doi: 10.17586/2226-1494-2021-21-3-401-409

An efficient mechanism to detect and mitigate an ARP spoofing attack in software-defined networks

Ghadeer Darwesh[1], Alisa A. Vorobeva[2]✉, Viktoriia M. Korzhuk[3]

[1,2,3] ITMO University, Saint Petersburg, 197101, Russian Federation

[1] ghadeerdarwesh32@gmail.com, https://orcid.org/0000-0003-1116-9410
[2] Alice_w@mail.ru✉, http://orcid.org/0000-0001-6691-6167
[3] vmkorzhuk@itmo.ru, http://orcid.org/0000-0002-0240-9067

Abstract
The work focuses on software-defined network security, as it was always one of these foremost critical concerns due to the centralized nature in SDN architecture where many serious attacks in traditional networks still appear in SDN networks such as ARP spoofing attack despite many existing security algorithms, methods and systems. In this work, we proposed a new approach to secure SDN from an ARP poisoning attack. The new solution extends the controller with a new module that uses a new algorithm to detect and mitigate the ARP spoofing attacks according to three states of each host in the network. The new mechanism involves the DHCP and manual assignment of IP addresses using three classes to classify the hosts according to their situations in the network. The CHT helps to set the host in an intermediate state between verifying and banning and detect the attack according to the next step of the host. The proposed mechanism was tested successfully in a simulated environment using Mininet and POX controller. The solution was effectively able to accomplish the objective for which it was built, with a limited overhead on the network. This proposed solution neither has an extra overload in the network, nor requires any changes in the infrastructure or additional hardware to install. According to the experiment results of this solution, the average time to detect the ARP spoofing attack is about 11 ms, with minor overhead on the controller CPU.

Keywords
ARP, Software-Defined Networking (SDN), ARP cache poisoning attack, ARP spoofing, SDN security, OpenFlow security

For citation: Darwesh G., Vorobeva A.A., Korzhuk V.M. An efficient mechanism to detect and mitigate an ARP spoofing attack in software-defined networks. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2021, vol. 21, no. 3, pp. 401–409. doi: 10.17586/2226-1494-2021-21-3-401-409

УДК 004.89

Эффективный механизм выявления и противодействия ARP-спуфинг атакам в программно-определяемых сетях

Гадир Дарвиш[1], Алиса Андреевна Воробьева[2]✉, Виктория Михайловна Коржук[3]

[1,2,3] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация

[1] ghadeerdarwesh32@gmail.com, https://orcid.org/0000-0003-1116-9410
[2] Alice_w@mail.ru✉, http://orcid.org/0000-0001-6691-6167
[3] vmkorzhuk@itmo.ru, http://orcid.org/0000-0002-0240-9067

Аннотация
Предмет исследования. В работе рассмотрены вопросы обеспечения безопасности программно-определяемых сетей. Архитектура таких сетей имеет централизованный характер и обеспечение защиты циркулирующей в них информации затруднено. Многие классические атаки, например, атака с подменой ARP, остаются актуальными для программно-определяемых сетей несмотря на существование различных алгоритмов, методов и систем защиты. **Метод.** Предложен новый подход к защите SDN от атаки подмены (отравления) ARP. Решение заключается в расширении функционала контроллера за счет дополнительного модуля, который

на основании нового алгоритма и анализа состояний хостов позволяет обнаружить атаки с подменой ARP и ослабить их влияние на сеть. Особенность предлагаемого механизма – совместное использование протокола DHCP, ручного назначения IP-адресов и классификации хостов сети на три класса в соответствии с их состоянием в сети. CHT помогает перевести хост в промежуточное между проверкой и блокировкой состояние и обнаружить атаку в соответствии со следующим шагом хоста. **Основные результаты.** Предложенный механизм успешно протестирован в смоделированной среде с использованием Mininet и контроллера POX. На основании выполненных экспериментов сделан вывод об эффективности предложенного решения для поставленной цели и соответствии условию ограниченности ресурсов сети. **Практическая значимость.** Достоинствами предложенного решения являются отсутствие дополнительной нагрузки сети и необходимость внесения изменений в инфраструктуру сети или установки дополнительного оборудования. Согласно результатам экспериментов показано, что среднее время обнаружения атаки ARP-спуфинга на основании предложенного механизма составляет около 11 мс и не повышает значительно нагрузку на центральный процессор контроллера.

Ключевые слова
ARP, программно-определяемые сети (SDN), отравляющая атака ARP-кэша, ARP-спуфинг, безопасность SDN, безопасность OpenFlow

Ссылка для цитирования: Дарвиш Г., Воробьева А.А., Коржук В.М. Эффективный механизм выявления и противодействия ARP-спуфинг атакам в программно-определяемых сетях // Научно-технический вестник информационных технологий, механики и оптики. 2021. Т. 21, № 3. С. 401–409 (на англ. яз.). doi: 10.17586/2226-1494-2021-21-3-401-409

Introduction

Software-defined networking (SDN) networks have made a quantum leap compared to traditional networks that depend on physical infrastructure such as routing devices to make decisions and lead packets through the network [1]. SDN enables centralized administration of the whole network by decoupling the control plane which controls the whole network from the data plane which forwards the traffic in the network. Separating the brain of the networks from the forwarding devices makes the network more dynamic. It helps the administrators to control the entire network via a single controller instead of configuring device by device and worrying about basic device types. These devices only forward packets and do not participate in making decisions. SDN uses the "OpenFlow" protocol to forward the controller's decisions and rules to the switches. While the term SDN becomes more popular giving promises for future solutions in the networking field, SDN security is still a real challenge due to the SDN architecture and centralized design of the network provided by the controller. From the attackers' point of view, the controller is a single point of failure and a rich place to steal information, other forwarding devices are also frail defenders against their attacks [2, 3]. These considerations open the entryway for new threats and more serious attacks comparing to traditional networks such as DOS, repudiation, ARP spoofing and other critical attacks. In the first section of this work, we will present a short clarification of the SDN architecture, then we will shed light on the ARP protocol and how ARP spoofing attack is performed in SDN. The next section will present the most effective achieved solutions and related works in this area. The last two sections will present our proposed mechanism to detect and prevent ARP poisoning attacks in SDN, test experiment and performance metrics evaluation.

