Hindawi
Security and Communication Networks
Volume 2022, Article ID 2196998, 9 pages
https://doi.org/10.1155/2022/2196998

Research Article

ASD: ARP Spoofing Detector Using OpenWrt

Yeonseon Jeong,[1] **Hyunghoon Kim**,[2] **and Hyo Jin Jo**[2]

[1]*Korea University, Seoul, Republic of Korea*
[2]*Soongsil University, Seoul, Republic of Korea*

Correspondence should be addressed to Hyo Jin Jo; hyojin.jo@ssu.ac.kr

Received 29 October 2021; Accepted 26 January 2022; Published 29 March 2022

Academic Editor: Ilsun You

The address resolution protocol (ARP) is one of the most important communication protocols in a local area network (LAN). However, since there is no authentication procedure, the ARP is vulnerable to cyberattack such as ARP spoofing. Since ARP spoofing can be connected to critical attacks, including a man-in-the-middle (MITM) attack, detecting ARP spoofing initially without returning false-positive alarms is important. In general, however, existing works for ARP spoofing are unable to distinguish between ARP spoofing and connections from virtual machine (VM) guests, which results in false-positive alarms. In this article, we propose an access point-based ARP Spoofing Detector (ASD) that can detect ARP spoofing attacks without returning a false-positive rate. Our proposed system distinguishes between ARP spoofing and connections from VM guests using three information tables, *AssocList*, *ARP cache table*, and *DHCP table*, which are commonly managed by the access point based on a Linux system. We evaluated the performance of ASD on ARP spoofing attack experiments.

1. Introduction

The address resolution protocol (ARP) is one of the most important communication protocols in a local area network (LAN). The MAC addresses of the client are required to communicate with said clients on a LAN, and the ARP is a procedure that determines the MAC address of a given IP address. For example, if Client A wants to communicate with Client B, Client A may send a broadcast ARP request, to which Client B, after receiving this broadcast, responds in kind to Client A with an ARP reply containing its MAC address.

However, there is no authentication procedure for ARP reply packets, meaning that the ARP is not equipped to handle malicious ARP reply packets. With knowledge of this fact, an attacker can perform an ARP spoofing attack on a target after modifying some ARP reply packets. ARP spoofing is particularly dangerous as it can easily lead to a variety of subsequent attacks such as sniffing, denial-of-service (DoS), and man-in-the-middle (MITM) attacks [1].

Unfortunately, existing solutions to the above-mentioned ARP spoofing attacks require modification of the ARP itself or installation of expensive hardware equipment (e.g., costly switches and servers). ARP spoofing attempts can be easily detected by checking the IP/MAC address mapping of a suspected target's cache table. If there is a MAC address that is mapped to more than two IP addresses in the ARP cache table, it is reasonable to suspect that the MAC address is the target of at least one ARP spoofing attack. However, this approach can also generate false-positive alarms when there is a virtual machine (VM) guest on a bridged network. To handle this particular ARP spoofing detection issue regarding VM guests on a bridged network, we propose a new intrusion detection system called ARP Spoofing Detector (ASD) that, when installed on an access point (AP), can detect ARP spoofing attacks on a LAN managed by the ASD-enabled AP without any additional hardware equipment. In addition, ASD can distinguish fake MAC addresses forged by an attacker from benign MAC addresses of VM guests on a bridged network. To our knowledge, this problem has not yet been resolved by any existing ARP spoofing detection methods. For the evaluation, ASD was implemented in OpenWrt [2], a Linux-based, open-source operating system (OS), and ARP spoofing attack tests were performed several times to verify whether or

not ASD could successfully and consistently detect ARP spoofing without returning false-positive alarms. The contributions of this article are described as follows:

(i) The proposed system, ASD, can distinguish ARP spoofing attack attempts and network connections from a VM guest on a bridged network. In addition, compared to existing works, ASD does not require additional equipment or modification of ARP processes.

(ii) ASD was implemented on an OpenWrt-based AP and its ability to handle ARP attacks without returning any false-positive rates. Additionally, ARP spoofing attack attempts can be monitored using a visualization tool provided in ASD.

This article is organized as follows. The background and related works are discussed in Section 2. Section 3 defines our attack model and attack scenario for ARP spoofing. Sections 4 and 5 describe the architecture of our proposed ARP Spoofing Detector (ASD) and corresponding evaluation results, respectively. Finally, Section 6 presents our conclusion.

2. Background and Related Works

2.1. Background

2.1.1. Address Resolution Protocol (ARP). The ARP is a protocol that maps logical addresses (i.e., IP addresses) to physical addresses (i.e., MAC addresses) on a network [3]. It is used to communicate between layer 2 and layer 3 in IPv4, and it stores the IP/MAC address pairings in a local ARP cache table. The ARP is constructed of requests and replies. When a client wants to know the MAC address for another specific client's IP address, it sends an ARP request packet to all clients on the same LAN. The client with the corresponding IP address responds using an ARP reply packet, which includes its MAC address.

