ISA Transactions 50 (2011) 119–130

Contents lists available at ScienceDirect

ISA Transactions

journal homepage: www.elsevier.com/locate/isatrans

LAN attack detection using Discrete Event Systems

Neminath Hubballi, Santosh Biswas *, S. Roopa, Ritesh Ratti, Sukumar Nandi

Department of Computer Science and Engineering, Indian Institute of Technology, Guwahati 781039, India

ARTICLE INFO

Article history:
Received 28 February 2010
Received in revised form
7 August 2010
Accepted 8 August 2010
Available online 30 August 2010

Keywords:
Discrete Event Systems
Failure detection
Network security
Local Area Network (LAN) Attack
Address Resolution Protocol (ARP)

ABSTRACT

Address Resolution Protocol (ARP) is used for determining the link layer or Medium Access Control (MAC) address of a network host, given its Internet Layer (IP) or Network Layer address. ARP is a stateless protocol and any IP–MAC pairing sent by a host is accepted without verification. This weakness in the ARP may be exploited by malicious hosts in a Local Area Network (LAN) by spoofing IP–MAC pairs. 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 a Discrete Event System (DES) approach for Intrusion Detection System (IDS) for LAN specific attacks which do not require any extra constraint like static IP–MAC, changing the ARP etc. A DES model is built for the LAN under both a normal and compromised (i.e., spoofed request/response) situation based on the sequences of ARP related packets. Sequences of ARP events in normal and spoofed scenarios are similar thereby rendering the same DES models for both the cases. To create different ARP events under normal and spoofed conditions the proposed technique uses active ARP probing. However, this probing adds extra ARP traffic in the LAN. Following that a DES detector is built to determine from observed ARP related events, whether the LAN is operating under a normal or compromised situation. The scheme also minimizes extra ARP traffic by probing the source IP–MAC pair of only those ARP packets which are yet to be determined as genuine/spoofed by the detector. Also, spoofed IP–MAC pairs determined by the detector are stored in tables to detect other LAN attacks triggered by spoofing namely, man-in-the-middle (MiTM), denial of service etc. The scheme is successfully validated in a test bed.

1. Introduction

The security and performance considerations in any organization with a 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 a department or building. A security threat to any computer, based on LAN specific attacks is always from a compromised or malicious host in the LAN. The basic step involved in most of these attacks comprises cache poisoning with falsified IP–MAC pairs, which may then lead to other attacks namely, MiTM, denial of service etc. [1].

Computers in the internet are identified by their IP addresses. IP addresses are used by the network layer for identifying the machine uniquely. At the data link layer, computers use another address known as a MAC address or hardware address. It is to be noted that IP addresses can be dynamic and change over time but the MAC address of a computer is constant unless the Network Interface Card (NIC) is replaced. To deliver a packet to the correct machine, the IP address has to be mapped to some MAC address. This dynamic binding (since IP is dynamic) between the IP and MAC address is done by the Address Resolution Protocol (ARP). ARP is responsible for finding the MAC address given the IP address. The data link layer uses the MAC address of the destination machine for sending the packets. If the host sending the packets does not know the MAC address of the destination host, it sends a broadcast request to know "What is the MAC address corresponding to the IP address". The host which has the IP address in the broadcast message sends a unicast reply message to the sender, mentioning its MAC address. In order to reduce the number of broadcast requests each machine maintains a table termed as the ARP cache, which holds the mapping between the IP and MAC. Entries in the ARP cache can be either static or dynamic. In the dynamic cache the entries are erased as they get older than a predefined duration. The problem with ARP is that, it is a stateless protocol. Any host after receiving any ARP response message will update its cache without verifying whether it has earlier sent a request corresponding to the response. This enables the malicious hosts to craft custom ARP packets and forge the IP–MAC pair.

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

* Corresponding author. Tel.: +91 9957561026; fax: +91 361 2692787.
E-mail addresses: neminath@cse.iitg.ernet.in (N. Hubballi), santosh_biswas@cse.iitg.ernet.in, santoshbiswas402@yahoo.com (S. Biswas), roopa.s@cse.iitg.ernet.in (S. Roopa), r.ratti@cse.iitg.ernet.in (R. Ratti), sukumar@cse.iitg.ernet.in (S. Nandi).

