Chinese Journal of Electronics
Vol.28, No.1, Jan. 2019

An Active Defense Solution for ARP Spoofing in OpenFlow Network*

XIA Jing[1], CAI Zhiping[1,2], HU Gang[1] and XU Ming[1]

(1. *College of Computer, National University of Defense Technology, Changsha 410073, China*)

(2. *School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing 210044, China*)

Abstract — As an emerging network technology, Software-defined network (SDN), has been rapidly developing for recent years due to its advantage in network management and updating. There are still a lot of open problems while applying this novel technology in reality, especially for meeting security demands. The Address resolution protocol (ARP) spoofing, a representative network attack in traditional networks is investigated. We implement the ARP spoofing in SDN network firstly and find that the threat of ARP attack still exists and has big impact on the network. We propose a novel mechanism as defense solution for ARP spoofing oriented to OpenFlow platform. Theoretical analyzation is given, and the mechanism is implemented as a module of POX controller. Experiment results and performance evaluations show that our solution can reduce the security threat of ARP spoofing remarkably on OpenFlow platform and related SDN platforms.

Key words — Software-defined network (SDN), Address resolution protocol (ARP) spoofing, OpenFlow.

I. Introduction

As the key outcome of extensive research efforts over the last decade, Software-defined network (SDN) has been proposed as an emerging network architecture that has several advantages: programmability, agility, open standards-based and vendor-neutral attributes[1]. In the past few years, SDN grows explosively as it is ideal to meet the high-bandwidth requirement of current applications. Many typical SDN platforms such as OpenFlow have been developed[2]. SDN has received tremendous interests from both industry and academia, and numbers of researches boom on SDN involves various aspects such as structural design, performance improvements, and so on[3]. However, the researches on security issues are quite rare while experts claimed that various security threats become a huge barrier to the application for SDN[4].

As an industry standard of SDN, OpenFlow is still based on Ethernet technology to keep maximum compatibility with traditional network equipment[5]. Thus, security threats in the Ethernet may still be effective in the OpenFlow network, and may even lead to more serious consequences due to the centralized control feature of SDN.

Address resolution protocol (ARP) spoofing is a representative security threaten in the era of Ethernet. In the procedure of ARP spoofing, an attacker exploits the flaws of ARP protocol to forge an address of the victim host, which can enable it to get the messages that belong to the victim. Although simple, this kind of attack has been in existence since the advent of Ethernet, and it is still difficult to find an efficient defense mechanism until now.

In this paper, we attend to study the problem of ARP spoofing in OpenFlow network. The threat of ARP spoofing is shown to still exist in SDN through our prototype system implementation. Furthermore, it shows up new features in OpenFlow network due to the novel network architecture, which leads to more serious consequences. We theoretically analyze the possible influence of ARP spoofing in OpenFlow based SDN. Then we build a testbed aimed at various OpenFlow controllers. Several experimental results show that simple ARP spoofing is still effective and has the additional features which do not exist in the traditional Ethernet.

There would be a possibility to construct efficient defense mechanism for addressing the problem of ARP spoofing in OpenFlow with the advantages of SDN (*e.g.*, centralized control). Hence, we propose an Active ARP inspection (AAI) mechanism as ARP spoofing defense solution in OpenFlow network. This defense mechanism

*Manuscript Received Mar. 27, 2017; Accepted June 5, 2017. This work is supported by the National Natural Science Foundation of China(No.61379145, No.61379144, No.61501482).

can be implemented as a module of POX controller. We conduct several tests on AAI. Results show that our solution can reduce the security threat of ARP spoofing remarkably with negligible performance impact.

The rest of the paper is organized as follows: the related works are described in Section II. After that the ARP Spoofing and its influence on OpenFlow Network is discussed in Section III. The detail of defense solution against ARP spoofing in OpenFlow network is presented in Section IV. And the effectiveness of AAI is verified by experimental evaluations in Section V. Finally, conclusions are given in Section VI.

