



ARP Spoofing Detection Algorithm: An Effective Approach

Goldendeep Kaur, Dr. Jyoteesh Malhotra
Computer Science & Engineering Department
Guru Nanak Dev University, Regional Campus, Jalandhar, Punjab, INDIA

Abstract: *With the expeditious evolution of computer network, there are increased network security exploits because of the defaults of network protocols. ARP spoofing is an appalling one that makes use of defaults of ARP protocols to attack the network and hence destroys the communication between hosts by sending wrong IP/MAC address. There are various security measures that are used to prevent ARP spoofing, but each has some constraints. This paper proposed an efficient algorithm to detect ARP poisoning by performing reverse ARP spoofing on the attacker. We inject ICMP echo requests on the network to probe for conflicts. This approach is faster, perceptive, adaptable and reliable in detecting attacks. It can also detect the real mapping during an attack.*

Keywords: ARP Protocol; ARP Cache; ARP Spoofing Attack; Reverse ARP Poisoning, Active IP Probing

I. Introduction

Every node on an IP/Ethernet network has two types of addresses. The first is the hardware address known as Media Access Control (MAC) address and the second is its IP address. Applications, which are above the layer four, use logical address to identify the destination host, i.e. IP address. IP addresses are assigned to the hosts and are logically independent of the physical address. The IP packets are enclosed into frames. The delivery of frames across the links is based on local address i.e. physical address (MAC numbers). The association of IP addresses into physical addresses is done through Address Resolution Protocol (ARP) Address Resolution Protocol. So Address Resolution Protocol is a telecommunication protocol that is used for the resolution of network layer addresses into link layer addresses. It accomplishes this task by building a correspondence table of IP and MAC addresses, using specialized packets broadcast on the local network. ARP operates below the network layer as a part of the Open Systems Interconnection (OSI) link layer and is used when IP is used over the Ethernet. It is different than other protocols in the TCP/IP suite. Rather than being a peer to peer protocol, it is an interface between the IP(a Layer three protocol) and an underlying Layer two protocol, on which it depends upon for transport of datagrams . [1] When a source device wants to communicate with another device, it checks its Address Resolution Protocol (ARP) cache to find if it already has a resolved MAC Address of the destination device. If there, it uses that address for communication. If ARP resolution is not there in its local cache, the source machine will generate an Address Resolution Protocol (ARP) request message with its data link layer address as the Sender Hardware Address and its own IPv4 Address as the Sender Protocol Address. It fills the target IPv4 Address as the Target Protocol Address. The destination MAC Address will be left blank, since the device is trying to find that. The source broadcasts the Address Resolution Protocol (ARP) request message to the local network. The message is received by each device on the LAN since it is a broadcast request. Each machine compares the Target Protocol Address (IPv4 Address of the machine to which the source is trying to communicate) with its own Protocol Address (IPv4 Address). Those devices whose IP address does not match will drop the packet without any action. When the targeted device checks the Target Protocol Address, it will obtain a match and will construct an Address Resolution Protocol (ARP) reply message. It takes the Sender MAC Address and the Sender Protocol Address fields from the Address Resolution Protocol (ARP) request message and uses these values for the Targeted Hardware Address and Targeted Protocol Address of the reply message. The destination device will update its Address Resolution Protocol (ARP) cache, since it needs to contact the sender machine soon. Destination device sends the Address Resolution Protocol (ARP) reply message and it is NOT a broadcast, but a unicast. The source machine will process the Address Resolution Protocol (ARP) reply from destination and stores the Sender Hardware Address as the layer 2 address of the destination. The source machine updates its Address Resolution Protocol (ARP) cache with the Sender Hardware Address and Sender Protocol Address it receives from the Address Resolution Protocol (ARP) reply message. [2]

