IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 May 2024, accepted 30 May 2024, date of publication 5 June 2024, date of current version 13 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3409679

RESEARCH ARTICLE

A Novel Mechanism for Detection of Address Resolution Protocol Spoofing Attacks in Large-Scale Software-Defined Networks

LAURENT PATRICE, RAMADHANI SINDE, (Member, IEEE), AND JUDITH LEO, (Member, IEEE)

School of Computational and Communication Science and Engineering, The Nelson Mandela African Institution of Science and Technology, Arusha 44740, Tanzania

Corresponding author: Laurent Patrice (patricel@nm-aist.ac.tz)

ABSTRACT Address Resolution Protocol (ARP) spoofing has been a long-standing problem with no clear remedy until now. The attacks can be launched easily utilizing an enormous number of publicly available tools on the web; however, they are extremely tough to counterattack due to ARP's stateless nature for not authenticating ARP replies for a subsequent request. Previous studies have demonstrated significant efforts to counterattack these assaults in Software-Defined Networks (SDN); however, much effort has been focused solely on detecting the assaults, with little effort being made to address performance bottlenecks, scalability, and Single Point of Failure (SPOF) issues in large-scale networks. In this study, we focus on developing ARP spoofing attacks detection mechanism in large-scale SDN that is immune to SPOF and provides enhanced network performance and scalability. The main purpose is to enable controllers to intercept and analyze all incoming ARP packets, learn address mappings, and store them in the application's memory to be used as a basis for ongoing ARP cache comparisons while maintaining a global cache in a controller. To achieve the goal of this study, a simulation experiment in a closed network environment was undertaken to precisely monitor network traffic and result patterns. Mininet and the Open Network Operating System were used to implement the data plane and OpenFlow controllers. The results show that, the proposed solution is resistant to ARP spoofing attacks, with an average detection and mitigation time of 4.3 and 26.19 milliseconds, respectively. Further significant improvements have been observed in alleviating SPOF and performance bottlenecks.

INDEX TERMS ARP cache poisoning, ARP spoofing, software-define network, network security, distributed controllers.

I. INTRODUCTION

The Internet has been an incredibly dynamic environment that is always evolving. It has become renowned for its quick expansion and widespread use due to the continual development of telecommunication networks and infrastructure worldwide [1]. Many businesses have managed to have a web presence since the Internet has brought an important opportunity for online businesses. Businesses are constantly looking for ways to make their products and services internet-enabled [2]. The fast growth of the Internet has resulted in most people relying on it for their daily activities [3], [4]. Due to the increased dependence on Internet usage, cyber-attacks and criminal activities have become prevalent in our computing world [5]. Cyber-attacks can result in many security consequences, including identity theft and financial loss due to business disruption, theft of information and money, and loss of intellectual property rights, which can damage the competitive advantage of an organization and its reputation. Any computing device with Internet access is subjected to

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Pau.

malicious and criminal attacks [6]. The surge in malicious attacks and activities presents a huge threat to network security. Computer systems and network assaults are possible at numerous levels, from the data link layer to the application layer of the OSI model.

Address Resolution Protocol (ARP) spoofing is one of the assaults that, if gone unnoticed, can be the initial step in further threats such as Denial of Service (DoS), session hijacking, and Man-in-the-middle (MITM) attacks [7], [8], [9]. ARP is a fundamental protocol for data link layer communication. It determines the hardware address of a network device based on its Internet Protocol (IP) address [10]. ARP's stateless nature exposes a number of flaws, one of which is ARP spoofing. The weakness is exploited via intercepting traffic between two hosts on a Local Area Network (LAN). Notwithstanding the ease with which an assault may be launched using an enormous number of freely tools accessible on the web, it is extremely tough to counterattack. End users generally remain unconscious of the fact that they are the targets of ARP spoofing assaults until they discover the disclosure of their confidential information, since the assailant prefers to monitor the conversation on the channel rather than interrupt it [8].

The end node intending to send a data frame in unicast without first identifying the target hardware address must first broadcast an ARP request message containing the IP address of the destination host with which it intends to communicate. The host with the intended IP address will respond to the sender with a unicast ARP reply message that includes its Media Access Control (MAC) address [8], [11]. The ARP request is prone to ARP spoofing since it need to be broadcasted. Moreover, ARP has a weakness in not verifying the authenticity of the ARP responses, whether or not the reply comes from the intended device [8], [9], [12], [13], [14], [15], [16], [17]. It lacks a reliable technique to authenticate the ARP responses to determine if they are genuine or fake. Furthermore, the attacking host can send an ARP reply to the target host despite the fact that the host has not requested it at that time. Attackers exploit the ARP weaknesses by linking the attacker's MAC address with the IP address of a target device [13], [14], [15], [17]. Since network communication in the Data Link Layer relies on the MAC addresses of the devices, the attacker is able to mirror both parties, listen to the conversation between these two parties, and gain access to the data shared by both parties [8], [15], [18]. Since ARP is the fundamental protocol for LAN communication and ARP spoofing assaults can be the initial step in further threats such as MITM, session hijacking, and DoS, implementing effective measures against ARP spoofing is important to prevent the impacts that can result from these attacks. Detecting and mitigating ARP spoofing attacks is critical in both traditional and Software-Defined Networking (SDN) architectures. Since these networks have distinct architectural designs, solutions for detecting these attacks vary.

SDN is a network communication design that seeks to overcome the constraints of traditional networks. Importantly, it overcomes the security challenges facing the traditional network by separating the traditional network operations into control and data planes to enable more intelligent network operations [19], [20], [21], [22]. It enables dynamic, programmable, and efficient network configurations to attain optimal network performance and monitoring [21]. SDN is intended to overcome the static design of traditional networks and introduce centralized network intelligence by separating the forwarding processes of network packets from the network control processes [22]. In a traditional network, each device processes its data and forwards packets in accordance with its algorithms and configurations. There is typically no central control node that controls the flow of data in the network. On the other hand, SDN implements the separation between the data forwarding and the control processes to introduce network controllers for defining data flow rules and run algorithms in the switches. The introduction of the centralized control plane with single or multiple controllers helps improve network performance and provides efficient mechanisms for network monitoring and management [13], [21], [22], [23], [24]. The control and management processes are shifted to the control plane, which has a variety of applications such as routing logics, network rules, administrative tools, and security measures [19], [21]. In the SDN network environment, switches are responsible for forwarding packets in accordance with data flow rules that have been installed by the controllers [25]. Introducing network controllers in SDN-based networks for analyzing traffic and controlling devices, enhances the network security features for the detection and mitigation of cyber-attacks. Although SDN offers significant advantages over traditional networks, its centralized control design creates additional challenges in terms of performance bottlenecks, reliability, interoperability, and fault tolerance [26], [27], [28].

