Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 12(3): 154-160, Sep. 2014 **Regular paper**

ARP Modification for Prevention of IP Spoofing

Jung-Ha Kang[1], Yang Sun Lee[2], Jae Young Kim[3], and Eun-Gi Kim[1*], *Member, KIICE*

[1]Department of Information and Communication Engineering, Hanbat National University, Daejeon 305-719, Korea
[2]Division of Computer Engineering, Mokwon University, Daejeon 302-729, Korea
[3]IT Convergence Technology Research Lab., Electronics and Telecommunications Research Institute, Daejeon 305-700, Korea

Abstract

The address resolution protocol (ARP) provides dynamic mapping between two different forms of addresses: the 32-bit Internet protocol (IP) address of the network layer and the 48-bit medium access control (MAC) address of the data link layer. A host computer finds the MAC address of the default gateway or the other hosts on the same subnet by using ARP and can then send IP packets. However, ARP can be used for network attacks, which are one of the most prevalent types of network attacks today. In this study, a new ARP algorithm that can prevent IP spoofing attacks is proposed. The proposed ARP algorithm is a broadcast ARP reply and an ARP notification. The broadcast ARP reply was used for checking whether the ARP information was forged. The broadcast ARP notification was used for preventing a normal host's ARP table from being poisoned. The proposed algorithm is backward compatible with the current ARP protocol and dynamically prevents any ARP spoofing attacks. In this study, the proposed ARP algorithm was implemented on the Linux operating system; here, we present the test results with respect to the prevention of ARP spoofing attacks.

Index Terms: ARP cache, ARP poison, ARP spoofing, MITM, Network attack, Spoofing detection

I. INTRODUCTION

Although the Internet protocol (IP) has become widely successful, it has led to many security issues. Many of these security issues are related to illegal host access. An address resolution protocol (ARP) spoofing attack is another security issue related to an illegal host access. This paper deals with the prevention of these ARP spoofing attacks.

ARP provides dynamic mapping between two different forms of addresses: the 32-bit IP address of the network layer and the 48-bit medium access control (MAC) address of the data link layer [1]. A host computer finds the MAC address of a default gateway or of the other hosts on the same subnet by using ARP, after which it can send data packets [2]. However, recently, there have been a considerable number of network attacks using ARP. The types of these attacks can vary from attacks interfering with the network operations of the host to spoofing attacks that allow the attacker to intercept data frames. Man in the middle (MITM) attacks are a form of potential spoofing attacks. An MITM attack means that the attacker intercepts the data frames of the target host, modifies them maliciously, and then forwards these modified frames [3]. A number of solutions have been proposed to prevent ARP spoofing attacks [4]. However, the existing solutions require additional systems or are not perfectly compatible with the current ARP.

In this paper, a modified ARP algorithm to prevent spoofing attacks is proposed. The proposed algorithm is backward compatible with the current ARP and can

Received 17 April 2014, **Revised** 02 May 2014, **Accepted** 09 July 2014
***Corresponding Author** Eun-Gi Kim (E-mail:egkim@hanbat.ac.kr, Tel: +82-42-821-1215)
Department of Information and Communication Engineering, Hanbat National University, 125 Dongseo-daero, Yuseong-gu, Daejeon 305-719, Korea.

Open Access **http://dx.doi.org/10.6109/jicce.2014.12.3.154** print ISSN: 2234-8255 online ISSN: 2234-8883

effectively prevent ARP spoofing attacks.

The rest of this paper is organized as follows: Section II describes the ARP operation and ARP spoofing attacks. In Section III, the proposed ARP protocol is described. In Sections IV and V, implementation of the proposed algorithm and test results for its verification are discussed. The conclusion is presented in Section VI.

