An ICMP based Secondary Cache approach for the detection and prevention of ARP Poisoning

Nikhil Tripathi[1, 2], B. M. Mehtre[2]
[1]School of Computer and Information Sciences, Hyderabad Central University, Hyderabad, India
[2]Center for Information Assurance and Management, Institute for Development and Research in Banking Technology (Established by Reserve Bank of India), Hyderabad, India
(nikhiltripathi684@gmail.com, bmmehtre@idrbt.ac.in)

Abstract - **Address Resolution Protocol (ARP) poisoning is one of the most basic technique employed in computer hacking. ARP poisoning is used when a host is used to poison ARP cache of another host in order to send packets to some other destination than the intended one.**

This paper presents a feasible technique to detect and prevent the ARP poisoning by removing the multiple entries for the same MAC address or IP address from the ARP table using a secondary cache. This secondary cache contains the entries according to Internet Control Message Protocol (ICMP) responses. Since this technique prevents multiple entries for same IP address or MAC address, it also mitigates IP exhaustion problem. The secondary cache is maintained at every host which makes this technique distributed in nature, thereby prevents it from single point failure. Experimental results are also provided to support the proposal.

Keywords - **Address Resolution Protocol, ARP Poisoning, IP Exhaustion, Man-in-the-Middle, ICMP, Network Security, Cyber Defense**

I. INTRODUCTION

According to RFC826 [1], the Address Resolution Protocol (ARP) is used by computers to bind logical addresses (IP) to physical addresses (MAC). MAC address works at Data Link Layer of TCP/IP protocol stack. Within LAN, an IP address is translated to a MAC address for the purpose of communication. If IP-MAC binding for a particular host is not present in ARP cache, ARP is used to get and store this binding.

ARP involves the exchange of two different types of messages, ARP Request and ARP Reply. ARP Request messages are sent to get the MAC address for a given IP address. ARP Reply messages are sent in response to the ARP request messages. After receiving reply messages, the hosts update their ARP cache with the IP-MAC binding. ARP is an unauthenticated and stateless protocol. It is unauthenticated in the sense that any illegitimate user can send fake ARP Reply messages by spoofing his/her real identity. It is stateless in the sense that anyone can send unsolicited ARP Reply messages even in the absence of corresponding ARP Request messages.

ARP poisoning is generally performed to get Man-in-the-Middle (MITM) position. Once an attacker gets MITM position, s/he can launch other fatal attacks, e.g. DoS Attacks, fake DNS replies, etc. Also, s/he can eavesdrop the secure connections using MITM attacks in Hyper Text Transfer Protocol Secure (HTTPS) [2].

These problematic scenarios demonstrate the weaknesses in the ARP operations. As a result, there should be an efficient mitigation approach for the prevention of ARP poisoning which should be robust and compatible with the existing networks. Also, the approach should not give rise to other loopholes which can be exploited for other attacks.

The rest of the paper is structured as follows: Section II discusses the background of ARP protocol and ARP poisoning. In Section III, we discussed the related work done for the mitigation of ARP poisoning attacks. We described our proposed solution in Section IV along with the algorithms and attack scenarios. Results of the experiment are shown in Section V. Finally, Section VI concludes the paper.

II. BACKGROUND

A. Address Resolution Protocol

In TCP/IP suite, ARP resides in the Data Link Layer while in the Open Systems Interconnection (OSI) model, it is often considered as residing between Data Link and Internet Layer. In the Internet Layer of TCP/IP stack, a host is identified by its 32-bit IP address. However, in Data Link Layer, an interface is identified by its 48-bit MAC address.

All the packets flowing between different hosts have source address and the destination address. The source address is used to identify the sender of the packet while the destination address is used to send the packet to the intended destination. Within LAN, the packets are sent using MAC address of the system which has the intended IP address. However, if the destination host belongs to another subnet, the packets are sent to the default gateway. In this case, the destination IP address is used to send the packet to the appropriate network and finally MAC address is used within that network to submit the message to destined host.

