Egyptian Informatics Journal 23 (2022) 113–126

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Scalable and secure SDN based ethernet architecture by suppressing broadcast traffic

Munther Numan Munther [a], Fazirulhisyam Hashim [a,*], Nurul Adilah Abdul Latiff [b], Kamal Ali Alezabi [c], Jiun Terng Liew [a]

[a] Department of Computer and Communication Systems Engineering, Faculty of Engineering, Universiti Putra Malaysia, Selangor, Malaysia
[b] School of Ocean Engineering, Universiti Malaysia Terengganu, Terengganu, Malaysia
[c] Institute of Computer Science and Digital Innovation (ICSDI), UCSI University, Kuala Lumpur, Malaysia

ARTICLE INFO

Article history:
Received 24 December 2020
Revised 22 March 2021
Accepted 11 August 2021
Available online 28 August 2021

Keywords:
Software-defined network (SDN)
Ethernet scalability
Address Resolution Protocol (ARP)
Dynamic host configuration protocol (DHCP)
ARP storm
Spoofing attack

ABSTRACT

Ethernet is one of the widespread protocols residing in the second layer of the seven-layers Open Systems Interconnection (OSI) model. Ethernet offers various advantages which enable its widespread use in all types of network topology and becomes an essential part of computer and network architecture. Despite its features, Ethernet suffers from scalability issues where the increasing number of hosts in a single broadcast domain will significantly expand the broadcast traffic in the network. Since the emergence of software-defined networking (SDN), researchers exploited various attractive features of SDN to suppress the broadcast traffic. Although capable in addressing the scalability issue of Ethernet, the existing SDN based solutions are lacking of security mechanism, which may expose the network to various ARP based attacks. Owing to this issue, this paper proposes a floodless and secure mechanism to suppress broadcast traffic. In general, the proposed solution utilizes SDN architecture and accommodates a multistage security algorithm. The multistage security algorithm consists of three stages; each stage incorporates specific analysis to identify the packet status or behavior, and react accordingly based on its status. To demonstrate the efficiency of the proposed solution, several ARP based attack scenarios are generated and evaluated using Mininet emulator. The performance evaluation indicates that the true positive ratio for attack detection in the proposed solution is 57.14% for the first stage, 66.66% for the second stage, and in some cases may achieve 100% for the final stage.

1. Introduction

Over 30 years ago, the Ethernet protocol was designed to provide flexibility, low cost, and user-friendly solution to network user [1]. However, the tremendous development of technologies and the increasing numbers of hosts in networks have exposed the vulnerabilities of Ethernet. Scalability is considered one of the most prominent problems in Ethernet networks. It occurs due to the growing number of hosts in the network, which eventually leads to the increase of exchanged broadcast traffic between the hosts. The Address Resolution Protocol (ARP) [2] is one of many protocols utilized by network host to complete their communication process. It operates between Data Link and Network Layers of the OSI layer model. ARP is the source of enormous broadcast traffic in Ethernet networks, and thereby it is regarded as the main reason for the Ethernet scalability issue. Several studies have tried to address the scalability issue by suppressing the ARP broadcast traffic. One notable approach is by using the new Software Defined Networking (SDN) paradigm. SDN offers flexible network management by decoupling control and data planes, and allows complex algorithms development [3].

In general, the existing solutions relied on constructing ARP proxy [4] features inside the SDN controller to suppress ARP broadcast traffic. While succeeded in suppressing the ARP broadcast traffic, the researchers had overlooked the security aspects. The addition of the ARP proxy feature inside the SDN controller will

* Corresponding author.
E-mail address: fazirul@upm.edu.my (F. Hashim).
Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.eij.2021.08.001
1110-8665/

open a serious security vulnerability that attackers can exploit. In particular, ARP suffers from a lack of authentication mechanism. Moreover, recent studies have shown the security vulnerabilities of ARP, notably, ARP spoofing and ARP storm [5]. In light of the aforementioned issue, this paper proposes a floodless and secure mechanism that is capable of suppressing the broadcast traffic while at the same time protecting the SDN network from ARP based attacks. The proposed solution incorporates some basic principles of recent studies and appends a novel multistage security algorithm. The security algorithm consists of three security stages that examine the behaviour of the incoming ARP packets. The first stage inspects the source IP address, while the second stage analyzes the source MAC address. Finally, the destination IP address is examined in the third stage. In general, this paper aimed to enhance the network by suppressing ARP broadcast traffic and offering protection to the SDN controller and all network hosts from ARP-based attacks. Furthermore, it is worth highlighting that while ARP can be applied in various networks, for the sake of discussion and simulation, this paper focuses on the Ethernet networks.

The rest of the paper organized as follow. Section 2 highlights the related works. The proposed solution discussed in detail in Section 3. Section 4 presents the results and discussions of the simulation experiments, followed by concluding remarks.

2. Related works

Many researchers have dealt with treating and improving the scalability of Ethernet networks. Thus, we classified the existing solutions into three types based on their proposed methods.

The first method, such as Etherproxy [6], and FCSEA [7], have suggested adding a dedicated device to the network. In principle, this new device is responsible for intercepting and analyzing the ARP broadcast traffic and recording useful information (i.e., from the analyzed traffic) in its local cache table. The analysis conducted by the new device is then used to directly reply to ARP broadcast traffic, thus reducing the number of broadcast traffic in the network. Nevertheless, adding a new device to the network may not be a feasible solution and can be regarded as troublesome for some networks.

On the other hand, the second method, such as SEATTLE [8], MOOSE [9], and ROOM [10] relied on a new network architecture for eliminating broadcast traffic in Ethernet networks. In general, the main idea is to eliminate the broadcast traffic by making the network switches recognizing the packet destination without flooding it inside the network. This can be achieved by utilizing information in hash tables inside the network switches and hierarchical MAC addresses. However, the new network architecture is inconvertible and not compatible with some existing protocols.

The third approach is the use of SDN architecture to suppress ARP broadcast traffic in Ethernet networks. In general, the network information can be collected through snooping on ARP and DHCP packets by OpenFlow switches or running discovery protocols such as LLDP [11]. Therefore, the controller has knowledge of all devices in its network.

Within the SDN-based solutions, we can further categorize them into two approaches. The first approach was based on enabling the ARP proxy feature in the central controller. For instance, in [12], the proposed approach aimed to allow the ARP proxy feature inside the central controller and forward all ARP and DHCP requests to the controller. The controller will reply to the received requests based on the network information stored inside hash tables via the ARP proxy feature. Moreover, this approach has inspired subsequent works [13–15] that used the same concept, which employs the centralized ARP proxy to suppress broadcast traffic.

Meanwhile, the second approach was based on flow rules inside OpenFlow switches to process ARP broadcast traffic. For instance, in [16], the OpenFlow switches forwarded the broadcast traffic through the appropriate port without flooding it inside the network. Likewise, [17] proposed the same concept with different topology, where a tree topology has been applied. In [18], the OpenFlow switches redirected ARP broadcast traffic to a dedicated ARP server. Meanwhile, in [19], installation of a flow rule has been applied for each IP; therefore, the OpenFlow switches can forward ARP broadcast traffic. A slightly different principle, installation of a flow rule for the most frequent IP addresses has been applied in [20]. Another concept has been proposed in [21], where cache port-based OpenFlow switches have been applied. In this method, the OpenFlow switches cache the frequent IP address for a specific period.

However, the aforementioned approaches allow a potential attacker to have direct contact with the SDN controller or ARP server, which makes the network susceptible to ARP-based attacks. Besides, none of the existing works can address both the scalability and security issues related to ARP traffic. Even though the work in [18] has expressed concern about the security of the network, lack of explanation and details of the security aspect have been presented by the authors. In light of this issue, our paper proposes an efficient and integrated solution to address both scalability and security aspects.

3. The proposed solution

The proposed solution is envisaged in providing a scalable and secure Ethernet network. In order to achieve this vision, the proposed approach, (i) suppresses the broadcast traffic to offer scalable networks, and (ii) utilizes multistage security algorithms to address ARP based attacks. As mentioned earlier, ARP and DHCP protocols rely on broadcast traffic; therefore, the proposed approach will not allow the OpenFlow switches to flood ARP and DHCP packets within the network. Rather than flooding ARP and DHCP packets, the proposed approach will reply directly to ARP request packets' sender through the ARP proxy feature and forward the DHCP packets to the DHCP server. The proposed approach builds the updated information tables by monitoring ARP and DHCP packets. The proposed approach relies on these tables to forward DHCP packets and send ARP reply packets. Moreover, the multistage security algorithm also relies on these tables to get information about packet senders. In principle, the attackers or malicious hosts can exploit ARP weakness to attack the SDN controller or other hosts. Particularly, the OpenFlow switches will forward the ARP packets directly to the SDN controller. Therefore, the multistage security algorithm is suggested for protecting the SDN controller and network hosts from ARP based attacks. Fig. 1 illustrates the basic diagram of the proposed approach. The incoming packet from OpenFlow Switch cannot reach to SDN controller application and tables unless it passes through the multistage security check to ensure the integrity of information and network devices.