II. Related Works

In traditional Ethernet networks, defense mechanisms against ARP spoofing have been in existence for many years, and the defensive methods can be divided into two categories according to the way in which they need to upgrade or modify the network switching device:

1) Modifying network switching device

In order to avoid ARP spoofing, network administrator can manually manage the allowed MAC address of each network device by upgrading the network switch with Cisco's Dynamic ARP inspection (DAI) technology[6]. But it is very inflexible due to the manually configure cost, and it is not practical for dynamic scenarios such as wireless networks with users randomly accessing. Antidote[7] avoids the MAC collision by giving the MAC address owner the higher privilege in switching device. However, if attackers attack the newly added host and the forged ARP reply message arrives earlier than the real message, the Antidote defense mechanism will fail completely. Philip[8] proposes a wireless LAN ARP spoofing defense method. After upgrading the wireless AP software, a network host's IP-MAC mapping table can be constructed in AP through tracking Dynamic host configuration protocol (DHCP) Acknowledgement (ACK) message in the network. However, this method is only applicable to dynamic network address configuration using DHCP. These mechanisms fight ARP spoofing by modifying network device which can maintain host compatibility, but they are not flexible enough and difficulty to use.

2) Improving the ARP protocol

S-ARP[9] and Ticket-based ARP (TARP)[10] are solutions that introduce encryption to arp protocol. They perform global authentication by introducing additional authentication servers, such as the Authoritative key distributor (AKD) used by S-ARP and Local ticket agent (LTA) used by TARP. But the authentication service of these methods introduces the problem of single-point failure. Furthermore, distribution mechanism is complex and these methods are also not practical for wireless networks. MR-ARP[11] and its improved version[12] modify the ARP protocol and propose an ARP spoofing defense mechanism which is based on fair voting mechanism. Similar to Antidote, the original MAC address holder has higher priority. In this scenario, when the network host detects an ARP reply message that conflicts with the MAC address in ARP cache, it sends a poll to neighboring host to decide whether to accept the new ARP reply message according to the voting result. These mechanisms can elimination ARP spoofing, but introduces new ARP protocol which need substantial modification of existing network host.

On the other hand, since research on OpenFlow network security is still in its infancy, few works have been done on ARP spoofing in SDN[13]. Crenshaw[14] implements a very simple ARP spoofing defense mechanism in POX controller, which requires the controller to check whether each ARP reply message it receives will cause duplicated IP entries in the ARP cache. When the duplication occurs, it is believed that an ARP spoofing attack occurs. This program is similar to Antidote, but has high false alarm rate. Cox[15] proposes Network flow guard for ARP (NFGA), a SDN security module which works by hashing a host's physical address with an appropriate IP-Port association to deny ARP spoofing at real-time. However, Mac-IP-Port mapping table based method is inflexible due to lacking of quick update mechanism. Nehra[16] proposes traffic pattern based solution to ARP related Threats (FICUR), which gathers and analyzes network parameters from ARP messages, and detects attacks by compare these parameters with logged historic parameters. Like Antidote, this method may cause high false alarm rate.

Based on the description about existing works above, we can get that the traditional defense mechanisms may require a lot of manpower and material resources making them inefficient, or may need to make large changes to existing device or protocol which introduce compatibility problems. In the OpenFlow network, there is existing initial research against ARP spoofing, but they did not carry out an in-depth study, and few mature and effective defense mechanisms have been proposed.

III. ARP Spoofing in OpenFlow Network

1. ARP protocol and ARP spoofing

In IP network, the ARP is one of the most important network layer protocols. It is used to solve the address resolution problem between IP protocol and Ethernet protocol. When a host tries to send IP messages to the other hosts with target IP address, the local ARP cache should be queried first. If the cache is hit, the corresponding target MAC address is obtained. Otherwise, an ARP request/response procedure is launched in accordance with ARP protocol.

Unfortunately, there is no security checking mechanism in the request/response procedure of the ARP protocol. When the initiate host receives an ARP reply message, it neither authenticates the host that sends the message nor checks whether the received ARP reply message is triggered by its own ARP request. All received ARP reply messages will be processed by the initiate host, and strictly follow the process mentioned above. Thus, the ARP spoofing can be carried out easily by exploiting this unauthenticated process.

