Published in IET Communications
Received on 4th August 2011
Revised on 1st December 2011
doi: 10.1049/iet-com.2011.0566

ISSN 1751-8628

ASA: agent-based secure ARP cache management

M. Oh[1] *Y.-G. Kim*[1] *S. Hong*[2] *S. Cha*[1]

[1]*Department of Computer Science and Engineering, Korea University, 1, 5-ga, Anam-dong, SungBuk-gu, 136-701 Seoul, Korea*
[2]*School of Computing, Soongsil University, 511 Sangdo Dong, Dongjak Gu, 156-743 Seoul, Korea*
E-mail: always@korea.ac.kr

Abstract: Address resolution protocol (ARP) is widely used to maintain mapping between data link (e.g. MAC) and network (e.g. IP) layer addresses. Although most hosts rely on automated and dynamic management of ARP cache entries, current implementation is well-known to be vulnerable to spoofing or denial of service (DoS) attacks. There are many tools that exploit vulnerabilities of ARP protocols, and past proposals to address the weaknesses of the 'original' ARP design have been unsatisfactory. Suggestions that ARP protocol definition be modified would cause serious and unacceptable compatibility problems. Other proposals require customised hardware be installed to monitor malicious ARP traffic, and many organisations cannot afford such cost. This study demonstrates that one can effectively eliminate most threats caused by the ARP vulnerabilities by installing anti-ARP spoofing agent (ASA), which intercepts unauthenticated exchange of ARP packets and blocks potentially insecure communications. The proposed approach requires neither modification of kernel ARP software nor installation of traffic monitors. Agent uses user datagram protocol (UDP) packets to enable networking among hosts in a transparent and secure manner. The authors implemented agent software on Windows XP and conducted an experiment. The results showed that ARP hacking tools could not penetrate hosts protected by ASA.

1 Introduction

Although the address resolution protocol (ARP) [1] is one of the oldest (e.g. RFC 826 was defined in 1982) and most widely used (e.g. IEEE 802.11 and Ethernet), it is also one of the most vulnerable protocols. Every host, including gateways, maintains the ARP cache table which contains information on all valid pairs of media access control (MAC) and IP addresses on the local network. ARP cache management scheme includes static and dynamic modes. The former requires operator intervention in maintaining currency of all the cache entries which stay permanently until manually removed. Although known to be secure enough, it is seldom used in practice [2]. The latter, despite advantages of offering automated operation, has timeout mechanisms which cause entries to be automatically deleted regularly (e.g. 2 min on Windows XP). Spoofing and denial of service (DoS) attacks are the most serious ARP threats when cache is kept in dynamic mode. Several hacking tools such as ARPSpoofing [3], Cain&Able [4], Ettercap [5], arp-sk [6] and tratt [7] are publicly available.

ARP vulnerability occurs because of the lack of proper authentication mechanism to deal with forged ARP requests and replies. A malicious host may unilaterally broadcast ARP requests as if it is the gateway and attempt to manipulate ARP tables stored at other hosts on the local network. Unsuspecting hosts would then send all outbound packets to the attacker host which could then intercept and/or deliberately modify payload data. Many proposed to mend the insecurities of the ARP protocol design. Unfortunately, most ideas seem impractical, in that they suggest the ARP design be modified, expensive hardware installed to monitor potentially malicious ARP traffic, or ARP packets be encrypted.

We propose to install agent software, ASA (Anti-ARP Spoofing Agent) between the physical and data network layers of the protocol stack to perform the following tasks: (i) Intercept all the ARP packets (e.g. inbound/outbound requests as well as replies); and (ii) Send user datagram protocol (UDP) packets to achieve automated and secure management of the cache entries in static mode. We implemented ASA and conducted an experiment in which existing ARP hacking tools were deployed. Although hosts without ASA protection were easily compromised, the hacking tools could not penetrate ASA-protected hosts.

This paper is organised as follows: Section 2 briefly explains the ARP protocol as defined in the RFC 826 as well as its security vulnerabilities. Section 3 surveys past proposals to enhance ARP protocol security. In Section 4, we describe architectural design of ASA as well as its implementation. We also report an experiment which demonstrated ASAs ability to automatically manage ARP cache in static mode. Finally, Section 5 concludes the paper.

2 Vulnerabilities of existing ARP implementation

When a host needs to communicate with another host on the same network whose MAC address is unknown, it broadcasts an ARP request to all the hosts connected by the same

Ethernet. Only the host with the same IP is expected to issue a reply which includes its MAC address. When ARP cache is managed in dynamic mode, cache entries can be easily fabricated by forged ARP messages because proper authentication mechanism is missing. Fig. 1 illustrates how ARP spoofing [8] attacks typically occur. An attacker sends ARP request messages and fools other hosts to believe it as the gateway. All the packets targeted at the gateway will then be accessible by the attacker host. It may also manipulate cache entries on hosts *A* and *B* such that they exchange packets with each other via the attacker host as intermediate. The malicious host may secretly steal information and relay packets or manipulate the payload data and forward.

DoS [9, 10] is another threat in which an attacker generates excessive number of ARP messages, thereby resulting in unacceptably slow networking performance. It can even disable the network access altogether if (i) it can manipulate gateway's MAC address stored at other hosts to an invalid (i.e. non-existing) value; or (ii) it redirects all gateway traffic to itself and chooses to drop the packets.