To effectively address performance bottlenecks, reliability, interoperability, and fault tolerance concerns, this study leverages the benefits of clustering and distributed controller architecture and employs the clustered and distributed SDN control plane architecture. It combines features for distributed as well as clustered controllers to achieve load balancing, scalability, efficient network management, and address Single Point of Failure (SPOF). SDN controller clustering connects group of controllers as one system to provide a unified SDN control plane. In this architecture, there is a primary controller and secondary controllers that provide redundancy when the primary controller fails. Clusters are commonly used to improve network service quality by decreasing downtime and failure by allowing other controllers in the group to take over in the case of a failure [29]. Each controller in the cluster is linked to other controllers and exchanges data to accommodate network demand. The primary controller shares the network information with the secondary controllers to address SPOF. The use of clustered controller architec-

ture has challenges related to scalability and performance bottlenecks, especially for large-scale, distributed networks, and service providers' networks [27]. Distributed controller architecture is recommended for addressing scalability and performance bottleneck concerns in large-scale networks. It involves the deployment of multiple controllers at different locations or network domains, where each controller instance manages a specific network segment [27], [28]. By combining the features for distributed and clustered control design in the solution we are proposing, will help address performance bottlenecks, reliability, scalability, interoperability, and SPOF issues, in addition to the detection of ARP spoofing attacks in large-scale SDN.

Preceding studies have demonstrated great efforts to enhance ARP security to prevent ARP spoofing attacks in SDN. However, much of the effort mainly focused on the detection of ARP spoofing attacks, while less effort was made on addressing performance bottlenecks, scalability, and fault tolerance in large-scale networks [7], [19], [20], [25], [30], [31]. Moreover, in the current network era, where optimal performance and minimal network service downtime are of paramount importance for large scale networks such as data centers and cloud-based LANs, it is uncertain whether the current methods are still relevant in these networks. In this paper, we present an approach for detecting ARP spoofing attacks in SDN that is immune to SPOF and overcomes performance constraints, reliability, and scalability concerns. The essential idea of this effort is to enable SDN controllers to intercept and analyze all incoming ARP packets, learn ARP mappings, and store them in the ARP spoofing detection application memory, rather than relying only on the controller's default ARP cache. The controllers will still maintain a global ARP mapping cache obtained when devices connect to the network. Thus, controllers will maintain two ARP cache maps, one in the memory of the application and the other in the controller's Operating System (OS) cache. The SDN controller, through the application, will compare the ARP mapping of the incoming packets with the two stored ARP maps to detect any illegal ARP mapping. The attacking host, once detected, will temporarily be blocked from sending additional network traffic to prevent damage that can result from the assaults. Through the distributed and clustered architecture, the solution will be able to address SPOF, scalability, and performance bottlenecks such that each controller will manage part of the network at a time while having a global view of the entire network as well as acting as a backup to other controllers. In case of a failure of any controller, backup controllers will immediately take over with minimal delays.

Our significant contributions for this work can be outlined as follows:

- We propose a mechanism for the detection of ARP spoofing attacks in large-scale SDN that is immune to SPOF and provides enhanced network performance and scalability.
- We implement an ARP spoofing detection approach that utilizes the detection application's ARP cache as a basis for ongoing comparisons with the global controller's ARP cache, which is implemented utilizing a distributed and clustered SDN control plane.
- We evaluate the efficacy of the proposed mechanism by conducting controlled experiments in network setups of different sizes and comparing results with the preceding studies.

II. RELATED WORKS

ARP spoofing has been a long-standing issue with no definitive solution until currently. Efforts are being made by researchers to enhance the ARP security to avoid spoofing and associated attacks in the SDN environment. In this part, we explore various similar efforts and their limitations.

Hnamte and Hussain [23] proposed an innovative solution to counterattack ARP spoofing assaults in SDN environment. The solution deployed a deep learning (Deep Neural Network) model in the detection scheme to enhance its capability to detect and counterattack the assaults. A dedicated machine for continuous monitoring of the network topology and detection of anomalies was deployed. The solution was evaluated in network setups of various sizes to test its effectiveness across different network sizes. The solution demonstrated excellent results in detecting the assaults as well as CPU utilization and network throughput, however, the proposed solution can suffer from SPOF in case the dedicated device for anomaly detection fails.

Jamil et al. [20] suggested an auto detection and mitigation methodology for ARP spoofing and Distributed Denial of Service (DDoS) attacks in SDN. The methodology deployed one SDN controller to monitor, manage traffic, define data flow rules to the SDN switches, and maintain communication path. In additional to the controller, a server was used for packet analysis from all hosts in the network and attacks monitoring. The proposed solution demonstrated fruitful results in detecting malicious ARP packets, however, it may suffer SPOF in case the server or the controller is under attack or failure. Furthermore, the use of a single controller can result in performance bottlenecks for large-scale networks.

A mechanism by Aldabbas and Amin [19] was proposed to handle address spoofing attacks in SDN-based IoT, which deployed a single controller to handle control plane operations. Furthermore, a server was deployed to gather ARP packets from the data plane and analyze them for possible malicious traffic. The deployment of the server to handle ARP packets was to reduce processing overheads to the controller, however, this approach can result in SPOF if the server fails. Moreover, the use of a single controller can result in performance bottlenecks for large-scale networks.

Saritakumar et al. [24] proposed an algorithm against ARP poisoning attacks in SDN that makes detection decisions based on IP-MAC IP binding as well as ARP reply counts. The controller keeps track of the MAC-Port details as well as the ARP reply iteration counts in the table for continuous monitoring. The algorithm demonstrated effectiveness

in detecting the assaults. However, there are limited details related to large-scale networks.

A study by Girdler and Vassilakis [13] focused on combating ARP spoofing in SDN by introducing an Intrusion Detection and Prevention System that analyzes incoming ARP packets to detect malicious activities and blocklist malicious MAC addresses. The system was evaluated for its effectiveness using specialized software that has been integrated with a user input validation library. The study demonstrated improvements in detecting assaults, intrusion prevention, firewalls, shorter timeouts, and packet dropping. However, there are limited details on how to counterattack SPOF and scalability constraints in large-scale networks.

Sun et al. [25] proposed an approach that used a cluster of controllers to analyze and detect spoofed ARP packets in real time. The controllers were configured in a distributed architecture to allow traffic distribution for performance enhancement. Furthermore, a server computer was used to manage communication among the SDN controllers and provide each controller with a global view of the entire network. The server gathered network information from the controller, stored it, and share it with other controllers to make them learn the global network IP-MAC mappings, however, the proposed solution can result in controllers operating in an isolated fashion in the event of a server failure, causing performance bottlenecks.

Ibrahim et al. [7] suggested an ARP spoofing attack solution that used an SDN controller with an extended proxy capability to detect and drop spoofed ARP packets. The use of ARP proxy functionality was to reduce the CPU load and roundtrip time on the controller. The proposed solution was robust in preventing ARP spoofing attacks and improving the controller's response time, however, the use of a single controller may result in SPOF and performance bottleneck issues in handling large volumes of traffic, especially for large-scale networks such as data centers and cloud-based SDN networks.

A mechanism for defending ARP spoofing attacks proposed by Xia et al. [31] was based on the OpenFlow platform, a module of the POX controller. The solution inspected the ARP packets from the hosts to identify and prevent the spoofed packets. The solution was able to defend against ARP spoofing attacks; however, performance bottlenecks and fault tolerance issues in large-scale networks were not well addressed due to the use of a single controller. Khalid et al. [30] proposed an ARP spoofing attack detection mechanism for checking every ARP packet using a single SDN controller. The suggested solution made use of a POX controller and DHCP server. The suggested solution was based on a reliable IP-MAC table present in the controller. The solution demonstrated resistance to ARP spoofing attacks; however, it may suffer SPOF and performance bottlenecks in large-scale networks due to the use of a single SDN controller.

