ON PREVENTING ARP POISONING ATTACK UTILIZING SOFTWARE DEFINED NETWORK (SDN) PARADIGM

Mohammad Z. Masoud, Yousf Jaradat and Ismael Jannoud
Computer and Communication Engineering Department
Al-Zaytoonah University of Jordan
130 Amman 11733 Jordan
{m.zakaria, y.jaradat, ismael.jannoud}@zuj.edu.jo

***Abstract*— in this work, SDN has been utilized to alleviate and eliminate the problem of ARP poisoning attack. This attack is the underlying infrastructure for many other network attacks, such as, man in the middle, denial of service and session hijacking. In this paper we propose a new algorithm to resolve the problem of ARP spoofing. The algorithm can be applied in two different scenarios. The two scenarios are based on whether a network host will be assigned a dynamic or a static IP address. We call the first scenario SDN_DYN; the second scenario is called SDN_STA. For the evaluation process, a physical SDN-enabled switch has been utilized with Ryu controller. Our results show that the new algorithm can prevent ARP spoofing and other attacks exploiting it.**

Keywords—Software Defined Network (SDN); Ryu Controller; Address Resolution Protocol (ARP) Poisoning Attack; Network security; Security threats

I. INTRODUCTION

Due to smartphone technology eruption and Internet applications proliferation, network protocols and devices have exploded to cover a vast number of new technologies. Mastering complexity is the term computer network developers and researchers utilize to describe this field. Different protocols have been proposed. Many devices have been designed and manufactured. This complexity revealed many issues in management, security, debugging and compatibility. New protocols and devices have been emerged to tackle these issues in one hand. On the other hand, complexity has escalated.

Software Defined Network (SDN) is defined as a new network architecture that is used to simplify network management and reduce complexity in network technology. In this paradigm, a network controller is considered as a new component in the network. This controller has omnipotent power to manage and program the network. Programming the network is not the same as configuring network devices. In the configuration process, network engineer uses a set of predefined commands to change the settings of the network devices to accomplish certain tasks. On the contrary, network programmability is the process by which a network professional write programs that controls multiple packet-forwarding devices that can be configured utilizing certain protocols such as, OpenFlow. For example, network layer-2 switches can be programmed through the controller to behave as any network devices, such as, firewall or routers. Using this approach, network management and protocol debugging complexity may be reduced. Moreover, many security issues, such as, address resolution protocol (ARP) poisoning can be tackled.

ARP poisoning, or ARP spoofing attack is defined as the ability of intruder to pretend to be gateway of a network. In other words, an intruder became a static route for all of network traffic. This attack is popular since it is easy to implement. ARP spoofing is serious attack since it is considered as the front door of many other network threats and breaches, such as, man-in-the-middle (MiM) attack, eavesdrop, MAC flooding and denial of service (DoS).

In this work, SDN paradigm will be utilized to tackle ARP poisoning attack. A new algorithm that utilizes SDN controller will be proposed. The new algorithm has the ability to stop ARP poisoning in local area networks (LANs) without any enhancement or modification of the classical ARP protocol. The new algorithm is considered as a new application which operates to associate the controller in decision making of forwarding or dropping ARP answers. To this end, an experiment has been conducted utilizing HP-2920-24G SDN-enabled switch and Ryu controller to evaluate the proposed algorithm.

The rest of this paper is organized as follows. Section II overviews SDN and ARP poisoning attack. We end this section by the related works that have been conducted in the area of SDN security. Section III introduces the proposed algorithm. Section IV demonstrated the conducted experiment and the harvested results. We conclude this paper in section V.

978-1-4799-7431-3/15/$31.00 ©2015 IEEE

II. BACKGROUND

This section consists of three main sections. In the first section, SDN paradigm will be introduced. Secondly, ARP poisoning attack will be demonstrated. Finally, SDN security related work will be presented.

A. SDN paradigm

SDN is the new network technology that separated data plane from control plane. Data plane is handled by the underlying hardware that passes flows or blocks it based on the configurations that are manipulated by the controller. However, control plane is handled by Openflow protocol and the controller. To understand these elements, Fig. 1 shows a simple diagram of SDN network topology. In this topology, three different types of hardware's can be observed; switch, server 'controller' and computers 'users' and two different type of links; controller switch link and access links. In the following subsections we will introduce the controller and the controller-switch link since they are the most important parts in this structure.

