



OPEN ACCESS

EDITED BY

Eduard Babulak,
National Science Foundation (NSF),
United States

REVIEWED BY

Shahzad Ashraf,
DHA Suffa University, Pakistan
Shitharth Selvarajan,
Leeds Beckett University, United Kingdom

*CORRESPONDENCE

Anis Yazidi
✉ anisy@oslomet.no

RECEIVED 07 August 2024

ACCEPTED 07 October 2024

PUBLISHED 30 October 2024

CITATION

Karmous N, Ben Dhiab Y, Ould-Elhassen
Aoueileyine M, Youssef N, Bouallegue R and
Yazidi A (2024) Deep learning approaches for
protecting IoT devices in smart homes from
MitM attacks. *Front. Comput. Sci.* 6:1477501.
doi: 10.3389/fcomp.2024.1477501

COPYRIGHT

© 2024 Karmous, Ben Dhiab, Ould-Elhassen
Aoueileyine, Youssef, Bouallegue and Yazidi.
This is an open-access article distributed
under the terms of the [Creative Commons
Attribution License \(CC BY\)](#). The use,
distribution or reproduction in other forums is
permitted, provided the original author(s) and
the copyright owner(s) are credited and that
the original publication in this journal is cited,
in accordance with accepted academic
practice. No use, distribution or reproduction is
permitted which does not comply with
these terms.

Deep learning approaches for protecting IoT devices in smart homes from MitM attacks

Nader Karmous¹, Yasmine Ben Dhiab¹,
Mohamed Ould-Elhassen Aoueileyine¹, Neji Youssef¹,
Ridha Bouallegue¹ and Anis Yazidi^{2,3*}

¹Innov'COM Laboratory, Higher School of Communication of Tunis, University of Carthage, Tunis, Tunisia, ²Department of Computer Science, OsloMet–Oslo Metropolitan University, Oslo, Norway, ³Department of Plastic and Reconstructive Surgery, Oslo University Hospital, Oslo, Norway

The primary objective of this paper is to enhance the security of IoT devices in Software-Defined Networking (SDN) environments against Man-in-the-Middle (MitM) attacks in smart homes using Artificial Intelligence (AI) methods as part of an Intrusion Detection and Prevention System (IDPS) framework. This framework aims to authenticate communication parties, ensure overall system and network security within SDN environments, and foster trust among users and stakeholders. The experimental analysis focuses on machine learning (ML) and deep learning (DL) algorithms, particularly those employed in Intrusion Detection Systems (IDS), such as Naive Bayes (NB), k-Nearest Neighbors (kNN), Random Forest (RF), and Convolutional Neural Networks (CNN). The CNN algorithm demonstrates exceptional performance on the training dataset, achieving 99.96% accuracy with minimal training time. It also shows favorable results in terms of detection speed, requiring only 1 s, and maintains a low False Alarm Rate (FAR) of 0.02%. Subsequently, the proposed framework was deployed in a testbed SDN environment to evaluate its detection capabilities across diverse network topologies, showcasing its efficiency compared to existing approaches.

KEYWORDS

Man-in-the-Middle, ARP spoofing, Software-Defined Networking, machine learning, Internet of Things, smart home, cybersecurity, intrusion detection system

1 Introduction

In the research papers by [Rostami and Goli-Bidgoli \(2024\)](#) and [Karmous et al. \(2023\)](#), within an SDN configuration featuring an IoT Message Queue Telemetry Transport (MQTT) broker system for smart homes, the MQTT broker typically connects to IoT devices and communicates with them. As illustrated in [Figure 1](#), the SDN controller and switches manage the network infrastructure and direct traffic flows. The MQTT broker is a central component in an IoT network that facilitates communication between IoT devices. IoT devices, including sensors, connect to the MQTT broker to publish a messages to hosts subscriber. The MQTT broker doesn't typically connect directly to SDN switches or the SDN controller. The MQTT broker typically connects to IoT devices and communicates with them. The MQTT protocol utilizes packets with minimized payload overhead and offers different Quality of Service (QoS) levels while reducing energy consumption, as demonstrated by [Toldinas et al. \(2019\)](#). This effectively addresses the challenges associated with the HTTP protocol, as highlighted by [Wukkadada et al. \(2018\)](#). This makes MQTT the optimal choice for future use in IoT devices exchanging data.

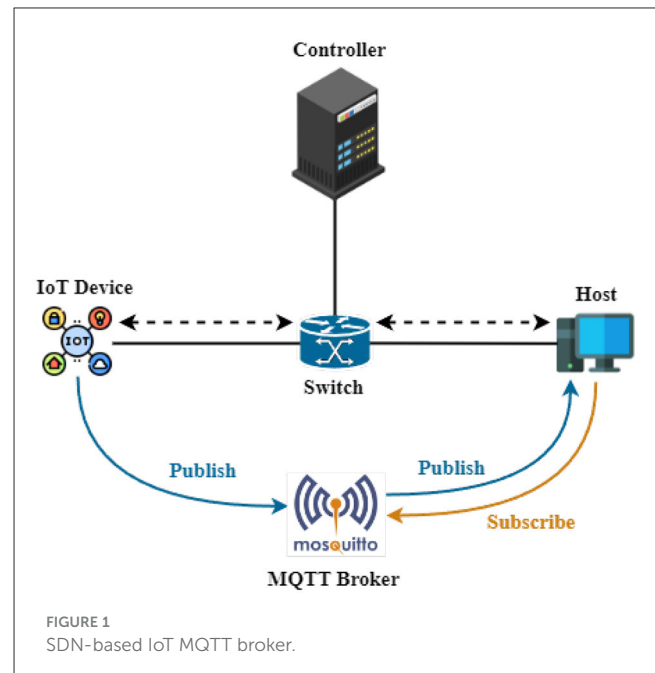
However, as these interconnections are vulnerable to attacks, there is a risk of experiencing denial-of-service (DoS) attacks, as indicated by the research of Karmous et al. (2024) and Bârli et al. (2021). Moreover, they can be exploited to gain full control through MitM attacks when targeting the endpoints, as noted by Conti et al. (2016). MitM attacks pose a significant threat, particularly to IoT devices within an SDN. In an SDN-based IoT ecosystem, a potential attacker could exploit vulnerabilities in the SDN infrastructure, such as compromised switches, to intercept communication between the host subscriber and the IoT publisher. The attacker positions themselves between the host subscriber and the IoT publisher, thereby gaining the ability to eavesdrop on or manipulate the data flowing between them. The main contribution of this paper is the development of an IDPS framework, as proposed by Abraham and Bindu (2021), aimed at detecting and mitigating these risks. This requires the implementation of robust security measures through an SDN framework that leverages ML algorithms to detect specific MitM subtypes, including the most common and dangerous ones like Address Resolution Protocol (ARP) spoofing, as highlighted by Meghana et al. (2017), and ARP flood, as noted by Du et al. (2021). The selection of the SDN framework is guided by the optimal results obtained using commonly employed metrics for evaluating the ML model, including accuracy, fit time, prediction time, and false alarm rate (FAR). To further assess the performance of the optimal classifier in real-world traffic scenarios, we deploy it within the SDN Ryu controller for the detection and real-time mitigation of MitM attacks, as demonstrated by Asadollahi et al. (2018). We then evaluate its performance across various network topology sizes, including small, medium, and large, as well as different types such as single, tree, and mesh topologies.

The structure of this paper is organized as follows: Section 1 introduces the paper. Section 2 reviews previous research on countering MitM attack methods. Section 3 provides relevant background knowledge. Section 4 outlines the methodology employed in this study. Section 5 presents the analysis and discussion of experimental results, comparing our method's outcomes with those of other relevant works. Finally, Section 6 concludes the paper.

2 Related works

In this section, we discuss previous works and methods that address MitM attacks.

Ashraf (2021) explored the proactive role of IoT devices in the development of smart cities by examining how IoT technologies, such as wireless sensors and big data management systems, are revolutionizing urban infrastructures. The study highlights the importance of smart traffic management, smart healthcare, energy utilization, and agricultural systems as key aspects of future smart cities. Through an Analytic Hierarchy Process (AHP), it demonstrates that around 98% of residents in smart cities report satisfaction with their environment, in contrast to those in traditional cities. However, the study acknowledges the challenges of handling IoT big data, such as congestion monitoring, and stresses the need for further innovations in deep learning and



data classification frameworks. Limitations include the lack of real-world implementations for some proposed frameworks, indicating that much of the research remains theoretical.

Sebbar et al. (2020) explored MitM attacks on SDN networks, demonstrating their feasibility, and proposed a detection model called CBNA-RF, which uses machine learning, specifically RF, as highlighted by Haddouchi and Berrado (2019), to detect and mitigate these attacks in real-time. The CBNA-RF model, tested in a controlled environment, shows high accuracy (97.4%) and performance (96.7%) in identifying malicious nodes and stopping MitM attacks without heavy authentication mechanisms. However, the model has limitations, such as its restricted testing environment. It aims to test the model in different network scenarios and under varying network sizes, and explore additional machine learning techniques for better results.

Farhin et al. (2020) introduced a fuzzy neural network (FNN) model, as detailed by Chang et al. (2024), integrated with SDN for detecting attacks in IoT environments. The model includes four adaptive neuro-fuzzy inference systems (ANFIS), where the outputs of three ANFIS are used as inputs to the fourth, which makes the final decision. The system was tested using the NSL-KDD dataset, as referenced by Zhao (2022), and demonstrated high accuracy in detecting various types of attacks, such as DDoS, MitM, SC (Gnad et al., 2017), and MC (Mwange and Cankaya, 2024; Beaman et al., 2021), by combining the features of multiple ANFIS. The proposed approach successfully mapped and ranked attacks based on input feature similarity, highlighting its potential for robust IoT security. However, the paper suggests that further improvements are needed in accuracy, particularly for MitM attacks. Additionally, the model should be tested with real traffic in IoT networks and under various scenarios.

Saritakumar et al. (2021) proposed a method to mitigate MitM attacks in SDN by detecting and dropping ARP spoofing packets. The authors implemented the proposed algorithm on both RYU

and POX, as discussed by [Kaur et al. \(2014\)](#), and evaluated its performance. The results show that the algorithm can effectively detect and mitigate MITM attacks by analyzing IP-MAC address bindings, reducing CPU utilization by up to 20% compared to before mitigation. The execution time of the mitigation algorithm is also lower on the RYU controller compared to POX. Overall, the proposed solution provides an efficient and lightweight approach to secure SDN networks against MITM attacks. However, this paper used a simple method to detect and mitigate MitM attacks. This method has multiple false alarms in real SDN network and needs to integrate machine learning to reduce false alarms and achieve higher accuracy.

[Cherian and Varma \(2022\)](#) discuss the mitigation of DDoS and MitM attacks in SDN-based IoT networks using a Belief-Based Secure Correlation (BBSC) approach. The proposed BBSC security algorithm uses a novel encryption algorithm and decryption algorithm as described by [Ezeofor and Ulasi \(2014\)](#) to ensure secure communication. The algorithm achieves detection time of 1.53ms and detection accuracy of 99.53% for 10 packets, and detection time of 7.26ms and detection accuracy of 99.53% for 100 packets. The results show significant improvements in detection accuracy and time compared to existing frameworks. The paper also discusses various techniques used to detect and mitigate DDoS and MitM attacks, including IPS-IDS System Framework, KRACK Mitigation Framework, and statistical methods including SPRT and SVM, as referenced by [Deore et al. \(2024\)](#) and [Abdullah and Abdulazeez \(2021\)](#). The proposed architecture integrates multi-point access and authentication cryptography, ensuring secure communication and authorization access policy. The performance evaluation of the BBSC algorithm shows better detection time and accuracy compared to existing methods, making it a promising solution for mitigating DDoS and MitM attacks in SDN-based IoT networks. However, this paper needs to test the model on a real testbed that considers periodic data transmission from IoT devices. Additionally, the algorithm should be improved further to ensure that attacks are efficiently detected and mitigated even with random data generation from IoT devices.

[Ahuja et al. \(2022\)](#) presented an approach to detect ARP attacks including ARP Poisoning and ARP Flooding, within SDNs using machine learning techniques. The authors developed a Python application at the SDN controller to collect and log relevant features into a traffic dataset, which was then used to train various machine learning models. Among these, the hybrid Convolutional Neural Network-Long Short Term Memory (CNN-LSTM) model, as referenced by [Ramaswamy and Chinnappan \(2023\)](#), achieved the best performance with an accuracy of 99.73%. The study highlights the high CPU utilization (over 97%) and significant memory usage during an attack, and the model's quick attack detection time of 63,000 microseconds further demonstrates its efficiency. The limitation of this paper is the need to test the performance of the hybrid CNN-LSTM model compared to other ML algorithms and validate the model on a real testbed to demonstrate its efficacy in detecting ARP poisoning and spoofing attacks.