II. ARP SPOOFING ATTACKS

A. Address Resolution Protocol

An Ethernet ARP frame format is illustrated in Fig. 1(a). The Ethernet header includes the Ethernet destination address, Ethernet source address, and frame type. The Ethernet address, also known as the MAC address, is a 48-bit address. In the case of an ARP frame, the value of the frame type field is '0x0806.' The ARP header includes the type of hardware address (its value for Ethernet is 1), the type of protocol address (its value for an IP address is 0x0800), the size of the hardware address (6 bytes for Ethernet), the size of the protocol address (4 bytes for an IP address), and the operation field (1 for an ARP request and 2 for an ARP reply). The payload fields of an ARP frame include the sender's Ethernet address, sender's IP address, target's Ethernet address, and target's IP address. With the ARP request frames, all the fields are filled in except the target Ethernet address. The target host receiving the ARP request fills in the target Ethernet address field with its Ethernet address, swaps the two sender addresses with the two target addresses, sets the operation code (OP) field to 2, and sends the ARP reply [2, 5].

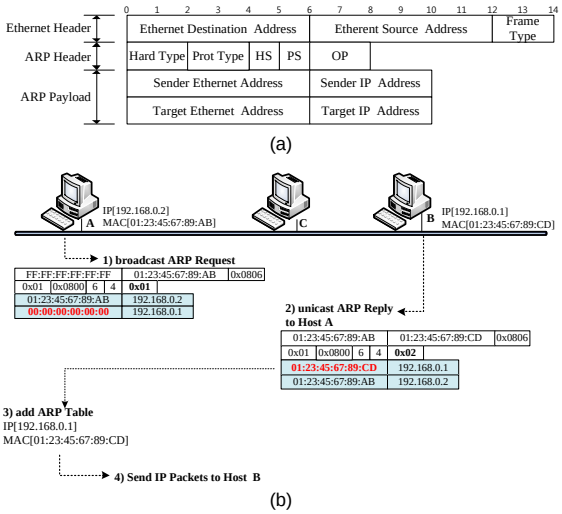


Fig. 1. Address resolution protocol (ARP) frame format and operation. (a) ARP frame format, (b) normal ARP operation.

Fig. 1(b) shows a normal ARP operation. First, host A broadcasts an ARP request to find out host B's MAC address for the destination IP [192.168.0.1]. Hosts B and C receive host A's ARP request, but host C drops the request because the target IP address is not the same as its IP address. Secondly, host B confirms that the target IP address of the ARP request is the same as its own IP address and sends out a unicast ARP reply to host A. Thirdly, host A, having received the ARP reply from host B, saves both the MAC address and the IP address of host B in its ARP table. Finally, host A sends IP packets to host B referencing the ARP entry that includes host B's MAC address and IP address.

B. ARP Spoofing Attacks

During ARP spoofing attacks, an attacker periodically sends out a forged unicast (or broadcast) ARP request and reply. The victim receiving the forged ARP request or reply then adds the poisoned ARP entry to its ARP table. The victim sends IP packets to the attacker referring to the poisoned ARP entry rather than the normal target host. ARP request attacks and ARP reply attacks are described in the following paragraphs separately [6]. ARP attacks are illustrated in Fig. 2.

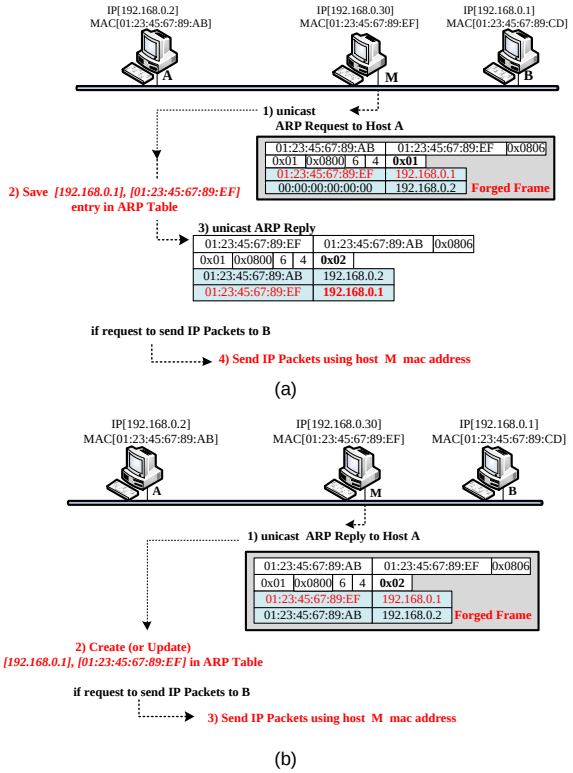


Fig. 2. Address resolution protocol (ARP) spoofing attacks. (a) Forged ARP request attack, (b) forged ARP reply attack.