A.1. Controller

Network controller is the main part of SDN network. It is a server where developers can develop and write their algorithms and topologies. On other words, they program how the network will react to different events. These applications can be written in different programming languages depending on the installed controller. Many controllers have been developed over the years. Table I shows a summary of some of SDN controllers. Many research papers have been conducted in the area of comparing the performance metrics, such as, delay, number of flows, and congestion of these controllers [1].

TABLE I: Examples of controllers

Controller	Development Language
Floodlight [2]	JAVA
OpenIRIS [3]	JAVA
Maestro [4]	JAVA
NOX [5]	C++/Python
POX [6]	Python
Ryu [7]	Python

A.2. Controller-switch link

The second important part of SDN structure is the protocol that runs on the link between the switch and the controller server. This protocol is called Openflow. Openflow defines message format, actions and rules. Before any message passes between the controller and the switch, TCP connection should be established over the controller-switch link. The following steps show how SDN structure operates.

First, the SDN-enabled switch handles messages from any user over any access link. It checks if the new arrived packet matches any flow rules injected in the switch. If the packet matches any flow rule, the switch acts according to the action attached to this rule. Secondly, if no rules have been found for this packet, the switch places the packet in a message and sends it to the controller over the controller-switch link. Thirdly, the controller handles the packet and analyzes it according to the running algorithm. Subsequently, it generates a decision rule for this flow. Finally, the controller sends this new flow rule to the switch to manipulate this type of packets.

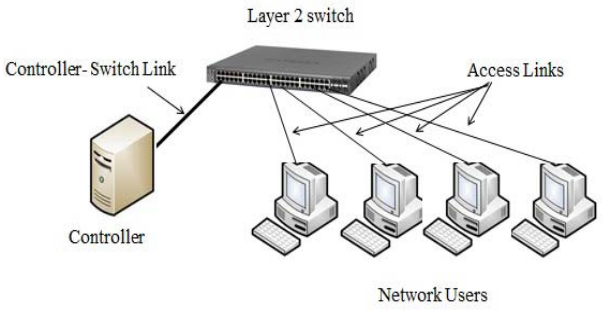

Figure 1: Simple SDN Network Topology

B. ARP Poisoning Attack

Before introducing ARP poisoning attack, ARP protocol should be overviewed. ARP is defined as the mapping protocol that maps an IP address to its convenient MAC or physical address. These MAC addresses are responsible of building the forwarding table. This table is the core of switching process. Each network device has a small ARP table to save the mapped IP-MAC addresses that a device requested. This table is called the ARP cache.

ARP protocol is simple and consists only of four messages; request, reply, reverse request and reverse reply. When a computer needs the MAC address of another machine it broadcasts an ARP request. The machine, which has the IP address, responses with its MAC in ARP reply. The other two massages works in vice versa [8].

ARP protocol contains many issues, such as, authentication and the acceptance of reply even if the host did not send a request. These vulnerabilities introduced the ARP poisoning attack.

This attack is simple, an intruder replies with a broadcast to the network with its MAC address as the physical address of the gateway. Devices change the MAC address of the gateway in their cache into the new MAC address. Subsequently, users start to forward their traffic to the intruder. Finally, the intruder can utilize the received traffic in different ways to generate different types of attacks [8].

Many methods and tools have been proposed to enhance the security of ARP. These methods are divided into two classes. The first class is preventing ARP poisoning. However, this class requires modifications of the classic ARP protocol [9].

The second class is detecting ARP poisoning and then recovery. These methods include DNS detection [10], RTT time method [11] and intelligent approach of reply [12]. However, all of these methods require manipulation or installation of new software's on computers.

C. SDN Security related work

Many research works have been conducted in the area of debugging and securing the control plane of SDN; NetPlumber [13], VeriFlow [14], NICE [15], SOFT [16]. Moreover, researchers have presented and implemented different types of attacks to compromise the structure of the network, such as, FortNox [17] and TopoGuard [18]. Although these conducted researches are about SDN security, however, they did not implement or tackle a data plane attack.

