An efficient fusion method for Protecting Software-Defined Networks Against ARP Attacks: Analysis and Experimental Validation

Ehab R. Mohamed, Heba M. Mansour, Osama M. El-Komy

Faculty of Computer and Informatics, Zagazig University, Zagazig, Egypt

Emails: Ehab.rushdy@gmail.com ; Eng.hebamohsen.222@gmail.com ; osamaelkomy@yahoo.com

Abstract

In this paper, to protect software-defined networks (SDN) from various ARP attacks, we implement a three-dimensional algorithm (TDA). The main objective of TDA is to limit the methods by which attackers can breach SDN privacy and to prevent the three main types of ARP attacks, such as ARP flooding, ARP spoofing, and ARP broadcasting. This work discusses the three different ARP attack types, which are broken down into five different scenarios, and how the proposed solution detects and mitigates each one. We simulated the five attack scenarios by creating five Python scripts utilizing the Scapy library. And then we applied an efficient TDA to restrict the five scenarios of ARP attacks more efficiently and faster than existing methods. TDA provides the Ryu controller with a modified module to detect and mitigate these types of attacks, using a three-dimensional secure channel to analyze incoming ARP packets, which works as a filter that analyzes and filters incoming ARP packets from malicious ones, and then giving the controller the choice to forward or drop the packet. To simulate our investigation and apply our proposed solution, we used a Mininet emulator. To evaluate TDA, we calculated the delay times, accuracy controller's throughput, bandwidth, and other metrics. The results that we showed after applying TDA 100 times on our test scenarios indicate that the accuracy is 99.9% for the three stages and that the detection and mitigation times are very short compared to the existing solutions, which are that the minimum detection time is only from 0.1ms to 3.6ms, and the minimum mitigation time is only from 0.3ms to 2.9ms. We evaluated our algorithm by other important metrics such as controller bandwidth, which ranged from 18 GB/sec to 17.7 GB/sec in the cases before and after the attack and 16.5GB/sec in the case of attack; controller throughput, which recorded 1.72GB/sec in the case under the attack and reached 2.11GB/sec in the case after defense; and CPU utilization, which recorded 30.4% during the attack and reduced to 0.3% after mitigation. These metrics proved that our algorithm achieved the highest efficiency compared to other work in this field.

Keywords: Software-defined networks; ARP spoofing; ARP flooding; ARP broadcasting; scapy library; Ryu controller

1. Introduction

In this paper, we investigate an important security issue that is one of the main problems of software-defined networks, propose an effective solution to defend against one of the main attacks that compromise SDN, and implement an intensive strategy called a Three-Dimensional Algorithm (TDA) against three species of ARP attacks that menace SDN architecture. Before we explain our experiment, we briefly answer what and why SDN The developing concept of SDN is reshaping how we design and manage networks, and it has created new opportunities for network programming that divides the control plane from the data plane as in [1] and [2], which is considered

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

the main advantage of SDN because administrators can manage, add features and resources, make configurations, and monitor the whole network intelligently, which facilitates the network to be optimized while keeping a global perspective of the entire network, but the controller becomes the target of numerous attack species as a result of this concentration. The Open Network Forum (ONF) characterizes SDN as an evolving architecture that is dynamic, cost-effective, manageable, and adaptive, making it perfect for the high bandwidth and dynamic nature of today's applications. This definition is similar to that in [3]. One of the most important protocols for SDN communication is the ARP protocol. Despite its significance, it is vulnerable to attack. We must understand these attacks so that the attacker does not exploit vulnerabilities in the ARP protocol to attack SDN and discuss proposed solutions to combat these attacks. We discuss three main types of ARP attacks. The first type is an ARP spoofing attack [1]. The most frequent attack is the ARP spoofing attack, in which rogue nodes send out ARP spoofing packets with fake IP or Mac address information. These packets clog up the network and damage the topology. [4] by blocking the controller. The second one is that ARP broadcasting may result in DDOS [5]. And the third is an ARP flooding attack, which overburdens the network by flooding many ARP requests [6], which leads to controller performance deterioration or the entire network going down. These types bring other forms of attack, such as DOS, MITM, etc. To defend against these types of attacks, we implemented an efficient TDA. We simulated our experiment using Linux Ubuntu to run Mininet emulator and created a tree network topology with a central remote controller connected with OpenFlow switches associated with virtual hosts. We used Ryu as a remote controller, then created five attack scenarios in Python using the Scapy library, applied our algorithm many times to each scenario, and recorded the results every time. These results show that TDA is effective, efficient, and more accurate than the existing solutions. TDA achieved accuracy of 99.9%, delay time ranged only from 0.1ms to 3.9ms for detection and from 0.3ms to 2.9ms for mitigation, CPU utilization in the case of attack reached 30.4% and in the case after the attack 0.3%, bandwidth ranged from 18 GB/sec to 17.7 GB/sec in the cases before and after the attack and 16.5GB/sec in the case of attack, and controller throughput recorded 1.72GB/sec in the case under the attack and reached 2.11GB/sec in the case after defense. These results show that TDA achieved efficiency, speed, inclusivity, and consistency.

The remaining contents of this paper are prepared as follows: "Section 2" mentions the previous work on this problem. "Section 3" illustrates the problem statement by focusing on the three types of ARP attacks and the five attack scenarios. "Section 4" explains the proposed solution. Implementation tools and environment setup tools are described in "Section 5". We discuss results and metrics in "Section 6". We document the conclusion and future work in "Section 7". Lastly, we use the RYU controller, which has superior performance in decreasing latency and increasing TCP throughput compared to other controllers [7], so this choice has achieved our aims perfectly. The proposed TDA takes a very small delay in attack detection and mitigation and is considered an inclusive solution to different types of attacks that achieves inclusivity, speed, and consistency.

2 Related Work

For various sorts of ARP spoofing attacks, many solutions have been put forth; in this part, we will highlight the most significant and current ones. In order to create an extensible and customizable SDN platform that can support various environments (for example, OpenVSwitch-based, npcap-based, etc.) and programming environments, Buzura et al. present in [5] the implementation of an extendable and customizable software solution for mitigating ARP spoofing-based attacks that are used in the data plane layer of the SDN. This solution enables an SDN provider to combine numerous technologies for different architecture components (traffic monitoring, data extraction, data analysis, and decision-making) based on the expertise available by implementing a number of specialized adaptations.

By suppressing ARP broadcast traffic and providing defense against ARP-based assaults to the SDN controller and all network hosts, Munther et al. seek to improve the network [6]. Instead of flooding ARP and DHCP packets, this method will immediately respond to the sender of ARP request packets using the ARP proxy function and forward the DHCP packets to the DHCP server. By observing ARP and DHCP packets, this method creates the updated information tables, which are then used to forward DHCP packets and transmit ARP reply packets. To learn more about packet senders, the multistage security algorithm also uses these tables.

The improved dataset presented by Ahuja et al. [7] includes a strong characteristic set for the identification of ARP attacks. There are two parts to this solution. To construct the dataset, the first unit collects the important features and writes them into a Comma Separated Values (CSV) file. The other unit sets up the machine learning (ML) model and trains it is using the suggested dataset to classify the traffic.