Background

SDN architecture. Document [4] clarified widely the architecture of SDN, where there are three primary components (Fig. 1).

The controller: forms the brain of the network which manages the entire network and controls all forwarding devices by making decisions and installs flows to these devices. The controller uses the southbound interface to connect with the OpenFlow devices and the northbound interface to communicate with applications.

Applications: are programs that determine the behavior of the network, they are load balancers, intrusion prevention systems (IPS), controls, intrusion detection systems (IDS), access controls or other important applications.

SDN data devices: they are underlying devices (routers, switches...) that receive instructions from the controller and forward the packets according to the flows that were installed.

OpenFlow. The OpenFlow protocol is widely explained in the ARP cache poisoning attack document [5]. OpenFlow is the protocol used in the SDN to install flows and build the forwarding tables in the forwarding devices. It helps the controller to build a comprehensive view of the network topology and forms a communication dialect between the controller and the data plane devices via its actions, statics and rules for matching packets. When a controller receives a packet coming from the switcher it decides how to deal with this packet and builds an OpenFlow message that includes instructions using FLOW_MOD commands. This decision will teach the switch how to interact with this packet and similar packets in the future.

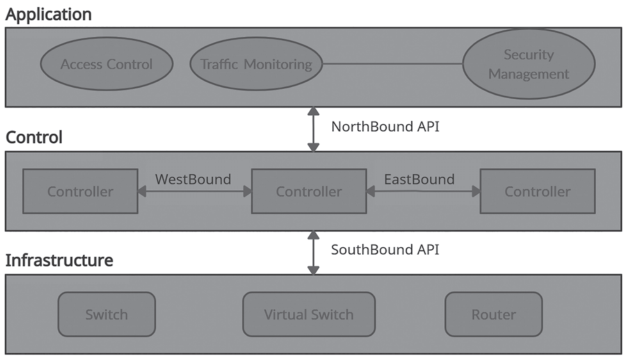


Fig. 1. Conceptual architecture of SDN

OpenFlow has numerous points of interest, it supports virtualization and includes all specifications to detect network vulnerabilities, applying security policies and communications between the controller and the forwarding device.

DHCP protocol. DHCP [6] stands for the dynamic host configuration protocol and is used on the IP networks where the DHCP server assigns IP addresses and other important information(such as default gateway and domain name server DNS) to hosts automatically when connected so they can communicate through the network.

DHCP greatly simplifies the setup and ensures that devices can join the network with the correct configurations. However, in some situations, the users need to use static IP addresses for some reason or because other devices connect to them frequently. If the user assigns himself an IP address that is already used by other devices, an IP address conflict occurs causing that one or both devices will lose connection until the conflict has been solved. To solve the conflict problem, most systems use ARP packets to detect a duplicate IP address.

ARP (Address Resolution Protocol):

— Traditional ARP. ARP is an OSI two-layer protocol specified in document RFC 826 [7]. Its main function is to find a MAC address for a known IP address. For this mapping process, it maintains an ARP table with IP-MAC mapping which stores in the RAM of the device. The ARP frame contains the IP and the MAC address of the sender and the receiver, it also has an operation code to clarify the ARP type message (1 for ARP request, 2 for the reply).

As we have mentioned, the ARP protocol is used to detect the duplicate address problem in the network, this can be done using the two ARP packets types:

— ARP probe: is an ARP request packet that is sent as broadcast frames to validate if an IP address is free to use. So, if the host wants to assign himself a new static IP address, first of all, it will send an ARP probe packet polling the network if this IP is already in use. If a host already has this address, it will answer with an ARP reply.

— ARP announcement: if the IP address is not in use and the ARP probe does not generate any response, then the host will start to announce that it has this IP address now by broadcasting an ARP request as an ARP announcement.

ARP Spoofing attack in SDN. ARP poisoning cache attack was always one of the most serious attacks in SDN due to the SDN design, where the attacker could get control over the entire network by impersonating the controller identity. Using faked ARP requests or replies, the attacker could poison the ARP cache of the controller or other hosts, which paves the way for other attack forms such as DOS, MITM attacks, etc.

— Spoofing in regular ARP: It is similar to an ARP spoofing attack in the traditional network. The attacker uses fake ARP requests or replies to start the attack and poison the ARP cache of other hosts in the network, taking advantage of that the controller does not have a mechanism to detect the attack and will forward the packets without checking.

— Spoofing in Proxy ARP: It's a more serious attack and will get the attacker a chance to control the whole network. In proxy ARP, the attacker will send fake packets and impersonate the identity of other hosts in the network to poison the controller cache.