0019-0578/$ – see front matter © 2010 ISA. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.isatra.2010.08.003

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, this scheme is not suitable for a dynamic environment.

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 the 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 has also been utilized in software solutions namely, ARPWATCH [4], COLASOFT-CAPSA [5]. These software solutions are cheaper than switches with port security but have a slower response time compared to switches. Obviously, these tools suffer from the same drawbacks as that of port security in switches.

Signature and anomaly based IDS: Signature based IDSs like Snort [6] 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, the ability of IDSs to detect all forms of ARP related attacks is limited [7]. Recently, Hsiao et al. [8], 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 Bayes 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 [9], TARP [10]. Addition of cryptographic features in ARP lead to a performance penalty [7] and change the basic ARP.

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

In [11], 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 a 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 man-in-the-middle (MiTM) attacks is proposed in [12]. The scheme assumes that any attacker involved in MiTM must have IP forwarding enabled. First, all systems with IP forwarding are detected (actively). Then the IDS attacks all such systems one at a time and poisons 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). In this way the IDS can differentiate real MiTM attackers from all systems with IP forwarding. There are several drawbacks in this approach, namely huge traffic in the case of a large network with all machines having IP forwarding, assumption of successful cache poisoning of the machine involved in an MiTM attack, cache poisoning (of the machine involved in an MiTM attack by IDS) exactly when the attack is going on etc.

From the review, it may be stated that an ARP attack prevention/detection scheme needs to have the following features

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

In this paper, a Discrete Event System (DES) based network IDS for detecting ARP related attacks has been proposed. A DES is characterized by a discrete state space and some event driven dynamics. DES have widely been used for failure detection and diagnosis of large systems like chemical reaction chambers, nuclear reactors etc. [13]. The basic idea is to develop a DES model for the system under normal conditions and also under each of the failure conditions. Following that, a state estimator called a diagnoser (or detector, if only detection of failure is required) is designed which observes sequences of events generated by the system to decide whether the states through which the system traverses correspond to the normal or faulty DES model. This idea has been used to develop host based IDS [14,15], where a sequence of system calls under normal conditions comprises the normal model and the sequence under compromised conditions comprises the failure (attack) model. So, the DES detector acts as the host IDS. A preliminary attempt has been made for network level IDS using the same idea in [16]. The work illustrated theoretically how DES may be applied to detect one type of ARP attack i.e., response spoofing.

The objective of the present paper is to design and implement a DES based network IDS that meets all the desired characteristics of an ARP attack detector (as listed above). Design of such an IDS requires certain extensions over the (classical) theory [13] and techniques which have been used in [14–16]. The basic motivations and the extensions are enumerated below:

(1) In the case of ARP, the DES framework needs to model not only sequences of events but also their time of occurrences. So the DES model used in the proposed technique extends the untimed model [14,15] with time information.
(2) Unlike host based IDS, in the case of spoofing (in LAN attacks) there is no difference in the sequence of ARP events compared to normal situations. To handle this situation, an active probing mechanism is used so that sequences of ARP packets are different under spoofing and normal conditions. It may be noted that the probing technique maintains the standard ARP.
(3) Active ARP probes generate extra traffic in the network. To minimize additional traffic, IP–MAC pairs corresponding to normal and spoofed conditions (decided by the detector) are recorded in tables. Following that, ARP probes are sent only to IP–MAC pairs which are absent in the tables. Interfacing results of the detector with the ARP probe sending module requires extension of the concept given in [16].
(4) Entries in the table are used to verify if spoofing leads to other attacks like man-in-the-middle and denial of service etc. Detecting attacks other than spoofing requires enhancement of [16].

The proposed network IDS based on failure detection theory of DES [13] has the following salient features

(1) Follows standard ARP and does not violate the principles of network layering structure.
(2) Generates minimal extra traffic in the network, as probes are sent only for unverified IP–MAC pairs.
(3) It involves installation of the DES detector (based network IDS) in just one host in the network.
(4) Detects a large set of LAN attacks namely, malformed packets, response spoofing, request spoofing, man-in-the-middle and denial of service.