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

ARPVM and ARPC are the two primary components of the approach Tchendji et al. used in this work [8] to detect attackers using a Bayes-Based Security Protocol (E2BaSeP) in both dynamically and statically addressing networks. The interplay of these entities will make it possible to realize the goal, which is the identification of malicious VMs prior to or following IP address reallocations inside the network.

Girdler et al. employed the IDPS in another work based IDPS [9] to locate devices whose MAC addresses had been previously blacklisted using the IDPS Configuration Tool. A packet-in event is transmitted from the OvS switch to the POX controller whenever a packet is received without a flow entry already present. These packet-in events are listened for by the IDPS, a POX extension. Following the event's acceptance, one of four distinct processes—ARP Request Spoofing, ARP Reply Spoofing, ARP Reply Destination Spoofing, and Blacklisted MAC Addresses—handles its processing. Any other packets are ignored, while those that fulfill the defined conditions are logged to the screen and file.

A detection approach for ARP poisoning attacks utilizing a Python script and the SCAPY module was put out by Majumdar et al. [10]. This preventive strategy was proposed and developed for ARP poisoning attacks by implementing static entries in the ARP table. The detection algorithm is based on evaluating the real MAC address and response MAC address of the ARP packet sniffed for any inconsistencies.

Ibrahim et al.'s research [11] aims to inspect each ARP packet in the network in order to identify and block any potential fake ones. An additional module that examines each ARP packet in the network has been added to the SDN controller. The disadvantage of this approach is that it does not recognize ARP flood attacks and only manifests itself as the network grows and traffic increases.

To stop LAN attackers from poisoning the ARP cache tables of other hosts present in the network, the authors Abdel Salam et al. [12] developed a technique that may be used for an application over an SDN POX controller written in Python.

NIDPS was employed by Ramachandran et al. [13] to track all network traffic. Each department in a large organization has its own local area network and is separated into sub-organizations. utilizing distinct NIDPS for each department to lessen the workload on NIDPS. On each computer in the organization, a little agent program called Information Agent (IA) is installed. To uncover spoofing operations, NIDPS talks with these agents.

This research was suggested by Hou X et al. [14] to use Snort, a potent lightweight network intrusion detection system (Network Intrusion Detection System, or NIDS), to identify different types of attacks. In addition to providing real-time notifications, it does protocol analysis, logs packets, searches or matches content, and analyzes traffic in real-time.

Finally, in [15], [16], the proposed solution used the S-ARP protocol as a replacement for the ARP protocol. The S-ARP protocol is a permanent solution to ARP spoofing, but the biggest drawback is that it will make changes to the network stack of all the hosts, which is not as scalable as going for a stack upgrade across all available operating systems, and S-ARP uses the Digital Signature Algorithm (DSA), which causes additional overhead in cryptographic calculations.

3 Problem Statement

In this section, we discuss in detail the different ARP attack scenarios on SDN and the three states that negatively exploit the centrality of the controller. The attacker can take several steps to poison the controller's ARP cache and impersonate other hosts; these scenarios are partitioned into three categories: flooding, spoofing, and broadcasting, which we briefly discussed. In Tab. 1, we showed a data row for every case and connected the scenario to one of the three attack types that we mentioned in the earlier sections. These three species of attack are branched out into five scenarios that we have proposed as follows: in the first scenario, an attacker can spoof several ARP requests and flood them to make the controller busy all the time. The number of packets sent is greater than the predefined threshold, which leads to a lack of service and is called a DDOS attack. This scenario of attack is based on an ARP flooding attack. In the second scenario, an attacker can spoof the ARP request packet information and send it with a fake or not-defined destination IP address to make the controller confused and search for the destination for some time. This scenario of attack is based on an ARP spoofing attack. In the third scenario, the attacker sniffs the sender information by sending a forged source IP address and source MAC address (which are not matched) to distract the controller from searching for this wrong information in the host table. This scenario of attack is based on an ARP spoofing attack. In the fourth scenario, the Ethernet header of the packet is not matched with the source ARP packet information. This scenario of attack is based on an ARP spoofing attack, and in the fifth scenario, the attacker spoofs several ARP replies to packets and broadcasts them. This scenario of attack is based on an ARP broadcasting attack.

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

The attacker exploits these scenarios and tries to send spoofed information advertising that the correct MAC address of the IP addresses of two communication points is the attacker's MAC address to connect to the attacker's machine instead of to each other, and then MITM occurs. Or trying to disable the controller for a period that leads to DDOS.

Table 1: Proposed five attack scenarios

Scenario	*Src_mac*	*Src_ip*	*Target_mac*	*Target_ip*	*Opcode*	*No. of packets*	*Ether_src_address*	*Based attack*
Scen. 1	00:00:00:00:00:01	10.0.0.1	ff:ff:ff:ff:ff:ff	10.0.0.9	1	250	00:00:00:00:00:01	flooding
Scen. 2	00:00:00:00:00:01	10.0.0.1	ff:ff:ff:ff:ff:ff	10.0.0.40	1	1	00:00:00:00:00:01	Spoofing (dst_ip_not found)
Scen. 3	00:00:00:00:00:02	10.0.0.2	ff:ff:ff:ff:ff:ff	10.0.0.3	1	1	00:00:00:00:00:09	Spoofing (Ether! = src_mac)
Scen. 4	00:00:00:00:00:03	10.0.0.1	ff:ff:ff:ff:ff:ff	10.0.0.4	1	1	00:00:00:00:00:03	Spoofing ([src_mac]! = src_ip)
Scen. 5	00:00:00:00:00:05	10.0.0.5	ff:ff:ff:ff:ff:ff	10.0.0.8	2	50	00:00:00:00:00:05	broadcasting

4. Proposed Solution

The goal of our proposed TDA is to reduce the detection and mitigation time of ARP spoofing, ARP flooding and ARP broadcasting attack faster, and therefore we can mitigate the intensity of attack, despite the attacker can use different ways to disturb the network especially restricting the controller, our algorithm succeeded in obstructing those tricks, our contribution is to divide five attack scenarios into three stages of defense for preventing three types of attack, ARP flood attack, ARP spoofing attack and ARP broadcasting attack in a few milliseconds latency and 99.9% accuracy, to implement our solution we developed the Ryu controller with an ideal module that protects the whole network from various attacks, this module acts typically as L3 learning switch module and is backed with new rules, which analyzes the incoming ARP packet only and verifies the authenticity of the sender and target information, we constructed a host table for storing host information which DHCP server assigned automatically to hosts when connected to be able to communicate through the network, if these information in the host table is matched the ARP packet information the switch forward the packet to the designated host, otherwise it does an alarm that there is an attack, and prevents this port from sending or receiving any other packets.

As in algorithm A in Fig. 1, in which packet filtering is made for ARP packets only and other packets can be forwarded as usual, once the ARP packet arrives, it must be analyzed, checking the SHA to see if it is in the blacklisted file. If it is, go to algorithm B; otherwise, call the dimension 1 algorithm. The first channel checks if there is an ARP flooding attack, as in the dimension 1 algorithm, to count the number of ARP packets arriving in the same port. If the number of packets is greater than the predefined threshold, then drop all packets from this port for a time. The second channel to check if there is an ARP spoofing attack, by testing the address validity as showed in dimension 2 algorithm, which checks the source address information (Ip address and the mac address) that must be matched with the host table addresses, the ARP header should be the same as the Ether header, and check the destination address if there is not found, if one of these conditions is true then an attack alarm is displayed so ARP spoofing attack detected, the third channel in dimension 3 algorithm, to confirm that there is no ARP reply broadcasting attack, by checking that ARP reply target mac address not broadcasted (THA != ff:ff:ff:ff:ff:ff), this attack detected if ARP reply is broadcasted, if one result of these three filters is true then the algorithm B in Fig.4 is called to detect the attack, display an alarm, add the mac address of the malicious packet to the blacklisted file, and calculate the detection and mitigation time in milliseconds, else go to algorithm A and sniff the next packet.