1) Forged ARP Request Attacks

During a normal ARP operation, shown in Fig. 1(b), host B saves host A's MAC address and IP address at the time of receipt of the ARP request from host A. Thus, the attacker sends out a forged ARP request by unicast or broadcast to the normal host, and thus, the normal host makes a poisoned ARP entry of only the forged ARP request. An ARP request attack is illustrated in Fig. 2(a). Hosts A and B are legitimate hosts, and host M is the malicious host. First, host M sends out a unicast ARP request forging host B's IP address to host A. The forged ARP request includes host B's IP address in the sender's IP address field and host M's MAC address in the sender's MAC address field. Secondly, host A stores the sender's IP address and the MAC address of the forged APP request in its ARP table. The stored ARP entry is [192.168.0.1] (host B's IP address) on "01:23:45:67:89:EF" (host M's MAC Address). Thirdly, host A sends out an ARP reply to host M. Finally, the MAC address for the IP address [192.168.0.1] is the poisoned ARP entry in host A's ARP table. If host A sends out IP packets with a destination IP address [192.168.0.1], these packets are sent out to host M, not host B.

2) Forged ARP Reply Attacks

The attacker sends out a forged unicast ARP reply to a specific host, and then, the host saves the sender's MAC address and the IP address of the forged ARP reply in its ARP table irrespective of whether a reply was solicited. Therefore, the normal host may refer to the poisoned entry. Fig. 2(b) shows an ARP reply attack. Firstly, host M sends out a unicast ARP reply forging host B's IP address to host A. Secondly, host A creates or updates a forged ARP entry from the malicious ARP reply of host M. Thirdly, host A refers to the poisoned ARP entry and sends out the frame; then, all frames that must be sent to host B are delivered to host M. Hosts mostly communicate with servers or hosts on other subnets. Thus, an attacker generally forges a gateway's IP address for an ARP spoofing attack [7].

III. PROPOSED ADDRESS RESOLUTION PROTOCOL

In this section, a modified ARP algorithm for the prevention of ARP spoofing attacks, described in Section II, is proposed. There are two main differences between the proposed ARP algorithm and the original ARP algorithm. The first difference is that the proposed ARP algorithm sends out broadcast ARP replies. This is a procedure wherein other hosts verify whether the mapping of the MAC address and IP address in the ARP reply is forged or not. The second difference is the new ARP message type, defined as an ARP notification.

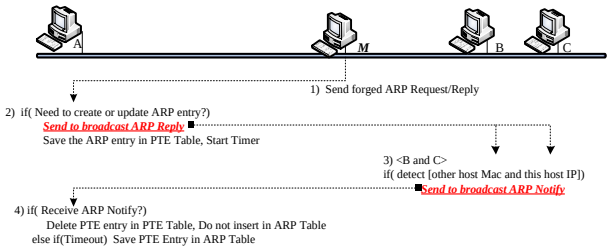


Fig. 3. Proposed address resolution protocol (ARP) algorithm.

The function of the ARP notification is to notify other hosts of the forged address. In other words, normal hosts can receive all ARP replies and thereby detect an ARP spoofing attack, and the host detecting the attack broadcasts the ARP notification. Hosts receiving an ARP notification do not save the relevant address in the ARP table.