In the procedure of ARP spoofing attack, an attacker which disguises himself as a target host with target IP address constructs and sends forged ARP reply messages to map his own MAC address to the target IP address. When the forged ARP reply messages are received by victims, they store the MAC address and IP address mapping in that message to their own ARP cache without authentication, resulting the destination MAC address is marked as the attackers' MAC address. The worst problem is that all messages are then switched to the attacker. The procedure is shown in Fig.1.

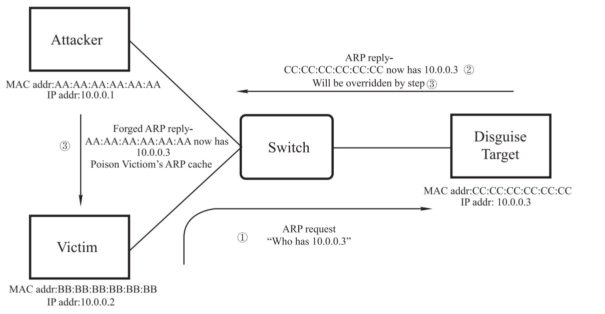

Attacker
MAC addr:AA:AA:AA:AA:AA:AA
IP addr:10.0.0.1
ARP reply-
CC:CC:CC:CC:CC:CC now has 10.0.0.3 ②
Will be overridden by step ③
③
Forged ARP reply-
AA:AA:AA:AA:AA:AA now has
10.0.0.3
Poison Victiom's ARP cache
Switch
Disguise
Target
MAC addr:CC:CC:CC:CC:CC:CC
IP addr: 10.0.0.3
Victim
①
ARP request
"Who has 10.0.0.3"
MAC addr:BB:BB:BB:BB:BB:BB
IP addr:10.0.0.2

Fig. 1. ARP spoofing procedure

On the base of ARP spoofing, an attacker can perform further attacks such as man-in-the-middle attack on a victim or a denial-of-service attack on a disguised target[17]. In actual environment, ARP spoofing may have variety of deformation, but its principle is always based on the forged ARP reply message to poison victim's ARP cache. Thus, ARP spoofing can be implemented as long as there are devices that are based on ARP for address resolution.

2. ARP spoofing in OpenFlow network

As a typical SDN implementation, OpenFlow follows the design principle of separation of control plane and data plane. A controller is used to centrally manage OpenFlow switches in the network. To keep compatibility with legacy network devices, OpenFlow retains a wealth of Ethernet technology. For example, in OpenFlow Switch Standard 1.4, the matching field of the flow table contains four entries that are closely related to the ARP protocol, Ether src, Ether dst, IP src and IP dst[5]. In fact, hosts are not aware of whether they are in OpenFlow network environment or not, and ARP is still the only protocol to resolve MAC address from IP address. Therefore, theoretically, there is possibility of performing ARP spoofing in OpenFlow network.

To evaluate the effect of ARP spoofing in the OpenFlow network, we build a testbed based on the OpenFlow platform following the typical ARP spoofing process shown in Fig.1.

The testbed is based on Mininet 2.2.0 which works in an Ubuntu Linux 14.04 workstation, with an OpenFlow switch connected to three hosts and one controller, where three hosts play the attacker, the victim and the camouflage target respectively.

In view of popularity, we choose three of the most popular OpenFlow controllers, *i.e.*, OVS, POX and OpenDaylight. The ARP spoofing attack is initiated by the arpspoof tool**, one forged ARP reply message is sent to the victim in every second. We observe the status of the network in the whole process of attack. Experimental results listed in Table 1 show that although all three kinds of controllers' and the victim hosts' ARP caches are contaminated, the ARP spoofing influence on various controllers are different.