As we have discussed, the proposed TDA branched out into five algorithms: algorithm A for packet-in sniffing and analyzing, algorithm B for packet forwarding or dropping if there is an attack, and the three algorithms of dimension (1, 2, 3) for packet filtering, which specifies the type of attack that tries to entrap the network. The detection and

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

prevention of ARP spoofing attacks take a very short time without needing complex cryptographic methods or installing additional software on each device, and we don't need to consume high network resources. We presented our algorithm in the following flowcharts: "Algorithm A flow chart as in Fig. 1, Dimension 1 algorithm, Dimension 2 algorithm, Dimension 3 algorithm, and Algorithm B flow chart as in Fig. 2".

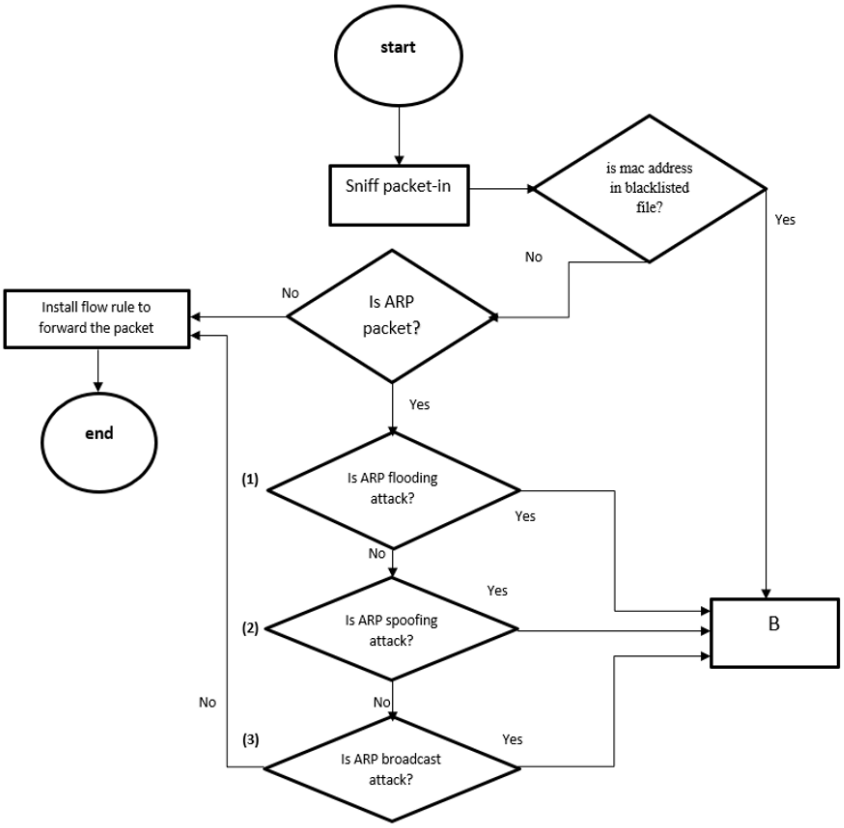


Figure 1: the flowchart of Algorithm (A)

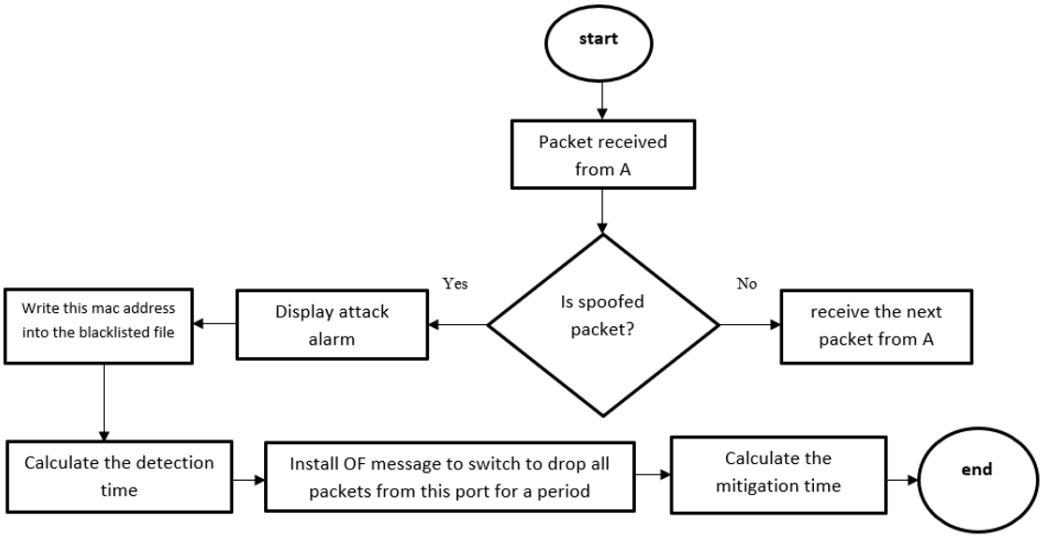


Figure 3: the flowchart of Algorithm (B)

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

Dimension 1 algorithm:

1: **Input:** Packet In
2: **Output:** ARP attack type
3: **Initializing:** host[], port_count[], mac_to_port[], Last=time.time()
4: **If** ARP packet
5: **If** dst in mac_to_port[dpid] **then** outport= mac_to_port[dpid][dst]
6: **If** inport not in port_count[] **then** update port_count[]
7: **Else**
8: **If** port_count[in_port] > 200
9: **then** now=time.time()
10: detection_time= now-last
11: set an alarm that """"ARP flood attack detected"""" in the calculated detection_time.
12: call handle spoof() to drop all packets come from this inport,
13: calculate mitigation_time.
14: **Else if** ARP.opcode==1 **then** port_count+=1

Dimension 2 algorithm:

1: **Input:** Packet In
2: **Output:** ARP attack type
3: **Initializing:** host[], mac_to_port[], ip_set(), Last=time.time()
4: **If** ARP.opcode==1
5: **If** source_mac not in host[] and source_ip not in ip_set() **then** update host[] , ip_set()
6: **Else**
7: **If** source_mac != source_ip
8: **then** calculate detection_time **and** set an alarm that """"fake source address""""
9: call handle spoof() to drop all packets come from this inport.
10: **Else**
11: **If** destination_ip not in host[]
12: **then** calculate detection_time **and** set an alarm that """"destination Ip not found""""
13: call handle spoof() to drop all packets come from this inport.
14: **Else**
15: **If** source_mac != ether_source
16: **then** calculate detection_time **and** set an alarm that """"fake ether address""""
17: call handle spoof() to drop all packets come from this inport.
18: calculate mitigation time.

