EFFICIENT MECHANISM FOR SECURING SOFTWARE DEFINED NETWORK AGAINST ARP SPOOFING ATTACK

Article · July 2019

DOI: 10.26682/sjuod.2019.22.1.14

2 authors:

Ahmad Baheej Al-Khalil
University of Dohuk
14 PUBLICATIONS **4** CITATIONS
SEE PROFILE

Harman Khalid
University of Duhok
1 PUBLICATION **1** CITATION
SEE PROFILE

EFFICIENT MECHANISM FOR SECURING SOFTWARE DEFINED NETWORK AGAINST ARP SPOOFING ATTACK

HARMAN **Y. I. K**HALID
PARISHAN **M. I**SMAEL
AHMAD **B**AHEEJ **AL-K**HALIL
Department of Computer Science, College of Science, University of Duhok, Kurdistan Region, Iraq

(Received: January 28, 2019; Accepted for publication: March 27, 2019)

ABSTRACT

Software Defined Network SDN is a new emerging paradigm of networking which decouples the data plane and the control plane. It is expected to be a solution to overcome the limitations of traditional networks. Conventional networks had several security problems, some of them disappeared by SDN and some others still exist such as Address Resolution Protocol ARP spoofing. This paper discusses the attacks of ARP spoofing and presents a deep study on the existing solutions either in traditional or SDN environments. A light, reliable, fast and effective mechanism has been proposed to prevent ARP spoofing, without any additional software or hardware by utilising SDN capabilities. In this work, the SDN controller has been extended by a module which checks every ARP packet in network to detect possible spoofed packets and stop them. Experiments were conducted on the simulated environment using Mininet to check the functionality of the proposed mechanism. The simulation results showed that the proposed mechanism is robust against ARP spoofing attack.

KEY WORDS: ARP cache Poisoning, ARP spoofing, ARP, Attack, DHCP, DoS, Mininet, SDN,Security.

1. INTRODUCTION

Nowadays the widely adoption of traditional Internet Protocol IP networks cannot hide the fact that they are complex and hard to manage large number of network devices. The new Internet-based systems technologies that appeared in this decade such as cloud services, Internet of Things IoT, Voice over IP VoIP, big data require high bandwidth, scalability, higher accessibility and dynamic management (Benson, Akella and Maltz, 2009)(Alsmadi and Xu, 2015). In order to enforce network policies, each network device should be configured separately by the network operators using low-level and often vendor-specific commands, and manually input these commands using command line or graphical user interfaces. In case of any failure to a section of network or adopting load changes, there must be an automatic reconfiguration and response mechanisms which are very hard to be performed in current IP networks. In addition to the management difficulties, the vertically integration of current network devices make it more complicated architecture. In the absence of unified control unit for current distributed control networks, the network management becomes very challenging task and the difficult configuration process lead to many errors, security gaps and network faults. New protocol designing may take several years to be fully matured and deployed. Therefore, a new paradigm to change the network architecture instead of IP, is considered as a difficult (Kreutz *et al.*, 2015). Furthermore, with the growth of network and its traffic, operational expenses of running an IP network increased rapidly.

The spark of solution for current network infrastructures limitation was glowed with the emerging of Software-Defined Networking SDN. SDN is a new networking paradigm that gives hope to change the inertia of current network model (Kreutz *et al.*, 2015). It moves the network model to be open, programmable, reliable, secure and manageable infrastructure (Klöti, Kotronis and Smith, 2013).