Fig. 2 illustrates how ARP spoofing attack can be easily and successfully launched using one of the public domain hacking tools. Fig. 2*a* displays the content of ARP cache of a host before an attack was launched. As is typically the case, ARP cache is maintained in dynamic mode, and 'genuine' MAC addresses of other hosts connected on the same Ethernet are shown. The ARP hacking tools (e.g. ARPSpoof, Cain&Able, Ettercap, arp-sk and tratt) acquire IP and MAC addresses on the Ethernet to build a list by transmitting the same ARP request to every IP on the same subnet. As shown in Fig. 2*b*, upon launching an ARP spoofing attack using ARPSpoof, physical addresses of all other hosts are changed to that of the malicious host (e.g. 00-00-00-11-11-11). All the packets to be delivered to another host, say 210.118.56.1, is redirected to the attacker and not to the intended recepient.

3 Related work

There have been numerous proposals to amend known weaknesses of the ARP protocol. Most of the countermeasures belong to the following categories: cryptographic, host-based or server-based approaches.

3.1 Cryptographic approaches

Bruschi *et al.* [11] proposed a secure address resolution protocol (SARP), which uses asymmetric key cryptography

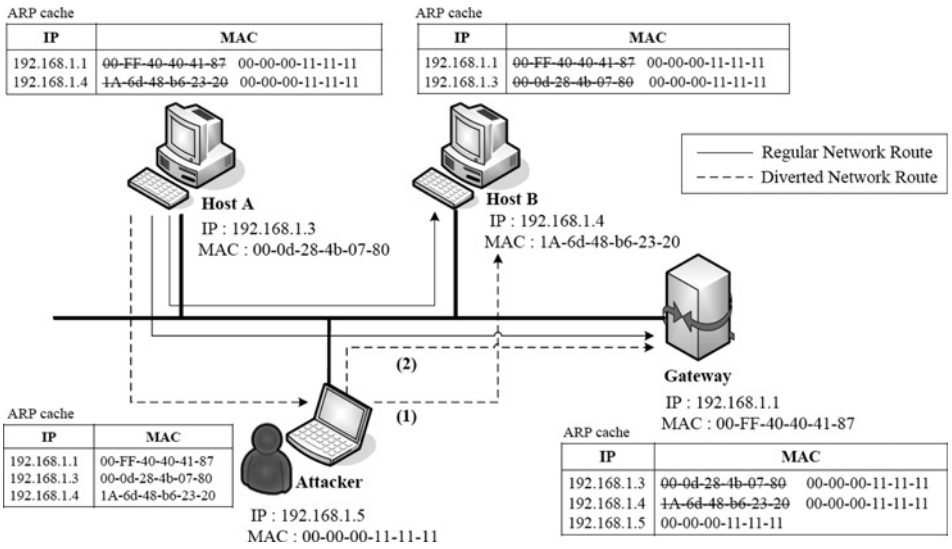

Fig. 1 *Example ARP spoofing attack*

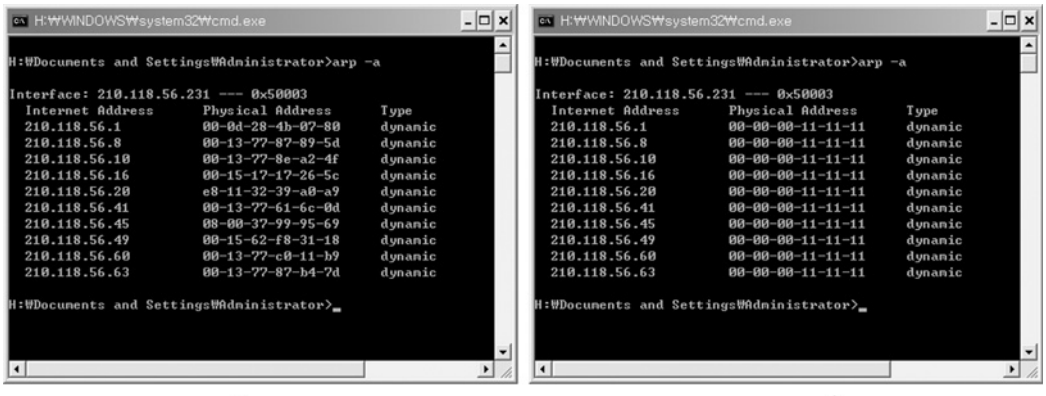

Fig. 2 *ARPSpoof in action*

a Before
b After

doi: 10.1049/iet-com.2011.0566

to authenticate the hosts in a local area network (LAN). In SARP, each host uses an invite–accept protocol to periodically register its IP–MAC pairs in a secure server. IP–MAC pairs are hashed by a message digest algorithm. This approach, however, requires modification of the ARP protocol as the sender needs to sign each ARP message with its private key, and the receiver needs to verify the signature with the sender's public key.

Goyal and Tripathy [12] used the combination of digital signatures and one-time password based on hash chains to authenticate a host. Their approach requires substantial overhead to perform signature generation, verification and key management.

Limmaneewichid and Lilakiatsakun [13] proposed an ARP authentication scheme based on ARP authentication trailer, named P-ARP, which consists of a magic number, nonce and the authentication data produced by the HMAC hash function. In order to hide the target IP address in an APR request message, additional operation such as hash function must be performed to create nonce and HMAC values. In addition, this approach is ineffective against ARP DoS attacks. Although cryptographic approach is generally an effective mechanism to guarantee integrity of ARP packets themselves, it often slows down the overall network throughput to an unacceptable level in practice.

3.2 Host-based approaches

Xing *et al.* [14] used WinPcaP library to capture and filter ARP packets. Whenever ARP response packet is received and the cache needs to be updated, it compares against the correct IP–MAC address pairs and corrects the contents of the local ARP cache if they are different. Ramachandran and Nandi [15, 16] checked inconsistencies of the addresses advertised by ARP request and TCP SYN packets. In order to build reliable IP–MAC pairs, they used the IP–MAC address advertised by ARP messages to build TCP SYN packets. When there are no ARP attacks, the destination MAC and IP in the TCP SYN packet has to be the source MAC and IP address reported in ARP messages. However, this approach would create a heavy traffic on the LAN if ARP DoS attacks are continuously generated to probe the network. Furthermore, this approach only detects an ARP attack but cannot prevent it.