3.1. Information collecting method

Generally, to respond to broadcast traffic, the SDN controller tries to obtain the necessary information from network devices. Therefore, two hash tables constructed, which are Switch-info and Host-info tables. The method steps as depicted in Fig. 2, where the SDN controller sends two OpenFlow messages to all OpenFlow switches. The first message is the OpenFlow Discover State Request

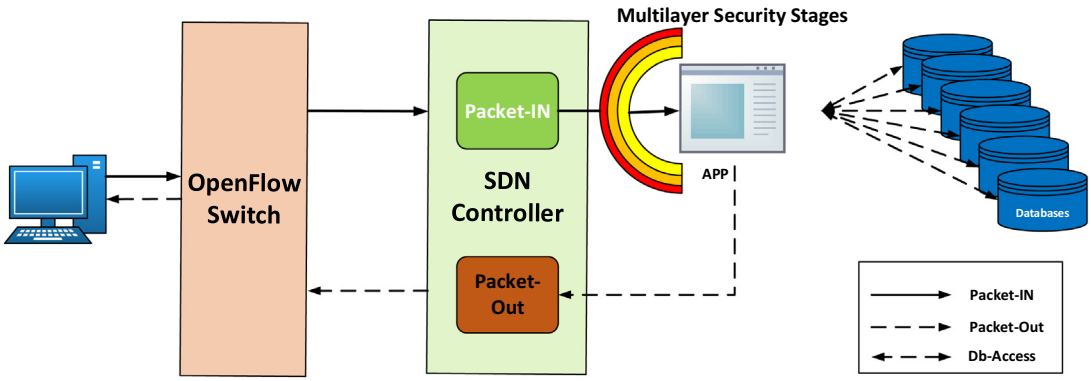


Fig. 1. The basic diagram of the proposed approach.

Message, while the second message is the OpenFlow Flow Mode Message. Through the first message, the SDN controller collects the OpenFlow switches status information. While by the second message, the SDN controller requests OpenFlow switches to forward ARP and DHCP packets directly to the SDN controller without flooding them in the network. Whenever the OpenFlow switch receives messages from the SDN controller, it first sends a reply message to the SDN controller called OpenFlow State Reply Message. Next, the OpenFlow switches enable a flow rule to directly forward ARP and DHCP packets to the SDN controller. Hence, ARP and DHCP packets forwarded to the SDN controller when a host sends any one of these packets. Consequently, through these procedures, the SDN controller can collect the network's information and construct hash tables. Moreover, to ensure hash tables' accuracy, the method monitors DHCP release packets and listens to OF Port Status Messages. Upon receiving one of them, a host removes from hash tables because these messages indicate that a host has left the network.

3.2. Floodless broadcast traffic method

The SDN controller responds directly to broadcast traffic based on the hash tables. In principle, when a host sends ARP or DHCP broadcast traffic, the OpenFlow switch forwards this traffic to the SDN controller depended on the SDN instructions. So, the SDN controller analyzes the received broadcast packets to know how can answer them. After that, the SDN controller uses ARP Proxy APP and hash tables to directly respond to ARP broadcast traffic. Furthermore, the SDN controller forwards the DHCP broadcast traffic to the DHCP server. Fig. 3 illustrates the floodless broadcast traffic method in the SDN networks.

3.3. ARP based attacks detection and mitigation algorithm

ARP based attack detection and mitigation algorithm is a multi-stage security algorithm that combines three checks to detect and mitigate ARP based attacks. In principle, ARP based attacks (APR Storm or ARP Spoofing) can occur simultaneously or separately. The attacker tries to change ARP packet information to generate one of the ARP based Attacks.

The ARP based attack detection and mitigation algorithm consist of three stages, as illustrates in Fig. 4. The first stage inspects the source MAC address, while the source IP address analysis occurs at the second stage. Finally, the third stage examines the destination IP address for incoming ARP packets.

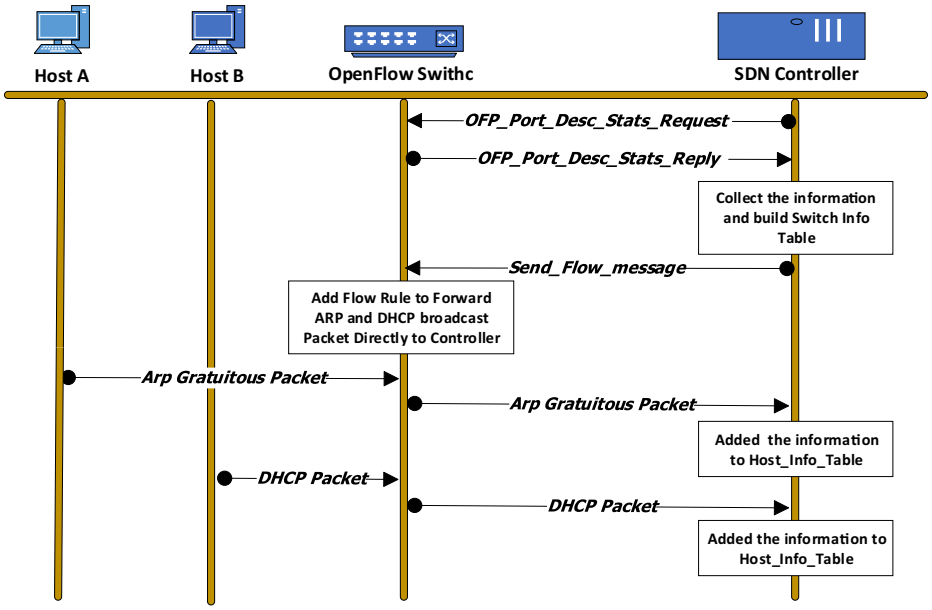


Fig. 2. The exchanged messages between SDN control and OpenFlow switches.

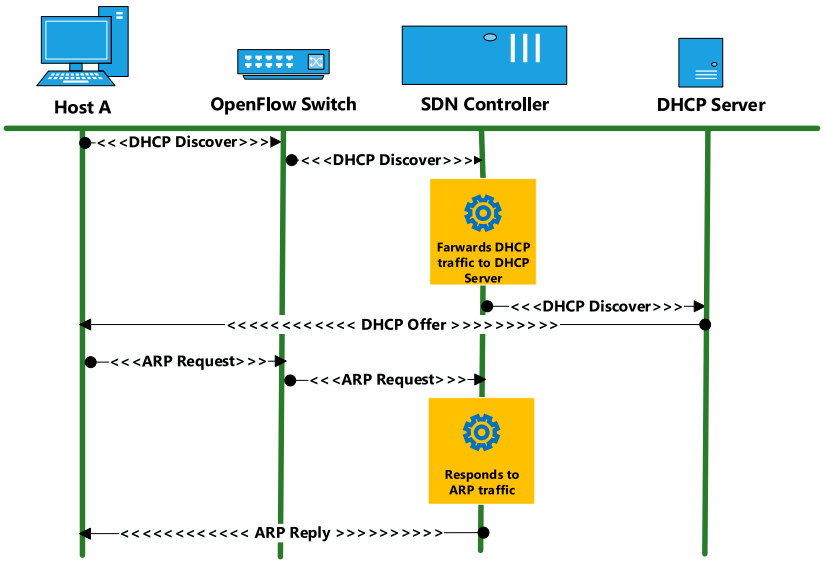


Fig. 3. Floodless broadcast traffic method.

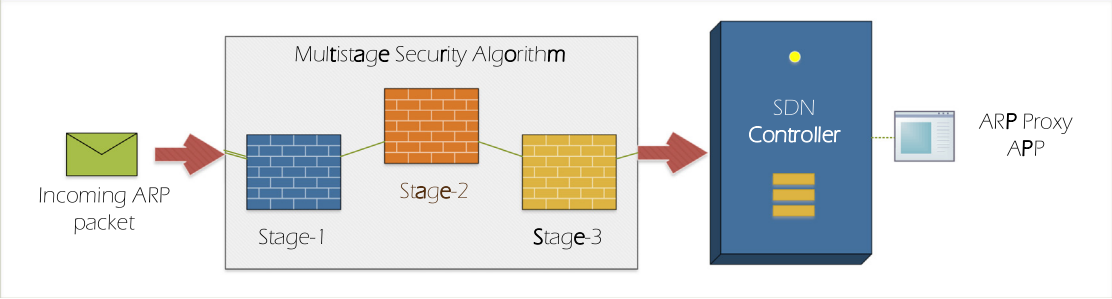


Fig. 4. Multistage security algorithm.

The algorithm compares and analyzes the ARP packet information with the hashes tables built by the information collecting method in each stage. The multistage security algorithm is illustrated in Fig. 5.

3.3.1. ARP storm detection and mitigation

The ARP storm attack principle involved sending a large number of ARP request packets to flood the network. The attacker uses only ARP request due to ARP request's broadcast nature, while ARP reply generates unicast traffic. All the possible conditions of the ARP storm attack are illustrated in Table 1.

As depicted in Table 1, the attacker can send ARP request packets in several formats. Scenarios No. 1 to 6 illustrate cases of sending forged request packets that contain wrong information in the source address fields (IP address or MAC address). These cases can be detected in the first or second stage of the detection algorithm. Scenario No. 7 is considered more sophisticated, where the attacker sends an ARP request with the wrong destination address. Therefore, it's hard to distinguish between the received requests (i.e., maybe a regular host who has made a mistake in the destination address for some reason). Thus, the third stage of the detection algorithm is relied on the threshold value to detect an attacker (Scenario No. 7).