Dimension 3 algorithm:

1: **Input:** Packet In
2: **Output:** ARP attack type
3: **Initializing:** host[], port_count[], mac_to_port[], Last=time.time()
4: **If** ARP.opcode==2
5: **If** destination_mac=="ff:ff:ff:ff:ff:ff"
6: **Then** calculate detection_time and set an alarm that """"broadcasting attack""""
7: call handle spoof() to drop all packets come from this inport.
8: calculate mitigation_time.

5. Implementation

In this section, we briefly mention the software used to implement TDA and simulate our experiments for all attack scenarios and attack detection and mitigation methods and show the final results we have achieved, consuming very little time without needing any additional hardware. We have five attack scenarios and five flow charts to detect and mitigate these attacks, applied to an SDN topology with depth 2 and fanout 3, and the following subsections contain all the methods and results of implementation.

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

5.1. Environment Setup

For implementing and testing our proposed TDA, we set up the following environments: We used Mininet version 2.2.2, which is a network emulator that creates virtual network hosts, switches, controllers, and links, with 2 GB of RAM and 2 cores, putting Mininet together with other tools on Linux Ubuntu 18.04. This Mininet image is installed on VirtualBox version 6.0.24 on the host machine, which is installed on Windows 10 with 16 GB of RAM and a Core i7 processor. We configured Putty and Xming on Windows for using Linux graphical applications remotely to open Xterms for each host and work on each device separately. Ryu is used as the SDN controller, and Ryu supports various protocols for managing the network devices, such as OpenFlow. Mininet has a virtual testbed that is responsible for creating virtual hosts and switches supported by the OpenFlow protocol. Then the SDN controller connected with the Mininet virtual hosts and the virtual switch, which is provided by OpenVswitch (OVS). The DHCP protocol is utilized for assigning IP-MAC addresses to hosts and registering these associations in the host table we created.

To simulate five modules of attacks, we used the Scapy library and wrote five Python scripts; each script presented exactly the attack scenario, and the controller could detect and prevent all attack scenarios perfectly in a very short time. These attack scripts run with the Mininet command.

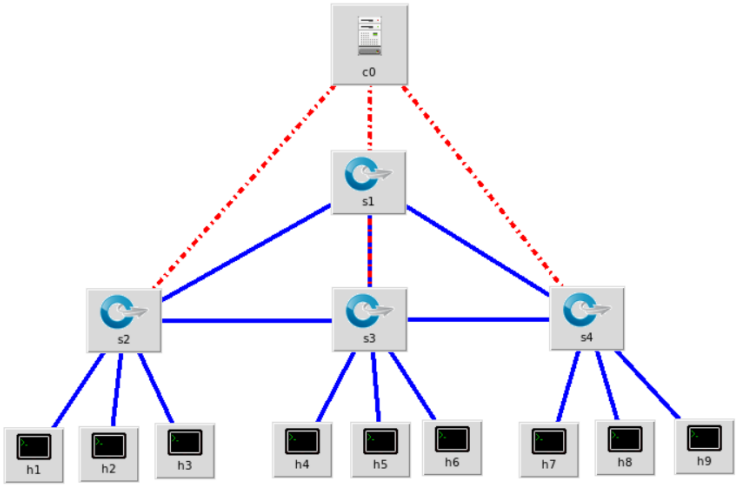


Figure 3: SDN topology (depth=2, fanout=3)

6. Results and Discussion

We experimented with different attack scenarios and applied our solution to detect and mitigate various ARP attacks for evaluating our algorithm results, such as attack detection time and attack mitigation time, for each scenario we discussed in the previous sections, testing every incoming packet and defining if this packet is a request or reply packet, then putting the conditions required to apply our algorithm. In the case of an ARP request packet, first, we observe the number of request packets per port. If the threshold we defined is exceeded, the ARP flooding attack is detected. Second, we analyze the ARP request packet information and apply other conditions to verify that no ARP spoofing attack is detected. In the case of the ARP reply packet, if the destination MAC is broadcast and the source and destination IP are not the same, then the ARP broadcast attack is detected. We show our results and discuss them. In Fig. 3, we have created a tree topology of nine hosts, four switches, and a remote controller. The Ryu controller is chosen for experimentation, and then we could evaluate the efficiency of our proposed detection and mitigation ARP spoofing attack algorithm. This controller is connected to OpenFlow OpenVswitch, which has interacted as a learning switch module. We modified and run this module in the Ryu controller component. This modification makes the open flow switch learn ARP addresses and match all fields in the packet header, therefore installing multiple flows to prevent the attack as much as possible based on the predefined conditions.

6.1 Results of Applying the Proposed TDA to the Five Attack Scenarios

After we finished setting up the environment as in Fig. 3, which has one controller, four switches, and nine hosts, we chose the Ryu to perform our experiments for the accuracy of its performance, and we modified the learning switch

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

module to be a three-dimension filter to prevent three types of ARP attacks. We simulated five different scenarios of the attack, then experimented with our algorithm 20 times for each attack scenario and evaluated its performance. In this section, we showed the results of these tests for each scenario; after that, we recorded 100 tests and documented the detection time and mitigation time for all experiments; after that, we computed the minimum, maximum, and average values for each scenario; and we will show these results of our experiments in the following lines, supported by figures and tables.

Table 2: Samples of simulation results for five attack scenarios.

Scenario number	*Attack type*	*ARP_OP*	*Detection time (ms)*	*Mitigation time (ms)*
1	Flooding attack	Request	0.2	0.6
2	Destination Ip not found (ARP spoofing)	Request	0.9	0.7
3	Fake source address (ARP spoofing)	Request	0.8	0.5
4	Fake Ether header (ARP spoofing)	Request	0.7	0.7
5	Broadcasting attack	Reply	0.7	0.6

We simulated attack scenario 1 using Scapy and run the Python script from the host terminal after establishing the network, which has an IP address equal to 10.0.0.1, and the IP address of the target is 10.0.0.2. When the attacker tries to send numerous ARP request packets to its target, the controller notices that by collecting statistics for each port, if the number of requests is more than the threshold, we defined that an alert has occurred that there is an ARP flood attack. We moderately identified the threshold as 200 requests for a port; this threshold is suitable for large networks to some extent. We provided our results based on two criteria (attack detection time and attack mitigation time) to evaluate our algorithm. We made H1 the attacker. Like in scenario 1 in Tab. 1, the attacker attempts to send many ARP requests, more than the threshold. Each time we tested, the controller succeeded in detecting and mitigating the attack by dropping all packets coming from this port for a predefined time, and the target becomes protected from the attack and cannot be accessed on subsequent attempts within the specified time. The results that we have recorded are the attack detection time, which is the time between when the PACKET_IN arrives at the controller and when the attack alarm is activated, and the attack mitigation time, which is the time interval from when the detection occurred to when the mitigation finished. As shown in Tab. 2, attack detection occurred after 0.2ms and attack mitigation occurred after 0.6ms. These results achieved tremendous speed and high efficiency to mitigate this type of attack. As in Tab. 3, we have computed the minimum, maximum, and average values for ARP attack detection time for this scenario, which are 0.4, 3.3, and 1.37ms, respectively, and the values for ARP attack mitigation are 0.3, 2.7, and 1.17ms for scenario 1. This is considered a very short time to detect and mitigate ARP flooding attacks compared to other proposed solutions, as we mentioned in the following section. And in scenario 2, the attacker as in Tab. 1, which has an IP address equal to 10.0.0.1, attempts to spoof the IP address of the destination, which has an IP address equal to 10.0.0.30, or any other IP address unfound in the network range, to confuse the controller by sending multiple ARP request packets and then disable the network, but the controller alerts in the shortest possible time that the target does not exist and there is an ARP spoofing attack and installs a flow rule to drop all packets coming from this attacker. The detection time of this attack is 0.9ms and the mitigation time is 0.7ms to prevent this type of attack, as shown in Tab. 2. And at Tab. 3, we find the minimum, maximum, and average values for the detection of the ARP spoofing attack in scenario 2. These values are 0.4, 3.6, and 1.16ms, respectively, but for mitigation, these values are only 0.3, 2.9, and 0.67ms. which is also considered a very small delay. The attacker in scenario 3 that we have established using Python scripts with an IP address of the source equal to 10.0.0.1 and a Mac address of the source equal to 00:00:00:00:00:02, which seems like a fake source address, has attempted to send an ARP request packet with a fake source IP address to poison the ARP cache and confuse the controller, hijack a session, or spy on the communication between the original endpoints, but the controller speedily detects the attack and maintains the network security by cutting off contact with the attacker. H1