(5) The only hardware requirement of the IDS is a switch with port mirroring facility.

Henceforth, in this paper IDS will refer to network IDS and if host IDS is discussed it will be explicitly mentioned. The rest of the paper is organized as follows. Section 2 presents the proposed approach. Section 3 deals with implementation of the proposed scheme and comparison with similar approaches reported in the literature. Finally the paper is concluded in Section 4.

2. Proposed scheme

In this section the proposed DES based intrusion detection scheme for ARP related attacks is presented.

The following assumptions are made regarding the LAN.

- Non-compromised hosts will send one response to an ARP request within a specific interval T_{req}.
- IDS is running on a dedicated and trusted machine with a fixed IP.
- Port mirroring is enabled at the switch so that IDS has a copy of all outgoing and incoming packets from all ports.

In the next subsection the probing technique is discussed. Following that an example is given to motivate the requirement of probing to generate different sequences of ARP packets under normal and attack scenarios.

2.1. Active probing technique

Upon receiving ARP requests or ARP replies, ARP probes are sent to the source IP of the request or reply packet. Before sending probes it is checked that the source IP–MAC pair is not yet verified by the DES detector to be genuine or spoofed.[1] Details of such verified IP–MAC pairs are recorded in tables, discussed below. Henceforth in the discussion, the following short notations are used:

IPS—Source IP Address; *IPD*—Destination IP Address; *MACS*—Source MAC Address; *MACD*—Destination MAC Address; *RQP*—ARP request packet; *RSP*—ARP response packet. Source IP of *RQP* is denoted as RQP_{IPS}; similar subscripting will be used to denote destination IP, source MAC and destination MAC for *RQP* and *RSP*.

(1) Every time an ARP request is verified to be genuine an entry is made in the Authenticated table, denoted as *AUTHT*. It has five fields, source IP $AUTHT_{IPS}$, source MAC $AUTHT_{MACS}$, destination IP $AUTHT_{IPD}$, destination MAC $AUTHT_{MACD}$ and time when the request packet is received by IDS $AUTHT_{Ti}$.
(2) For spoofed requests the details are entered in another table called the Spoofed table (denoted as *SPOOFT*) which has the same fields as the Authenticated table.
(3) Every time an ARP response is verified to be genuine (or spoofed) an entry is made in the Authenticated (Spoofed) table.

In the case of ARP requests only four out of five fields in the tables can be filled; *MACD* left as "——" as no destination MAC address is associated with requests. However, in the case of responses all five fields are filled with appropriate values.

$\langle TableName \rangle_{MAX}$ represents the maximum number of elements in a table at a given time. These tables not only help in minimizing ARP probes but will also be used for determining if spoofing has led to other attacks like man-in-the-middle, denial of service etc.