SDN architecture allows the users to enhance network security by providing clear view over the network for easy management, maintenance, control and reactivity. Security of SDN is an important concern since there is no security features in its architecture. Many research papers and analysis such as (Alsmadi and Xu, 2015)(Kreutz,

Ramos and Verissimo, 2013)(Li, Hong and Bowman, 2011)(Brooks and Yang, 2015)(Scott-Hayward, O'Callaghan and Sezer, 2013) have been done and showed that various security attack could conducted on SDN. Some of spoofing attacks such as IP spoofing, Domain Name System DNS spoofing and Address Resolution Protocol ARP spoofing etc are still seen as a serious concern for SDN architecture since they cannot mitigated by a plain SDN controller. Usually ARP spoofing attack is the first step in other threads such as Denial of Services DoS and a Man-in-the-Middle MIM, where important information regarding the network user can be stolen by attacker. ARP spoofing attack is the most common attack in Local Area Network LAN. ARP is used by the host in the network to get the physical address of host willing to communicate with. Since ARP protocol does not have security mechanism, it is used by intruder to impersonate other hosts in network. In this paper, a mechanism for validating ARP protocol has been proposed. The mechanism is mainly focusing on the detection and the prevention of ARP spoofing by malicious host that sends crafted ARP packet to poison ARP table of other hosts in network.

2. RELATED WORKS

Many tools and methods have been proposed and used to overcome the ARP spoofing protocol in traditional network infrastructure. Network devices such as switches are not designed to detect and prevent ARP attacks therefore they require integrating a feature or a software in order to perform detection or prevention to such an attack. Since the switch has limited resources, its opportunity for solving this threat by itself is not possible, because it cannot handle additional intensive computation tasks. Therefore, the prevention and detection techniques were offloaded to the host itself. The proposed techniques that been used in traditional network are no longer applicable in SDN due to the different architectures of them (Balagopal, Agnise and Rani, 2017). In SDN, switch acts as a forwarding device and does not have capability of processing the packets. Thus, the switch in SDN has less responsibility and power than the switch in traditional network, but the SDN central management and controller accomplished with global view give ability to move intensive tasks of switch to the powerful controller. For this reason, SDN gives the duty of prevention and detection of spoofing attack to controller. Although some methods and techniques in traditional network in ARP spoofing can work in SDN, but they are not taking the advantage of basic SDN principle of separating data and control planes.

Current SDN controller such as POX, Floodlight, Open Daylight and Beacon are not safe form ARP spoofing attacks as there were many studies showed that these controllers can be affected by poisoning attacks (Ubaid *et al.*, 2017). Many studies have dealt with ARP poisoning in SDN environment. A method called FICUR has been proposed by (Nehra, Tripathi and Gaur, 2017) to detect ARP poisoning and ARP flooding attack as a module in extended controller. The proposed method used three functions, the first function helps to create a detection list of poisoning and flooding attacks by analysing frame header, the second function checks ARP entries for same source IP address and different MAC address in that list and the third function gives average of observed thresholds to check if the existing number of ARP request is more than the average threshold from history. If it is more, then it may be ARP flood attack. In an alternative study, the authors of (Alharbi *et al.*, 2016) proposed SARP NAT method to detect ARP-Request attack which has similar working principle of Network Address Translation NAT. SARP NAT prevents the ARP table poisoning by replacing the potentially spoofed source IP and MAC field of ARP header with a known dummy values that are not found in the network, and save the original packet information in list of pending ARP-Requests, then modified ARP request to destination host. When the destination host response, it shows in the pending list for corresponding ARP request and replaces the dummy value inserted before with the original values saved in the list. The drawback of this method is that there is no mechanism for detecting the ARP-Reply attack. They only accept ARP-Reply for corresponding request in the list, otherwise the reply is dropped. Additionally even if there is no mechanism for ensuring that the reply with corresponding ARP in list is not spoofed. In a detailed study, (Abdelsalam and El-sisi, 2015) proposed an algorithm to detect the spoofing packet in the network. They used a table containing all IP-MAC association of all hosts in the network and these mapping were provided by Dynamic Host Configuration Protocol DHCP server. By modifying the controller and extending a module that handled all ARP requests in the network, they could analyse the packet header and check if packet holds spoofing characteristics. The limitation of this mechanism is that it can only be used in DHCP environment. In similar study, the authors in (Masoud, Jaradat and Jannoud, 2015) designed an