The possibility of detecting the ARP storm attacks is clarified in Table 2. Depending on Table 2, it is possible to derive the following formula, representing the correct state when the incoming packet will pass through all stages to reach the SDN controller.

$$F = A \wedge B \wedge C \tag{1}$$

where, A is Source MAC Address, B is Source IP address, and C is Target Destination IP.

Meanwhile, the mitigation part tries to protect the SDN controller and hosts from the impact of the ARP storm attack. To achieve that, two hash tables are created, namely the ARP attackers and the potential attackers. Then, the information of attackers and suspect hosts (i.e., IP address, MAC address, Port. No, and Switch. ID) is added respectively on these tables. The records of these tables will be erased according to the predefined periods, i.e., 30 min for the ARP attackers and 1 min for the potential attackers, respectively (discusses in the upcoming section). Hence, the proposed approach will contain all hosts' information (normal, suspect, and an attacker).

Consequently, the proposed approach checks the Ethernet source address and destination IP address for the received ARP packet. Generally, if any field has forge information, the proposed approach will check the Ethernet source address with both hash tables (ARP Attackers and Potential Attackers). In case the Ethernet source address is found in the potential attackers' table, the number of attempts will check. However, if the number of attempts exceeds the potential threshold value, the Ethernet source address will be added to the ARP attackers table. Moreover, the OpenFlow flow mode message is created to send to the OpenFlow switch. For other cases, the algorithm adds the Ethernet source address to the potential attackers' table and increase the number of attempts by

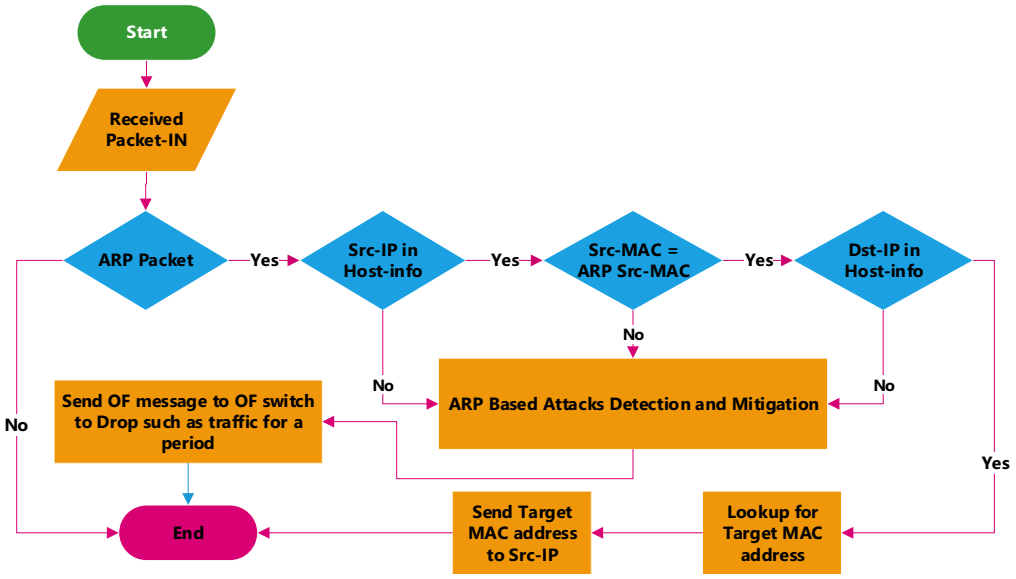


Fig. 5. Flowchart of multistage security algorithm.

Table 1
Possible scenario of ARP storm packet attack.

Case No.	Source MAC Address	Source IP Address	Target Dest. IP	Target Dest. MAC*	Status	Description of the situation
1	0	0	0	–	ARP packet has incorrect information in Source MAC, Source IP, and Target Destination IP.	Malicious packet – it is unlikely that a normal host sends ARP packet with incorrect source and destination addresses.
2	0	0	1	–	ARP packet has incorrect information in Source MAC address and Source IP address.	Malicious packet – it is unlikely that a normal host sends ARP packet with incorrect source address.
3	0	1	0	–	ARP packet have incorrect information inthe Source MAC address and Destination IP address.	Malicious packet – it is unlikely that a normal host sends ARP packet with incorrect source and destination addresses.
4	0	1	1	–	ARP packet has incorrect information in the Source MAC address	Malicious packet – it is unlikely a normal host sends ARP packet with incorrect source MAC address.
5	1	0	0	–	ARP packet has incorrect information in the Source IP address and Destination IP address	Malicious packet – it is unlikely that a normal host sends ARP packet with wrong source IP address and destination addresses.
6	1	0	1	–	ARP packet has incorrect information in the Source IP address	Malicious packet – it is unlikely that a normal host sends ARP packet with incorrect source IP address.
7	1	1	0	–	ARP packet has incorrect information in Destination IP address	Maybe a legitimate or malicious packet, therefore the sender is considered as a potential attacker.
8	1	1	1	–	Normal ARP Request	Legitimate ARP Request

* This field is left blank because it will carry the MAC address information of the destination host.

Table 2
The possibility of detecting ARP storm packet.

Case No.	Source MAC Address	Source IP Address	Target Destination IP	Target Destination MAC	Status
1	0	0	0	–	It is detected as an attempt to set up an attack at the first stage, due to containing the wrong source MAC address.
2	0	0	1	–	
3	0	1	0	–	
4	0	1	1	–	
5	1	0	0	–	It is detected as an attempt to set up an attack at the second stage, due to containing the wrong source IP address.
6	1	0	1	–	
7	1	1	0	–	They are considered as potential cases of attack at the third stage, due to containing the wrong target destination MAC address.
8	1	1	1	–	Normal ARP Request

1. Fig. 6 illustrated the ARP based attack detection and mitigation algorithm.

3.3.2. ARP Spoofing Detection and Mitigation

The ARP spoofing detection and mitigation method are proposed to provide full protection to the SDN networks from ARP spoofing attacks. However, the attacker may send an ARP packet with forged information to attack hosts or the SDN controller. Therefore, two stages of inspection applied; both stages are depending on the comparison of ARP packets information with the hash tables. The proposed stages are part of the ARP storm detection algorithm. As previously discussed, the first stage checks

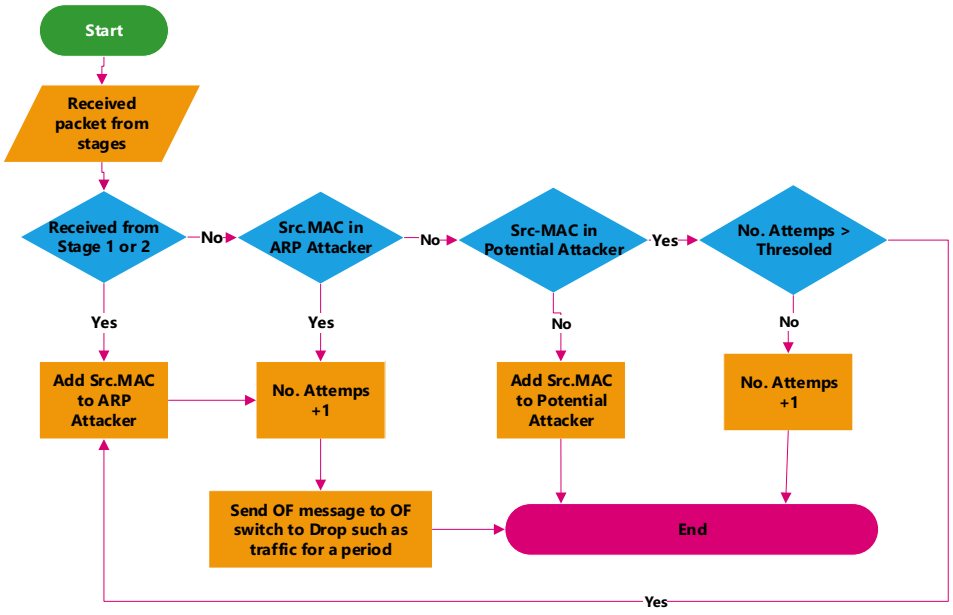


Fig. 6. ARP based attacks detection and mitigation.

the source MAC, while the Source IP will examine at the second stage. In contrast, the destination IP represents the victim's address; therefore, the third stage did not apply to spoofing detection. In general, the attacker can change the packet information with forged information using some computer tools such as Ettercap and Dsniff. Thus, any ARP packet that has incorrect information will be considered a spoofing attempt. Based on the above, the possibility of detecting the ARP spoofing attacks are illustrated in Table 3. Likewise, based on Table 3, the following formula can be derived:

$$F = A \wedge B \tag{2}$$

where, A is Source MAC address, B is Source IP address.

On the other hand, the mitigation part is illustrated in Fig. 6, where the sender information added to the ARP attackers table. Furthermore, the SDN controller sends a flow rule to the OpenFlow switches to drop ARP traffic for the attacker for a specific period.

3.4. Parameter settings of the proposed solution

In this section, we discuss all related parameters that have been utilized for the proposed solution.