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

acts as an attacker, as shown in Tab. 1. It sends the spoofed packet, and the controller discovers this attack and drops all packets that come from it. This takes a very short time. As shown in Tab. 2, the detection time is 0.8ms and the mitigation time is 0.5ms. In this scenario, we have calculated the minimum, maximum, and average values in Tab. 3 for the detection of an ARP spoofing attack in scenario 3, and these values are 1.0, 1.4, and 0.47ms only. For mitigation of this type of attack, these results are 0.3, 1.0**,** and 0.55ms. In scenario 4 of attack, as in Tab. 1, the attacker spoofs the ARP request packet by sending forged ether header information to poison the ARP cache. We have simulated this attack with a fake ether information source address of 10.0.0.2 and run the attack from the host terminal H1, and then the network has fallen under the threat of the attacker, but in Tab. 2, our solution has to get past this danger in a short period, and the controller makes an alert that there is an ARP spoofing attack. It takes only 0.7ms for attack detection and 0.7ms for attack mitigation 0.7ms, this is a very short time to obtain these results with such efficiency. In Tab. 3, the minimum, maximum, and average values are 0.3, 2.2, and 1.0ms for detection and 0.3, 2.4, and 0.67ms for mitigation of this attack scenario. We simulated the attack scenario 5 using the Scapy library, and we have a Python script with the IP of the source equal to 10.0.0.1, the IP of the target equal to 10.0.0.3, the mac address of the target equal to ff:ff:ff:ff:ff:ff, and the operational code of the ARP reply. The source then broadcasts several ARP reply to packets. Tab. 1 shows that the attacker tries to broadcast several ARP replies to impede the controller, but the controller can detect and mitigate this type of attack efficiently and quickly. We notice in Tab. 2 that the controller takes only 0.7ms for attack detection and 0.6ms for attack mitigation; this tells us that our algorithm achieved our goals in speed and efficiency. We documented the minimum, maximum, as in Tab. 3, and average values after simulation broadcasting attack scenarios and found these results to be 0.1, 1.2, and 0.66ms**,** and for mitigation, these values are 0.3, 0.7, and 0.7ms in the order we mentioned. That represents too little time.

Table 3: Scenario 1 Times in (MS)

Scenario number	Delay time	Min. time	Max. time	Avg. time
Scenario 1	Attack detection	0.4	3.3	1.37
	Attack mitigation	0.3	2.7	1.17
Scenario 2	Attack detection	0.4	3.6	1.16
	Attack mitigation	0.3	2.6	0.67
Scenario 3	Attack detection	0.1	1.4	0.47
	Attack mitigation	0.3	1	0.55
Scenario 4	Attack detection	0.3	2.2	1
	Attack mitigation	0.3	2.4	0.67
Scenario 5	Attack detection	0.1	1.2	0.6
	Attack mitigation	0.3	0.7	0.7

6.2 Metrics

This paper discussed in the problem statement section five attack scenarios which belong to three attack types, we implemented each scenario separately, after the creation of SDN tree topology with depth 2, fanout equal to 3, and with remote Ryu controller using Mininet, we applied the learning switch modified module of Ryu and carried out five attack python scripts while recording our results, we depended on some important metrics to evaluate our algorithm, we have used Iperf tool to compute bandwidth and throughput and we have used Top command to calculate CPU utilization, these metrics are attack detection time, attack mitigation time, controller throughput, bandwidth, false positive rate, false negative rate, and accuracy. Time delay can be branched to the attack detection time and the attack mitigation time, the attack detection time is the time spent from arriving the malicious Packet-In until the controller detects it, after doing many tests and applying the proposed technique we noticed that our results confirm that is very little time, and the attack mitigation time is the time from detecting the attack to the controller can mitigate it and also this is taking a small fraction of the second as in Tab. 2, we recorded these results by

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

simulating each scenario 20 times and running the three-dimensional algorithm to detect and mitigate the three types of the attack, we founded that each type of attack but rather each scenario has recorded different values of latency, and we will discuss these results of latency for each scenario in this section, we computed the minimum, average and maximum values of latency for each scenario, and documented the final values in Tab. 3. Latency is the time to detect and mitigate the ARP flooding attack. The detection time is the time from the packet's arrival until the controller detects it, and the mitigation time is from detecting the attack until the controller can prevent it. The proposed TDA can take little time to detect and mitigate this type of attack. In scenario 1, the attacker must send several ARP packets greater than the threshold, then the controller compares the number of packets sent from one port with the predefined threshold, sends an alert, and calculates the detection time. After that, the controller sends a new flow rule to the switch to close this port and drop all packets coming from it, and then calculates the mitigation time. ARP traffic can be monitored separately with one controller, which makes detection and mitigation very quick. The proposed solution can detect and mitigate this type of attack for both ARP requests and ARP replies, and the proposed solution can achieve the lowest values of latency compared to other solutions. As in Fig. 4, the detection time ranges from 0.4ms to 3.3ms and the mitigation time from 0.3ms to 2.7ms. The proposed solution can take a short time for detection and mitigation of this type of attack in scenario 2. The attacker must send an ARP request packet containing fake destination address information. The controller then compares the information with the conditions of the three attack types and knows that there is an ARP spoofing attack. The controller sends an alert and calculates the detection time. After that, the controller sends a new flow rule to the switch to close this port and drop all packets coming from it, and then calculates the mitigation time. ARP monitoring traffic with one controller makes detection and mitigation very quick. The proposed solution can detect and mitigate this type of attack for both ARP requests and ARP replies, and the proposed solution can achieve the lowest values of latency compared to the existing solutions. As in Fig. 5, the detection time ranges from 0.4ms to 3.6ms and the mitigation time from 0.3ms to 2.9ms. The proposed solution can take little time for detection and mitigation of this type of attack in scenario 3. The attacker must send an ARP request packet with fake source address information, and then the controller compares the information with the conditions of the three attack types and determines that there is an ARP spoofing attack. The controller makes an alert and calculates the detection time. After that, the controller sends a new flow rule to the switch to close this port and drop all packets coming from it, and then calculates the mitigation time. ARP traffic monitoring with one controller makes detection and mitigation possible in very little time; the proposed TDA can detect and mitigate this type of attack for both ARP requests and ARP replies and can achieve the lowest values of latency compared to the existing solutions. As in Fig. 6, the detection time ranges from 0.1ms to 1.4ms and the mitigation time from 0.3ms to 1ms. The proposed solution can take little time for detection and mitigation of this type of attack in scenario 4. The attacker must send an ARP request packet with fake Ether address information, then the controller compares the information with the addresses in the IP-to-MAC table and finds that there is an ARP spoofing attack and issues alerts. And calculates the detection time. After that, the controller sends a new flow rule to the switch to close this port and drop all packets that come from it, then calculates the mitigation time. ARP monitoring traffic with one controller makes detection and mitigation happen in a very short time. TDA solution can detect and mitigate this type of attack for both ARP requests and ARP replies, and TDA can achieve the lowest values of latency compared to other solutions. As in Fig. 7, the detection time ranges from 0.3ms to 2.2ms and the mitigation time from 0.3ms to 2.4ms. The proposed solution can take little time for detection and mitigation of this type of attack in scenario 5. The attacker must broadcast many ARP reply to packets, which the controller analyzes and extracts some information from. If the packet is an ARP reply and the packet was broadcast, the controller sends an alert and calculates the detection time; after that, the controller sends a new flow rule to the switch to close this port and drop all packets coming from this port, then calculates the mitigation time. ARP traffic can be monitored separately with one controller, which makes detection and mitigation very quick. The proposed solution can detect and mitigate this type of attack for both ARP requests and ARP replies, and the proposed solution can achieve the lowest values of latency compared to other solutions. As in Fig. 8, the detection time ranges from 0.1ms to 1.2ms, and the mitigation time from 0.3ms to 0.7ms.