To resolve the IP address into the MAC address, a host sends a broadcast ARP request. This request is received by every host within the subnet. However, according to the protocol, only the host with the intended IP address replies to the requesting host with its MAC address. All other hosts simply drop the request message because they do not possess the intended IP address. This IP-MAC binding is stored in a volatile cache memory which is known as ARP cache/table. This cache is updated at regular time intervals to avoid old and invalid

entries. Also, the IP-MAC bindings can be entered manually in the ARP cache. These manual entries are also known as static entries.

Suppose host A wants to communicate with another host B. A checks its ARP cache for a binding of B's IP address with a MAC address. If A finds such entry, it will use the given MAC address. Otherwise, as shown in Fig. 1, A will broadcast an ARP request. When B will receive this request, it will send ARP Reply message. This message will be a unicast one sent to A. A will store this binding in its ARP cache. ARP Reply is shown in Fig. 2.

B. ARP Cache Poisoning

This attack exploits the vulnerability of the ARP operations. Since ARP is an unauthenticated and stateless protocol, an attacker can easily send spoofed ARP replies. As a result, the receiving host updates its ARP cache without any doubt.

Fig. 3 demonstrates this attack. Suppose the attacker is host C. C sends a spoofed ARP Reply to A saying that 'Host B's IP address maps to C's MAC address' and another spoofed ARP Reply to B saying that 'Host A's IP address maps to C's MAC address'. As a result, A and B update their ARP cache according to the received ARP Replies. This type of update in the A's and B's caches make C to get MITM position. Once MITM position is achieved, the attacker can launch different attacks like DNS spoofing, session hijacking, DoS attacks, etc.

III. RELATED WORK

Several solutions have been proposed in order to prevent the ARP poisoning attacks. Using static ARP entries is the simplest mechanism to protect the ARP poisoning. Somnuk Puangpronpitag and Narongrit Masusai [3] have proposed a mechanism for ARP spoofing detection and prevention using static IP-MAC pair entries of every host on the network. However, this solution is not feasible for organizations having large LAN segment. Some simple solutions include monitoring

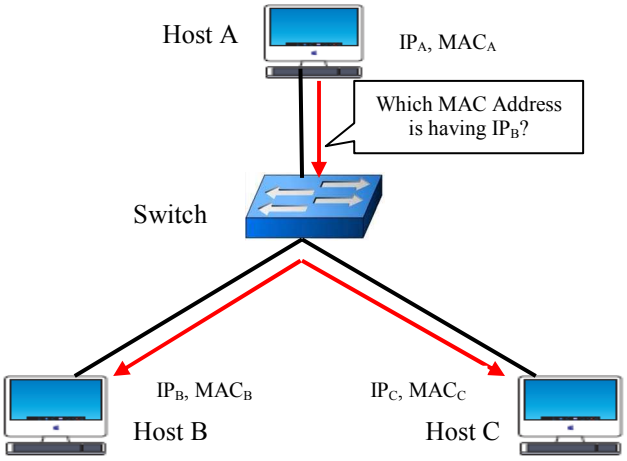

Fig. 1. Host A broadcasts ARP Request for Host B

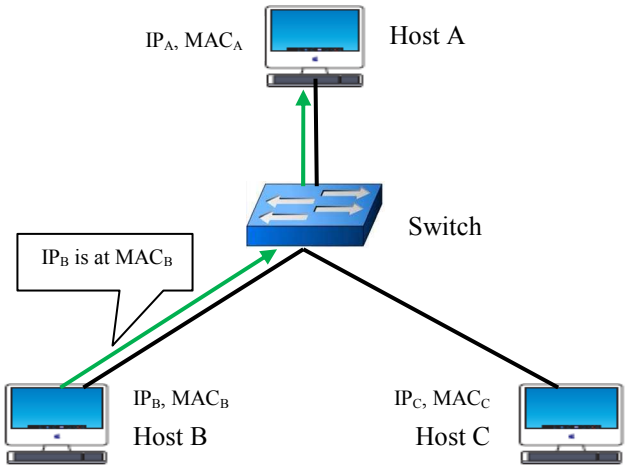

Fig. 2. Host B sends a unicast ARP Reply to Host A

of ARP cache [4]. In Linux, it can be performed using "arp -a" command. Some approaches involve restriction of ICMP packets to prevent ARP poisoning via ICMP packets [5].