Table 1. Influence of ARP spoofing

Controller	Character	Influence on host	Influence on controller
OVS controller	Simulate traditional Layer 2 switches	ARP cache be poisoned immediately	Nothing
POX	Research oriented	ARP cache poisoned immediately	Throw software exception
Open daylight	Production oriented	ARP cache be poisoned after a period of continuous attack (victim host ARP cache timeout)	Issues error flow table rules

For OVS controller, the influence of ARP spoofing on it is exactly the same as a traditional switch due to that it completely simulates traditional Layer-2 learning switch. A relatively complex Layer-3 learning switch module has been loaded in POX, but ARP spoofing is not considered in the software implementation of this module, which means it still could be affected by ARP spoofing. It should be noted that even normal network communication will cause the POX controller throwing software exception after the ARP spoofing attack, which means the controller will enter abnormal situation, even though there is no software crash.

As a more sophisticated SDN controller, intra-subnet network communications have been fully optimized in OpenDaylight. The OpenFlow switches managed by the controller do not perform Layer 2 switching at all when

** *http://www.monkey.org/dugsong/dsniff* handling these communications and uses IP address as the forwarding destination. Furthermore, OpenDaylight controller provides defense mechanism to ARP spoofing in some degree since it does not perform any forwarding operation if an ARP reply message does not have corresponding request. In fact, after ARP spoofing attack, the OpenDaylight controller manage to eliminate all ARP reply messages without corresponding requests in the beginning. However, the judge logic which decide whether the ARP reply message is forged or not in OpenDaylight is primitive. When the victim host's ARP cache is timeout and send a valid ARP request message, the OpenDaylight controller matches this request message to one of the forged ARP reply message, and forwards it to the victim host. Eventually, the ARP cache of the victim host is contaminated. More seriously, the ARP cache inside controller is also contaminated by forged ARP reply messages, which causes controller issue error flow table rules, further leads to network disruption.

3. Features of ARP spoofing in OpenFlow network

It has been verified that ARP spoofing can be effectively conducted in the OpenFlow network through experiments above. We can conclude that this ARP spoofing has the following new features in OpenFlow network:

1) ARP spoofing for the current variety of OpenFlow controller is ubiquitous. Among the three controllers we tested, the simplest ARP spoofing attack initiated by the traditional arpspoof tool can contaminate the victims' ARP cache and may lead to network disruption.

2) In OpenFlow network, threat of ARP spoofing is of no doubt diversified. In traditional network, the ARP spoofing mainly target gateway, and then through the man-in-the-middle attack to realize sniffing or tampering with the traffic data. In OpenFlow network, due to the in-band control plane, a malicious user could select an OpenFlow switch/controller as the attack target. Combining with side channel attacks, he could sniff the important data of control plane, such as encryption information, control plane information, and even modify the network configuration.

3) The influence of ARP spoofing varies for different types of controllers. Since the network behavior of OpenFlow networks depend entirely on the controller, any change in controller's implementation may affect ARP spoofing, as shown in our tests mentioned above.

4) The ARP spoofing could have more serious implications in OpenFlow networks as the network logic is more dependent on the controller. For example, in the POX scenario, the controller will throw an exception; In the OpenDaylight scenario, messages send to the camouflage target host might not be delivered to the right target, due to the error flow table. This leads to DoS attack against the camouflage target, even if it is not the intent of attacker.

5) Controller software quality in OpenFlow network is actually unpredictable. Unlike traditional commercial network devices which have been designed and verified for a long time, the user-written controller software is more vulnerable to various security threats. Such a network is clearly more fragile than traditional one. It is obvious that problems of ARP spoofing clearly stem from poor quality control in the software design.

IV. Active ARP Inspection

Based on the analysis above, we then propose an ARP spoofing defense mechanism based on centralized management feature of OpenFlow network, called Active ARP inspection (AAI).