3.4.1. Threshold value

The threshold value is used as an indicator to identify any malicious attempt that flood the network by generating the ARP storm attacks. In principle, a host usually sends different rates of ARP packets depending on its operations and network topology. According to [22], sending 200 ARP packets per minute for a network consisting of 100 hosts considers a regular rate.

However, some Cisco devices, through the Dynamic ARP Inspection (DAI), classified device interfaces into two groups, namely, trusted interface (TI) and untrusted interface (UI). By default, the incoming ARP packet rate for untrusted interfaces is 15 packets per second (pps). If the incoming ARP packet rate exceeds the 15 pps, the interface will be automatically shut down [23].

Accordingly, our proposed approach will allow a host to send 36 ARP packets with erroneous information in one minute before it is considered as a malicious host. The number is determined based on the process of analyzing the uses of the ARP protocol. In which the protocol is used in several fields and with other protocols within the network. For example, the ARP protocol is used with the ICMP protocol [24], where each ICMP echo message generates three ARP request packets for an unknown destination. A normal user or network administrator can use the ICMP protocol to check the network device's connectivity using the PING command [25]. The PING command will send 4 ICMP echo messages, which will create 12 ARP request packets. Therefore, we found it unreasonable for a user to send more than 36 messages containing false information within a minute. Based on the above, the below formula illustrates the threshold condition in Scenario No. 7, where P_i is the ARP packet arrived in the i^{th} time slot that contains the wrong destination address, TH is the Threshold Value, and n is the maximum time slot number.

Table 3
The possibility of detect ARP spoofing packet.

Case No.	Source MAC Address	Source IP Address	Target Destination IP*	Target Destination MAC**	Status
1	0	0	–	–	It is detected at the first stage as an attempt to create the ARP spoofing attack because the packet contains an incorrect source MAC address.
2	0	1	–	–	
3	1	0	–	–	It is detected at the second stage as an attempt to create the APP storm attack because the packet contains an incorrect source IP address.
4	1	1	–	–	Normal ARP Packet.

* This field left blank because it will carry the IP address information of the victim host.
** This field left blank because it will carry the MAC address information of the victim host.

$$\sum_{i=0}^{n}(P_i) <= TH \quad (3)$$

3.4.2. Block period setting

The blocking period defines the time to ignore ARP packets sent from a potential attacker or malicious host. In general, actions taken by the proposed approach summarized as follows. Firstly, the attacker's information added to hash tables (ARP attackers or potential attackers). Secondly, a specific period added to OpenFlow floodmode messages and sent to OpenFlow switches. Finally, the OpenFlow switches dropping similar traffic for a specific period based on SDN controller instructions. However, the content of ARP hash tables erases in different periods depending on the information types. In general, the attacker information in the ARP attacker table will be erased after 30 min, while the information will be removed from the potential attacker after 1 min. This is because it is impossible to block a network hosts forever; for example, the commercial solutions (i.e., Cisco Switches) allow the network administrator to define the blocking period from 0.5 to 1092.25 min [23].

Nonetheless, the blocking period in the proposed approach increases depending on the number of attacker attempts. The period allows OpenFlow switches to ignore the same incoming packets from attackers. Therefore, the proposed approach assumes two different periods, one of which for spoofing attackers and another for storm attackers. The time factor is selected based on the Cisco switch default time configuration, wherein Cisco switches the interface will automatically shut down for 300 s if any issue happened [26]. So, the proposed approach will block the attacker 300 s for each attacking attempt. Table 4 describes periods for blocking ARP based attacks in the proposed approach.

3.4.3. ARP storm rate

Essentially, during an ARP storm attacks, the attacker sends numerous ARP packets. As mentioned earlier, attackers do not care about ARP packet information in comparison with the amount of ARP packets sent. Therefore, two scenarios have been simulated to create ARP storm attacks (will discuss in the next sections). The attacker sends a certain amount of ARP packets depending on the scenario. However, no specific value can define the ARP storm attack [27]. Thus, different ARP rates are set for sending from the attacker.

3.5. Simulation methodology

The Linux Ubuntu operating system with the specifications described in Table 5 used to run the Mininet emulator to perform SDN networks experiments. In principle, the Mininet offers a virtual testbed for creating virtual hosts and switches supported by the OpenFlow protocol. Correspondingly, OpenVswitch (OVS) provides a virtual switch connected with the Mininet virtual hosts and the SDN controller. Furthermore, network traffic analysis programs used to capture and analyze data traffic between the SDN controller, OpenFlow switches, and hosts, such as Wireshark and Tcpdump. The Top command-line tool in Linux is used to capture the CPU consumption generated by the proposed solution. Moreover, the Ryu SDN controller selected as a centralized controller for the proposed solution. Finally, Scapy is a python library that supports a large number of protocols, which can send and receive packets. Thus, the Scapy is used to generate ARP based attacks with multiple scenarios.

Table 4
The blocking periods.

Attempts	Time Period for Spoofing Case (Seconds)	Time Period for Storm Case (Seconds)
First attempt	300	180
Second attempt	600	600
Third attempt	900	900
Above Third attempts	900	900

Table 5
Specification of the operating system.

Operating System	Linux Ubuntu 14.04 LTS
Processor	Intel Core *i*5 – 2320
CPU	3.00 GHz
RAM	4.00 GB

3.5.1. Network topology

Fig. 7 illustrates the network topology used in our analysis. We consider a tree network topology with an SDN central controller connected to the OpenFlow switches associated with several network hosts. The tree topology consists of fixed depth equal to 2 and different fanout. Fanout indicates two important concepts; firstly, the numbers of OpenFlow switches in the last stage. Secondly, the number of hosts associated with each OpenFlow switch at the bottom of the tree. In the proposed solution, the fanout of tree topology changes according to the scenario and applied cases. In general, the SDN controller connects with the proposed solution via the northbound interface, while it associates with the OpenFlow switches through the southbound interface. Moreover, the network hosts connect with the OpenFlow switches through virtual links. Finally, the network topology is programmed using Python programming language and is implemented through the Mininet emulator program.

3.5.2. Simulation scenarios

To evaluate the efficiency of the proposed solution, several scenarios are considered, as illustrated in Table 6.

However, some of the scenarios can be further divided into two sub-cases (i.e., case (a) and case (b)). The details of these scenarios and respective sub-cases are as follows:

Scenario No.1 (i.e. ARP Proxy): This scenario shows the role of the proposed approach in suppressing broadcast traffic. Although the fanout of tree topology increases from 5 until 25 and a fixed ARP request rate is utilized, it reduces the generated network traffic in SDN networks.

Scenario No.2 (i.e. ARP-Storm Detection): This scenario illustrates the proposed approachability in detecting the ARP storm attacks in SDN networks. It utilizes fixed fanout with different ARP request rates. Also, it is simulated twice to generate ARP storm attacks in two different methods. In principle, an attacker sends forged ARP request packets at each case. An attacker forges the source address in the first case, while it forges the second case's destination address.

Scenario No.3 (i.e. ARP Spoofing Detection): The third scenario describes the proposed approachability in detecting the ARP spoofing attacks in SDN networks. It utilizes fixed fanout with different ARP request rates. Moreover, it is simulated twice to generate ARP spoofing attacks in two different methods. In principle, an attacker sends forged ARP request packets at each case. Where an attacker forged the source MAC address in the first case, while it forged the source IP address in the second case.

Scenario No.4 (i.e. CPU Measure): This scenario measures the amount of CPU consumption due to the proposed approach. It is simulated twice to measure the CPU consumption with potential cases. In the first case, tree topology's fanout increases from 5 to

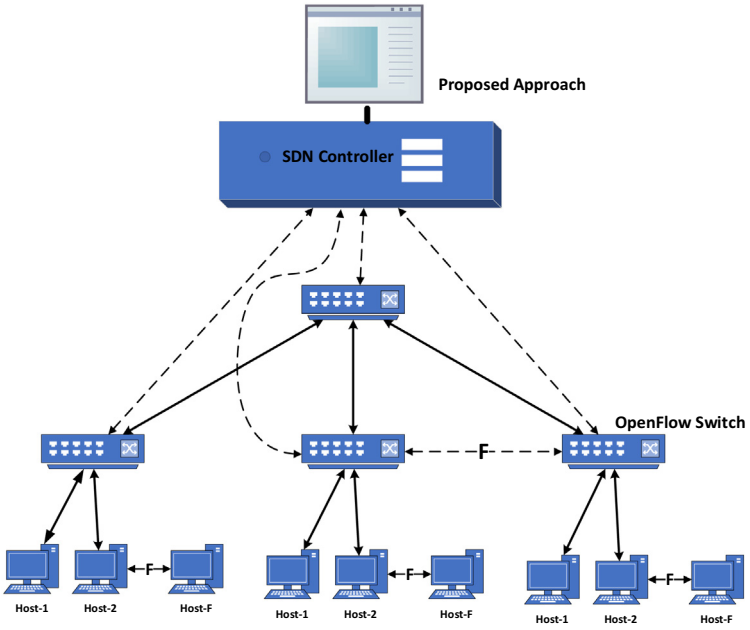


Fig. 7. Network topology.

Table 6
Cases applied in each scenario.