Related works

Since 2010 several real studies have been carried out in SDN security and effective solutions have been proposed to detect and mitigate ARP spoofing attacks in SDN. In the following section, we will focus on the common and latest research in this field and their limitations in performance or other critical features:

FICUR. FICUR [8] extends the POX controller with an application that monitors and analyzes the ARP traffic to detect and mitigate the ARP spoofing attack according to pre-defined traffic patterns. FICUR can handle attacks against Proxy and Regular ARP. FICUR results described in [8] show that this solution can detect the ARP spoofing attack in 16.056 ms. However, this solution is still not effective for large-scale networks, the author also did not consider the delay in performance metric evaluation. Other works like FICUR based on the traffic pattern is "OrchSec" which has its limitations too.

SPHINX. This solution presented in [2] uses the OpenFlow messages to create flow graphs, then employs the metadata maintained by the flow to detect the ARP spoofing attacks. This mechanism works correctly to achieve the goal and can be successfully deployed over four different controllers. The experiment with 14 switches and 10 servers in [2] shows that the mechanism can quickly detect the ARP spoofing attack with an attack detection time of 44 μs. However, this solution needs a long time to collect and modify the metadata which affects the efficiency of the detection process. DistBlockNet is another solution that depends on the flow graph mechanism, but it was designed for IoT-based networks and it is not ready to be used in traditional networks.

Solution based on IP_AC address binding. An example of this type of solution is SARP_NAT [9]. SARP_NAT simply stores a database with MAC address and IP address for hosts that have sent an ARP request and check always if an incoming ARP reply is a response to a request that was stored in the database. But this solution also has its shortcomings as it causes too much load on the controller for large-scale networks. Other solutions in this category are Scalable Ethernet Traffic Architecture Using SDN (SEASDN) [10], L3-ARPSEC [11], Secure Binder [12], NetWatch [13], NFGA [14]. They use maintained database for IP_MAC bindings with different proposed algorithms to detect and mitigate the attack [15]. However, each one has its limitations either in CPU utilization, response time, or other important metrics.

Patches. Solutions such as "Anticap[1] and Antidote" uses the OS patches to mitigate the ARP spoofing attack but this solution has its limitations because it requires different

[1] M. Barnaba and M. Barnaba, "anticap." Available at: https://github.com/antifork/anticap (accessed: 17.04.2021).

patches for different operation systems, which effects the whole communication process.

ARP cryptographic. Using cryptography in communication is still a reliable solution, but the encryption and decryption process causes overload on the network and consumes more resources. S-ARP is a proposed mechanism that uses cryptography to protect nodes in the network [16].

Software tools. Tools like XArp[1] and ARPWatch[2] are also used to detect the attack in SDN but still have their trust problems such as false positives and false negatives.

Static ARP mapping. This solution stops the attack by using the manual configuration of ARP entries. Thus, the attacker has no chance to spoof an already registered host. Works [17, 18] are an example of such a mechanism, but it is obvious that these solutions are effective only in small-scale networks and cause a great overhead on large-scale ones.

The given short review strengthened our motivation to start this work and come up with a new solution, that should detect and mitigate the ARP spoofing attack in SDN within a short time and limit overload on the controller.

The proposed solution

To detect the ARP cache poisoning attack, we have extended the POX controller with a new module that plays the role of a firewall. This module analyses incoming traffic and sorts the sender according to pre-defined conditions.

Our new module will work exactly as the L2 learning switch module in forwarding packets and install rules to switches, but also it will effectively analyze ARP packets and mitigate the ARP spoofing attack in a short time.

This solution is effective because the new POX controller module analyses all ARP types and effectively detects the attack with no effects on the ARP normal work. It also considers the two ways to assign IP to the host (static and automatic using DHCP). Also, it prevents the controller from faked ARP flooding and stops the attacker in a very short time. It also includes a mechanism to block any host who wants to flood the controller with a huge number of packets and to affect its decision. Our three classes mechanism provides an easy way to classify the hosts in the network, which can be used in further solutions to stop other attacks on SDN.

Solution design. The component was extended with our proposed algorithm, with no need for additional software or hardware to be installed. Also, the ARP spoofing part in the algorithm does not affect the normal L2 component to correctly forward packets and provide communications between hosts in the network.

Intercept packets. Our mechanism will process the ARP packets only, so for incoming packets to the controller, we will filter packets according to their type. ARP packets will enter the function implementing our algorithm, other packets will be forwarded as usual, according to the rules included in the L2 forwarding switch module.

Classify the hosts according to the current situation. The main purpose of this mechanism is to classify the hosts in our SDN network into three classes, so that it will be easier to decide how to deal with incoming packets:
- VHT: Verified hosts table, contains the hosts that we have already verified. While they have no changes in their addresses and act in a normal way, their packets will be forwarded and can be used to verify the states of other new hosts.
- CHT: The candidate host table, contains the hosts which are in the middle way to be in the VHT, they are still acting as usual, but they did not finish yet the verifying steps.
- BHT: Stands for the banned host table.The attacker will be classified in this table, so the controller will direct the switches to drop their packets for some time that can be managed.

Build the mechanism to detect the attack. Our algorithm starts processing the ARP packets by checking the source MAC address of the Ethernet packet and the source MAC address included in the ARP packet.

If the two MAC addresses are not the same, the controller will give a warning, that a new ARP spoofing attack was detected and direct the switches to drop the packet and block the attacker for some time.

Another important mechanism in our solution is to count the similar ARP packets on the switches and the controller, which have the same metadata. The algorithm uses a separate process to count the number of similar incoming packets during some periods and will detect an ARP spoofing attack when this parameter will satisfy some conditions.

This condition will protect the controller against critical attacks which will try to fool our algorithm and send professional tricky fake packets.