The probing technique has two main modules namely, ARP REQUEST-HANDLER() and ARP RESPONSE-HANDLER(). These are elaborated in Algorithms 1 and 2, respectively. Algorithm 1 processes ARP request packets in the network. For any ARP request packet *RQP*, it first verifies if it is malformed (i.e., any changes in the immutable fields of the ARP packet header or different MAC addresses in the MAC and ARP header field) or unicast. These cases are abnormal and a decision can be made on observation of any such (single) packet, unlike sequences of ARP packets in the case of spoofing. Detection of such abnormal cases is trivial and we do not consider them in DES modeling and attack detection. Algorithm 1 simply exits (by setting a Status flag) on such ARP request packets. 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 need not send ARP probes to requests from the IDS as we assume that IP–MAC pairing of the IDS is known and validated. If none of the above cases are matched, then RQP_{IPS} is searched in the Authenticated table. If a match is found as $AUTHT_{IPS}[i]$ and the corresponding source MAC address $AUTHT_{MACS}[i]$ in the table is the same as RQP_{MACS}, the packet has a genuine IP–MAC pair which is already determined by the detector. *This genuineness is obtained without explicit use of the DES detector. Also, ARP probes are not sent in this case thereby saving some ARP traffic.* RQP_{IPS}, RQP_{MACS}, RQP_{IPD}, —— *and Time of receipt are stored in the Authenticated table.* In the case of a match in the source IP and mismatch in the source MAC address (i.e., $RQP_{IPS} = AUTHT_{IPS}[i]$ and $RQP_{MACS} \neq AUTHT_{MACS}[i]$) the packet is detected to be spoofed *without explicit use of the DES detector and no ARP probes are sent.* The details of the *RQP* are stored in the Spoofed table. On the other hand if RQP_{IPS} is not found in *AUTHT* (in field $AUTHT_{IPS}$), then both RQP_{IPS} and RQP_{MACS} are searched in *SPOOFT* (i.e., $RQP_{IPS} = SPOOFT_{IPS}[i]$ and $RQP_{MACS} = SPOOFT_{MACS}[i]$). If a match is found then it implies that the IP–MAC pair is already detected to be spoofed. No ARP probes are sent and details of the request packet are added in the *SPOOFT*. If both the above cases are not satisfied then an ARP probe request is sent (broadcast) by Algorithm 1 requesting the *MAC* for the source IP RQP_{IPS}. In another case i.e., gratuitous ARP requests, an ARP probe is sent. Gratuitous ARP requests are broadcast (e.g., entry of a new host or change in NIC card) to inform other hosts in the LAN about its IP–MAC pair. Gratuitous ARP requests can be determined if $RQP_{IPS} = RQP_{IPD}$. For such a Gratuitous request, an ARP probe is sent to the RQP_{IPS} without verifying in the tables because such requests are made when a host with a "new IP–MAC pair" comes up which needs to be verified by the detector afresh.

As already discussed, any DES model has states, and transitions are made among states on occurrence of some (discrete) events. These events are modeled based on certain changes in the system. For example, in a chemical reaction chamber, temperature moving above the threshold can be modeled as an event which leads to a change in state to make the heater off [13]. Sensors are kept in the system to measure such changes, e.g., a thermocouple/temperature sensor in the case of the chemical chamber. In our case, we consider the following changes in the LAN as events: Receipt of *RQP*, sending of an ARP probe request (*PRQP*), receipt of an ARP probe response (*PRSP*), receipt of response (other than for an ARP probe) *RSP*. These requests/responses with their respective fields (e.g., source IP of *RSP*) are considered as measured events of the DES model. In addition, when Algorithm 1 decides some IP–MAC pair to be genuine/spoofed using the Authenticated table/Spoofed table, it intimates the same; this is captured by the event "detected (*DTD*)".

Algorithm 1 is first illustrated using a flow chart in Fig. 1 and formalized subsequently.

Algorithm 1: **ARP REQUEST HANDLER**
Input : *RQP* - ARP request packet
Output: Intimate receipt of *RQP*, sending of ARP probe request and detected *DTD*, Updated *AUTHT* and *SPOOFT*

[1] The technique to determine if an IP–MAC pair is genuine or spoofed is detailed in Section 2.4.