Scenario	Scenario Name	Case	Topology Type	Fanout	ARP Request Sent	Case Summary
1	ARP Proxy	(a)	Tree	5–25	1	Normal ARP Proxy case
	ARP Storm Detection	(a)	Tree	10	500–2500	Case (a) of ARP storm attack
2		(b)	Tree	10	500–2500	Case (b) of ARP storm attack
	ARP Spoofing Detection	(a)	Tree	10	1–5	Case (a) of ARP spoofing attack
3		(b)	Tree	10	1–5	Case (b) of ARP spoofing attack
	CPU Measure	(a)	Tree	5–25	100	Case (a) of CPU measure
4		(b)	Tree	10	50–700	Case (b) of CPU measure

25 with fixed ARP request rates. However, in the second case, a fixed fanout of tree topology and different ARP request rates used.

4. Results and discussions

This section presents the results of our simulation scenarios. Here, we analyze and discuss both the scalability and security performance of the proposed solution. Then, we compare the results against existing work in the respective area.

4.1. SDN networks under broadcast traffic effects

As mentioned earlier, the conventional SDN controller cannot deal with the relentless amount of broadcast traffic. In principle, the SDN controller requests from the OpenFlow switches will flood the broadcast packets. Consequently, the single broadcast packet repeats in the network and generates numerous network traffics. The first experiment was applied to illustrate this point by sending single broadcast traffic into an SDN network tree topology with increased fanout from 5 to 25. Besides, the same experiment was repeated in the SDN network that enabled ARP proxy features with specifications explained in (Scenario No. 1). The experiment results are presented in Fig. 8 which illustrates the impact of increased network hosts in a network domain and the number of network traffic generated in the data and control planes due to a single broadcast packet. Furthermore, it illustrates the ARP proxy role to reduce network traffic due to a single broadcast packet.

The single broadcast traffic generates 37 to 677 packets in the data plane and 12 to 53 packets in the control plane when the fanout increase from 5 to 25. On the other hand, when the ARP proxy feature is enabled on the SDN controller, there are only two packets generates in both data and control planes due to a single broadcast packet. It is apparent from this figure that the SDN controller with the ARP Proxy features is capable of reducing the generated traffic due to a single broadcast packet in both data and control planes and eventually addressing the scalability issue.

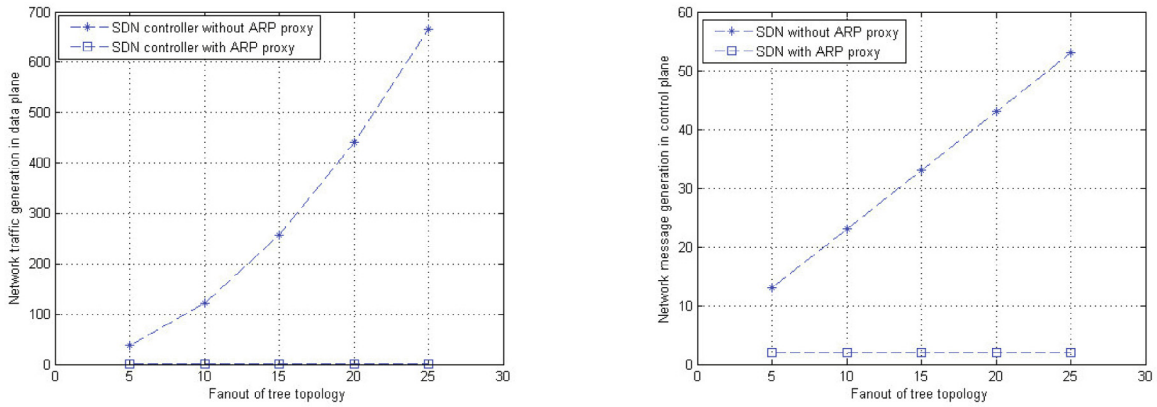
In general, a single broadcast packet can generate a quantity of network traffic in both control and data planes, as observed in Fig. 9. The broadcast packet will repeat (Host - 1) and (Switch-1) until reaching the destination in the data plane. Therefore, we can derive the following equation to generated network traffic in the Data Plane (*DP*) due to single broadcast traffic.

$$DP = Hosts + (Switches - 1) \tag{4}$$

In the control plane, the single broadcast packet will generate two packets at each OpenFlow switch. The first packet will ask the SDN controller about the action applied for the received broadcast packet, and it's called Packet_IN, as shown in Fig. 10. The second packet is Packet_OUT, which sends from SDN controller to OpenFlow switch, and it contains the SDN actions for received packets as appears in Fig. 11. Therefore, generated network traffic in Control Plane (*CP*) can be measured by:

$$CP = No.\ of\ OpenFlow\ Switches * 2 \tag{5}$$

On the other hand, the quantity of generated network traffic is decreased significantly when the ARP proxy feature enabled on the SDN controller, as clarified in Fig. 12. In the data plane, there are only two packets, one for request and another for the reply. While,



(a) Data plane. (b) Control plane.

Fig. 8. Network traffic generated due to the single broadcast packet.

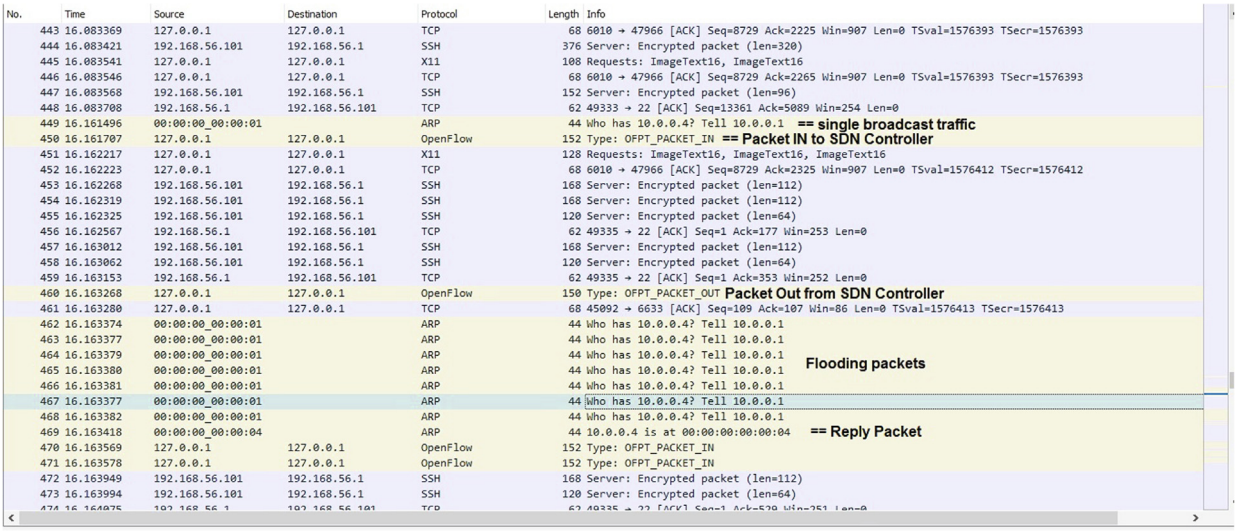


Fig. 9. Wireshark capture traffic for generated traffic SDN networks under broadcast traffic effects.

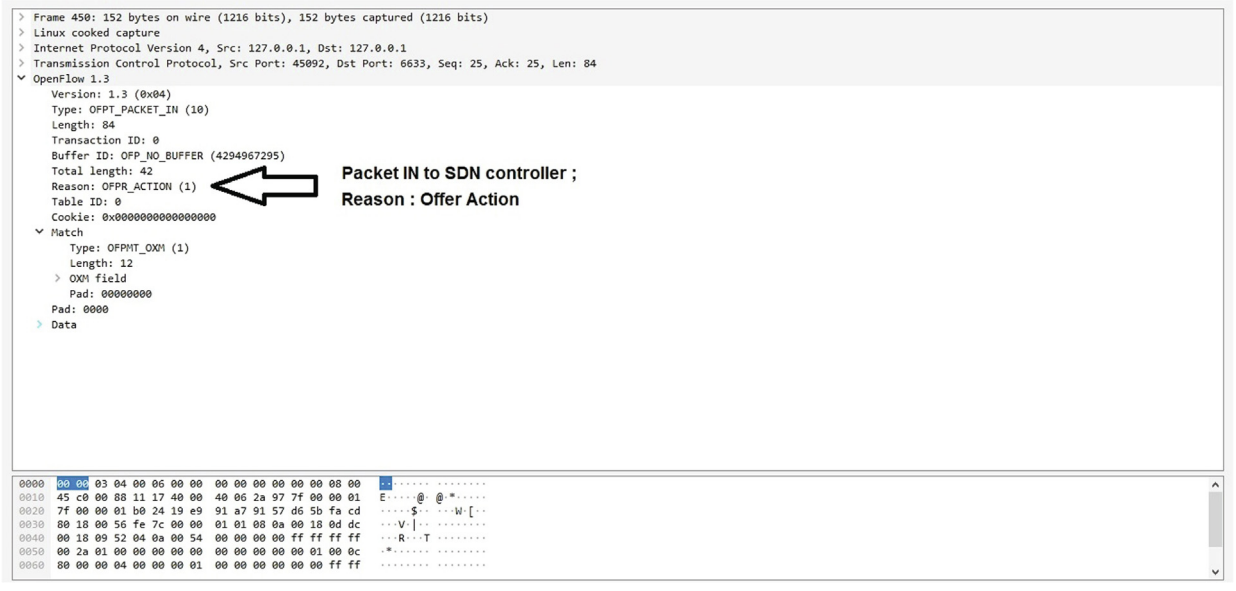


Fig. 10. Wireshark capture traffic for OpenFlow Packet IN under broadcast traffic effects.