algorithm to be utilised in one of two scenarios static or DHCP environment. The two scenarios have the same mechanism, the only difference is the method of obtaining IP-MAC mapping. They depend on the main table of the controller which contains IP-MAC mapping association of all hosts in the network. The algorithm inspects all ARP-Reply and compare the details in ARP header and if a spoofing is found then the packet will be discarded. In DHCP environment the main table is filled inspecting DHCP packets in the network and extracting IP and MAC. While in static IP-MAC, pair of each device is inserted manually. However, this is not an efficient method in large scale network it requires its table to be continuously updated by administrators which is a difficult task. In the same context, the authors of (Solomon, 2015) suggested a reactive ARP query which keeps the partial view of the network in the data plane along with existing whole view in the controller, shifting some of the responsibility of the controller back to switches. In OpenFlow protocols the forwarding devices are not required to process the packets, they are required to forward packets according to the rules in the flow table. The reason behind giving only forwarding task to those devices is to make them faster in transmitting packets. However, their mechanism requires changes to the OpenFlow protocol by giving processing task back to the forwarding devices.

3. THE PROPOSED MECHANISM

The proposed mechanism exploited SDN's features of central control and management, global view and ability to gather required data from network. The collected data will be useful to detect and mitigate ARP spoofing attack. The proposed framework utilized Layer 2 L2 learning and DHCP server which are existing components of POX controller.

A. Layer 2 Learning Switch Application

The controller runs L2 Learning Switch application to programmatically and dynamically manage SDN switches. Once a packet enters the switch and does not match any flow rules in its flow table, the switch will redirect this packet to the controller. L2 application will process this packet and direct it to the switch with an action regarding that packet by installing flow rule in the switch flow table. L2 application allows the switch to forward the next related packets faster without sending them to the controller (Solomon, 2015). Unfortunately L2 application does not provide any mitigation against ARP spoofing attack, and the switch is only required to forward the packets but not processing them. Therefore attacker can craft spoofed packets and poison the victim ARP cache.

The proposed mechanism modified L2 learning switch component module presented in POX controller which is written in Python. The proposed mechanism will initialise modified L2 learning module for each switch that is connected to the controller. The switch will populate its flow table with flow rule instructions to forward ARP packets to the controller for the purpose of responding or inspecting for possible poisoned packet.

B. DHCP and Main Table

The proposed mechanism utilises DHCP protocols to implement its main table in the controller. This central table is composed of IP-MAC associations for each device on a given network and it is used later for validate ARP packets. Since DHCP is enabled in the network, the first arrived packet from new host will be considered as a discovering packet, as the host will be asking for an IP from DHCP server. Therefore it is important to install flow rule in the flow table of each switch which is connected to the controller in order to forward DHCP packet to controller. The DHCP server is critical in the proposed mechanism and the POX controller already has it installed as an optional module. Once the host starts asking for an IP address in DHCP environment, it will broadcast DHCP discovering packet. Then the DHCP server response with DHCP offer packet. The proposed mechanism inspects this packet to extract the IP and MAC addresses form its header, and record the mapping information in the main table which exist in the controller. This table plays important role in the mechanism since the detecting spoofing ARP depends and later gives an acknowledgement on it. It is important to mention that the environment of the proposed mechanism has only one DHCP server. This is because the new host will accept the only offer from single server by sending DHCP request message and the later give acknowledgement. Furthermore, when host IP addresses are used as index keys in the main table, they do not allow the same IP address to be given to two different hosts in the same network.

C. The Proposed Mechanism Design

The L2 learning is modified with a proposed algorithm to perform functions of prevention of ARP spoofing. It does not require OpenFlow protocol and ARP protocol to change or enforce specific topology. The prevention mechanism is just extended to the L2 component and does not require additional software or hardware to be added to the network. In addition, the ARP spoofing mitigation part does not involve any cryptographic mechanism, therefore it is light, fast and able to detect the ARP spoofing attack immediately. Figure 1 depicts the process of the Spoofing detection and prevention of the propose mechanism.

