Mitigating ARP Spoofing Attacks in Software-Defined Networks

Ahmed M.AbdelSalam, Ashraf B. El-Sisi
Faculty of Computers and Information,
Menoufia University.
Menoufia, Egypt
ahmed.abdelsalam@ci.menofia.edu.eg
ashraf.elsisi@ci.menofia.edu.eg

Vamshi Reddy K
Department of Computer Science & Engineering,
Manipal Institute of Technology.
Manipal, India
k.vamshi2008@gmail.com

Abstract— **Software-Defined Networks (SDN) are emerging as an attractive solution to overcome the limitations of traditional networks. They provide network programmability and promote rapid innovation in protocol design, network management and network security. Today, network security is the most important concern for any computer network administrator. Traditional networks had several security problems, some of them no longer exist with the advent of SDN and others still do exist. ARP spoofing or ARP cache-poisoning attack is mainly seen in LAN networks, which has no efficient solution to mitigate in traditional networks but SDN provides a unique way to solve this problem without any changes in the network. ARP spoofing is exploited in different forms, mainly request and reply based attacks. In this paper, a solution is proposed to mitigate both of these types of ARP attacks in Software defined networks. The proposed solution extends the SDN controller to include an ARP module, which detects and stops the attack, and avoids overloading or Denial of Service on the controller. The solution is fast, reliable and tested for different attack scenarios. Openflow is used for the communication between the controller and switches, emulated by Mininet.**

Keywords: Software-Defined Networking; ARP; ARP spoofing mitigation; Security.

I. INTRODUCTION

Traditional computer networks are built from large number of end devices like PC's, Servers, Printers, IP Phones and intermediate devices such as hubs, switches, routers, access points and firewalls. These intermediate devices mainly forward data from one end device to another. They incorporate special algorithms to build their forwarding tables and forward the traffic accordingly. They are vertically integrated: the control plane and date plane are coupled together. Control plane has the logic to handle the traffic and the data-plane forwards the traffic based on the decisions made by control-plane. Devices operating in traditional networks have many proprietary features that are not easy to configure and may lead to some incompatibility between devices from different vendors. In addition, traditional networks have become very hard to manage in a large scale like Datacenters. They require network engineers for configuration and maintenance. Consequently, this increases the operational cost of a network. Network protocols are also hard to evolve, they require years of testing and standardization before being used into production network. Moreover, they have no built in security features.

Software-Defined Networking (SDN) is a new paradigm in networking that changed the way the traditional networks are built and managed. It aims to solve most of the limitations in traditional networks by decoupling the control plane from the data plane. In SDN, network devices have become simple forwarding devices and only implement the data plane logic. The control or network intelligence is implemented in a centralized controller, as the controller is the network brain. SDN will shape networking by giving the network owner or operator the ability to customize and optimize their networks. It speeds up the innovation in computer networking at software development speeds rather than a slow hardware driven approach present in the traditional networks.

Traditional networks have several security problems, as they are not designed with security as the main criteria. With the increase in the number of nodes connected to a network, security becomes the most important factor for the successful communication. With emergence of SDN, most of these problems no longer exist but there are some, which threaten even today's networks. Spoofing attacks were first discovered in early traditional networks, they cannot be mitigated with a plain SDN controller. When a user impersonates another user's identity on a network it will be considered a spoofing attack. By using this attack, an attacker can steal other user's data or bypass the access control. IP spoofing, DNS spoofing, and ARP spoofing are different forms of spoofing attacks.

ARP spoofing is the most common form of spoofing in Local Area Networks (LAN). Section II of this paper will present the details of ARP Protocol and ARP spoofing attacks. The efforts previously done to mitigate ARP spoofing Attacks in traditional networks will be examined in section III. In Section IV, the proposed solution to ARP spoofing problem in SDN will be shown. Furthermore, in Section V, we have tested the proposed solution in different scenarios to show that the attack can be detected and completely mitigated. In Section VI, the results for the conducted experiments are discussed. Finally, in Section VII, the paper is concluded with further modifications, extensions and enhancements to the proposed architecture.