[Adhikari et al. \(2022\)](#) proposed an encryption-based solution to prevent MitM attacks in the Southbound Interface (SBI) of SDN architecture. The solution combines Elliptic-Curve Diffie-Hellman (ECDH) key exchange, as described by [Haakegaard and](#)

[Lang \(2015\)](#), and Advanced Encryption Standard (AES) 256-bit encryption, referenced by [Saha et al. \(2024\)](#), to secure the communication between the SDN controller and the underlying switches. The authors demonstrate how the MitM attack can be carried out using tools like Bettercap and SSLStrip, and then present their proposed algorithm to generate ECDH keys and use them for AES encryption of the traffic between the controller and switches. The experimental setup involves an ODL controller, a Mininet emulator for the infrastructure layer, and a Kali Linux machine for launching the MitM attack. The authors claim that their approach effectively prevents the MitM attack and can also be used to secure other types of malicious activities in the SDN environment. However, encryption is not an effective method to protect against MITM attacks. It is weak in protecting against ARP flooding attacks: ARP flooding occurs at the data link layer (Layer 2) of the OSI model, while ECDH and AES protect communications at higher layers (typically Layer 4 or above). While ARP flooding can disrupt the network by overwhelming devices with excessive ARP requests or replies, it does not compromise the encryption provided by ECDH and AES. These cryptographic algorithms are specifically designed to ensure the confidentiality and integrity of data during communication. However, they do not mitigate ARP flooding directly or address its potential impact on network operations, such as traffic redirection or IoT device overload.

[Saritakumar et al. \(2023\)](#) proposed a method to detect ARP spoofing attacks in SDN environments by monitoring ARP traffic for inconsistencies. The technique involves injecting ARP request packets into the network and comparing IP-MAC address bindings with incoming packets. The implementation using Mininet, RYU controller, and open vswitches demonstrates that as the percentage of attack increases, the latency and packet loss of legitimate traffic also increase. The detection time improves with higher attack percentages, while memory usage remains relatively stable. However, CPU utilization peaks at 0.77% during 100% attack scenarios, indicating efficient resource use even under full attack conditions. The limitation of this work is the presence of false alarms when the CPU utilization of the SDN network's operating system is recorded between 80% and 100%, which may not be due to ARP attacks. This problem can be addressed by using machine learning methods or by analyzing the CPU execution programs.

[Gowda and Dayananda \(2023\)](#) presented a solution for detecting and preventing ARP spoofing attacks within an SDN framework, utilizing the Ryu controller and OpenFlow protocol. The test environment included an Ubuntu virtual system with 2 GB RAM and a Mininet emulator. The performance metrics focused on ARP attack detection time and CPU utilization of the Ryu controller. The proposed solution demonstrated a detection time for ARP attacks and maintained a low CPU utilization of 2.0% during attacks, which dropped to 0.7% post-mitigation. The solution effectively identified and blocked malicious ARP packets by analyzing and comparing ARP packet information against stored mappings, proving its efficacy in enhancing network security against ARP spoofing attacks. However, this work needs additional metrics to evaluate performance and should also be tested on larger and more complex network topologies to provide a scalable approach to network security and incorporating machine learning algorithms.

Alani et al. (2023) presented ARP-PROBE, an ARP spoofing detection system designed for IoT systems, leveraging a deep neural network (DNN) architecture. The system's neural network model was optimized using Talos to select the best architecture and hyperparameters, which included 24 neurons in the first hidden layer, 16 neurons in the second and third layers, and 8 neurons in the last hidden layer. The model was trained using a balanced dataset with 428,918 samples, achieving a high accuracy of 99.99% during 10-fold cross-validation. Testing on a second dataset confirmed the model's robustness with consistent accuracy, precision, recall, and F1 score of 0.999. The model's predictions were explained using SHAP values to enhance transparency and trust in the decision-making process. The study demonstrates that ARP-PROBE is an efficient and reliable tool for detecting ARP spoofing in resource-constrained IoT environments. The shortcoming of this paper is that when the DNN model is deployed on a real SDN, it has high CPU and resource consumption. This could affect the functionality of resource-constrained IoT devices and prevent the detection of attacks in real-time.

Khedr et al. (2023) introduced the P4-HLDMC framework for detecting and mitigating DDoS and ARP attacks in SD-IoT networks. The framework integrates machine learning, stateful P4, and a multi-controller SDN architecture to enhance attack detection. Experimental results show that the SDP4-MEV model achieved the highest accuracy rate of 99.15% for ARP Spoof and 99.78% for ARP Flood, outperforming other models. The framework also demonstrated high detection rates, low false-alarm rates, low latency, and high throughput, surpassing existing methods. The Improved ODL controller emerged as the optimal choice for implementing the framework, offering improved resource allocation, scalability, and stability, achieving a throughput of 28,640 packets per second and the lowest CPU and memory usage among tested controllers. The limitation of this work is that it does not address other ARP attacks, which are threats to IoT devices, such as ARP flooding. Additionally, it has not been tested on real SDN networks with different topologies and sizes to evaluate performance.

Aoueilyne et al. (2024) used a simple Python algorithm in Ryu Controller to detect a MitM attack on IoT devices using SDN. The implemented algorithm could detect both ARP floods and ARP spoofing and mitigate it. If the ARP packet count for a port exceeds 20, it triggers the flood handling mechanism and checks for IP-MAC address mismatches to detect ARP spoofing. The author mitigated the impact of ARP spoofing or ARP floods by blocking the in_port, which refers to the port of the switch where the packet was received. The disadvantage of this paper is that the algorithms have a False Alarms that could detect a wrong signal for MitM attacks, especially for ARP floods.

Hnamte and Hussain (2024) presented a novel approach for detecting and mitigating ARP spoofing attacks within SDNs using a Deep Neural Network (DNN) model as described by Mittal (2020). By employing a self-generated dataset tailored to SDN environments, the proposed system achieved an unprecedented detection accuracy of 100%, with negligible loss rates and rapid detection times. The results highlight the system's robustness and adaptability, as it significantly reduces false positives and negatives, ensuring precise and timely detection of ARP spoofing incidents.

Experimental evaluations demonstrated that the system efficiently manages increased CPU loads during attacks, maintaining stability across varying network scales. The study underscores the effectiveness of integrating deep learning techniques for real-time anomaly detection in enhancing network security without compromising performance. The limitation of this paper is that the proposed DNN model learns the training data too well, including the noise and outliers, indicating clear overfitting. Additionally, the model is tested only on a small SDN topology network.

Table 1 provides a brief summary of the above related works, identifying the methods 249 used, their advantages, and their disadvantages.

3 Background

This section delineates the fundamental types of MitM attacks, which pose a genuine threat. It subsequently details the SDN architecture layers in IoT devices.

3.1 MitM attacks types

Both ARP flood and ARP spoofing are forms of MitM attacks that exploit vulnerabilities in ARP, which is used to map IP addresses to MAC addresses in a local network. While they share similarities, they have distinct characteristics and objectives.

3.1.1 ARP flood

In an ARP flood attack, as shown in Figure 2, the attacker sends a significant volume of ARP requests, frequently employing forged source IP addresses, directed at the target network. This barrage of requests has the potential to inundate the target's ARP cache, leading to an overflow of incorrect or redundant entries. When the switch becomes overwhelmed, it may transition into hub mode. In hub mode, the switch indiscriminately forwards traffic to all computers connected to the network. Consequently, the attacker could exploit this situation to capture all network traffic using sniffing software.

3.1.2 ARP spoofing

As shown in Figure 3, ARP spoofing, also known as ARP poisoning, takes advantage of a flaw in the ARP protocol. The ARP protocol operates under the assumption that ARP responses correlate with the IP address in the ARP request message, yet it lacks a mechanism for authenticating the legitimacy of ARP responses. This inherent design limitation in the ARP protocol provides an avenue for exploitation by hackers. The consequences of successful ARP spoofing can include unauthorized access, interception of network traffic, and potential for MitM attacks. The attacker can redirect traffic meant for the legitimate device to their own system, enabling eavesdropping or other malicious activities.

TABLE 1 Comparison of recent studies on MitM attacks.

Paper	Method used	Advantage	Disadvantage
Ashraf (2021)	Analytic Hierarchy Process (AHP)	AHP is a valuable tool for analyzing the role of IoT and smart systems	Lack of real-world implementations for some proposed frameworks
Sebbar et al. (2020)	RF method	Can detect and mitigate MitM attacks in real-time	Needs to improve accuracy
Farhin et al. (2020)	Fuzzy neural network	Can detect MitM attacks	- Missing the mitigation step - Needs to improve accuracy
Saritakumar et al. (2021)	SDN controller method	- Adaptable to RYU and POX controllers - Reduces CPU utilization - Can detect and mitigate MitM attacks in real-time	False alarms
Cherian and Varma (2022)	Belief-Based Secure Correlation (BBSC) approach	- High accuracy and low detection time - Can detect and mitigate MitM attacks	Needs to be tested on a real testbed for IoT devices
Ahuja et al. (2022)	Hybrid CNN-LSTM model	- High accuracy - Can detect and mitigate MitM attacks	Needs to be tested on a real testbed
Adhikari et al. (2022)	Combines ECDH key exchange and AES 256-bit encryption	- Secures communication between the SDN controller and the underlying switches - Effectively prevents MitM ARP spoof attacks	Needs an effective method to protect against MitM ARP flood attacks
Saritakumar et al. (2023)	Injecting ARP request packets	- Real-time detection - High latency	False alarms
Gowda and Dayananda (2023)	Method integrated on RYU controller	- Decreases CPU utilization - Can detect and mitigate MitM attacks	Needs additional metrics to evaluate performance
Alani et al. (2023)	Deep learning method	- Feature extraction and selection - Low false negative rate - High accuracy	High CPU and resource consumption
Khedr et al. (2023)	Modified ensemble voting (MEV) algorithm	- Incorporation of 17 new features - Low false-alarm - High detection rates	- Overfitting - Tested only on small SDN network
Aoueilayine et al. (2024)	ARP packet count and IP-MAC address mismatches	- Real-time detection and mitigation - Reduces CPU utilization	False alarms
Hnamte and Hussain (2024)	DNN model	- Real-time detection and mitigation - High accuracy and low FAR - Reduces CPU utilization	- Needs to be tested on a real testbed for IoT devices - Need to address ARP flooding attacks

3.2 Theoretical study

In the theoretical studies, we examine the selection of the SDN framework in comparison to other potential frameworks for IoT devices. We then compare the frameworks used in recent works with our chosen framework in terms of security and effectiveness.

3.2.1 Motivation

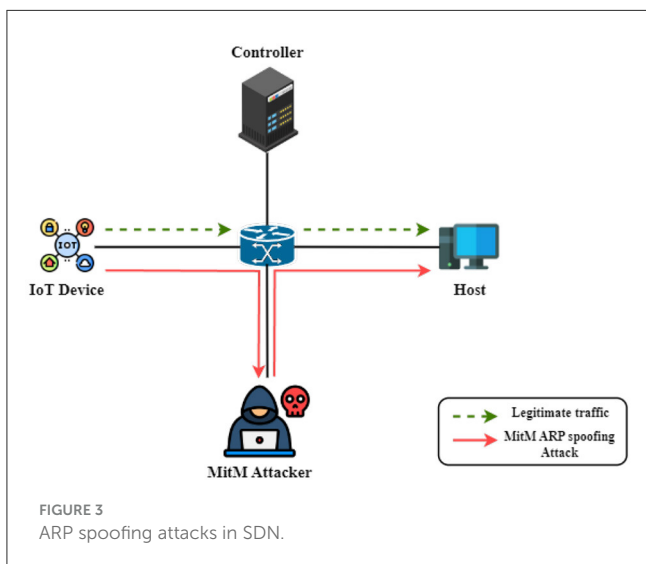
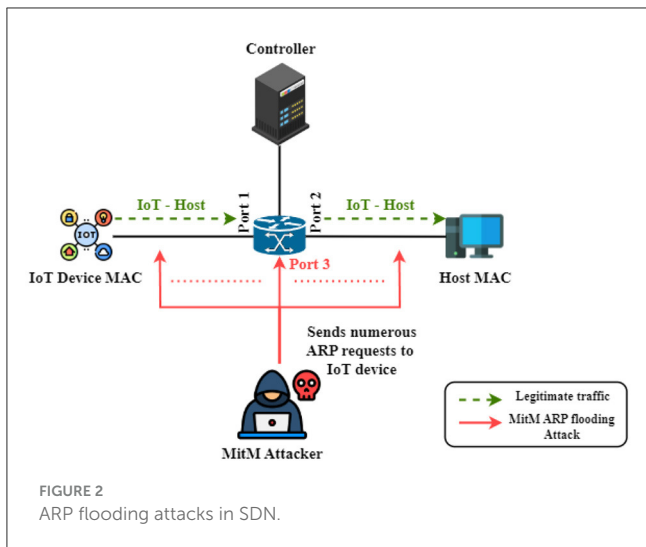
Compared to other related works, our approach stands out by utilizing a synthetic MitM dataset collected through the Ryu controller in an SDN environment. Our method effectively detects and mitigates, in real time, two types of MitM attacks : ARP flooding and ARP spoofing using a CNN-based IDPS model with high accuracy, a low false alarm rate, and a short detection time. Additionally, our approach is adaptable, scalable, and works across multiple SDN network topologies and varying network sizes.

