International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 18 (2017) pp. 7466-7469

Empowering SDN Firewall against ARP Poison Routing

Deepa Balagopal

Research Scholar, Department of Computer Applications,
Karpagam Academy of Higher Education, Eachanari, Coimbatore– 641 021, India.
Orcid Id: 0000-0002-1717-5734

X.Agnise Kala Rani

Professor, Department of Computer Applications,
Karpagam Academy of Higher Education, Eachanari, Coimbatore– 641 021, India.
Orcid Id: 0000-0002-1465-0595

Abstract

The Software defined paradigm emerged as a consequence of the complexity involved in the establishment and maintenance of the traditional computer networks. With the Controller as the main player, the paradigm attempts to convert the switches into mere forwarding devices. But, the de-coupling of the control plane and the data plane exposes Software defined networks to numerous security threats – ARP poison routing is one among them. In this paper, we present our restrictive firewall which has the added capability to identify spoofed ARP packets. The module detects the poisoned packets, analyses them and identifies the victim and the attacker. It then provides suitable alert to the network administrator while diverting the poisoned packets from the network.

Keywords: SDN, Firewall, ARP Poisoning, Spoofing, POX.

INTRODUCTION

The Software defined network paradigm is an attempt at making the otherwise rigid computer networks programmable and flexible. In addition to network virtualization and dynamic network policy management, it provides a greater control over the network devices at a reduced operational cost. On the other hand, the new architecture also requires the old networking software and applications be rewritten to fit in with the current scenario.

With SDN, there lies a greater responsibility on the administrator to ensure the security of the network not only from rogue elements outside, but also from within. Insider threat is a serious security concern since the Controller is a powerful entity within the SDN [1]. For instance, a host can pretend to be someone else and flood the network with fake packets thus affecting its performance and eating up the bandwidth. The presence of such malicious end-hosts can cause havoc to a network. Though such attacks are similar to those in traditional networks, a different solution needs to be applied due to the difference in architecture [2].

In [3] the researchers proposed a firewall application that can restrict communication between devices within a network by enabling the switches as filters. This application successfully blocks any interaction between hosts unless otherwise specified. With the intention of strengthening the firewall, the researchers ran some sample tests of Address Resolution Protocol ARP) Poisoning and identified that the SDN controller was vulnerable in spite of the presence of firewall and it was indeed possible for an existing host to turn malicious and launch such an attack on its peer. To counter this, the NetWatch application has been modified and the functionality to detect ARP poisoning and the attacker has been added. Once the attacker is identified, NetWatch prevents the attacker from flooding the victim with fake request or reply packets and prompts an alert so that the attacker can be nullified.

ARP POISONING

The ARP protocol operates below the internetwork layer and is used to map the MAC address of the host with the IP address. The ARP packet structure is made up of the Sender Hardware Address (SHA), the Target Hardware Address (THA), the Sender Protocol Address (SPA) and the Target Protocol Address (TPA). SPA & TPA fields contain the IP address values of the sender and receiver. MAC addresses are obtained from SHA and THA fields.

Within a network, the ARP request packets are broadcasted by a source and the corresponding MAC is obtained from the intended host via an ARP reply packet. The host devices within the network maintain an ARP cache (dynamic or static) consisting of these address mappings and this is updated periodically based on the reply packets received without any authentication.

The absence of authentication is the fundamental problem leading to ARP attack and flooding [4]. In this form of attack, a malicious host can corrupt the ARP cache of another device thus affecting the communication within the network. This is

a common problem that small and large networks face. ARP attacks target specific hosts by using their MAC address and fake responses on their behalf, at the same time flooding the network with ARP requests. The flooding results in slow communication on the network especially with the host that is being targeted by the attack.

Several solutions for overcoming this security threat have been proposed for traditional networks [5]. For instance, Dynamic ARP Inspection could be used to keep an eye on all DHCP messages [6] or static ARP cache entries can be set up using software like Arpwatch [7].

Unfortunately these solutions cannot be put to use in SDN networks due to the fundamental difference in architecture. SDN network is made up of switches that are merely forwarding devices whereas the traditional solutions require intelligent switches. OpenFlow in SDN requires the non-matching packets to be sent by the switch to the Controller. This opens up numerous possibilities for tampering.

ARP POISONING DEFENCE IN NETWATCH : AN OVERVIEW

The software defined paradigm offers capabilities such as collection of statistics from the network and programming of forwarding devices connected to the network. The proposed method in this paper relies on periodic collection of flow statistics from the connected switches, which are then analysed for possible poisoning by a rogue host. The application which is an extension of our restrictive firewall has been written using Python over POX controller [3].

Threat Model

The researchers assume that the Controller within the SDN is a trusted entity, whereas the hosts are untrusted. This implies that the communication from Controller to switches can be trusted but not the other way around. In this paper, only the internal attacks are being considered and hence the focus is only on the Open Flow messages within the network.