Fig. 3 shows an overview of the ARP algorithm proposed in the study. Firstly, host M, the attacker, transmits the forged ARP request (or ARP reply), usually by unicast. Secondly, host A, the victim, checks whether its IP address is the same as the target IP address of the ARP request frame from host M. If it is the same, host A adds the sender's MAC address field and IP address field to its processing table entry (PTE) table. Then, host A broadcasts an ARP reply in order to verify the sender's addresses and starts its PTE timer. Thereafter, hosts B and C receive the broadcast ARP reply from host A. Hosts B and C can detect the presence of an ARP spoofing attack if host B(or host C)'s IP address is the same as the target IP address but its MAC address is different from the target MAC address. Thus, hosts B and C broadcast an ARP notification in the event of attack detection. Finally, host A receives the broadcast ARP notification from host B or host C prior to timeout and then, deletes the relevant entry from its PTE table. If no ARP notification arrives prior to the timeout, host A saves the PTE in its ARP table.

The frame format of an ARP notification is the same as that illustrated in Fig. 1(a), and the operation value is defined as 0x0B. The MAC address of the suspected attacking hosts is entered into the target MAC address field of the ARP notification. The IP address of the victim is

Table 1. Modified address resolution protocol (ARP) state transition table

Symbol	Idle
User Request	*1
ARP Request (M[S], IP[S], M[T], IP[T])	*2
ARP Reply (M[S], IP[S], M[T], IP[T])	*3
ARP Notification (M[S], IP[S], M[T], IP[T])	*4
Timeout	*5

entered into the target IP address field of the ARP notification. The host receiving the ARP notification does not create or update the forged address in its ARP Table. Consequently, the function of the ARP notification is to prevent ARP spoofing attacks. Table 1 shows the state transition table of the proposed ARP algorithm.

*1:
i. Send ARP Request(M[This Host], IP[This Host], 0, IP[Requested Host]) to Ether(Bca, M[This Host]);
ii. Requested.IP[Requested Host] = TRUE;

*2:
i. If(IP[S] == IP[This Host])
- Send ARP Notification(M[This Host], IP[This Host], M[S], IP[S]) to Ether(Bca, M[This Host]);

ii. Else if(IP[T] == IP[This Host])
- Send Reply(M[This Host], IP[This Host], M[S], IP[S]) to Ether(Bca, M[This Host]);
- Save M[S], IP[S] into PTE;
- Start Timer;
- PTE for "M[S], IP[S]".next state = Wait;

*3:
i. If(Requested.IP[IP[T]] == TRUE)
/* Normal Reply by solicitation */
- Save M[S], IP[S] into PTE;
- Start Timer;
- PTE for "M[S], IP[S]".next state = Wait;
- Requested.IP[IP[T]] == FALSE;

ii. Else if(IP[T] == IP[This Host])
/* Unsolicited ARP reply */
- Send Received Reply frame to Ether(bca, M[This Host]);
- Save M[S], IP[S] into PTE;
- Start Timer;
- PTE for "M[S], IP[S]".next state = wait;

iii. Else if(IP[S] == IP[This Host])
- Send ARP Notification(M[This Host], IP[This Host], M[T], IP[T]) to Ether(Bca, M[This Host]);

*4:
If(PTE for "M [S], IP[S]".next state == Wait)
- Purge the entry in PTE;

*5:
If(PTE for "MS[S], IP[S]".next state == Wait)
- Save M[S], IP[S] into ARP Table;
- Purge the entry in PTE;

IV. IMPLEMENTATION

The ARP source code is generally included in the kernel of an operating system. The ARP source code included in the kernel must be modified in order to implement the proposed algorithm. Thus, an ARP source in Linux Kernel was used for the implementation of the proposed ARP algorithm; this was done because Linux is an open-source operating system [8-10]. Modification of the ARP code can lead to frequent replacement of the kernel image. Thus, embedded systems are used for development convenience [11]. Embedded systems based on the ARM 6410 CPU were used. The kernel that was ported to the ARM 6410 board was Linux Kernel version 2.6.21. In this section, the original ARP source code is described, and then, the modified code to implement the proposed algorithm is described in detail.