3.2.2 The criteria used for the comparison various frameworks

The criteria for comparing various frameworks are based on the following key aspects:

1. **Security performance:** this includes detection accuracy, false alarm rate, and the ability to prevent specific types of attacks (such as MitM attacks) in real time. This is critical as IoT environments are highly susceptible to such attacks.
2. **Response time:** the time taken to detect and mitigate attacks is essential, particularly in IoT networks where latency can impact overall system functionality. Lower detection times ensure proactive threat response.
3. **Scalability:** the ability of the system to operate effectively across different SDN topologies and varying network sizes (small, medium, and large). Scalability is important for ensuring that the solution can be applied to a wide range of IoT environments.
4. **Resource efficiency:** the consumption of CPU and other resources during operation is considered to assess the feasibility of deploying the framework in resource-constrained IoT devices and networks.
5. **Adaptability:** the ability to work with multiple network configurations (single, tree, mesh), which is vital in dynamic IoT environments.

These criteria are crucial for securing IoT devices, as they directly influence the robustness, real-time protection, and operational efficiency of the system. Our study



compares these criteria across different frameworks, as outlined below.

3.2.3 Framework selected

Compared to other potential frameworks, such as the Zero Trust Security Framework (Stafford, 2020) and the Blockchain-based Security Framework (Krishnan et al., 2018), we opted for the SDN framework to secure IoT devices due to its centralized control, which enables granular segmentation and dynamic policy enforcement—essential for efficiently managing diverse IoT ecosystems. Unlike the Zero Trust Security Framework, SDN offers both scalability and flexibility while integrating seamlessly with legacy systems, ensuring smooth deployment in complex environments. Furthermore, compared to the Blockchain-based Security Framework, SDN provides superior performance and traffic visibility without adding significant complexity, making it a pragmatic choice for securing IoT networks. While SDN offers numerous advantages, including centralized control, dynamic policy enforcement, and enhanced visibility, its broader

applicability and potential drawbacks require careful consideration. One key challenge is its reliance on a central controller, which may become a single point of failure, raising concerns about resilience and robustness in large-scale, mission-critical IoT environments. Additionally, SDN’s centralized architecture can introduce scalability and latency issues, particularly when managing distributed IoT networks with geographically dispersed devices. Security is another potential drawback, as attacks on the controller or data plane could disrupt the entire network. Finally, integrating SDN into legacy infrastructures may prove complex, requiring substantial reconfiguration and expertise. Thus, while SDN is a powerful tool for securing IoT devices, its limitations must be addressed to ensure its broader and more effective application.

The Table 2 summarizes the outcome of our comparison, illustrating that our proposed CNN-based IPDS stands out in terms of detection accuracy, response time, scalability, and adaptability, making it a highly suitable solution for securing IoT devices in SDN environments.

3.2.4 Theoretical comparative study

Syed et al. (2024) proposed a secure group authentication method using Dickson polynomials and blockchain technology. By leveraging physically unclonable functions (PUFs) for hardware security and decentralized blockchain smart contracts, it improves communication security, protecting against tampering, replay, and impersonation attacks. The scheme enhances performance in both security and efficiency compared to traditional methods. The authors Padmaja et al. (2022) explored the use of AI (forming AIoT) in securing IoT applications. AI models such as Decision Trees, SVM, and Naïve Bayes are applied to IoT data for real-time anomaly detection and decision-making, with a focus on healthcare, smart cities, and edge computing. The paper finds Decision Trees to be more accurate and efficient than other models, especially in IoT health-related applications. Selvarajan et al. (2023) introduced a hybrid AI and blockchain framework, AILBSM, for securing Industrial IoT (IIoT) systems. It integrates Lightweight Consensus Proof-of-Work (LCPoW) for secure data authentication and uses a neural network (COSNN) for detecting anomalies with high accuracy. The framework protects against attacks like data poisoning and ensures privacy and data integrity through blockchain technology. Manoharan et al. (2023) discussed securing IoT networks using blockchain combined with machine learning algorithms. The decentralized nature of blockchain ensures tamper-proof data transmission, while AI and deep learning models improve security and data transmission efficiency. This approach is presented as essential for safeguarding smart cities, homes, and industries. Shitharth et al. (2023) focused on improving security for edge computing in IoT environments. It proposes a multi-attack intrusion detection system using a combination of Backpropagation and Radial Basis Function neural networks. The model achieves high accuracy in detecting various attacks, making it a robust solution for anomaly detection in edge-assisted IoT networks.

Our work achieves an impressive detection accuracy of 99.96% with a minimal false alarm rate of just 0.02% in identifying and mitigating MitM attacks within SDN-based IoT environments.

TABLE 2 Comparison of security frameworks for IoT environments.

Framework	Security performance	Response time	Scalability	Resource efficiency	Adaptability
Zero trust security framework	Strong access control but lacks real-time MitM attack prevention	Varies (Authentication focus)	Moderate: Complexity in managing distributed IoT	Varies with device load	Moderate
Blockchain-based framework	Enhanced tamper-proof communication, less real-time focus on MitM attacks	Higher due to block validation	Limited: Blockchain's overhead affects large networks	High resource consumption	Low due to blockchain's latency issues
AIoT using decision trees	High detection accuracy for anomaly detection but lower focus on real-time mitigation	Moderate: Detection-based response	Moderate: Depends on AI model used	Moderate: AI processing demands resources	Moderate
AILBSM (AI and blockchain)	High accuracy for anomaly detection, lacks real-time MitM prevention	Higher response time due to blockchain	Low: Blockchain overhead impacts large-scale networks	High resource consumption	Low
Our CNN-based IPDS (SDN)	Detection Accuracy: 99.96% False Alarm Rate: 0.02%	1 - 3.5 seconds	High: Tested with 64 hosts in various topologies	CPU usage: 12% - 22%	High: Works with single, tree, mesh networks

This high level of precision is critical for maintaining operational security, as it allows for rapid and accurate threat identification. Additionally, our system provides real-time response times ranging from 1 to 3.5 s, effectively preventing data breaches during active attacks and taking a proactive approach rather than merely reacting after an attack occurs. In comparison, existing methods, such as AIoT and blockchain models, tend to focus primarily on authentication and long-term anomaly detection, lacking the immediate intrusion prevention capabilities necessary for addressing MitM attacks effectively. Our CNN-based Intrusion Prevention Detection System (IPDS) not only excels in accuracy but also maintains CPU usage between 12% and 22% during the mitigation process, ensuring resource efficiency and scalability, even in larger networks with up to 64 hosts. Furthermore, we have tested our IPDS across various network topologies, including single, tree, and mesh configurations, proving its versatility and applicability in diverse SDN environments. This adaptability stands in contrast to AI and blockchain hybrid models, which often have limitations based on specific applications and may face scalability challenges due to higher resource demands. Ultimately, our IPDS directly mitigates active MitM attacks as soon as they are detected, offering integrated, real-time prevention capabilities that other intelligent intrusion detection systems typically lack, thus ensuring robust protection for network security.

3.3 SDN architecture

In an SDN-based IoT architecture, the network is divided into three distinct layers: the data plane, the control plane, and the application layer. Each layer has specific functions and responsibilities, making the network more flexible and manageable.

3.3.1 Data plane

The data plane, also known as the forwarding plane or user plane, is responsible for the actual forwarding of network traffic. It deals with the transmission and reception of data packets, making

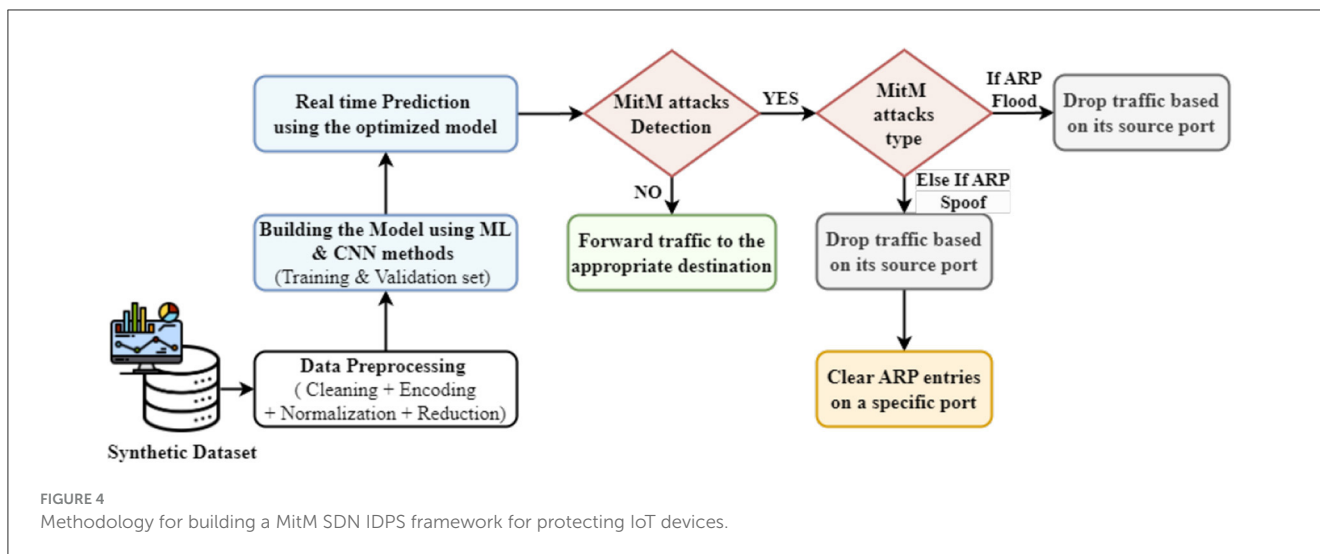
forwarding decisions based on the information contained in the packets. The data plane is implemented in the network devices, such as switches and routers. The primary functions of the data plane include packet forwarding, encapsulation/decapsulation, and basic packet processing. It operates on the rules and instructions set by the control plane. In our work, we utilized an OpenFlow-enabled switch where the data plane forwards packets according to the flow table entries determined by the control plane.

3.3.2 Control plane

The control plane is responsible for making decisions about how data should be forwarded through the network. It manages the network's state, maintains routing tables, and determines the best path for data traffic. The control plane communicates with the data plane to update forwarding instructions. The control plane handles tasks such as routing, signaling, network topology discovery, and maintaining a global view of the network. It is responsible for programming the data plane to ensure that network devices behave as desired. In our project, we employed the Ryu controller as our SDN controller due to its inherent flexibility and programmability, particularly leveraging the Python programming language. The choice of Ryu was driven by its rapid communication capabilities with OpenFlow-enabled switches, enabling efficient programming of their flow tables. The combination of Python scripting and Ryu's capabilities allowed us to dynamically and seamlessly control the network, defining and managing flow entries with agility and precision. This approach significantly enhanced the efficiency and adaptability of our network architecture.

3.3.3 Application layer

In the context of SDN, the application layer represents the software applications or services that leverage the programmability of the network. These applications utilize the capabilities provided by the SDN controller to implement specific network policies, optimizations, or functionalities. Applications running on top of the SDN controller can provide network services,



orchestrate resources, and respond to dynamic changes in the network. They use the programmability of the SDN architecture to create custom solutions for network management and optimization. In our project, the application focuses on Security Applications that employ machine learning methods to detect and mitigate MitM threats while implementing robust security policies. [Supplementary Figure 1](#) summarizes the SDN architecture employed in our project.

4 Methodology

In this section, we will explain the methodology steps used to implement our IDPS framework. However, before that, we will provide an overview of the hardware and software tools used in our work.

4.1 Software and hardware equipment used

For the software, we utilized VMs and Ubuntu 22.04 LTS to establish the environment that facilitated the creation of the network topology on the virtual machine running Linux. The network topology itself was constructed using Mininet, and we employed the Ryu controller to manage and orchestrate network functions. This comprehensive setup allowed us to efficiently manage and control the network within the virtualized environment. Python 3 is widely employed for application development, encompassing OpenFlow-based SDN networks, the Ryu controller, and the creation of custom SDN topologies. Moreover, Python 3 is utilized to implement our machine learning method, which is integrated into the Ryu controller for ensuring a secure SDN network. Mosquitto MQTT, as discussed by [Mishra \(2018\)](#) is a lightweight, open-source message broker that implements the MQTT protocol, designed for low-bandwidth, high-latency environments. It facilitates communication between IoT devices, supporting a range of QoS levels for reliable message delivery. Mosquitto is known for its efficiency and ease of integration in SDN-IoT applications. We employed Macof tools

to flood the switch's MAC address table, as used by [Sankar \(2022\)](#) in his work, and we specifically utilized Ettercap tools for ARP flooding attacks, which are also used for network sniffing and possess the capability to collect data exchanged between two devices by acting as a MitM attack, as employed by [Fathima and Santhiyakumari \(2021\)](#) in his paper. Positioned between communicating devices on a network, Ettercap can intercept and capture the traffic flowing between them. This functionality enables the analysis of exchanged data, encompassing plaintext usernames, passwords, or other sensitive information. In our work, Ettercap is applied for ARP spoofing attack. Macof is a network security tool, an acronym for MAC Over Flow, designed specifically to generate a significant volume of ARP requests within a network. The tool floods the network with ARP packets in an attempt to overwhelm and disrupt normal network communication. In our project, we employed the Macof tool to generate an extensive number of ARP requests. The excess ARP requests from Macof are intended to saturate the MAC address table of a switch.

As depicted in [Figure 4](#), the methodology illustrates the steps to implement our framework for detecting and mitigating MitM attacks on IoT devices in a smart home within an SDN network. These steps will be explained clearly.