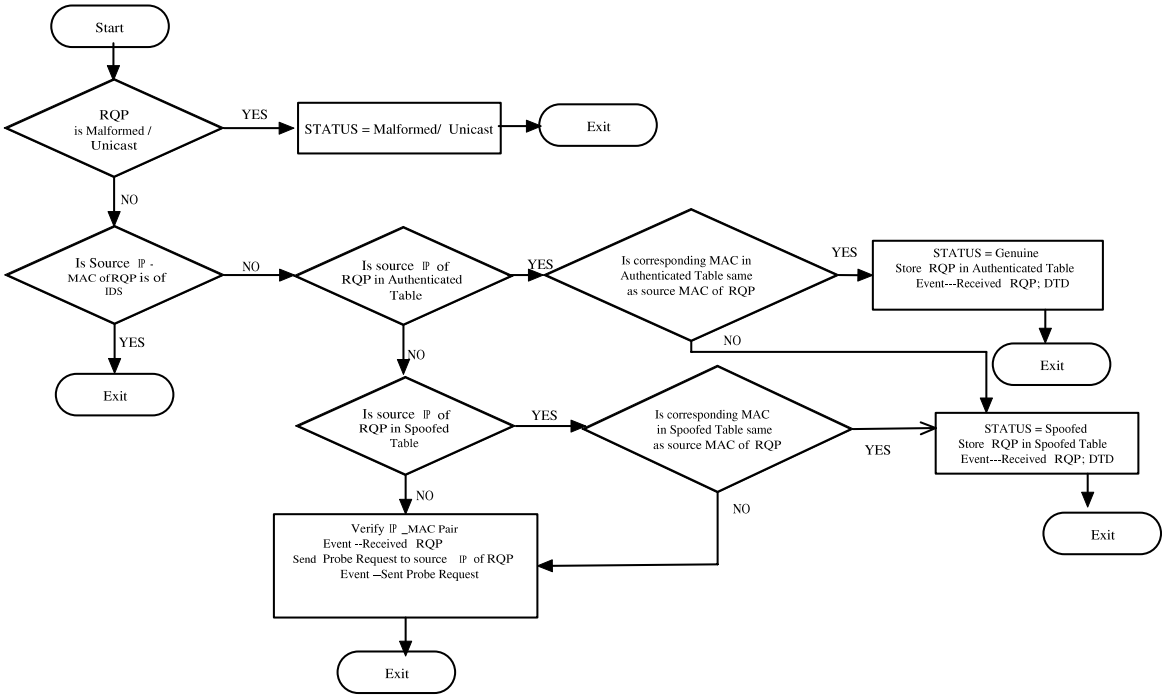


Fig. 1. Flow chart for ARP request handler.

IF(RQP is *malformed*) OR (RQP is *Unicast*) "Status is abnormal" and EXIT
IF($RQP_{IPS} = IP(IDS)$) AND ($RQP_{MACS} = MAC(IDS)$) EXIT
IF($RQP_{IPS} = AUTHT_{IPS}[i]$ AND $RQP_{MACS} = AUTHT_{MACS}[i]$, for some i, $1 \leq i \leq AUTHT_{MAX}$) "Status is Genuine"; add {RQP_{IPS}, RQP_{MACS}, RQP_{IPD}, ––, time of receipt} in $AUTHT$; Intimate RQP and DTD; EXIT
IF($RQP_{IPS} = AUTHT_{IPS}[i]$, for some i, $1 \leq i \leq AUTHT_{MAX}$) AND ($RQP_{MACS}! = AUTHT_{MACS}[i]$)
"Status is Spoofed"; add {RQP_{IPS}, RQP_{MACS}, RQP_{IPD}, ––, time of receipt} in $SPOOFT$; Intimate RQP andDTD; EXIT
IF($RQP_{IPS} = SPOOFT_{IPS}[i]$ and $RQP_{MACS} = SPOOFT_{MACS}[i]$, for some i, $1 \leq i \leq SPOOFT_{MAX}$)
"Status is Spoofed"; add {RQP_{IPS}, RQP_{MACS}, RQP_{IPD}, ––, time of receipt} in $SPOOFT$; Intimate RQP and DTD; EXIT
ELSE(/* Gratuitous i.e., $RQP_{IPS} == RQP_{IPD}$ or none of the above conditions */)
Intimate receipt of RQP; Send ARP Probe Request $PRQP$ to RQP_{IPS} from IDS; Intimate sending of the ARP probe request $PRQP$;

Algorithm 2 is the ARP response handler. For any ARP reply packet RSP, the algorithm first finds the malformed packets and sets the status accordingly. Next, it verifies whether the reply packet is for any ARP probe (sent by the IDS). The response for an ARP probe can be determined if RSP_{IPD} is the IP of IDS and RSP_{MACD} is the MAC of IDS. For these packets no probe is sent and the algorithm intimates the receipt of these response packets ($PRSP$). If the response is not malformed or a reply to an ARP probe, the IDS sends an ARP probe request (broadcast) to the source IP RSP_{IPS} requesting its MAC. Before sending the probe, Algorithm 2 does a check in the Authenticated table and Spoofed table, similar to the one discussed in Algorithm 1.