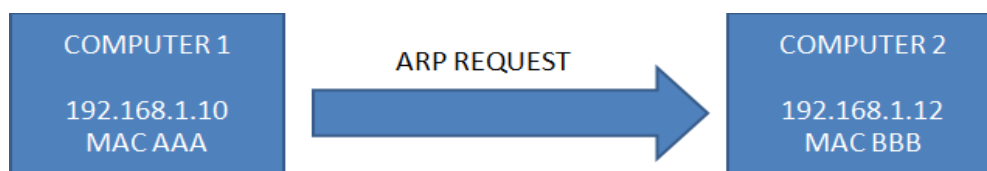


Figure 1: Address Resolution Protocol

As ARP is a stateless protocol which means that it treats each request as an independent transaction, so most operating systems will automatically cache the ARP replies, inconsiderate of whether they have sent out an actual request. Since ARP does not offer any method for authenticating ARP replies in the network, ARP replies are vulnerable to be spoofed by other hosts on a network other than the one from which a response is expected. So this paper focuses on new proposed solution to detect ARP spoofing. It is able to detect both the attack as well as the attacker trying to perform the attack.

The remainder section of this paper is organized as follows: Section II describes defence strategies, Section III describes the proposed algorithm, Section IV describes the comparison of the proposed solution with existing techniques, and finally Section V concludes the paper.

II. Defence Strategies

In order to have secure communications, we should be able to detect and mitigate these attacks. “Security is a process not a product.” ARP Cache poisoning on the network can go completely undetected if appropriate measures are not taken. It has been observed that there is no universal defence against these attacks. In fact one of the simplest methods is to use static ARP entries. As static entries cannot be updated, spoofed ARP replies can be ignored. But this method is not suitable practically for large networks to manually add each entry into the cache. Free Intrusion Detection Systems like Snort [6], ARPWatch [5], XArp are working on detection mechanism but not able to provide complete defence. Kernel based patches like Anticap [8] and Antidote [10] prevents ARP poisoning by rejecting the ARP replies that contains a MAC address different from the current entry in the cache for same IP address. But this solution is also available for a limited number of specific kernels. On the other hand, Port Security detects MAC cloning but does not able to prevent ARP Spoofing. Some High end Cisco switches proposed a feature known as Dynamic ARP Inspection [4] that allows the switch to block invalid <IP, MAC> pairings. It uses local pairing table that is built using a feature known as DHCP snooping to detect which pairings are invalid. But the high cost of switches makes this feature ineffective. So no complete solution is devised so far for these attacks. This current scenario led to the development of this new algorithm. The algorithm presented here detects the source of ARP poisoning and raise alarm to the network administrators.

III. Proposed Algorithm

The proposed algorithm has three phases. The first phase is the process of filtering the packets where rules are applied from the rule base on the incoming packets and the packets are captured from the network traffic. If there is any mismatch that contradicts with the rule base then alarm is raised to the network administrator thereby indicating ARP poisoning. The second phase is the Reverse ARP Poisoning phase which involves poisoning the cache of attacker instead. Suppose there are 3 hosts. Host A and Host B are legitimate hosts and Host X is an attacker. Attacker sends ARP reply to Host A using HostB_IP and HostX_MAC. Then Host A will perform Reverse ARP poisoning to poison the cache of attacker. Host A will send ARP reply to Host X containing mapping between HostB_IP and HostA_MAC thereby poisoning the cache of attacker.

The third phase is the Active IP Probing that attempts to find the real and mitm hosts. As Host A poison the cache of attacker i.e. Host X. Now Active IP probing is performed by the Host A to differentiate between real and mitm hosts by sending ICMP request to HostB_IP and HostX_MAC. Since Host X is attacker and will try to forward this probe request to Host B. But as its cache is already poisoned by Host A, so it attempts to forward this request to HostA_MAC thereby indicating that Reverse ARP poisoning is successful and Host X is attacker. In this way this proposed solution detects the ARP poisoning as well as the attacker.