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

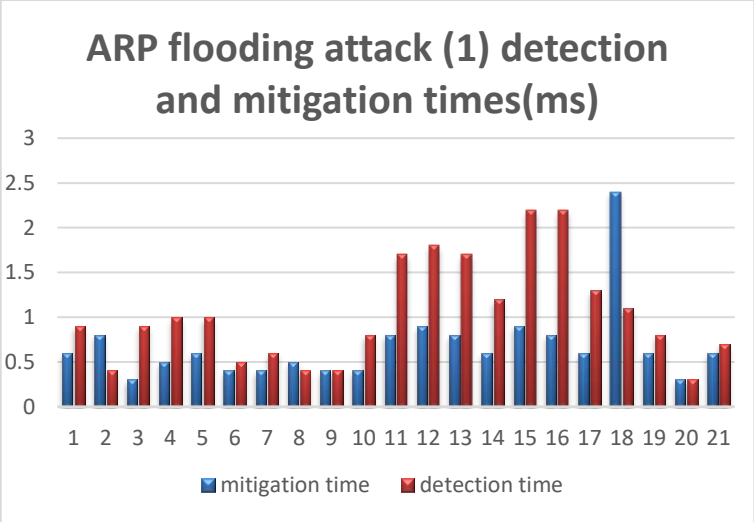


Figure 4: Scenario (1) time delay

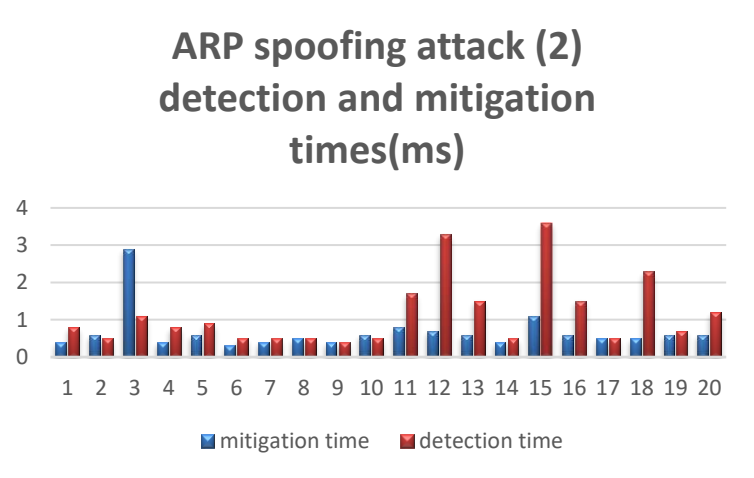


Figure 5: Scenario (2) time delay

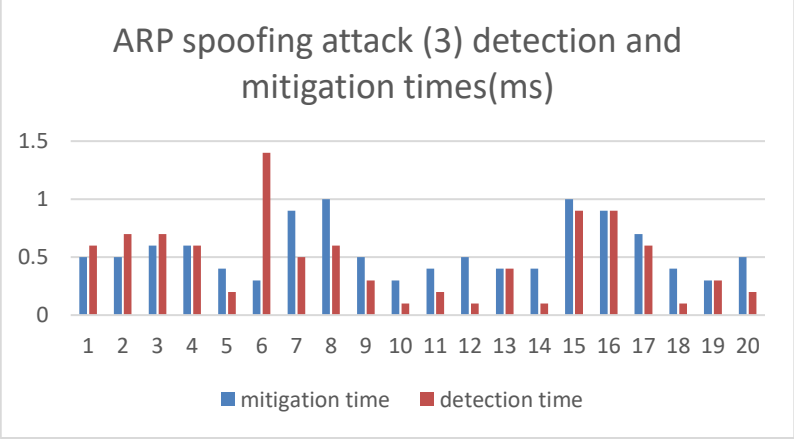


Figure 6: Scenario (3) time delay

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

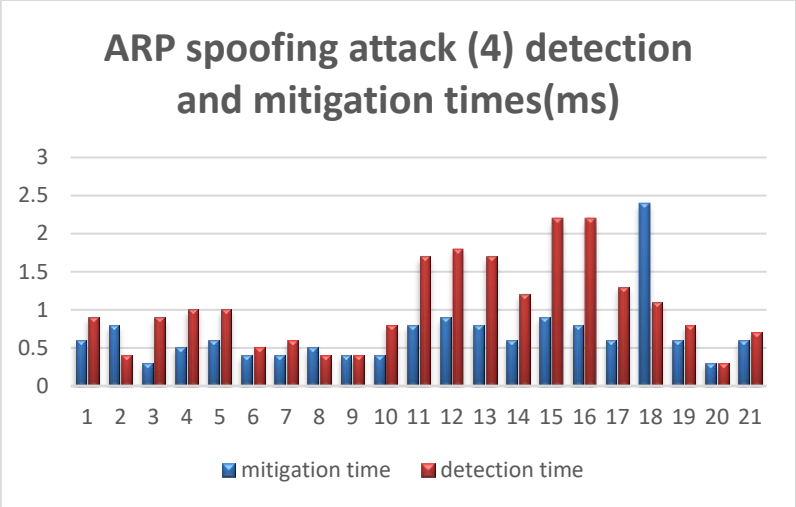


Figure 7: Scenario (4) time delay

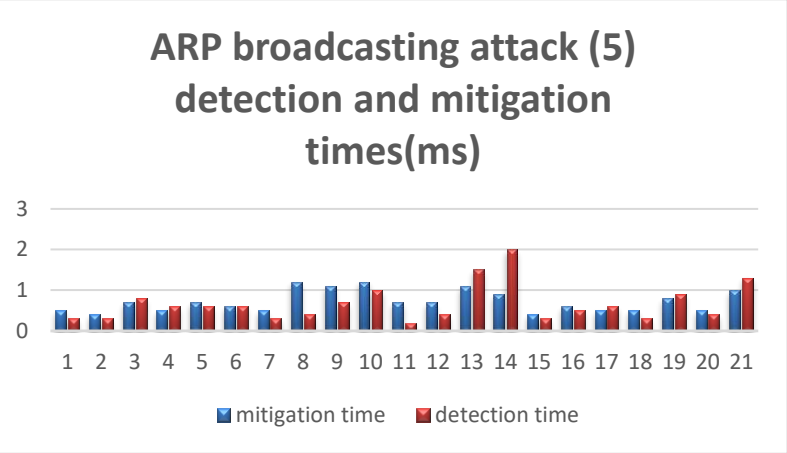


Figure 8: Scenario (5) time delay

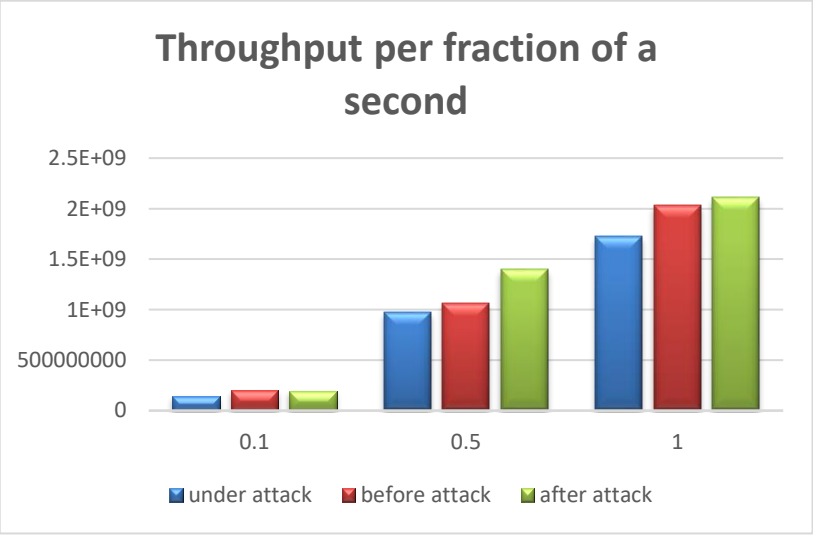
There are other metrics we measured to evaluate the network and controller performance, these are sensitivity (TPR), specificity (TNR), false positive ratio (FPR), false negative ratio (FNR), and the accuracy which is considering the main metric to evaluate the performance, the accuracy of our solution is equal to 99.9% after many experiments, we computed it as in Eq. (1).

$$\textbf{\textit{Accuracy}}=(TP+TN)/(TP+TN+FP+FN) \quad (1)$$

Controller throughput is an important metric to evaluate the network's performance, especially the controller's throughput, which is a practical metric that measures actual packet delivery. We calculated the controller throughput using the Iperf tool, which is the amount of data over time. As in Fig. 9, in the case of an under attack, after applying our algorithm to the five attack scenarios 20 times, we recorded the value of the throughput for three cases, which are before, under, and after the attack, and we noticed that the controller throughput recorded the lowest change in the case of an under attack