Journal of Network and Computer Applications 246 (2026) 104396

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Research paper

ARProof: A cross-protocol approach to detect and mitigate ARP-spoofing attacks in smart home networks

Md Mizanur Rahman [a,*], Faycal Bouhafs [a], Sayed Amir Hoseini [a], Frank den Hartog [a,b]

[a] School of Systems and Computing, University of New South Wales, Canberra, ACT 2610, Australia
[b] Network Engineering and Cybersecurity, University of Canberra, Bruce, ACT 2617, Australia

ARTICLE INFO

Keywords:
Smart home
Internet of Things (IoT)
ARP spoofing
Man-in-the-Middle (MITM) attacks
Machine Learning

ABSTRACT

Smart homes are increasingly vulnerable to cyberattacks that lead to network instability, causing homeowners to lodge complaints with their Broadband Service Providers (BSPs). Therefore, effective and timely detection of cyberattacks is crucial for both customers and BSPs. Address Resolution Protocol (ARP) spoofing is one of the most common attacks that can facilitate larger and more severe follow-up attacks. Unfortunately, there are currently no methods that can effectively detect and mitigate ARP spoofing in smart homes from a BSP's perspective. Current Machine Learning (ML)-based methods often rely on a single dataset from a controlled lab environment designed to mimic a single home, assuming that the results will generalize to all smart homes. Our findings indicate that this assumption is flawed. They are also unsuitable for smart homes from a BSP's perspective, as they require custom applications, introduce additional overhead, and often rely on the injection of probing traffic into the network. To address these issues, we developed an algorithm that can detect ARP spoofing in smart home networks, regardless of the network structure or connected devices. It uses a cross-protocol strategy by correlating ARP packets with Dynamic Host Configuration Protocol (DHCP) messages to validate address bindings. We evaluated our method using four public datasets and two real-world testbeds, achieving 100% detection accuracy in all scenarios. In addition, the algorithm requires only little computational overhead, confirming its suitability for use by BSPs to detect and mitigate ARP spoofing attacks in smart homes.

1. Introduction

The Internet of Things (IoT) is revolutionizing various sectors by connecting a diverse range of devices to the Internet. The number of IoT devices is predicted to reach 50 billion in 2030 (Sinha, 2023), and the market size is expected to grow to USD 755.98 billion (SkyQuest, 2025). A prominent example of this shift is the transition from conventional homes to smart homes (Rahman et al., 2025a). These smart homes feature advanced capabilities that enhance the performance of household appliances. For example, within the United States, households presently possess an average of 21 IoT devices. This number is expected to increase substantially by 2030, indicating a greater adoption and integration of IoT technology (Blinder, 2023).

Despite the growing popularity of smart home technologies, many IoT device manufacturers prioritize prototyping, production efficiency, and performance over security (Shaikh et al., 2018). This focus often results in inadequate security protocols for IoT devices, leaving smart home devices vulnerable to threats that can lead to network performance disruptions caused by cyberattacks. Since most smart home users are typically unaware of the underlying causes of these disruptions, they often submit support tickets to Broadband Service Providers (BSPs). The resulting surge in support requests significantly increases the load on customer service operations, and not only slows down response times, but also reduces the overall efficiency.

Although IoT security has recently gained attention in academia and industry, many challenges require further examination (Shokry et al., 2022; Sodhro et al., 2022). The Broadband Forum (Anon, 2025) has reported various types of attacks that can affect smart homes. These attacks include Denial-of-Service (DoS), jamming, eavesdropping, Man-in-the-Middle (MITM), wormhole, and IP spoofing (Broadband-Forum, 2013, 2012, 2022). MITM attacks constitute a significant concern for smart home environments (Bhushan et al., 2017) and are underexplored within the body of smart home research, with a limited number of studies dedicated specifically to addressing this form of threat Sivasankari et al. (2024). Such attacks allow adversaries to intercept, alter, or insert malicious content into communications without the knowledge of either party involved (Stallings and Brown, 2018). Studies indicate that MITM attacks represent 19% of successful cyberattacks

* Corresponding author.
E-mail addresses: md_mizanur.rahman@unsw.edu.au (M.M. Rahman), F.Bouhafs@unsw.edu.au (F. Bouhafs), s.a.hoseini@unsw.edu.au (S.A. Hoseini), frank.denhartog@canberra.edu.au (F. den Hartog).

https://doi.org/10.1016/j.jnca.2025.104396
Received 1 July 2025; Received in revised form 6 November 2025; Accepted 22 November 2025
Available online 24 November 2025

worldwide, contributing to approximately $2 billion in annual financial losses (Palatty, 2025).

One prevalent technique for executing MITM attacks is Address Resolution Protocol (ARP) spoofing (Morsy and Nashat, 2022). ARP is an essential network protocol that allows devices in a local network to identify each other and ensure data packets reach the correct destination (Alani and Alani, 2014). In ARP spoofing, an attacker sends falsified ARP messages within the local network, associating the attacker's MAC address with the IP address of a legitimate device. This manipulation allows the attacker to intercept communications and initiate an MITM attack (Xia et al., 2019). Additionally, ARP spoofing can serve as a gateway for other severe attacks, such as Distributed Denial of Service (DDoS) (Rahman et al., 2025a) and session hijacking (AbdelSalam et al., 2015). Therefore, detecting ARP spoofing is crucial, as it not only enhances network security but also reduces service disruptions that lead to customer complaints.

Unfortunately, there are no ARP spoofing detection methodologies available today that are effective as well as suitable for smart home applications. Conventional detection techniques frequently exhibit limited performance and compatibility with the broad range of IoT devices. They also might need specific software installations (Morsy and Nashat, 2022), or they depend on the availability of protocols such as Simple Network Management Protocol (SNMP) (Hsiao et al., 2009) or Software-Defined Networking (SDN) (Ma et al., 2016), which are not prevalent in modern smart homes or cellular networks (Motamary, 2021). Specifically, smart home networks commonly include low-cost IoT devices that do not support SNMP due to hardware constraints (Matoušek et al., 2021; Aboubakar et al., 2022). And although research has demonstrated that SDN switches can in theory be deployed in smaller networks for the purpose of intrusion detection (Manso et al., 2019), current SDN implementations typically rely on centralized controllers and programmable switches (Ahmad and Mir, 2021; Chen et al., 2023) with hardware fabrics that are rarely designed for use in or with consumer home networks.

More recent approaches for ARP spoofing detection often use Machine Learning (ML)-based methodologies (Anthi et al., 2021), but they also encounter limitations within smart home contexts. They often rely on datasets derived from controlled laboratory environments, thereby reducing their efficacy across various smart home configurations (Rahman et al., 2025a). They are also frequently resource-intensive, incurring high computational overhead, and often rely on the injection of probing traffic into the network, which genuine applications in the network may perceive as noise. There is thus a need for a resource-efficient solution that addresses these limitations: minimizing computational and memory demands, avoiding the generation of network-layer noise, and functioning effectively across various network environments without requiring configuration for each specific deployment.

In this paper, we present a resource-efficient algorithm for real-time ARP spoofing detection, tailored for resource-constrained environments such as smart home networks. This algorithm is based on the notion that the existing approaches focus on a single protocol layer. Instead, we apply cross-protocol analysis, a technique that examines interactions across different communication protocols. We evidence that this approach offers high detection accuracy with minimal computational overhead.

The subsequent sections of this paper are structured as follows. The next section provides a discussion on ARP spoofing attacks and reviews current detection and mitigation methods, including ML-based approaches. In Section 3, we examine the limitations of ML algorithms in heterogeneous environments such as smart homes through cross-dataset testing. To address these limitations, Section 4 presents a cross-protocol approach aimed at achieving a generalized solution for heterogeneous networks such as smart homes. This section also evaluates the effectiveness of the proposed model in detecting ARP spoofing across multiple datasets, reports real-time detection results for ARP-based and ARP-less attacks, and discusses remaining limitations. Finally, Section 5 summarizes the research findings and concludes the study.

