An Active Intrusion Detection System for LAN Specific Attacks

N Hubballi, S Roopa, R. Ratti, F.A. Barbhuiya, S Biswas, A Sur, S Nandi *, and V Ramachandran

Department of Computer Science and Engineering
Indian Institute of Technology
Guwahati, India - 781039
{neminath, roopa.s, r.ratti, ferdous, santosh_biswas, arijit, sukumar}@iitg.ernet.in, vivek.securitywizard@gmail.com
http://www.iitg.ernet.in

Abstract. Local Area Network (LAN) based attacks are due to compromised hosts in the network and mainly involve spoofing with falsified IP-MAC pairs. Since Address Resolution Protocol (ARP) is a stateless protocol such attacks are possible. Several schemes have been proposed in the literature to circumvent these attacks, however, these techniques either make IP-MAC pairing static, modify the existing ARP, patch operating systems of all the hosts etc. In this paper we propose an Intrusion Detection System (IDS) for LAN specific attacks without any extra constraint like static IP-MAC, changing the ARP etc. The proposed IDS is an active detection mechanism where every pair of IP-MAC are validated by a probing technique. The scheme is successfully validated in a test bed and results also illustrate that the proposed technique minimally adds to the network traffic.

Key words: LAN Attack, Address Resolution Protocol, Intrusion Detection System

1 Introduction

The security and performance considerations in any organization with sizeable number of computers lead to creation of LANs. A LAN is a high-speed communication system designed to link computers and other data processing devices together within a small geographic area, such as department or a building. Security threat to any computer, based on LAN specific attack is always from a compromised machine. The basic step involved in most of these attacks comprise cache poisoning with falsified IP-MAC pairs which may then lead to other attacks namely, man in the middle, denial of service etc [1]. ARP is used by hosts in a LAN to map network addresses (IP) to physical addresses (MAC). ARP is a stateless protocol and so when an ARP reply is received, the host

* The work reported in this paper is a part of the project "Design, Development and Verification of Network Specific IDS using Failure Detection and Diagnosis of DES", Sponsored by Department of Information Technology, New Delhi, INDIA.

updates its ARP cache without verifying the genuineness of the IP-MAC pair of the source [1].

There are number of solutions proposed in the literature to detect, mitigate and prevent such attacks. The schemes can be broadly classified as:

Static ARP Entries[2]: The most foolproof way to prevent ARP attacks is to manually assign static IPs to all systems and maintain the static IP-MAC pairings at all the systems. However, in a dynamic environment this is not a practical solution.

Security Features[3]: One possible action to combat ARP attacks is enabling port security (CIS) on the switch. This is a feature available in high-end switches which tie a physical port to a MAC address. These port-address associations are stored in Content Addressable Memory (CAM) tables. A change in the transmitter's MAC address can result in port shutdown or ignoring the change. The problem with this approach is, if the first sent packet itself is having a spoofed MAC address then the whole system fails. Further, any genuine change in IP-MAC pair will be discarded (e.g., when notified by Gratuitous request and reply).

Software based solutions: The basic notion of port security involving observation of changes in IP-MAC pairs in switches have also been utilized in software solutions namely, ARPWATCH [4], ARPDEFENDER [5], COLASOFT-CAPSA [6]. These software solutions are cheaper than switches with port security but have slower response time compared to switches. Obviously, these tools suffer from the drawbacks as that of port security in switches.

Signature and anomaly based IDS: Signature based IDSs like Snort [7] can be used to detect ARP attacks and inform the administrator with an alarm. The main problem with IDSs is that they tend to generate a high number of false positives. Furthermore, ability of IDSs to detect all forms of ARP related attacks is limited [8]. Recently, Hsiao et al. [9], have proposed an anomaly IDS to detect ARP attacks based on SNMP statistics. A set of features are extracted from SNMP data and data mining algorithms such as decision tree, support vector machines and bays classifier have been applied to classify attack data from normal data. Reported results show that false negative rates are as high as 40%.

Modifying ARP using cryptographic techniques: Several cryptography based techniques have been proposed to prevent ARP attacks namely S-ARP[10], TARP [11]. Addition of cryptographic features in ARP may lead to performance penalty [8]. Also, it calls for upgradation of network stacks of all the hosts in the LAN, which makes the solution nonsalable.

Active techniques for detecting ARP attacks: The IDS in active detection of ARP attacks, sends probe packets to systems in the LAN in addition to observations in changes of IP-MAC pairs.