Table 1 presents a summary for comparative analysis of prior studies for ARP spoofing attacks in SDN, with their strengths and limitations highlighted.

As discussed in the literature, most of the studies have inadequately addressed SPOF, scalability, and performance bottlenecks in large-scale networks. Their efforts mainly focused on the detection of forged ARP packets. Addressing SPOF, scalability, and performance bottlenecks is of high priority for large-scale networks to enhance optimal network performance, scalability, interoperability, and service availability. In this study, we seek to develop a mechanism for detecting ARP spoofing attacks in SDN that is immune to SPOF and helps alleviate scalability as well as performance bottleneck issues.

III. PROPOSED SOLUTION

This work presents a solution to prevent ARP spoofing attacks on large-scale SDN networks that is immune to SPOF and performance bottlenecks. This part begins by describing the detection of ARP spoofing attacks, followed by how the solution addresses SPOF and performance bottlenecks.

A. ARP SPOOF DETECTOR

As discussed earlier, the previously mentioned solutions are evidently not fully practicable in large-scale SDN networks with massive network traffic that demand excellent performance and high network service availability. Given this vantage point, we develop an SDN application prototype that resists ARP spoofing assaults while taking into consideration SPOF, scalability, and performance bottleneck difficulties. The ARP spoofing detection application intercepts and analyzes all incoming ARP packets from the data plane. It extracts the IP-MAC details for ARP requests and replies and store them into ARP requests and reply logs, respectively, in its memory. We make the assumption that, at first, when devices connect to the network, they are all legitimate, and their valid IP-MAC details are captured by the SDN controller to construct the global ARP cache table that is stored in its memory. The host is considered malicious only when it starts sending malicious ARP payloads to impersonate other legitimate hosts. After the application has extracted the IP-MAC details of the incoming ARP packets, it then compares them with the previously stored information and with the global ARP table stored in the memory of the controller to check for any illegal IP-MAC associations. Any ARP reply packet is first checked to see if it has its corresponding ARP request in the ARP request log. In the event that there is no corresponding request, the packet is flagged as malicious and dropped. The ARP reply packet is only valid if it has its corresponding request, otherwise, it is malicious.

Furthermore, the application checks for associations of a MAC address with multiple IP addresses in the ARP table mappings. A valid IP-MAC mapping is only when a single MAC address is associated with a single IP address. If the

TABLE 1. Comparisons of ARP spoofing related works in SDN environment.

Study	Proposed solution	Strengths	Limitations
Hnamte and Hussain [23]	An innovative solution to counterattack ARP spoofing assaults in SDN environment that uses DNN model to improve assaults detection	Dynamic ARP cache management, real-time traffic analysis, adaptation to network of varying sizes, and adaptive flow rule orchestration	May suffer from SPOF in case the dedicated device for anomaly detection fails
Jamil et al. [20]	An auto detection and mitigation methodology for ARP spoofing and Distributed Denial of Service (DDoS) attacks in SDN	Enhanced network security, CPU utilization, and network latency	• Limited to use of a single controller • May suffer from SPOF in case a server for packet analysis fails
Aldabbas and Amin [19]	A mechanism for detection of ARP spoofing attacks in SDN-based IoT	Enhanced network throughput, attack detection, and reduced mitigation time by 35%	• Limited to use of a single controller • A server failure for packet analysis can result in SPOF
Saritakumar et al. [24]	Algorithm against ARP poisoning attacks in SDN that make detection decisions based on IP-MAC IP binding as well as ARP reply counts	Improved network security	There is limited discussion related to large-scale networks
Girdler and Vassilakis [13]	Intrusion Detection and Prevention System which analyses incoming ARP packets to detect malicious activities and blacklist malicious MAC addresses	Improved detection of the assaults, intrusion prevention, firewall, shorter timeouts, and packet dropping	Limited details to counterattack SPOF and scalability constraints in large-scale networks
Sun et al. [25]	An approach to combat ARP spoofing in cloud computing environment that used a cluster of controllers to analyze and detect ARP spoofed packets in real time	Increased network performance, statistical data monitoring	In the event of a server failure, can result in controllers operating in isolated fashion
Ibrahim et al. [7]	A Secure Mechanism to Prevent ARP Spoofing and Broadcasting in SDN	Robust in detecting and mitigating ARP spoofing and broadcasting threats	Limited to use of a single controller
Xia et al. [31]	A mechanism for defending ARP spoofing attack in SDN	Reduced security risks for ARP spoofing	Limited to use of a single controller
Khalid et al. [30]	A mechanism for detection of ARP spoofing attacks with extended module for checking every packet in the network for detecting any possible spoofed packets	Lightweight, reliable IP-MAC table for continuous monitoring	Limited to use of a single controller

application detects that a single MAC address is associated with multiple IP addresses, the packet is flagged as malicious and dropped. The detected host transmitting the spoofed ARP packets is temporarily blocked from further sending malicious ARP packets and is isolated from network communication. We further make the assumption that, the attacker may connect to the target network using stolen credentials or any other method and use a legitimate host in the network to send malicious ARP packets. With this perspective, we consider blocking the attacking host temporarily rather than permanently to avoid service downtime in case attackers have compromised the legitimate hosts that

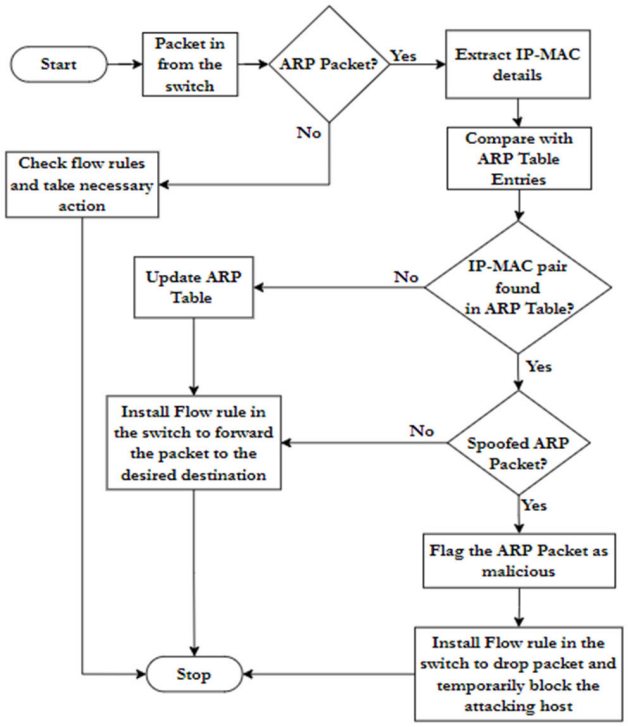
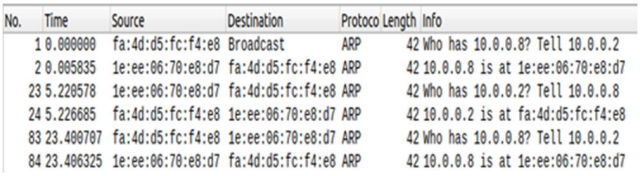


FIGURE 1. ARP spoof detector flowchart.