Other SDN research attempted to tackle data plane attacks even by implementing simple firewalls over SDN controller [19], or as the simple firewall that can be found in the example directory of Ryu controller' or as enforcing network policies [20]. These are good examples of security implementation in SDN network. However, these researches work did not attempt to solve the issue of ARP poisoning attack.

III. THE PROPOSED ALGORITHM

The design philosophy of the new algorithm is to prevent ARP poisoning attack utilizing some characteristics of the SDN approach. The first characteristic is the centrality found in SDN so that any application or algorithm will be deployed in the controller, and that will affect all users of the network. The second characteristic is the ability of the SDN approach to prevent ARP spoofing attack without modifying the original ARP protocol. This is guaranteed in SDN structure since we are analyzing packets and adding rules according to their content only. No new field will be added to the ARP messages.

The proposed algorithm consists of two scenarios. The two scenarios are differing by how the IP assignment is done to the hosts. In the first scenario the hosts are assigned dynamic IP addresses utilizing the DHCP protocol; we call this scenario SDN_DYN. In the second scenario the hosts are assigned static IP addresses; we call this scenario SDN_STA. Each scenario consists of two parts. We focused only on IPv4 addresses since the traffic load of IPv6 addresses over the Internet is less than 1% as reported by the collected Internet traffic of [21].

A. SDN_DYN Scenario

SDN_DYN scenario is designed based on DHCP protocol. The first part of the protocol seeks to find the mapped IP-MAC address of the gateway. If the network's users obtain their IP addresses from a DHCP server, SDN controller can inspect DHCP reply messages to obtain the gateway IP address sent to the users in the option field of the message. Subsequently, it maps this IP address to its MAC address from other exchanged messages. The second part is to inspect all ARP replies. If the IP address or the MAC address contained in any ARP reply is reply is not the same as the recorded of the recorded mapped value, the reply will be dropped.

One thing to be mentioned is that, ARP replies should be sent to the controller and their rule should be sent once and not a flow based rule. Alg. 1 shows the pseudo code of this scenario

B. SDN_STA Scenario

In SDN_STA scenario all pairs of MAC-IP addresses of the users connected to a switch are recorded in a table in the controller. The first part of this scenario starts after the forwarding table of the switch is completed. In this way, the controller will contain a list of all MAC addresses in the network. Whenever a host sends any packet, the controller records its IP address and maps it to its MAC address. In the second part, the controller inspects any ARP reply against the recorded pairs.

The two scenarios can be turn on or off. They can work simultaneously or separated from each other.

Algorithm 1: The Proposed Algorithm

```
Xi: new arrived msg from switch i
Ctrl: Controller
Si: switch i
Map: table of mapped pairs
Map(MAC,IP): a pair of mapped MAC-IP
Map(G): the MAC-IP mapped of gateway
Command(x, y): command type x send to switch y
1: If (Xi)
2:     if (ARP reply)
3:         if Match(Map(MAC,IP), Xi) then
4:             command(Once forward, Si)
5:         elf Match(Map(MAC,IP), Xi(IP)) or Match(Map(MAC,IP), Xi(MAC)) then
6:             command(Drop, Si)
7:         else
8:             add(Xi(MAC,IP),MAP)
9:             command(Once forward, Si)
10:        end
11:    if (DHCP reply) then
12:        temp=option(getaway IP)
13:        Map(G)=Match(MAP(MAC,IP),temp)
14:        command(Once forward, Si)
15:    else
16:        add(Xi(MAC,IP),MAP)
17:        command(Once forward, Si)
18:    end
19: end
```

IV. EXPERIMENT

To evaluate the performance of the new algorithm, a testbed is implemented. Figure 2 shows the experiment environment.

The experiment consists of HP-2920-24G SDN-enabled switch, 4 computers and an access point, that is used as a DCHP server for the SDN_DYN scenario. The computers specifications are 1.8 GHz i5 core; 4G RAM and are equipped with NIC cards. Ubuntu 14 OS has been installed on one of these computers a long with the Ryu controller. This PC has been utilized as a controller and has been connected to the *mgmt* port of the switch. The second and the third PC's have windows 7 OS installed. Finally, the fourth PC has Kali Linux OS installed. These three PC's have been connected as clients to the network.