4.2 Create synthetic dataset

The Ryu application consistently monitors connected OpenFlow switches, actively solicits and receives flow statistics, and then records and stores this information in a CSV file. The collected dataset includes three categories: normal traffic labeled as 0, ARP flooding labeled as 1, and ARP spoofing labeled as 2. The dataset has a memory size of 128 MB and consists of 16 features, as shown in [Table 3](#), including labeled columns, with a total of 1,122,520 records, as illustrated in [Supplementary Figure 2](#).

4.3 Data preprocessing

Data preprocessing is a crucial step in the data analysis and machine learning workflow. It involves transforming raw data into

TABLE 3 Feature descriptions.

Feature name	Description
Timestamp	Captures timing for packet traffic within an SDN network.
Datapath_ID	Uniquely identifies a switch network device.
In_port	The port on a switch where a packet was received.
Out_port	The port on a switch where a packet should be forwarded.
Src_mac	The MAC address of the source device that originated the packet.
Dst_mac	The MAC address of the destination device to which the packet is being sent.
Src_ip	The IP (Internet Protocol) address of the source device.
Dst_ip	The IP address of the destination device.
Time_to_live	Prevents packets from circulating indefinitely in the network.
Protocol	Indicates the protocol used by the packet, such as TCP, UDP, or ICMP.
Tp_src	The source port number, typically used in TCP or UDP packets.
Tp_dst	The destination port number, also typically used in TCP or UDP packets.
Icmpv4_code	Denotes the code associated with ICMP messages for error reporting or diagnostic purposes.
Icmpv4_type	Specifies the type of ICMP message being sent, such as echo request or echo reply.
Packet_size_bytes	Indicates the size of the packet in bytes.
Label	Tag for normal, ARP flood, and ARP spoof records, where 0 indicates normal, 1 indicates ARP flood attacks, and 2 indicates ARP spoof.

a clean and usable format. This process can significantly impact the quality of the analysis and the performance of machine learning models. Here are the typical steps involved in data preprocessing:

4.3.1 Data cleaning

This essential step includes:

1. Drop Duplicates: Dropping duplicates in Our ML model is a common preprocessing step aimed at ensuring the dataset is clean and free from redundant information.
2. Handling Missing Values: Replaces all missing values columns with a specified constant value.

After applying data cleaning techniques, we have reduced the dataset from 1,122,520 rows \times 16 columns to 627,543 rows \times 16 columns.

4.3.2 Data transformation

Encoding Categorical Data: Converting categorical data into numerical format using the label encoding technique. The columns that are categorical are: timestamp, src_mac, dst_mac, src_ip, and dst_ip. The label encoder is applied to these columns.

Normalization/Scaling: We standardizes features by removing the mean and scaling to unit variance. It's used to preprocess CSV data for machine learning by transforming it to have a mean of 0 and a standard deviation of 1.

4.3.3 Data reduction

We used Principal component analysis (PCA) for dimensionality reduction, as described by Zhang et al. (2024). This step is crucial for simplifying complex datasets by transforming them into a lower-dimensional space while retaining essential patterns and structures. It helps in improving computational efficiency and reducing noise, thereby enhancing the performance of machine learning algorithms and mitigating the risk of overfitting to the training data. Additionally, PCA aids in visualizing high-dimensional data and identifying significant features that contribute most to variance within the dataset. Based on the cumulative explained variance plot in Figure 5, selecting eight principal components effectively balances dimensionality reduction and variance retention. PCA is applied with 8 components, transforming the original data into this new feature space.

4.3.4 Data splitting

Dividing the dataset into training, validation, and testing sets to evaluate the performance of machine learning models. In our work we divides the dataset into training, validation, and testing sets in the proportions of approximately 60%, 20%, and 20%, respectively.

4.3.5 Data balancing

We handle imbalanced datasets using the Synthetic Minority Over-sampling Technique (SMOTE), as discussed by Widodo and Bambang Setiawan (2024) to ensure the model performs well across all three classes. Before applying SMOTE (as shown in Figure 6), the three classes are imbalanced, potentially leading to biased models favoring the majority class and prioritizing accuracy, which can be misleading.

4.3.6 Overfitting mitigation strategies

To address overfitting, several methods are implemented in our paper. First, PCA reduces the dataset's dimensionality by retaining the most important features, simplifying the model and removing noise. Dropout is applied to the fully connected layer, randomly dropping neurons during training to prevent the model from over-relying on specific neurons. L2 regularization is used in the dense layers, penalizing large weights to discourage the model from fitting noise in the data. Early stopping is employed to halt training once the validation loss stops improving, preventing further overfitting. Additionally, learning rate reduction decreases the learning rate when our CNN model's performance plateaus, enabling finer adjustments in later training epochs. Together, these techniques help improve our CNN model's generalization to unseen data by reducing overfitting.

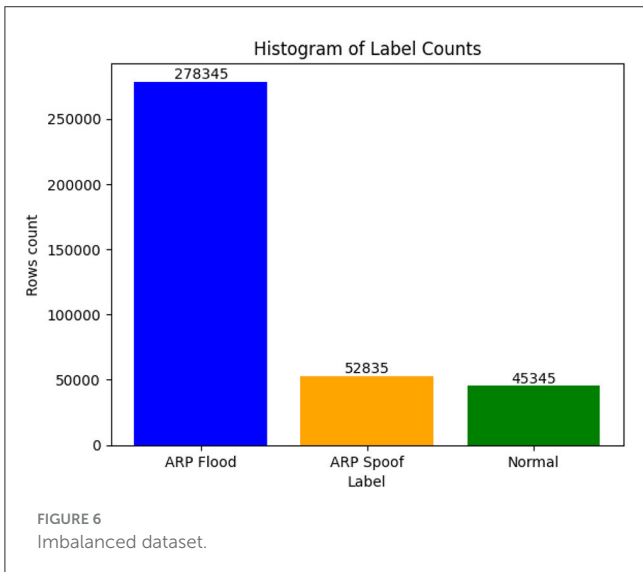
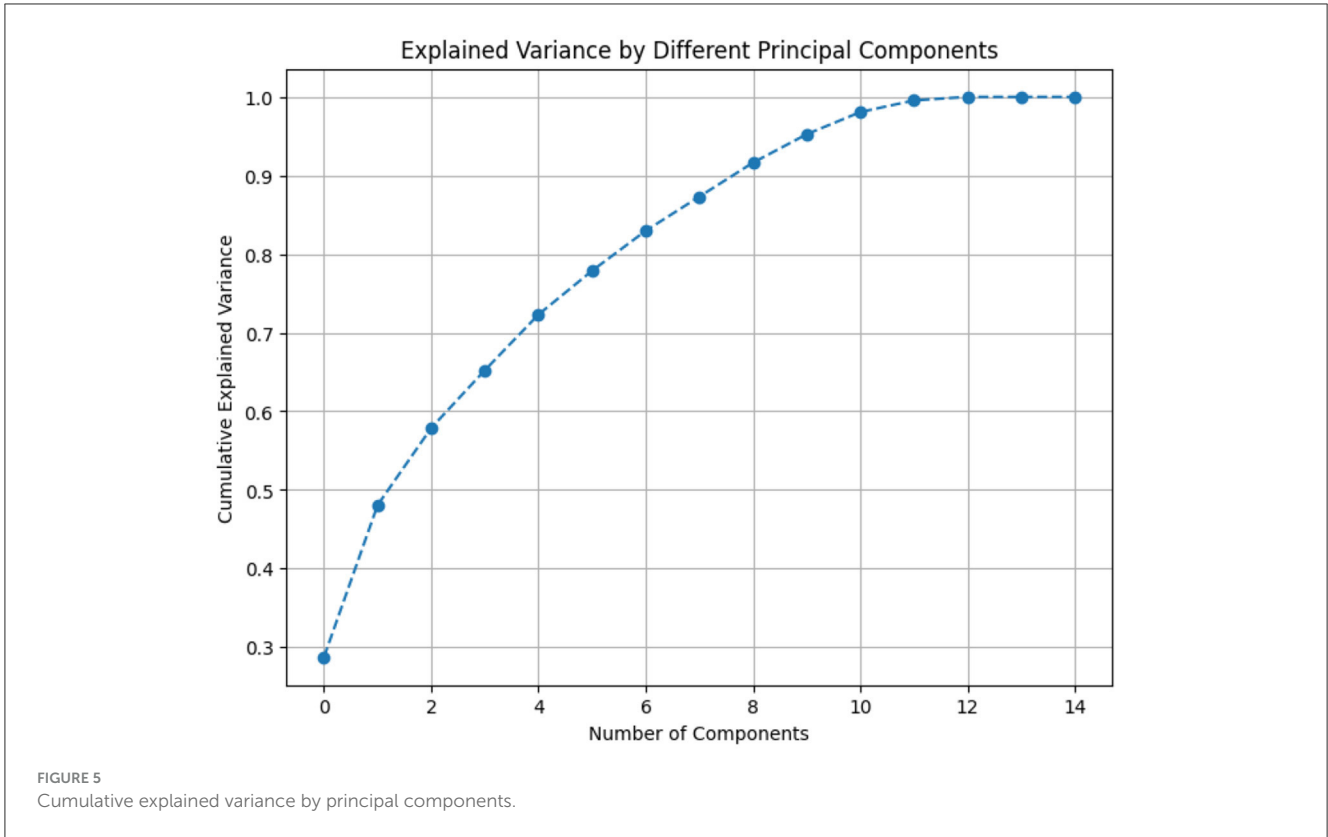


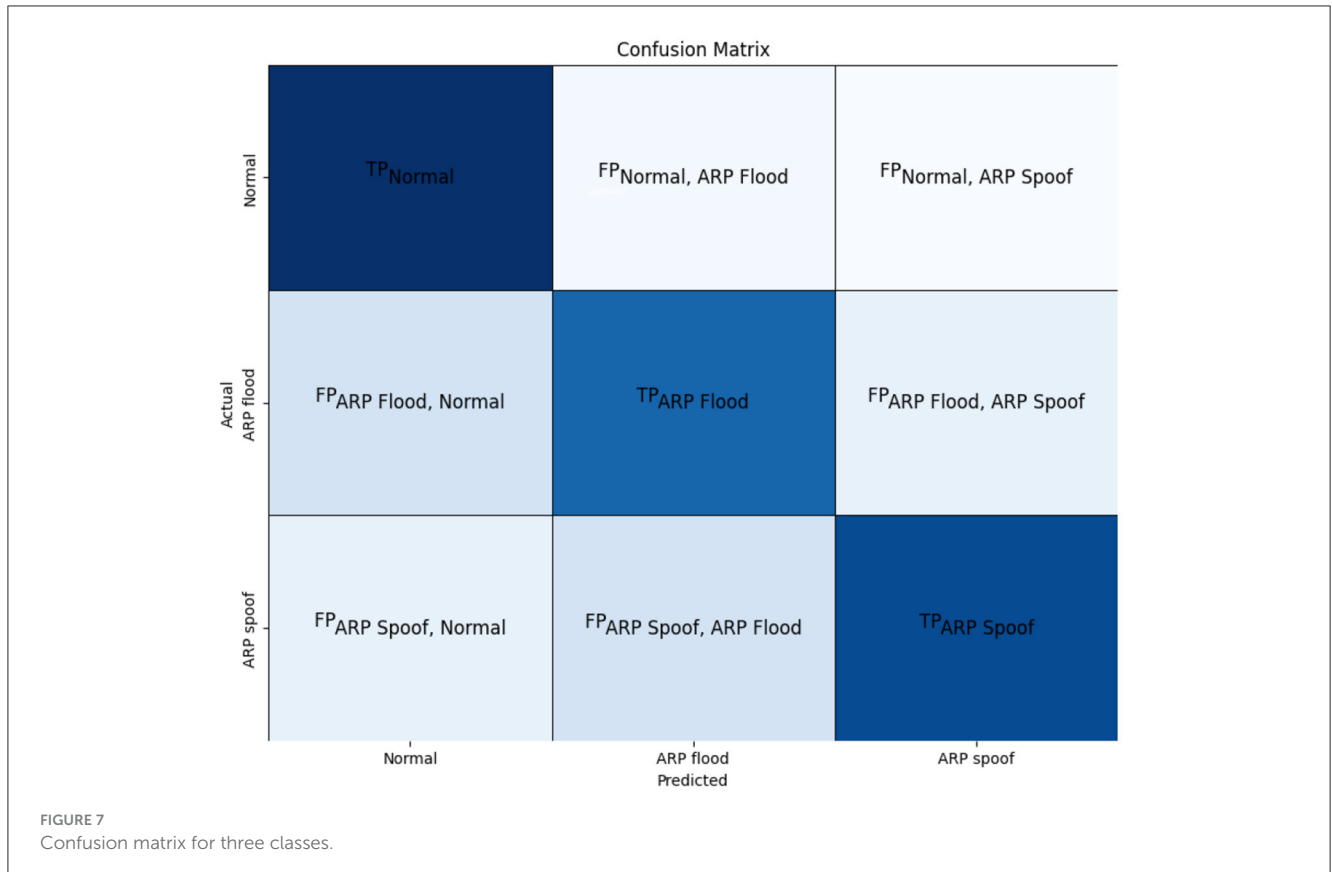
TABLE 4 Parameters employed in ML and CNN method for our Framework SDN model.