2.1.2. ARP Spoofing Attack. ARP request and reply packets are used to manage the ARP cache table, which contains information that maps IP addresses to MAC addresses. However, there is no authentication process for ARP reply packets, so attackers can exploit this vulnerability and launch a variety of ARP spoofing attacks at this weak point. For example, if a host receives a malicious ARP reply packet forged by an attacker, the host unwittingly updates the ARP cache table without detecting the malicious packet and, subsequently, the host's MAC address becomes changed to the attacker's MAC address on the host's ARP cache table.

An attacker can easily launch any ARP spoofing attack with suites and tools such as Ettercap and Cain & Abel [4] and can likewise execute a MITM attack and intercept the communications between two clients and forge messages to each one [5]. Recently, MITM attacks have come under particular scrutiny as a security threat for not only PCs but also IoT devices [6]. Figure 1 shows an example of ARP spoofing-based MITM attacks.

2.2. Related Works.

Existing solutions for ARP spoofing can be divided into two types. One type involves preventing ARP spoofing (i.e., the prevention type), which requires modifying the ARP, while the other involves detecting ARP spoofing (i.e., the detection type).

2.2.1. Prevention Methods for ARP Spoofing. Table 1 represents the summary of existing research on preventing ARP spoofing attacks. All existing solutions face noteworthy limitations like the need to install an additional device, production of heavy network traffic, or addition of heavy computation overhead. We classify the existing solutions into four categories: (1) a hardware-based approach, (2) a cryptography-based approach, (3) a voting-based approach, (4) a central server-based approach, and (5) a dynamic host configuration protocol-based approach. In the hardware-based approach [7], installation of additional devices is necessary, which results in several hardware management burdens for the user, like installation and management costs. In cryptography-based approaches [8–10], existing solutions unfortunately produce significant cryptographic operation overhead during operations like key exchange protocol and verification. Bruschi et al. [8] proposed a secure address resolution protocol (S-ARP). In this protocol, all hosts have a public/private key pair distributed by the authoritative key distributor (AKD), and messages are signed by a digital signature algorithm for the prevention of ARP spoofing. Lootah et al. [9] introduced ticket-based address resolution protocol (TARP) to reduce the computational resources of S-ARP using an additional entity called Local Ticket Agent (LTA). Likewise, Goyal and Tripathy [10] presented an enhanced S-ARP mechanism that adopts one-time passwords based on hash chains. In voting-based approaches [11, 12], existing solutions create burdensome network traffic due to the high communication cost between voters. Nam et al. [11] presented a MITM-resistant address resolution protocol (MR-ARP) based on two key concepts, namely, long-term IP/MAC mapping table and voting. To overcome the limitation of MR-ARP, which requires far too much computational time, [12] suggested the Generalized MR-ARP (GMR-ARP), which increases the fairness of voting by dropping reply packets that appear too early. In the central server-based approaches [13–15], existing solutions have a single point of failure problem that could occur in the central server. Gouda and Huang [13] proposed a secure server that uses two protocols: an invite-accept protocol and a request-reply protocol. Demuth and Leitner [14] introduced a multiple server-based method that requires three additional servers for gateway protection, client protection, and server protection. Ortega et al. [15] presented a two-pronged OpenWrt-based server and switch method. The server responds to an ARP request with IP/MAC address mapping, while the switch blocks all ARP responses.

In dynamic host configuration protocol- (DHCP-) based approaches [16–18], the existing solutions based on the DHCP table are limited by the VM connection. Masoud et al. [16] updated the ARP cache table based on DHCP. When the network's user receives any ARP reply, the ARP reply is

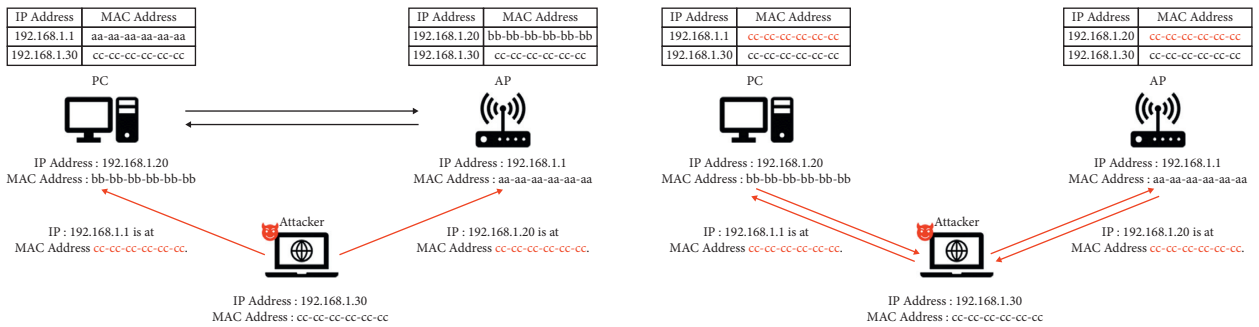

Figure 1: Attack scenario for an ARP spoofing-based MITM attack.