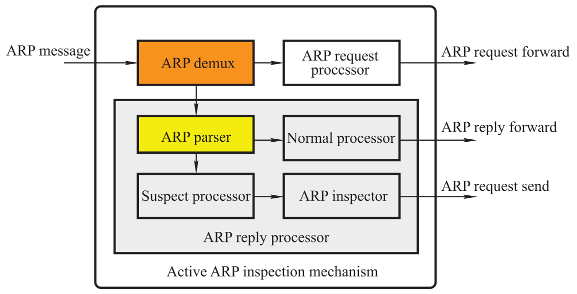


Fig. 2. Infrastructure of active ARP inspection

AAI can be an application deployed in OpenFlow controller. The infrastructure is shown in Fig.2. By installing appropriate rules in OpenFlow switchs, all ARP-related messages on the local subnet are forwarded to the controller and subsequently would be processed by the AAI. ARP request and reply messages are separated by the ARP Demux. The former is sent to corresponding OpenFlow switch by the ARP Request Processor, and the latter is handled by the ARP Reply Processor.

The ARP reply processor consists of two data structures and four logical components. The data structures include ARP reply message waiting queue (Waiting Queue) and ARP reply resolution cache (ARP Cache), while the logical components include ARP reply message parser (ARP Parser), normal ARP reply message processor (Normal Processor), suspicious ARP reply message processor (Suspect Processor) and forged ARP reply message inspector (ARP Inspector).

The Waiting Queue is a simple FIFO table, and all unprocessed ARP reply messages are stored in it. The ARP Cache consists of four columns: "IP address", "MAC address", "Switch ID", and "Switch Port" which represent the host IP address, host MAC address, identity, and port number of the switch connected to the host respectively. Each host is associated with one entry that will be updated when ARP reply message arrived in AAI. A possible ARP Cache fragment is shown in Table 2.

Table 2. ARP reply resolution cache

No.	IP Addr.	MAC Addr.	Switch ID	Switch Port
1	10.0.0.2	00-23-5A-00-00-01	Switch 1	Port 2
2	10.0.0.8	00-23-5A-00-00-15	Switch 2	Port 4
...	...	...	...	...
New	10.0.0.1			

The working process of ARP Reply Processor can be summarized as a finite state machine. Fig.3 shows the general structure for this finite state machine, which possess 5 states: Normal Mode, Inspection Mode, Waiting Mode, Timeout Mode and Judgement Mode.

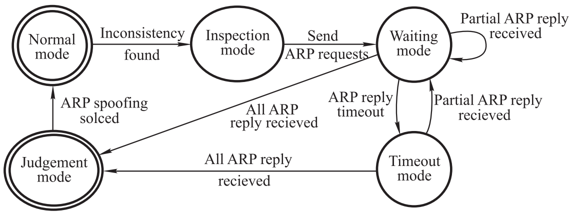


Fig. 3. The finite state machine of ARP reply processor

In Normal Mode, the ARP Parser extracts the ARP reply message from the head of the Waiting Queue for resolve. When the resolution is consistent with the ARP Cache, the ARP reply message will be sent to the Normal Processor. Otherwise, the ARP reply message will be labeled as Suspicious ARP reply message, and then the ARP Reply Processor enters Inspection Mode.

In Inspection Mode, the ARP Parser suspend normal ARP reply message resolving, and the Suspicious ARP reply message causing state change is passed to the Suspect Processor. Suspect Processor will find out all the entries in ARP Cache which is inconsistency with this message. After that it forwards these entries to ARP Inspector. In ARP Inspector, ARP request messages are constructed and sent to the hosts corresponding to the suspicious ARP reply message inconsistency (conflicting hosts). Then, the ARP Reply Processor enters Waiting Mode, waiting for the ARP reply message responding from the conflicting hosts.

In Waiting Mode, the ARP Parser preprocesses the ARP reply message at first. General ARP reply messages not coming from conflicting hosts will be pushed into Waiting Queue. Otherwise they will be collected to judge whether the Suspicious ARP reply message is forged.

If no ARP reply message is received before timeout, the host will be considered as non-existent, and ARP Reply Processor enters Timeout Mode until another ARP reply message received. After all the ARP reply messages from conflicting hosts are received or timeout, the ARP Reply Processor enters Judgement Mode.

In Judgement Mode, ARP Reply Processor distinguishes Suspicious ARP Reply Message based on the results listed in Table 3. If the results are Normal, the message will be forwarded and the ARP Cache is updated. Otherwise, an alarm is sent to the controller and a packet-dropping rule that matches the Suspicious ARP Reply Message is installed to the switch connecting to the attack host. Then the ARP Reply Processor resumes Normal Mode. The ARP reply message process is shown in Algorithm 1.