II. ARP Spoofing

ARP [1] protocol is specified in RFC 826 [2]. It is crucial for successful communication in Computer Networks. Sending devices use ARP protocol to know the MAC address of the next hop device in path from the sender to receiver. Every host in the network has its own ARP cache that contains the recent IP to MAC bindings. These mappings can be learnt dynamically by ARP protocol or can be added manually as static rules. If two hosts *host_A* and *host_B* want to communicate, *host_A* searches in its ARP cache for the next hop's MAC address. If search is successful, then it uses the returned MAC address, or else, ARP protocol will start to operate for it. In ARP protocol, *host_A* sends an ARP request for the next hop IP by broadcasting its request into the network. *host_C*, which is one hop away from *host_A* in the path to *host_B*, replies with an ARP reply message. *host_A* updates its ARP table with the next hop IP, MAC pair sent by *host_C* and subsequent IP packets use this cached entry present in the table. One more consequence of ARP protocol is that *host_C* also adds an entry for *host_A* IP-MAC in its ARP cache table.

ARP spoofing is a technique used by attackers to perform cache poisoning by inserting false IP to MAC address mappings in victim's ARP cache. ARP spoofing attack comes in different forms namely request and response attacks. In request attack, an attacker broadcasts ARP request message with forged source IP-MAC in the ARP header. When victim receives this spoofed ARP message, it updates its ARP cache table with the attacker's forged IP-MAC pair. When the victim sends its subsequent packets destined to the forged IP, the packets will be destined to the MAC of the host specified by the attacker in the forged IP-MAC pair. This way attacker can intercept or deny the traffic sent to a user in the network. The other form of attack is the response attack, in which the attacker will either respond to normal ARP request with forged ARP replies that maps the next hop IP to the attacker MAC address or send spoofed ARP replies without having requests being issued. ARP spoofing may be used to launch either one of the following attacks [3]:

1. *DoS attacks:* the attacker will prevent the two communicating hosts from getting connected to each other.
2. *Host impersonation attack*: the attacker will receive packets intended to the victim and can reply to these packets on behalf of the victim.
3. *Man-In-The-Middle (MITM) attack*: the attacker will be able to monitor all the traffic between two communicating hosts.

Due to the popularity, importance, and danger of the ARP spoofing attacks, various tools have been implemented to launch this attack. These tools are available on most of the operating systems. Ettercap [4], Dsniff [5], Yersinia [6] and Cain and able [7] are the most popular tools used to launch ARP spoofing attacks. They can also be used to execute other data link layer attacks such as MAC flooding attacks, STP attacks, DHCP attacks, and VLAN attacks.

III. Mitigating ARP Spoofing Attacks in Traditional Networks

The several solutions proposed to solve ARP spoofing problem in traditional can belong to either one of the following categories [8]:

1) ARP authentication: ARP protocol will be modified in order to not process ARP packets unless it contains a predefined cryptographic code. S-ARP [9] and TARP [10] are examples of these techniques. But this method leads to some incompatibility with those devices that still operate using the standard ARP protocol.

2) Operating system patching: The operating system will be patched to prevent ARP spoofing. However, this technique require a different patches for different operating systems kernels. Moreover, the patched operating system may not communicate correctly with other operating systems that might not have been patched. Anticap [11] and antidote [12] are the solutions for ARP spoofing problems, that patch the operating system kernel.

3) Dynamic ARP Inspection (DAI): Some switches (e.g., Cisco catalyst) are equipped with a security feature called Dynamic ARP Inspection (DAI) [13]. Switches use DAI to stop ARP spoofing attacks by intercepting all ARP requests and responses and dropping packets with invalid IP-to-MAC address mappings. The main problem of this solution is that not all switches are equipped with the DAI feature. It also not viable for most organizations to buy new switches equipped with the DAI feature instead of the old ones.

4) ARP mitigation tools: There are several software tools that have been developed to mitigate the ARP spoofing attacks. Some of these tools prevent the attack and others only detect the attack. XArp [14], ARPWatch [15], Anti Netcut [16], NoCut [17], and AntiARP [18] are example of these software tools. However, these have many problems such as high rate of false positives and false negatives and ineffectiveness.

5) Static ARP mappings: The most effective way to prevent ARP spoofing attacks is to configure the ARP entries manually. This way, the attacker will not have any chance to spoof the other host's ARP cache. But, this solution is not viable for large scale networks. It can't be used in networks that use DHCP addressing schemes and in Datacenters, where there is a frequent change in IP addresses. In addition, it will be error prone and requires a great overhead on the network administrator. Several solutions have been proposed in attempt to ovecome these drawbacks. DAPS [8], NIDPS [3], and the techniques proposed in [19], [20], [21] are examples of solutions that make use of static ARP entries. Each of these techniques have their own limitations.