No.	Time	Source	Destination	Protoco	Length	Info
1	0.000000	fa:4d:d5:fc:f4:e8	Broadcast	ARP	42	Who has 10.0.0.8? Tell 10.0.0.2
2	0.005835	1e:ee:06:70:e8:d7	fa:4d:d5:fc:f4:e8	ARP	42	10.0.0.8 is at 1e:ee:06:70:e8:d7
23	5.220578	1e:ee:06:70:e8:d7	fa:4d:d5:fc:f4:e8	ARP	42	Who has 10.0.0.2? Tell 10.0.0.8
24	5.226685	fa:4d:d5:fc:f4:e8	1e:ee:06:70:e8:d7	ARP	42	10.0.0.2 is at fa:4d:d5:fc:f4:e8
83	23.400707	fa:4d:d5:fc:f4:e8	1e:ee:06:70:e8:d7	ARP	42	Who has 10.0.0.8? Tell 10.0.0.2
84	23.406325	1e:ee:06:70:e8:d7	fa:4d:d5:fc:f4:e8	ARP	42	10.0.0.8 is at 1e:ee:06:70:e8:d7

FIGURE 2. ARP packets exchange for communication establishment.

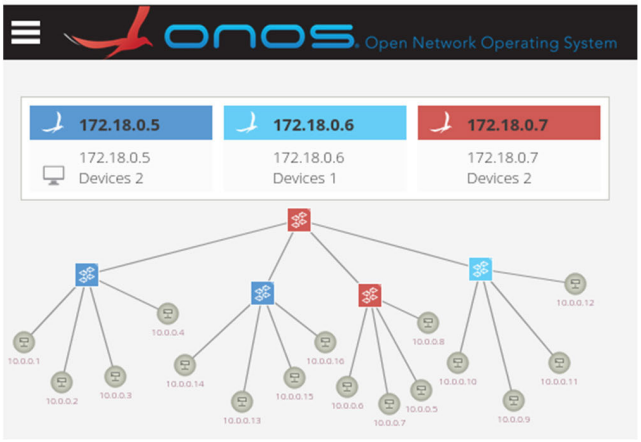


FIGURE 3. A cluster of ONOS instances in distributed architecture.

provide critical network services to the users to achieve their goals.

The ARP cache in the application memory remains unchanged from external applications and acts as the basis for ongoing comparison to the controller's default ARP cache. The comparisons between the controller's ARP cache and that of the application aid in detecting changes in the ARP cache that are susceptible to spoofing. If the ARP cache changes, the application will immediately detect and temporarily block the host sending malicious ARP packets. The ARP spoofing detection flow is as presented in Fig. 1. The decision to block the assaulting host is based on an experiment that determines how long it takes for the host to establish network communication. According to the experiment simulations, the connection is established within a period of 5 seconds, as shown in Fig. 2.

Based on these data, we determine the number of malicious ARP packets identified within 10 seconds. If at least three malicious ARP packets are discovered, the attacking host gets flagged as malicious and blocked. We provide a duration of 10 seconds and at least 3 malicious ARP packets to avoid false positive findings that may result in blocking legitimate hosts and disrupting network services.

B. ADDRESSING SPOF, SCALABILITY, AND PERFORMANCE BOTTLENECKS

In addressing SPOF, scalability, and performance bottlenecks, we apply the distributed and clustered control plane architecture. A cluster of three SDN controllers is configured in a distributed manner. Each controller manages a specific network segment while acting as backup to other controllers through the primary and secondary principle. Clusters form a unified system that has a global view of the entire network and exchanges information to meet network demands. On the other hand, distributed systems allow distributing network traffic among the SDN controllers to overcome SDN controller overloads, thus enhancing network performance and scalability. The distribution of network traffic is based on network segment rather than individual network traffic in the sense that network traffic originating from a particular network segment are managed by a particular SDN controller and are automatically redirected to another controller only if the primary controller for that particular network segment has failed. The use of three controllers is based on the fact that if one of the controllers fails, the other two can still share and distribute the network load among themselves, reducing controller overload. Whereas if we use two controllers and one fails, the remaining controller may experience performance bottlenecks. Based on this assumption, we conclude that, for large-scale networks, at least three controllers should be implemented to adequately address SPOF, scalability, and performance bottlenecks.

IV. MATERIALS AND METHODS

In this section, we describe the materials and methods used in implementing a mechanism for the detection of ARP spoofing attacks.

A. EXPERIMENTAL SETUP

A computer running Ubuntu 22.04 LTS OS with an Intel Core i5, 16GB of RAM, and 10GB of swap memory was used for

TABLE 2. Experimental simulation setup details.

Experiment setup	Number of switches	Number of hosts	Recorded experimental simulations
1	5	16	6
2	7	36	6
3	9	64	6
4	11	100	6
5	13	144	6
6	15	196	6

experimental simulations. We configured three instances of Open Network Operating System (ONOS)[1] controllers, version 2.7.0, running in Docker[2] containers. ONOS is an open source SDN controller that supports distributed SDN control plane. The controllers were configured as a cluster and in a distributed manner, as indicated in Fig. 3. Docker containers are lightweight and less resource consuming as compared to full virtual machines. Docker packages applications, their dependencies, and system libraries into a container to ensure that applications function effectively and reliably across several environments [32], [33], [34], [35]. The use of Docker containers was selected to minimize computational resource constraints, as may be the case for using full virtual machines. Mininet[3] tool version 2.3.0 was used as a testbed for simulation of data plane network operations. The experiment was conducted in a closed network environment to accurately monitor the network traffic and result patterns. The experiment simulation setup details are as summarized in Table 2. One of the host in the network topology was used as the attacker to send spoofed ARP payloads. Arpspoof tool was used to craft and launch the attacking payloads. Wireshark[4] tool was used to capture ARP packets in pcap data format for further analysis. The captured packets were scrutinized for any discernible discrepancies. For instance, the legitimate MAC address for the host with IP 10.0.0.9 is 66*:82:a7:8a:27:48*; however, it is associated with the second MAC address of *d2:17:c1:27:4e:87,* as indicated in Fig. 4.

Performance metrics for resource usage, such as memory and CPU, were measured. Prometheus,[5] cAdvisor[6] (Container advisor), and Grafana[7] tools were used. cAdvisor is an open-source software capable of extracting resource utilization and performance metrics from Docker containers to help understand the resource consumption of the running containers. The cAdvisor's gathered data is exposed to Prometheus, an open-source tool well suited for dynamic and containerized environments. Prometheus gathers metrics from containers, applications, and services and provides a query language for analyzing them. Grafana is used in conjunction with Prometheus for the best visualization of the analyzed data. It provides interactive and customizable dashboards and helps present the analyzed data in the form of charts and graphs.