Different improvements to ARP protocols are also suggested in S-ARP [6], Ticket based ARP [7] and Enhanced ARP [8]. In Secure Address Resolution Protocol, each host has its own public/private key certificates. These certificates are used to authenticate every ARP message flowing through the network. The Authoritative Key Distributor (AKD) acts as a central server to distribute the public key of a host. This scheme also requires Secure-DHCP instead of usual DHCP. This scheme requires changes in the implementation of ARP protocol which is not feasible. Also, the AKD may lead to a problem of single point failure.

In Ticket based ARP, a ticket is appended with every ARP message. The ticket is distributed to hosts by a Local Ticket Agent (LTA). Also, the LTA is the issuer of the tickets. However in this case also, LTA creates a single point failure problem.

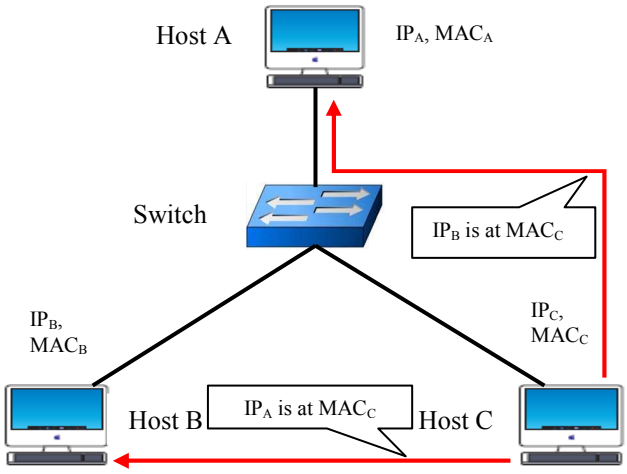

Fig. 3. Host C performing ARP poisoning on A and B

The Antidote approach [9] checks if the previous MAC is alive. If it is alive, the ARP cache is not updated and new fake MAC address is added to banned address list. However, if attacker comes before the real user within the network, he can already poison the ARP cache. This causes the real MAC address to get banned. As a result, the attacker can easily poison the other hosts because the poisoned victim cannot reply to the ARP requests sent by the hosts.

Sumit Kumar and Shashikala Tapaswi [10] proposed a centralized technique for detection and prevention of ARP poisoning. In this technique, an ARP Central Server (ACS) is used to validate the ARP tables' entries of all the hosts within the network. Clients also maintain a secondary long term cache in this approach. However, this technique does not address the IP exhaustion problem which an attacker can create within a network. Moreover, this technique is centralized in nature. As a result, the failure of ACS server leaves the network insecure.

Gopi Nath Nayak and Shefalika Ghosh Samaddar [11] proposed two solutions in order to prevent ARP poisoning. The first one sends *arping* request messages to the default gateway at fixed time intervals. However, this technique is limited for the checking of host to gateway traffic only. The traffic between one host to another host is not analyzed for detection and prevention of ARP poisoning attack. The second technique monitors the ARP table at regular intervals. It generally checks how many IP addresses are associated with the MAC address of gateway. If the number is more than 1, it alerts the user about possible poisoning. The disadvantage is that the MAC address of the gateway should be known in advance.

Gao Jinhua and Xia Kejian [12] proposed an ICMP protocol based detection algorithm for ARP spoofing. This algorithm collects and analyzes the ARP packets and then injects ICMP echo request packets to probe for malicious host according to its response packets. However, the algorithm is based on a database available at Detection host. Thus, it causes the single point failure problem. Also, the algorithm does not address the issue of ARP poisoning using fake ICMP echo requests. An attacker can send a fake ICMP echo request with the spoofed source IP address of a legitimate host and the source MAC address of itself. As a result, when the victim host receives this message, it will update its ARP cache with illegitimate binding having attacker's MAC address bind to a spoofed legitimate IP address allotted to another host within the subnet. The attacker can send the same type of fake ICMP echo requests to default gateway also so as to get the MITM position between the default gateway and the victim host.