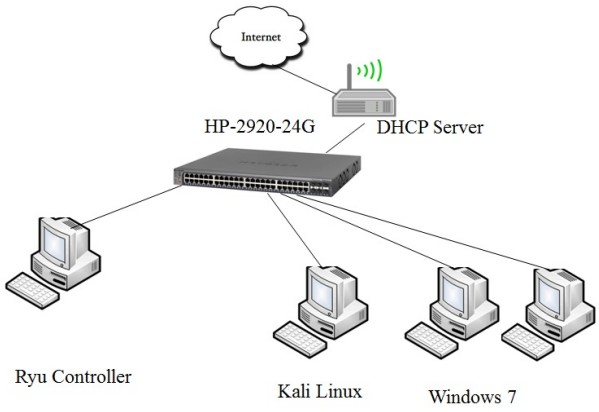

Figure 2: Experiment Environment

Python language has been utilized to program the proposed algorithm. This is one of the reasons to select Ryu SDN controller. The second reason is that Ryu controller is well-documented controller unlike most of the other controllers.

To write the proposed algorithm, the application program first should consist of a complete switch code; forwarding tables, STP and Openflow connection. Secondly, the proposed algorithm can be plugged to the switching code. Openflow v1.1 has been used since we have not upgraded the switch framework.

The experiment consisted of three scenarios according to the mode of operation of the switch, these modes are: normal switch, SDN switch and SDN switch with the proposed algorithm. In normal switching mode, the controller has no control over the switch and the switch will operate as usual. By using the normal mode of the switch we attempted to prove the simplicity and the harm caused by ARP poisoning attack.

The first scenario (normal switch mode) is started by surfing the Internet from one of Windows 7 PCs. The second Windows 7 PC will be used as an intruder. Cain & Abel software [22] has been downloaded and installed. This software can implement ARP poisoning attack with one click. Subsequently, Wireshark [23], a network analyzer, has been downloaded and installed. After implementing the attack, Wireshark has successfully captured all packets of the network. This attack is known as eavesdropping.

Now we utilized Kali Linux as the intruder. Ettercap [24] has been utilized to sniff the data after implementing ARP attack. We also could manipulate network packets to generate DoS attack and MiM.

The second scenario (SDN switch mode) consisted of the same parts as the first scenario. However, the switch has been configured to operate as SDN switch and a simple switching topology has been executed over Ryu controller. The results of this scenario were identical to the first scenario

Finally, in the third scenario (SDN switch with the proposed algorithm) we ran the enhanced simple switch topology on Ryu controller. We utilized the two previous intruders. We attempted to generate ARP poisoning attack. However, we failed to do so. Table II shows a summary of the three scenarios

TABLE II: Results of Experiment's Scenarios

Switch mode	Intruder	Result of ARP Poisoning
Normal switch	Windows 7	Pass
	Kali Linux	Pass
SDN simple switch	Windows 7	Pass
	Kali Linux	Pass
SDN switch with the proposed algorithm	Windows 7	Failed
	Kali Linux	Failed

To evaluate the DHCP part of our protocol (SDN_DYN), we added a DHCP server into our environment by adding an access point with DHCP capability. The controller has recorded the MAC-IP paired map in all attempts, and thus the ARP spoofing attack has failed.

V. Conclusion

SDN is a new computer network approach that introduces a controller as a new omnipotent component that has a complete or general view of the network. Moreover, this controller has the ability to program the underlying network devices. In this work, SDN approach has been utilized to tackle ARP cache poisoning attack 'ARP poisoning' in LANs. A new algorithm has been proposed and implemented to prevent this attack. An experiment has been conducted to evaluate the proposed algorithm. HP2920-24 SDN-enabled switch and open-source Ryu controller has facilitated our experiment. Our result demonstrated that the new algorithm prevented ARP poisoning and many breaches and attacks utilizing ARP poisoning.