B. APPLICATION DEVELOPMENT AND DEPLOYMENT

In the development of the ARP spoofing detection application, the Extreme Programming (XP) software development technique was used so as to adapt to the continuously changing application requirements. The implementation of the application was based on ONOS controllers. The ONOS v2.7.0 architype framework was used as a base for building the application to allow seamless integration with existing ONOS applications. The Java programming language and the ONOS Java-based API were used to develop the application. Apache Maven[8] was used to compile the code and build the ONOS Application aRchive (.OAR) file. OAR is a file format for ONOS installer files required for the deployment of ONOS applications in the controller. Maven supports an easy way of compiling and building ONOS applications through its project object model (POM) and a set of plugins for dependence management. The installer file for the developed detection application was deployed in the configured controllers for testing and evaluation of its efficacy. The application was tested by running simulation attacks in a controlled environment. Attack payloads were crafted and launched through the Xterm console of the attacking host in mininet.

The application log was examined and compared with the captured details from Wireshark by monitoring the switch port to which the malicious host is connected.

C. SDN CONTROLLERS CLUSTERING

SDN controllers clustering was achieved through the Atomix[9] framework. It allows the development of distributed applications in a cloud-based environment. Prior to configuring the controllers cluster, the cluster of Atomix nodes was configured for data storage and coordination of the controllers. Three Atomix nodes were configured as a cluster to connect to each other, share cluster information, and provide redundancy for controller coordination. The Atomix cluster provided a base for the formation of the SDN controllers cluster. Three ONOS controllers were configured as a cluster on top of the Atomix cluster, whereby each controller was configured to link with other controllers to have a global view of the network and be able to exchange information with other controllers in the cluster.

[1]Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions (opennetworking.org).

[2]Docker: Accelerated Container Application Development.

[3]Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet.

[4]Wireshark: The world's most popular network protocol analyzer.

[5]Prometheus - Monitoring system & time series database.

[6]GitHub - google/cadvisor: Analyzes resource usage and performance characteristics of running containers.

[7]Grafana: The open observability platform | Grafana Labs.

[8]Maven – Welcome to Apache Maven.

[9]Atomix.

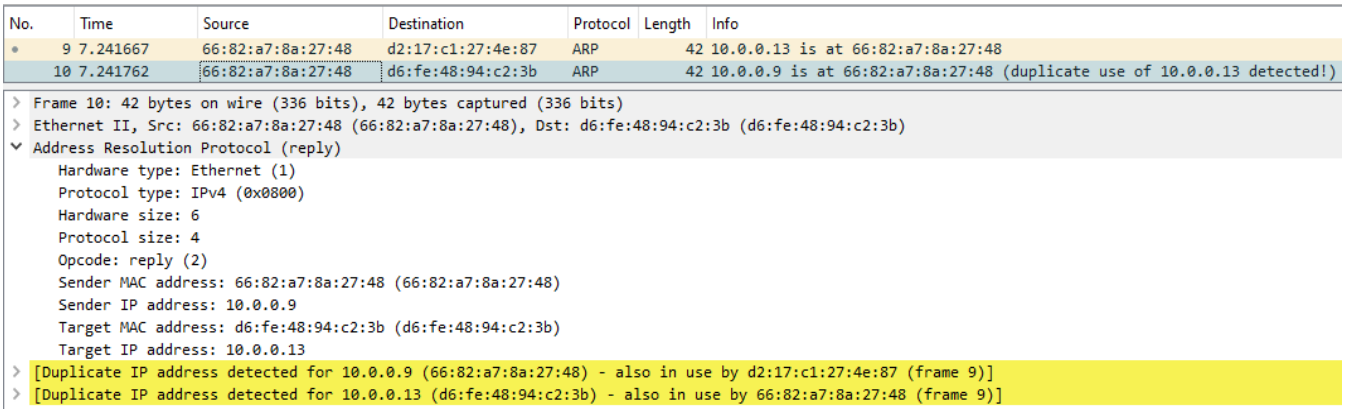


FIGURE 4. Malicious ARP packet.

V. RESULTS AND DISCUSSION

In this section, we present the findings of the proposed mechanism.

A. DETECTION OF SPOOFED ARP PACKETS

The efficacy of the suggested solution against ARP spoofing attacks was evaluated by collecting data from controlled experiments and analyzing them using Wireshark and the application log. Spoofed ARP payloads crafted using Arpspoof were launched by one of the hosts acting as the attacker to poison the ARP caches of other hosts in the network. The proposed solution effectively identified spoofed ARP packets, including spoofed gratuitous ARP reply packets, and blocked the malicious host from sending further malicious traffic. A gratuitous ARP reply is a type of ARP packet that responds without a corresponding request. The mechanism demonstrated enhanced network security through real-time detection and mitigation of ARP spoofing assaults. The experimental results for the detection of spoofed ARP packets are as depicted in Fig. 5.

B. ARP SPOOFING DETECTION AND MITIGATION TIME

The study conducted experiments with various simulation scenarios and numbers of hosts to determine response time variances as the network grows. We utilized 5 switches and 16 hosts, 7 switches and 36 hosts, 9 switches and 64 hosts, 11 switches and 100 hosts, 13 switches and 144 hosts, and 15 switches and 196 hosts as experimental settings. Various simulation scenarios were used to assess how well the proposed mechanism scales with the number of hosts, the detection mechanism's performance across different network sizes, the detection mechanism's resource utilization as the network grows, and the mechanism's adaptability to different network sizes. We recorded and analyzed the time discovery to measure the average duration from the launch of an attack to its detection. We describe time discovery as the time period when the attacker launches the fabricated ARP packet (when it arrives at the SDN switch as packet_in) and when the SDN controller detects it. We further measured and analyzed the mitigation time. The mitigation time is the time duration from the detection of the attack to the time when the attacker is blocked. The results demonstrated excellent performance in detecting and mitigating spoofed ARP packets, as illustrated in Fig. 6. A slight increase in detection and mitigation time was observed when the number of hosts was increased. The observed slight increase in detection and mitigation time provides confidence that the proposed solution is suitable for large-scale networks. In the proposed solution, the mitigation time was observed to be larger than the detection time due to the overhead for locating the attacking host connection point in the network as well as the provision of detecting at least three malicious ARP packets before blocking the attacker.