In [12], a database of known IP-MAC pairs is maintained and on detection of a change the new pair is actively verified by sending a probe with TCP SYN packet to the IP under question. The genuine system will respond with a SYN/ACK or RST depending on whether the corresponding port is open or closed. While this scheme can validate the genuineness of IP-MAC pairs, it violates the network layering architecture. Moreover it is able to detect only ARP spoofing attacks.

An active scheme for detecting MiTM attack is proposed in [13]. The scheme assumes that any attacker involved in MiTM must have IP forwarding enabled. First, all systems with IP forwarding are detected (actively). Once all such systems are detected, the IDS attacks all such systems one at a time and poison their caches. The poisoning is done in a way such that all traffic being forwarded by the attacker reaches the IDS (instead of the system, the attacker with IP forwarding wants to send). So, the IDS can differentiate the real MiTM attackers from all systems with IP forwarding. There are several drawbacks in this approach, namely huge traffic in case of a large network with all machines having IP forwarding, assumption of successful cache poisoning of the machine involved in MiTM attack, cache poisoning (of the the machine involved in MiTM attack by IDS) exactly when the attack is going on etc.

So, from the review, it may be stated that an ARP attack preventation/detection scheme needs to have the following features

- Should not modify the standard ARP or violate layering architecture of network
- Should generate minimal extra traffic in the network
- Should not require patching, installation of extra softwares in all systems
- Should detect a large set of LAN based attacks
- Hardware cost of the scheme should not be high

In this paper we propose an active IDS for ARP related attacks. The technique involves installation of the IDS in just one system in the network, do not require changes in the standard ARP and do not violate the principles of layering structure as is the case with [12] (while sending active ARP probes). In addition, the IDS has no additional hardware requirements like switches with port security. Our proposed scheme detects all spoofing attempts and in addition, identify the MAC of the attacker in case of successful MiTM attacks.

Rest of the paper is organized as follows. In Section 2 we present the proposed approach. In Section 3 we discuss the test bed and experimental results. Finally we conclude in Section 4.

2 Proposed Scheme

In this section we discuss the proposed active intrusion detection scheme for ARP related attacks.

2.1 Assumptions

The following assumptions are made regarding the LAN

1. Non-compromised (i.e., genuine) hosts will send one response to an ARP request within a specific interval T_{req}.
2. IDS is running on a trusted machine with static IP-MAC. It has two network interfaces: one is used for data collection in the LAN through port mirroring and the other is exclusively used for sending/receiving ARP probes requests/replies.

2.2 Data Tables for the Scheme

Our proposed scheme ensures the genuineness of the IP-MAC pairing by an active verification mechanism. The scheme sends verification messages termed as probe requests upon receiving ARP requests and ARP replies. To assist in the probing and separating the genuine IP-MAC pairs with that of spoofed ones, we maintain some information obtained along with the probe requests, ARP requests and ARP replies in some data tables. The information and the data tables used are enumerated below. Henceforth in the discussion, we use the following short notations: IPS - Source IP Address, IPD - Destination IP Address, $MACS$ - Source MAC Address, $MACD$ - Destination MAC Address. Fields of any table would be represented by $\langle TableName \rangle_{\langle field \rangle}$; e.g., RQT_{IPS} represents the source IP filed of "Request table. Also, $\langle TableName \rangle_{MAX}$ represents the maximum elements in the table at a given time.

1. Every time an ARP request is sent from any host querying some MAC address, an entry is created into the "Request table (denoted as RQT)" with source IP (RQT_{IPS}), source MAC (RQT_{MACS}) and destination IP(RQT_{IPD}). Also the time τ when the request was received is recorded in the table as RQT_{τ}. Its entries timeout after T_{req} seconds. The value of T_{req} will depend on the ARP request-reply round trip time, which can be fixed after a series of experiments on the network. According to [14], the approximate ARP request-reply round trip time in a LAN is about 1.2 ms - 4.8 ms.
2. Every time an ARP reply is received from any host replying with the MAC address corresponding to some IP address, an entry is created in the "Response table (denoted as RST)" with source IP (RST_{IPS}), source MAC (RST_{MACS}), destination IP (RST_{IPD}) and destination MAC (RST_{MACD}). Also the time when the response was received is recorded in the table. Its entries timeout after T_{resp} seconds. The T_{resp} value can be determined based on the maximum ARP cache timeout value of all the hosts in the LAN.
3. When some IP-MAC pair is to be verified, an ARP probe is sent and response is verified. The probe is initiated by IDS, upon receiving either a Request or a Response. The source IP address and the source MAC address from the Request/ Response packets used for verification are stored in "Verification table (denoted as $VRFT$)". The entries in this table are source IP ($VRFT_{IPS}$) and source MAC ($VRFT_{MACS}$).
4. Every time any IP-MAC pair is verified and found to be correct, an entry is created for the pair in the "Authenticated bindings table (denoted as $AUTHT$)". There are two fields in this table, IP address ($AUTHT_{IP}$) and MAC address ($AUTHT_{MAC}$)
5. Every time a spoofing attempt or an unsolicited response is detected, an entry is created in the "Log table (denoted as LT)" with source IP(LT_{IPS}), source MAC (LT_{MACS}), destination IP(LT_{IPD}) and destination MAC (LT_{IPD}). Also the time when the spoof or unsolicited response was detected is recorded in the table as LT_{τ}. The table has same fields as that of the Response table.
6. "Unsolicited response table (denoted as $URSPT$)" is used for storing the number of unsolicited responses received by each host within a specified time interval (δ). Every time an Unsolicited response is received, an entry is created in the Unsolicited response table with destination IP ($URSPT_{IPD}$), time ($URSPT_{\tau}$)when