NetWatch Functionality

NetWatch collects topology information from the network through OpenFlow Control messages. It then reads the policy file and installs flow table entries in the switches based on the rules specified. From this point onwards the communication between network hosts is restricted to those mentioned in the policy file. Since this functionality has already been discussed in [3], [8] we are limiting the discussion to the new module added.

Apart from the control aspect, the application is also equipped to periodically analyse the ARP traffic through the network. If an unnaturally high amount of ARP packets is seen, the traffic is diverted to the Controller. The Controller checks if the packet is spoofed. If spoofing is detected, an entry is installed in the switches to drop the ARP packets for a pre-defined time period.

Figure 1 shows NetWatch's workflow which consists of two stages. First, it initiates the process of flow statistics collection and instructs the connected switches to transmit the ARP packets to the Controller so as to obtain the mapping between IP addresses and MAC addresses of the hosts. The firewall policies are also entered into the switch flow tables. The IP-MAC database stores the mapping information collected during this stage.

Secondly, having collected the network device information, the application cancels the earlier instruction involving the ARP packets and focuses on the flow statistics. From this point, the Controller is not involved in processing the packets. This helps in avoiding traffic bottleneck at the Controller level.

The redirect to controller instruction is re-issued only if an unusual amount of ARP packet movement is observed. In this scenario, the Assessor module collects the ARP Packet data for analysis while also temporarily suspending traffic between the devices. It uses the information collected in the first stage to identify a mismatch and issues an alert.

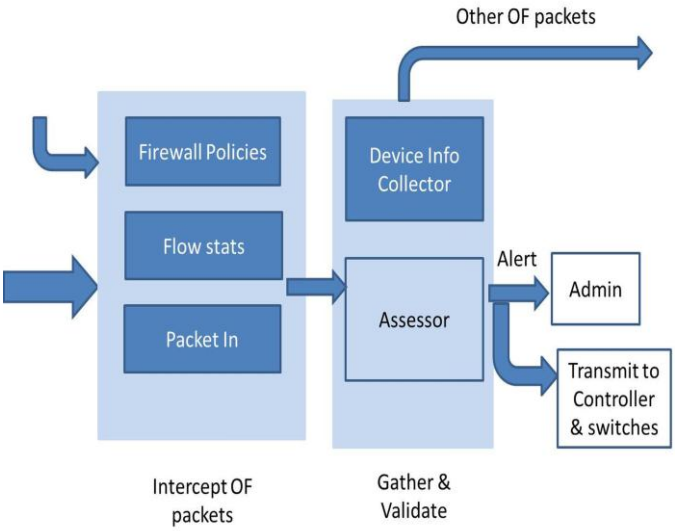

Figure 1: NetWatch workflow

IMPLEMENTATION

The ARP detection module has been implemented as a part of our firewall application and is integrated with the SDN Controller. NetWatch is written in Python and is compatible with POX. This section describes the implementation details of our application.

Experimental Setup

Ubuntu Virtual Machine with 2 cores and 2 GB of RAM have been used for the setup. The host machine has Windows 8 as the Operating system and has Intel Core i5 processor with 4 GB of RAM.

Mininet is used to create the virtual network with Linux OS and Open Virtual Switch software. The Mininet emulator is installed on Ubuntu 14.10 Linux virtual machine that runs on VirtualBox. The controller used is POX and NetWatch is run as a module on the SDN controller. A 100 Mbps link connects each host to the network whereas a 1 Gbps link exists between switches. Tests were performed to verify the capability of the application to detect and handle the poisoning attack.

As described in [3] NetWatch is a "reject all unless specified in the policy file" firewall. So, some rules were added to the policy file one of which was to allow communication between host 3 and host 4. Subsequently, using a socket program the ARP Cache was corrupted by flooding the victim with fake ARP request/reply packets. For example, host 1 (the attacker) corrupts the ARP cache of host 4 by replacing the host 3's MAC address with its own MAC address. Now, host 1 is pretending to be host 3.This affects the communication between hosts 3 and 4.

Defence against poison routing

As mentioned in Section A, host 1 poisons host 4 with an invalid entry. It floods the victim with hundreds of spoofed ARP request/reply packets. The *flowstats handler* in NetWatch keeps an eye on the ARP packets travelling across the network. It detects any unusually high amount of ARP traffic and installs a new rule in the switch flow table to redirect the ARP packets to the Controller. The *Packet-In* event of the Controller matches the packet IP & MAC parameters with its own IP-MAC database. At this stage, it can identify the attacker and the victim. The procedural algorithm is shown in Figure 2.

Once the attack is detected, NetWatch adds a new flow table entry to drop all packets from the attacker for a fixed duration and issues an alert. Unfortunately, since it is not possible to determine dynamically if the attacker has been neutralized, NetWatch works under the assumption that the malicious host will be blocked within certain duration once the alert is issued.