Studies by Jamil et al. [20], Aldabbas and Amin [19], and Sun et al. [25] deployed a server computer in the control plane so as to reduce the controller overheads in processing and analyzing ARP packets. The detection and mitigation response times were recorded in these studies. For instance, [20] at a network setup of 20 hosts observed a detection time of about 0.2 seconds; [19] at a network setup of 50 hosts observed a detection time of about 2 seconds; and [25] at the network setup of 120 hosts recorded a detection time of about 191.3 milliseconds; while in the proposed solution, at a network setup of 144 hosts, a detection time of about 5.5 milliseconds was observed, which is much smaller as compared to the observed time in these studies. As the number of hosts in the proposed solution increased to 196, the detection time was observed to be about 11.2 milliseconds, which was still significantly shorter than in prior studies. On the other hand, the mitigation time of the proposed solution was compared with that observed in the previous studies. Whereas, [19] observed a mitigation time of about 8 seconds at a network setup of 50 hosts, and [20] observed a mitigation time of about 0.3 seconds at a network setup of 20 hosts. In the proposed solution, a mitigation time of 24.83 milliseconds was observed at a network setup of 64 hosts. As the number of hosts increased to 196, a mitigation time of about 38.33 milliseconds was observed, which was still shorter as compared

```
karaf@root > log:display org.onosproject.arpspoofdetector
10:45:02.135 WARN [ArpSpoofingDetection] ARP packet with illegal IP-MAC associations detected..!!
10:45:02.136 WARN [ArpSpoofingDetection] IP '10.0.0.124' is associated with MAC addresses '0E:01:F2:26:C4:1F' and '3A:E9:30:1A:7F:C0'
10:45:02.142 WARN [ArpSpoofingDetection] ARP packet with illegal IP-MAC associations detected..!!
10:45:02.145 WARN [ArpSpoofingDetection] ARP packet with illegal IP-MAC associations detected..!!
10:45:02.145 WARN [ArpSpoofingDetection] IP '10.0.0.62' is associated with MAC addresses 'AA:54:5E:9F:E7:44' and '3A:E9:30:1A:7F:C0'
10:45:02.149 WARN [ArpSpoofingDetection] IP '10.0.0.124' is associated with MAC addresses '0E:01:F2:26:C4:1F' and '3A:E9:30:1A:7F:C0'
10:45:02.150 WARN [ArpSpoofingDetection] Detected 3 Malicious ARP packets from MAC 3A:E9:30:1A:7F:C0 within the last 10 seconds:
10:45:02.154 WARN [ArpSpoofingDetection] Detected 3 Malicious ARP packets from MAC 3A:E9:30:1A:7F:C0 within the last 10 seconds:
10:45:02.160 INFO [ArpSpoofingDetection] Temporarily blocked device for 60 seconds with MAC: 3A:E9:30:1A:7F:C0
```

FIGURE 5. Detected malicious ARP packets.

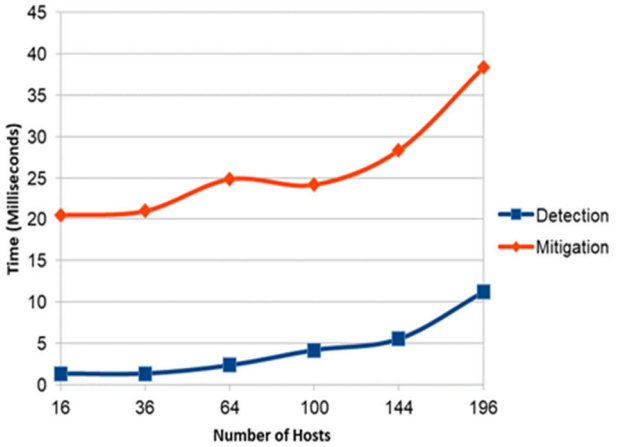


FIGURE 6. Detection and mitigation response time.

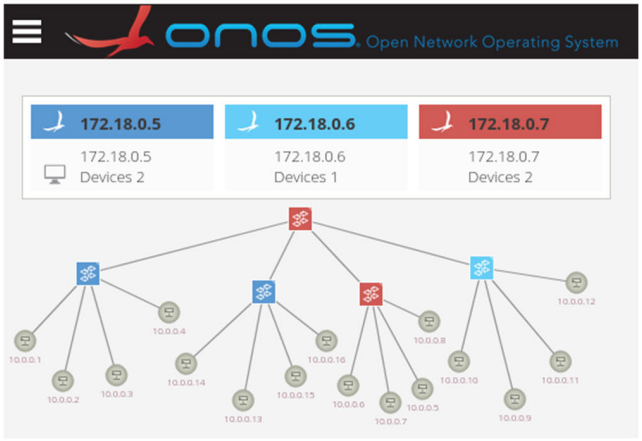


FIGURE 7. Network setup with three clustered controllers in a distributed setup.

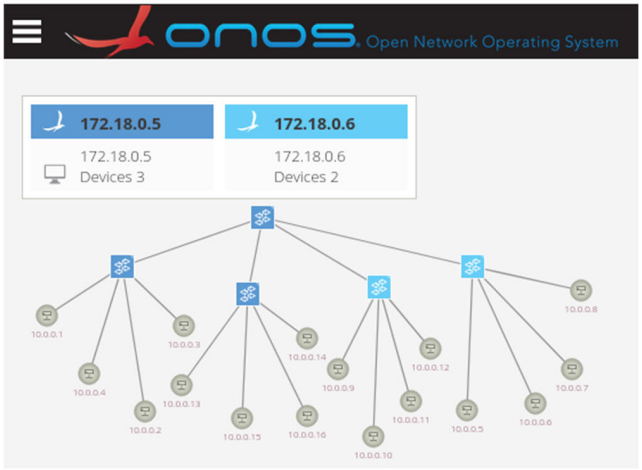


FIGURE 8. Switches and hosts automatically relocated to other controllers after one of the controller was shut down.

to existing solutions. From the experimental results, the proposed mechanism demonstrated significant improvements in terms of shorter detection and mitigation response times, and it is evident that the use of clustered controllers in a distributed fashion improves network performance and is suitable for large-scale networks that scale based on demand. Fig. 6 shows that the detection and mitigation response curves grow less steep as the number of hosts increases. This indicates that the suggested solution can support networks to scale up, which is the best sign for large-scale networks.

C. ADDRESSING SINGLE POINT OF FAILURE AND SCALABILITY

The SPOF was handled by establishing a cluster of controllers with a global view of the network data. We ran a simulation to see how SDN controllers share network information among themselves. The testing findings revealed that the flow rules residing in one controller were replicated in the other controllers. This confirmed that controllers have a global view of the network and can exchange data among themselves. We next ran a simulation experiment in which we turned off one of the controllers and observed the behavior of the network. The results indicated that switches and hosts handled by the controller that was shut down were immediately transferred to the administration of the other remaining controllers acting as secondary controllers to the one being shut down. Furthermore, the controller primary balancing process was carried out automatically to distribute the relocated hosts among the remaining controllers. The simulation results proved that the mechanism is robust against SPOF and can handle performance bottlenecks through load balancing. Fig. 7 and Fig. 8 illustrates the results for SPOF and load balancing.

Most of the prior studies have inadequately addressed SPOF by relying mostly on the use of a single controller or a single dedicated machine to analyze and detect malicious ARP packets. Inadequate existence of redundant controllers or servers for detection of malicious ARP traffic may result in SPOF constraints. Reliability for detection of spoofed ARP

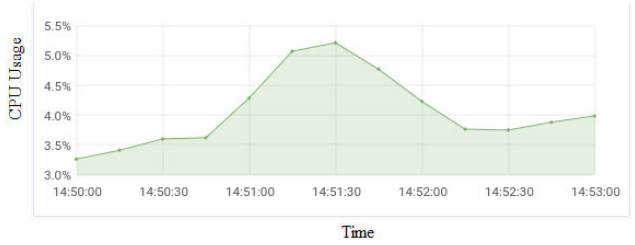


FIGURE 9. CPU usage.