the unsolicited response was received and unsolicited response counter ($URSPT_{counter}$) for each IP.
In general, ARP replies are received corresponding to the ARP requests. If unsolicited responses are observed in the network traffic, it implies an attempt of ARP attack. On receiving any such unsolicited ARP reply by a host, the corresponding unsolicited response counter is incremented along with the time stamp (in the Unsolicited response table). The entries may timeout after $T_{unsolicit}$ which can be fixed after considering the maximum cache timeout period for all the hosts in the network. However, there is an exception to the fact that all unsolicited ARP replies are attempts for attack. Gratuitous ARP responses are unsolicited which are generated by systems at startup to notify the network, its IP-MAC address. Gratuitous ARP responses are not entered into the table and are handled separately.

The proposed attack detection technique has two main modules namely, ARP REQUEST-HANDLER() and ARP RESPONSE-HANDLER(). These are elaborated in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 process all the request ARP packets in the network. For any ARP request packet RQP, it first checks if its is malformed (i.e., any changes in the immutable fields of the ARP packer header or different MAC addresses in the MAC and ARP header field) or unicast; if so, a status flag is set accordingly and stops further processing of this packet. If the packet is not unicast or malformed, but a request packet from (IDS) i.e., RQP_{IPS} is IP of IDS and RQP_{MACS} is MAC of IDS, Algorithm 1 skips processing of this packet; we do not consider ARP request from IDS as we assume that IP-MAC pairing of the IDS is known and validated. If the ARP request is not from IDS, the source IP (RQP_{IPS}), source MAC (RQP_{MACS}), destination IP (RQP_{IPD}) and time τ when the request packet is received is recorded in Request table.

The algorithm next finds whether the packet received is a Gratuitous ARP request and the status flag is set accordingly. Gratuitous ARP request can be determined if $RQP_{IPS} == RQP_{IPD}$. For such Gratuitous ARP request, ARP probe is sent for checking the correctness of the IP-MAC pair. Hence, the VERIFY IP-MAC() module is called for RQP along with τ (the time information when RQP was received).

If none of the above cases are matched, then RQP_{IPS} is searched in the Authenticated bindings table. If a match is found as $AUTHT_{IPS}[i]$(where i is the i^{th} entry in the $AUTHT$) and the corresponding MAC address $AUTHT_{MACS}[i]$ in the table is same as RQP_{MACS}, the packet has genuine IP-MAC pair which is already recorded in the Authenticated bindings table. In case of a mismatch in the MAC address (i.e., $RQP_{MACS} \neq AUTHT_{MACS}[i]$) the packet is spoofed with a wrong MAC address and hence the status flag is set as spoofed. Also, the details regarding the non-genuine RQP is stored in the Log table. It may be noted that this checking of spoofing could be done without ARP probe thereby reducing ARP traffic for verification.

Also, it may be the case that IP-MAC pair given in RQP_{IPS} is not verified as yet and no match can be found in Authenticated bindings table. In such a case, ARP probe is to be sent by IDS to RQP_{IPS} and RQP_{MACS} for verifying the correctness of RQP_{IPS}-RQP_{MACS} pair. This is handled by the VERIFY IP-MAC() module with RQP and τ as parameters.