IV. Mitigating ARP spoofing attacks in Software Defined networks

Network security in SDN can be improved either by using SDN's features and capabilities or by solving the security problems of SDN itself [22]. SDN gives us the ability to gather the required data from the network and provides a way to analyze and detect the attack signature. After the analysis, network can also be reprogramed for enforcing any security policy. The proposed method is an application over SDN POX controller written in python; it will prevent LAN attackers from poisoning the ARP cache tables of other hosts present in the network. This application does the following:

- It detects and mitigates ARP request and ARP reply spoofing attacks.
- It does port level ARP packet count monitoring, to detect large influx of malicious packets and stops them by installing Openflow rules.
- It avoids controller overloading by installing flow entries on switches to drop packets coming towards the controller, in case the attacker tries to bring down the controller.
- It differentiates edge switches (where security policy enforcement should be applied) from the other intermediate switches. (Where usual forwarding logic is applied).

The flowchart of the proposed application is shown in Figure 1. The proposed application initializes the L2 learning switch modules present on the controller for each of the switches present in the network. It then starts the DHCP server for serving the lease requests by the hosts on the network. DHCP server is required to be on the controller, as the controller will keep track of the IP leases and uses them later in detecting the attack. Edge switches have the hosts connected to them

Then, the controller starts monitoring on each of the ports on the switches connected to it. It also installs flow entries on the edge switches to forward ARP and DHCP lease packets to it for analysis. When the controller receives a packet from the edge switch, it will process according to the protocol present in the packet. ARP and DHCP packets are handled according to the proposed logic to detect any attack. If a packet comes from an intermediate switch, controller does plain L2-L3 processing on the packet. The following classification is done to describe how the DHCP and ARP packets are handled.

A. DHCP packets

The DHCP protocol is used to dynamically assign IP and other network configuration parameters to the hosts. A DHCP client will exchange messages with the DHCP server present on the controller to get IP address and other parameters. The controller extracts the IP and MAC addresses from the DHCP header and uses them to add an entry into the known hosts list. This known hosts list will keep track of all the IP addresses leased by the DHCP server.

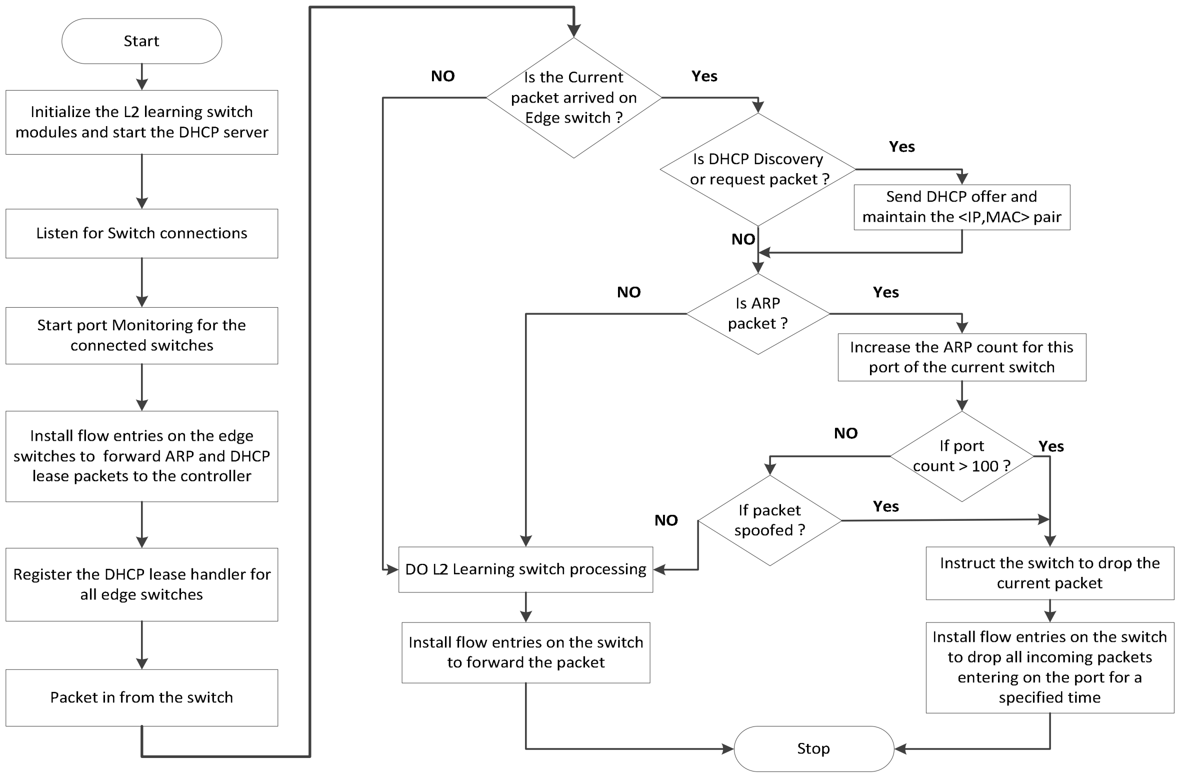
Start
Initialize the L2 learning switch modules and start the DHCP server
Listen for Switch connections
Start port Monitoring for the connected switches
Install flow entries on the edge switches to forward ARP and DHCP lease packets to the controller
Register the DHCP lease handler for all edge switches
Packet in from the switch
Is the Current packet arrived on Edge switch ?
NO
Yes
Is DHCP Discovery or request packet ?
Yes
Send DHCP offer and maintain the <IP,MAC> pair
NO
Is ARP packet ?
NO
Yes
Increase the ARP count for this port of the current switch
If port count > 100 ?
NO
Yes
If packet spoofed ?
NO
Yes
DO L2 Learning switch processing
Install flow entries on the switch to forward the packet
Instruct the switch to drop the current packet
Install flow entries on the switch to drop all incoming packets entering on the port for a specified time
Stop