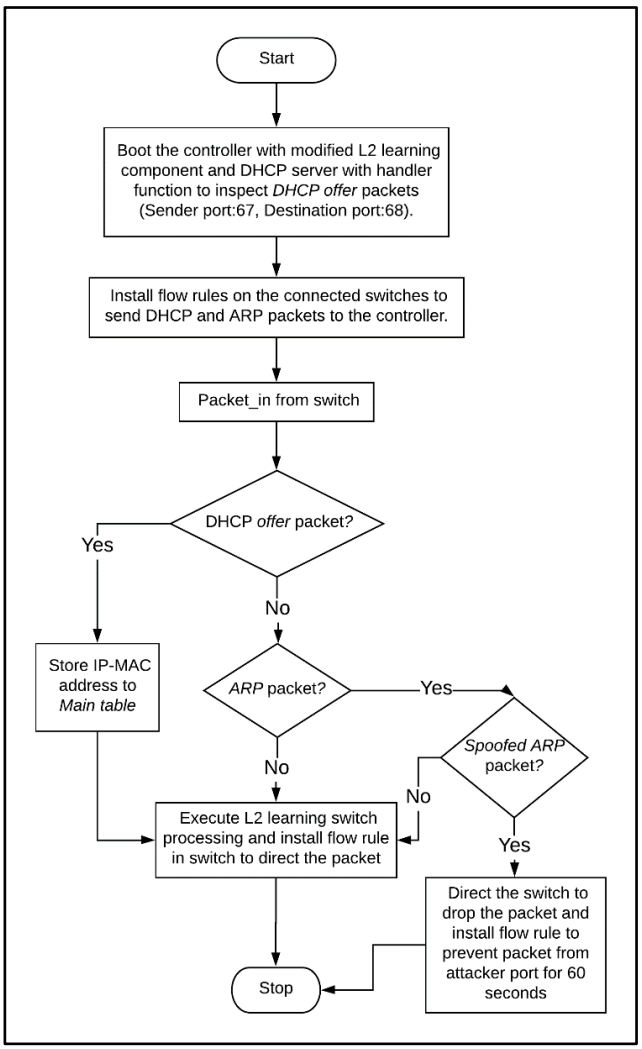

Fig. (1): ARP Poisoning Detection and Prevention Process

The designed ARP spoofing mechanism followed the recommendation and requirement which are presented in (Song *et al.*, 2014) (Abad and Bonilla, 2007).
For any ARP spoofing countermeasure:

- It should be easy to deploy and does not require installation of additional software on each hosts as this incurs additional costs
- It should minimize cryptographic processing
- It should provide timely detection and prevention
- It should minimize hardware costs
- It should be backward compatible with existing ARP process
- It should detect and prevent all types of ARP attacks
- It shouldn't slow the ARP request/reply process significantly
- It should consume low network resources

According to the above guidelines, the proposed mechanism is deployed in modified L2 learning component in the controller and does not require any additional software or device in hosts or in network. It does have cryptographic technique, and it can detect and prevent all types of ARP spoofing attacks as they enter the controller and pass through checking process. It does not require any changes to the current ARP protocol. On the other hand, the mitigation mechanism produces a little slowness of ARP request/reply process and creates a little overhead in controller's CPU load. This is due to the extra checking functionality extended to the controller and the requirement of sending all ARP packets to controller for checking.

D. The Prevention Algorithm

The following pseudocode demonstrates how does each ARP packet, that is sent by switch, is analysed by the controller to detect and drop the poisoned ARP reply or ARP request packet. If the packet passed the test successfully it will be processed normally according to the L2 forwarding application of pox controller.