Table 1: Summary of existing prevention methods for ARP spoofing attacks.

Category	Research work	Pros/cons
Hardware-based	Bhaiji [7]	Only available in certain switches
Cryptography-based	Bruschi et al. [8]	Heavy performance overhead due to the use of a public key algorithm
	Lootah et al. [9]	It requires an additional entity called ticket agent
	Goyal et al. [10]	The extended work of [8]
Voting-based	Nam et al. [11]	Based on a voting system that is slower than a rule-based system
	Nam et al. [12]	The extended work of [11]
Central server-based	Gouda and Huang [13]	It requires modification of the ARP protocol used by all hosts
	Demuth and Leitner [14]	It requires three additional servers and is, thus, not cost-effective
	Ortega et al. [15]	It uses the OpenWrt
DHCP-based	Masoud et al. [16]	It can create overhead at the SDN controller
	Cox et al. [17]	It could install incorrect drop flow rules for genuine hosts in multiple switches scenario
	Sun et al. [18]	It is used in SDN-based cloud computing environments

dropped if the IP/MAC address mapping in the ARP reply is not the same as IP/MAC address in the DHCP reply. Since this approach analyzes all packets, this will increase the computational overhead of the network controller, that is, a server on which developers can write proprietary or customized topologies and algorithms in the software-defined network (SDN). Cox et al. [17] presented Network Flow Guard (NFG) as a modular application to detect and prevent ARP replies from unauthorized hosts by monitoring DHCP from valid DHCP servers. Sun et al. [18] took advantage of DHCP to obtain reliable IP/MAC address mapping in order to ensure the accuracy of ARP attack detection. However, ARP cache table management based on DHCP does give a VM guest on a bridged network an error.

2.2.2. Detection Methods for ARP Spoofing. There are several detection methods for ARP spoofing like ARP-Guard [19], snort [20], ARPDefender [21], and Arpwatch [22]. However, these methods can all suffer from false-positive detection [23] because they cannot distinguish between ARP spoofing attacks and connections from VM guests on a bridged network. In light of this, there is a clear need for an ARP spoofing detection algorithm that can effectively deal with connections from VM guests without returning unnecessary false alarms.

3. Adversary Model

In this section, we propose an adversary model for ARP spoofing attacks and the corresponding attack scenario. Then, we discuss the problem statement addressed in this article.

3.1. Adversary Model. In this model, we assume the presence of an attacker who can perform an ARP spoofing-based MITM attack. This adversary is assumed to be able to access the LAN, through which the attacker would be able to locate and connect to a target (i.e., a victim). Note that, due to the abundance of public places, such as cafeterias, where public AP services are provided, it is possible for an attacker to access the network where the attack target exists. Also, note that other types of spoofing attacks like DHCP or IP spoofing are out of the scope of this article. In addition, compromising attacks on the AP should be handled by other protection methods like remote attestation or control-flow integrity (CFI).

3.2. Attack Scenario. In Figure 1, a host (IP address: 192.168.1.20, MAC address: bb-bb-bb-bb-bb-bb) is assumed to have access to the AP (IP address: 192.168.1.1, MAC address: aa-aa-aa-aa-aa-aa). If an attacker (IP address: 192.168.1.30, MAC address: cc-cc-cc-cc-cc-cc) is on the same LAN, the attacker can perform an ARP spoofing-based MITM attack by modifying ARP packets between the AP and a user who has accessed the AP. As a result of the ARP spoofing-based MITM attack, the attacker's MAC address is assigned to both the AP and the attacker in the target's ARP cache tables.

To consider various attack scenarios, as shown in Table 2, we define five types of ARP spoofing attacks: Simple Attack 1, Simple Attack 2, Multi-Attack 1, Multi-Attack 2, and Complex Attack. While Simple Attacks 1 and 2 target only one victim, Multi-Attacks 1 and 2 are performed against two victims. Both the Simple Attacks 1 and 2 and Multi-Attacks 1 and 2 can be performed by a host OS or a VM guest OS. If both a host OS and a VM guest OS perform ARP spoofing attacks on a target at the same time, this case is defined as a Complex Attack.