ML model	Gridsearch parameters
NB	Default parameters.
KNN	n_neighbors = 10.
RF	n_estimators = 100, criterion = "entropy", random_state = 42.
CNN	<ul style="list-style-type: none"> • First convolutional layer: 32 filters with a kernel size of 3. • MaxPooling1D layer: Pool size of 2. • Dense layer: 128 units with L2 regularization (0.001). • Dropout layer: Dropout rate of 0.5 applied to the dense layer. • Activation functions: ReLU for hidden layers, Softmax for the output layer. • Optimizer: Adam with a learning rate set to 0.001. • Epochs: Performed for 50 epochs. • Batch size: Batch size for training is set to 64. • Early stopping: Early stopping with patience of 5. • Learning rate scheduler: Learning rate reduction on plateau with factor 0.5 and patience of 3. • Input reshape: Input data reshaped to fit the Conv1D layer.

4.3.7 Algorithms selection

We selected machine learning algorithms suitable for IDS that are effective for multiclass classification tasks. The chosen algorithms include NB as used by Reddy et al. (2022), kNN as employed by Zhang (2021), RF, and CNN as applied by Alzubaidi et al. (2021) due to their effectiveness in detecting attacks, as demonstrated by Mohammadpour et al. (2022), Shakir and Mohsin (2024), Mohammed and Talib (2024), and Rakine et al. (2024). We used the GridSearch approach, as implemented

by Jiménez et al. (2008), to optimize hyperparameters by exhaustively searching a predefined grid of values for each algorithm, including NB, kNN, RF, and CNN. The goal was to find the best combination of hyperparameters that maximizes model performance. The best parameters found by GridSearch are indicated in Table 4.



4.4 Evaluation metrics

To test the model performance results, we used the following evaluation metrics:

4.4.1 Accuracy

Accuracy measures the proportion of correctly predicted instances out of the total instances. It is given by the equation:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Instances}} \quad (1)$$

4.4.2 Precision

Precision measures the proportion of true positive predictions out of all positive predictions. It is given by the equation:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (2)$$

4.4.3 Recall

Recall measures the proportion of true positive predictions out of all actual positives. It is given by the equation:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (3)$$

4.4.4 F1 Score

The F1 Score is the harmonic mean of precision and recall, balancing the two metrics. It is given by the equation:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

4.4.5 False alarm rate

FAR measures the proportion of false alarms out of the total actual negatives. It is given by the equation:

$$\text{FAR} = \frac{\text{False Alarms}}{\text{Total Actual Negatives}} \quad (5)$$

4.4.6 Training time

Training time is the duration taken to train the model on the training dataset. It is typically measured in seconds or minutes.

4.4.7 Testing time

Testing time is the duration taken to evaluate the model on the testing dataset. It is typically measured in seconds or minutes.

4.5 Confusion matrix for three classes

A confusion matrix for three classes (Normal, ARP Flood, and ARP Spoof) is a table used to describe the performance of a classification model. It is structured as shown in [Figure 7](#):

TABLE 5 Performance metric results of the ML and CNN models.

Algorithm	Accuracy	Confusion matrix	FAR	Train Time (sec)	Test time (s)
NB	0.9450	11256 0 3714 2555 90165 0 3044 0 14775	0.0342	1.27	0.10
kNN	0.9724	13255 0 1715 125 92595 0 1745 0 16074	0.0155	4.95	12.81
RF	0.9881	14191 0 779 12 92708 0 714 0 17105	0.0070	344.84	0.94
CNN	0.9996	14942 0 28 0 92720 0 26 0 17793	0.0002	2393.48	10.33

Where: - TP_{Normal} , $TP_{ARP\ Flood}$, and $TP_{ARP\ Spoof}$ are the true positives for Normal, ARP Flood, and ARP Spoof respectively. - $FP_{Normal, ARP\ Flood}$, $FP_{Normal, ARP\ Spoof}$ are the false positives where instances of Normal were incorrectly predicted as ARP Flood or ARP Spoof. - $FP_{ARP\ Flood, Normal}$, $FP_{ARP\ Flood, ARP\ Spoof}$ are the false positives where instances of ARP Flood were incorrectly predicted as Normal or ARP Spoof. - $FP_{ARP\ Spoof, Normal}$, $FP_{ARP\ Spoof, ARP\ Flood}$ are the false positives where instances of ARP Spoof were incorrectly predicted as Normal or ARP Flood.

True Positives (TP) are the correctly predicted positive instances. False Positives (FP) are the incorrectly predicted positive instances. True Positives (TP) are the correctly predicted positive instances. False Negatives (FN) are the actual positive instances that were incorrectly predicted as negative.

False Alarms (FA) are the actual negative instances that were incorrectly predicted as positive. Total Actual Negatives are the actual negative instances in the dataset.

True Positives (TP) are the correctly predicted positive instances. True Negatives (TN) are the correctly predicted negative instances. Total Instances are the total number of instances in the dataset.

5 Results

This section provides details on the experimental setup, the presentation of the results, and a comparative analysis of the algorithms. It also includes a performance comparison across different SDN topologies (single, tree, and mesh), along with the implementation in a real SDN environment.

5.1 Performance of each classification model

The Table 5 presents performance metrics of various machine learning algorithms, which are: NB, kNN, RF, and CNN. The metrics evaluated include Accuracy, Confusion Matrix, False Alarm

Rate (FAR), training time in seconds, and test time in seconds. Among the algorithms, CNN achieves the highest accuracy of 99.96%, closely followed by RF with 98.81%, kNN with 97.24%, and NB with 94.50%. CNN also demonstrates the lowest FAR of 0.0002, indicating superior performance in minimizing false alarms. However, it requires significantly longer training and test times compared to the other models, whereas RF has the lowest training time. Overall, CNN emerges as the best-performing algorithm in terms of accuracy and false alarm rate, despite its longer computational requirements.

In Figure 8, the training and validation loss of a CNN model are plotted over 50 epochs. The training loss (blue line) decreases rapidly in the initial epochs and continues to decline at a slower pace, eventually stabilizing toward the end. The validation loss (orange line) follows a similar pattern, starting higher but decreasing gradually and stabilizing after around 10 epochs. The close proximity of the training and validation loss curves suggests that the model is not overfitting and maintains good generalization performance, as both losses stabilize at low values.

In Figure 9, the training and validation accuracy of the CNN model are shown over 50 epochs. The training accuracy (blue line) increases sharply during the first few epochs and then continues to rise more gradually, stabilizing at around 99.8%. The validation accuracy (orange line) remains consistently high throughout the training process, slightly exceeding the training accuracy and fluctuating minimally above 99.8%. This close alignment between the training and validation accuracy curves indicates that the model achieves high performance on both the training and unseen data, demonstrating effective learning and generalization capabilities.

5.2 Performance results on a real SDN

We tested the performance of the models on a real SDN using small, medium, and large networks, as well as single, tree, and mesh topologies, as shown in Table 6.

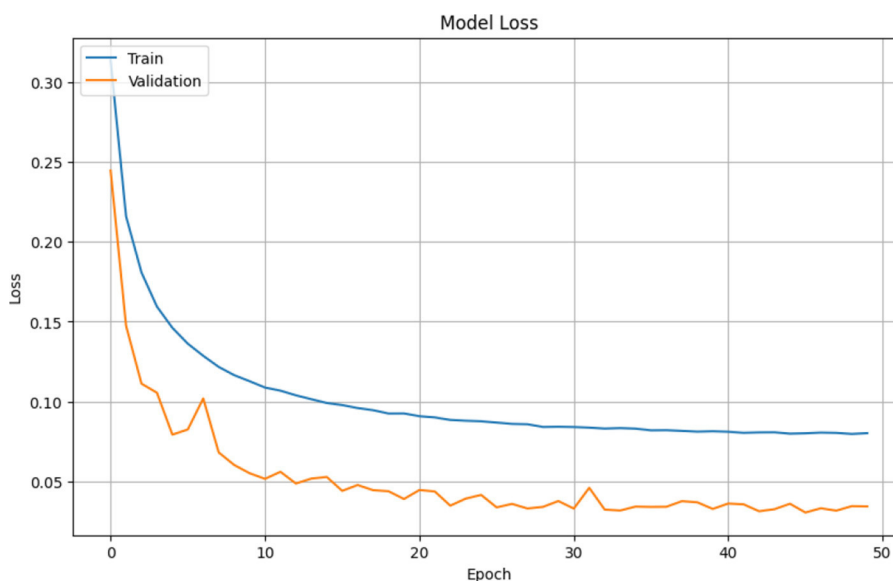


FIGURE 8
Training and validation loss over epochs.

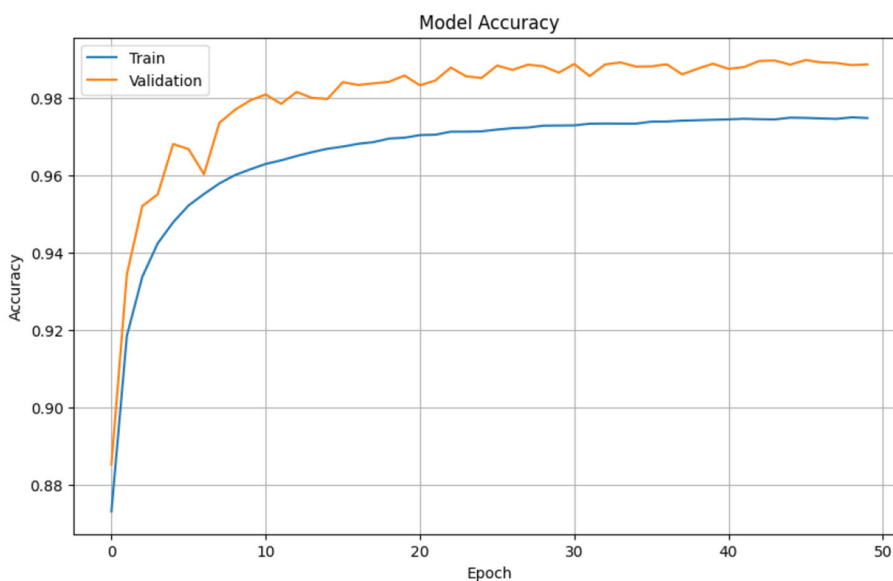


FIGURE 9
Training and validation accuracy over epoch.

The Table 7 illustrates the detection times for various ML and CNN methods in different SDN network topologies and sizes. The CNN method consistently exhibits the longest detection times across all scenarios, highlighting its slower performance compared to traditional ML algorithms such as NB, KNN, and RF. Among the ML algorithms, RF shows the best performance in terms of speed, particularly in larger network sizes, followed closely by KNN and then NB. While CNN's detection times are longer, it may provide superior accuracy in detecting complex attack patterns, suggesting that the choice of algorithm should balance detection speed with the need for accuracy and robustness in SDN environments.

Tables 8, 9 present CPU usage percentages for various ML methods and a CNN under MitM ARP spoofing and ARP flooding attacks across diverse SDN network topologies and sizes. CNN consistently exhibits the highest CPU usage across all scenarios, indicating its greater computational demand compared to traditional ML algorithms. Among the ML methods, KNN generally consumes more CPU resources than NB and RF, with RF showing moderate usage and NB being the least resource-intensive. As network size increases from small to large, CPU usage for all methods escalates, with CNN's usage increasing most significantly. These findings suggest that while CNN may

TABLE 6 Configuration details for SDN encompassing multiple topology types and sizes.

Network topology type	Network size	Host	Switch	Controller
Single network size	Small	4	1	1
	Medium	16	1	1
	Large	64	1	1
Tree network size	Small	4	3	1
	Medium	16	15	1
	Large	64	63	1
Mesh network size	Small	4	4	1
	Medium	16	16	1
	Large	64	64	1

TABLE 7 MitM detection time in seconds for ML and CNN methods in SDN encompassing multiple topology types and sizes.

Network topology type	Network size	RF (s)	GBM (s)	XGB (s)	CNN (s)
Single network	Small	1.0012	1.0043	1.0205	1.0801
	Medium	1.0023	1.0087	1.0412	1.1604
	Large	2.0046	2.0175	2.0824	2.3208
Tree network	Small	1.0013	1.0054	1.0256	1.1003
	Medium	1.0031	1.0102	1.0510	1.2005
	Large	2.0052	2.0204	2.1020	2.4007
Mesh network	Small	1.0022	1.0063	1.0304	1.1205
	Medium	1.0043	1.0125	1.0610	1.2410
	Large	3.0065	3.0250	3.1220	3.4820

TABLE 8 CPU usage under MitM ARP spoofing for ML and CNN methods in SDN across multiple topology types and sizes.

Network topology type	Network size	NB (%)	KNN (%)	RF (%)	CNN (%)
Single network	Small	7.11	13.78	11.38	22.92
	Medium	8.32	15.52	12.66	25.29
	Large	9.57	17.35	14.01	27.37
Tree network	Small	8.73	15.77	12.02	25.25
	Medium	10.02	17.05	13.94	27.42
	Large	11.54	19.10	16.29	30.01
Mesh network	Small	9.78	17.30	13.94	27.12
	Medium	13.49	19.06	16.66	29.24
	Large	17.71	21.12	19.92	32.48

offer enhanced detection capabilities, it requires substantially more computational resources, making it less efficient for environments with limited CPU capacity compared to traditional ML methods, particularly NB and RF.