Algorithm 2 is first illustrated using a flow chart in Fig. 2 and formalized subsequently.

Algorithm 2: **ARP RESPONSE HANDLER**
Input: RSP - ARP response packet
Output: Intimate receipt of RSP, ARP probe response $PRSP$, sending of ARP probe request $PRQP$ and detected DTD, Updated $AUTHT$ and $SPOOFT$

IF(RSP is *malformed*) "Status is abnormal" and EXIT
IF(RSP_{IPD} = IP(IDS)) AND (RSP_{MACD} = MAC(IDS)) Intimate receipt of $PRSP$; EXIT
IF($RSP_{IPS} = AUTH_{IPS}[i]$ AND $RSP_{MACS} = AUTH_{MACS}[i]$, for some i, $1 \leq i \leq AUTHT_{MAX}$)
"Status is Genuine"; add {RSP_{IPS}, RSP_{MACS}, RSP_{IPD}, RSP_{MACD}, time of receipt} in $AUTHT$; Intimate RSP and DTD; EXIT
IF($RQP_{IPS} = AUTH_{IPS}[i]$, for some i, $1 \leq i \leq AUTH_{MAX}$) AND ($RQP_{MACS}! = AUTH_{MACS}[i]$)
"Status is Spoofed"; add {RSP_{IPS}, RSP_{MACS}, RSP_{IPD}, RSP_{MACD} and time of receipt} in $SPOOFT$; Intimate RSP and DTD; EXIT
IF($RQP_{IPS} = SPOOFT_{IPS}[i]$ and $RQP_{MACS} = SPOOFT_{MACS}[i]$, for some i, $1 \leq i \leq SPOOFT_{MAX}$)
"Status is Spoofed"; add {RSP_{IPS}, RSP_{MACS}, RSP_{IPD}, RSP_{MACD}, time of receipt} in $SPOOFT$; Intimate RSP and DTD; EXIT
ELSE(/* Gratuitous i.e., $RSP_{IPS} == RSP_{IPD}$ or none of the above conditions */)
Intimate receipt of RSP; Send ARP Probe Request $PRQP$ to RSP_{IPS} from IDS; Intimate sending of the ARP probe request $PRQP$;

2.2. An example

In this sub-section an example is used to illustrate handling of spoofed response packets by Algorithm 2. Further, this example also highlights the difference in ARP packet sequences (after active probing) in the case of spoofing versus normal scenarios. Here, the network has four hosts A, B, D and E; E is the IDS and D is the attacker. The machines A, B, D and E are assigned IP addresses 10.0.1.1, 10.0.1.2, 10.0.1.4 and 10.0.1.5 respectively. The MAC addresses of the hosts A, B, D and E are 08:4D:00:7D:C1, 08:4D:00:7D:C2, 08:4D:00:7D:C4 and 08:4D:00:7D:C5, respectively. 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 probe replies.

Fig. 3 shows the sequence of packets (indicated with packet sequence numbers) injected in the LAN when attacker D is sending