A. Original ARP Instance in the Linux Kernel

"arp_tbl" as the ARP table is declared by "struct neigh_table" in the ARP instance of Linux kernel 2.6.21, and the functions called for ARP processing are as shown in Fig. 4 [8, 9].

As shown in Fig. 4, the ARP frames are processed by the function arp_process(). The proposed algorithm to be implemented is mostly related to this function.

B. Implementation Details

In this study, the ARP notification frame's OP code was defined as ARPOP_NOTIFY (11), and the modified arp_process() function was mainly used for implementing the proposed algorithm. Management functions, timer, and structure for the PTE table were also implemented.

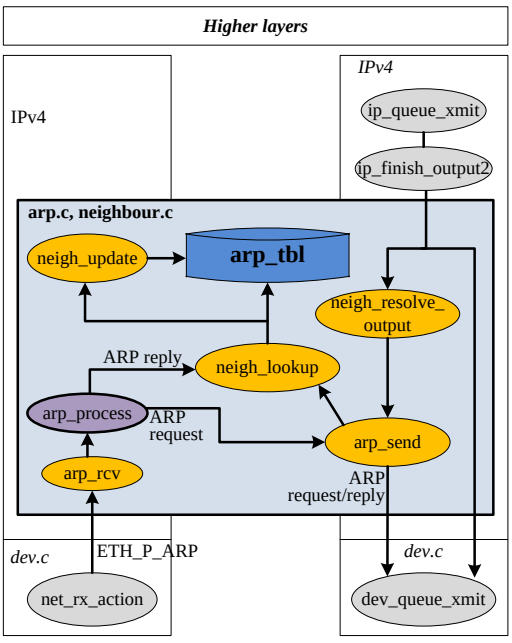


Fig. 4. Function flow related to address resolution protocol (ARP) processing.

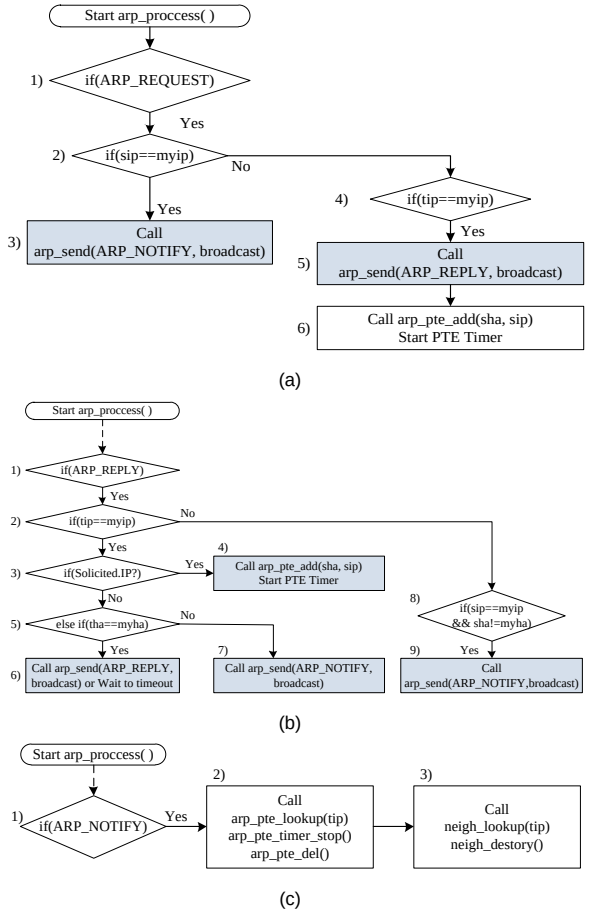


Fig. 5. Function flow related to address resolution protocol (ARP) processing. (a) Modification of ARP request message processing, (b) modification of ARP reply message processing, and (c) modification of ARP notification message.