2. Overview of ARP spoofing attacks and detection methods

This section provides a comprehensive overview of ARP spoofing attacks and the existing detection methodologies. Section 2.1 outlines the fundamental mechanisms underlying ARP spoofing attacks. Subsequently, Section 2.2 discusses conventional detection approaches, while Section 2.3 focuses on ML-based methods for detection.

2.1. ARP spoofing attacks

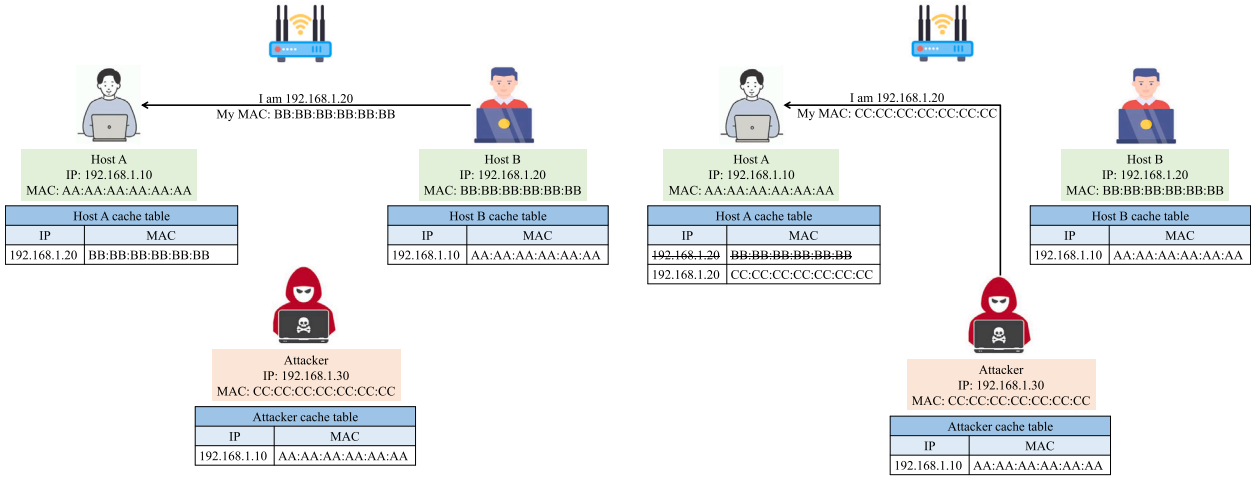
Network applications operate using network layer addresses, primarily IP addresses. However, network interface cards require a layer 2 address, specifically a MAC address. A MAC address is a unique hardware-based address used to forward packets to the next hop. As operating systems and applications are often configured with an IP address, ARP protocol enables devices to resolve the MAC address associated with a given IP address (Meghana et al., 2017). When a sender device needs the recipient's MAC address, it broadcasts an ARP request packet within an Ethernet frame. The recipient responds with an ARP reply packet which helps the sender to map the IP address to its corresponding MAC address (Bhirud and Katkar, 2011). To reduce frequent ARP requests, each device maintains a cache table that stores the IP-MAC pairs of devices it has previously communicated with (Conti et al., 2016; Trabelsi and El-Hajj, 2010; Sun et al., 2020).

ARP functions efficiently under typical network conditions, however, ARP's stateless nature allows it to be tricked by fraudulent messages, which can lead to incorrect IP-MAC mappings in the cache table (David, 1982; Wang et al., 2021). In an ARP spoofing attack, an attacker transmits fake ARP messages, altering the cache tables of other hosts on the network. As a result, data intended for a specific host is intercepted by the attacker. These attacks are executed either by responding to ARP requests for a targeted host with falsified replies or by sending unsolicited ARP responses to deceive other devices (Frankel et al., 2007; Mandal et al., 2021; Hijazi and Obaidat, 2019). Fig. 1 demonstrates how an attacker alters the ARP cache table by injecting a fraudulent ARP response.

2.2. Conventional detection methods

One of the most common approaches to ARP spoofing mitigation involves making ARP cache entries static. Hijazi and Obaidat (2018) proposed an effective solution by collecting and registering IP-MAC pairs, which were then used to generate a static ARP table. This method offers lightweight and efficient protection but is ineffective in dynamic environments such as smart homes, where devices frequently join and leave the network. A similar approach was introduced by AbdelSalam et al. (2014), which automated static ARP entry configuration using ARP_client software on devices and an ARP_server on a central node. The system authenticated and disseminated device information while integrating security measures. However, both techniques face scalability issues and require dedicated software installations, limiting their practicality in smart home networks.

Other signature-based detection mechanisms have been explored. Arote and Arya (2015) introduced an ICMP-based technique that sent ICMP trap requests to network devices and analyzed their responses. A voting mechanism then determined the presence of spoofing and updated the ARP cache. While effective, this approach may struggle with diverse IoT devices, leading to increased false positives and slower detection. Another strategy, D-ARP, proposed by Morsy and Nashat (2022), introduced signed ARP packet keys alongside traditional ARP packets to ensure request–response integrity. While this method improves accuracy, it faces challenges in large-scale smart home environments due to its reliance on cryptographic key management. Recently, the same authors proposed VB-ARP (Morsy and Nashat, 2025), a Boyer–Moore majority-voting approach (Boyer and Moore,



(a) ARP cache table for Host A and B.

(b) Forged ARP reply to host A.

Fig. 1. ARP communication scenarios.

1991) to identify malicious ARP packets. They achieved 100% accuracy in their experimental setup with no false positives or negatives. However, this approach depends on assumptions that may not hold in real homes, including a permanent majority of trustworthy devices and an attacker's inability to compromise multiple nodes to skew the vote.

Patrice et al. (2024) proposed a promising but SDN-based ARP spoofing detection method using a dual ARP cache and distributed controllers. Similarly, Girdler and Vassilakis (2021) developed an SDN-based ARP spoofing detection system, using a POX controller to verify Ethernet frame and ARP payload consistency. It achieved 100% detection with no false positives but had slow response times. Both approaches are impractical for home gateways and decentralized smart home networks though, as they require a full SDN setup with centralized, external SDN controllers and enterprise-grade equipment.

In summary, conventional ARP spoofing detection methods for smart homes generally face limitations, including: (1) limited scalability with diverse IoT devices, (2) reliance on custom/ proprietary software installations on smart home devices, which is not always possible, (3) dependence on SNMP or SDN frameworks, and (4) limited performance in dynamic, resource-constrained environments.

2.3. Applying ML to ARP spoofing detection

ML-based techniques have gained attention for ARP spoofing detection, particularly in SDN environments. Ma et al. (2016) used Mininet to generate an ARP spoofing dataset tailored for SDN environments, achieving up to 80% accuracy with a Bayes-based method. Further advancements were made by Sebbar et al. (2020), who applied ML techniques in SDN-based networks, proposing a random forest-based model (Rahman et al., 2019; Shome et al., 2020) that achieved 97.5% accuracy. Ahuja et al. (2022) presented a hybrid CNN-LSTM model that reached 99.73% accuracy, while Hnamte and Hussain (2024) developed a deep neural network model achieving 100% accuracy. However, these approaches are limited to SDN environments, which are uncommon in smart homes, restricting their applicability.

ML-based approaches have also been investigated for environments without SDN. Hsiao et al. (2009) collected SNMP traffic data from a simulated network and tested it using Naive Bayes, Decision Tree, and SVM classifiers, with SVM achieving the highest accuracy at 80%. However, this method is less applicable to smart homes, where many devices lack SNMP compatibility.

Abdulla et al. (2020) employed neural networks and ARIMA models, achieving 90% accuracy, while Alani et al. (2023) introduced the ARP-PROBE model, which analyzed 21 network features and reached 99.98% accuracy. In our previous work (Rahman et al., 2024b), we conducted a study on how well existing ML models perform the task of ARP Spoofing detection, given various smart home testbed datasets in existence today. We used four distinct public datasets that included a diverse range of devices. By analyzing the PCAP files from these datasets, we extracted features and applied seven different ML algorithms. The results indicated that the XGBoost (Chen and Guestrin, 2016) algorithm achieved the highest accuracy, reaching 96.7%. Furthermore, a feature importance analysis revealed that only 25 out of 86 features were necessary to attain this level of accuracy.