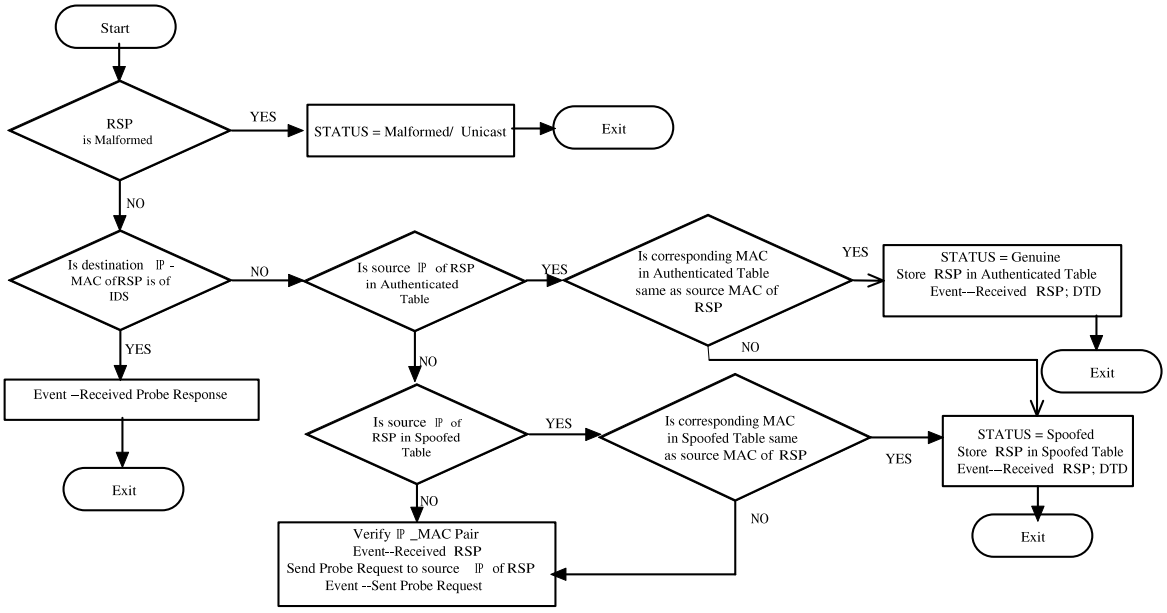


Fig. 2. Flow chart for ARP response handler.

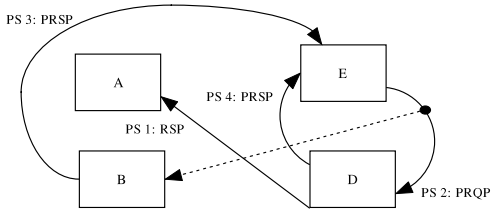


Fig. 3. Example of spoofed reply.

Table 1
ARP cache of machine A under spoofed reply.

IP	MAC	Interface
10.0.1.2	08:4D:00:7D:C4	eth0

a spoofed reply as "IP(B)–MAC(D)" to host A and its verification (using active probe). As packet sequence 1, D sends an ARP response *RSP* to A telling that the IP of B is associated with the MAC of D. This ARP response conversation is viewed in tcpdump (one of the utilities for watching packets visible to an interface) as:

08:4D:00:7D:C4 08:4D:00:7D:C1 60: arp reply 10.0.1.2 is-at 08:4D:00:7D:C4.

As ARP is a stateless protocol, after receiving the response from D, A updates its cache with IP–MAC pair as IP(B)–MAC(D); the ARP cache of A with the update is shown in Table 1.

It may be noted that under normal condition "B's reply to A with IP–MAC pair:IP of B-MAC of B" and attack condition "D's reply to A with IP–MAC pair:IP of B-MAC of D", there is no change in the sequence of ARP packets. Under the attack, all traffic A wants to send to B will be sent to D.

Now the use of active probing is discussed that creates a difference in the sequences of ARP packets under normal and attack situations. Initially the ARP cache table of all the machines is empty. Also, it is assumed that the Authenticated table and Spoofed table are empty (at this instant).

On receiving the *RSP* from D to A (packet sequence 1), ARP RESPONSE HANDLER() at E, intimates the receipt of *RSP* (event 1), sends ARP probe request *PRQP* to IP of B to query the corresponding MAC (packet sequence 2) and intimates sending of the ARP probe request *PRQP* (event 2). This ARP probe request conversation is viewed in tcpdump as:

Table 2
Spoofed table entry when spoofing is detected.

SRC IP	SRC MAC	Dest IP	Dest MAC	Time
10.0.1.2	08:4D:00:7D:C4	10.0.1.1	08:4D:00:7D:C1	00:00:30

08:4D:00:7D:C5 ff:ff:ff:ff:ff:ff 42: arp who-has 10.0.1.2 tell 10.0.1.5.