The main part of the algorithm works as follows the DHCP server:
1. If the host asks the DHCP server to give an IP, then the DHCP server is responsible for giving a trusted IP address and confirms that this IP is not spoofed. Receiving a “DHCP ACK” from the server will announce that this IP is verified and the controller will add it to the VHT.
2. If the host is trying to set its IP address manually, then the controller will consider the following conditions:
   - If the host is new and has no entry in the VHT, then look at its packet series. To pass the checking process and to be registered in the VHT, the host should first send a number of ARP probes (register in the CHT) and then an ARP announcement. The announcement should follow the ARP probe after some time with no responses from other hosts that this IP is already in use.
   - If the host is already in the VHT and sends packets, they will be forwarded, taking into account that if the packet is an ARP reply sent as a response to the ARP probe that an IP address already in use, then

[1] Christoph Mayer. XArp — Advanced ARP Spoofing Detection. 2018. Available at: http://www.xarp.net/ (accessed: 17.04.2021).

[2] Arpwatch. Lawrence Berkeley National Laboratory. 2009. Available at: ftp://ftp.ee.lbl.gov/arpwatch.tar.gz (accessed: 17.04.2021).

the corresponding candidate host will be removed from the CHT.

To ensure that the controller will deal correctly with the attacker attempts aiming to poison the ARP cache, let us check the attacker's states while it is trying to start the ARP spoofing attack:

1. It has not been registered and has not asked for an IP address yet but it wants to start the attack immediately. Hence, the attacker will try to send an ARP probe (register in the CHT) and then an ARP announcement to spoof some IP, but after receiving a response from any host that it has already this IP the attacker will be removed from the CHT. Sending the ARP announcement without an entry in the CHT will then detect the attacker and add him to the BHT. Any other behavior from the attacker will cause also its blocking and detecting the attack immediately.
2. The second situation is that it had already been registered and had taken its place in the VHT before it decided to start the attack. If the attacker wants to spoof any other address, then he has to send packets with different addresses and will not consider in the VHT and the controller will act with him exactly in the same way as described above.

The flowchart. The flowchart of our proposed solution to detect and mitigate the ARP poisoning attack is shown below (Fig. 2).

Implementation

Environment setup. To test our mechanism, Mininet version 2.2.2 was used with 2 cores and 2 GB of RAM. The host machine has Windows 10 installed as the operating system and has an Intel Core i5 processor with 6 GB of RAM. The POX controller was installed as a remote controller Ubuntu Virtual Machine with 2 cores and 2 GB of RAM and connected to the Mininet via a virtual interface. As mentioned before, a number of components (such as forwarding.l2_learning, openflow.webservice, openflow.of_01, Log, host_tracker) are used in