Despite their promising performance, these ML-based methods often rely on specific datasets and may not generalize well across different smart home setups. This issue is explored in greater depth in the following section.

3. The problem of generalizing ML-based methods across smart homes

This section investigates the generalization challenges of ML approaches for ARP spoofing detection in heterogeneous smart home networks. Section 3.1 examines the cross-dataset performance issues, Section 3.2 details our experimental methodology, and Section 3.3 presents the results demonstrating the scalability limitations.

3.1. Cross-dataset performance of current ML models for smart home management

BSPs typically manage millions of client networks, many of them smart home networks. ML could, in principle, be of great assistance if not for a significant scalability issue: BSPs would need to continuously collect large amounts of data from each individual client network and continuously train and test models for each individual client network. This is caused by the large heterogeneity in smart home networks, i.e., individual networks often differ a lot from each other. This has been evidenced by various recent studies.

Booij et al. (2022), for instance, demonstrated that training on one IoT dataset (e.g. TON_IoT) does not achieve high accuracy when tested on a different IoT dataset (e.g. Aposemat IoT-23 Garcia et al., 2020), with classification accuracy dropping to 20%–30% for various attacks. This is despite achieving 99.99% accuracy when training and testing on the same dataset. The results indicate that these datasets are best suited for only their own respective setups and devices, and

face significant performance drops when applied to different contexts. The authors attributed this to collection setups not being benchmarked or standardized. In Rahman et al. (2025a), we consequently examined various existing smart home network traffic data models and found that they suffer from the same issue.

In the remainder of this section 3, we therefore investigate how well the current ML-based ARP-spoofing detection techniques can be applied to smart home networks in general. For that, we analyze the performance of existing ML-based approaches for ARP spoofing attack detection across different client networks, i.e. with various setups and devices.

3.2. Dataset selection and methodology for cross-dataset testing of ML-based ARP spoofing detection

To conduct this experiment, we chose our ARP Spoofing Based MITM Attack dataset (Rahman et al., 2024a,b), which was designed by extracting features from four public datasets, e.g., CIC IoT (Neto et al., 2023), UQ IoT IDS (Chong et al., 2021), IoT Network Intrusion (Kang et al., 2019), and ARP PCAP Files (Researcher111, 2023). Each of these datasets has a variety of smart home devices with different device types. After we developed this dataset, we also employed different ML algorithms to evaluate their performance in detecting ARP spoofing-based MITM attacks. Among these, the XGBoost algorithm demonstrated the highest accuracy (Rahman et al., 2024b). The features of the dataset are listed in Table A.4 (see appendix).

In Rahman et al. (2024b), XGBoost was configured as follows: a learning rate of 0.1, 200 estimators, a maximum depth of 8, a subsample ratio of 0.8, column sampling by tree at 0.8, $\gamma = 1$, and L2 regularization (*reg lambda* = 1). The best features were selected as follows: bidirectional_bytes, application_name, src2dst_syn_packets, dst2src_bytes, bidirectional_syn_packets, application_is_guessed, src2dst _mean_ps, dst2src_min _piat_ms, requested_server_name, src2dst_bytes, src2dst_max_piat_ms, application_confidence, src2dst_min_ps, ip_version, src2dst_psh_packets, src2dst _packets, bidirectional_cwr_packets, dst2src_max_piat_ms, bidirectional_min _ps, src2dst_max_ps, protocol, bidirectional_last_seen_ms, bidirectional_max_ps, src_port, and dst_port. We chose XGBoost with this configuration and these features to analyze the performance of training on one dataset and testing on others:

The performance of the ML algorithms was measured using several evaluation metrics, including accuracy, precision, recall or True Positive Rate (TPR), and F1 score (Rahman et al., 2024c,d,b).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

$$\text{Recall/TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

With TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives. These metrics provide a robust framework for evaluating the performance of ML algorithms in classifying smart home devices. Accuracy serves as a general measure of the classifier's overall correctness, indicating the proportion of correctly classified instances out of the total. However, accuracy alone can be misleading, particularly in scenarios with imbalanced class distributions, as it may disproportionately reflect the performance on the majority class while overlooking the minority classes (Rahman et al., 2025b).

To address this limitation, precision and recall are included to offer more granular insights into the classifier's performance. Precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive, effectively minimizing false positives. This is particularly important in smart home environments, where false positives (e.g., misclassifying a benign device as malicious) can lead to unnecessary disruptions. On the other hand, recall evaluates the classifier's ability to identify all actual positive instances within a given class, thereby reducing false negatives. In the context of smart homes, false negatives (e.g., failing to detect a malicious device) can have severe security implications.

The F1 score harmonizes precision and recall into a single metric, providing a balanced measure that is especially useful when there is a significant trade-off between the two. This is critical in smart home environments, where both over-classification (false positives) and under-classification (false negatives) can lead to functional inefficiencies or security vulnerabilities (Rahman et al., 2024c,d,b).

3.3. Results

To simulate typical BSP scenarios, we trained the XGBoost model on one dataset and evaluated its performance on the three other datasets. We repeated that procedure for every dataset. Our findings are shown in Table 1. The results indicate that each dataset achieves its highest performance when tested on itself. For example, the CIC IoT dataset attained an accuracy, precision, recall, and F1 score of 0.955 when evaluated on itself. However, its performance dropped significantly when tested on other datasets, with the lowest accuracy observed on the ARP PCAP Files dataset (0.6463) and a corresponding F1 score of 0.5203. This scenario happened for each dataset and indicates the difficulty of generalizing models across diverse datasets.

This result aligns with the findings in Rahman et al. (2025b) and Booij et al. (2022). And in contrast to Booij et al. (2022), all datasets in our study were collected using the same method, and we used a single tool to extract features from the PCAP files. Our classification focused solely on distinguishing between benign traffic and ARP spoofing attacks. And, yet, training on one dataset and testing on others still resulted in poor performance. This indicates that simply developing a large-scale dataset for training and testing across different BSP client networks is unlikely to be effective, as each network is unique and exhibits different characteristics.

4. ARProof: A novel cross-protocol approach

This section introduces ARProof, our novel algorithm designed to detect and mitigate ARP spoofing attacks in smart home networks. In Section 4.1, we detail the algorithm's design. Section 4.2 presents the experimental setup, evaluation results, and a comparative analysis with existing methods. Lastly, Section 4.3 highlights the limitations of this approach.

4.1. The ARProof algorithm design

We have developed an algorithm that utilizes ARP and Dynamic Host Configuration Protocol (DHCP) packets to dynamically maintain a real-time IP–MAC address mapping. This mapping serves as the foundation for detecting and mitigating ARP spoofing attacks in real-world network environments. In addition, the algorithm verifies MAC–IP consistency on all IPv4 packets (TCP, UDP, and ICMP) observed at the gateway.

ARP operates on a trust-based mechanism, lacking authentication, which makes it susceptible to spoofing attacks. To mitigate this vulnerability, our algorithm integrates DHCP as an independent validation source. DHCP, which automatically assigns IP addresses to network devices, provides authoritative mappings via DHCPACK messages that include both assigned IP and MAC addresses. By leveraging this data, our algorithm verifies ARP responses and detects inconsistencies indicative of spoofing activity. The IPv4 check closes the static-ARP bypass because forged packets that avoid ARP are still compared against the DHCP-anchored binding.

Table 1
XGBoost performance metrics for ARP-spoofing detection, by training one dataset and testing on others.