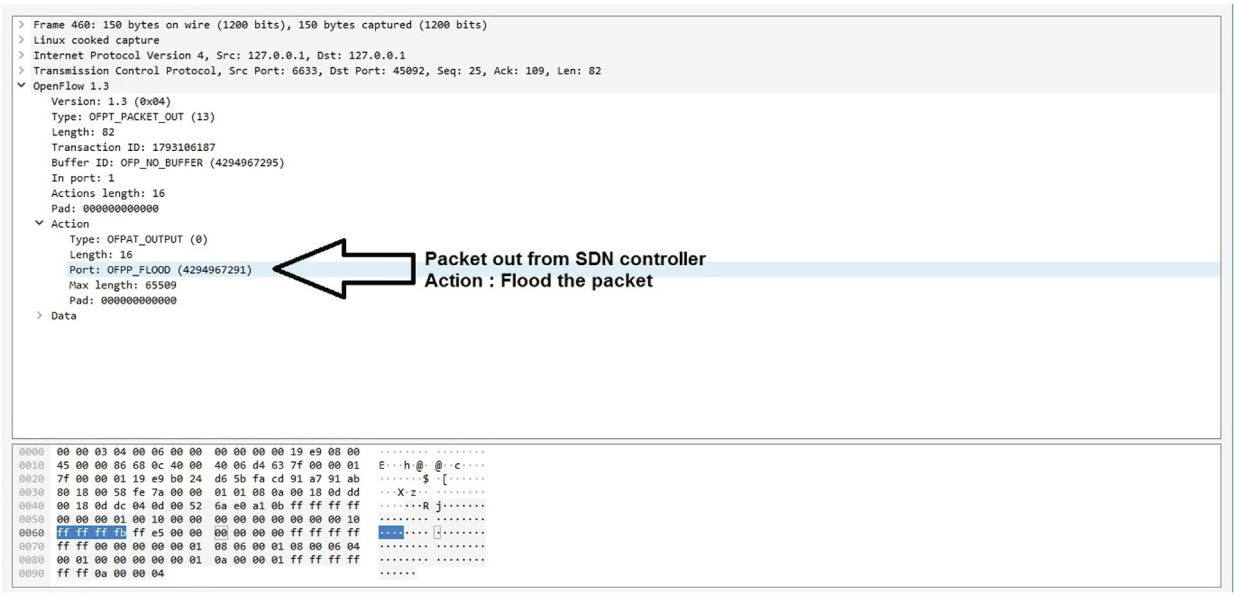


Fig. 11. Wireshark capture traffic for OpenFlow packet out under broadcast traffic effects.

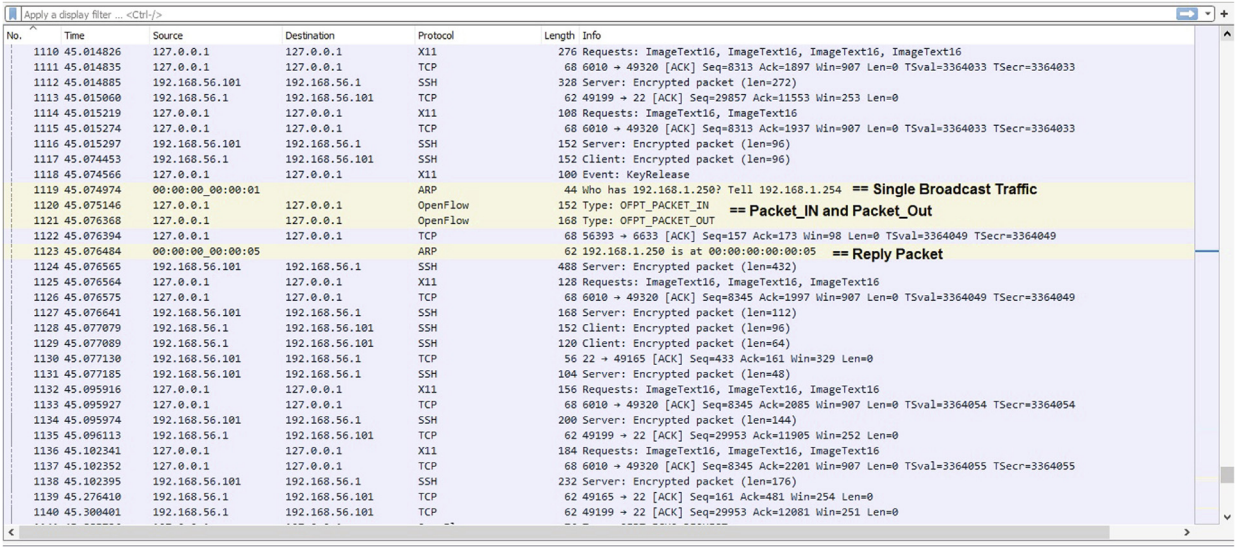


Fig. 12. Wireshark capture traffic for generated traffic in SDN Networks with ARP proxy feature.

in the control plane, there are also two generated packets between OpenFlow switch and SDN controller (OF Packet_IN and OF Packet_OUT) as illustrated in Figs. 13 and 14. Therefore, the generation network traffic can always represent:

$$DP = 2 \tag{6}$$

$$CP = 2 \tag{7}$$

4.2. SDN networks under ARP based attacks

As shown in the previous section, the SDN controller with the ARP proxy feature is a viable way of addressing the Ethernet scalability problem. Nevertheless, this approach opens new security gaps to the network, stemming from ARP's inherent limitations. The basic idea of accommodating the SDN controller with the ARP proxy feature is to redirect the ARP broadcast packets straight to the SDN controller. Unfortunately, the attacker may exploit this vulnerability to send invalid ARP packets to attack the SDN controller (i.e., becoming the point of attack). Therefore, two experiments have been performed (Scenarios No. 2 and 3) to demonstrate the attacker's ability to penetrate the network based on this vulnerability. Our proposed solution is envisaged to address this issue by detection and mitigating the ARP based attacks (i.e., ARP storm and spoofing) through a multistage security algorithm.

4.3. Multistage security algorithm

The incoming ARP packets pass through all stages before the ARP proxy can respond to any request (Refer to subsection 3.3). Thus, the security performance of the detection is assessed based on the true positive ratio (TPR) and false positive ratio (FPR) of every individual stage. The TPR represents the correct detection, while false detection represents the FPR. The TPR and FPR are mea-

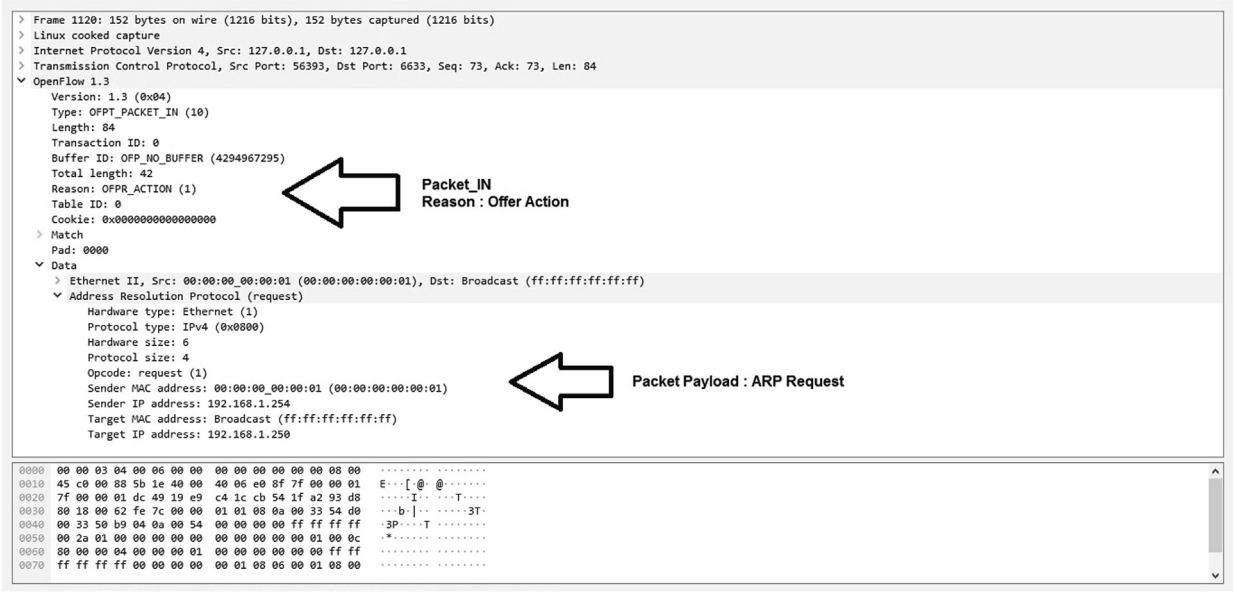


Fig. 13. Wireshark capture traffic for OpenFlow Packet IN with ARP proxy feature.