Hou *et al.* [17] used a network intrusion detection system (e.g. Snort [18, 19]) to detect ARP attacks. They expanded the original ARP spoofing plug-ins in Snort by adding an ARP inspection module. The ARP inspection module automatically binds the correct gateway IP–MAC address in static mode. Likewise, Barbhuiya *et al.* [20] used a host-based intrusion detection system to detect ARP attacks. Their approach checked the integrity and authenticity of ARP replies using a combination of digital signatures and one time passwords based on hash chains. Information included in digital signatures (e.g. IP address to MAC address mapping, the local clock time and the tip of hash chain etc.) is used to verify the password. The host-based IDS answered ARP requests by sending digital signature as the ARP reply. Unfortunately, as acknowledged by the authors, at present this approach can only detect ARP attacks.

Philip [21] proposed a technique to prevent ARP cache poisoning attacks in wireless access point-based networks which may include wired clients. The access point creates a mapping table, which stores the mapping of IP addresses to MAC addresses. Since all the wireless clients registered with the access point have to obtain an IP address using the DHCP protocol, the mapping table contains the correct mapping of all the wireless clients that communicate through this access point. Therefore whenever the access point receives an ARP request or reply, it uses its mapping table to verify whether the mapping in the ARP request or reply is valid. Unfortunately, there are a couple of serious drawbacks to this approach. First, it does not support hosts that have static IP addresses. Second, it is designed to work with only Linksys routers, and this approach requires manufacturers to release device driver source code to update firmware which handles transmission of packets from one wireless client to another. Intellectual property issues prevent this approach from becoming widely adapted in practice.

Nam *et al.* [22] proposed man-in-the-middle (MITM)-resistant address resolution protocol (MR-ARP). Their approach consists of two parts: (i) mapping table (e.g. long-term table) keeps the values of IP–MAC pairs for connected hosts over longer periods; and (ii) conflict resolution mechanism based on voting prevents hosts against MITM attacks being launched. However, their approach was installed and evaluated on a small number of hosts. In addition, if a host is disabled by another type of DoS attack in the worst case, then the MITM attack cannot be valid.

Lastly, Trabelsi and El-Hajj [23] proposed a stateful ARP cache management mechanism based on a fuzzy logic. The stateful ARP cache, unlike existing and stateless ARP cache, does not update ARP cache entries upon receiving ARP requests. Instead, fuzzy logic controller aggregates various host properties (e.g. trust level and importance of each host) to detect malicious host. However, their approach is impractical because it is very difficult to decide the level of trustiness and importance of each host although they assume that the hosts in the network have different level of trustiness and importance.

3.3 Server-based approaches

Gouda and Huang [24] proposed an architecture in which a secure server is connected to the Ethernet and communications with the server take place using invite-accept and request-reply protocols. All ARP requests and replies occur between a host and the server, and replies are authenticated using shared pair keys. Kwon *et al.* [25] proposed a similar approach to securely manage IP addresses in a distributed network. This approach uses an agent which retrieves genuine IP–MAC pairs from a host and forwards them to the manager to construct reliable IP–MAC mapping. The manager node monitors if IP addresses of licensed hosts are changed, and unauthorised hosts are disconnected as they are assumed to have suffered spoofing attacks. Ortega *et al.* [26] proposed a scheme that can be used to block ARP attacks in small office, home office (SOHO) LANs. The scheme consists of two elements, namely a server that updates the ARP cache and a switch that blocks all ARP messages. However, they failed to address how the server collects the correct IP–MAC mappings so that it may generate correct reply to the incoming ARP requests.

Lootah *et al.* [27] implemented a secure IP–MAC address mapping in which an ARP reply is generated with an attached signature when a request is issued. A ticket is appended as a variable length payload. This approach uses a local ticket agent (LTA), a key management server, to issue a public key to obtain the IP–MAC from the ticket. This approach

is backward compatible with existing ARP, but it is susceptible to replay attacks. Furthermore, addition of cryptographic features in ARP may lead to some performance overhead. Pansa and Chomsiri [28] proposed revision of the dynamic host configuration protocol (DHCP) definition to include authentication of network devices and inclusion of mapping information between IP and MAC addresses. Although flawless in concept, it would cause serious compatibility problems because DHCP is one of the most popular protocols. In general, approaches based on secure server are generally ineffective because the server itself becomes the primary target of DoS attacks and has the potential of becoming a single point of failure.

Some high-end Cisco switches provide a feature called dynamic ARP inspection [29] through which the switch may drop ARP packets containing invalid IP/MAC pairs. The switch binds a physical port to a MAC address and maintains valid associations in content addressable memory (CAM) tables. However, if the first sent packet itself is a spoofed MAC address, the whole system fails. Likewise, various network monitoring tools (e.g. such as ARP-GUARD [30], ARPWATCH [31] and ARPDEFENDER [32]) inspect if ARP tables are arbitrarily changed. They maintain IP–MAC addresses of the ARP cache, periodically compare if changes have been made to the ARP cache, and alert administrators if necessary. These tools are cheaper than switches with port security but have slower response time compared to switches [16]. Furthermore, false alarms occur when genuine IP (or MAC) address changes occur.

4 ASA-based static and automated cache management

Primary contribution of this paper is that we found a way to maintain the integrity of ARP cache entries automatically in the static mode. Proposed approach grants only the ASA agent the authority to update the ARP cache table, and the known vulnerabilities of the ARP protocol are eliminated without requiring a secure server or modifying the existing protocol implementations. We implemented our idea, ASA, to demonstrate its effectiveness in practice and conducted an experiment in which existing ARP hacking tools were launched.