IV. PROPOSED SOLUTION

In this paper, we propose an algorithm which makes use of ICMP requests and replies to store the IP-MAC bindings in a secondary ARP cache. Every client in the network maintains a secondary ARP cache which may be just a text file having IP-MAC bindings. This secondary table/cache is a permanent cache with lifetime validity. The entries are made and validated in this cache using ICMP protocol. The secondary cache makes sure that there are no multiple entries for a single IP address or for a single MAC address. Then according to these entries present in secondary cache, the primary cache is also updated either by adding or deleting the entries. The algorithm is made different for entering hosts and already existing hosts. Entering host is a new host which entered into a network when other hosts are already available in the network. Existing host is a host which is already present when a new host enters into the network. It is understood that all the entering hosts are existing hosts also. Thus, entering hosts will execute the algorithm for entering as well as existing hosts. The entering hosts' algorithm will be executed only once when a host will enter into the network. Already existing hosts need not execute the entering host algorithm.

According to the algorithm, when host C sends an ARP Reply to host B to bind the C's claiming IP address (IP_{CC}) to C's MAC address (M_C), B checks its secondary cache for an entry associated with either M_C or IP_{CC}. This results in two situations.

1) *No entry in secondary cache:* If there is no such entry, B will send an ICMP echo request to IP_{CC}. If B does not receive the echo reply back, it will straightaway reject the new binding (IP_{CC}-M_C) and delete it from primary cache. Otherwise, if B receives the echo reply back, it will create an entry for IP_{CC}-M_C binding in secondary cache. Also, B will create a static entry in primary cache having validity of 10 minutes. B sends the ICMP request to make sure that the sender is not sending fake ARP Replies using some random and invalid MAC addresses to create the IP exhaustion problem.

2) *Entry found in Secondary Cache:* This situation further gives rise to three cases.

(a) *Entry matching with C's MAC address (M_C):* If B finds an entry having C's MAC address (M_C) in secondary cache, it looks for the IP address bind to it, say IP_{MC}. Now, B will generate and send an ICMP echo request destined to IP_{MC}. This request is generated using raw socket programming. If B does not receive the echo response back, B will consider that C has released the IP address IP_{MC} and thus, it will delete the old binding (IP_{MC}-M_C) and update the new binding (IP_{CC}-M_C) in secondary cache. This entry will also be made statically in primary cache for 10 minutes. Otherwise, if B receives the echo reply back, it means the old mapping exists and M_C still possesses IP_{MC}. As a result, B will not update its secondary cache with the IP_{CC}-M_C binding. This is due to the fact that one interface at one system can have only one IPv4 address at a time. Moreover, since the primary cache has automatically been updated with the new binding (IP_{CC}-M_C) due to the ARP reply sent by C, B will delete this entry from primary cache and update it with older entry (IP_{MC}-M_C) by adding it statically for 10 minutes. This is done using secondary cache because it contains the older mapping. This prevents ARP poisoning. Since there is only one entry for one MAC addresses, no MAC address can have more than one

IP address. This enables the solution to prevent the IP exhaustion problem as well.

(b) *Entry matching with C's claiming IP address (IP_{CC}):* In this case, B will generate and send an ICMP echo request destined to IP_{CC}. The MAC address (M_O) corresponding to this claiming address in secondary cache will be the destination MAC address for sending the echo request. This request also is generated using raw socket programming. If B does not receive the echo response back, B will consider that M_O has released IP_{CC} and thus, it will delete the old binding (IP_{CC}-M_O) and update the new binding (IP_{CC}-M_C) in secondary cache. This entry will also be made statically in primary cache for 10 minutes. Otherwise if B receives the echo response back, it means IP_{CC} is already being used by M_O. At a time, one IP address can be allotted to only one host within the network. Moreover, since the primary cache has automatically been updated with the new binding (IP_{CC}-M_C) due to the ARP reply sent by C, B will delete this entry from primary cache and update it with older entry (IP_{CC}-M_O) by adding it statically for 10 minutes. This is done using secondary cache because it contains the older mapping. This prevents ARP poisoning attacks.