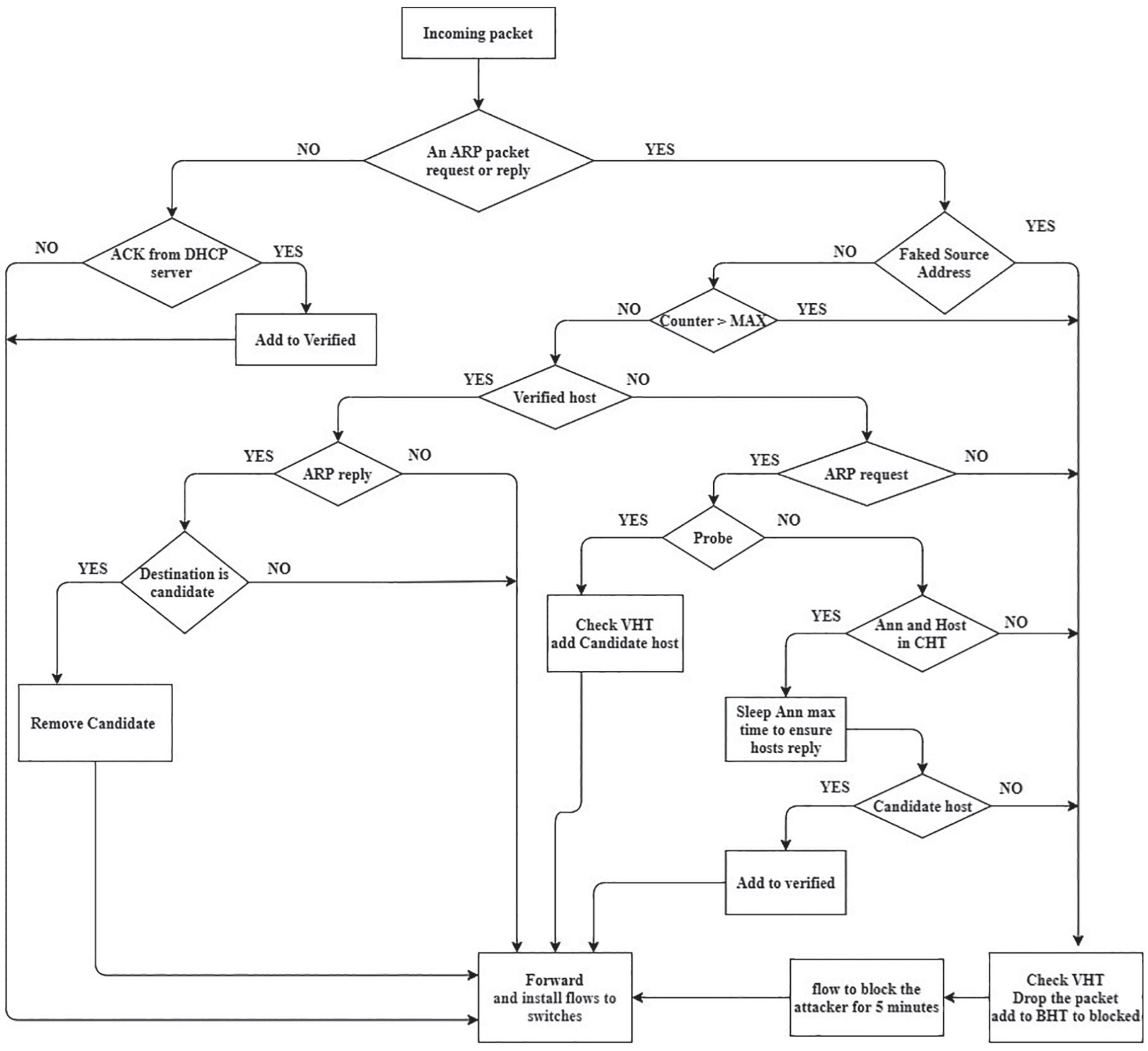


Fig. 2. Flowchart to detect and mitigate the ARP poisoning attack

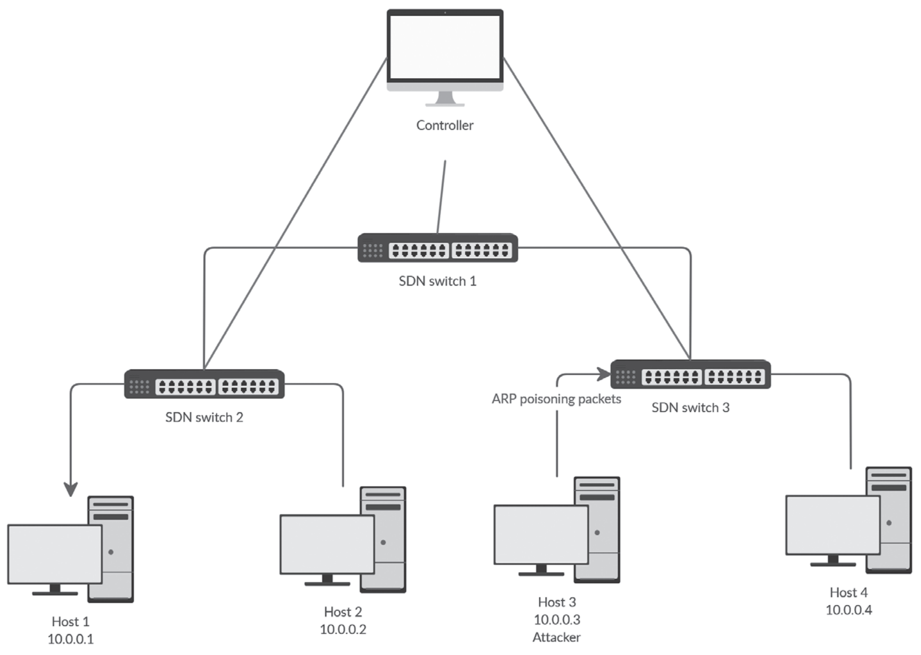
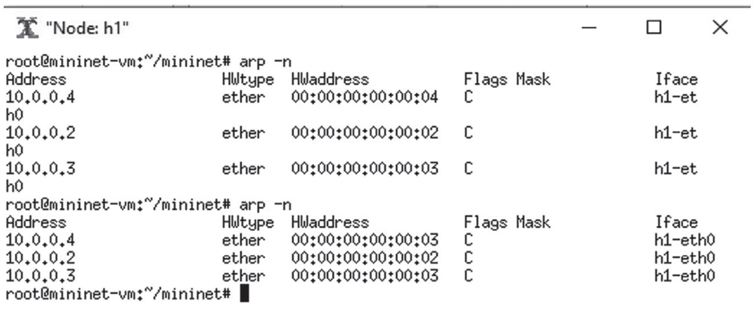


Fig. 3. Topology of SDN network used in the experiments



```
"Node: h1"
root@mininet-vm:~/mininet# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.4                 ether   00:00:00:00:00:04   C                     h1-et
h0
10.0.0.2                 ether   00:00:00:00:00:02   C                     h1-et
h0
10.0.0.3                 ether   00:00:00:00:00:03   C                     h1-et
h0
root@mininet-vm:~/mininet# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.4                 ether   00:00:00:00:00:03   C                     h1-eth0
10.0.0.2                 ether   00:00:00:00:00:02   C                     h1-eth0
10.0.0.3                 ether   00:00:00:00:00:03   C                     h1-eth0
root@mininet-vm:~/mininet# █
```

Fig. 4. Poisoning cache on h1

the experiment. All tests were performed on the topology shown in Fig. 3. In order to collect and gather statistics about the connectivity and throughput in our network, we used the Iperf tool[1]. In this experiment, all links of the network will have 100 Mbps bandwidth, 5 ms delay, 0 % loss and maximum packet queue size equal to 1000. We had to limit these parameters to prevent the load on the controller and get more accurate results.

Test Scenario. Using the arpspoof tool in the dsniff package[2], various spoofed ARP packets were forwarded to the victim to simulate a real ARP spoofing attack.

The experiment was performed according to the topology shown in Fig. 3: a network with Controller, 3 switches, and 4 hosts. Using the arpsoof tool, the attacker (Host 3) sent the faked packets to the victim (Host 1).

Fig. 4 shows the result of the ARP spoofing attack. The second command was executed after the attack had been performed.