<Notations>

- sip: sender's IP address
- sha: sender's MAC address
- tip: target's IP address
- tha: target's MAC address
- myip: this host's IP address
- myha: this host's MAC address

The modified arp_process() function will receive ARP_REQUEST frame, ARP_REPLY frame or ARP_NOTIFY frame and the processing flow of each frame is illustrated in Fig. 5.

1) ARP Request Frame Processing Shown in Fig. 5(a)

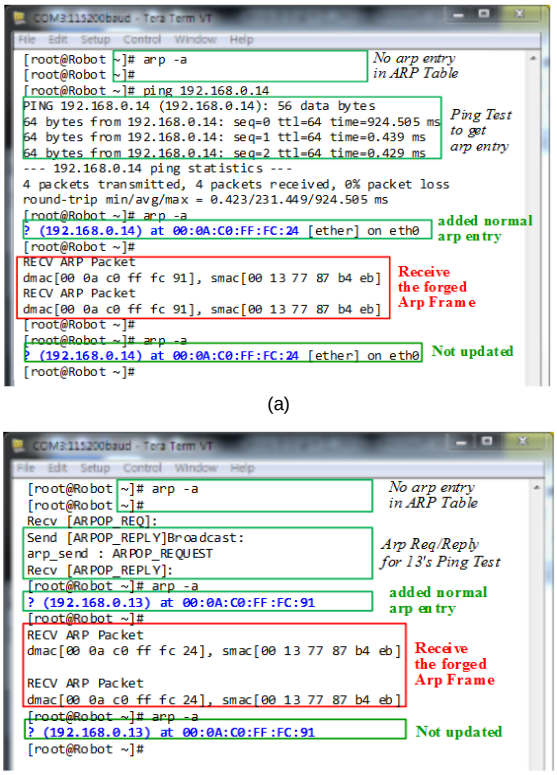
- 2) to 3): If sip is the same as myip (the host's IP address), the host broadcasts an ARP notification frame due to an attack.
- 4) to 6): If tip is the same as myip, the host broadcasts an ARP reply. Then, the host adds sha and sip to the PTE entry and starts the PTE timer.

2) ARP Reply Frame Processing Shown in Fig. 5(b)

- In cases 2), 3), and 4), suppose that (tip == myip) and Solicited.IP: The ARP reply is normally the solicited frame. Thus, an ARP reply including sha and sip is added to the PTE table, and the PTE timer is started and counts down until timeout.
- In cases 2), 5), and 6), suppose that (tip == myip) and (tha == myha): This is an unsolicited ARP reply. Thus, the ARP reply is retransmitted by broadcast for its verification from other hosts. Then, sha and sip are added to the PTE table, and the PTE timer is started.
- In cases 2), 5), and 7), suppose that (tip == myip) and (tha != myha): This is an ARP reply attack using myip (this host's IP address), and the host broadcasts an ARP notification.
- In cases 8) and 9), suppose that (sip == myip) and (sha != myha): This is an ARP reply attack and the host broadcasts an ARP notification.

3) ARP Notification Processing Shown in Fig. 5(c)

- When it receives an ARP notification, stop the PTE timer and delete the PTE including tip.
- Further, delete the ARP entry related to the tip and tha of the ARP notification.



(a)

(b)

Fig. 6. The Victim's Console Log. (a) Host A console log, (b) host B console log.

V. TESTS AND RESULTS

Hosts A and B as victims are embedded boards loaded with the Linux OS. Host M as an attacker is a personal computer, and the Cain & Abel program, an ARP spoofing attack tool, operates at host M. Further, all ARP frames on the subnet were captured using the Wireshark tool operating at Host M.

As shown in Fig. 6, a Ping test to make the ARP entry was attempted first. After the creation of the ARP entry, ARP spoofing attacks were attempted. The results confirmed that the normal ARP entry remained unchanged after the attacks.

ARP notification transmissions were verified as shown in Table 2.