Fig. 3 illustrates how our idea (Fig. 3*b*) is different from the current practices (Fig. 3*a*) in ARP cache management by comparing a sequence of steps taken for host *A* to establish communication channel with host *B*. In the current ARP environment, host *A* can start a communication with host *B* immediately if the source host's ARP cache contains the MAC address of the destination (Step 1). Otherwise, host *A* must issue an ARP request, by including host *B*'s IP address, to all the hosts on the Ethernet whereas expecting only the host *B* to respond (Steps 2 and 3). Host *B*, the destination host with the same IP, learns that it is supposed to send back its MAC address in a unicast ARP reply packet (Steps 4 and 5). Host *A* updates its ARP cache table, and it may now proceed (Step 6).

In the proposed environment, it is naive to assume that ASA agent software is installed on all the hosts, and some hosts (e.g. host *B*) would attempt ARP communication with those who have the ASA agent installed. Networking must continue to be supported whereas security is never compromised. In an ASA-protected environment, hosts share IP–MAC pairs of all hosts in advance as soon as they appear or disappear on the Ethernet using a method explained in Section 4.2. A host on which the ASA agent is installed relies on UDP packets, instead of ARP, to exchange IP–MAC pairs of all hosts in an Ethernet.

If host *B*'s MAC address is missing, host *A*'s ASA-agent broadcasts ARP requests to all hosts on Ethernet, and Host

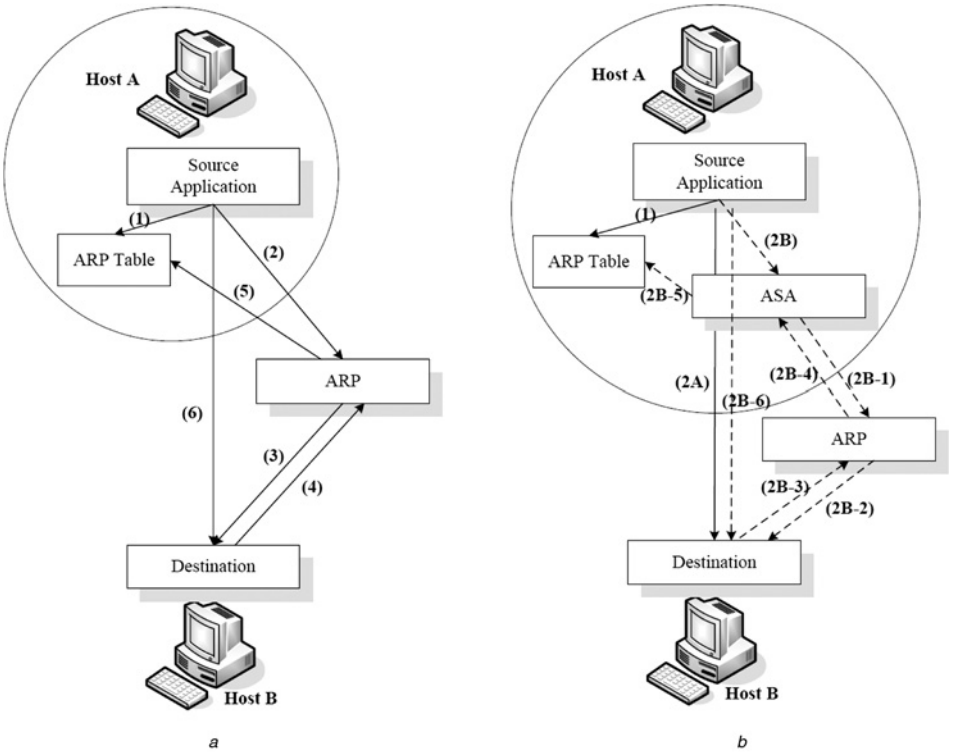

Fig. 3 *Overview of the proposed ASA-based approach*

a Existing approach
b Proposed approach

doi: 10.1049/iet-com.2011.0566

B would return its MAC address, via unicast, in an ARP reply packet. Host *B* may not have ASA-installed or could even be malicious (Steps 2B-1 and 2B-2). ASA intercepts the incoming ARP reply message, modifies host *A*'s cache table, and the communication may begin (Steps 2B-3 through 2B-6). It is crucial to realise that only ASA software is authorised to update ARP cache table according to the policy described in Section 4.2. Current ARP kernel software need not be modified or removed, and installation of ASA is sufficient to eliminate threat from ARP spoofing attacks. In our approach, ASA is built on Windows XP and implemented using the Windows device driver toolkit supported by Visual C++ programming language.

4.1 Anti-ARP spoofing agent (ASA) design architecture

ASA aims to protect ARP cache entries of from ARP poisoning attacks, and it consists of the following components: ARP Filter Driver ('ASA Filter'), ARP Controller ('ASA Controller'), and ARP Controller UI ('ASA UI') (see Fig. 4).

ASA filter is implemented between an ARP layer (e.g. data link layer) and a network interface card (NIC) (e.g. physical layer). Because it is implemented in a lower layer than the ARP layer, it is possible to intercept all incoming or outgoing ARP messages and have a complete control on the ARP cache table being maintained at each host. ASA Filter, as depicted in detailed in Figs. 6 and 7, blocks all ARP messages generated by the hosts on the local Ethernet except those generated by gateway host. Such policy is strictly enforced because the counterpart host might be hostile or have been compromised.