3.3. Problem Statement. Recently, it has been customary for machines to utilize virtualization strategies. In a virtualization environment, there are a host OS and a guest OS. The former is the actual software on a computer that interacts with the primary hardware, and the latter is the software installed on a VM. If a guest OS is created on a bridged network, the guest OS can be assigned an IP address from an AP as if it was directly connected to the AP. When the guest OS and the AP communicate with each other, the guest OS transmits and receives packets using the host OS's MAC address. This is shown in Figure 2. Thus, the guest OS's MAC address displays as the host OS's MAC address in the ARP cache table of the AP even though no ARP spoofing attack has occurred. It seems problematic that the states of the ARP cache tables are identical when there is an ARP spoofing attack and when there is simply a guest OS on a bridged network, as shown in Figure 3. As such, there needs to be an algorithm that can distinguish between these two scenarios in order to effectively and accurately handle both ARP spoofing attacks and innocuous VM connections. In this article, we propose an IDS that can distinguish VM connections from ARP spoofing attacks and detect ARP spoofing attacks. A detailed description of our proposed system and its architecture follow in the next section.

4. Proposed System

In this section, we describe the system design of our proposed ARP Spoofing Detector, ASD, in detail. ASD is composed of three phases and uses three key pieces of information—*AssocList*, *ARP cache table*, and *DHCP table*—to distinguish between an ARP spoofing attack and a guest OS connection.

(i) AssocList: AssocList shows client information (e.g., MAC address and signal strength) connected to the AP. However, in the case of VM connections, the guest OS's MAC address is not shown. Figure 4 shows an example of AssocList as seen on OpenWrt.

(ii) ARP cache table: the ARP cache table shows client information (e.g., IP address and MAC address) managed by the ARP. However, in the case of VM connections, the guest OS's MAC address is shown as the host OS's MAC address.

(iii) DHCP table: the DHCP table shows client information (e.g., IP address and MAC address) that has been allocated by the DHCP server. Unlike the ARP cache table, the guest OS's MAC address is shown as its own actual MAC address in the case of VM connections.

The proposed ASD can be installed on any Linux-based AP system and can detect all ARP spoofing attempts that occur on the AP network. Figure 5 shows the system overview of ASD. Next, we explain the three phases of our protocol.

4.1. Phase 1: Comparison between AssocList and ARP Cache Table. In order to identify clients that are connecting to the AP, ASD generates a client list in Phase 1 by extracting all MAC addresses from the AssocList. Then, the extracted MAC addresses are compared against the MAC addresses stored in the ARP cache table. If all MAC addresses included in the client list appear in the ARP cache table once, it is regarded as a normal situation and ASD terminates. Otherwise, there is an abnormal situation in which duplicate MAC addresses exist in the ARP cache table due to an ARP spoofing attack or a VM connection status. In this case, ASD proceeds with Phase 2 to deal with the duplicate MAC addresses.

4.2. Phase 2: Comparison between ARP Cache Table and DHCP Table. If there is a client that has fallen victim to ARP spoofing or a guest OS via a VM connection, the actual MAC address used by the victim or the guest OS does not appear in the ARP cache table. However, in the DHCP table, the MAC address used by the victim or the guest OS is provided. Therefore, ASD can identify the MAC address used by the victim or the guest OS by comparing IP/MAC address pairs stored in the ARP cache table and the DHCP table. The detailed procedures of this phase are as follows.

First, the comparison target IP addresses mapped to the duplicate MAC addresses provided in Phase 1 are extracted from the ARP cache table. Second, the MAC addresses associated with the target IP addresses are extracted from the DHCP table. Finally, if a MAC address extracted from the DHCP table matches a duplicate MAC address provided in Phase 1, it is concluded that the node using the MAC address is either an ARP spoofing attacker or a host OS. Otherwise, the node itself may be a victim of an ARP spoofing attack, or it may be a guest OS.

After Phase 2, ASD proceeds with Phase 3 to determine its final decision.

4.3. Phase 3: Comparison between DHCP Table and AssocList. In Phase 3, ASD checks whether or not the MAC address extracted from the DHCP table is included in the AssocList. Remember that the MAC address used by a guest OS does not appear in AssocList. Therefore, if the MAC address of a node determined to be a victim of ARP spoofing or to be a guest OS in Phase 2 is found in the AssocList, ASD considers the node itself as a victim of ARP spoofing. Otherwise, the node is a guest OS on a bridged network. As such, ASD is able to distinguish between an ARP spoofing attacker and a host OS. If there is a node using the same MAC address as the victim of ARP spoofing in the ARP cache table, ASD

Table 2: ARP spoofing attack types.