Table 2. Sniffing data caused by the proposed ARP for the solicited ARP entry

No.	Time	Src	Dest	Info	Comment
→45	34.1362	~:91	Broadcast	Who has 192.168.0.14? Tell 192.168.0.13	Normal Request for Ping
→46	34.1694	~:24	Broadcast	192.168.0.14 is at 00:0a:c0:ff:fc:24	Normal Reply
→47	34.5987	~:24	Broadcast	Who has 192.168.0.13? Tell 192.168.0.14	Normal Request
→48	34.6345	~:91	Broadcast	192.168.0.13 is at 00:0a:c0:ff:fc:91	Normal Reply
→52	40.0915	~:91	Broadcast	192.168.0.13 is at 00:0a:c0:ff:fc:91	Normal Reply
→53	40.1760	~:24	Broadcast	192.168.0.14 is at 00:0a:c0:ff:fc:24	Normal Reply
→80	66.3661	~:eb	~:91	Who has 192.168.0.13? Tell 192.168.0.14	Request Attack
→81	66.3664	~:eb	~:24	Who has 192.168.0.14? Tell 192.168.0.13	Request Attack
→82	66.3668	~:eb	~:91	192.168.0.14 is at 00:13:77:87:b4:eb	Reply Attack
→83	66.3673	~:eb	~:24	192.168.0.13 is at 00:13:77:87:b4:eb	Reply Attack
→84	66.4091	~:91	Broadcast	192.168.0.13 is at 00:0a:c0:ff:fc:91	13's Reply
→85	66.4163	~:24	Broadcast	192.168.0.14 is at 00:0a:c0:ff:fc:24	14's Reply
→86	66.4526	~:91	Broadcast	192.168.0.13 is at 00:0a:c0:ff:fc:91	Broadcast unconf irmArp Reply
→87	66.4609	~:24	Broadcast	192.168.0.14 is at 00:0a:c0:ff:fc:24	Broadcast unconf irmArp Reply
→88	66.4978	~:91	Broadcast	Unknown ARP opcode 0x000b	13's ARP notification
→89	66.5064	~:24	Broadcast	Unknown ARP opcode 0x000b	14's ARP notification
→90	66.5432	~:91	Broadcast	Unknown ARP opcode 0x000b	13's ARP notification
→91	66.5523	~:24	Broadcast	Unknown ARP opcode 0x000b	14's ARP notification

ARP: address resolution protocol.

VI. CONCLUSIONS

ARP attacks can be prevented by using packet filtering, port security settings, or separate device addition methods [4]. These solutions have several issues, including economic inefficiency, use of system resources, and compatibility aspects. In this study, an algorithm that can dynamically prevent spoofing attacks by the modification of ARP was proposed.

We focused on broadcasting an ARP reply and an ARP notification. The broadcast ARP reply was used for checking whether the ARP information was forged. The broadcast ARP notification was used for preventing a normal host's ARP table from being poisoned. Further, the proposed algorithm was implemented in Linux Kernel and tested in a real LAN environment. The results showed that the designed algorithm can prevent ARP spoofing attacks. In the same ARP spoofing attacks, the original ARP protocol's ARP table was poisoned, but the proposed ARP's table was not poisoned. In this mechanism, an ARP notification frame might be maliciously used for isolating a specific host from a network. However, an ARP notification on the network means that any existing types of ARP attacks can be detected. Furthermore, this algorithm can prevent MITM attacks in which an attacker intercepts and forges a victim's frames.

Further studies focusing on solutions for a malicious use of ARP notification are needed, and a study on the Internet Engineering Task Force (IETF)'s Source Address Validation Improvements (SAVI) protocols is planned.

ACKNOWLEDGMENTS

This research was financially supported by the Ministry of Education (MOE) and National Research Foundation of Korea (NRF) through the Human Resource Training Project for Regional Innovation (No. 2013H1B8A2032154) and a grant (12-TI-C01) from Advanced Water Management Research Program funded by Ministry of Land, Infrastructure and Transport of Korea.

REFERENCES

[1] D. C. Plummer, "An Ethernet address resolution protocol," *RFC 826*, 1982.

[2] W. R. Stevens, *TCP/IP Illustrated (Volume 1. The Protocols)*. Reading, MA: Addison-Wesley, 1994.