ASA controller keeps ARP cache table in static mode at all times and is responsible for maintaining its integrity. Because ASA filter blocks all of the 'original' ARP messages, one must provide an equivalent method to enable mutual exchange of IP–MAC pairs between hosts. ASA uses UDP for such purpose. For example, when a node is inserted or its MAC address changed, ARP cache entries are configured by ASA controller using UDP packets. Finally,

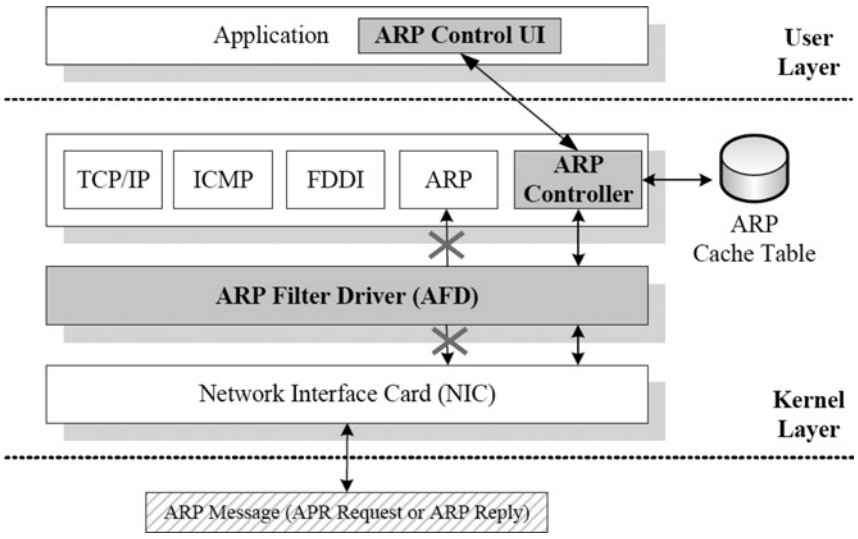

Fig. 4 *Protocol stack of ASA*

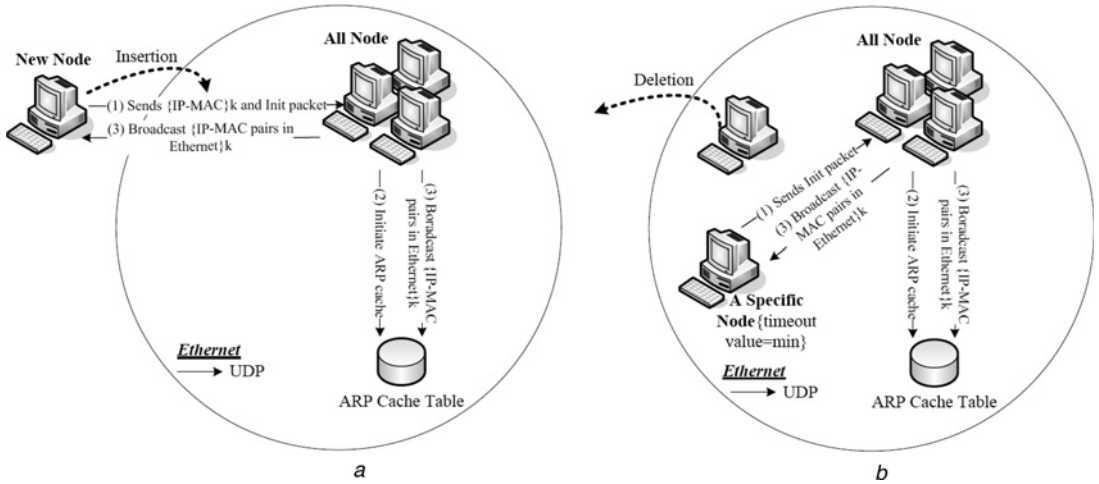

Fig. 5 *Building static ARP cache entries*

a Insertion of a new node in a Ethernet
b Node deletcon and rebuilding ARP cache entries

the ARP controller UI provides user interface, in which a user can configure the ASA environment such as turning on or off the ASA agent.

When a new host joins local Ethernet, as shown in Fig. 5*a*, the following steps are taken to update ARP configurations at all other hosts. First, ASA-protected broadcasts its IP–MAC pair and the 'init' packet (e.g. 'init' event) to all hosts in the Ethernet. Upon receiving the 'init' packet, all ASA-protected hosts discard their ARP entries and broadcast each host's IP–MAC pairs encrypted with symmetric key k to effectively protect the packet content from being contaminated by an attacker. All hosts create ARP cache table again based on newly received information.

In the traditional ARP implementation, timeout mechanism is used for a host to update the ARP cache table whenever any host encounters timeout [33]. In ASA design, however, we chose to introduce random timeout values between 2 and 10 min. Whenever any node experiences timeout, all the hosts connected on the local network broadcast its own and encrypted IP–MAC address pair, and all the hosts rebuild its ARP cache (see Fig. 5*b*). It must be noted that such policy is compatible with existing ARP policies, and the overhead appears to be low enough because the packets containing the IP–MAC values are relatively small (i.e. 32 or 48 bytes).

4.2 ARP filtering policy

There are three possible types of hosts one must address when developing an effective filtering policy of ARP messages: ASA-protected hosts, gateway and other hosts on the same network which do not have ASA installed. Fig. 6 illustrates how ARP filtering policy works in each situation.