Attack type	Attacker	Victim
Simple Attack 1	Host OS	One client
Simple Attack 2	VM guest OS	One client
Multi-Attack 1	Host OS	Two clients
Multi-Attack 2	VM guest OS	Two clients
Complex Attack	Host OS and VM guest OS	One client each

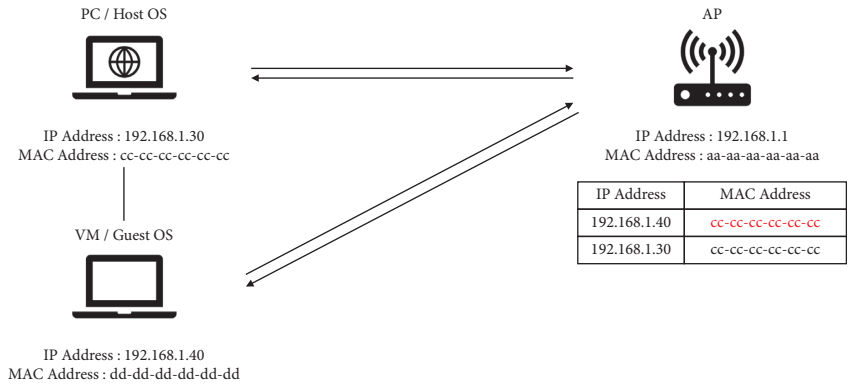

Figure 2: ARP cache table of an AP with VM connections.

ARP Spoofing Attack

	IP Address	MAC Address
Victim	192.168.1.20	cc-cc-cc-cc-cc-cc
Attacker	192.168.1.30	cc-cc-cc-cc-cc-cc

Virtual Machine

	IP Address	MAC Address
Guest	192.168.1.40	cc-cc-cc-cc-cc-cc
Host	192.168.1.30	cc-cc-cc-cc-cc-cc

Figure 3: Two cases of the ARP cache table of an AP.

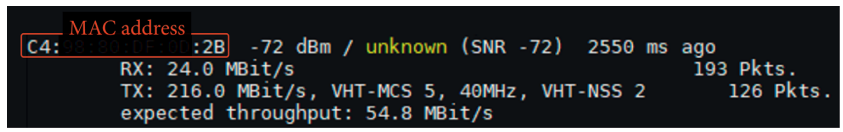

Figure 4: AssocList on OpenWrt.

determines the node to be an ARP spoofing attacker. Likewise, if there is a node using the same MAC address as the guest OS in the ARP cache table, ASD determines the node to be a host OS.

The detailed steps are described in Pseudo-algorithm 1.

5. Evaluation

In this section, we describe the monitoring system and the performance of the ASD in various attack scenarios. In addition, we evaluate the execution time of ASD and compare ASD with existing solutions.

5.1. Implementation. Figure 6 shows the experimental setup with Mi WiFi Mini, a monitoring tool, and Kali Linux for ARP spoofing attacks. We implemented the proposed system, ASD, on Xiaomi Mi WiFi Mini using OpenWrt firmware. OpenWrt is an embedded OS based on Linux and is primarily used on embedded devices to route network traffic. Then, we simulated multiple ARP spoofing attacks and guest OS connections to evaluate whether ASD is able to distinguish ARP attacks from VM guest connections on a bridged network. To simulate ARP spoofing attacks, we used Ettercap provided by Kali Linux. Ettercap is an open-source network security tool for packet sniffing and staging MITM attacks on a LAN. In the ARP spoofing-based MITM test, we firstly scanned for hosts. The AP and victim were selected, and spoofed ARP messages were generated. As shown in Figure 7, when an ARP spoofing attack occurs, ASD shows the victim, ARP spoofing attacker, and VM connections. The victim is represented in red and the attacker and VM connections are represented in blue. Note that we released the source code for ASD as open-source on GitHub [24].

5.2. Performance Evaluation. To evaluate the robustness of ASD, we performed experiments with the five types of ARP spoofing attacks identified in Section 3.2. As shown in Table 3, ASD can detect all types of ARP spoofing attacks with a 100% success rate and returns no false alarms.