compared to its behavior in the cases of before and after the attack. Bandwidth is considered the main measurement that evaluates controller performance; bandwidth is a theoretical metric measuring potential packet delivery. We calculated the controller bandwidth using the Iperf tool, and after applying the proposed TDA to the five attack scenarios 20 times, we took the results at different fractions of a second and recorded the value of the bandwidth for the three cases before, under, and after the attack. These results tell us that the bandwidth recorded from 18 GB/sec to 17.7 GB/sec in the cases before and after the attack and 16.5 GB/sec in the case under the attack, as shown in Fig. 10. These results indicate that the bandwidth recorded the lowest rate in the case of under the attack. CPU utilization is an important metric to monitor because high CPU usage can

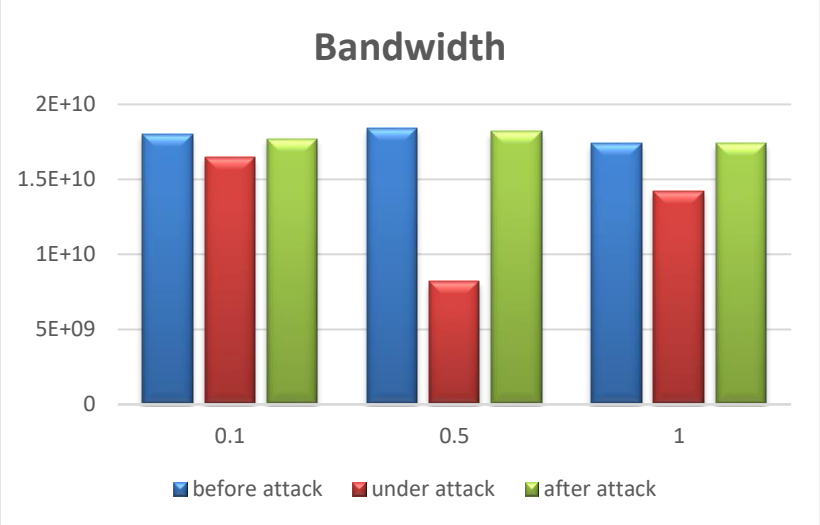
Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

negatively impact the performance of a system. CPU utilization is the amount of work a CPU does to perform the tasks given to it. For calculating the CPU utilization, we used the top command. After doing our experiments 20 times, we tracked the results of CPU utilization over the three cases, which are before, under, and after, and we have found that, as shown in Fig. 11, the behavior of the CPU is highly changed during the attack and may reach 30.4%, while in other cases it may behave normally with a minimum of precision and achieve from 0.3% to 2.6%.



Time (second)

Figure 9: Throughput char



Time (second)