Training dataset	Metric	Testing datasets			
		CIC IoT	UQ IoT IDS	IoT network intrusion	ARP PCAP files
CIC IoT	Accuracy	0.955	0.7892	0.8848	0.6463
	Precision	0.955	0.9158	0.9082	0.4353
	Recall	0.955	0.7892	0.8848	0.6463
	F1 Score	0.955	0.8400	0.8962	0.5203
UQ IoT IDS	Accuracy	0.5027	0.940	0.9511	0.6626
	Precision	0.7500	0.949	0.9046	0.4390
	Recall	0.5027	0.940	0.9511	0.6626
	F1 Score	0.3363	0.944	0.9273	0.5281
IoT network intrusion	Accuracy	0.5026	0.9422	0.931	0.6626
	Precision	0.2526	0.8878	0.934	0.4390
	Recall	0.5026	0.9422	0.931	0.6626
	F1 Score	0.3363	0.9142	0.932	0.5281
ARP PCAP files	Accuracy	0.4839	0.9385	0.9511	0.980
	Precision	0.3059	0.8876	0.9046	0.981
	Recall	0.4839	0.9385	0.9511	0.980
	F1 Score	0.3326	0.9123	0.9273	0.980

Our algorithm continuously monitors ARP and DHCP traffic, updating the IP–MAC mapping dynamically. By cross-referencing ARP and DHCP mappings, the algorithm detects discrepancies indicative of ARP spoofing. Any inconsistencies between ARP and DHCP records trigger detection mechanisms, enabling timely threat identification and mitigation. By integrating ARP and DHCP validation, our approach enhances ARP spoofing detection effectiveness, reducing false positives and improving reliability. To balance latency and precision, we use a short evidence window. If the same MAC address asserts an IP address that DHCP already bound to another device at least θ times within Δt, we mark it as confirmed.

Algorithm 1 presents our proposed method, while Fig. 2 illustrates the flowchart for real-time detection of both ARP-based and ARP-less spoofing. For each packet observed in the network, the algorithm first checks whether it is a DHCP ACK packet. If so, it extracts the source MAC address and the newly assigned IP address, subsequently updating the device list with this information and labeling the IP assignment as DHCP-based. For each valid ARP packet, the algorithm proceeds by extracting the source IP and MAC addresses and querying the device list for entries corresponding to the extracted IP address. If a matching entry is found, the algorithm compares the MAC address in the packet with the stored MAC address. A discrepancy between the two indicates an overlapping IP claim. In such cases, if the IP address was previously assigned as static, the packet is classified as a suspected ARP spoofing attempt. Alternatively, if the MAC address is known in the device list but has not been associated with the claimed IP, the packet is flagged as a confirmed ARP spoofing attack. Conversely, if the IP and MAC addresses match the known record, the packet is classified as benign. If no matching IP is found in the device list, the packet is also treated as benign by default. For every IPv4 packet, we compare the source MAC–IP tuple with the table. If the source IP is DHCP-bound to a different MAC, we increase a per-tuple counter. Once it meets $(\theta, \Delta t)$, the attack is confirmed. The gateway applies layer-2 and, if needed, layer-3 drop rules for the offending MAC. It can also revoke the DHCP lease and flush conflicting ARP cache entries to restore correct forwarding.

4.2. Evaluation of the proposed algorithm

In this section, we evaluate the performance of our proposed algorithm for detecting and preventing ARP spoofing attacks. We perform this evaluation in two phases: (1) offline analysis using four public datasets listed in Section 3 and (2) real-time testing using two smart home testbeds.

In both phases, the same algorithm is used to validate IP-MAC mappings by correlating ARP and DHCP traffic, enabling cross-protocol verification. In the offline phase, the algorithm was applied to captured network traces from four datasets. It achieved 100% detection accuracy across all scenarios, as summarized in Table 2. For real-time testing, we implemented two testbeds featuring a variety of smart home devices:

- `Lab-1 testbed`: This testbed includes 10 IoT devices and 2 non-IoT devices, all connected to a central gateway. The devices are categorized into seven functional types: audio, camera, hub, lighting, PC, router, and video doorbell. IoT devices using Zigbee connectivity, such as a bulb, motion sensor, and switch, communicate with the central gateway through a Philips Hue Zigbee hub. A Raspberry Pi is connected to the central gateway via a LAN connection for real-time traffic analysis. The remaining devices are connected to the gateway via Wi-Fi. The topology of the Lab-1 testbed setup is shown in Fig. 3(a).
- `Lab-2 testbed`: This testbed consists of 6 IoT devices and 1 non-IoT device, all connected to a central gateway. The devices are grouped into six functional types: audio, camera, lighting, PC, router, and video doorbell. A Raspberry Pi is connected to the router via a LAN connection for real-time traffic analysis, while the remaining devices connect to the router through Wi-Fi. Fig. 3(b) illustrates the topology of the Lab-2 testbed setup.

We implemented the proposed algorithm in Python for deployment on Raspberry Pi devices to provide real-time detection of both ARP-based and ARP-less IP forgery. The Raspberry Pi 4 units in the testbed run Raspberry Pi OS and are equipped with a 64-bit quad-core ARM Cortex-A72 processor and 8 GB of RAM. In the Python implementation, we captured packets in real time using Scapy (Rohith et al., 2018) and applied the detection logic to the extracted link-layer and network-layer features. The algorithm was configured with parameters $\theta = 3$ and $\Delta t = 3$ s for all experiments. The experimental campaign ran for 20 days in total. During testing, the devices remained in normal operation.

To simulate ARP-based attacks, we used a Kali Linux laptop running Bettercap (Maaz et al., 2023). For each testbed, we selected ten IP addresses at random (both IoT and non-IoT devices) and repeated the selection ten times, producing 100 distinct ARP-spoofing scenarios. ARP-based packets were injected at random intervals. In all instances, our method successfully identified the ARP spoofing attacks in real-time.

For ARP-less experiments, we generated forged IPv4 traffic from the Kali host using custom frame-crafting scripts implemented with Scapy. We executed ten distinct ARP-less scenarios per testbed. In each scenario, the attacker transmitted 500 forged packets distributed over five randomly chosen destination IPs (100 packets per destination), yielding 5000 forged packets in total. Packets within a scenario were