Table 3. Forged ARP reply message inspect result

Suspect	Conflicting host			Result
	IP	MAC	Switch	
Change	Any	Any	Any	Forge
No change	No conflict	No conflict	No conflict	Normal
No change	Conflict	Any	Any	Forge
No change	Any	Conflict	Any	Forge
No change	Any	Any	Conflict	Forge

Algorithm 1 ARP reply message process

```
1:  parse ARP reply message
2:  if (ARP reply message leads to inconsistency) then
3:    EntrySet ← conflicting entrys in ARP Cache
4:    for host in EntrySet
5:      send ARP request to host
6:    end for
7:    while not all ARP reply collected or timeout
8:      recieve ARP reply
9:    end while
10:   judge ARP reply message based on Table 3
11: else
12:   forward ARP reply message
13: end if
```

V. Evaluation

In order to evaluate the mechanism presented in Section IV, we implemented it as a POX module and evaluated it under the same conditions as the Mininet test environment described in Section III.

1. Functionality evaluation

The main purpose of the functionality test is to verify the validity and accuracy of the AAI. In addition to the AAI module, the POX controller only loads the Layer 2 learning switch module (forwarding.l2_learning) during the testing process, The arpspoof tool is used to initiate ARP spoofing on a single host at 6 seconds after the start of the test. Throughout the testing process, the victim host continuously issues ping commands to camouflage target in order to detect the communication packet loss rate between victim host and camouflage target.

For the POX controller with AAI module loaded and the POX controller without AAI model, the test results shown in Fig.4. When the AAI module is not loaded, the ARP cache of the victim host is immediately poisoned after ARP spoofing is started, and the Internet control message protocol (ICMP) request message from the ping command fails to reach the target host, the ICMP packet loss rate jumps from 0% to 100%.

When the AAI module is loaded, all forged ARP reply messages are intercepted, ARP cache of the victim host is not poisoned, and neither the switch nor the controller is affected by ARP spoofing. The ICMP packet loss rate between the victim host and the camouflage target is always 0%.

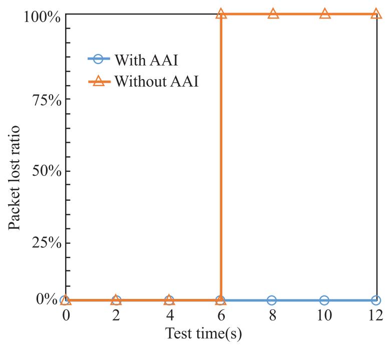


Fig. 4. ARP spoofing packet loss rate

2. Performance evaluation

The experiments evaluate two aspects of AAI: the performance influence on the controller, and the ARP message processing performance of the AAI module.

We use Cbench[***] to test the performance of the controller after loading the AAI module. Cbench is an open source OpenFlow controller performance testing tool. In the process of testing, it mimics OpenFlow switch and send Packet_In message to the controller. The Packet_In message contains virtual MAC and IP address pairs to simulate the virtual host. Random data is also sent to simulate the communications liabilities. controller performance can be measured by checking the controller response speed to these messages. Since the response of the controller to Packet_In is a FlowMod message, the unit of the Cbench test result is the number of flow per second (flow/sec).

The objects of the Cbench test are the POX controllers' performance with and without loading AAI module. In addition to the AAI module, the POX controller only loads the layer 2 learning switch module (forwarding.l2_learning), link and topology discovery module (openflow.discovery, openflow.topology). Cbench is configured as 16 virtual switches, and each of them is configured with 10000 different virtual MAC and IP address pairs. 100 rounds of throughput tests are carried out and the average results are shown in Table 4.

Table 4. Cbench throughput test results (flow/sec)

	Min.	Max.	Avg.	S.D.
With AAI	5679.76	8307.89	7919.66	404.60
Without AAI	6315.44	9090.56	8399.18	712.11

The test results show that even in the controller with only the most basic module attached, the AAI module only cut its throughput by 5.7%. In actual system, the impact of AAI module on controller performance will be smaller since more practical modules will be loaded on controller.