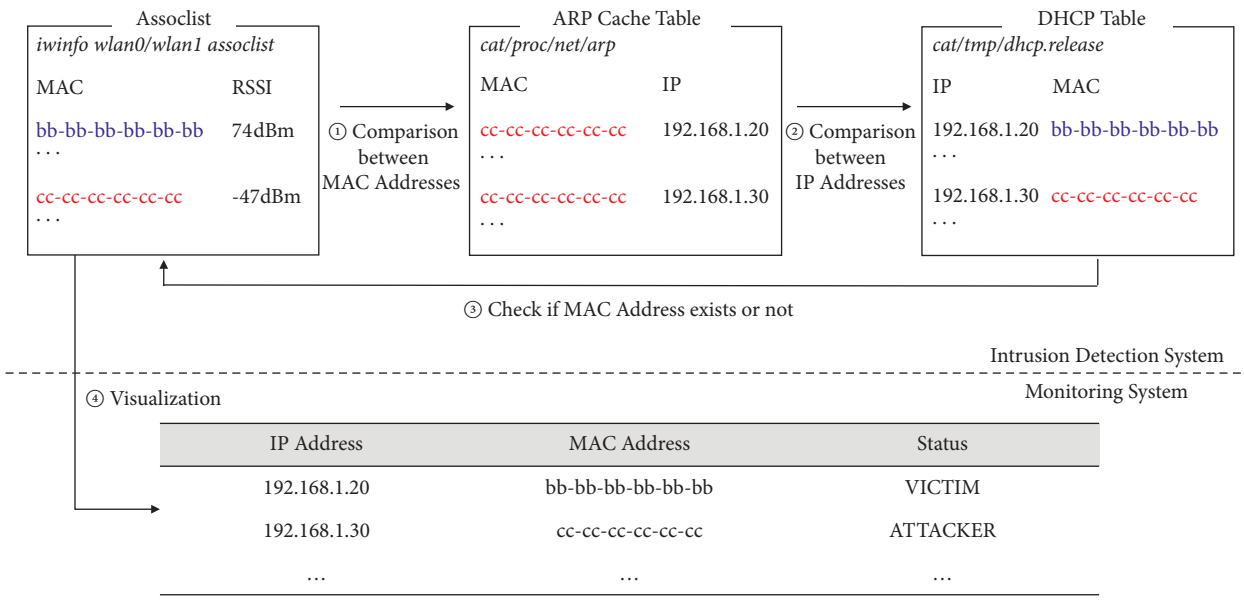

Figure 5: System design of the ARP Spoofing Detector (ASD).

Output: info including IP address, MAC address, and Status
(1) info ⟵ Create an empty list;
(2) client_list⟵ Extract all MAC addresses in Assoclist;
(3) arp_table ⟵ Extract all IP addresses and MAC addresses in ARP Cache Table;
(4) **for** mac ***in*** client_list **do**
(5) arp_ip ⟵ IP addresses that match MAC addresses as mac in arp_table;
(6) **for** ip ***in*** arp_ip **do**
//**Normal** Status
(7) **if** *count of* arp_ip < 2 **then**
(8) Append \{ ip, mac, Normal\} in info;
(9) **else**
(10) dhcp_mac ⟵ MAC addresses that match IP addresses as ip in DHCP Table;
(11) **if** dhcp_mac ***in*** client_list **then**
//Attacker or Host OS Status
(12) **if** mac = dhcp_mac **then**
(13) Append \{ip, mac, Host OS\} in info;
//Victim Status
(14) **else**
(15) Append\{ ip, mac, Victim\} in info;
(16) **end**
//Guest OS Status
(17) **else**
(18) Append\{ ip, mac, Guest OS\} in info;
(19) **end**
(20) end
(21) **end**
(22) **end**
(23) **return** info

Algorithm 1: ASD algorithm.

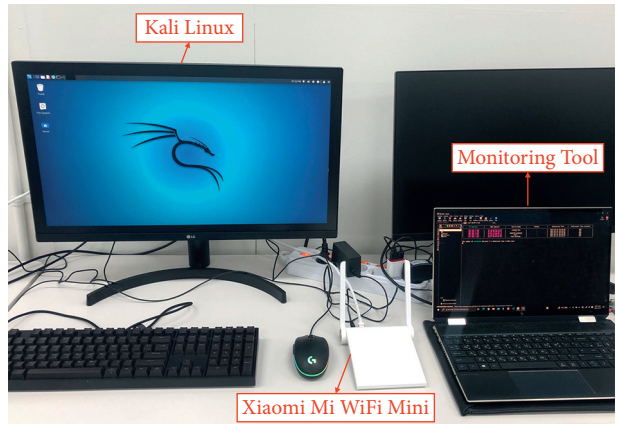

Figure 6: Experimental setup.