Fig. 1: ARP spoofing mitigation flowchart

B. ARP packets

The controller handles ARP packets based on the packet type namely request or reply.

1) ARP request packets: ARP request packet will be considered spoofed, if it satisfies one of the following conditions

- Source MAC address of ethernet header and source MAC address of ARP header are not the same.
- Destination IP address in the ARP header is not present in the known hosts list of the DHCP server.
- The MAC address binding present in the known hosts list for the Source IP of the ARP header doesn't match with the source MAC address of the ARP header.

2) ARP reply packets: the ARP reply packet will be considered spoofed, if it satisfies one of the following conditions

- Source MAC address of ethernet header and source MAC address of ARP header are not the same.
- Destination MAC address of ethernet header and destination MAC address of ARP header are not the same.
- The MAC address binding present in the known hosts list for the Source IP of the ARP header doesn't match with the Source MAC address of the ARP header.
- The MAC address binding present in the known hosts list, for the Destination IP of the ARP header doesn't match with the Destination MAC address of the ARP header.
- Destination MAC address of the ethernet header has a value of (FF:FF:FF:FF:FF:FF).

If any of these stated conditions is true, the controller knows that this is an attack signature, installs a flow entry on the forwarding device, to stop the incoming malicious traffic.

If an attacker knows the current algorithm that the ARP packets are sent to the controller for inspection, then he can simply send large number of ARP packets to perform DOS attack. To avoid such problem, the controller in this proposed solution keeps track of the number of received ARP packets on a certain switch port by using the port-monitoring algorithm. This algorithm collects the ARP packet count at successive intervals. Port monitoring algorithm is run on every port present on all of the edge switches that keeps track of the ARP packet counts for every second. At any time, if controller witnesses a spike in the ARP packet count for a port, it installs a flow entry on the switch to stop packets coming from that port for a specified amount of time. This prevents the attack packets entering into the network and protects the controller.

V. IMPLEMENTATION

Experiments are conducted for various attack scenarios to prove the feasibility of the proposed solution. The details of these experiments are shown in this section.

A. Test Enviroment

The experiment is conducted on an Ubuntu Virtual Machine (VM) with 2 cores and 2 GB of RAM. The host machine is running on Windows 8 and has Intel Core i5 processor with 4 GB of RAM, 4x1 Intel Core i5 cores with Intel VT enabled.

Mininet [23], a network emulator is used for creating the testbed. It creates a virtual network of hosts, switches, controllers and links according to a given topology. Hosts and switches created by Mininet have Linux OS and Open Virtual Switch software (OVS) respectively. These switches support Openflow and are orchestrated by a SDN controller. The Mininet emulator is installed on Ubuntu 14.10 Linux virtual machine (VM) that runs on VirtualBox. The VM has two network adapters, Network Address Translation (NAT) adapter and Host-only adapter. The NAT adapter allows the VM to connect to the internet for downloading the required software packages. The Host-only adapter is used for the communication between the VM (guest machine) and the physical machine (host machine). POX [24] is used as the SDN controller. DHCP server is also run on the POX controller as a service, with a predefined lease IP range to serve the clients on the network. The proposed solution is run as a module on the POX controller for the tests.