1. If source MAC of Ethernet not like Source MAC of ARP
2. Spoofed (Drop)
3. Else
4. If source MAC-IP addresses mapping in ARP header not found in Main table.
5. Spoofed (Drop)
6. Else
7. If Destination IP of ARP not found in Main Table
8. Spoofed (Drop)
9. Else
10. If Destination MAC is Broadcast
11. If ARP Reply and Destination MAC is Broadcast and Source IP not like Destination IP
12. Spoofed (Drop)
13. Else
14. Broadcast
15. Else
16. Forward to designated host

4. EMULATING ENVIRONMENT SETTING

In order to implement and test the proposed mechanism, Mininet version 2.2.2 (Lantz, Heller and McKeown, 2010) has been used, which is an open-source network emulator allocated for SDN educational and research purposes. Mininet version 2.2.2 is used with other useful tools which are combined together in pre-built on Ubuntu 14.04 with 1 GB of RAM and 1 core CPU. The virtual machine image of Mininet is installed in VirtualBox on the host machine that is running on Windows 10 and has Intel Core i7 processor with 16 GB of RAM. To prevent the load on the controller that is created by spoofing detection algorithm and to obtain accurate results, it is very important to limit the link parameters in the proposed mechanism. In addition, it is essential to mention that the speed of link (Cho, 2019) is recommended to be 10 or 100 Mb/s rather than 1 Gb/s. This is because the forwarding devices or switches in the test environment share resources such as CPU and memory, which have slower performance than the dedicated hardware switches.

In this research experiments, all links of the network have been configured to 100 Mbps bandwidth, 5ms delay, 0% loss and 1000 maximum packet queue size. A simple Tree topology of 4 hosts has been chosen to evaluate the efficiency of the proposed ARP mitigation algorithm. In addition, OpenVswitchs are connected to a single remote controller with an IP address of 192.168.201.5 as shown in Figure 2.

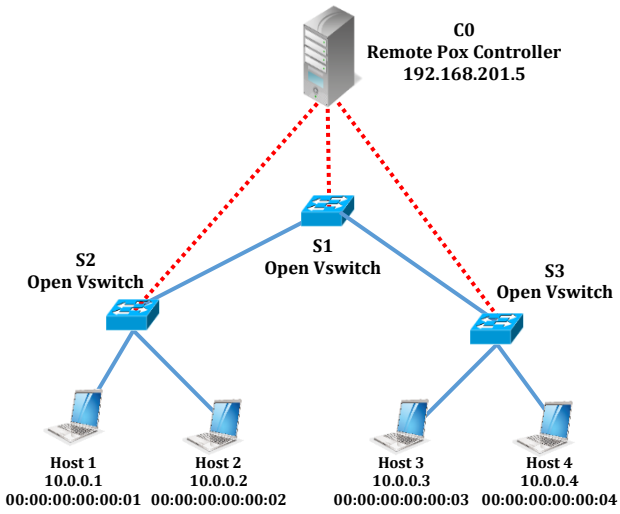

Fig. (2): Tree Topology (depth = 2 , fanout = 2)

POX controller has been used in this work's testbed as a remote controller. It is a framework for interacting with OpenFlow switches and written in Python. As it has been mentioned before, forwarding L2 Learning is utilised as a module of pox controller component to make OpenVswitch a layer 2 learning switch. Enabling this component makes the OpenFlow switch learns Ethernet MAC addresses, and matches all fields in the packet header so it may install multiple flows in the network for each pair of MAC addresses. This module is modified to perform ARP checking for poisoning packets. Another module of pox component which is proto.dhcpd plays a noble role in the proposed mechanism which acts as a DHCP server. This services are used in the proposed mechanism to populate the Main table in the controller with the details of IP-MAC mapping addresses of all hosts in the network.

4. RESULTS AND DISCUSSION

The conducted experiments have been used to evaluate the functionality of the proposed ARP mitigation mechanism against the ARP spoofing attack. The topology in Fig. 2 Considers Host 1 as an attacker who is trying to poison ARP cache of Host 4.

In Host 1, Scapy tool (Biondi, 2019) generates various ARP packets and each one has a different spoof field of ARP header, which is encapsulated in Ethernet frame as shown in Figure 3.