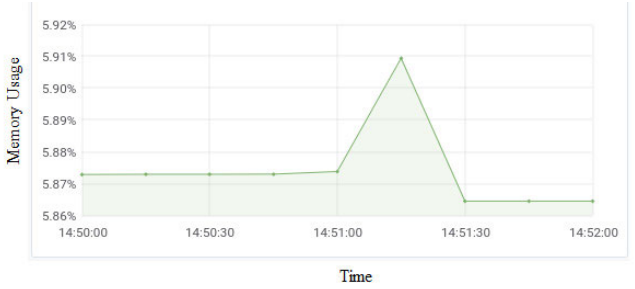


FIGURE 10. Memory usage.

packets being a critical factor, our proposed mechanism has demonstrated high reliability across networks of different sizes. As large-scale networks scale based on demand, the efficacy evaluation of our proposed mechanism was subjected to network setups of varying sizes to test its applicability in large-scale networks. While most of the prior studies have focused on small to medium-sized networks, our proposed mechanism has demonstrated improved performance ranging from small-sized to large-scale networks, making it suitable for large-scale network deployments.

D. MEMORY AND CPU USAGE

Memory and CPU utilization were measured to evaluate the resource consumption before, during, and after the attacks. Attack payloads were launched, and monitoring tools were utilized to collect and analyze the memory and CPU usage metrics. As indicated in Fig. 9 and Fig. 10, attack payloads were launched at around 14:50:45. During the attacking period, a slight increase in memory and CPU utilization was observed. The results show that the proposed solution is efficient in terms of resource consumption. The increase in the resource utilization sustained shortly due to the blocking of the attacking host immediately after the attacks were detected, preventing the attacker from sending further malicious ARP packets. The controllers remained unaffected by the attacker payloads, effectively consuming resources.

VI. CONCLUSION AND FUTURE WORK

ARP spoofing is a vital technique for launching other attacks such as DoS, MITM, and session hijacking. This study offers technological techniques for preventing ARP spoofing attacks in large-scale SDN networks. The detection mechanism keeps two IP-MAC mappings, one in the detection application's memory and the other in the controller's OS. The ARP map in the application memory serves as a basis for ARP cache comparisons with the controller's cache, which aids in detecting any ARP cache modifications. Once detected, the attacking host is temporarily banned from transmitting malicious traffic in order to avoid additional harm from these attacks. The assailant is blocked when at least three malicious ARP packets are detected within 10 seconds to avoid false-positive results. The results indicated that the proposed detection mechanism is resilient against ARP spoofing as it detects and mitigates assaults in significant time. SPOF, scalability, and performance issues were addressed by implementing distributed clustered controllers, with each controller serving as a backup to others. Furthermore, load balancing was done automatically to distribute the load among the controllers. The results demonstrated that the mechanism is robust to SPOF, scalability, and performance constraints. While the proposed approach achieves the study's purpose, there is still room for improvement. Future work might focus on integrating the proposed mechanism with Machine Learning models to enhance ARP spoofing detection efficiency. The model can be built to include attack patterns for other types of network assaults to enhance the mechanism for counterattacking multiple types of attacks. Moreover, integrating the mechanism with network monitoring tools to provide additional features for sending notifications to network administrators for further analysis and reporting may enhance its efficiency. Since the proposed mechanism was evaluated using a machine with inadequate computing resources, the efficacy of the mechanism may be further evaluated by utilizing computers with high computing resources before real-world deployments.

REFERENCES

[1] H. Alshehri and F. Meziane, "Current state of Internet growth and usage in Saudi Arabia and its ability to support e-commerce development," *J. Adv. Manage. Sci.*, vol. 5, no. 2, pp. 127–132, Mar. 2017.
[2] C. Koliouska, Z. Andreopoulou, B. Manos, and P. Lefakis, "The role of Internet in economic development of Vikos-Aoos National Park, Greece," in *Proc. Econ. Balk. East. Eur. Ctries. Chang. World*, vol. 202, Greece, 2013, pp. 202–211.
[3] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya, and J. K. Kalita, "Network attacks: Taxonomy, tools and systems," *J. Netw. Comput. Appl.*, vol. 40, pp. 307–324, Apr. 2014.
[4] M. V. Pawar and J. Anuradha, "Network security and types of attacks in network," *Proc. Comput. Sci.*, vol. 48, pp. 503–506, Jan. 2015.
[5] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," *J. Comput. Syst. Sci.*, vol. 80, no. 5, pp. 973–993, Aug. 2014.
[6] S. Hansman and R. Hunt, "A taxonomy of network and computer attacks," *Comput. Secur.*, vol. 24, no. 1, pp. 31–43, 2005.
[7] H. Y. Ibrahim, P. M. Ismael, A. A. Albabawat, and A. B. Al-Khalil, "A secure mechanism to prevent ARP spoofing and ARP broadcasting in SDN," in *Proc. Int. Conf. Comput. Sci. Softw. Eng. (CSASE)*, Apr. 2020, pp. 13–19.
[8] P. Phetchai, J. D. Ndibwile, D. Fall, S. Kashihara, and S. Tuarob, "Securing low-computational-power devices against ARP spoofing attacks through a lightweight Android application," in *Proc. 21st Int. Comput. Sci. Eng. Conf. (ICSEC)*, Nov. 2017, pp. 1–6.
[9] A. A. Galal, A. Z. Ghalwash, and M. Nasr, "A new approach for detecting and mitigating address resolution protocol (ARP) poisoning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 6, pp. 377–382, 2022.
[10] D. Plummer, *An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48. Bit Ethernet Address for Transmission on Ethernet Hardware*, document RFC 2070-1721, 1982.
[11] J. Kaur and P. Sondhi, "ARP spoofing-based MITM attack in data link layer using the hybrid method-CONVLSTM-ECC," *J. Posit. Sch. Psychol.*, vol. 6, no. 3, pp. 1298–1303, 2022.