B. Experiment Setup

The topology used to test the proposed solution is shown in Figure 2. The topology consists of three Openflow OVS switches (S1, S1, and S3). Switches S1 and S2 are edge switches and S3 is intermediate (core) switch. These switches are connected in a tree topology. It has 4 end users (H1, H2, H3, and H4), three of them (H1, H2, and H3) are normal users and the fourth one (H4) acts as an attacker. This topology also has a single controller (default POX controller with the proposed ARP spoof detection module), created by the Mininet emulator. All the emulated links have 100 Mbps bandwidth with 5ms delay, 0% loss and with a maximum packet queue size of 1000. Every host in the emulated topology has the same CPU-performance.

C. Test Scenario

Three different attack scenarios are created to verify the attack mitigation by using the proposed algorithm. In order to execute the tests, an ARP spoofing software (C code) called *arp-spoof* is developed by using UNIX raw sockets, to generate spoofed ARP request and reply traffic. ARP request attack, ARP reply attack, and DoS attack against the SDN controller are the attack scenarios tested in this section.

Tests are performed on each of the attack scenarios to evaluate attack detection time, attack mitigation time, load on the controller and throughput of the network. *Iperf*, a tool, which collects statistics about the network connectivity and throughput of the path between two hosts, is used. Packet loss and throughput statistics between two non-malicious hosts in the network are collected while an attacker is executing his ARP cache-poisoning exploit. At the same time the other statistics like Attack detection and mitigation time are tabulated.

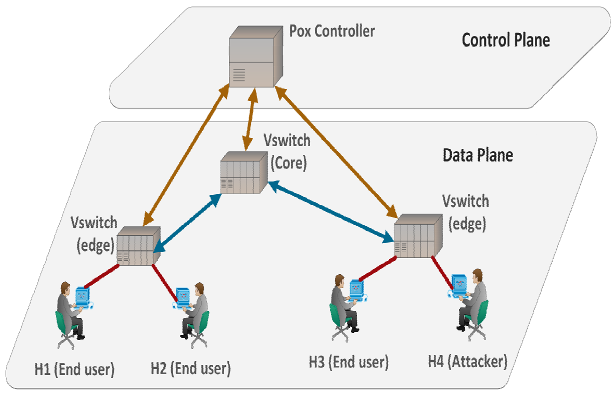

Fig. 2: Test Experiment Setup

For example, the attacker 'Trudy is the one, sending spoofed ARP packets, and the victim 'Alice' is the one, which gets affected by the Trudy's packets. Trudy spoofs the identity of another host 'Bob' with its own MAC address and poisons Alice's ARP cache, to prevent Alice's communication with Bob.

Attack detection time is the time elapsed between the initiation of an attack by the attacker and the detection of the attack by the controller. *Attack mitigation time* is the time between the detection of the attack by the controller and the installation of the flow rules on the switch. Load on the controller is the process load on the CPU.

D. *ARP request attack*

The first form of ARP spoofing attack is the ARP request attack. This is exploited by sending a storm of spoofed ARP requests to into the network to poison the victim's ARP cache. To simulate this, the *arp-spoof* is run with the request attack option selected. This will inject the packets into the network at a rate of 1000 packets/second. Now, the victim host in the network will accept the attacker's spoofed packet and fills its ARP cache table with a wrong IP-to-MAC address mapping of the spoofed identity. The victim host will not be able to communicate with others in the network until its cache gets refreshed. When we run the proposed mitigation solution on the controller, it will detect the request attack at the switch on which the attacker is connected and installs a flow entry to drop packets coming from the attacker's port. This way, attacker's packers are filtered at the switch nearest to him, thereby protecting the entire network.

E. *ARP reply attack*

ARP reply attack is another type of spoofing attack similar to the request attack. Here, the cache of the victim is poisoned by the unsolicited ARP replies sent by an attacker with a spoofed identity.