TABLE 9 CPU usage under MitM ARP flooding for ML and CNN methods in SDN across multiple topology types and sizes.

Network topology type	Network size	NB (%)	KNN (%)	RF (%)	CNN (%)
Single network	Small	41.27	71.27	57.27	75.27
	Medium	49.71	79.71	55.71	83.71
	Large	53.75	83.75	59.75	87.75
Tree network	Small	49.17	79.17	65.17	83.17
	Medium	51.29	81.29	67.29	85.29
	Large	55.88	85.88	71.88	89.88
Mesh network	Small	52.72	82.72	68.72	86.72
	Medium	56.84	86.84	72.84	90.84
	Large	59.17	89.17	75.17	93.17

TABLE 10 CPU usage after mitigating MitM attacks for ML and CNN methods in SDN across multiple topology types and sizes.

Network topology type	Network Size	NB (%)	KNN (%)	RF (%)	CNN (%)
Single network	Small	3.22	4.13	4.55	12.74
	Medium	3.71	4.93	5.18	13.88
	Large	4.01	5.11	5.75	15.12
Tree network	Small	4.09	4.72	6.26	14.12
	Medium	5.37	5.55	7.78	15.25
	Large	5.92	6.29	9.03	17.04
Mesh network	Small	5.72	6.89	8.06	17.12
	Medium	7.24	8.18	9.72	19.29
	Large	8.93	9.52	11.24	22.02

The Table 10 shows the CPU usage percentages after mitigating MitM attacks for different ML methods and a CNN method across various SDN network topologies and sizes. As seen, the CNN method consistently incurs the highest CPU usage post-mitigation across all scenarios, indicating its substantial computational overhead. Among the ML algorithms, KNN tends to use more CPU resources than NB and RF, with RF generally consuming more than NB. The CPU usage for all methods increases with network size, with CNN exhibiting the steepest rise. These results suggest that while CNN might offer robust mitigation capabilities, it comes at the cost of significantly higher CPU usage, making it less suitable for environments with constrained computational resources compared to ML methods like NB and RF, which are more efficient in terms of CPU usage.

5.3 ML algorithm selection

The ML algorithm chosen for our framework IPDS leans toward CNN due to its superior performance in accuracy and

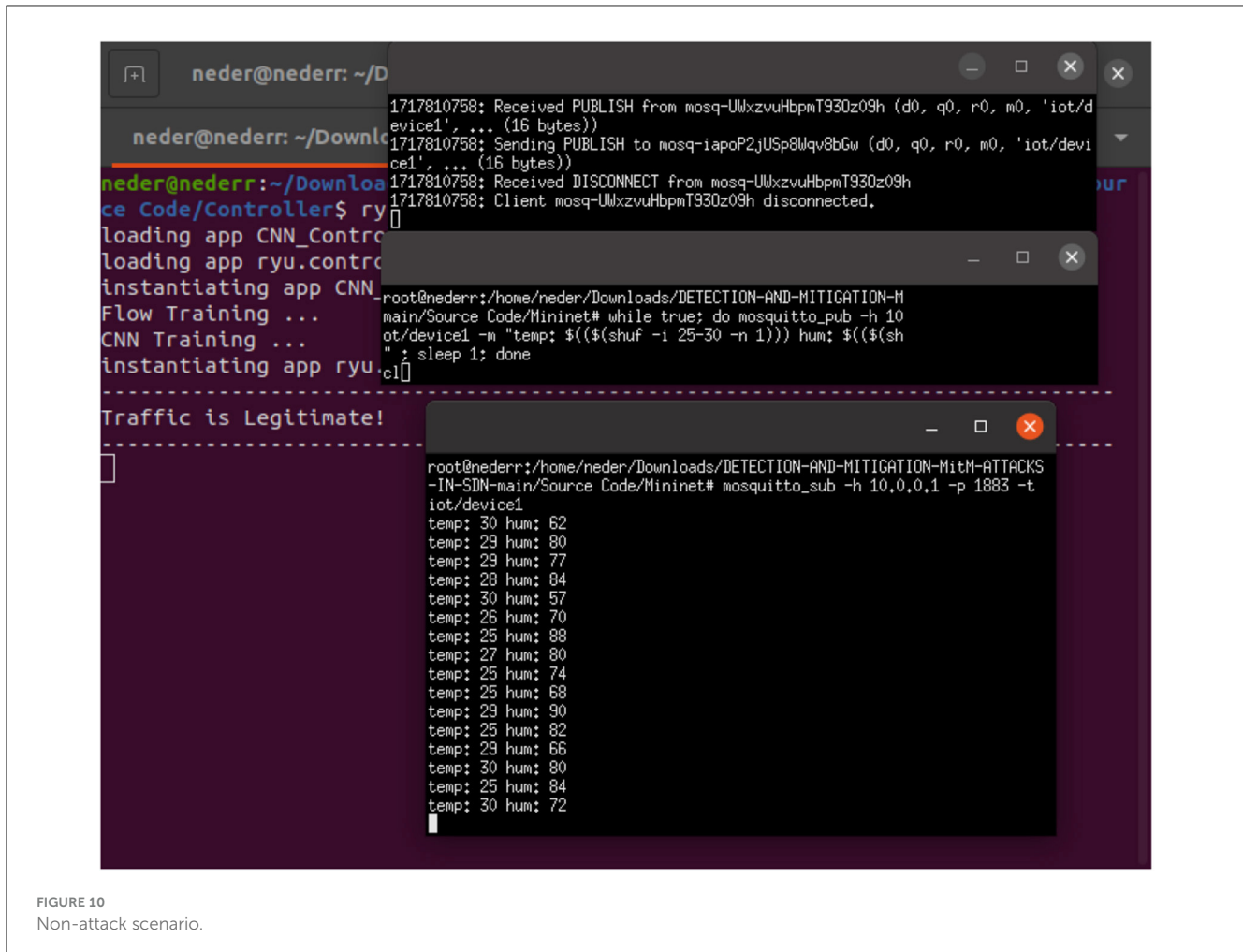


FIGURE 10
Non-attack scenario.

FAR compared to other algorithms. It also shows comparable performance to other algorithms across various metrics.

5.4 Interpretability of recommendations

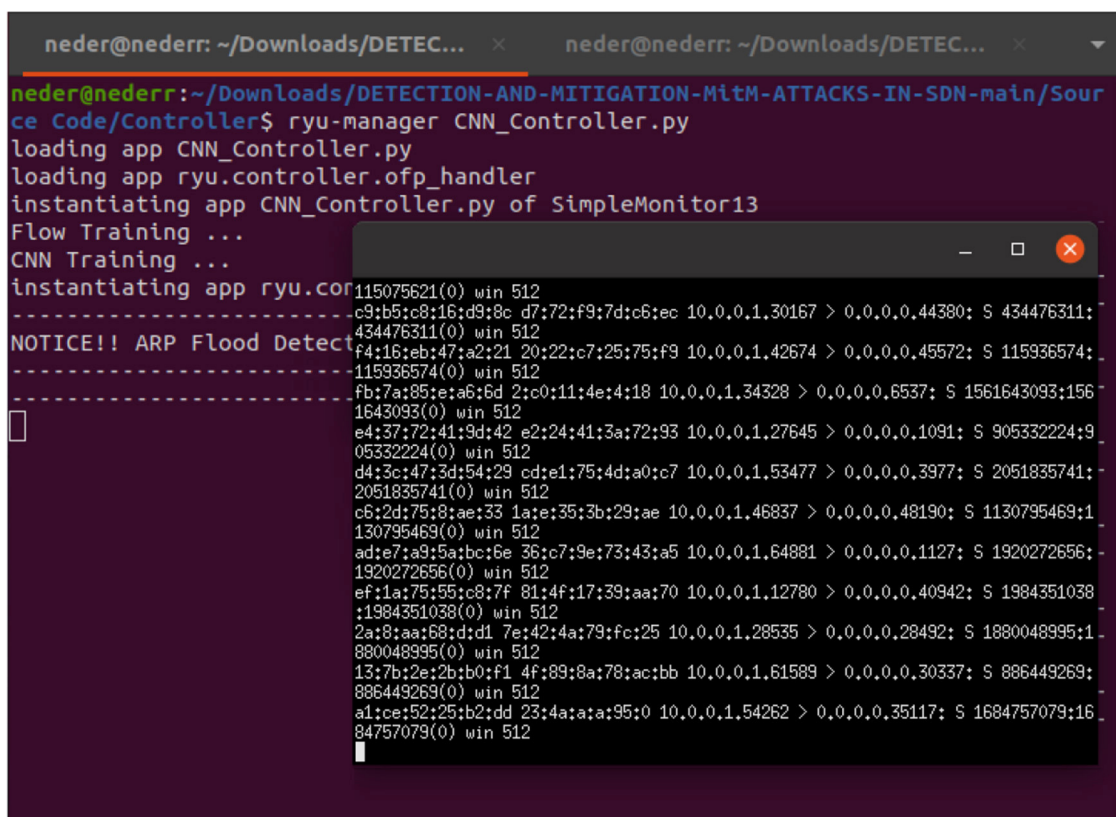
The security recommendations in our proposed framework are derived by interpreting how key features such as IP addresses, MAC addresses, port numbers, packet size, and protocol types influence the model's decision-making process. By analyzing the patterns in these features, the CNN model can detect abnormal behaviors indicative of ARP spoofing or ARP flooding attacks. For instance, variations in source or destination MAC/IP addresses may trigger alerts, and abnormal traffic volumes detected through Packet size bytes or `Tp_src/Tp_dst` fields suggest possible attacks. Moreover, the model's high detection accuracy (99.96%) and low false alarm rate (0.02%) are reflective of its ability to effectively classify and isolate attack traffic, as demonstrated through multiple network environments (single, tree, mesh) and sizes (4, 16, and 64 hosts). This interpretability ensures that security recommendations align with observable network behaviors, providing actionable insights for mitigating potential threats in real-time.

5.5 Detection

Supplementary Figure 3 describes the MitM detection algorithm identifies whether a MitM attack is present based on the flow predictions. If the majority (more than 90%) of the traffic is legitimate, it assumes no attack. If less than 90% is legitimate, it uses the classification of the non-legitimate traffic to make a decision. The actual labels for non-legitimate traffic are divided into two classes based on their parity (odd or even) when excluding legitimate traffic labeled as 0. This method aims to minimize false alarms by setting a high threshold for detecting attacks.

5.6 Mitigation

Supplementary Figure 4 describes the MitM mitigation algorithm, which mitigates MitM attacks by blocking malicious traffic based on a prediction value. It takes the `malicious_flow_traffic` and `predict` as inputs, extracts the `source_port`, and performs actions based on the prediction. If `predict` is 1, it updates the switch's flow table and blocks packets from the source port. If `predict` is 2, it additionally clears ARP entries for the destination port. These steps aim to isolate and



```

neder@nederr: ~/Downloads/DETEC... x neder@nederr: ~/Downloads/DETEC... x
neder@nederr:~/Downloads/DETECTION-AND-MITIGATION-MitM-ATTACKS-IN-SDN-main/Source Code/Controller$ ryu-manager CNN_Controller.py
loading app CNN_Controller.py
loading app ryu.controller.ofp_handler
instantiating app CNN_Controller.py of SimpleMonitor13
Flow Training ...
CNN Training ...
instantiating app ryu.controller.ofp_handler
-----
115075621(0) win 512
c9:b5:c8:16:d9:8c d7:72:f9:7d:c6:ec 10.0.0.1.30167 > 0.0.0.0.44380: S 434476311:
434476311(0) win 512
f4:16:eb:47:a2:21 20:22:c7:25:f9 10.0.0.1.42674 > 0.0.0.0.45572: S 115936574:
115936574(0) win 512
-----
fb:7a:85:e:a6:6d 2:c0:11:4e:4:18 10.0.0.1.34328 > 0.0.0.0.6537: S 1561643093:156
1643093(0) win 512
e4:37:72:41:9d:42 e2:24:41:3a:72:93 10.0.0.1.27645 > 0.0.0.0.1091: S 905332224:9
05332224(0) win 512
d4:3c:47:3d:54:29 cd:e1:75:4d:a0:c7 10.0.0.1.53477 > 0.0.0.0.3977: S 2051835741:
2051835741(0) win 512
c6:2d:75:8:ae:33 1a:e:35:3b:29:ae 10.0.0.1.46837 > 0.0.0.0.48190: S 1130795469:1
130795469(0) win 512
ad:e7:a9:5a:bc:6e 36:c7:9e:73:43:a5 10.0.0.1.64881 > 0.0.0.0.1127: S 1920272656:
1920272656(0) win 512
ef:1a:75:55:c8:7f 81:4f:17:39:aa:70 10.0.0.1.12780 > 0.0.0.0.40942: S 1984351038
:1984351038(0) win 512
2a:8:aa:68:d:d1 7e:42:4a:79:fc:25 10.0.0.1.28535 > 0.0.0.0.28492: S 1880048995:1
880048995(0) win 512
13:7b:2e:2b:b0:f1 4f:89:8a:78:ac:bb 10.0.0.1.61589 > 0.0.0.0.30337: S 886449269:
886449269(0) win 512
a1:ce:52:25:b2:dd 23:4a:a:a:95:0 10.0.0.1.54262 > 0.0.0.0.35117: S 1684757079:16
84757079(0) win 512
-----
NOTICE!! ARP Flood Detected
-----

```

FIGURE 11
ARP Flood attacks detection.

block malicious traffic swiftly to prevent further exploitation of the network.