The fundamental principles enforced by the policy are: (i) ASA blocks all inbound ARP requests except those generated by gateway; and (ii) exchange of encrypted UDP messages containing IP–MAC values guarantee consistency and integrity of ARP tables kept at hosts protected by ASA agents. The filtering policy must also address the possibility of an ASA-protected host having to communicate with other local hosts on which ASA agent is not installed. Should an ARP request come from host C, it clearly does not have ASA agent installed (Case 2). Otherwise, it would not have sent the ARP request in the first place. Therefore the ASA filter at host A intercepts the ARP request message but returns host A's MAC address to enable communication. It must be emphasised that such ARP requests never result in updating host A's ARP table. When host A needs to establish a network connection to host C which does not have the ASA agent installed (Case 3), ASA UI asks the 'user' if it is really OK to establish the communication with a potentially malicious host. Such a case is shown in lines 5 through 8 in the pseudo-code of a filtering algorithm as shown in Fig. 7. Upon receiving confirmation, regular ARP protocol is used and host C's reply is trusted.

The gateway needs to receive a special attention as it serves as the interface with other networks. The gateway also needs to maintain mappings between IP–MAC pairs to support data transmission from the internal network to the external network and vice versa. The ASA-protected host obtains the gateway's MAC address during a booting stage existing ARP messages and that gateway's MAC address is securely managed (Case 4). Fig. 7 describes the implemented filtering policy in pseudo-code annotated with comments indicating the corresponding pattern.

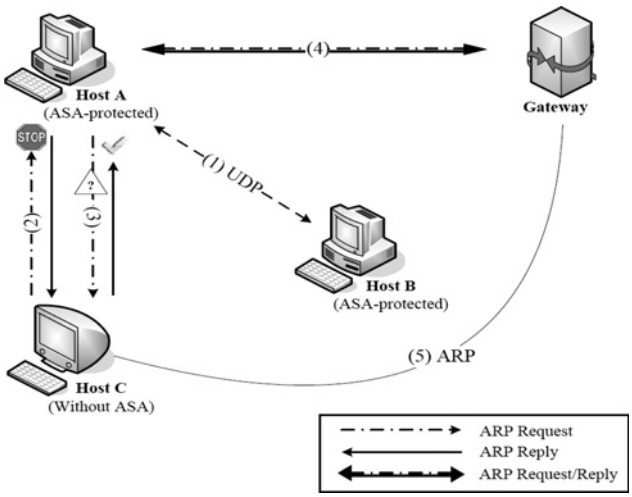

Fig. 6 *ARP filtering policy enforced by ASA*

Protection of the gateway against ARP attacks, whereas essential in achieving overall network security, is outside the scope of ASA design. The routers are frequently used as gateways, and the ARP table is usually stored in firmware. In such a configuration, the installation of ASA on commercial routers by the end user (as opposed to manufacturers) is impractical. Should the gateway's MAC address change whereas the network is in operation, the timeouts triggered at any host would force all ASA-protected hosts to build the ARP table from scratch. Although the gateway might become a victim of ARP attacks, the ARP tables maintained at ASA-protected hosts remain intact. The ASA-protected hosts can detect when the gateway's ARP table is attacked by analysing the inbound ARP reply message sent by an attacker to change the gateway's ARP cache entries. In the worst case, even when the gateway becomes the victim of an ARP attack, host A's traffic is never intercepted by a malicious host.

A malicious host may disguise its identity as a gateway by corrupting the gateway's MAC address stored in the ASA-protected host's ARP table with that of the malicious host. In such a case, the ARP table remains uncorrupted because the ASA blocks all the inbound ARP requests and replies. If an attacker attempts to change the MAC address of a host A in the gateway's ARP table with that of its own, it would have to broadcast an ARP request. Such attempts can be easily monitored by each host. Finally, the ARP DoS attack can be effectively defeated, despite some overhead involved, should an attacker generate an overwhelming number of ARP messages. In the ASA design, as shown in case 2 (lines 11–12 in Fig. 7), ARP requests from untrusted hosts are honored only once and the rest ignored during a given time interval (e.g. random values between 2 and 10 min).

4.3 Evaluation

In order to empirically evaluate the effectiveness of ASA, we implemented an ASA agent, installed in a network illustrated in Section 2, and launched the public domain ARP attack tools to mimic the ARP spoofing attacks. Whereas the hosts were easily compromised earlier (Fig. 2*b*), the ARP tables are securely and automatically managed while kept in static mode (Fig. 8).

Comparison between the proposed idea and the existing techniques, shown in Table 1, is provided below with respect to five criteria: category, operational mode of the

doi: 10.1049/iet-com.2011.0566

ARP cache table, communication protocol, types of security assurance it provides against ARP attacks targeted at a local host as well as the gateway.

The proposed method overcomes the weaknesses identified in the existing proposals. Most importantly, the ARP cache table is always kept in static mode. There are no known security vulnerabilities when the ARP cache table is kept in static mode. Management of the ARP cache table, however, is fully automated whenever changes to the local network membership occur. Furthermore, our technique does not mandate the ASA be installed on all the hosts connected to the local Ethernet. Although hosts on which the ASA is not installed remain unprotected against the known ARP attacks, security compromise at such vulnerable hosts never results in security failure at ASA-protected hosts.

Second, the proposed technique poses no compatibility problems. The existing and widely used communication protocol definitions need not be modified at all. Our approach uses UDP, instead of vulnerable ARP, to exchange IP–MAC pairs among the ASA-protected hosts. When the ASA-protected hosts need to communicate with the gateway or other hosts that are outside the protection of ASA, the ARP protocol is used to accomplish semantic-preserving networking capability whereas never resulting in security failure at the ASA-protected hosts.

Lastly, protection of the gateway is outside the scope of the proposed approach. Although the gateway host may become victims of the ARP attacks and the internet working capability is temporarily disabled, the ASA-protected hosts remain secure and can detect the fact that integrity of the gateway host has been corrupted.