Algorithm :

```
Procedure FlowstatsHandler
 Stats ← Event statistics
 If large ARP packet count in Stats then
    Set Examine_Packet flag
    Set flow table rule "send packet to Controller"
End procedure
Procedure HandlePacketIn(pkt)
 If Examine_Packet is True then
  If isSpoofed(pkt) is True then
       Set flow table rule " drop ARP packets from source mac"
       Send alert
End procedure
```

Figure 2: The FlowstatsHandler & HandlePacketIn procedures in NetWatch.

Performance Assessment

The simulation results compare the performance of NetWatch with ARP attack defence to that of the standard POX controller without the defence mechanism. Experiments were performed using tree topology as well as linear topology. A custom Mininet Data Centre topology with four racks was also used for testing. Each rack has four hosts and a single top-of-rack switch. These switches are connected to a central root switch. In all the cases, the latency and TCP bandwidth prior to and after the attack was measured.

As depicted in Figure 3, it was observed that latency is on the lower side for NetWatch when compared to the POX stock component. This is because NetWatch inserts ICMP flow rules into the flow tables as soon as the connection is established.

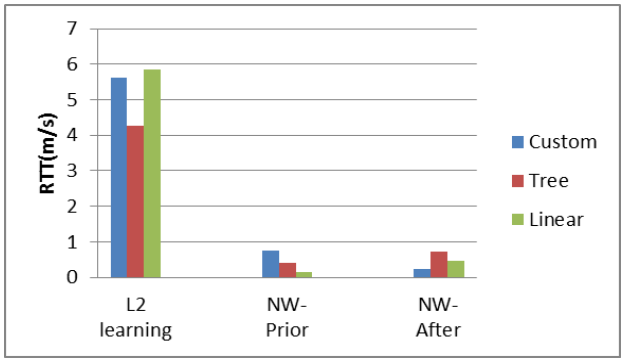

Figure 3: Comparison of latency experienced in different scenarios

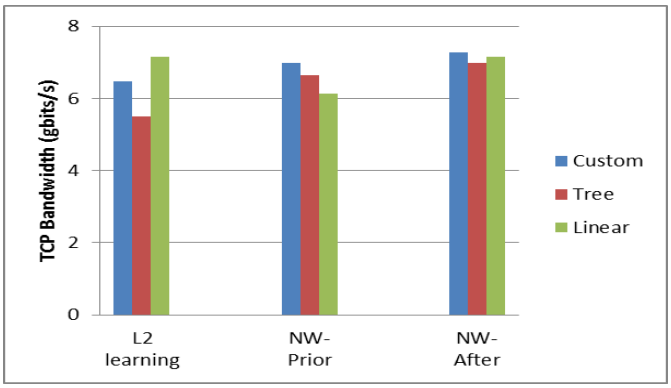

Figure 4: Comparison of bandwidth in different scenarios

As far as bandwidth (Figure 4) was concerned no significant variations albeit very minor during the test run was found. The one low point which the researchers would like to mention is the small down-time of the network at the moment when the ARP attack is discovered. This has been introduced to divert the continuously flooded spoofed packets from the victim and the Controller. The network bounces back to its working state after a predefined period putting back all the rules in place.

CONCLUSION

In this paper, a solution for detecting ARP poison routing for SDN networks using a firewall has been introduced. This feature can effectively deal with insider ARP poison routing in a small scale network. NetWatch utilizes the programmability of SDN to combine multiple functionalities into a single component instead of having numerous applications to counter various problems with in the network. As far as we are aware, a completely open source utility which combines firewall with ARP poison detection is first of its kind. With two utilities in one, our component is a cost-effective way to protect small scale networks from insider attacks.

REFERENCES

[1] I.Alsmadi and D.Xu, “Security of software defined networks: A survey,” computers & security, vol. 53, pp. 79-108, Sep. 2015.

[2] Cabaj, K., Wytrebowicz, J., Kuklinski, S., Radziszewski, P. and Dinh, K.T., “SDN Architecture Impact on Network Security,” in FedCSIS Position Papers, pp. 143-148, Sep. 2014.

[3] D. Balagopal and X.A.K Rani, “NetWatch: Empowering software-defined network switches for packet filtering,” in 2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), pp. 837-840. IEEE 2015.

[4] S.Whalen. (2001)“An introduction to arp spoofing." Node99 [Online].

[5] C.L.Abad and R.I. Bonilla, “An analysis on the schemes for detecting and preventing ARP cache poisoning attacks,” in 2007 International Conference on Distributed Computing Systems Workshops (ICDCSW'07), pp. 60-60. IEEE 2007.

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

[7] P.Kwan, Foundry Networks, Inc.,” System and method for ARP anti-spoofing security,” U.S. Patent 7,562,390, 2009.

[8] D. Balagopal and X.A.K Rani, “Leveraging the Power of Software Defined Paradigm to Control Communication in a Network, ”Indian Journal of Science and Technology, 9(14), 2016.