5.7 Deployment in an SDN environment

To deploying our CNN-based model within an SDN environment for smart home utilization, we implemented a topology featuring 8 hosts, 7 switches, and 1 controller, illustrated in [Supplementary Figure 5](#). Host h1 operates as a temperature and humidity IoT device, consistently transmitting real-time data to authorized subscribers. Host h2 serves as a subscriber device, while h3 functions as a MitM attacker. During ARP flood incidents, h3 targets h1. In ARP attacks, h3 intercepts data exchanges between h1 and h2.

5.7.1 Integrity of training data

After deploying our CNN-based IDPS framework, we focused on ensuring both the integrity of the training data and the model's resilience against potential attacks. To address these challenges, we implemented data validation and monitoring processes to ensure data integrity, detect anomalies in the training data, and maintain overall reliability. Additionally, we regularly update the model with new data to adapt to evolving threats and changing data patterns. These combined efforts ensure data quality and

enhance the model's performance in live SDN networks and real-world scenarios.

5.7.2 Detection of MitM attacks

To detect MitM attacks, we utilized the algorithm mentioned previously.

5.7.2.1 Normale state

In the normal state or non-attack scenario, as illustrated in [Figure 10](#), host h1 publishes real-time temperature and humidity data every second via the MQTT protocol on port 1883 to the topic named "device1/iot". Host 2 subscribes to the same topic, "device1/iot", to collect the data as a legitimate device. The Ryu controller, based on a CNN model, continuously detects in real-time that this traffic is legitimate.

5.7.2.2 ARP flood attack detection

We used macof tools to launch ARP flood attacks targeting host h1, which is the IoT device. The goal was to overwhelm the switch's resources, causing network congestion, and to exhaust the switch's processing capacity, memory, and forwarding tables. As [Figure 11](#) shows, The Ryu controller based on a CNN model could detect ARP flood attacks targeting IoT device h1 in real time.


```

neder@nederr:~/Downloads/DETECTION-AND-MITIGATION-MitM-ATTACKS-IN-SDN-main/Sour
ce Code/Controller$ ryu-man
loading app CNN_Controller.
loading app ryu.controller.
instantiating app CNN_Contr
Flow Training ...
CNN Training ...

-----
NOTICE!! ARP Spoof Detected
-----

Sat Jun 8 02:30:30 2024 [578391]
TCP 10.0.0.2:35340 --> 10.0.0.1:1883 | S (0)

Sat Jun 8 02:31:05 2024 [57482]
TCP 10.0.0.2:35340 --> 10.0.0.1:1883 | S (0)

Sat Jun 8 02:35:41 2024 [826417]
TCP 10.0.0.1:1883 --> 10.0.0.2:38334 | AP (31)
0...iot/device1temp; 28 hum; 79

Sat Jun 8 02:35:53 2024 [732386]
TCP 10.0.0.2:38334 --> 10.0.0.1:1883 | AP (2)
**

Sat Jun 8 02:35:57 2024 [490382]
TCP 10.0.0.2:38334 --> 10.0.0.1:1883 | AP (2)
**

Sat Jun 8 02:36:26 2024 [684035]
TCP 10.0.0.1:1883 --> 10.0.0.2:38334 | AP (31)
0...iot/device1temp; 28 hum; 79

```

FIGURE 12
ARP Spoof attacks detection.

5.7.2.3 ARP spoof attacks detection

We used Ettercap tools to launch an ARP spoofing attack to intercept communication between the IoT device h1 and the subscriber host device h2. As shown in Figure 12, the attacker h3 could collect temperature and humidity data sent by the publishing IoT device h1 to the subscriber h2 via the MQTT protocol on port 1883, on the topic `iot/device1`. The Ryu controller, based on a CNN model, could detect ARP spoofing in real time.

5.7.3 Mitigation of MitM attacks

To mitigate the impact of MitM attacks in SDN, we block the `in_port`, the switch port on which a packet enters in the case of ARP spoofing and ARP flooding, as shown in Figure 13. Additionally, we clear ARP entries on the specific port (`in_port`) to further prevent malicious activity in the case of ARP flooding, as shown in Figure 14.

6 Discussion

In this section, we analyze and interpret our results, comparing them with closely related works. The selection criteria for these works include their novelty, use of the SDN framework, and their effectiveness in mitigating Man-in-the-Middle (MitM)

attacks on IoT devices. We identified three works that meet these criteria:

Alani et al. (2023), proposed an ARP spoofing detection system for IoT systems. Their model was trained on the IoT Network Intrusion Dataset using a balanced dataset comprising 428,918 samples, achieving an accuracy of 99.98% with a False Alarm Rate (FAR) of 0.026%. Their approach outperformed existing related work by 0.38% in accuracy and significantly reduced the FAR. However, their model exhibited longer testing times compared to other related works.

Khedr et al. (2023), focused on detecting and mitigating DDoS and ARP attacks in SDN for IoT networks using a MEV algorithm with six classifiers trained on the Edge-IIoTset dataset for enhanced anomaly detection. The MEV model achieved an accuracy of 99.89% and a FAR of 0.016%, outperforming other related works by 0.5% in accuracy and 0.024% in FAR.

Hnamte and Hussain (2024), built an IDPS harnessing SDN technology. They integrated a deep learning-based Deep Neural Network (DNN) model to detect and mitigate ARP spoofing attacks across a small SDN network. The model achieved a perfect detection rate with an accuracy of 100% and a FAR of 0%, outperforming previous works by 0.02% in accuracy and reducing the FAR by 0.0009%.

Our proposed method utilizes a CNN with PCA for feature reduction, trained on a synthetic CSV dataset generated by a Ryu controller. It classifies ARP flood, ARP spoofing, and normal traffic into three distinct classes. Compared to existing methods,

```
neder@nederr:~/Downloads/DETECTION-AND-MITIGATION-MitM-ATTACKS-IN-SDN-main/Source Code/Controller$ ryu-manager CNN_Controller.py
loading app CNN_Controller.py
loading app ryu.controller.ofp_handlers
instantiating app CNN_Controller.py
Flow Training ...
CNN Training ...
instantiating app ryu.controller
-----
NOTICE!! ARP Spoof Detected!!!
Mitigation process in progress!
-----
Traffic is Legitimate!
-----

* |=====| 100,00 %
2 hosts added to the hosts list...
ARP poisoning victims:
GROUP 1 : 10.0.0.1 00:00:00:00:00:01
GROUP 2 : 10.0.0.2 00:00:00:00:00:02
Starting Unified sniffing...

Text only Interface activated...
Hit 'h' for inline help

Mon Jun 10 01:00:40 2024 [969259]
10.0.0.2:0 --> 10.0.0.1:0 | (0)
```

FIGURE 13 Mitigate ARP spoof attacks detection.

```
neder@nederr:~/Downloads/DETECTION-AND-MITIGATION-MitM-ATTACKS-IN-SDN-main/Source Code/Controller$ ryu-manager CNN_Controller.py
loading app CNN_Controller.py
loading app ryu.controller.ofp_handlers
instantiating app CNN_Controller.py
Flow Training ...
CNN Training ...
instantiating app ryu.controller
-----
NOTICE!! ARP Flood Detected!!!
Mitigation process in progress!
-----
Traffic is Legitimate!
-----

29:fb:dd:6f:ae:d3 a:5a:cf:39:61:b6 10.0.0.1.11688 > 0.0.0.0.40897: S 41985
8555:419858555(0) win 512
30:f1:ec:68:17:3a 9:92:f0:2c:6c:9 10.0.0.1.41044 > 0.0.0.0.4568: S 1082411
797:1082411797(0) win 512
f6:fc:3d:34:b1:9c 8b:40:ac:5:85:c8 10.0.0.1.39433 > 0.0.0.0.36547: S 13902
86252:1390286252(0) win 512
69:49:68:18:b7:6c 69:a3:c9:53:3d:76 10.0.0.1.19170 > 0.0.0.0.35350: S 7980
56079:798056079(0) win 512
34:66:8a:51:e2:2d e6:45:45:5c:73:7c 10.0.0.1.23618 > 0.0.0.0.47981: S 6934
95830:693495830(0) win 512
6b:ee:56:6d:44:23 d5:8:c4:7:3c:d9 10.0.0.1.44363 > 0.0.0.0.8052: S 7771542
19:777154219(0) win 512
6c:14:11:25:c1:2b 2c:f9:ce:2e:22:74 10.0.0.1.38067 > 0.0.0.0.60245: S 9543
54414:954354414(0) win 512
ba:17:7e:1e:e7:d 54:a2:3e:58:e1:36 10.0.0.1.43125 > 0.0.0.0.22796: S 18263
13534:1826313534(0) win 512
cf:28:80:c:87:7d ab:66:35:5b:4f:3a 10.0.0.1.25893 > 0.0.0.0.19176: S 85645
7082:856457082(0) win 512
12:31:21:47:fa:51 66:93:35:43:a5:37 10.0.0.1.33372 > 0.0.0.0.24480: S 1661
836429:1661836429(0) win 512
```

FIGURE 14 Mitigate ARP flood attacks detection.

TABLE 11 Performance comparison.

References	Dataset used	Method used	Accuracy	FAR	Mitigation	Environment
Alani et al. (2023)	IoT Network Intrusion Dataset	Deep learning	99.98%	0.026%	X	SDN-IoT
Khedr et al. (2023)	Edge-IIoTset dataset	MEV algorithm	99.89%	0.016%	X	SDN-IoT with ODL controller
Hnamte and Hussain (2024)	The IoT Network Intrusion MITM-ARP-Spoofing dataset	DNN method	100%	0%	X	SDN-IoT with Ryu controller
Our CNN method	Synthetic	CNN based PCA method	99.96%	0.02%	-Block source_port -Clear ARP cache	SDN-IoT with Ryu controller

our approach achieves a detection and mitigation accuracy of 99.96%, outperforming other approaches by 0.09% in accuracy. Additionally, it achieves a FAR of 0.02%, which is 0.006% better than the aforementioned approaches.

Table 11 summarizes the comparison between related works and our work.

7 Conclusion

This paper presents a CNN method to detect and mitigate MitM attacks in a SDN environment. The proposed method functions as an IDPS and is specifically designed for IoT devices in smart home settings. Our framework, which employs CNNs, achieves a 99.96% accuracy rate in detecting MitM attacks and maintains a low False Alarm Rate (FAR) of 0.02%. Additionally, To address the scalability concern in this article, the proposed CNN-based IDPS was tested on varying network sizes (4, 16, and 64 hosts) and topologies (mesh, single, and tree) within SDN environments, showing strong detection performance with minimal degradation. This framework not only detects MitM attacks but also focuses on quickly isolating and blocking malicious traffic to prevent further exploitation by an attacker. By updating the switch's flow table and managing ARP entries, it ensures that the network can effectively handle identified threats. We tested the performance of the IDPS model on SDN using Mininet to create a network with hosts and IoT devices, with Ryu as the controller. Future work will focus on testing the system in larger and more varied real-world SDN environments to further validate its scalability and performance. Our work addresses the challenge of managing increased computational demands on the Ryu controller and real-time processing requirements posed by CNN-based IDPS techniques, which could affect system performance. To mitigate these issues, strategies such as offloading the IDPS to more powerful servers, employing distributed controllers, optimizing models through compression or hardware acceleration, and exploring faster algorithms like ensemble methods are essential for ensuring scalability in larger networks. However, the implementation of CNN-based IDPS introduces computational overhead, impacting real-time detection capabilities as device numbers grow. Despite enhancing security, these methods may strain processing resources, particularly in resource-constrained IoT environments. Future optimizations will focus on simplifying models and adopting efficient architectures to minimize overhead while maintaining effective performance.

Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: <https://www.kaggle.com/datasets/nedernido/mitm-multiclass-dataset>.

Author contributions

NK: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. YB: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. MO-E: Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. NY: Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. RB: Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. AY: Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing.

Funding

The author(s) declare that no financial support was received for the research, authorship, and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships

that could be construed as a potential conflict of interest.