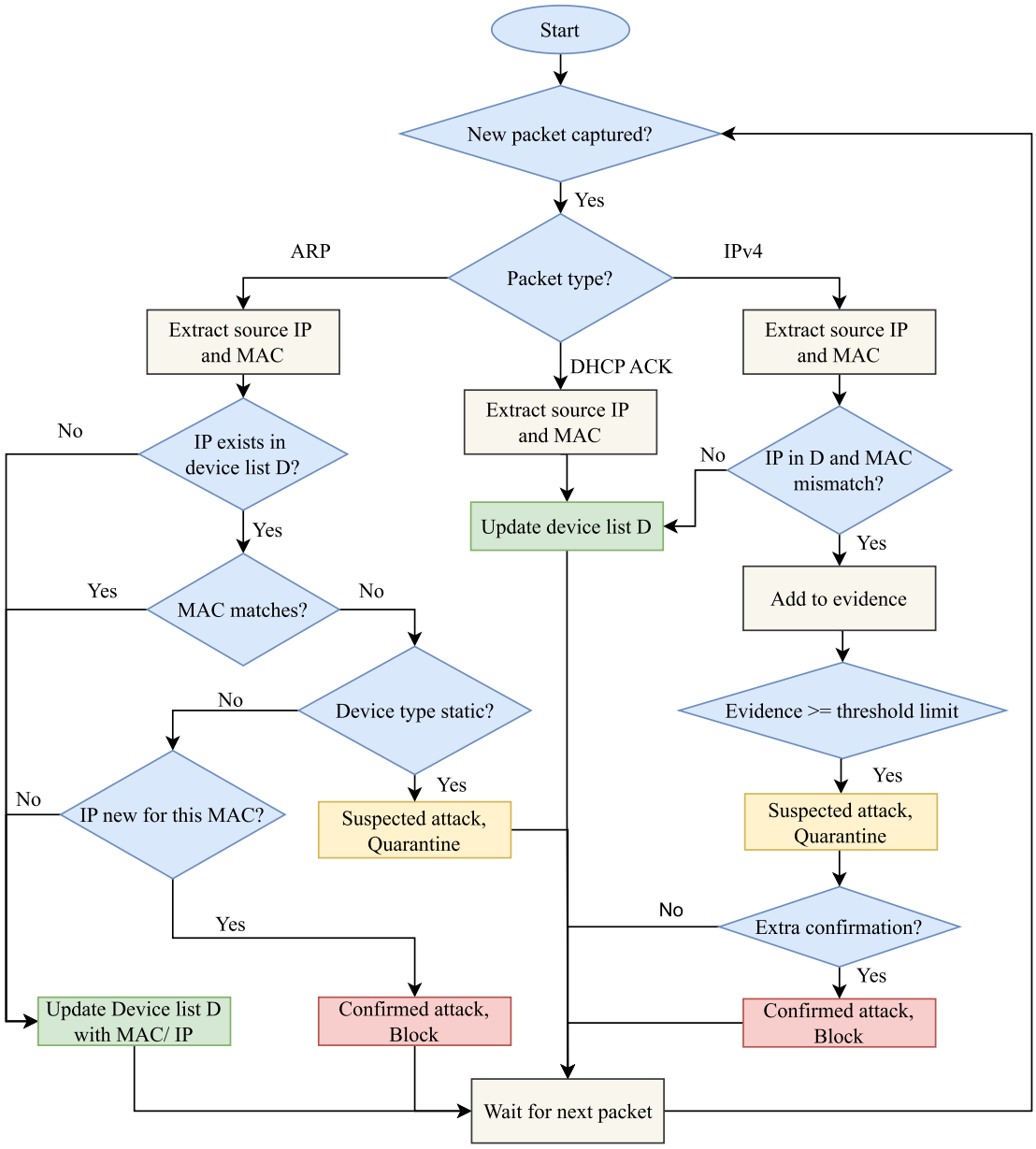


Fig. 2. Flowchart of the ARProof algorithm.

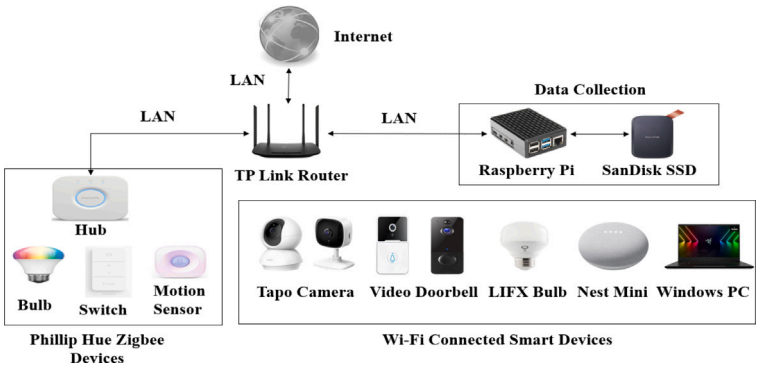
Table 2
Performance of our proposed approach on different public datasets.

Dataset	Total packets	Benign ARP packets	ARP-spoofing packets	Accuracy
CIC IoT	649,089	47,142	8,865	100%
UQ IoT IDS	15,302	720	538	100%
IoT network intrusion	194,184	281	788	100%
ARP PCAP files	16,285	24	26	100%

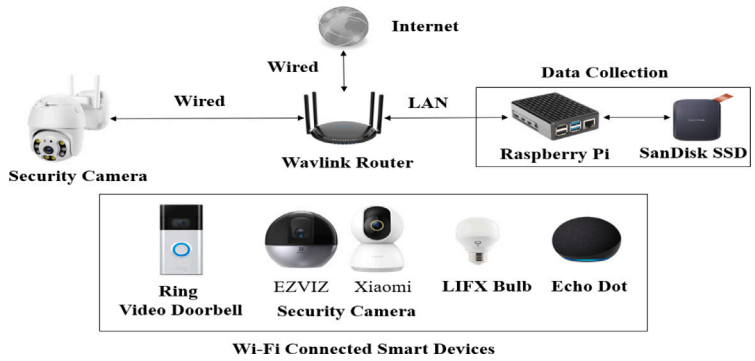
emitted at roughly 0.5-second intervals. Attack parameters (packet size, destination, and per-round timing) were varied across scenarios to exercise detector behavior under diverse conditions. The algorithm processed frames in real-time and raised alerts for every forged packet observed.

We validated the algorithm using static ARP configurations by adding static IP-MAC entries to the gateways and updating those bindings in the detector. We then sent forged IPv4 packets using Scapy and attempted ARP spoofing with Bettercap, as in our previous tests. The algorithm successfully flagged both ARP-based and ARP-less spoofing in real-time, achieving 100% accuracy.

The primary distinction between ML-based methods and our approach is the timing of attack detection. In ML-based approaches, attacks are detected only after they occur, whereas our model identifies and blocks the attacker during the initial spoofing attempt, preventing the attack from happening. Besides, our algorithm overcomes the limitations of existing non-ML methods and the comparison is shown in Table 3. Here, we evaluate ARProof against other methods based



(a) Lab-1 Testbed Topology



(b) Lab-2 Testbed Topology

Fig. 3. Topology of the lab testbeds.

Table 3
Comparison of models with respect to key characteristics.

Model	Utilize ICMP	Utilize DHCP	Gateway based	Smart home oriented	Dynamic	Scalable	Customized software required	Requires active probing/injects extra traffic
Hijazi and Obaidat (2018)	✓	×	✓	✓	×	×	×	×
Arote and Arya (2015)	✓	×	✓	×	✓	×	×	✓
AbdelSalam et al. (2014)	✓	✓	×	×	✓	×	✓	✓
Morsy and Nashat (2022)	✓	×	×	×	✓	×	✓	✓
Morsy and Nashat (2025)	×	×	×	×	✓	×	✓	×
Patrice et al. (2024)	×	×	×	×	✓	✓	✓	×
Girdler and Vassilakis (2021)	×	×	×	×	✓	×	✓	×
ARProof	✓	✓	✓	✓	✓	✓	×	×

on eight key criteria. ARProof is the only method that satisfies all deployment-oriented criteria: it runs at the gateway, is designed for smart homes, adapts to device and address changes, is scalable, uses both DHCP and ICMP, and does not require endpoint software or injecting extra traffic.

Furthermore, we analyzed the CPU and memory usage of the algorithm on the Raspberry Pi. The algorithm is highly resource efficient, requiring only 42.28 MB of memory and 0.8% of CPU resources on the Raspberry Pi. These findings confirm the suitability of the algorithm for deployment in resource-constrained environments, such as smart home networks.

Our algorithm automatically adapts to any network configuration, ensuring 100% accuracy. BSPs can readily implement this approach in the routers or gateways they provide.

Our algorithm can be implemented as a lightweight agent on a home gateway or router. It continuously monitors network packets in real-time, detecting both ARP-based and ARP-less spoofing attempts. When a spoofing event occurs, it is displayed in the gateway control panel for smart home users and local administrators. Each alert includes clear evidence, such as timestamps and event counts. Users have the option to add devices to a temporary exception list while conducting experiments. Each exception is documented with a reason and an expiration date. Additionally, containment actions, like rate limiting or quarantining, are also time-limited and reversible. All alerts, exceptions, and enforcement actions are logged for auditing and post-incident analysis.