Algorithm 1: ARP REQUEST HANDLER
Input : RQP - ARP request packet, τ - time at which RQP was received, Request table, Verification table, Authenticated bindings table
Output: Updated Request table, Updated Log table, Status

1: **if** (RQP is $malformed$) **then**
2: Status=$malformed$
3: **else if** (RQP is $Unicast$) **then**
4: Status=$Unicast$
5: **else if** ($RQP_{IPS} == IP(IDS)$ && $RQP_{MACS} == MAC(IDS)$) **then**
6: EXIT
7: **else**
8: ADD RQP_{IPS}, RQP_{MACS}, RQP_{IPD} and τ to the Request table
9: **if** ($RQP_{IPS} == RQP_{IPD}$) **then**
10: Status=Gratutious Packet
11: VERIFY IP-MAC(RQP, τ)
12: **else**
13: **if** (RQP_{IPS} == $AUTHT_{IPS}[i]$ (for some i, $1 \leq i \leq AUTHT_{MAX}$) **then**
14: **if** ($RQP_{MACS} == AUTHT_{MACS}[i]$) **then**
15: Status= $Genuine$
16: **else**
17: Status=$Spoofed$
18: ADD RQP_{IPS}, RQP_{MACS}, RQP_{IPD}, $NULL$, and τ to the Log table
19: **end if**
20: **else**
21: VERIFY IP-MAC(RQP, τ)
22: **end if**
23: **end if**
24: **end if**

Algorithm 2 is an ARP response handler. For any ARP reply packet RSP, the algorithm determines whether the reply is malformed; if malformed, a status flag is set accordingly and the next packet is processed. Otherwise, the source IP (RSP_{IPS}), source MAC (RSP_{MACS}), destination IP (RSP_{IPD}) and timestamp τ of the received packet are recorded in Response table. Next, it verifies whether the packet is a Gratuitous ARP reply by checking if $RSP_{IPS} == RSP_{IPD}$. For such Gratuitous ARP reply, ARP probe is sent to check the correctness of the IP-MAC pair. Hence, the VERIFY IP-MAC() module is called.
If the reply packet is not Gratuitous, next it verifies if it is a reply for any ARP probe sent by the VERIFY IP-MAC() module (i.e., ARP probe by IDS). The response for the ARP probe can be determined if RSP_{IPD} == $IP(IDS)$ and RSP_{IPS} has an entry in the Verification table. For such such response packets, Algorithm 2 calls SPOOF-DETECROR() module.

If none of the above cases holds, the reply packet is then matched for a corresponding request in the Request table, using its source IP. If a match is found (i.e., RSP_{IPS}== $RQT_{IPD}[i]$), the RSP_{IPS} is searched in the Authenticated bindings ta-

ble. If a match is found and the corresponding MAC address in the table is same as RSP_{MACS}, the packet has genuine IP-MAC pair (which is already recorded in the Authenticated bindings table). In case of a mismatch in the MAC address (i.e., $RSP_{MACS} \neq AUTHT_{MACS}[j]$) the packet may be spoofed with a wrong MAC address and hence the status flag is set as spoofed. Also, the details regarding the non-genuine RSP is stored in the Log table. If the RSP_{IPS} is not present in the Authenticated bindings table, then an ARP probe is sent for verification by the VERIFY IP-MAC() module.

If there was no corresponding request for the response packet in the Request table, then it is an unsolicited response packet. Hence, the UNSOLICITED-RESPONSE-HANDLER() is called with the destination IP of the RSP and τ. Also this entry is added into Log table in order to check the MiTM attempts.

Algorithm 2: ARP RESPONSE HANDLER
Input : RSP - ARP response packet, τ - time at which RSP was received, Request table, Verification table, Authenticated bindings table
Output: Updated Response table, Updated Log table, Status

1: **if** RSP is $malformed$ **then**
2: Status= $malformed$
3: **else**
4: Add RSP_{IPS}, RSP_{MACS}, RSP_{IPD}, RSP_{MACD} and τ to Response table
5: **if** ($RSP_{IPS} == RSP_{IPD}$) **then**
6: Status= $Gratuitous$
7: VERIFY IP-MAC(RSP, τ)
8: **else**
9: **if** ($RSP_{IPD} == IP(IDS)$ && ($RSP_{IPS} == VRFT_{IPS}[k]$))(for some k, $1 \leq k \leq VRFT_{MAX}$)) **then**
10: EXIT
11: **else**
12: **if** ($RSP_{IPS} == RQT_{IPD}[i]$ (for some i, $1 \leq i \leq RQ_{MAX}$)) **then**
13: **if** ($RSP_{IPS} == AUTHT_{IPS}[j]$ (for some j, $1 \leq j \leq AUTHT_{MAX}$)) **then**
14: **if** ($RSP_{MACS} == AUTHT_{MACS}[j]$) **then**
15: Status= $Genuine$
16: **else**
17: Status=$Spoofed$
18: Add RSP_{IPS}, RSP_{MACS}, RSP_{IPD}, RSP_{MACD} and τ to Log table
19: **end if**
20: **else**
21: VERIFY IP-MAC(RSP, τ)
22: **end if**
23: **else**
24: Add RSP_{IPS}, RSP_{MACS}, RSP_{IPD}, RSP_{MACD} and τ to Log table