(c) *Entry with IP_{CC} bound to M_C:* If B finds an entry which already shows the binding IP_{CC}-M_C, B will directly ignore this reply because this reply will also lead to same binding.

The algorithms for the entering hosts and the existing hosts are explained as follows:

A. Algorithm for entering hosts

1) After receiving IP address from DHCP server, send ARP reply to the broadcast address to make other hosts aware of this IP-MAC binding.
2) Send ICMP echo request to every host.
3) Receive the echo reply coming from host and thus, receive the IP-MAC binding due to this reply.
4) Check the secondary cache whether the MAC address captured (in echo reply) is already mapped to any IP address.
5) If found no such entry, accept the IP-MAC binding and store it in secondary cache. Also, update the primary cache with static entry of 10 minutes for this binding and go to Step 2 to send echo requests to other hosts.
6) If MAC address is already bound to an IP address, send echo request destined to the IP address bound with this MAC address in the secondary cache.
   a) If echo reply comes back, ignore the newer binding which is received in 3rd Step, delete this binding from primary cache and update it statically with the older binding for 10 minutes. Go to Step 2.
   b) If no echo reply comes back, remove the older binding, accept the newer one and update it in secondary cache. Also make this entry static in primary cache for 10 minutes. Go to Step 2.

Initially, the entering host sends a broadcasted ARP reply so that every existing host receive it and after proper validation using secondary cache look up, store this new binding in their secondary cache.

B. Algorithm for existing hosts

1) When receive an ARP Reply (IP-MAC binding), check the secondary cache whether the MAC address or IP address are already mapped to any other IP address or MAC address respectively.
2) If IP address is not bound to any other MAC address, go to Step 5.
3) If IP address is already bound to same MAC address which is received in ARP Reply, just ignore this reply and go to Step 1.
4) If IP address is already bound to a MAC address which is not same as received in ARP Reply, send an echo request destined to this IP address with the bound destination MAC address found in this secondary cache entry.
   a) If echo reply comes back, ignore the new IP-MAC binding received in step 1, delete it from primary cache and update it statically for 10 minutes with the binding available in secondary cache. Go to Step 1.
   b) If no reply comes back, delete the older entry, accept the newer one and update it in secondary cache. Also make this entry static in primary cache for 10 minutes. Go to Step 1.
5) If MAC address is not bound to any other IP address, accept the IP-MAC binding and store it in secondary cache. Also, update the primary cache with static entry of 10 minutes for this binding and go to Step 1.
6) If MAC address is already bound to same IP address which is received in ARP Reply, just ignore this reply and go to Step 1.
7) If MAC address is already bound to an IP address which is not same as received in ARP reply, send an echo request destined to the IP address bound with this MAC address in the secondary cache.
   a) If echo reply comes back, ignore the newer IP-MAC binding received in step 1, delete it from primary cache and update it statically for 10 minutes with the binding available in secondary cache. Go to Step 1.
   b) If no reply comes back, remove the older binding, accept the newer one and update it in secondary cache. Also make this entry static in primary cache for 10 minutes. Go to Step 1.

C. Attack Scenarios

We assume different attack scenarios in which attack is possible. The attack scenario is divided into four situations.

1) *Attacker as an entering host*: This situation arises when attacker, C, enters into a network while other hosts, A and B, are already present in the network. In this case, if C sends to A an ARP Reply with C's MAC address (M_C) bound to B's IP address (IP_B), A will check its secondary cache to see if there is any entry related to M_C or IP_B. Since A and B were present in the network before C, they are having these entries. As a result, A will ignore this ARP Reply. Thus, the ARP poisoning tried by C will not be successful.

2) *Attacker as an existing host:* This situation will arise when C is already present in the network before another host, A, which is yet to come into the network. Here, two cases arise. The first case happens when C sends such fake ARP replies to A so that A will consider C as Default Gateway (DG). The second case happens when C sends such fake ARP replies to A so that A will consider C as another host, B. Both the cases are discussed in the next two points.