The author(s) declared that they were an editorial board member of Frontiers, at the time of submission. This had no impact on the peer review process and the final decision.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

- Abdullah, D. M., and Abdulazeez, A. M. (2021). Machine learning applications based on SVM classification a review. *Qubahan Acad. J.* 1, 81–90. doi: 10.48161/qaj.v1n2a50
- Abraham, J. A., and Bindu, V. R. (2021). "Intrusion detection and prevention in networks using machine learning and deep learning approaches: a review," in *2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)* (Coimbatore: IEEE).
- Adhikari, T., Kule, M., and Khan, A. K. (2022). "An ECDH and AES based encryption approach for prevention of MiTM in SDN southbound communication interface," in *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (Kharagpur: IEEE), 1–5.
- Ahuja, N., Singal, G., Mukhopadhyay, D., and Nehra, A. (2022). Ascertain the efficient machine learning approach to detect different arp attacks. *Comp. Elect. Eng.* 99:107757. doi: 10.1016/j.compeleceng.2022.107757
- Alani, M. M., Awad, A. I., and Barka, E. (2023). Arp-probe: An arp spoofing detector for internet of things networks using explainable deep learning. *Internet of Things* 23:100861. doi: 10.1016/j.iot.2023.100861
- Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujali, A., Al-Shamma, O., Santamaria, J., et al. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *J. Big Data* 8, 1–74. doi: 10.1186/s40537-021-00444-8
- Aoueilayne, M. O. E., Karmous, N., Bouallegue, R., Youssef, N., and Yazidi, A. (2024). "Detecting and mitigating MiTM attack on IOT devices using SDN," in *International Conference on Advanced Information Networking and Applications* (Cham: Springer Nature Switzerland).
- Asadollahi, S., Goswami, B., and Sameer, M. (2018). "RYU controller's scalability experiment on software defined networks," in *2018 IEEE international conference on current trends in advanced computing (ICCTAC)* (Bangalore: IEEE).
- Ashraf, S. (2021). A proactive role of iot devices in building smart cities. *Intern. Things Cyber-Phys. Syst.* 1, 8–13. doi: 10.1016/j.iotcps.2021.08.001
- Bärli, E. M., Yazidi, A., Viedma, E. H., and Hougnerud, H. (2021). DoS and DDoS mitigation using variational autoencoders. *Comp. Networ.* 199:108399. doi: 10.1016/j.comnet.2021.108399
- Beaman, C., Barkworth, A., Akande, T. D., Hakak, S., and Khan, M. K. (2021). Ransomware: recent advances, analysis, challenges and future research directions. *Comp. Security* 111:102490. doi: 10.1016/j.cose.2021.102490
- Chang, Q., Zhang, Z., Wei, F., Wong, J., Pedrycz, W., and Pal, N. R. (2024). Adaptive nonstationary fuzzy neural network. *Knowl.-Based Syst.* 288:111398. doi: 10.1016/j.knsys.2024.111398
- Cherian, M. M., and Varma, S. L. (2022). Mitigation of DDoS and MiTM attacks using belief based secure correlation approach in sdn-based IoT networks. *Int. J. Comp. Networ. Inform. Security* 14:52. doi: 10.5815/ijcnis.2022.01.05
- Conti, M., Dragoni, N., and Lesyk, V. (2016). A survey of man in the middle attacks. *IEEE Commun. Surv. Tutor.* 18, 2027–2051. doi: 10.1109/COMST.2016.2548426
- Deore, R. E., Shashibhushan, B., Mahadik, D., and Godase, G. (2024). The two-sided SPT sign charts. *Qual. Reliab. Eng. Int.* 40, 1014–1025. doi: 10.1002/qre.3451
- Du, J., Gao, X., Wang, J., Liu, S., Cao, W., and Song, Y. (2021). "Research on an approach of arp flooding suppression in multi-controller sdn networks," in *2021*

Supplementary material

The Supplementary Material for this article can be found online at: <https://www.frontiersin.org/articles/10.3389/fcomp.2024.1477501/full#supplementary-material>

SUPPLEMENTARY FIGURE 1
SDN architecture in IoT.

SUPPLEMENTARY FIGURE 2
Histogram of row counts for dataset label classes.

SUPPLEMENTARY FIGURE 3
MitM attack detection algorithm.

SUPPLEMENTARY FIGURE 4
MitM attack mitigation algorithm.

SUPPLEMENTARY FIGURE 5
SDN network topology.

IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom) (New York City, NY: IEEE).

Ezeofor, C. J., and Ulasi, A. G. (2014). Analysis of network data encryption & decryption techniques in communication systems. *Int. J. Innovat. Res. Sci. Eng. Technol.* 3, 17797–17807. doi: 10.15680/IJIRSET.2014.0312008

Farhin, F., Sultana, I., Islam, N., Kaiser, M. S., Rahman, M., and Mahmud, M. (2020). "Attack detection in internet of things using software defined network and fuzzy neural network," in *2020 Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)* (Kitakyushu: IEEE).

Fathima, K. M. M., and Santhiyakumari, N. (2021). "A survey on network packet inspection and arp poisoning using wireshark and ettercap," in *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)* (Coimbatore: IEEE).

Gnad, D. R. E., Oboril, F., and Tahoori, M. (2017). "Voltage drop-based fault attacks on fpgas using valid bitstreams," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)* (Ghent: IEEE).

Gowda, S. S., and Dayananda, R. B. (2023). "Detection and prevention of arp attack in software defined networks," in *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (Delhi: IEEE), 1–5.

Haakegaard, R., and Lang, J. (2015). *The Elliptic Curve Diffie-Hellman (ECDH)*. Available at: <https://koelab.cs.ucsb.edu/teaching/ecc/project/2015Projects/Haakegaard+Lang.pdf> (accessed September 10, 2024).

Haddouchi, M., and Berrado, A. (2019). "A survey of methods and tools used for interpreting random forest," in *2019 1st International Conference on Smart Systems and Data Science (ICSSD)* (Rabat: IEEE).

Hnamte, V., and Hussain, J. (2024). Enhancing security in software-defined networks: an approach to efficient arp spoofing attacks detection and mitigation. *Telemat. Inform. Reports* 14:100129. doi: 10.1016/j.teler.2024.100129

Jiménez, A. B., Lázaro, J. L., and Dorronsoro, J. R. (2008). "Finding optimal model parameters by discrete grid search," in *Innovations in Hybrid Intelligent Systems* (Cham: Springer Berlin Heidelberg), 120–127.

Karmous, N., Abdelkader, M. O., Abdelkader, M., and Romdhani, L. (2024). Software-defined-networking-based one-versus-rest strategy for detecting and mitigating distributed denial-of-service attacks in smart home internet of things devices. *Sensors* 24:5022. doi: 10.3390/s24155022

Karmous, N., Aoueilayne, M. O., Abdelkader, M., and Youssef, N. (2023). "Enhanced machine learning-based sdn controller framework for securing iot networks," in *Advanced Information Networking and Applications, 2023* (Cham: Springer).

Kaur, S., Singh, J., and Ghuman, N. S. (2014). "Network programmability using pox controller," in *ICCCS International Conference on Communication, Computing & Systems* (Chengdu: IEEE), 138.

Khedr, W. I., Gouda, E., and Mohammed, E. R. (2023). P4-hldmc: A novel framework for ddos and arp attack detection and mitigation in SD-IoT networks using machine learning, stateful p4, and distributed multi-controller architecture. *Mathematics* 11:3552. doi: 10.3390/math11163552

- Krishnan, K. N., Jenu, R., Joseph, T., and Slipa, M. (2018). "Blockchain based security framework for iot implementations," in *2018 International CET Conference on Control, Communication, and Computing (IC4)* (Thiruvananthapuram: IEEE).
- Manoharan, H., Manoharan, A., Selvarajan, S., and Venkatachalam, K. (2023). *Implementation of Internet of Things with Blockchain Using Machine Learning Algorithm: Enhancement of Security with Blockchain* (Pennsylvania: IGI Global), 399–430.
- Meghana, J. Subashri, T., and Vimal, K. R. (2017). "A survey on arp cache poisoning and techniques for detection and mitigation," in *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)* (Chennai: IEEE).
- Mishra, B. (2018). "Performance evaluation of MQTT broker servers," in *International Conference on Computational Science and Its Applications* (Cham: Springer International Publishing).
- Mittal, S. (2020). A survey on modeling and improving reliability of dnn algorithms and accelerators. *J. Syst. Architect.* 104:101689. doi: 10.1016/j.sysarc.2019.101689
- Mohammadpour, L., Ling, T. C., Liew, C. S., and Aranyanfar, A. (2022). A survey of cnn-based network intrusion detection. *Appl. Sci.* 12:8162. doi: 10.3390/app12168162
- Mohammed, M. S., and Talib, H. A. (2024). Using machine learning algorithms in intrusion detection systems: a review. *Tikrit J. Pure Sci.* 29, 63–74. doi: 10.25130/tjps.v29i3.1553
- Mwange, C. M., and Cankaya, E. C. (2024). "Android trojan horse spyware attack: a practical implementation," in *2024 12th International Symposium on Digital Forensics and Security (ISDFS)* (San Antonio, TX: IEEE).
- Padmaja, M., Shitharth, S., K., Prasuna, K., Chaturvedi, A., Kshirsagar, P. R., Vani, A., et al. (2022). Grow of artificial intelligence to challenge security in iot application. *Wireless Pers. Commun.* 127, 1829–1845. doi: 10.1007/s11277-021-08725-4
- Rakine, I., El Guemmat, K., Ouahabi, S., and Issam, A. (2024). "IoT intrusion detection: a review of ml and dl-based approaches," in *2024 4th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)* (Fez, Morocco: IEEE).
- Ramaswamy, S. L., and Chinnappan, J. (2023). Review on positional significance of lstm and cnn in the multilayer deep neural architecture for efficient sentiment classification. *J. Intellig. Fuzzy Syst.* 45, 6077–6105. doi: 10.3233/JIFS-230917
- Reddy, E. M. K., Gurralla, A., Hasitha, V. B., and Kumar, K. V. R. (2022). "Introduction to naive bayes and a review on its subtypes with applications," in *Bayesian Reasoning and Gaussian Processes for Machine Learning Applications*, 1–14. doi: 10.1201/9781003164265-1
- Rostami, M., and Goli-Bidgoli, S. (2024). An overview of qos-aware load balancing techniques in sdn-based iot networks. *J. Cloud Comp.* 13:89. doi: 10.1186/s13677-024-00651-7
- Saha, A., De Sarkar, G., Dutta, D., and Karmakar, K. (2024). A survey on the advanced encryption standard (AES): a pillar of modern cryptography. *Int. J. Comp. Sci. Mobile Comp.* 13, 68–87. doi: 10.47760/ijcsmc.2024.v13i04.008
- Sankar, R. (2022). *MAC flooding with MACOF & some major countermeasures. Kali Linux Tutorials*. Available at: <https://kalinixtutorials.com/macof/> (accessed July 19, 2024).
- Saridakumar, N., Anusuya, K. V., and Balasaraswathi, B. (2021). "Detection and mitigation of MITM attack in software defined networks," in *Proceedings of the First International Conference on Combinatorial and Optimization, ICCAP 2021* (Chennai: IEEE).
- Saridakumar, N., Anusuya, K. V., and Krishnakumar, S. (2023). "Detection of arp spoofing attacks in software defined networks," in *2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS)* (Coimbatore: IEEE), 422–426.
- Sebbar, A., Zkik, K., Baddi, Y., Boulmalf, M., and El Kettani, M. D. (2020). MITM detection and defense mechanism cbna-rf based on machine learning for large-scale sdn context. *J. Ambient Intell. Humaniz. Comput.* 11, 5875–5894. doi: 10.1007/s12652-020-02099-4
- Selvarajan, S., Srivastava, G., Khadidos, A. O., Khadidos, A. O., Baza, M., Alshehri, A., et al. (2023). An artificial intelligence lightweight blockchain security model for security and privacy in iiot systems. *J. Cloud Comp.* 12:38. doi: 10.1186/s13677-023-00412-y
- Shakir, V., and Mohsin, A. (2024). A comparative analysis of intrusion detection systems: Leveraging algorithm classifications and feature selection techniques. *J. Appl. Sci. Technol. Trends* 5, 34–45. doi: 10.38094/jastt501186
- Shitharth, S., Mohammed, G. B., Ramasamy, J., and Srivel, R. (2023). "Intelligent intrusion detection algorithm based on multi-attack for edge-assisted internet of things," in *Security and Risk Analysis for Intelligent Edge Computing* (Cham: Springer International Publishing), 119–135.
- Stafford, V. (2020). "Zero trust architecture," in *NIST Special Publication 800*, 207.
- Syed, S. A., Manickam, S., Uddin, M., Alsufyani, H., Shorfuzzaman, M., Selvarajan, S., et al. (2024). Dickson polynomial-based secure group authentication scheme for internet of things. *Sci. Rep.* 14:4947. doi: 10.1038/s41598-024-55044-2
- Toldinas, J., Lozinskis, B., Baranauskas, E., and Dobrovolskis, A. (2019). "MQTT quality of service versus energy consumption," in *2019 23rd International Conference Electronics* (Palanga: IEEE).
- Widodo, A. O., and Bambang Setiawan, R. I. (2024). Machine learning-based intrusion detection on multi-class imbalanced dataset using smote. *Procedia Comput. Sci.* 234:578–583. doi: 10.1016/j.procs.2024.03.042
- Wukkadada, B., Wankhede, K., Nambiar, R., and Nair, A. (2018). "Comparison with HTTP and MQTT in internet of things (IoT)," in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)* (Coimbatore: IEEE).
- Zhang, H., Yin, G., and Rubin, D. B. (2024). PCA rerandomization. *Can. J. Statist.* 52, 5–25. doi: 10.1002/cjs.11765
- Zhang, S. (2021). Challenges in knn classification. *IEEE Trans. Knowl. Data Eng.* 34, 4663–4675. doi: 10.1109/TKDE.2021.3049250
- Zhao, R. (2022). "NSL-KDD," in *IEEE Dataport*. doi: 10.21227/8rpg-qt98