25: UNSOLICITED-RESPONSE-HANDLER(RSP_{IPD}, τ)
26: **end if**
27: **end if**
28: **end if**
29: **end if**

The main modules discussed in Algorithms1 and Algorithm 2 are assisted by three sub-modules namely, VERIFY IP-MAC(), SPOOF-DETECTOR() and UNSOLICITED-RESPONSE-HANDLER(). Now, we discuss these sub-modules in detail.

VERIFY IP-MAC() (Algorithm 3) sends ARP probes to verify the correctness of the IP-MAC pair given in the source of the request packet RQP or response packet RSP. Every time a probe is sent, its record is inserted in Verification table. Before, sending the ARP probe request, we need to verify if there is already such a request made by the IDS and response is awaited. This can be verified by checking IP and MAC in the Verification table; if a match pair is found the module is exited. A spoofing may be attempted if IP matches the entry in the Verification table but MAC does not; in this case, the status is set as spoofed and Log table is updated. This checking in the Verification table (before sending probe) limits the number of ARP probes to be sent for any known falsified IP-MAC address, thereby lowering extra ARP traffic. If the corresponding IP address is not found in the Verification table, a probe request is sent and the algorithm adds the IP and the MAC into the Verification table. At the same time SPOOF-DETECTOR() module is called which waits for a round trip time and analyzes all entries in the Response table collected during the round trip time (as replies against the probe).

Algorithm 3: VERIFY IP-MAC
Input : RP- ARP request/reply packet, τ - time of arrival of RSP, Verification table
Output: Updated Verification table, Status

1: **if** ($RP_{IPS} \in VRFT_{IPS}[i]$) (for some i, $0 \leq i \leq VRFT_{MAX}$) **then**
2: **if** ($RP_{MACS} == VRFT_{MACS}[i]$) **then**
3: EXIT
4: **else**
5: Status=$Spoofed$
6: Add RP_{IPS}, RP_{MACS}, RP_{IPD}, RP_{MACD} and τ to Log table
7: **end if**
8: **else**
9: Send ARP $Probe\ Request$ to RP_{IPS}
10: Add RP_{IPS} and RP_{MACS} to the Verification table
11: SPOOF-DETECTOR(RP, τ)
12: **end if**

SPOOF-DETECTOR() (Algorithm 4) is called from VERIFY IP-MAC() after sending the ARP $Probe\ Request$ to source IP of the packet to be checked for spoofing (RP_{IPS}). As discussed, it is assumed that all replies to the ARP probe will be sent within T_{req} time. So, SPOOF-DETECTOR() waits for T_{req} interval of time, thereby

collecting all probe responses in the Response table. As it is assumed that non-comprised hosts will always respond to a probe, at least one response to the probe will arrive. In other words, in one of the replies to the probe, genuine MAC for the IP RP_{IPS} would be present. Following that, Response table will be searched to find IP-MAC (source) pairs having IP of RP_{IPS}. If all IP-MAC pairs searched have same MAC, packet under question is not spoofed. In case of the packet being spoofed, more than one reply will arrive for the probe, one with genuine MAC and the other with spoofed MAC. The reason for assuming more than one replies in case of spoofing is explained as follows. Let a packet be spoofed as IP(of B)-MAC(of D). Now for the ARP probe to B, B will reply with IP(of B)-MAC(of B) leading to tracking the attacker (MAC (of D)). To avoid self identification, attacker D has to reply to all queries asking for B with spoofed IP-MAC pair IP(B)-MAC(D). The IDS has no clue whether IP(B)-MAC(D) or IP(B)-MAC(D) is genuine; only possibility of spoofing is detected.

If spoofing attempt is determined Log table is updated. If the packet is found genuine, Authenticated bindings table is updated with its source IP (RP_{IPS}) and the corresponding MAC.