3) *Attack against DG and host A:* When A enters into the network, C may send fake ARP replies to A with C's MAC address (M_C) bound to DG's IP address (IP_{DG}). This causes A to consider that C is DG for the subnet. This is possible because being an entering host, initially, A's secondary cache is not having any entry at all. However, C will not be able to poison DG's cache because DG was already present in the network before C and thus, DG will be having an entry for M_C bound to C's IP address. As a result, C will not be able to get the MITM position.

4) *Attack against two hosts A and B:* If C wants to get MITM position between two hosts A and B where A is the entering host, C can poison A's cache by claiming itself as B. But C will not be able to poison B's cache because B may be present in the network before C, so B will quickly receive IP-MAC binding of A when A will enter into the network. If B also comes after C, C will not be able to predict the IP address which B will be going to get when it will enter into the network. Thus, C cannot get the MITM position in this case also.

V. Experimental Results

We implemented the algorithm on Debian based *Kali Linux* [13] operating system. We used a command line interface program *PackEth* [14] for generating customized packets. Also, we made use of raw socket programming and Bash scripting wherever required. We performed this experiment on a network having three hosts. We are implementing the 1st attack scenario which we discussed earlier. The IP and MAC address of these hosts are shown in the table I. The primary ARP cache of victim (A) is shown in the Fig. 4 before the attack is launched by attacker (C).

When C sent a spoofed ARP reply to A, the IP-MAC binding in A's primary cache got poisoned. Attacker C, 172.16.7.152, sent an ARP reply packet to victim A, 172.16.7.159 with IP-MAC binding 172.16.7.52 is at 00:24:8c:81:9c:db.

In normal network, the IP-MAC binding at victim host changed to 172.16.7.52—00:24:8c:81:9c:db. A's primary cache is shown in Fig. 5 after being poisoned.

After implementing the algorithm, when A received the ARP Reply, it checked its secondary cache. It found out that there is already an entry for 172.16.7.52. This is shown in Fig. 6. So, it sent an echo request to the old binding present in secondary cache. Since it received the echo reply back, it came to know that older binding already exists. As a result, it ignored the new binding and deleted it from the primary cache and updated it statically with the binding available in secondary cache for 10 minutes. The ICMP echo request with B's IP-MAC binding is sent and the resulting echo reply is shown in Fig. 8 using *Wireshark* [15]. The echo request was sent using PackEth as shown in Fig. 7. The final primary cache of A with static entry is shown in Fig. 9.

TABLE I
IP Address and MAC Address of Different Hosts

Host	IP Address	MAC Address
Gateway(B)	172.16.7.52	6c:20:56:dc:ac:43
Victim(A)	172.16.7.159	00:26:22:04:50:2a
Attacker(C)	172.16.7.152	00:24:8c:81:9c:db

Fig. 4. A's ARP cache before attack.

Fig. 5. A's ARP cache after attack.

Fig. 6. Detection of fake ARP reply at A.