[3] R. Braden, "Requirements for Internet hosts: communication layers," *RFC 1122*, 1989.

[4] W. R. Stevens and G. R. Wright, *TCP/IP Illustrated (Volume21. The Implementation)*. Reading, MA: Addison-Wesley, 1994.

[5] Linux source code [Internet], Available: http://www.kernel.org/pub/linux/kernel/v2.6.

[6] S. G. Bhirud and V. Katkar, "Light weight approach for IP-ARP spoofing detection and prevention," in *Proceedings of the 2nd Asian Himalayas International Conference on Internet (AH-ICI)*, Kathmandu, Nepal, pp. 1-5, 2011.

[7] W. Xing, Y. Zhao, and T. Li, "Research on the defense against ARP spoofing attacks based on WinPcaP," in *Proceedings of the 2nd International Workshop on Education Technology and Computer Science (ETCS)*, Wuhan, China, pp. 762-765, 2010.

[8] K. Wehrle, F. Pahlke, H. Ritter, D. Muller, and M. Bechler, *The Linux Networking Architecture: Design and Implementation of Network Protocols in the Linux Kernel*. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[9] J. Corbet, *Linux Device Drivers*, 3rd ed. Beijing: O'Reilly, 2005.

[10] A. Bremler-Barr and H. Levy, "Spoofing prevention method," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, Miami, FL, pp. 536-547, 2005.

[11] K. Yaghmour, *Building Embedded Linux Systems*, 2nd ed. Sebastopol, CA: O'Reilly, 2008.

Jung-Ha Kang

Dr. Kang received her B.S., M.S., and Ph.D. in Information and Communication Engineering from Hanbat National University, Republic of Korea, in 1997, 2001, and 2014, respectively. She was Principal Research Engineer at R&D Center, Fumate Co., Ltd. from 2002 to 2012. Her main research interests include computer networks, wireless communication, cryptography, and network security.

Yang Sun Lee

Prof. Lee received his B.S. and M.S. in Electrical & Electronic Engineering from Dongshin University and his PhD degree from Dept. of IT Engineering, Mokwon University, in 2001, 2003, and 2007, respectively. He also received his second PhD degree from Graduate School of Engineering, Fukuoka Institute Technology (FIT), Japan, in 2012. He was Senior Engineer at R&D Center, Fumate Co., Ltd. from 2007 to 2009. Further, he was Research Professor at Dept. of Information Communication Engineering, Chosun University from 2009 to 2011. Since 2012, he has worked as Assistant Professor in Division of Computer Engineering, Mokwon University, Korea. He also serves as a guest editor and is a member of the editorial staff and review committee of the Information Journal, Security and Communication Networks (Wiley InterScience), Wireless Personal Communications (Springer), International Journal of Communication Systems (Wiley InterScience), IET Signal Processing (IET Journal), IET Communications (IET Journal), and many other journals. His current research interests include UWB, wireless multimedia communication, software engineering, network transmission scheme, and ubiquitous sensor networks. He is a member of the KITCS, KICS, KIICE, KIIT, and KONI, and a fellow member of FTRA.

Jae Young Kim

Dr. Kim received his B.S. degree from the School of Electronic Engineering, Yonsei University, Korea, in 1990, and his M.S. and Ph.D. in Electronic Engineering from Yonsei University, Seoul, Korea, in 1992 and 1996, respectively. From 1996 to 1999, he worked as a communication systems engineer at Daewoo Electronics Ltd. He joined the Electronics and Telecommunications Research Institute (ETRI) in 1999. His main research interests include wireless sensor networks and security.

Eun-Gi Kim

Dr. Kim received his B.S., M.S., and Ph.D. degrees from Department of Electronic Engineering, Korea University, in 1987, 1989, and 1994, respectively. He is currently a full professor at the Department of Information and Communication Engineering of the Hanbat National University. His main research interests include computer networks, embedded system SW, cryptography, and network security.