Algorithm 4: SPOOF-DETECTOR
Input : RP- ARP request/reply packet, T_{req} - Time required for arrival of all responses to an ARP probe (ARP request-reply round trip time), Response table
Output: Updated Authenticated bindings table, Updated Log table, Status,

1: Wait for T_{req} time interval
2: **if** (RP_{IPS} == $RST_{IPS}[i]$) & & ($RP_{MACS} \neq RST_{MACS}[i]$)(for some i, $1 \leq i \leq RST_{MAX}$) **then**
3: Status=$Spoofed$
4: Add RP_{IPS}, RP_{MACS}, RP_{IPD}, RP_{MACD} and τ to Log table
5: Add RST_{IPS}, RST_{MACS}, RST_{IPD}, RST_{MACD} and τ to Log table
6: EXIT
7: **end if**
8: Update Authenticated bindings table with RP_{IPS},RP_{MACS}

UNSOLICITED-RESPONSE-HANDLER() (Algorithm 5) is invoked whenever an unsolicited ARP reply packet is received (i.e., ARP reply packet did not find a matching ARP request in the Request table) and is used for detection of denial of service attacks. Entries in the Unsolicited response table maintains the number of unsolicited responses received against individual IP addresses along with the timestamp of the latest such reply. This algorithm finds out whether the IP address against which unsolicited reply is received currently has a matching entry in the Unsolicited response table. This algorithm declares the detection of DoS attack if the number of unsolicited ARP replies against a particular IP, within a time interval (δ) exceeds a preset threshold DoS_{Th}. If a matching entry is not found a new entry is made into the Unsolicited response table.

Algorithm 5: UNSOLICITED-RESPONSE-HANDLER
Input : IP- Destination IP of the RSP , τ - Time when RSP is received, δ- Time window, DoS_{Th}- DoS Threshold, Unsolicited response table

Output: Status

if (IP == $URSPT_{IPD}[i]$), (for some i, $1 \leq i \leq URSPT_{MAX}$) **then**
  if ($\tau - URSPT_{\tau}[i] < \delta$) **then**
    $URSPT_{counter}[i]$++
    $URSPT_{\tau}[i]$= τ
    if ($URSPT_{counter}[i] > DoS_{Th}$) **then**
      Status=DoS
      EXIT
    end if
  else
    $URSPT_{counter}[i]$=1
    $URSPT_{\tau}[i]$= τ
  end if
else
  ADD IP, τ and 1 to the Unsolicited Response table
end if

Algorithms 1 - Algorithm 5 can detect spoofing, malformed APR packets, and denial of service attacks. To detect man in the middle attack another module MiTM-DETECTOR() is used and is discussed next. This module needs to scan through the Log table at certain intervals to determine man in the middle attacks. As spoofing or unsolicited replies are required for MiTM [1], the module MiTM-DETECTOR() is invoked whenever a spoofing is detected or an unsolicited response is received. In either of these two cases an entry is added to the Log table and MiTM-DETECTOR() is invoked with its source IP, source MAC and destination IP. This module analyzes all the entries in the Log table to detect the possible MiTM attacks (as each spoofing attempt or unsolicited replies are recorded in the Log table). If for a particular source IP - destination IP pair, there is another record having the destination IP- source IP pair (flipped version of earlier one) with same MAC address, within a particular time interval T_{MiTM}, then it detects the possibility of MiTM attack and the associated attacker's MAC. The algorithm first determines a subset of entries of the Log table whose source MAC matches the source MAC of APR packet last added to the Log table. Also, only those entries of the Log table are considered which have arrived within (*and not including*) T_{MiTM} time of the arrival of the ARP packet last added. Thus, we obtain a subset of the Log table as LT'. Then, if there is an entry in LT' where the source IP matches the destination IP of the packet last added and the destination IP of LT' matches the source IP of the packet last added, MiTM is detected.

Algorithm 6: MiTM-DETECTOR()
Input : IPS - Source IP of the entry added to Log table, $MACS$- Source MAC of the entry added to Log table, IPD - Destination IP of the entry added to Log table, τ - time when the entry was added in Log table, T_{MiTM} - Time interval for arrival of packets causing MiTM, Log table
Output: Status

1: $LT' = \{LT | \forall i\ (LT_{MACS}[i] == MACS)$ && $(\tau$ - $LT_{\tau}) < T_{MiTM}\ \}$
2: **if** $(LT'_{IPS}[j] == IPD)$ && $(LT'_{IPD}[j] == IPS)$,(for any j, $1 \leq j \leq LT'_{MAX}$) **then**
3: Status=$MiTM$ and "attacker is $MACS$"
4: **end if**