F. *Denial of service against SDN Controller*

Denial of service attack (DOS) is not a form of ARP spoofing but always comes in conjunction with ARP attacks. This happens when the attacker tricks the proposed algorithm by sending a continuous burst of non-spoofed ARP packets. According to the algorithm, the ARP packets are diverted to the controller by the switches. Controller analyzes these incoming packets and does nothing but forwarding; this may bring the controller down as the controller is oblivious to the fact that attacker can overhaul it by sending a large number of good ARP packets. This problem is solved in the proposed solution by introducing *Port Monitor* module that counts the number of ARP packets received on every switch port. If this number reaches a pre-defined threshold (threshold is 100 in this scenario), the controller installs a flow entry at the switch to drop traffic from this port for a pre-defined amount of time. To test this, we have used the same tool *arp-spoof* to generate good ARP packets with a high rate and injected them into the network. POX controller running the proposed algorithm detects that this packet flow is abnormal and stops it from entering into the network.

VI. Performance Evaluation

Attack detection time and mitigation time for the topology shown in Figure 2 are measured and shown in the Figure 3.

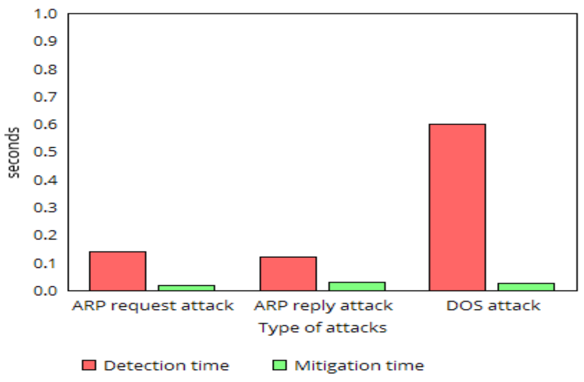

Fig. 3: Attack detection and mitigation time

Throughput test is also performed for the topology shown in Figure 2. Link bandwidth of 100 Mbps is assumed for all the links in this test. Figure 4 shows the graph with the various values of TCP connection throughput before, during and after the attack.

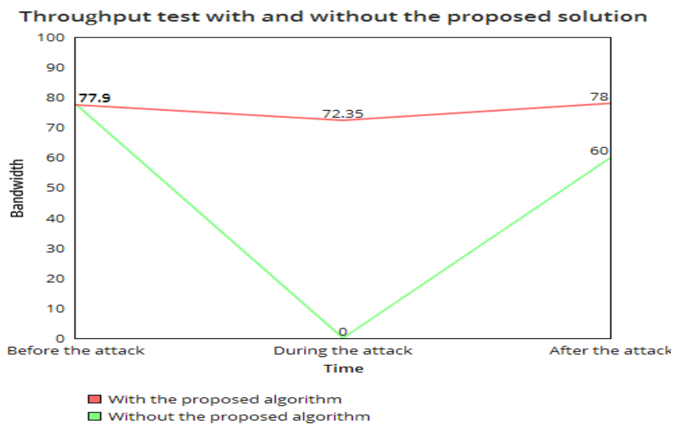

Fig. 4: Throughput test between the victim and a host.

Load on the controller for the two attack scenarios are shown in the Figure 5 below. Firstly, in the request ARP spoofing attack, the controller as soon as it detects an attack, installs a flow entry on the switch to drop all packets at the switch level, thereby avoiding any load. Lastly, DOS attack with the non-malicious ARP packets; controller uses the port monitor to prevent any load. The results are quite impressive showing minimal or no overhead on the controller.

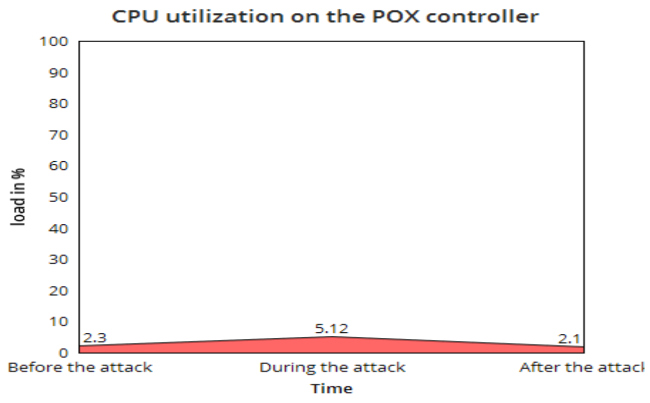

Fig. 5: CPU Utilization on the POX controller.