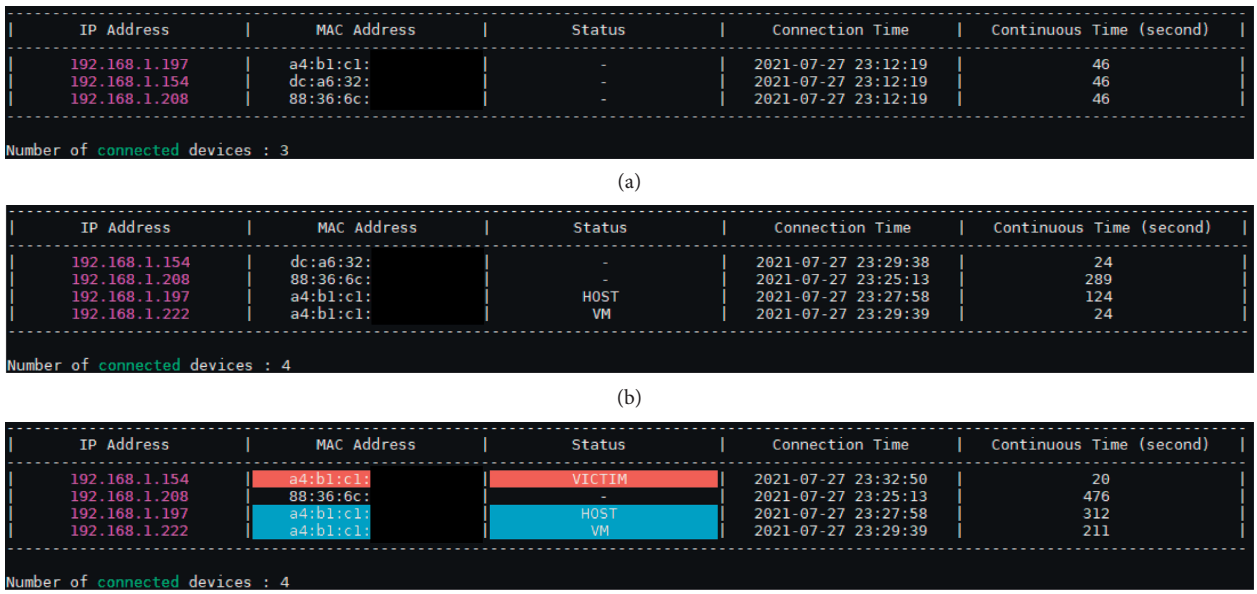

Figure 7: ASD monitoring system. (a) Normal status. (b) VM connection status. (c) ARP spoofing attack status.

In addition, we measured the detection time of the ARP spoofing attacks to evaluate the performance of ASD. As shown in Figure 8, we confirmed that detection time depends on the number of devices. When there is one device connected to the AP and there is no ARP spoofing attack, ASD Phase 1 execution time was about 200 ms. When there were 10 devices, however, the Phase 1 execution time was about 1 second. If there is an ARP spoofing attack, ASD takes more time checking the ARP spoofing attack and VM connection than if there was no abnormality detected because it needs to undergo both Phases 2 and 3. The time difference between the normal case and the ARP spoofing attack case is around 60 milliseconds, and we found that ASD effectively detects the ARP spoofing attack without false alarms in a short amount of time.

5.3. Comparison of Existing Solutions. In this section, we compare ASD with existing methods, which are DHCP-based solutions [16–18], as shown in Table 4. Even though three existing solutions do not require any protocol modification or additional equipment, they can return detection errors in regard to ARP spoofing attacks because it is impossible to distinguish between ARP spoofing attack attempts and normal VM connection attempts just by observing the DHCP table. In addition, these protocols do not provide a visualization function like a monitoring tool to further explain details about the abnormal status. On the other hand, ASD does not generate false alarms as it effectively distinguishes between ARP spoofing attacks and VM connections, and it additionally provides visualization of detection results through a monitoring tool.

6. Conclusion

In this article, we propose the ARP Spoofing Detector (ASD) that can detect ARP spoofing attacks using OpenWrt. ASD is notable because it can detect ARP spoofing without

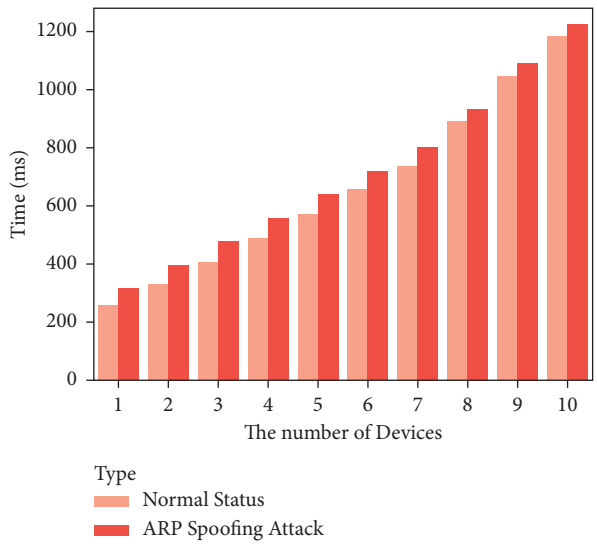

Figure 8: ASD detection time by the number of devices.

Table 3: Experimental results of ARP spoofing attacks.

Attack type	ASD detection result
Simple Attack 1	√
Simple Attack 2	√
Multi-Attack 1	√
Multi-Attack 2	√
Complex Attack	√

Table 4: Comparison with existing solutions.