4.3. Limitations

Although the algorithm achieved 100% accuracy in our lab runs, it has the following limitations:

1. The reported result reflects the specific datasets, testbeds, and attacker models used in this study. Broader evaluation is required to establish statistical confidence and operational limits:

Algorithm 1 ARProof: ARP-spoofing attack detection and mitigation

Input: P: Captured network packets; D: Device information with attributes: IP, MAC, Type, DHCP
Output: ARP spoofing detection results and updated device list D
Step 1: Initialize counters and parameters
Set B = 0 (Benign packets), S = 0 (Suspected attacks), C = 0 (Confirmed attacks), O = 0 (Overlapping IP claims)
Choose confirmation threshold θ (e.g., 3 packets) and time window Δt (e.g., 3s)
Initialize an empty mismatch evidence map $E[(\text{MAC}, \text{IP})] \leftarrow 0$
for each packet p in P **do**
  if p is a DHCP ACK packet **then**
    Extract the source MAC address from p
    Extract the assigned IP address from p
    Update device list D with this IP–MAC mapping and mark as "dhcp"
  end if
  if p is a valid ARP packet **then**
    Extract the source IP and source MAC address from p
    Retrieve all entries in D that match the source IP
    if matching entries exist **then**
      for each matching entry e in D **do**
        if the MAC address in e does not match the source MAC **then**
          Increment overlap counter O
          if the DHCP type in e is "static" **then**
            Increment suspected spoofing counter S, quarantine device
          else if the source MAC exists in D but the source IP is new for this MAC **then**
            Increment confirmed spoofing counter C, block that device
          end if
        end if
      end for
    end if
    Ensure D has an entry for source MAC; add source IP to its IP set if missing
  end if
  if p is an IPv4 packet (TCP/UDP/ICMP) **then**
    Extract source MAC and source IP
    Look up e_{mac} = D[source MAC], and e_{ip} = D[source IP]
    if e_{ip} exists and e_{ip}.MAC $\neq$ source MAC **then**
      if e_{ip} is authoritative (DHCP or reserved static, not stale) **then**
        Add timestamp to evidence queue $E[(\text{source MAC}, \text{source IP})]$ and prune entries older than now $-\Delta t$
        if evidence count $\geq \theta$ within Δt **then**
          Increment suspected spoofing counter S, quarantine device
          if extra confirmation (e.g., ARP mismatch, DHCP conflict, admin confirm) **then**
            Increment confirmed spoofing counter C, block device
          end if
        end if
      else
        Update D with (source MAC, source IP)
      end if
    else
      Update D with (source MAC, source IP)
    end if
  end if
end for

this includes additional public datasets, longer-duration traces that capture diurnal and weekly variation, sensitivity analyses under noisy and adversarial conditions, and ISP/BSP-scale pilot deployments with reported confidence intervals.
2. Validation on Raspberry Pi hardware does not fully reflect the diversity of consumer gateways or carrier-grade equipment. Therefore, practical deployment necessitates testing on representative gateway hardware, considering various operating systems.
3. The algorithm relies on up-to-date records of IP-MAC bindings. It will not be effective against attacks where an adversary forges both the IP and MAC addresses.
4. The confirmation threshold ($\theta = 3$) and evidence time window ($\Delta t = 3$ s) worked well in our test environments but may lead to more false positives in noisy networks.

5. Conclusion and future work

Despite the growing need for robust security measures, state-of-the-art ARP spoofing detection techniques face notable challenges when applied to smart home networks. A key issue with current ML-based methods is their reliance on datasets that are typically sourced from single-lab testbeds simulating a single home environment. These methods are often presumed to have universal applicability to all smart homes; however, our analysis demonstrated that this assumption is not valid. ML performance varies significantly between datasets and algorithms, and only a few features substantially impact the ability to distinguish between normal and attack instances. Furthermore, our findings indicate that training on one dataset and testing on another leads to poor performance. As a result, BSPs will need to collect and store data for each client network, and subsequently train individual models for each network. This demonstrates that current ML approaches lack scalability and cost efficiency for real-world ARP spoofing detection and mitigation.

To address these limitations, we proposed a novel and efficient algorithm for real-time ARP spoofing attack detection and mitigation, specifically designed for resource-constrained environments such as smart home networks. Unlike existing ML and non-ML methods, our approach introduces a cross-protocol design that leverages both ARP and DHCP traffic for validating IP-MAC bindings. This cross-verification mechanism enhances detection reliability, reduces false positives, and enables proactive mitigation of spoofing attempts before they can impact network performance. We validated our algorithm using four public datasets and two real-world testbeds, achieving 100% accuracy in all scenarios. The algorithm also demonstrated minimal memory and CPU usage, confirming its efficiency and suitability for deployment in smart home gateways and BSP-managed networks.

Future work will focus on deploying the proposed algorithm within commercial routers and broadband gateways used by BSPs to evaluate its practicality in real-world smart home environments. We aim to collaborate with service providers and gateway vendors to conduct pilot deployments on real gateway hardware and vendor firmware, including OpenWrt builds. Additionally, we plan to test the algorithm under various network conditions and device configurations. A systematic sensitivity analysis of the algorithm's parameters (θ and Δt) will be conducted to characterize the trade-off between detection latency and false positive rates across diverse network environments.

We also plan to extend our approach to other forms of spoofing attacks, such as DHCP and DNS spoofing, and to devices in the home that use IPv6 on the network layer. The latter can be achieved by learning authoritative IPv6-to-MAC bindings from DHCPv6 or from StateLess Address AutoConfiguration (SLAAC) / router advertisement events, and then validating those bindings against neighbor discovery solicitations and advertisements. This will help establish a unified, protocol-aware security framework capable of addressing multiple network-layer threats in smart homes.

Table A.4
List of features in the datasets.

Feature	Description
id	A unique identifier assigned to each flow
expiration_id	Identifier marking the expiration of a flow
src_ip	IP address of the source
src_mac	MAC address of the source
src_oui	Organizationally Unique Identifier (OUI) of the source
src_port	Port number used by the source
dst_ip	IP address of the destination
dst_mac	MAC address of the destination
dst_oui	OUI of the destination
dst_port	Port number used by the destination
protocol	Protocol used in the network flow
ip_version	Internet Protocol version (IPv4/IPv6)
vlan_id	Identifier for the VLAN
tunnel_id	Identifier for the network tunnel
bidirectional_first_seen_ms	Timestamp of the first packet in bidirectional flow
bidirectional_last_seen_ms	Timestamp of the last packet in bidirectional flow
bidirectional_duration_ms	Total duration of the bidirectional flow
bidirectional_packets	Count of packets in the bidirectional flow
bidirectional_bytes	Total bytes transferred in bidirectional flow
src2dst_first_seen_ms	Timestamp of the first packet from source to destination
src2dst_last_seen_ms	Timestamp of the last packet from source to destination
src2dst_duration_ms	Duration of data transfer from source to destination
src2dst_packets	Packet count from source to destination
src2dst_bytes	Byte count from source to destination
dst2src_first_seen_ms	Timestamp of the first packet from destination to source
dst2src_last_seen_ms	Timestamp of the last packet from destination to source
dst2src_duration_ms	Duration of data transfer from destination to source
dst2src_packets	Packet count from destination to source
dst2src_bytes	Byte count from destination to source
bidirectional_min_ps	Smallest packet size in bidirectional flow
bidirectional_mean_ps	Average packet size in bidirectional flow
bidirectional_stddev_ps	Standard deviation of packet sizes in bidirectional flow
bidirectional_max_ps	Largest packet size in bidirectional flow
src2dst_min_ps	Smallest packet size from source to destination
src2dst_mean_ps	Average packet size from source to destination
src2dst_stddev_ps	Standard deviation of packet sizes from source to destination
src2dst_max_ps	Largest packet size from source to destination
dst2src_min_ps	Smallest packet size from destination to source
dst2src_mean_ps	Average packet size from destination to source
dst2src_stddev_ps	Standard deviation of packet sizes from destination to source
dst2src_max_ps	Largest packet size from destination to source
bidirectional_min_piat_ms	Smallest inter-arrival time (ms) in bidirectional flow
bidirectional_mean_piat_ms	Average inter-arrival time (ms) in bidirectional flow
bidirectional_stddev_piat_ms	Standard deviation of inter-arrival times (ms) in bidirectional flow
bidirectional_max_piat_ms	Largest inter-arrival time (ms) in bidirectional flow
src2dst_min_piat_ms	Smallest inter-arrival time (ms) from source to destination
src2dst_mean_piat_ms	Average inter-arrival time (ms) from source to destination
src2dst_stddev_piat_ms	Standard deviation of inter-arrival times (ms) from source to destination
src2dst_max_piat_ms	Largest inter-arrival time (ms) from source to destination
dst2src_min_piat_ms	Smallest inter-arrival time (ms) from destination to source
dst2src_mean_piat_ms	Average inter-arrival time (ms) from destination to source
dst2src_stddev_piat_ms	Standard deviation of inter-arrival times (ms) from destination to source
dst2src_max_piat_ms	Largest inter-arrival time (ms) from destination to source
bidirectional_syn_packets	Total SYN packets in bidirectional flow
bidirectional_cwr_packets	Total CWR packets in bidirectional flow
bidirectional_ece_packets	Total ECE packets in bidirectional flow
bidirectional_urg_packets	Total URG packets in bidirectional flow
bidirectional_ack_packets	Total ACK packets in bidirectional flow
bidirectional_psh_packets	Total PSH packets in bidirectional flow
bidirectional_rst_packets	Total RST packets in bidirectional flow
bidirectional_fin_packets	Total FIN packets in bidirectional flow
src2dst_syn_packets	SYN packets sent from source to destination
src2dst_cwr_packets	CWR packets sent from source to destination
src2dst_ece_packets	ECE packets sent from source to destination
src2dst_urg_packets	URG packets sent from source to destination
src2dst_ack_packets	ACK packets sent from source to destination
src2dst_psh_packets	PSH packets sent from source to destination
src2dst_rst_packets	RST packets sent from source to destination
src2dst_fin_packets	FIN packets sent from source to destination
dst2src_syn_packets	SYN packets sent from destination to source
dst2src_cwr_packets	CWR packets sent from destination to source
dst2src_ece_packets	ECE packets sent from destination to source
dst2src_urg_packets	URG packets sent from destination to source
dst2src_ack_packets	ACK packets sent from destination to source
dst2src_psh_packets	PSH packets sent from destination to source
dst2src_rst_packets	RST packets sent from destination to source
dst2src_fin_packets	FIN packets sent from destination to source
application_name	Traffic-generating application name
application_category_name	Category of the application
application_is_guessed	Indicator if the application is guessed
application_confidence	Confidence level in the application's classification
requested_server_name	Server name requested by the client
client_fingerprint	TLS fingerprint of the client
server_fingerprint	TLS fingerprint of the server
user_agent	HTTP header's user agent string
content_type	Content type specified in HTTP requests
Label	Classification of flow as normal or an attack