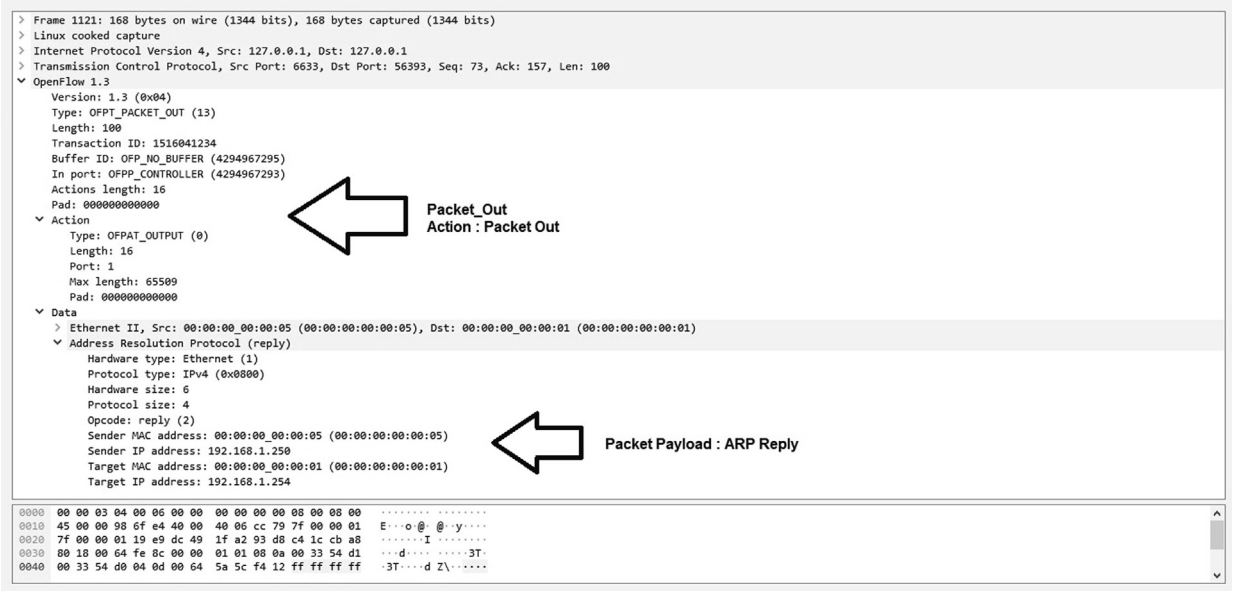


Fig. 14. Wireshark capture traffic for OpenFlow Packet Out with ARP proxy feature.

sured using the following equations, where they are applied on Tables 1 and 2:

$$TPR = \frac{TP}{TP + FN} \tag{8}$$

where *TP* is the number of true positive and *FN* is the number of false negative.

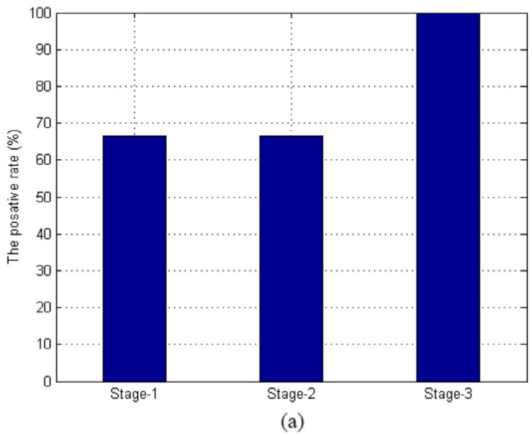
$$FPR = \frac{FP}{FP + TN} \tag{9}$$

where *FP* indicates the number of false positive and *TN* is the number of true negative.

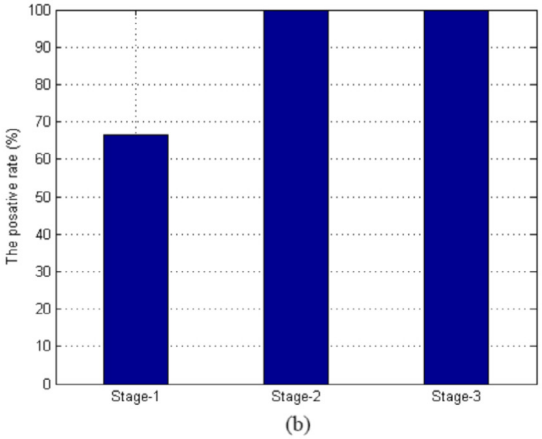
Incredibly, the FPR result for all stages is equal to zero, while the TPR results varied according to attack type. Firstly, the TPR results for the ARP storm scenarios achieved respectively 66.66%, 66.66%, and 66.66% on all stages (refer to Fig. 15(a)). Meanwhile, the TPR results for ARP spoofing scenarios achieved respectively 66.66%, 100%, and 100% on first, second, and third stage, respectively (refer to Fig. 15(b)). Finally, the TPR results for ARP based attack scenarios (both attacks happened simultaneously) achieved 57.42%, 66.66%, and 100% on the first, second and third stage, respectively (refer to Fig. 15(c)).

4.4. Comparison with existing works

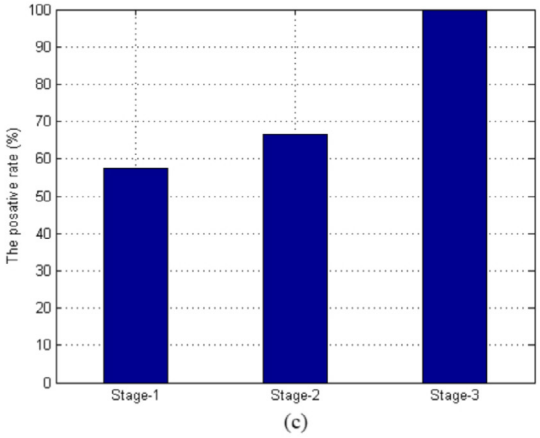
In this section, we compare the proposed solution results with existing work FSDM [12]. In principle, FSDM adopted the APR proxy with SDN networks to minimize the broadcast traffic. Thus, FSDM addressed the Ethernet scalability issue. Fig. 16 illustrates the result of scenario No. 1, where the ability of FSDM and our proposed solution to reduce the broadcast traffic in data and control planes, respectively. It is worth highlighting the scalability perfor-



(a) ARP storm attack scenario.

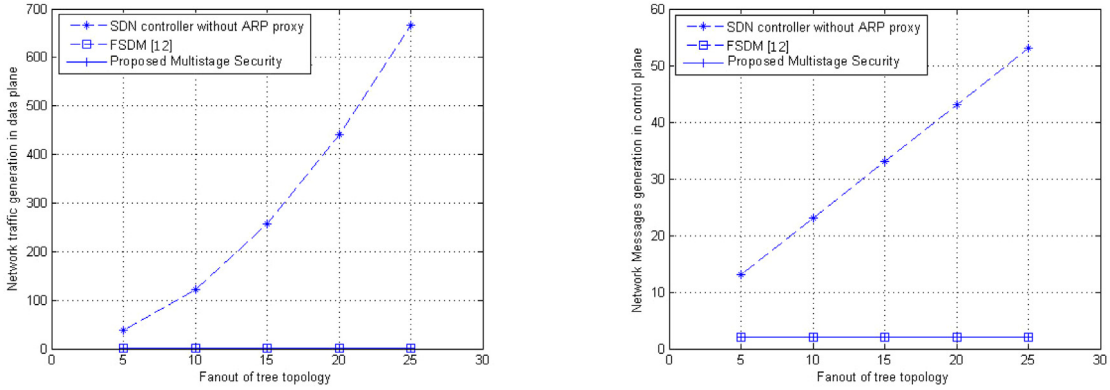


(b) ARP spoofing attack scenario.



(c) ARP based attack scenario.

Fig. 15. The TPR for a multistage security algorithm.

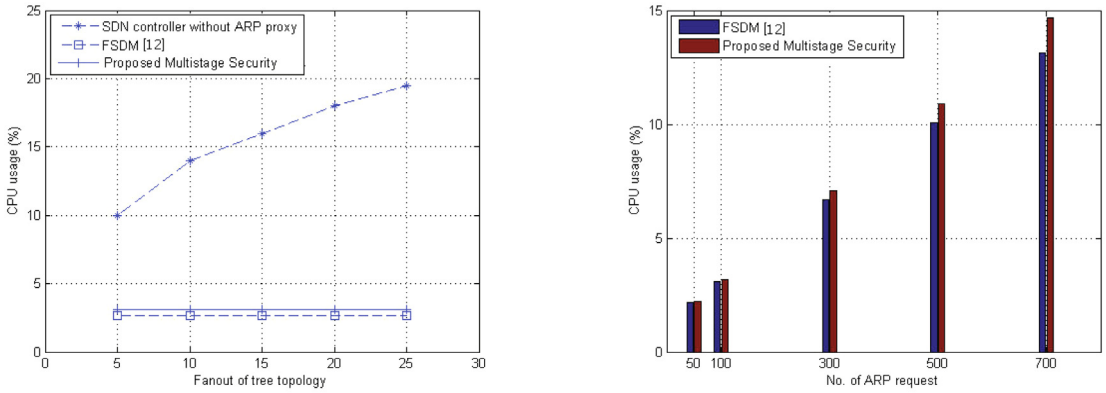


(a) Data plane. (b) Control plane.