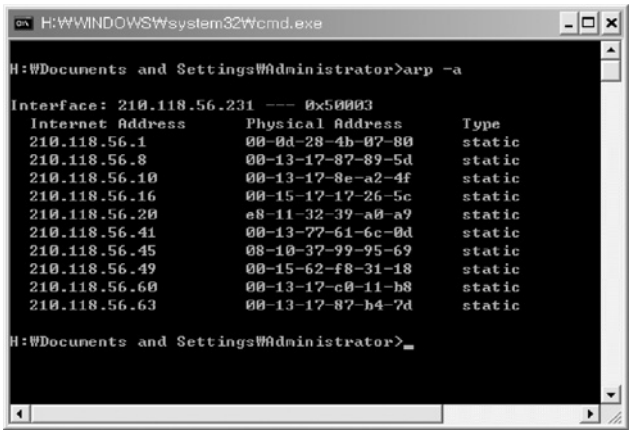

Fig. 8 *Secure and automated ARP cache management whereas kept in static mode*

5 Conclusion

In this paper, we addressed how ARP attacks can be effectively defeated without modification of the ARP kernel software. Many approaches have attempted to address vulnerability associated with ARP cache management in dynamic mode. Yet, the security risk remained the same.

1: **If** a host sends an outbound ARP message:
2: **If** the ARP message is an outbound ARP request:
3: **If** the IP address in the outbound ARP request corresponds to the gateway's IP address: /* **Case 4** */
4: ASA broadcasts the outbound ARP request to the gateway.
5: **Else**,
6: ASA reports the outbound ARP request to a user of the host.
7: **If** source is identified and approved by the user: /* **Case 3** */
8: ASA broadcasts the outbound ARP request to the source.
9: **Else**,
10: ASA drops the outbound ARP request.
11: **Else**, **if** the ARP message is an outbound ARP reply: /* **Cases 2 and 4** */
12: ASA broadcasts the outbound ARP reply. /* ASA broadcasts the outbound ARP reply only once within random timeout values in **Case 2** */
13: **Else**, **If** a host receives an inbound ARP message:
14: **If** the ARP message is an inbound ARP request:
15: **If** the IP-MAC pair in the ARP request corresponds to the gateway's IP-MAC pair registered in ASA-protected Host: /* **Case 4** */
16: ASA broadcasts an outbound ARP reply to the gateway,
17: and it inserts the IP-MAC pair into ARP cache entries.
18: **Else**: /* **Case 2** */
19: ASA drops the inbound ARP request,
20: and it broadcasts an outbound ARP reply to the source of the inbound ARP request.
21: **Else**, if the ARP message is an inbound ARP reply:
22: **If** the IP address in the inbound ARP reply corresponds to the gateway's IP-MAC pair registered in ASA-protected Host: /* **Case 4** */
23: ASA inserts the IP-MAC pair of the inbound ARP reply into ARP cache entries.
24: **If** the IP address in the inbound ARP reply corresponds to the IP address of the outbound ARP request sent by an ASA-protected host: /* **Case 3** */
25: ASA inserts the IP-MAC pair of the inbound ARP reply into ARP cache entries.
26: **Else**:
27: ASA drops the inbound ARP reply.

Fig. 7 *Pseudo-code to process ARP messages in an ASA-protected host*

Table 1 Comparison of various approaches to defend hosts against ARP attacks

	Category	Operational mode of ARP cache table	Communication protocol	Types of security assurance against ARP attacks	
				At local host	At gateway
Bruschi *et al.* [11]	cryptography, server-based	dynamic	modified ARP	defeat ARP spoofing attacks	N/A
Goyal and Triphyp [12]	cryptographic server-based	dynamic	modified ARP	defeat ARP spoofing attacks	N/A
Limmaneewichid and Lilakiatsakun [13]	cryptographic	dynamic	ARP	defeat ARP spoofing attacks	N/A
Xing *et al.* [14]	host-based (application)	dynamic	ARP	defeat ARP spoofing attacks	N/A
Ramachandran and Nandi [15, 16]	host-based	dynamic	ARP	detection of ARP spoofing	N/A
Hou *et al.* [17]	host-based (IDS plug-ins)	dynamic (gateway's IP–MAC pairs in static mode)	ARP	detection of ARP spoofing	N/A
Barbhuiya *et al.* [20]	host-based IDS, cryptographic	dynamic	ARP	detection of ARP spoofing	N/A
Philip [21]	host-based (router)	dynamic	ARP	defeat ARP spoofing attacks	N/A
Nam *et al.* [22]	host-based	dynamic (ARP table, long-term table)	ARP	defeat ARP spoofing attacks	N/A
Trabelsi and EL-Hajj [23]	host-based	dynamic	ARP	defeat ARP spoofing attacks	N/A
Gouda and Huang [24]	server-based cryptographic	dynamic	ARP	defeat ARP spoofing attacks	N/A
Kwon *et al.* [25]	server-based (server, agent)	dynamic	ARP	defeat ARP spoofing attacks	N/A
Ortega *et al.* [26]	server-based	dynamic	ARP	defeat ARP spoofing attacks	N/A
Lootah *et al.* [27]	server-based cryptographic	dynamic	ARP	only detection of ARP spoofing	N/A
Pansa and Chomsiri [28]	server-based	dynamic	ARP, modified DHCP	defeat ARP spoofing attacks	N/A
proposed approach	host-based (agent)	static	UDP among ASA-protected hosts	defeat ARP spoofing and DoS attacks	detection of gateway being spoofed or under DoS attacks

We developed an agent-based approach which essentially blocks the unauthenticated exchange of ARP requests and replies among hosts. We demonstrated in an experiment that the ARP cache can be managed securely and automatically in static mode. The proposed agent-based approach, ASA, uses UDP packets containing IP–MAC pairs encrypted by a symmetric key to block the ARP messages.