	Additional equipment	Visualization	Detection error on VM connections
Masoud et al. [16]	No	No	Yes
Cox et al. [17]	No	No	Yes
Sun et al. [18]	No	No	Yes
ASD	No	Yes	No

additional equipment or protocol modification. In addition, ASD does not generate false alarms caused by VM connections on a bridged network, and ASD provides a monitoring tool to help visual analysis of any ARP attack attempts. The proposed system was evaluated through ARP spoofing attack experiments, and as a result, we confirmed that it detects ARP spoofing attacks without returning false alarms.

Data Availability

The data will be available at https://github.com/Yeon-Seon/SpoofingIDS.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (no. NRF-2021R1A4A1029650).

References

[1] F. Fayyaz and H. Rasheed, "Using jpcap to prevent man-in-the-middle attacks in a local area network environment," *IEEE potentials*, vol. 31, no. 4, pp. 35–37, 2012.
[2] Openwrt: , [Online], Available https://openwrt.org/.
[3] D. C. Plummer, "Rfc 826: an ethernet address resolution protocol," *InterNet Network Working Group*, 1982.
[4] A. Lockhart, *Network Security Hacks: Tips & Tools for Protecting Your Privacy*, O'Reilly Media, Inc., California, CL, USA, 2006.
[5] M. D. Prieto, J. P. Blázquez, J. H. Joancomartí, and J. A. Moreno, "Towards secure mobile p2p applications using jxme," *Journal of Internet Services and Information Security (JISIS)*, vol. 2(1.2), pp. 1–21, 2012.
[6] D. Caputo, L. Verderame, A. Ranieri, A. Merlo, and L. Caviglione, "Fine-hearing google home: why silence will not protect your privacy," *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*vol. 11, no. 1, pp. 35–53, 2020.
[7] Y. Bhaiji, *Security Features on Switches*, O'Reilly Media, Inc., California, CL, USA, 2009.
[8] D. Bruschi, O. Alberto, and E. Rosti, "S-arp: a secure address resolution protocol," in *Proceedings of the 19th Annual Computer Security Applications Conference*, pp. 66–74, IEEE, Las Vegas, NV, USA, December 2003.
[9] W. Lootah, W. Enck, and P. McDaniel, "Tarp: ticket-based address resolution protocol," *Computer Networks*, vol. 51, no. 15, pp. 4322–4337, 2007.
[10] V. Goyal and R. Tripathy, "An efficient solution to the arp cache poisoning problem," in *Australasian Conference on Information Security and Privacy*, pp. 40–51, Springer, Berlin, Germany, 2005.
[11] S. Nam, D. Kim, and J. Kim, "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] M. G. Gouda and C.-T. Huang, "A secure address resolution protocol," *Computer Networks*, vol. 41, no. 1, pp. 57–71, 2003.
[14] T. Demuth and A. Leitner, "Arp spoofing and poisoning: traffic tricks," *Linux magazine*, vol. 56, pp. 26–31, 2005.
[15] A. P. Ortega, E. M. Xavier, D. C. Luis, and L. A. Cristina, "Preventing arp cache poisoning attacks: a proof of concept using openwrt," in *Proceedings of the Latin American Network Operations and Management Symposium*, pp. 1–9, IEEE, Punta del Este, Uruguay, October 2009.
[16] M. Z. Masoud, Y. Jaradat, and I. Jannoud, "On preventing arp poisoning attack utilizing software defined network (sdn) paradigm," in *Proceedings of the IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, pp. 1–5, IEEE, Amman, Jordan, November 2015.
[17] J. H. Cox, J. C. Russell, and L. O. Henry, "Leveraging sdn for arp security," in *Proceedings of the SoutheastCon*, pp. 1–8, IEEE, Norfolk, VA, USA, March 2016.
[18] S. Sun, X. Fu, B. Luo, and X. Du, "Detecting and mitigating arp attacks in sdn-based cloud environment," in *Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer*

Communications Workshops (INFOCOM WKSHPS), pp. 659–664, IEEE, Toronto, ON, Canada, July 2020.

[19] I. I. S. GmbH, "Arp-guard," 2022, https://www.arpguard.com/en/info/product/arp-guard_en.html.

[20] X. Hou, Z. Jiang, and X. Tian, "The detection and prevention for arp spoofing based on snort," in *Proceedings of the International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 5, October 2010.

[21] I. ARPDefender, "What is arpdefender designed for?" 2010, http://www.arpdefender.com/home.

[22] Lnr Group, "arpwatch, the ethernet monitor program; for keeping track of ethernet/ip address pairings," vol. 17, 2012.

[23] J. Belenguer and C. T. Calafate, "A low-cost embedded ids to monitor and prevent man-in-the-middle attacks on wired lan environments," in *Proceedings of the The International Conference on Emerging Security Information, Systems, and Technologies (SECUREWARE 2007)*, pp. 122–127, IEEE, Valencia, Spain, October 2007.

[24] Github: 2022, https://github.com/Yeon-Seon/SpoofingIDS.