We wrote a special tool to repeatedly send ARP request messages from hosts in OpenFlow network. By checking the time difference between the ARP request and ARP reply message on the port of the OpenFlow switch, the performance of the AAI module can be calculated.

In order to eliminate other modules' impact on the performance of AAI module, in this test, the POX controller only loads a modified hub module besides the AAI module. This modified hub module broadcasts all the messages except the ARP message and forwards ARP messages to the controller.

To investigate the effect of network size on the performance of AAI module, we built complex network test scenarios with multiple switches. Using the built-in topology generation script, Mininet can simulate any network topology. The script can generate the network topology randomly according to the number of switches, the number of switch ports and the number of hosts. Network connectivity between all the switches and hosts can also be ensured too. We built three types of network topology scenarios including 10, 100, 200 switches respectively, and 8 ports per switch. In each scenario, we select six different settings which have from 5 to 1000 network hosts respectively.

The performance of AAI module is given in Fig.5, where the horizontal axis shows the host number in the network. The number of ARP messages is set to 1000. We calculate the average response time by recording the ARP requests sending time and the ARP responses receiving time.

From the test results are shown in Fig.5, the average response time increase slowly as the number of hosts. We can get the performance of the AAI module is reduced slightly when the complexity of the network topology is increased. But when the number of hosts increased, the performance of AAI module is almost unchanged.

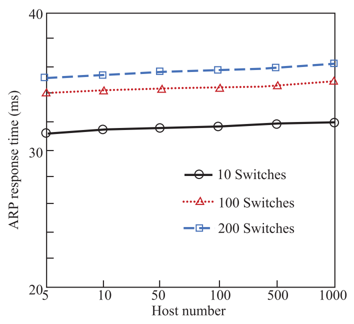


Fig. 5. Performance of AAI module

*** *https://github.com/mininet/oflops/tree/master/cbench*

3. Discussion

As in previous approaches, such as NFGA and FICUR, are neither flexible nor accurate. NFGA is still using "MAC: IP: Port" triplet binding relationship to identify forged ARP messages, although an auxiliary technology that use DHCP messages to build dynamic table is introduced, the system is still not flexible enough. For the FICUR, the judge of the suspicious ARP message is only according to the historical log, so, in dynamic network environment, lot of false positives may be accrued. Compared with those defense mechanisms of ARP Spoofing in OpenFlow network, our approach has multiple advantages, while ensuring a second inspection of suspicious ARP messages, also ensures the flexibility of the method and has a better balance of accuracy and flexibility in the past.

VI. Conclusions

In this paper, we analyze the characteristics of ARP spoofing in OpenFlow network. On this basis, we propose Active ARP Inspection, a mechanism for ARP spoofing defense in OpenFlow network by exploiting the SDN advantages. With this mechanism, OpenFlow network can defense ARP spoofing with nearly no impact on network performance. In future work, we will further improve this mechanism to defense more complex attacks in more volatile network.

References

[1] Y.A. Shu, "Heterogeneous networking architecture based on SDN", *Chinese Journal of Electronics*, Vol.26, No.1, pp.166–171, 2017.

[2] N. McKeown, T. Anderson, H. Balakrishnan, *etal.*, "OpenFlow: Enabling innovation in campus networks", *ACM SIGCOMM Computer Communication Review*, Vol.38, No.2, pp.69–74, 2008.

[3] Z.-P. Cai, Z.-J. Wang, K. Zheng, *etal.*, "A distributed TCAM coprocessor architecture for integrated longest prefix matching, policy filtering, and content filtering", *I*EEE Transaction on Computers, Vol.62, No.3, pp.417–427, 2013.

[4] W. Wang, W. He and J. Su, "Network intrusion detection and prevention middlebox management in SDN", *2015 IEEE 34th International Performance Computing and Communications Conference* (*IPCCC*), Nanjing, China, pp.1–8, 2015.