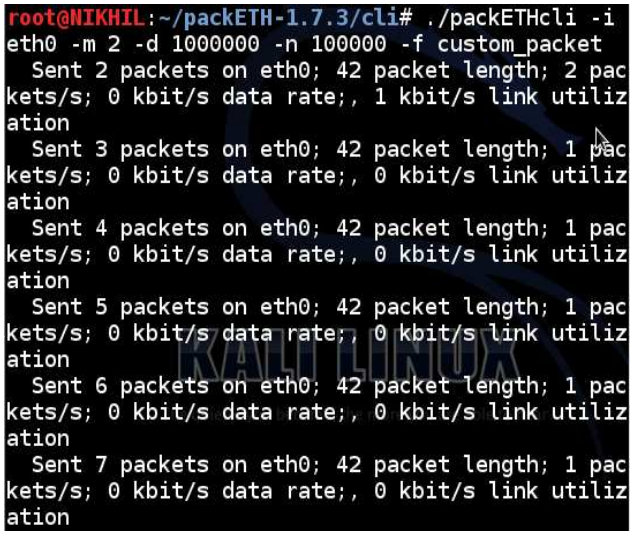

Fig. 7. Echo requests sent according to the old binding using PackEth

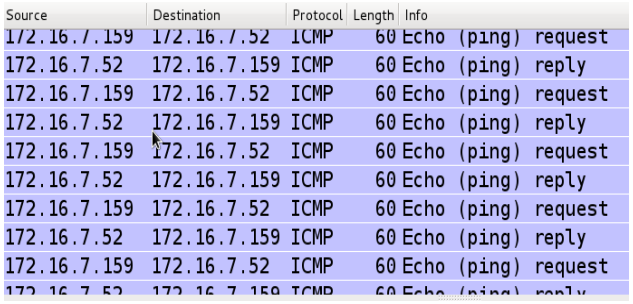

Fig. 8. Echo requests and replies

Fig. 9. A's final ARP cache with static entry

VI. CONCLUSION

The solution proposed in this paper can be a possible solution for the ARP poisoning as well as IP exhaustion problem. It makes use of secondary cache in which entries are made and removed using ICMP messages. This cache helps to validate the entries present in primary cache. Thus, it prevents the rogue and inconsistent multiple entries in secondary as well as primary cache. As a result, ARP poisoning is not possible. Since there is only one entry for one MAC addresses, no MAC address can have more than one IP address. This enables the solution to prevent the IP exhaustion problem as well.

The secondary cache can just be a simple text file and all the elements used in the solution can work with the existing ARP. This makes this solution backward compatible with the existing networks. Though the number of message exchanges increased due to the algorithm, the solution is very cost effective for any privacy critical application.

REFERENCES

[1] David C. Plummer, An Ethernet Address Resolution Protocol, RFC826, Internet Engineering Task Force, November 1982.

[2] F. Callegati, W. Cerroni, and M. Ramilli, "Man-in-the-middle attack to the https protocol," Security & Privacy, IEEE, vol. 7, no. 1, pp. 78–81, 2009.

[3] Somnuk Puangpronpitag, Narongrit Masusai, "An Efficient and Feasible Solution to ARP Spoof Problem", 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. ISBN: 978-1-4244-3387-2.

[4] X. Hou, Z. Jiang, and X. Tian, "The detection and prevention for arp spoofing based on snort," in Computer Application and System Modeling (ICCASM), 2010 International Conference on, vol. 5. IEEE, 2010, pp. V5–137.

[5] T. Chomsiri, "Sniffing packets on LAN without ARP spoofing," in Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on, vol. 2. IEEE, 2008, pp. 472–477.

[6] D. Bruschi, A. Ornaghi, and E. Rosti, "S-arp: a secure address resolution protocol," in Computer Security Applications Conference, 2003. Proceedings. 19th Annual. IEEE, 2003, pp. 66–74.

[7] W. Lootah, W. Enck, and P. McDaniel, "Tarp: Ticket based address resolution protocol," vol. 51, no. 15. Elsevier, 2007, pp. 4322–4337.

[8] S. Nam, D. Kim, and J. Kim, "Enhanced ARP: preventing ARP poisoning based man-in-the-middle attacks," Communications Letters, IEEE, vol. 14, no. 2, pp. 187–189, 2010.

[9] I. Teterin, "Antidote," 2002. Available at: http://online.securityfocus.com/archive/1/299929.

[10] S. Kumar, S. Tapaswi, "A centralized detection and prevention technique against ARP poisoning," Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on, Publication Year: 2012 , Page(s): 259 – 264.

[11] G. Nath Nayak and S. Ghosh Samaddar, "Different flavors of man-in- the-middle attack, consequences and feasible solutions," in Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, vol. 5. IEEE, 2010, pp. 491–495.

[12] Gao Jinhua and Xia Kejian, "ARP spoofing detection algorithm using ICMP protocol," in Computer Communication and Informatics (ICCCI), 2013 International Conference on, IEEE, Publication Year: 2013, Page(s): 1-6

[13] Available at: http://www.kali.org

[14] Available at: http://packeth.sourceforge.net/packeth/Home.html

[15] Available at http://www.wireshark.org/