CRediT authorship contribution statement

Md Mizanur Rahman: Writing – original draft, Visualization, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Faycal Bouhafs:** Writing – review & editing, Validation, Supervision. **Sayed Amir Hoseini:** Writing – review & editing, Validation, Supervision. **Frank den Hartog:** Writing – review & editing, Validation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. List of features in the ARP spoofing-based MITM attack dataset

See Table A.4.

Data availability

Dataset is already published online and available for research purpose.

References

AbdelSalam, A.M., El-Sisi, A.B., Reddy, V., 2015. Mitigating ARP spoofing attacks in software-defined networks. In: 2015 25th International Conference on Computer Theory and Applications. ICCTA, IEEE, pp. 126–131.

AbdelSalam, A.M., Elkilani, W.S., Amin, K.M., 2014. An automated approach for preventing ARP spoofing attack using static ARP entries. Int. J. Adv. Comput. Sci. Appl. 5 (1).

Abdulla, H., Al-Raweshidy, H., Awad, W.S., 2020. ARP spoofing detection for IoT networks using neural networks. In: Proceedings of the Industrial Revolution & Business Management: 11th Annual PwR Doctoral Symposium. PWRDS.

Aboubakar, M., Kellil, M., Roux, P., 2022. A review of IoT network management: Current status and perspectives. J. King Saud Univ. - Comput. Inf. Sci. 34 (7), 4163–4176.

Ahmad, S., Mir, A.H., 2021. Scalability, consistency, reliability and security in SDN controllers: A survey of diverse SDN controllers. J. Netw. Syst. Manage. 29 (1), 9.

Ahuja, N., Singal, G., Mukhopadhyay, D., Nehra, A., 2022. Ascertain the efficient machine learning approach to detect different ARP attacks. Comput. Electr. Eng. 99, 107757.

Alani, M.M., Alani, M.M., 2014. TCP/IP model. In: Guide to OSI and TCP/IP models. pp. 19–50.

Alani, M.M., Awad, A.I., Barka, E., 2023. ARP-PROBE: An ARP spoofing detector for internet of things networks using explainable deep learning. Int. Things 23, 100861.

Anon, 2025. Broadband forum. https://www.broadband-forum.org/, (Accessed: 2025-04-23).

Anthi, E., Williams, L., Javed, A., Burnap, P., 2021. Hardening machine learning denial of service (DoS) defences against adversarial attacks in IoT smart home networks. Comput. Secur. 108, 102352.

Arote, P., Arya, K.V., 2015. Detection and prevention against ARP poisoning attack using modified ICMP and voting. In: 2015 International Conference on Computational Intelligence and Networks. IEEE, pp. 136–141.

Bhirud, S., Katkar, V., 2011. Light weight approach for IP-ARP spoofing detection and prevention. In: 2011 Second Asian Himalayas International Conference on Internet (AH-ICI). IEEE, pp. 1–5.

Bhushan, B., Sahoo, G., Rai, A.K., 2017. Man-in-the-middle attack in wireless and computer networking—A review. In: 2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA)(Fall). IEEE, pp. 1–6.

Blinder, C., 2023. Average number of smart devices in a home. URL https://www.consumeraffairs.com/homeowners/average-number-of-smart-devices-in-a-home.html, (Accessed: 2025-01-28).

Booij, T.M., Chiscop, I., Meeuwissen, E., Moustafa, N., Hartog, F.T.H.d., 2022. ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets. IEEE Int. Things J. 9 (1), 485–496.

Boyer, R.S., Moore, J.S., 1991. MJRTY—A fast majority vote algorithm. In: Automated Reasoning: Essays in Honor of Woody Bledsoe. Springer, pp. 105–117.

Broadband-Forum, 2012. Layer 2 control mechanism for broadband multi-service architectures (TR-207). URL https://www.broadband-forum.org/pdfs/tr-207-1-0-0.pdf, (Accessed: February 07, 2025).

Broadband-Forum, 2013. Subscriber sessions (TR-146). URL https://www.broadband-forum.org/pdfs/tr-146-1-0-0.pdf, (Accessed: February 07, 2025).

Broadband-Forum, 2022. Functional requirements for broadband residential gateway devices (TR-124). URL https://www.broadband-forum.org/pdfs/tr-124-7-0-0.pdf, (Accessed: February 07, 2025).

Chen, T., Guestrin, C., 2016. XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794.

Chen, X., Wu, C., Liu, X., Huang, Q., Zhang, D., Zhou, H., Yang, Q., Khan, M.K., 2023. Empowering network security with programmable switches: A comprehensive survey. IEEE Commun. Surv. & Tutorials 25 (3), 1653–1704.

Chong, E., Ma, J., Lwin, K.T., 2021. UQ IoT-IDS 2021. http://dx.doi.org/10.48610/17b44bb, https://espace.library.uq.edu.au/view/UQ:17b44bb, (Accessed: July 18, 2024).

Conti, M., Dragoni, N., Lesyk, V., 2016. A survey of man in the middle attacks. IEEE Commun. Surv. & Tutorials 18 (3), 2027–2051.

David, C., 1982. An ethernet address resolution protocol. RFC826.

Frankel, S., Eydt, B., Owens, L., Scarfone, K., 2007. Establishing wireless robust security networks: A guide to IEEE 802.11 I. NIST Spec. Publ. 800–897.

Garcia, S., Parmisano, A., Erquiaga, M.J., 2020. IoT-23: A labeled dataset with malicious and benign IoT network traffic. (Accessed: 2025-09-27).

Girdler, T., Vassilakis, V.G., 2021. Implementing an intrusion detection and prevention system using software-defined networking: Defending against ARP spoofing attacks and blacklisted MAC addresses. Comput. Electr. Eng. 90, 106990.

Hijazi, S., Obaidat, M.S., 2018. A new detection and prevention system for ARP attacks using static entry. IEEE Syst. J. 13 (3), 2732–2738.