VI. Acknowledgment

This research was supported in part by Al-Zaytoonah University of Jordan fund (2/11/2014). We would like to thank the University for the equipment and tools they provided for this work.

REFERANCES

[1] Benamrane, Fouad, Mouad Ben Mamoun, and Redouane Benaini. "Short: A Case Study of the Performance of an OpenFlow

Controller." *Networked Systems*. Springer International Publishing, 2014. 330-334.

[2] Wallner, Ryan, and Robert Cannistra. "An SDN approach: quality of service using big switch's floodlight open-source controller." *Proceedings of the Asia-Pacific Advanced Network* 35 (2013): 14-19.

[3] Lee, Byungjoon, et al. "IRIS: The Openflow-based Recursive SDN controller." *Advanced Communication Technology (ICACT), 2014 16th International Conference on*. IEEE, 2014.

[4] Maestro, http://code.google.com/p/maestro-platform/, visited in 2/2015 online

[5] Yeganeh, Soheil Hassas, Amin Tootoonchian, and Yashar Ganjali. "On scalability of software-defined networking." *Communications Magazine, IEEE* 51.2 (2013): 136-141.

[6] POX SDN controller, http://www.noxrepo.org/pox/about-pox/, visited 2/2015 online

[7] Ryu SDN controller, http://osrg.github.io/ryu/, visited in 1/2015 online

[8] Nachreiner, Corey. "Anatomy of an ARP poisoning attack." *Retrieved July* 4 (2003): 2005.

[9] Nam, Seung Yeob, Dongwon Kim, and Jeongeun Kim. "Enhanced ARP: preventing ARP poisoning-based man-in-the-middle attacks." *Communications Letters, IEEE* 14.2 (2010): 187-189.

[10] AbdelallahElhadj H., Khelalfa H., and Kortebi H., "An Experimental Sniffer Detector: SnifferWall," *Technical Document*, Securie des Communications sur Internet, 2002.

[11] Trabelsi Z., Rahmani H., Kaouech K., and Frikha M., "Malicious Sniffing Systems Detection Platform," *in Proceedingss of IEEE/IPSJ International Symposium on Applications and the Internet*, Tunisia, pp. 201-207, 2004.

[12] Khan, Abdul Nasir, Kalim Qureshi, and Sumair Khan. "An intelligent approach of sniffer detection." *Int. Arab J. Inf. Technol.* 9.1 (2012): 9-15.

[13] P. Kazemian, M. Chang, H. Zeng, S. Whyte, G. Varghese, and N. McKeown. Real time network policy checking using header space analysis. In Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2013.

[14] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey. Veriflow: Verifying network-wide invariants in real time. In Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2013.

[15] P. Peresini D. Kostic M. Canini, D. Venzano and Jennifer Rexford. A nice way to test openflow applications. In Proceedings of the 9thUSENIX Symposium on Networked Systems Design and Implementation (NSDI), 2012.

[16] M. Kuzniar, P. Peresini, M. Canini, D. Venzano, and D. Kostic. A soft way for openflow switch interoperability testing. In Proceedings of ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2012.

[17] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu. A security enforcement kernel for openflow networks. In Proceedings of ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'12), August 2012.

[18] Hong, Sungmin, et al. "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures." NDSS, 2015.

[19] Suh, Michelle, et al. "Building firewall over the software-defined network controller." *Advanced Communication Technology (ICACT), 2014 16th International Conference on*. IEEE, 2014.

[20] Qazi, Zafar Ayyub, et al. "SIMPLE-fying middlebox policy enforcement using SDN." *ACM SIGCOMM Computer Communication Review*. Vol. 43. No. 4. ACM, 2013.

[21] Pujol Enric, Richter Philipp, Chandrasekaran Balakrishnan, Smaragdakis Georgios, Feldmann Anja ,Maggs Bruce MacDowell, Ng KeungChi "Back-Office Web Traffic on The Internet", Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14, 2014

[22] Cain & Abel, http://oxid.it , visited in 2/2015 online

[23] *Wireshark,* https://www.wireshark.org/, visited in 2/2015 online

[24] *Ettercap, https://ettercap.github.io/ettercap/, visited in 2/2015online*