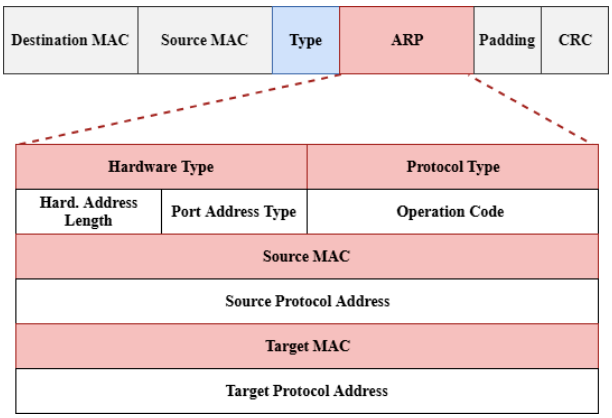

Fig. (3): ARP Frame Structure

Table 1 summarises the attempts of attacker (Host 1) to craft spoofed packets to poison ARP cache of Host 4. Each time the attacker poisoned one field of ARP header corresponding to each checking condition of mitigation pseudocode in previous section.

Spoofed Field	Request / Reply	Description	Action
Source MAC of ARP Header	Request and Reply	ARP header source MAC address is spoofed	Detected and Dropped
Source MAC of Ethernet Header	Request and Reply	Sender MAC of Ethernet header	Detected and Dropped
Source MAC of ARP and MAC Headers	Request and Reply	Sender MAC of ARP header and Sender MAC of Ethernet header are same but spoofed (but the IP-MAC of sender are not in table)	Detected and Dropped
Sender Protocol Address	Request and Reply	The source IP are faked (the IP-MAC of sender are not in table)	Detected and Dropped
Target Protocol Address	Request and Reply	The destination IP are faked (the IP of sender are not in table)	Detected and Dropped
Broadcast ARP Reply with Different Sender and Target Protocol Addresses	Reply only	ARP reply that source IP and Destination IP not same (not gratuitous Reply)	Detected and Dropped

Table 1: Experimented Spoofed Fields of ARP Packet

A. ARP Poisoning Detection and Mitigation Time

The proposed detection's mechanism is robust in term of functionality against all different types of spoofing packets, where all of them have been dropped. The attack detection and mitigation times have been measured as illustrated in Fig. 4.

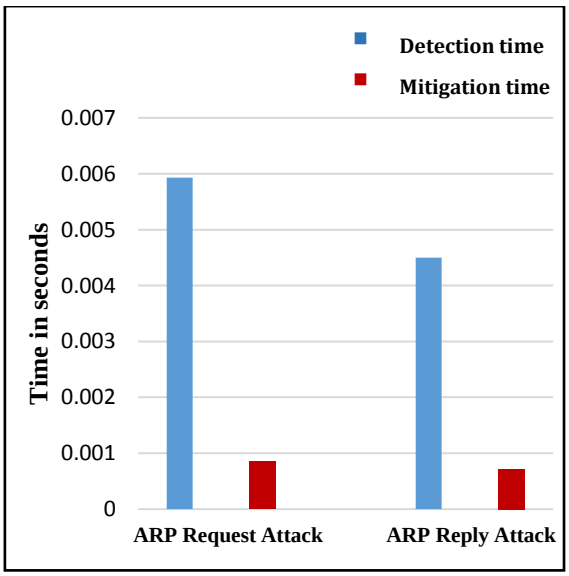

Fig. (4): Detection and Mitigation Time

Attack Detection time is the time period when spoofed packet enters the controller as packet_in for validation until detection time. Attack mitigation time is a time taken by the proposed mechanism to mitigate an attack after its detection. Fig. 4 shows that the detection time of the spoofed packet is around 6ms for ARP request and 5ms for ARP reply, which are very short times. The mitigation action has been immediate and it took only around 0.8 ms for both ARP request and ARP reply.

B. CPU Load

The CPU load that is created by the proposed mechanism on the Pox controller has been compared with the original L2 learning component of pox controller. The modified L2 learning component with mitigation module has been labelled as "Mitigation L2" see Fig. 5.

A number of ARP requests has been generated and sent from Host 1 to Host 4 (Figure 2) using ARPING tool. The results were recorded using NMON tool. The proposed scenario considers a single ARP traffic of Host 1 sending ARP request to obtain the MAC of Host 4.