Although clearly effective, one must never stop exploring how the proposed technique might be defeated by new attacks. One possibility includes the strength of k-encryption algorithm to authenticate the UDP packet exchange. Perhaps, a stronger authentication mechanism might become necessary to provide superior protection against sniffing attacks. In addition, experimental validation has been conducted in controlled and small-scale networks. Effectiveness must be validated next in a large and industrial setting.

6 Acknowledgments

This work was partially supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract.

7 References

1 RFC-826: 'An ethernet address resolution protocol', 1982
2 Kozierok, C.M.: 'TCP/IP guide' (No Starch Press, 2005, 1st edn.)
3 http://arpspoof.sourceforge.net, accessed July 2011
4 http://www.oxid.it/cain.html, accessed July 2011
5 http://ettercap.sourceforge.net/index.php, accessed July 2011
6 http://sid.rstack.org/arp-sk/, accessed July 2011
7 http://www.toolcrypt.org/tools/tratt/index.html, accessed July 2011
8 http://en.wikipedia.org/wiki/ARP_spoofing, accessed July 2011
9 Kumar, S.: 'Impact of distributed denial of service (DDoS) attack due to ARP storm', *Lect. NotesComput. Sci.*, 2005, **3421**, pp. 997–1002
10 Yu, J., Fang, C., Lu, L., Li, Z.: 'Mitigating application layer distributed denial of service attacks via effective trust management', *IET Commun.*, 2010, **4**, (16), pp. 1952–1962
11 Bruschi, D., Ornaghi, A., Rosti, E.: 'S-ARP: a secure address resolution protocol'. Proc. 19th Annual Computer Security Applications Conf. (ACSAC2003), Las Vegas, NV, USA, December 2003, pp. 66–74
12 Goyal, V., Tripathy, R.: 'An efficient solution to the ARP cache poisoning problem', *Lect. Notes Comput. Sci.*, 2005, **3574**, pp. 40–51
13 Limmaneewichid, P., Lilakiatsakun, W.: 'P-ARP: A novel enhanced authentication scheme for securing ARP'. Proc. 2011 Int. Conf. on Telecommunication Technology and Applications, May 2011, pp. 83–87
14 Xing, W., Zhao, Y., Li, T.: 'Research on the defense against ARP spoofing attacks based on WinPcaP'. Proc. Second Int. Workshop on Education Technology and Computer Science (ETCS2010), Wohan, China, March 2010, pp. 762–765

doi: 10.1049/iet-com.2011.0566

15 Ramachandran, V., Nandi, S.: 'Detecting ARP Spoofing: an active technique', *Inf. Syst.Secur.*, 2005, **3803**, pp. 239–250
16 Hubballi, N., Roopa, S., Ratti, R., *et al.*: 'An active intrusion detection system for LAN specific attacks', *Lect. Notes Comput. Sci.*, 2010, **6059**, pp. 129–142
17 Hou, X., Jiang, Z., Tian, X.: 'The detection and prevention for ARP spoofing based on Snort'. Proc. Second Int. Conf. on Computer Application and System Modeling (ICCASM2010), XiaMen, China, November 2010, pp. 137–139
18 Snort, http://www.snort.org, accessed July 2011
19 Caswell, B., Beale, J., Foster, J.C., Faircloth, J.: 'Snort 2.1- intrusion detection' (Syngress, 2007)
20 Barbhuiya, F.A., Roopa, S., Ratti, R., *et al.*: 'An active host-based detection mechanism for ARP-related attacks', *Commun. Comput. Inf. Sci.*, 2011, **132**, (2), pp. 432–443
21 Philip, R.: 'Securing wireless networks from ARP cache poisoning'. Maser's thesis, San Jose State University, 2007
22 Nam, S.Y., Kim, D., Kim, J.: 'Enhanced ARP: preventing ARP poisoning-based man-in-the-middle attacks', *IEEE Commun. Lett.*, 2010, **14**, (2), pp. 187–189
23 Trabelsi, Z., El-Hajj, W.: 'Preventing ARP attacks using a fuzzy-based stateful ARP cache'. Proc. IEEE Int. Conf. on Communications (ICC2007), June 2007, pp. 1355–1360
24 Gouda, M.G., Huang, C.T.: 'A secure address resolution protocol', *Comput. Netw.: Int. J. Comput. Telecommun. Netw.*, 2003, **41**, (1), pp. 57–71
25 Kwon, K., Ahn, S., Chung, J.W.: 'Network security management using ARP spoofing', *Lect. Notes Comput. Sci.*, 2004, **3043**, pp. 142–149
26 Ortega, A.P., Marcos, X.E., Chiang, L.D., Abad, C.L.: 'Preventing ARP cache poisoning attacks: a proof of concept using OpenWrt'. Proc. Network Operations and Management Symp. (APNOMS2009), September 2009, pp. 1–9
27 Lootah, W., Enck, W., Mcdanie, P.: 'TARP: ticket-based address resolution protocol'. Proc. 21st Annual Computer Security Applications Conf. on (ACSAC2005), Tucson, AZ, USA, December 2005, pp. 108–116
28 Pansa, D., Chomsiri, T.: 'Architecture and protocols for secure LAN by using a software-level certificate and cancellation of ARP protocol'. Proc. Int. Conf. on Convergence and Hybrid Information Technology (ICCIT2008), Busan, Korea, November 2008, pp. 21–26
29 Cisco Systems, Configuring Dynamic ARP Inspection, Catalyst 6500 Series Switch Cisco IOS Software Configuration Guide, Release 12.2SX. 2006
30 http://www.arp-guard.com, accessed July 2011
31 http://www.arpalert.org, accessed July 2011
32 http://www.arpdefender.com, accessed November 2011
33 RFC-1121: 'Act One – The Poems, Network Working Group', 1989