VII. Conclusion and Future work

In this paper, a solution to ARP spoofing problem in Software-Defined Networks is presented. We started with a discussion of the main differences between traditional and Software-Defined Networks and then focused on security concerns for these networks. The proposed solution makes use of the features of SDN to reliably mitigate both ARP request and reply attacks with minimum latency. This solution uses port level ARP packet monitoring to prevent Denial of Service (DoS) attacks against the controller and doesn't have any extra overhead in the network. It also requires no infrastructure changes, when compared to other traditional ARP spoofing solutions.

This solution is currently described for a single controller in a LAN network, it can be extended to handle ARP attacks in a multiple controller setup with high availability. The proposed solution can be extended not only to mitigate ARP spoofing attacks but all kinds of data link layer attacks. A survey can be made to research on how to use SDN features to provide a complete solution to prevent most of the Spoofing attacks.

References

[1] W. R. Stevens, TCP/IP Illustrated, Vol. 1: The Protocols, Addison-Wesley Professional Computing Series, 1994.

[2] D. C. Plummer, "An Ethernet Address Resolution Protocol," November 1982. [Online]. Available: https://tools.ietf.org/html/rfc826. [Accessed 01 August 2015].

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

[4] A. Ornaghi and M. Valleri, "Ettercap," [Online]. Available: http://ettercap.github.io/ettercap. [Accessed 01 August 2015].

[5] D. Song, "DSniff," [Online]. Available: http://naughty.monkey.org/dugsong/dsniff. [Accessed 01 August 2015].

[6] "Yersinia hacking tool," [Online]. Available: http://yersinia.sourceforge.net. [Accessed 01 August 2015].

[7] "Cain and able," [Online]. Available: http://www.oxid.it/cain.html. [Accessed 01 August 2015].

[8] Puangpronpitag, S. and Masusai, N., "An efficient and feasible solution to ARP Spoof problem," in *6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, Chonburi, 2009.

[9] Bruschi, D. , Ornaghi, A. and Rosti, E., "S-ARP: a secure address resolution protocol," in *19th Annual Computer Security Applications Conference*, 2003.

[10] Wesam Lootah , William Enck and Patrick McDaniel , "TARP: Ticket-based address resolution protocol," *Computer Networks: The International Journal of Computer and Telecommunications Networking,* vol. 51, no. 15, pp. 4322-4337 , 2007.

[11] M. Barnaba, "anticap," Antifork, [Online]. Available: https://antifork.org/git/anticap. [Accessed 01 August 2015].

[12] I.Teterin, "Antidote," [Online]. Available: http://www.securityfocus.com/archive/l/299929. [Accessed 01 August 2015].

[13] "Dynamic ARP Inspection," Cisco Systems, [Online]. Available: http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/dynarp.html.. [Accessed 01 August 2015].

[14] "XArp - Advanced ARP Spoofing Detection," [Online]. Available: http://www.xarp.net/. [Accessed 01 August 2015].

[15] "arpwatch," Lawrence Berkeley National Laboratory, August 2009. [Online]. Available: ftp://ftp.ee.lbl.gov/arpwatch.tar.gz. [Accessed 01 August 2015].

[16] "Anti netcut version 2.0," [Online]. Available: http://www.tools4free.net. [Accessed 01 August 2015].

[17] A. Ali, "NoCut 1.001a," [Online]. Available: http://www.download.com/NoCUT/3000-2085_410520090.. [Accessed 01 August 2015].

[18] ColorSoft, "AntiARP," [Online]. Available: http://www.antiarp.com/English/e_index.htm. [Accessed 01 August 2015].

[19] X. HOU, Z. JIANG and X. TIAN, "The detection and prevention for ARP Spoofing based on Snort," in *International Conference on Computer Application and System Modeling (ICCASM 2010)*, 2010 .

[20] A. P. Ortega, X. E. Marcos, L. D. Chiang and C. L. Abad, "Preventing ARP Cache Poisoning Attacks: A Proof of Concept using OpenWrt," in *Latin American Network Operations and Management Symposium*, 2009.

[21] A.-z. Qian, "The Automatic Prevention and Control Research of ARP Deception and Implementation," in World Congress on Computer Science and Information Engineering, 2009.

[22] Diego Kreutz, Fernando M. V. Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky and Steve Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE ,* vol. 103 , no. 1, pp. 14 - 76, 2015.

[23] B. Lantz, B. Heller and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010.

[24] NoxRepo.org, "About POX," [Online]. Available: http://www.noxrepo.org/pox/about-pox/. [Accessed 01 August 2015].