The working of the proposed system is based on capturing the ARP packets. If the ARP packet received is an unknown packet then we add this newly seen <IP,MAC> pairing to an array named “CurrArray” otherwise if it is already known mapping then add it to database. Now the next packet is captured and if it is also an unknown mapping, we add this mapping to new array named “NextArray”. These captured packets are then compared. If the MAC address of both unknown bindings is same, then check if their IP addresses are different. If they have different IP addresses but same MAC address then it is an indication of ARP spoofing and an alarm is raised to Network administrator. Now we have to find the attacker’s IP and MAC address. We inject ICMP echo requests to both the mappings and wait for their reply. If single ARP reply is received then probably it is the real host and we will add it to database. If more than one ARP replies are received this means one is the real host and other one is attacker. After that we will differentiate between the real and mitm host by applying reverse ARP poisoning on the attacker. In this way we can get the attacker’s IP and MAC address

The pseudo code of the proposed algorithm is as follows:

Input: ARP Packets.

Output: Identifying Real and Mitm Hosts.

- Capture Packets.
- Split ARP packets i.e. <IP,MAC> Status.
- If (ARP type= Unknown)
- Add <IP,MAC> to CurrArray
- EndIf
- Loop
- Split Next ARP packet i.e. <IP,MAC> Status

- If (NextARP type= Unknown)
- Add <IP,MAC> to NextArray
- If (CurrMAC==NextMAC)
- If (CurrIP!= NextIP)
- Raise Alarm
- EndIf

- EndIf
- EndIf
- Detecting false binding
- Send ICMP packet to the source of each ARP reply
- If (Single ARP reply)