2.3 An example

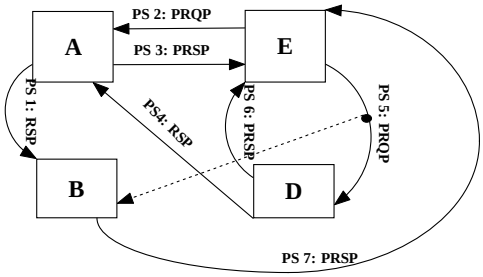

Fig. 1. Example of a Normal and Spoofed Reply

In this sub-section we illustrate ARP reply verification in normal and spoofed cases. Here, the network has five hosts A, B, C, D and E; E is the third party IDS and D is the attacker. Port mirroring is enabled at the switch so that E has a copy of all outgoing and incoming packets from all ports. Also, E has a network interface to solely send ARP probes and receive ARP probes replies.

Figure 1 shows the sequence of packets (indicated with packet sequence numbers) injected in the LAN when (i) A is sending a genuine reply to B with IP(A)-MAC(A) followed by ARP probe based verification (of the reply), (ii) attacker D is sending a spoofed reply as "IP(B)-MAC(D) " to host A and its verification. The sequences of packets as recorded in Request table, Response table, Verification table and Authenticated bindings table are shown in Table 1 - Table 4.

Genuine reply from A to B and its verification

- Packet Sequence (PS) 1: Reply is sent by A to B for informing its MAC address (to B). Response table is updated with a new entry corresponding to the request packet sent by A .
- Packet Sequence 2: Since there is no entry for IP-MAC of A in Authenticated bindings table, E will send an ARP Probe to know MAC of A and entry is made in the Verification table.
- Packet Sequence 3: Following PS 2, SPOOF-DETECTOR() starts. Within T_{req} only A will respond to this ARP Probe request and Authenticated bindings table is updated with the valid entry of IP-MAC of A.

Spoofed reply from D to A and its verification

- Packet Sequence 4: Let D respond to A with IP of B and its own MAC (D), which is recorded in the Response table.
- Packet Sequence 5: Since there is no entry for IP-MAC of B in Authenticated bindings table therefore E will send an ARP probe to know B's MAC. IP (B)-MAC(D) is entered in the Verification table.
- Packet Sequence 6 and 7: SPOOF-DETECTOR() is executed. Within T_{req}, both B and attacker D will respond to the ARP Probe request (sent to know MAC of B) with their own MACs. These responses are recorded in the Response table.
  There are two entries in Response table for IP(B), one is MAC of B and other is MAC of D. So response spoofing is detected.

Table 1. Request table

PS	SRC IP	SRC MAC	Dest IP
-	-	-	-

Table 2. Response table

PS	SRC IP	SRC MAC	DEST IP	DEST MAC
1	IP A	MAC A	IP B	MAC B
3	IP A	MAC A	IP E	MAC E
4	IP B	MAC D	IP A	MAC A
6	IP B	MAC D	IP E	MAC E
7	IP B	MAC B	IP E	MAC E

Table 3. Verification table

PS	IP	MAC
2	IP A	MAC A
5	IP B	MAC D

Table 4. Authenticated bindings table

PS	MAC
IP A	MAC A

3 Experimentation

The test bed created for our experiments consists of 5 machines running different operating systems. We name the machines with alphabets ranging from A-E. Machines A-E are running the following OSs: Windows Xp, Ubuntu, Windows 2000, Backtrack 4 and Backtrack 4, respectively. The machine D with Backtrack 4 is acting as the attacker machine and machine E is set up as the IDS. These machines are connected in a LAN with a CISCO catalyst 3560 G series switch [15] with port mirroring enabled for system E.

The tables mentioned above are created in mysql database. The algorithms are implemented using C language. The IDS has two preemptive modules namely, packet grabber and packet injector. Packet grabber sniffs the packets from the network, filters ARP packets and invoke either the Algorithm 1 or Algorithm 2 depending upon the

packet type. Packet injector generates the ARP probes necessary for verification of IP-MAC pairs. Attack generation tools Ettercap, Cain and Abel were deployed in machine D and several scenarios of spoofing MAC addresses were attempted.

Table 5. Comparison of ARP Attack Detection Mechanisms

ATTACKS	PROPOSED	ACTIVE [12]	COLASOFT [6]	ARPDEFENDER [5]
ARP spoofing	Y	Y	Y	Y
ARP MiTM	Y	N	N	Y
ARP DoS	Y	N	Y	N
Network Scanning	Y	N	N	N
Malformed Packets	Y	Y	N	N
MAC Flooding	Y	N	N	Y

In our experiments we tested our proposed scheme with several variants of LAN attack scenarios (including the one discussed in the example above). Table 5 presents the types of LAN attacks generated and detected successfully by the proposed scheme. Also, in the table we report the capabilities of other LAN attack detecting tools for these attacks.