[1] iPerf — The TCP, UDP and SCTP network bandwidth measurement tool. Available at: https://iperf.fr/ (accessed: 17.04.2021).

[2] dsniff : Bionic (18.04) : Ubuntu. Available at: https://launchpad.net/ubuntu/bionic/+package/dsniff (accessed: 17.04.2021).

Performance evaluation metrics

For the previous testing scenario, the proposed solution has worked perfectly to detect and mitigate the ARP spoofing attack. It was able to drop the spoofed packets and block the attacker for all the following attempts to poison the victim's ARP cache:
— a spoofed MAC address of the ARP header;
— a spoofed MAC address of the Ethernet header;
— spoofed MAC addresses of the ARP header and Ethernet header;
— a faked IP address of the sender;
— a faked IP address of the destination.

Here are important metrics, which were measured during the experiment.

Attack Detection Time. This metric is defined as the total time elapsed to detect the ARP Cache Poisoning attack, Fig. 5 presents the time elapsed for the process of detection of the ARP spoofing attack. It shows that the time elapsed for the detection is very short and it took only 2 ms to detect faked ARP requests and 1.7 ms for ARP replies.

Attack Mitigation Time. This metric is defined as the total time elapsed from detecting the attack to completely blocking the attacker. The time elapsed for the mitigation of

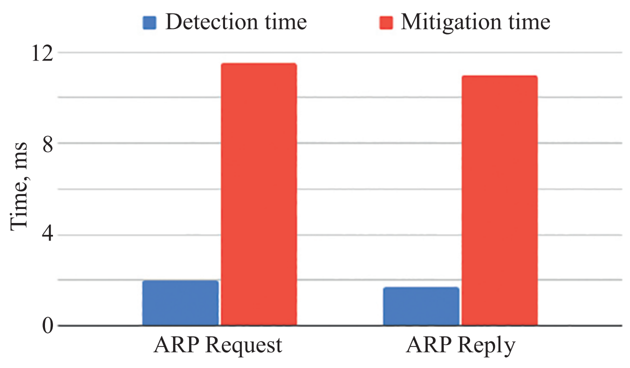


Fig. 5. Detection and mitigation time results

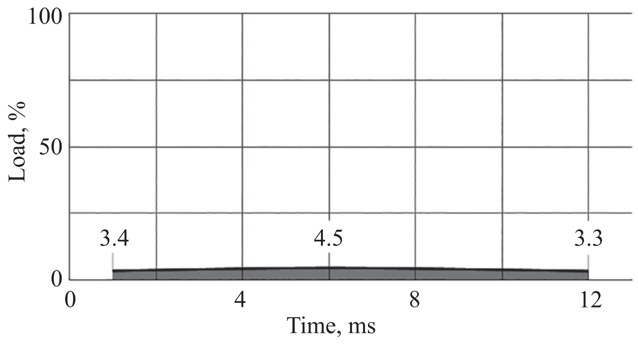


Fig. 6. CPU Utilization on the POX controller

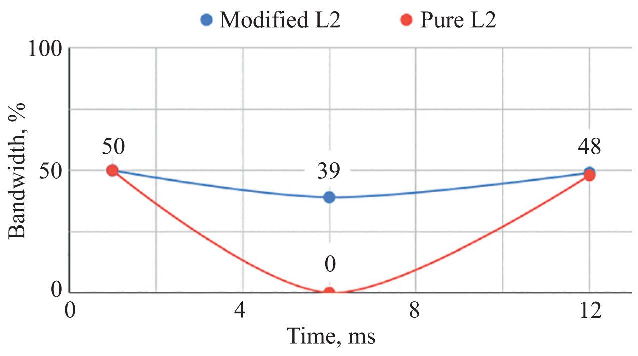


Fig. 7. Throughput test between the host and the victim

the ARP spoofing attack was presented in Fig. 5. It shows that the time elapsed for the mitigation process is very short too, and it took only 11 ms for attacks with ARP requests and 11.5 ms with ARP replies.

CPU Utilization of the Controller. It should be noted that the controller plays a significant role in the network, runs applications and performs many tasks. Thus, CPU utilization is a very important metric that should be taken into account. The NMON tool[1] was used to record the results. And the CPU usage was measured before, during and after the attack. According to (Fig. 6), we can notice that the CPU usage with the modified component, was increased from 3.4 % to 4.5 % (by about 0.9 %) during the attack. Then the module successfully stopped the attack and the CPU usage returned back to 3.3 %. This result is achieved due to the algorithm we built. Here the controller processes incoming ARP packets, while the white table helps to make fast decisions and consume more CPU utilization only in dubious situations during the attacks where the controller requires more load to process the verification from the correct host and block the attacker.

Throughput. Using the Iperf tool we performed the throughput test for the topology shown in Fig. 3. All links of the network will have 100 Mbps bandwidth, 5 ms delay. We compared two throughput scenarios, the first one with the normal L2 learning component in the POX controller (pure L2), another one is our modified component with the mitigation mechanism (modified L2). In this scenario, a TCP load has been generated using the Iperf and sent from Host 4 to Host 1. According to (Fig. 7), we can notice that the controller with the pure L2 component is hardly affected by the attack, and the throughput reached 0 in some moments. With the modified component, we can notice that the throughput has decreased only 10 % of the bandwidth and back to its normal situation in a very short time after blocking the attacker. The figure below (Fig. 7) shows the throughput test results before, during and after the attack for the two scenarios.

[1] nmon for Linux | Main / HomePage. Available at: http://nmon.sourceforge.net/pmwiki.php (accessed: 17.04.2021).