Probably Real Host

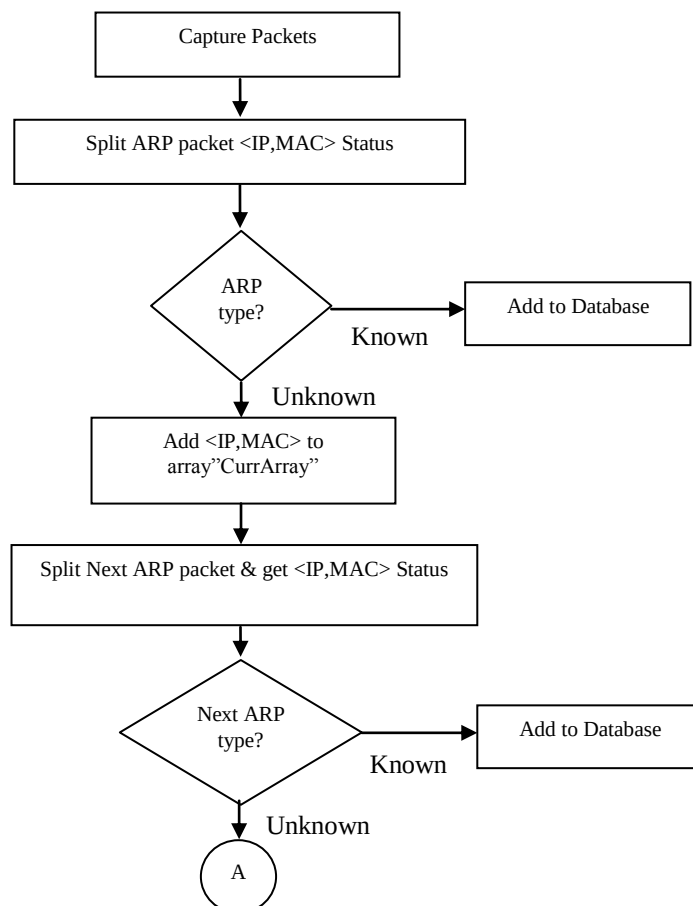
Add it to database

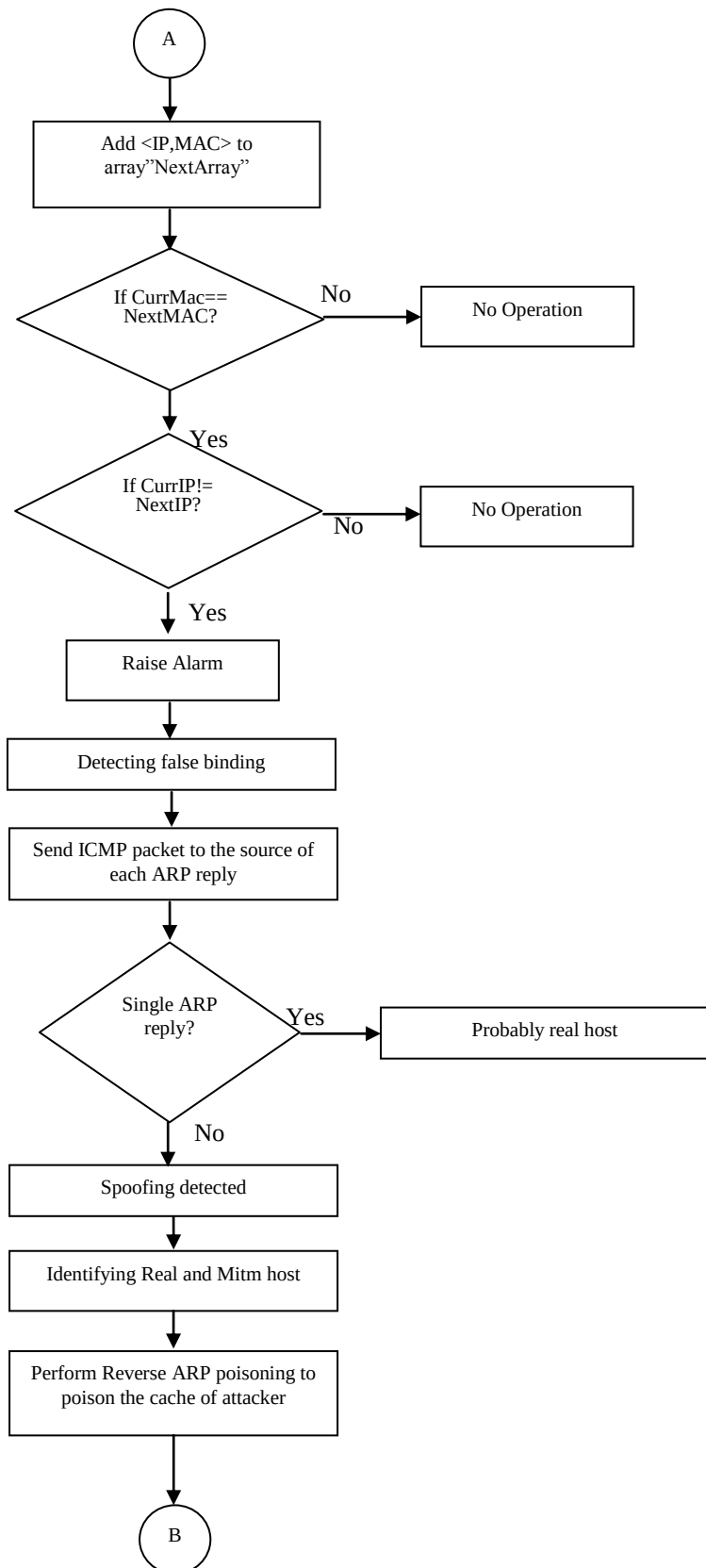
- EndIf
- If (More than one ARP reply)