Fig. 16. Network traffic generated due to the single broadcast packet. Comparison between the proposed multistage security and FSDM.

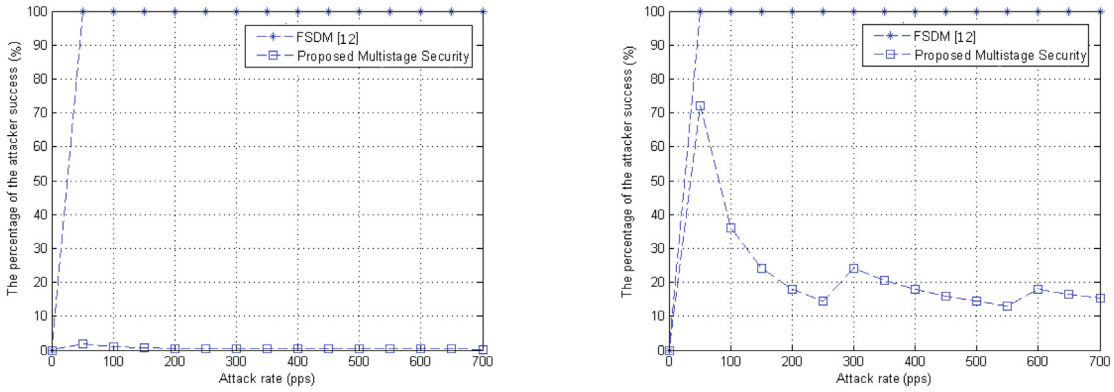
mance. While FSDM achieved similar performance to our proposed solution, it does not offer any security mechanism against ARP based attacks.

Next, we measure and evaluate the amount of CPU consumption (i.e., scenario No.4) imposed by the proposed solution. This scenario comprises two cases, and each case is repeated ten times to ensure the validity of the results, as illustrated in Fig. 17. Observed from the two subfigures, Fig. 17(a) and (b), the amount of CPU consumption imposed by the proposed solution is slightly higher compared to FSDM. This is mainly due to the implemented multistage security algorithm in the proposed solution to inspects all the received ARP packets. Unlike our proposed solution, FSDM



(a) Case-1 (b) Case-2

Fig. 17. The CPU consumption of the proposed multistage security vs. FSDM.



(a) Case 1 (b) Case 2

Fig. 18. The possibility for a success ARP storm attack.

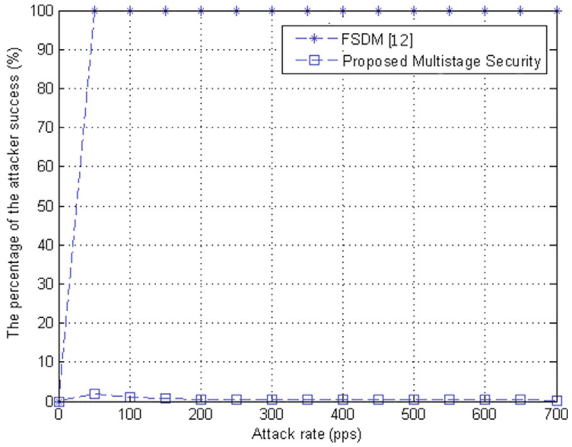


Fig. 19. The possibility for a success ARP spoofing attack.

does not apply any security mechanism, and therefore, it is severely vulnerable to ARP based attacks, as illustrated in Figs. 18 and 19 (Scenarios No. 2 and 3).

5. Conclusion

This paper highlighted the scalability and security issues of Ethernet networks when using SDN. With the emphasis on ARP's broadcast nature and two dominant ARP-based attacks, ARP storm and ARP spoofing, this paper proposed a multistage security algorithm for governing the attack detection and attack mitigation processes while maintaining the scalability of the SDN based Ethernet networks. The proposed algorithm utilized the features of SDN architecture like centralization to build hash tables. It also incorporated the ARP proxy responsible for addressing the scalability issue, notably directly replying to ARP request packets. Several attack scenarios have been implemented using the Mininet simulator to demonstrate the performance evaluation for the proposed algorithm. The results obtained show that the proposed solution can detect and mitigate the ARP based attacks and offer scalability to the Ethernet networks.

Declaration of Competing Interest

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

References

[1] Metcalfe RM, Boggs DR. Ethernet: Distributed packet switching for local computer networks. Commun ACM 1976;19(7):395–404.
[2] Plummer DC. An ethernet address resolution protocol. RFC826 1982.
[3] AbdelAzim NM, Fahmy SF, Sobh MA, Eldin AMB. A hybrid entropy-based dos attacks detection system for software defined networks (sdn): A proposed trust mechanism. Egypt Inf J..

[4] Carl-Mitchell S, Quarterman JS. Using arp to implement transparent subnet gateways, 1987. p. RFC1027.
[5] Song MS, Lee JD, Jeong Y-S, Jeong H-Y, Park JH. Ds-arp: a new detection scheme for arp spoofing attacks based on routing trace for ubiquitous environments. Sci World J 2014;2014:1–8.
[6] Elmeleegy K, Cox AL. Etherproxy: Scaling ethernet by suppressing broadcast traffic. In: IEEE INFOCOM 2009. IEEE; 2009. p. 1584–92..
[7] Asadujjaman A, Moni SS, Alam MS. Fcsea: A floodless carrier-grade scalable ethernet architecture. In: 2016 9th International conference on electrical and computer engineering (ICECE). IEEE; 2016. p. 427–30.
[8] Kim C, Caesar M, Rexford J. Floodless in seattle: a scalable ethernet architecture for large enterprises. In: ACM SIGCOMM computer communication review, vol. 38. ACM; 2008. p. 3–14..
[9] Scott M, Moore A, Crowcroft J. Addressing the scalability of ethernet with moose. In: Proc. DC CAVES workshop; 2009..
[10] Qian C, Lam SS. A scalable and resilient layer-2 network with ethernet compatibility. IEEE/ACM Trans Netw 2014;24(1):231–44.
[11] Committee LS, et al. Ieee standard for local and metropolitan area networks: Station and media access control connectivity discovery. ieee standard 802.1ab; 2016..
[12] Wang J, Huang T, Liu J, Liu Y. A novel floodless service discovery mechanism designed for software-defined networking. China Commun 2014;11(2):12–25.
[13] Cho H, Kang S, Lee Y. Centralized arp proxy server over sdn controller to cut down arp broadcast in large-scale data center networks. In: 2015 International conference on information networking (ICOIN). IEEE; 2015. p. 301–6.
[14] Li J, Gu Z, Ren Y, Wu H, Shi S. A software-defined address resolution proxy. In: 2017 IEEE symposium on computers and communications (ISCC). IEEE; 2017. p. 404–10.
[15] di Lallo R, Lospoto G, Rimondini M, Di Battista G. How to handle arp in a software-defined network. In: 2016 IEEE NetSoft Conference and Workshops (NetSoft). IEEE; 2016. p. 63–7.
[16] Kataoka K, Agarwal N, Kamath AV. Scaling a broadcast domain of ethernet: Extensible transparent filter using sdn. In: 2014 23rd International conference on computer communication and networks (ICCCN). IEEE; 2014. p. 1–8.
[17] Xu Y, Lu X, Zhang T. A novel efficient sdn based broadcast scheme. In: 2015 International conference on computer science and intelligent communication. Atlantis Press; 2015..
[18] Alasadi E, Al-Raweshidy HS. Ssed: Servers under software-defined network architectures to eliminate discovery messages. IEEE/ACM Trans Netw 2017;26 (1):104–17.
[19] Schneider F, Bifulco R, Matsiuk A. Better arp handling with inspired sdn switches. In: 2016 IEEE international symposium on local and metropolitan area networks (LANMAN). IEEE; 2016. p. 1–6.
[20] Sumadi FDS, Risqiwati D, Syaifuddin S. Semi-reactive switch based proxy arp in sdn. In: 2018 5th International conference on electrical engineering, computer science and informatics (EECSI). p. 478–82.
[21] Groma M, Boros T, Helebrandt P. Scalable cache-based address resolution protocol handling in software-defined networks. In: 2019 XXVII international conference on information, communication and automation technologies (ICAT). IEEE; 2019. p. 1–6..
[22] Orzach Y. ARP and IP analysis in network analysis using wireshark cookbook. Packt Publishing Ltd; 2013.
[23] Cisco System. Configuring dynamic arp inspection, Cisco Nexus 1000v Security Configuration Guide. 2012;13–18..
[24] Postel J. Internet control message protocol. RFC792 1981.
[25] Lima FH, Vieira LF, Vieira MA, Vieira AB, Nacif JAM. Water ping: Icmp for the internet of underwater things. Comput Netw 2019;152:54–63.
[26] Hucaby D. CCNP SWITCH, 642-813 official certification guide. Cisco Press; 2010.
[27] Orzach Y. Ethernet, LAN switching, and wireless LAN in network analysis using wireshark cookbook. Packt Publishing Ltd; 2013.