Conclusion

In this work, we presented a new mechanism to prevent the software-defined networks against ARP spoofing attack. The solution works effectively with minimum latency to mitigate both ARP requests and replies attacks. The new mechanism involves the DHCP and manual assignment of IP addresses using three classes to classify the hosts according to their situations in the network. The CHT helps to set the host in an intermediate state between verifying and banning and detect the attack according to the next step of the host. This proposed solution neither has an extra overload in the network, nor requires any changes in the infrastructure or additional hardware to install. Moreover, it does not affect the usual work of the protocols like ARP or OpenFlow. This work has also presented a review of the proposed solutions against ARP spoofing attacks in the SDN environment. According to the experiment results of this solution, the average time to detect the ARP spoofing attack is very short, with minor overhead on the controller CPU.

For the current work, the new mechanism its limitations, namely, the use of a single controller network and testing in a virtual environment. For future work, the solution can be further developed to handle ARP attacks in multiple controller SDN networks with high availability, and it is highly recommended to test the described method on a physical testbed to observe the performance parameters in real network environments.

References

1. Kreutz D., Ramos F.M.V., Verissimo P.E., Rothenberg C.E., Azodolmolky S., Uhlig S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 2015, vol. 103, no. 1, pp. 14–76. doi: 10.1109/JPROC.2014.2371999
2. Dhawan M., Poddar R., Mahajan K., Mann V. SPHINX: Detecting security attacks in software-defined networks. *Proc. 2015 Network and Distributed System Security Symposium*, 2015, pp. 8–11. doi: 10.14722/ndss.2015.23064
3. Hong S., Xu L., Wang H., Gu G. Poisoning network visibility in software-defined networks: New attacks and countermeasures. *Proc. 2015 Network and Distributed System Security Symposium*, 2015. doi: 10.14722/ndss.2015.23283
4. Feamster N., Rexford J., Zegura E. The Road to SDN: An intellectual history of programmable networks. *Queue*, 2013, vol. 11, no. 12, pp. 2560327. doi: 10.1145/2559899.2560327
5. Nathan A.J. Scobell A. How China sees America: The Sum of Beijing's Fears. *Foreign Affairs*, 2012, vol. 91, no. 5, pp. 32–47.
6. Droms R. *RFC 2131 — Dynamic Host Configuration Protocol. 1997.* Available at: https://tools.ietf.org/html/rfc2131 (accessed: 04.11.2020).
7. Plummer D. *An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*. doi: 10.17487/RFC0826
8. Nehra A., Tripathi M., Gaur M.S. FICUR: Employing SDN programmability to secure ARP. *Proc. 7th IEEE Annual Computing and Communication Workshop and Conference. (CCWC)*, 2017, pp. 7868450. doi: 10.1109/CCWC.2017.7868450
9. Alharbi T., Durando D., Pakzad F., Portmann M. Securing ARP in Software Defined Networks. *Proc. 41st IEEE Conference on Local Computer Networks (LCN)*, 2016, pp. 523–526. doi: 10.1109/LCN.2016.83
10. Jehan N. Haneef A.M. Scalable Ethernet Architecture using SDN by Suppressing broadcast traffic. *Proc. 5th International Conference on Advances in Computing and Communications (ICACC)*, 2015, pp. 24–27. doi: 10.1109/ICACC.2015.66
11. De Oliveira R., Shinoda A., Schweitzer C., Iope R., Prete L. L3-ARPSec — A Secure Openflow Network Controller Module to control and protect the Address Resolution Protocol. *Proc. XXXIII Simpósio Brasileiro de Telecomunicações*, 2015, pp. 1–4. doi: 10.14209/sbrt.2015.29
12. Jero S., Koch W., Skowyra R., Okhravi H., Nita-Rotaru C., Bigelow D. Identifier binding attacks and defenses in software-defined networks. *Proc. 26th USENIX Security Symposium*, 2017, pp. 415–432.
13. Balagopal D., Rani X.A.K. NetWatch: Empowering software-defined network switches for packet filtering. *Proc. 1st International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, 2015, pp. 837–840. doi: 10.1109/ICATCCT.2015.7456999
14. Cox J.H., Clark R.J., Owen H.L. Leveraging SDN for ARP security. *Proc. IEEE SoutheastCon 2016*, 2016, pp. 7506644. doi: 10.1109/SECON.2016.7506644
15. Shah Z., Cosgrove S. Mitigating ARP Cache Poisoning attack in Software-Defined Networking (SDN): A survey. *Electronics*, 2019, vol. 8, no. 10, pp. 1095. doi: 10.3390/electronics8101095
16. Bruschi D., Ornaghi A., Rosti E. S-ARP: A secure address resolution protocol. *Proc. 19th Annual Computer Security Applications Conference (ACSAC)*, 2003, pp. 66–74. doi: 10.1109/CSAC.2003.1254311
17. Hou X., Jiang Z., Tian X. The detection and prevention for ARP Spoofing based on Snort. *Proc. 2010 International Conference on Computer Application and System Modeling (ICCASM)*, 2010, pp. V5137–V5139. doi: 10.1109/ICCASM.2010.5619113
18. Ortega A.P., Marcos X.E., Chiang L.D., Abad C.L. Preventing ARP cache poisoning attacks: A proof of concept using OpenWrt. *Proc. 6th IEEE/IFIP Latin American Network Operations and Management Symposium (LANOMS)*, 2009, pp. 5338799. doi: 10.1109/LANOMS.2009.5338799

