabled Infrastructure: Traffic Analysis," *In 10th EAI International Conference on Mobile Multimedia Communications*, Dec. 2017, pp. 34–40. Accessed: Mar. 25, 2024. [Online]. Available: https://eudl.eu/doi/10.4108/eai.13-7-2017.2270385

[6] J. C. Correa Chica, J. C. Imbachi, and J. F. Botero Vega, "Security in SDN: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 159, p. 102595, June 2020, doi: 10.1016/j.jnca.2020.102595.

[7] D. Bruschi, A. Di Pasquale, A. Lanzi, and E. Pagani, "Ensuring cybersecurity for industrial networks: A solution for ARP-based MITM attacks," *J. Comput. Secur.*, vol. Preprint, no. Preprint, pp. 1–29, Jan. 2024, doi: 10.3233/JCS-230023.

[8] N. G. Camacho, "The Role of AI in Cybersecurity: Addressing Threats in the Digital Age," *J. Artif. Intell. Gen. Sci. JAIGS ISSN3006-4023*, vol. 3, no. 1, Art. no. 1, Mar. 2024, doi: 10.60087/jaigs.v3i1.75.

[9] O. Yurekten and M. Demirci, "SDN-based cyber defense: A survey," *Future Gener. Comput. Syst.*, vol. 115, pp. 126–149, Feb. 2021, doi: 10.1016/j.future.2020.09.006.

[10] A. N. Alhaj and N. Dutta, "Analysis of Security Attacks in SDN Network: A Comprehensive Survey," in *Contemporary Issues in Communication, Cloud and Big Data Analytics*, H. K. D. Sarma, V. E. Balas, B. Bhuyan, and N. Dutta, Eds., Singapore: Springer, 2022, pp. 27–37. doi: 10.1007/978-981-16-4244-9_3.

[11] J. Xie *et al.*, "A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges," *IEEE Commun. Surv. Tutor.*, vol. 21, no. 1, pp. 393–430, 2019, doi: 10.1109/COMST.2018.2866942.

[12] Z. Shah and S. Cosgrove, "Mitigating ARP Cache Poisoning Attack in Software-Defined Networking (SDN): A Survey," *Electronics*, vol. 8, no. 10, Art. no. 10, Oct. 2019, doi: 10.3390/electronics8101095.

[13] N. Mazhar, R. Salleh, M. A. Hossain, and M. Zeeshan, "SDN based Intrusion Detection and Prevention Systems using Manufacturer Usage Description: A Survey," *Int. J. Adv. Comput. Sci. Appl. IJACSA*, vol. 11, no. 12, Art. no. 12, 59/31 2020, doi: 10.14569/IJACSA.2020.0111283.

[14] N. Saritakumar, K. v Anusuya, and B. Balasaraswathi, "Detection and Mitigation of MITM Attack in Software Defined Networks," *In Proceedings of the First International Conference on Combinatorial and Optimization (ICCAP) 2021*, December 7-8, 2021, Chennai, India, Dec. 2021. Accessed: Mar. 25, 2024. [Online]. Available: https://eudl.eu/doi/10.4108/eai.7-12-2021.2314735

[15] V. Rohatgi and S. Goyal, "A Detailed Survey for Detection and Mitigation Techniques against ARP Spoofing," in *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, India: IEEE, Oct. 2020, pp. 352–356. doi: 10.1109/I-SMAC49090.2020.9243604.

[16] F. Mvah, V. Kengne Tchendji, C. Tayou Djamegni, A. H. Anwar, D. K. Tosh, and C. Kamhoua, "GaTeBaSep: game theory-based security protocol against ARP spoofing attacks in software-defined networks," *Int. J. Inf. Secur.*, vol. 23, no. 1, pp. 373–387, Feb. 2024, doi: 10.1007/s10207-023-00749-0.

[17] T. Girdler and V. G. Vassilakis, "Implementing an intrusion detection and prevention system using Software-Defined Networking: Defending against ARP spoofing attacks and Blacklisted MAC Addresses," *Comput. Electr. Eng.*, vol. 90, p. 106990, Mar. 2021, doi: 10.1016/j.compeleceng.2021.106990.

[18] V. K. Tchendji, F. Mvah, C. T. Djamegni, and Y. F. Yankam, "E2BaSeP: Efficient Bayes Based Security Protocol Against ARP Spoofing Attacks in SDN Architectures," *J. Hardw. Syst. Secur.*, vol. 5, no. 1, pp. 58–74, Mar. 2021, doi: 10.1007/s41635-020-00105-x.

[19] A. K. Rangisetti, R. Dwivedi, and P. Singh, "Denial of ARP spoofing in SDN and NFV enabled cloud-fog-edge platforms," *Clust. Comput.*, vol. 24, no. 4, pp. 3147–3172, Dec. 2021, doi: 10.1007/s10586-021-03328-x.

[20] H. Aldabbas and R. Amin, "A novel mechanism to handle address spoofing attacks in SDN based IoT," *Clust. Comput.*, vol. 24, no. 4, pp. 3011–3026, Dec. 2021, doi: 10.1007/s10586-021-03309-0.

[21] D. G, V. A.a, and K. V.m, "AN EFfiCIENT MECHANISM TO DETECT AND MITIGATE AN ARP SPOOfiNG ATTACK IN SOFTWARE-DEfiNED NETWORKS," *Научно-Технический Вестник Информационных Технологий Механики И Оптики*, vol. 21, no. 3, pp. 401–409, 2021.

[22] S. Buzura, M. Lehene, B. Iancu, and V. Dadarlat, "An Extendable Software Architecture for Mitigating ARP Spoofing-Based Attacks in SDN Data Plane Layer," *Electronics*, vol. 11, no. 13, Art. no. 13, Jan. 2022, doi: 10.3390/electronics11131965.

[23] G. Darwesh, A. a. Vorobeva, and V. m. Korzhuk, “An efficient mechanism to detect and mitigate an ARP spoofing attack in software-defined networks,” *Научно-Технический Вестник Информационных Технологий Механики И Оптики*, vol. 21, no. 3, Art. no. 3, 2021.