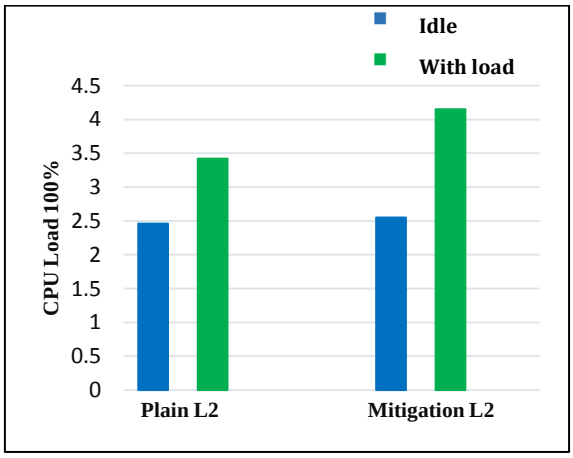

Fig. (5): POX Controller CPU Utilisation

The controller of original L2 learning component has no additional functionality except ARP traffic, where the overall Pox controller CPU usage was 3.4%. By extending controller with mitigation mechanism, there was an increase in CPU load that reach to 4.2%. Comparing results in Fig. 5, the expense of mitigation functionality on the CPU load is noticeable. This increase in CPU load is caused by the effect of sending all ARP packets to the controller in order to check the infected packet. While in the original L2 learning component, the ARP packet is dealt with as a normal packet, where the controller simply directs the switch to forward similar packets to specific destination in a period of time. Therefore, this will prevent ARP packets to visit the controller.

5. CONCLUSION AND FUTURE WORK

This paper, introduced SDN, OpenFlow and ARP protocol. The ARP protocol features which leads to security gap resulting of many threats such as ARP poisoning attack has been intensively overviewed. In addition, this research has covered many studied techniques and methods which has been proposed by others against ARP spoofing attack in traditional and SDN environment. Also, it highlighted how the SDN infrastructure capabilities such as global view, programmability and centralised control could be exploited to implement effect prevention mechanism to mitigate this attack.

This work presented an efficient approach, making use of SDN features to mitigate both ARP Request and Reply based spoofing attacks in SDN environment. The proposed mechanism does not require infrastructure changes, changing of ARP or OpenFlow protocol. Moreover, there is no need to install additional software or hardware in network. The proposed mechanism depends on a trusted IP-MAC Main table exist on the controller and it does work in conjunction with DHCP server. Experimental results showed that the proposed mechanism is robust against ARP threats, very fast in detection and prevention, and it has a minor controller CPU overhead.

Since the current mechanism is limited to the use of single controller environment it is highly recommended to implement it to be used in Distributed controller environment. The proposed mechanism is based on DHCP technology to populate its Main table in controller. This mechanism is applicable in network with DHCP server. The mechanism is essential to overcome the adaptability issue in network environment without DHCP server. In other word, how to get IP-MAC pair details of every host in network without help of other technology or finding mechanism independently to obtain these details for feeding the Main table. Since every ARP packet visits the controller for validation process, in large network this may lead to broadcasting storm in network. There is a need for a mechanism to suppress ARP broadcast by centrally processing ARP packets. The fact that if the attacker know every ARP packet visiting the controller, he can craft huge number of correct ARP packets targeting the controller in order to overwhelm it. Since the Controller considered as the single point of failure of networks. There must be a mechanism of port level ARP packet monitoring to prevent Denial of Service DoS attacks against the controller.

6. REFERENCES

Abad, C. L. and Bonilla, R. I. (2007) 'An analysis on the schemes for detecting and preventing ARP cache poisoning attacks', *Proceedings - International Conference on Distributed Computing Systems*, pp. 60–66. doi: 10.1109/ICDCSW.2007.19.

Abdelsalam, A. M. and El-sisi, A. B. (2015) 'Mitigating ARP Spoofing Attacks in Software-Defined Networks', *ICCTA 2015, At Alexandria, Egypt*, (October).