Figure 10: Bandwidth chart

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

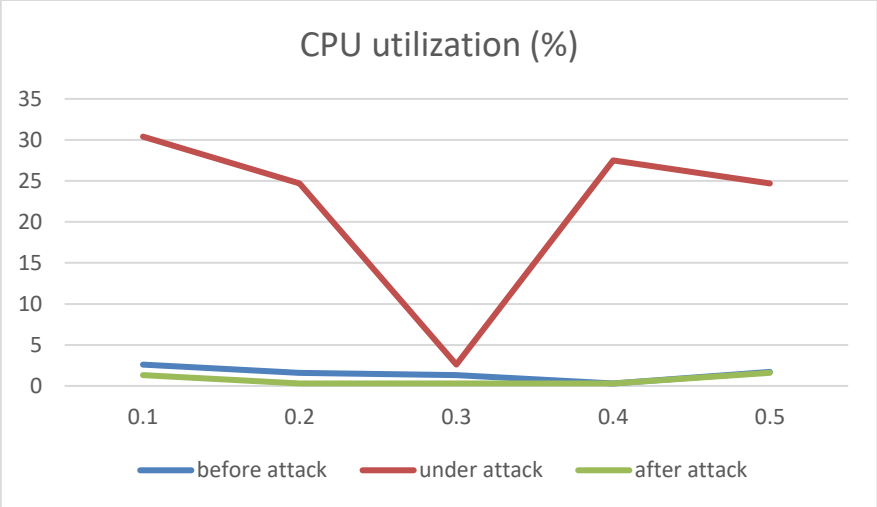


Figure 11: CPU utilization line chart

6.3 Proposed TDA Vs Existing Solutions

In this subsection, we compare our proposed solution with other previous solutions based on more than one important measurement, such as delay time and accuracy, and compare TDA results with the existing solutions to defend the three types of attack. While comparing the delay time of the proposed TDA with other solutions, we can notice a largely different result. Our proposed solution has a minimum delay time of 0.1m**s** to 3.6ms, but other solutions could not achieve these results, like [9], which achieved a delay from 02.024589sec to 03.933365sec only, and [4], which achieved a delay from 0.8ms to 6ms. And accuracy which is considered the main measurement to evaluate performance; our experiments achieved the highest accuracy of 99.9%, but other solutions like [7] have achieved only 99.73%. TDA defined three dimensions of ARP security issues and achieved 99.9% accuracy, but the existing solutions only addressed two or one of these issues. The three attack types are documented in Table 8. Finally, we noticed that our technique is much faster, simpler, and more reliable than other solutions and is inclusive of all three types of attacks, whereas other authors did not take these three types into account. And based on actual outcomes, we can state that our strategy was effective in producing pleasing outcomes.

Table 8: Proposed work vs existing works.

Authors	ARP flooding attack	ARP spoofing attack	ARP broadcasting attack
Wei H et al. [17]	√		
Tchendji V et al. [8]		√	√
Ibrahim H et al. [18]		√	√
Aldabbas H et al.[19]	√	√	
Ahuja N et al. [7]			
Proposed TDA	√	√	√

7. Conclusion

Because of the importance of SDN and being significantly more adaptable, with agility, programmability, centralized management, and open connectivity, administrators can manage the network from a centralized controller, alter configuration options, allocate resources, and boost network capacity without putting in more hardware. SDN also offers complete network visibility, which results in stronger network security compared to traditional networks. However, because of its centralized architecture, SDN security continues to be a significant concern. Thus, in this paper, we focused on SDN security issues, especially ARP attack types, and proposed very

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

effective solutions to detect and mitigate three species of ARP attacks: ARP spoofing attacks, ARP flooding attacks, and ARP broadcasting attacks, which are the cause of other types of attacks such as MITM and DDOS attacks. TDA simulated five scenarios of the three types of attack using the Scapy library to create five attack modules and a three-dimensional module work as a filter to defend these three main types of ARP attacks, and the proposed solution efficiently defends all five scenarios of attack using a simple and efficient three-dimensional algorithm by passing the ARP packet into three filters to prevent the occurrence of these three types of attack. The proposed solution avoids the high cost of installing additional software or hardware and works effectively and efficiently with little overhead and without changing the network infrastructure or installing additional hardware, depending on the properties of SDN architecture, especially centralizing the controller, which means the controller has a global view of the overall network, which in turn simplifies monitoring, controlling, and configuring data plane devices. In comparison with other solutions, our solution could successfully prevent three types of attacks that use ARP vulnerabilities more accurately and with a minimum time delay. Future work may expand to use multiple controllers that need to be managed dynamically to ensure global connectivity. To overcome some limitations in future work, static ARP address allocation and dynamic ARP address allocation must be used together, and the proposed solution must be tested on a large data set and set up in a real environment to get a more authorized network. However, the central controller is the most vulnerable part of the SDN, and future work must take into consideration the vulnerabilities of other planes to achieve network consistency. Finally, SDN is a very important network that must be taken care of for issues like security.

References

[1] J. Xia, Z. Cai, G. Hu, and M. Xu, "An active defense solution for ARP spoofing in OpenFlow network," *Chinese Journal of Electronics*, vol. 28, no. 1, pp. 172–178, 2019.

[2] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015.

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

[4] 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 *2020 International Conference on Computer Science and Software Engineering (CSASE)*, IEEE, 2020, pp. 13–19.

[5] S. Buzura, M. Lehene, B. Iancu, and V. Dadarlat, "An Extendable Software Architecture for Mitigating ARP Spoofing-Based Attacks in SDN Data Plane Layer," *Electronics (Basel)*, vol. 11, no. 13, p. 1965, 2022.

[6] M. N. Munther, F. Hashim, N. A. A. Latiff, K. A. Alezabi, and J. T. Liew, "Scalable and secure SDN based ethernet architecture by suppressing broadcast traffic," *Egyptian Informatics Journal*, vol. 23, no. 1, pp. 113–126, 2022.

[7] N. Ahuja, G. Singal, D. Mukhopadhyay, and A. Nehra, "Ascertain the efficient machine learning approach to detect different ARP attacks," *Computers and Electrical Engineering*, vol. 99, p. 107757, 2022.

[8] V. K. Tchendji, F. Mvah, C. T. Djamegni, and Y. F. Yankam, "E2BaSeP: Efficient Bayes based security protocol against ARP spoofing attacks in SDN architectures," *Journal of Hardware and Systems Security*, vol. 5, no. 1, pp. 58–74, 2021.

[9] 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," *Computers & Electrical Engineering*, vol. 90, p. 106990, 2021.

[10] A. Majumdar, S. Raj, and T. Subbulakshmi, "ARP poisoning detection and prevention using Scapy," in *Journal of Physics: Conference Series*, IOP Publishing, 2021, p. 012022.

[11] 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 *2020 International Conference on Computer Science and Software Engineering (CSASE)*, IEEE, 2020, pp. 13–19.

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023

[12] A. M. AbdelSalam, A. B. El-Sisi, and V. Reddy, "Mitigating ARP spoofing attacks in software-defined networks," in *2015 25th International Conference on Computer Theory and Applications (ICCTA)*, IEEE, 2015, pp. 126–131.

[13] S. G. Bhirud and V. Katkar, "Light weight approach for IP-ARP spoofing detection and prevention," in *2011 Second Asian Himalayas International Conference on Internet (AH-ICI)*, IEEE, 2011, pp. 1–5.

[14] X. Hou, Z. Jiang, and X. Tian, "The detection and prevention for ARP Spoofing based on Snort," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, IEEE, 2010, pp. V5-137.

[15] V. Ramachandran and S. Nandi, "Detecting ARP spoofing: An active technique," in *International conference on information systems security*, Springer, 2005, pp. 239–250.

[16] D. Bruschi, A. Ornaghi, and E. Rosti, "S-ARP: a secure address resolution protocol," in *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, IEEE, 2003, pp. 66–74.

[17] H.-C. Wei, Y.-H. Tung, and C.-M. Yu, "Counteracting UDP flooding attacks in SDN," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, IEEE, 2016, pp. 367–371.

[18] 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 *2020 International Conference on Computer Science and Software Engineering (CSASE)*, IEEE, 2020, pp. 13–19.

[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, 2021.

Doi: https://doi.org/10.54216/FPA.120101
Received: January 03, 2023 Revised: April 02, 2023 Accepted: June 01, 2023