Литература

1. Kreutz D., Ramos F.M.V., Verissimo P.E., Rothenberg C.E., Azodolmolky S., Uhlig S. Software-defined networking: A comprehensive survey // Proceedings of the IEEE. 2015. V. 103. N 1. P. 14–76. doi: 10.1109/JPROC.2014.2371999
2. Dhawan M., Poddar R., Mahajan K., Mann V. SPHINX: Detecting security attacks in software-defined networks // Proc. 2015 Network and Distributed System Security Symposium. 2015. P. 8–11. doi: 10.14722/ndss.2015.23064
3. Hong S., Xu L., Wang H., Gu G. Poisoning network visibility in software-defined networks: New attacks and countermeasures // Proc. 2015 Network and Distributed System Security Symposium. 2015. doi: 10.14722/ndss.2015.23283
4. Feamster N., Rexford J., Zegura E. The Road to SDN: An intellectual history of programmable networks // Queue. 2013. V. 11. N 12. P. 2560327. doi: 10.1145/2559899.2560327
5. Nathan A.J. Scobell A. How China sees America: The Sum of Beijing's Fears // Foreign Affairs. 2012. V. 91. N 5. P. 32–47.
6. Droms R. RFC 2131 — Dynamic Host Configuration Protocol. 1997 [Электронный ресурс]. URL: https://tools.ietf.org/html/rfc2131 (дата обращения: 04.11.2020).
7. Plummer D. An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. doi: 10.17487/RFC0826
8. Nehra A., Tripathi M., Gaur M.S. FICUR: Employing SDN programmability to secure ARP // Proc. 7th IEEE Annual Computing and Communication Workshop and Conference (CCWC). 2017. P. 7868450. doi: 10.1109/CCWC.2017.7868450
9. Alharbi T., Durando D., Pakzad F., Portmann M. Securing ARP in Software Defined Networks // Proc. 41st IEEE Conference on Local Computer Networks (LCN). 2016. P. 523–526. doi: 10.1109/LCN.2016.83
10. Jehan N. Haneef A.M. Scalable Ethernet Architecture using SDN by Suppressing broadcast traffic // Proc. 5th International Conference on Advances in Computing and Communications (ICACC). 2015. P. 24–27. doi: 10.1109/ICACC.2015.66
11. De Oliveira R., Shinoda A., Schweitzer C., Iope R., Prete L. L3-ARPSec — A Secure Openflow Network Controller Module to control and protect the Address Resolution Protocol // Proc. XXXIII Simpósio Brasileiro de Telecomunicações. 2015. P. 1–4. doi: 10.14209/sbrt.2015.29
12. Jero S., Koch W., Skowyra R., Okhravi H., Nita-Rotaru C., Bigelow D. Identifier binding attacks and defenses in software-defined networks // Proc. 26th USENIX Security Symposium. 2017. P. 415–432.
13. Balagopal D., Rani X.A.K. NetWatch: Empowering software-defined network switches for packet filtering // Proc. 1st International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT). 2015. P. 837–840. doi: 10.1109/ICATCCT.2015.7456999
14. Cox J.H., Clark R.J., Owen H.L. Leveraging SDN for ARP security // Proc. IEEE SoutheastCon 2016. 2016. P. 7506644. doi: 10.1109/SECON.2016.7506644
15. Shah Z., Cosgrove S. Mitigating ARP Cache Poisoning attack in Software-Defined Networking (SDN): A survey // Electronics. 2019. V. 8. N 10. P. 1095. doi: 10.3390/electronics8101095
16. Bruschi D., Ornaghi A., Rosti E. S-ARP: A secure address resolution protocol // Proc. 19th Annual Computer Security Applications Conference (ACSAC). 2003. P. 66–74. doi: 10.1109/CSAC.2003.1254311
17. Hou X., Jiang Z., Tian X. The detection and prevention for ARP Spoofing based on Snort // Proc. 2010 International Conference on Computer Application and System Modeling (ICCASM). 2010. P. V5137–V5139. doi: 10.1109/ICCASM.2010.5619113
18. Ortega A.P., Marcos X.E., Chiang L.D., Abad C.L. Preventing ARP cache poisoning attacks: A proof of concept using OpenWrt // Proc. 6th IEEE/IFIP Latin American Network Operations and Management Symposium (LANOMS). 2009. P. 5338799. doi: 10.1109/LANOMS.2009.5338799

Authors

Ghadeer Darwesh — Student, ITMO University, Saint Petersburg, 197101, Russian Federation, http://orcid.org/0000-0003-1116-9410, ghadeerdarwesh32@gmail.com

Авторы

Дарвиш Гадир — студент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, http://orcid.org/0000-0003-1116-9410, ghadeerdarwesh32@gmail.com

Alisa A. Vorobeva — PhD, Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, SC 57191359167, http://orcid.org/0000-0001-6691-6167, Alice_w@mail.ru

Viktoriia M. Korzhuk — PhD, Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, SC 56875395200, http://orcid.org/0000-0002-0240-9067, vmkorzhuk@itmo.ru

Воробьева Алиса Андреевна — кандидат технических наук, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, SC 57191359167, http://orcid.org/0000-0001-6691-6167, Alice_w@mail.ru

Коржук Виктория Михайловна — кандидат технических наук, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, SC 56875395200, http://orcid.org/0000-0002-0240-9067, vmkorzhuk@itmo.ru

Received 30.03.2021
Approved after reviewing 13.04.2021
Accepted 07.06.2021

Статья поступила в редакцию 30.03.2021
Одобрена после рецензирования 13.04.2021
Принята к печати 07.06.2021