Alharbi, T. *et al.* (2016) 'Securing ARP in Software Defined Networks', *Proceedings - Conference on Local Computer Networks, LCN*, pp. 523–526. doi: 10.1109/LCN.2016.83.

Alsmadi, I. and Xu, D. (2015) 'Security of Software Defined Networks: A Survey', *Computers and Security*. Elsevier Ltd, 53, pp. 79–106. doi: 10.1016/j.cose.2015.05.006.

Balagopal, D., Agnise, X. and Rani, K. (2017) 'Empowering SDN Firewall against ARP Poison Routing', *International Journal of Applied Engineering Research ISSN*, 12(18), pp. 973–4562. Available at: http://www.ripublication.com.

Benson, T., Akella, A. and Maltz, D. (2009) 'Unraveling the complexity of network management', *6th USENIX Symposium on Networked Systems Design and Implementation*, pp. 335–348. doi: 10.1007/978-1-59745-177-2_17.

Biondi, P. (2019) *Scapy*. Available at: https://scapy.net/.

Brooks, M. and Yang, B. (2015) 'A Man-in-the-Middle attack against OpenDayLight SDN controller', *Proceedings of the 4th Annual ACM Conference on Research in Information Technology - RIIT '15*, pp. 45–49. doi: 10.1145/2808062.2808073.

Cho, I. (2019) *Introduction to Mininet*. Available at: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.

Klöti, R., Kotronis, V. and Smith, P. (2013) 'OpenFlow: A security analysis', in *21st IEEE International Conference on Network Protocols (ICNP)*. Goettingen: IEEE, pp. 1–6. doi: 10.1109/ICNP.2013.6733671.

Kreutz, D. *et al.* (2015) 'Software-defined networking: A comprehensive survey', *Proceedings of the IEEE*, 103(1), pp. 14–76. doi: 10.1109/JPROC.2014.2371999.

Kreutz, D., Ramos, F. M. V. and Verissimo, P. (2013) 'Towards secure and dependable software-defined networks', *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, p. 55. doi: 10.1145/2491185.2491199.

Lantz, B., Heller, B. and McKeown, N. (2010) 'A Network in a Laptop', *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks - Hotnets '10*, pp. 1–6. doi: 10.1145/1868447.1868466.

Li, D., Hong, X. and Bowman, J. (2011) 'Evaluation of security vulnerabilities by using ProtoGENI as a launchpad', *GLOBECOM - IEEE Global Telecommunications Conference*, pp. 1–6. doi: 10.1109/GLOCOM.2011.6134465.

Masoud, M. Z., Jaradat, Y. and Jannoud, I. (2015) 'On preventing ARP poisoning attack utilizing Software Defined Network (SDN) paradigm', *2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies, AEECT 2015*, pp. 0–4. doi: 10.1109/AEECT.2015.7360549.

Nehra, A., Tripathi, M. and Gaur, M. S. (2017) 'FICUR: Employing SDN programmability to secure ARP', *2017 IEEE 7th Annual Computing and Communication Workshop and Conference, CCWC 2017*. doi: 10.1109/CCWC.2017.7868450.

Scott-Hayward, S., O'Callaghan, G. and Sezer, S. (2013) 'SDN Security: A Survey', *SDN4FNS 2013 - 2013 Workshop on Software Defined Networks for Future Networks and Services*, pp. 1–7. doi: 10.1109/SDN4FNS.2013.6702553.

Solomon, N. (2015) *Mitigating Layer 2 Attacks: Re-Thinking the Division of Labor*. The Interdisciplinary Center, Herzliya.

Song, M. S. *et al.* (2014) 'DS-ARP: A new detection scheme for ARP spoofing attacks based on routing trace for ubiquitous environments', *Scientific World Journal*, 2014, pp. 1–8. doi: 10.1155/2014/264654.

Ubaid, F. *et al.* (2017) 'Mitigating Address Spoofing Attacks in Hybrid SDN', *IJACSA) International Journal of Advanced Computer Science and Applications*, 8(4), pp. 562–570. doi: 10.1016/j.cellsig.2015.02.009.