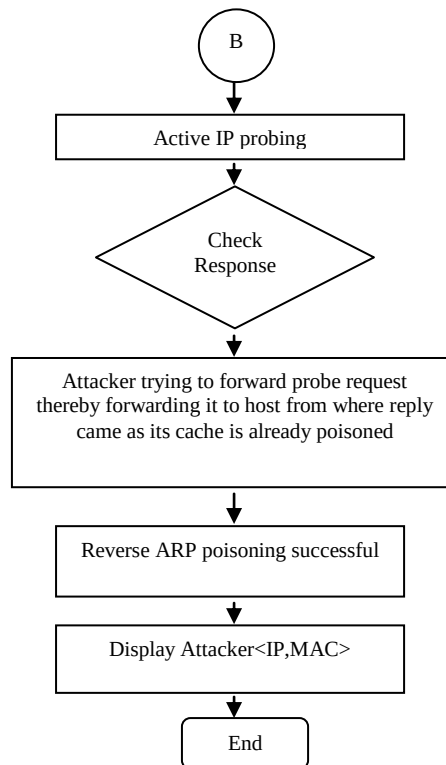
Spoofing Detected

- EndIf
- Identifying Real and Mitm Hosts
- Perform Reverse ARP poisoning to poison the cache of attacking host
- Perform IP probing
- Check Response
- Attacker will try to forward this request to host from which it came as attacker's cache is already poisoned.
- Reverse ARP poisoning successful.
- Attacker detected.
- Display attacker's <IP,MAC>.

The flowchart representation of the algorithm is shown as follows:







IV. Comparison with Existing Techniques

It has been observed that our technique is expeditious and reliable due to the following reasons:

- **Scalability:** It can be used in large networks to detect ARP poisoning attacks because the time lag between ARP attack and its detection is very less. So it is better than the existing passive techniques.
- **Network Overhead:** It does not blindly add the unknown traffic in the ARP cache but firstly verifies the authenticity of the unknown ARP packets by sending one ICMP echo requests per newly seen packet.
- **Efficiency:** It also differentiates between the real and mitm hosts during an actual attack.
- **Compatibility:** Our solution is backward compatible with Address Resolution Protocol and can be easily matched to run on dynamic environments.

V. Conclusion

In this paper, we proposed a new comprehensive approach to overcome the limitations faced by the current schemes. It is safe to conclude that improvements need to be made in the existing techniques to mitigate these attacks and our technique is scalable and perceptive in detecting ARP poisoning attacks. Moreover we inject single ICMP echo request for each newly seen packet to probe the authenticity of unknown traffic. It is an active approach as the time lag between the attack and its detection is very less. It can also detect the real mapping during an actual attack. In the end, as a future work we want to extend our work through accomplishing experimental observations to include real time scenarios of ARP spoofing attacks.

VI. References

- [1] S. Whalen, "An introduction to ARP spoofing," 2600: The Hacker Quarterly, vol. 18, no. 3, Fall 2001, Available: http://servv89pn0aj.sn.sourcedns.com/_g_bpprorg/2600/arp_spoofing_intro.pdf
- [2] D. Plummer. An ethernet address resolution protocol, Nov.2010. RFC 826.
- [3] M. Carnut and J. Gondim. ARP spoofing detection on switched Ethernet networks: A feasibility study. In *Proceedings of the 5th Simp'osio Seguranc,a em Inform'atica*, Nov.2010.
- [4] Cisco Systems. *Configuring Dynamic ARP Inspection*, chapter 39, pages 39:1–39:22. 2012. Catalyst 6500 Series Switch Cisco IOS Software Configuration Guide, Release 12.2SX
- [5] L. N. R. Group. arpwatc, the Ethernet monitor program; for keeping track of ethernet/ip address pairings. (Last accessed April 17, 2012).
- [6] Snort Project, The. Snort: The open source network intrusion detection system. <<http://www.snort.org>>.
- [8] "ARP-Guard," (accessed 28-July-2013). [Online]. Available: <http://www.arp-guard.com>.
- [7] Zouheir Trabelsi and Khaled Shuaib. Spoofed ARP Packets Detection in Switched LAN Networks. J. Filipe and M.S. Obaidat (Eds.): ICETE 2013, CCIS 9, pp. 81–91
- [8] M. Barnaba, "anticap", (accessed 17 April 2013) Online. Available: <http://www.antifork.org/anticap>.
- [9] V. Goyal and V. Abraham "An efficient Solution to the ARP cache poisoning problem", in *Proceedings of 10th Australasian Conference on Information Security and Privacy*, Jul 2013, pp 40-51.
- [10] I. Teterin, Antidote, SecurityFocus, <http://online.securityfocus.com/archive/1/299929>, last accessed, Apr. 2012.
- [11] M. Gouda and C.-T. Huang. A secure address resolution protocol. *Computer Networks*, 41(1):57–71, Jan. 2012.