Hijazi, S., Obaidat, M.S., 2019. Address resolution protocol spoofing attacks and security approaches: A survey. Secur. Priv. 2 (1), e49.

Hnamte, V., Hussain, J., 2024. Enhancing security in software-defined networks: An approach to efficient ARP spoofing attacks detection and mitigation. Telemat. Inform. Rep. 14, 100129.

Hsiao, H.-W., Lin, C.S., Chang, S.-Y., 2009. Constructing an ARP attack detection system with SNMP traffic data mining. In: Proceedings of the 11th International Conference on Electronic Commerce. pp. 341–345.

Kang, H., Ahn, D.H., Lee, G.M., Yoo, J.D., Park, K.H., Kim, H.K., 2019. IoT network intrusion dataset. http://dx.doi.org/10.21227/q70p-q449.

Ma, H., Ding, H., Yang, Y., Mi, Z., Yang, J.Y., Xiong, Z., 2016. Bayes-based ARP attack detection algorithm for cloud centers. Tsinghua Sci. Technol. 21 (1), 17–28.

Maaz, S., Sinha, D.K., Sinha, G., 2023. Examination of different network security monitoring tools. In: Mobile Computing and Sustainable Informatics: Proceedings of ICMCSI 2023. Springer, pp. 653–666.

Mandal, S., Khan, D.A., Jain, S., 2021. Cloud-based zero trust access control policy: An approach to support work-from-home driven by COVID-19 pandemic. New Gener. Comput. 39 (3), 599–622.

Manso, P., Moura, J., Serrão, C., 2019. SDN-based intrusion detection system for early detection and mitigation of DDoS attacks. Information 10 (3), 106.

Matoušek, P., Ryšavỳ, O., Polčák, L., 2021. Unified SNMP interface for IoT monitoring. In: 2021 IFIP/IEEE International Symposium on Integrated Network Management. IM, IEEE, pp. 938–943.

Meghana, J.S., Subashri, T., Vimal, K., 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, IEEE, pp. 1–6.

Morsy, S.M., Nashat, D., 2022. D-ARP: An efficient scheme to detect and prevent ARP spoofing. IEEE Access 10, 49142–49153.

Morsy, S., Nashat, D., 2025. VB-ARP: Boyer–Moore majority voting algorithm based defense for ARP spoofing. J. Netw. Netw. Appl. 4 (4), 172–177.

Motamary, S., 2021. Implementing infrastructure-as-code for telecom networks: Challenges and best practices for scalable service orchestration. Available at SSRN 5240118.

Neto, E.C.P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., Ghorbani, A.A., 2023. CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. Sensors 23 (13), 5941.

Palatty, N.J., 2025. How many cyber attacks per day: The latest stats and impacts in 2024. URL https://www.getastra.com/blog/security-audit/how-many-cyber-attacks-per-day/, (Accessed: February 07, 2025).

Patrice, L., Sinde, R., Leo, J., 2024. A novel mechanism for detection of address resolution protocol spoofing attacks in large-scale software-defined networks. IEEE Access 12, 80255–80265.

Rahman, M.M., Bouhafs, F., den Hartog, F., 2025a. A survey on the effectiveness of existing smart home cyber attacks detection solution: A broadband service providers' perspective. IEEE Open J. Commun. Soc. 1.

Rahman, M.M., Bouhafs, F., Hoseini, S.A., den Hartog, F., 2024a. ARP spoofing based MITM attack dataset. http://dx.doi.org/10.34740/KAGGLE/DSV/9173677, URL https://www.kaggle.com/dsv/9173677.

Rahman, M.M., Bouhafs, F., Hoseini, S.A., den Hartog, F., 2024b. Feature relevance for detecting address resolution protocol spoofing in smart homes with machine learning. In: Proceedings of the 51st International Conference on Computers and Industrial Engineering (CIE51). pp. 1–10.

Rahman, M.M., Bouhafs, F., Hoseini, S.A., den Hartog, F., 2024c. The influence of device type aggregation on the classification of smart home devices using machine learning algorithms. In: Proceedings of the 27th International Conference on Computer and Information Technology. IEEE, pp. 1–6.

Rahman, M.M., Bouhafs, F., Hoseini, S.A., den Hartog, F., 2025b. UNSW HomeNet: A network traffic flow dataset for AI-based smart home device classification. Comput. Ind. Eng. 204, 111041.

Rahman, M.M., Chayan, M.M.H., Mehrin, K., Sultana, A., Hamed, M.M., 2024d. Explainable deep learning for cyber attack detection in electric vehicle charging stations. In: Proceedings of the 11th International Conference on Networking, Systems, and Security (NSysS '24). ACM, pp. 1–7.

Rahman, M.M., Shome, A., Chellappan, S., Islam, A.A.A., 2019. How smart your smartphone is in Lie detection? In: Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. pp. 338–347.

Researcher111, 2023. ARP spoofing PCAP file. URL https://github.com/researcher111/ARP-pcap-files/blob/master/arpspoof.pcap, (Accessed: July 18, 2024).

Rohith, R., Moharir, M., Shobha, G., et al., 2018. Scapy- a powerful interactive packet manipulation program. In: 2018 International Conference on Networking, Embedded and Wireless Systems. ICNEWS, IEEE, pp. 1–5.

Sebbar, A., Zkik, K., Baddi, Y., Boulmalf, M., Kettani, M.D.E.-C.E., 2020. MitM detection and defense mechanism CBNA-RF based on machine learning for large-scale SDN context. J. Ambient. Intell. Humaniz. Comput. 11 (12), 5875–5894.

Shaikh, F., Bou-Harb, E., Crichigno, J., Ghani, N., 2018. A machine learning model for classifying unsolicited IoT devices by observing network telescopes. In: 2018 14th International Wireless Communications & Mobile Computing Conference. IWCMC, IEEE, pp. 938–943.

Shokry, M., Awad, A.I., Abd-Ellah, M.K., Khalaf, A.A., 2022. Systematic survey of advanced metering infrastructure security: Vulnerabilities, attacks, countermeasures, and future vision. Future Gener. Comput. Syst. 136, 358–377.

Shome, A., Rahman, M.M., Chellappan, S., Islam, A.B.M.A.A., 2020. A generalized mechanism beyond NLP for real-time detection of cyber abuse through facial expression analytics. In: Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. MobiQuitous '19, Association for Computing Machinery, New York, NY, USA, pp. 348–357.

Sinha, S., 2023. Number of connected IoT devices. URL https://iot-analytics.com/number-connected-iot-devices/, (Accessed: 2025-01-28).

Sivasankari, R., Rakshith, D., Ramesh, M., Gokulavasan, S., 2024. Prevention and detection of ARP spoofing-a survey. In: Next-Gen Technologies in Computational Intelligence: Proceedings of the International Conference on Next-Gen Technologies in Computational Intelligence (NGTCA 2023). CRC Press.

SkyQuest, 2025. IoT smart homes market insights. URL https://www.skyquestt.com/report/iot-smart-homes-market, (Accessed: 2025-01-28).

Sodhro, A.H., Awad, A.I., van de Beek, J., Nikolakopoulos, G., 2022. Intelligent authentication of 5G healthcare devices: A survey. Internet Things 20, 100610.

Stallings, W., Brown, L., 2018. Computer Security: Principles and Practice, fourth ed. Pearson.

Sun, S., Fu, X., Luo, B., Du, X., 2020. Detecting and mitigating ARP attacks in SDN-based cloud environment. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, pp. 659–664.

Trabelsi, Z., El-Hajj, W., 2010. On investigating ARP spoofing security solutions. Int. J. Int. Protoc. Technol. 5 (1–2), 92–100.

Wang, Y., Feng, X., Chen, Y., Zhou, L., Zhu, Y., Wu, J., 2021. A dual detection method for siemens inverter motor modbus RTU attack. J. Comput. Commun. 9 (07), 91–108.

Xia, J., Cai, Z., Hu, G., Xu, M., 2019. An active defense solution for ARP spoofing in OpenFlow network. Chin. J. Electron. 28 (1), 172–178.