[5] Open Networking Foundation, "OpenFlow Switch Specification-Version 1.4.0", available at *https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/ openflow/openflow-spec-v1.4.0.pdf*, 2013-10-14.

[6] Y. Bhaiji, "Network security technologies and solutions", *CCIE professional development series*, 2008.

[7] I. Teterin. "Antidote". available at *http://online.securityfocus.com/archive/1/299929*, 2002-11-14.

[8] R. Philip, "Securing wireless networks from arp cache poisoning", available at *http://www.cs.sjsu.edu/faculty/ stamp/students/Roney298report.pdf*, 2007-05.

[9] D. Bruschi, A. Ornaghi and E. Rosti, "S-arp: A secure address resolution protocol", *Proc. of the IEEE 19th Annual Computer Security Applications Conference*, Las Vegas, NV, USA, pp.66–74, 2003.

[10] W. Lootah, W. Enck and P. McDaniel, "Tarp: Ticket-based address resolution protocol", *Computer Networks*, Vol.51, No.15, pp.4322–4337, 2007.

[11] S. Y. Nam, D. Kim, J. Kim, *et al.*, "Enhanced arp: Preventing arp poisoning-based man-in-the-middle attacks", *IEEE Communications Letters*, Vol.14, No.2, pp.187–189, 2010.

[12] S. Y. Nam, S. Djuraev and M. Park, "Collaborative approach to mitigating ARP poisoning-based Man-in-the-Middle attacks", *Computer Networks*, Vol.57, No.18, pp.3866–3884, 2013.

[13] D. Kreutz, F. Ramos, P. Esteves Verissimo, *et al.*, "Software-defined networking: A comprehensive survey", *Proceedings of the IEEE*, Vol.103, No.1, pp.14–16, 2015.

[14] A. Crenshaw. "Security and software defined networking: Practical possibilities and potential pitfalls", available at *http://www.irongeek.com/i.php?page=security/security-and-software-defined-networking-sdn-openflow*, 2013.

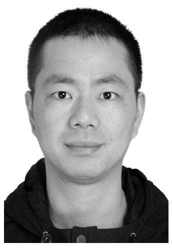
[15] J.H. Cox, R.J. Clark and H.L. Owen, "Leveraging SDN for ARP security", *SoutheastCon 2016*, Norfolk, VA, USA, pp.1–8, 2016.

[16] A. Nehra, M. Tripathi and M.S. Gaur, "FICUR: Employing SDN programmability to secure ARP", *2017 IEEE 7th Annual Computing and Communication Workshop and Conference* (*CCWC*), Las Vega, NV, USA, pp.1–8, 2017.

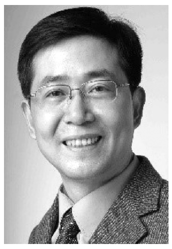
[17] S. Whalen. "An introduction to arp spoofing", available at *https://www.security-audit.com/files/intro to arp spoofing.pdf*, 2001.

XIA Jing received the B.S. and M.S. degrees in computer science from National University of Defense Technology (NUDT), China, in 2003 and 2009, respectively. Now, he is an assistant professor in College of Computer, NUDT. His current research interests include network security and software-defined networking.
(Email: jingxia@nudt.edu.cn)

CAI Zhiping (corresponding author) received the B.S., M.S., and Ph.D. degrees in computer science from NUDT, China, in 1996, 2002, and 2005, respectively. Now, he is a professor of College of Computer, NUDT. His current research interests include network security and big data. He is a senior member of CCF. (Email: zpcai@nudt.edu.cn)



HU Gang received the Ph.D. degree of computer science and technology from NUDT, China in 2010. Now, he is an assistant professor in College of Computer, NUDT. His current research interests include the spectrum management of cognitive radio networks, wireless security, and wireless software defined networks.
(Email: hugang@nudt.edu.cn)



XU Ming is currently a professor and director of Network Engineering Department in College of Computer, NUDT, China. Prof. Xu is a senior member of CCF and member of IEEE and ACM. His major research interests include mobile computing, wireless network, cloud computing and network security.
(Email: xuming@nudt.edu.cn)