[12] F. X. D. Christopher and C. Divya, "Address resolution protocol based attacks: Prevention and detection schemes," in *Proc. Int. Conf. Comput. Netw., Big Data IoT (ICCBI)*. Cham, Switzerland: Springer, 2020, pp. 247–256.
[13] T. Girdler and V. G. Vassilakis, "Implementing an intrusion detection and prevention system using software-defined networking: Defending against ARP spoofing attacks and blacklisted MAC addresses," *Comput. Electr. Eng.*, vol. 90, Mar. 2021, Art. no. 106990.
[14] S. M. Morsy and D. Nashat, "D-ARP: An efficient scheme to detect and prevent ARP spoofing," *IEEE Access*, vol. 10, pp. 49142–49153, 2022.
[15] H. I. Nasser and M. A. Hussain, "Provably curb man-in-the-middle attack-based ARP spoofing in a local network," *Bull. Electr. Eng. Inform.*, vol. 11, no. 4, pp. 2280–2291, 2022.
[16] V. Rohatgi and S. Goyal, "A detailed survey for detection and mitigation techniques against ARP spoofing," in *Proc. 4th Int. Conf. I-SMAC (IoT Social, Mobile, Anal. Cloud) (I-SMAC)*, Oct. 2020, pp. 352–356.
[17] Z. Shah and S. Cosgrove, "Mitigating ARP cache poisoning attack in software-defined networking (SDN): A survey," *Electronics*, vol. 8, no. 10, p. 1095, Sep. 2019.
[18] M. H. Alzuwaini and A. A. Yassin, "An efficient mechanism to prevent the phishing attacks," *Iraqi J. Electr. Electron. Eng.*, vol. 17, no. 1, pp. 125–135, 2021.
[19] H. Aldabbas and R. Amin, "A novel mechanism to handle address spoofing attacks in SDN based IoT," *Cluster Comput.*, vol. 24, no. 4, pp. 3011–3026, Dec. 2021.
[20] F. Jamil, H. Jamil, and A. Ali, "Spoofing attack mitigation in address resolution protocol (ARP) and DDoS in software-defined networking," *J. Inf. Secur. Cybercrimes Res.*, vol. 5, no. 1, pp. 35–46, 2022.
[21] K. Benzekki, A. El Fergougui, and A. E. Elalaoui, "Software-defined networking (SDN): A survey," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5803–5833, 2016.
[22] A. Montazerolghaem, "Software-defined load-balanced data center: Design, implementation and performance analysis," *Clust. Comput.*, vol. 24, no. 2, pp. 591–610, 2021.
[23] V. Hnamte and J. Hussain, "Enhancing security in software-defined networks: An approach to efficient ARP spoofing attacks detection and mitigation," *Telemat. Inform. Rep.*, vol. 14, Jun. 2024, Art. no. 100129.
[24] N. Saritakumar, K. Anusuya, and S. Ajitha, "Detection and mitigation of ARP poisoning attack in software defined network," presented at the *Proc. 1st Int. Conf. Combinatorial Optim.*, Chennai, India, Dec. 2021, pp. 1–9.
[25] S. Sun, X. Fu, B. Luo, and X. Du, "Detecting and mitigating ARP attacks in SDN-based cloud environment," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 659–664.
[26] A. Abdelaziz, A. T. Fong, A. Gani, U. Garba, S. Khan, A. Akhunzada, H. Talebian, and K. K. R. Choo, "Distributed controller clustering in software defined networks," *PLoS ONE*, vol. 12, no. 4, 2017, Art. no. e0174715.
[27] B. Almadani, A. Beg, and A. Mahmoud, "DSF: A distributed SDN control plane framework for the East/West interface," *IEEE Access*, vol. 9, pp. 26735–26754, 2021.
[28] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 333–354, 1st Quart., 2018.
[29] I. M. Suartana, M. A. N. Anggraini, and A. Z. Pramudita, "High availability in software-defined networking using cluster controller: A simulation approach," in *Proc. 3rd Int. Conf. Vocational Educ. Electr. Eng. (ICVEE)*, Oct. 2020, pp. 1–5.
[30] H. Y. Khalid, P. M. Ismael, and A. B. Al-Khalil, "Efficient mechanism for securing software defined network against ARP spoofing attack," *J. Duhok Univ.*, vol. 22, no. 1, pp. 124–131, 2019.
[31] J. Xia, Z. Cai, G. Hu, and M. Xu, "An active defense solution for ARP spoofing in OpenFlow network," *Chin. J. Electron.*, vol. 28, no. 1, pp. 172–178, Jan. 2019.
[32] M. T. Chung, N. Quang-Hung, M.-T. Nguyen, and N. Thoai, "Using Docker in high performance computing applications," in *Proc. IEEE 6th Int. Conf. Commun. Electron. (ICCE)*, Jul. 2016, pp. 52–57.
[33] A. M. Potdar, D. Narayan, S. Kengond, and M. M. Mulla, "Performance evaluation of Docker container and virtual machine," *Proc. Comput. Sci.*, vol. 171, pp. 1419–1428, Jun. 2020.
[34] E. N. Preeth, F. J. P. Mulerickal, B. Paul, and Y. Sastri, "Evaluation of Docker containers based on hardware utilization," in *Proc. Int. Conf. Control Commun. Comput. India (ICCC)*, Nov. 2015, pp. 697–700.
[35] P. Saha, A. Beltre, P. Uminski, and M. Govindaraju, "Evaluation of Docker containers for scientific workloads in the cloud," in *Proc. Pract. Exper. Adv. Res. Comput.*, 2018, pp. 1–8.

LAURENT PATRICE received the B.Sc. degree in computer science from the University of Dar es Salaam, Dar es Salaam, Tanzania, in 2010. He is currently pursuing the master's degree in information systems and network security with The Nelson Mandela African Institution of Science and Technology, Arusha, Tanzania.

From 2011 to 2014, he was a Computer Systems Analyst with CATS Tanzania Ltd. From 2014 to 2018, he was a Computer Systems Administrator with Mzumbe University. In 2018, he began working as an Academician with Mzumbe University, where he teaches computing science studies with the Faculty of Science and Technology. His research interests include network security, penetration testing and ethical hacking, intrusion detection, and hacking countermeasures.

RAMADHANI SINDE (Member, IEEE) received the B.Sc. degree in engineering and technologies in telecommunication and the M.Sc. degree in engineering and technologies in telecommunication specializing in radio engineering, communication systems and equipment from Moscow Technical University of Communication and Informatics, Russia, in 2009 and 2011, respectively, and the Ph.D. degree in information and communication science and engineering with electronics and telecommunication engineering from The Nelson Mandela African Institution of Science and Technology (NM-AIST), Arusha, Tanzania, in 2020.

Since 2014, he has been a Visiting Researcher with the University of Ghana, Carleton University, Oldenburg University, Pedagogical University, Mozambique, and Dhirubhai Ambani Institute of Information and Communication Technology. He is currently a Lecturer with the School of Computation and Communication Science and Engineering, NM-AIST. He is an IoT and Embedded Systems Professional Trainer. He has worked on environmental conservation and biodiversity using emerging technologies, such as the Internet of Things and artificial intelligence. He is working on an energy-efficient IoT-based system using deep reinforcement learning for energy-optimized systems aiming at biodiversity conservation and human-wildlife conflict in Tanzania's national parks and game reserves. His research interests include modeling and performance evaluation of mobile and wireless communications, embedded systems, the Internet of Things, and artificial intelligence enabler embedded systems.

JUDITH LEO (Member, IEEE) received the Bachelor of Engineering degree in electronics and ICT majoring in engineering from Visvesvaraya Technological University, Karnataka, India, in 2009, and the M.Sc. degree in information and communication science and engineering majoring in information technology system development and management, mobile app, and geographical information system and the Ph.D. degree in information and communication science and engineering majoring in machine learning and the Internet of Things from The Nelson Mandela African Institution of Science and Technology (NM-AIST), Arusha, Tanzania, in 2017 and 2020, respectively.

She is currently a Lecturer with the School of Computation and Communication Science and Engineering, NM-AIST. She has experience working with multidisciplinary research action that integrates diverse areas in information and communication technology, gender inclusion and policies, leadership in STEMI, environment, health, social, and business areas. She has won several grants, awards, and scholarships. Her research interests include data science and data-driven solutions, the use of artificial intelligence (AI), machine learning (ML), mobile apps, and the Internet of Things (IoT) for solving societal and industrial challenges, the area of optical fiber installation, bandwidth, and energy optimization, advanced network configuration, system development, and modeling.