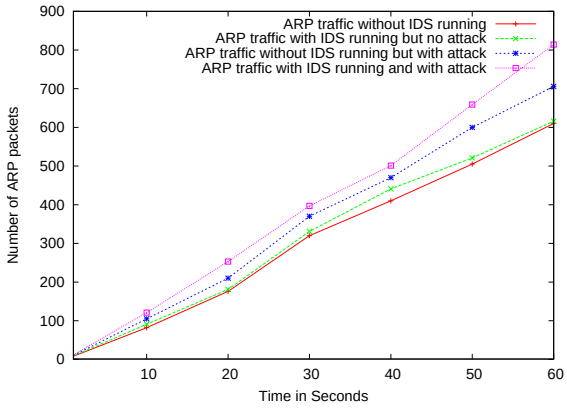

Fig. 2. ARP traffic

Figure 2 shows the amount of ARP traffic generated in the experimentation in 4 cases. The first case is of normal operation in the absence of the IDS. Second case is when the IDS is running and there are no attacks generated in the network. Third case is when we injected 100 spoofed IP-MAC pairs into the LAN and IDS is not running. Fourth case is when we injected 100 spoofed IP-MAC pairs into the LAN with IDS running. We notice almost same amount of ARP traffic under normal situation with and

without IDS running. Once genuine IP-MAC pairs are identified (by probing) they are stored in Authenticated bindings table. Following that no probes are required to be sent for any ARP request/reply from these IP-MAC pairs. In case of attack, a little extra traffic is generated by our IDS for the probes. With each spoofed ARP packet, our IDS sends a probe request and expects at least two replies (one from normal and the other from the attacker), thereby adding only three APR packets for each spoofed packet.

4 Conclusion

In this paper we presented an IDS for detecting a large class of LAN specific attacks. The scheme uses an active probing mechanism and does not violate the principles of network layering architecture. This being a software based approach does not require any additional hardware to operate.

At present the scheme can only detect the attacks. In other words, in case of spoofing it can only determine the conflicting IP-MAC pairs without differentiating the spoofed IP-MAC and genuine IP-MAC pair. If to some extent diagnosis capability can be provided in the scheme, some remedial action against the attacker can be taken.

References

1. Held, G.: Ethernet Networks: Design, Implementation, Operation, Management. 1 edn. John Wiley & Sons, Ltd. (2003)
2. Kozierok, C.M.: TCP/IP Guide. 1 edn. No Starch Press (October 2005)
3. LTD, C.S.P.: Cisco 6500 catalyst switches
4. arpwatch: www.arpalert.org
5. arpdefender: www.arpdefender.com
6. colasoft capsa: www.colasoft.com
7. weight intrusion detection, S.L.: www.snort.org
8. Abad, C.L., Bonilla, R.I.: An analysis on the schemes for detecting and preventing arp cache poisoning attacks. In: ICDCSW '07: Proceedings of the 27th International Conference on Distributed Computing Systems Workshops, Washington, DC, USA, IEEE Computer Society (2007) 60–67
9. Hsiao, H.W., Lin, C.S., Chang, S.Y.: Constructing an arp attack detection system with snmp traffic data mining. In: ICEC '09: Proceedings of the 11th International Conference on Electronic Commerce, New York, NY, USA, ACM (2009) 341–345
10. Gouda, M.G., Huang, C.T.: A secure address resolution protocol. Comput. Networks. **41**(1) (2003) 57–71
11. Lootah, W., Enck, W., McDaniel, P.: Tarp: Ticket-based address resolution protocol, Los Alamitos, CA, USA, IEEE Computer Society (2005) 106–116
12. Ramachandran, V., Nandi, S.: Detecting arp spoofing: An active technique. ICISS05:1st International Conference on Information Systems Security, LNCS **3803** (2005) 239–250
13. Trabelsi, Z., Shuaib, K.: Man in the middle intrusion detection. In: Globecom, San Francisco, California, USA, IEEE Communication Society (2006) 1–6
14. Sisaat, K., Miyamoto, D.: Source address validation support for network forensics. In: JWICS '06: The 1st Joint Workshop on Information security. (2006) 387–407
15. Whitepaper, C.: http://www.cisco.com