The probe request is sent, as the source IP–MAC pair of this *RSP* is not yet verified (Authenticated table and Spoofed table being empty). As the ARP probe is a broadcast, both B and attacker D will get the probe. B will definitely respond to the probe as *PRSP* with IP(B)–MAC(B) to E (packet sequence 3) as it is assumed that the non-compromised host (B) will always respond to the probe. This ARP probe response conversation is viewed in tcpdump as:

08:4D:00:7D:C2 08:4D:00:7D:C5 60: arp reply 10.0.1.2 is-at 08:4D:00:7D:C2.

Now the IDS can know that the response made in packet sequence 1 is false (as it had IP(B)–MAC(D)) and the system administrator can generate an alarm (and also track the attacker MAC (D)). To avoid self-identification, attacker D has to reply to all queries for MAC of IP(B) with IP(B)–MAC(D); this ARP probe response from D is viewed in tcpdump as:

08:4D:00:7D:C4 08:4D:00:7D:C5 60: arp reply 10.0.1.2 is-at 08:4D:00:7D:C4.

It may be noted that the order of reply by the attacker and the genuine host is not fixed. So, if both attacker and the genuine host reply to a request with their MACs, the IDS can detect a spoofing but cannot point to the attacker's MAC. So, it is reasonable to believe that an attacker would reply to all queries against the IP it is spoofing. To summarize, at least one reply to a probe (sent to verify IP(B), say) will arrive and have the genuine IP–MAC of B. In the case of spoofing, more replies will arrive which may have different MACs. Once the spoofing is detected an entry is made in the Spoofed table as shown in Table 2.

In this example, we assume that the genuine host B replies (packet sequence 3) before attacker D (packet sequence 4) to the ARP probe. The replies in packet sequence 3 and packet sequence 4 are processed by ARP RESPONSE HANDLER() as "Intimate receipt of *PRSP*" thereby generating event 3 and event 4, respectively. It may be noted that in the two replies, different MACs are associated with the IP being probed i.e., once MAC(B) is associated with IP(B) and

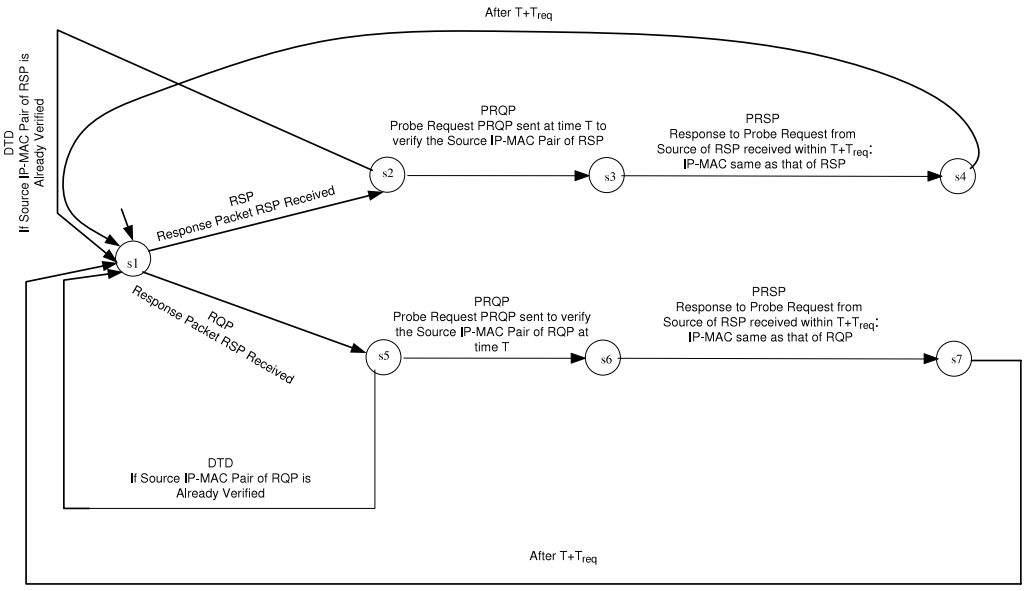


Fig. 4. State based ARP model under normal conditions.

Table 3
Tabulation of the packet sequences and events in the example.

PS: Events	SRC IP	SRC MAC	Dest IP	Dest MAC
PS 1:*RSP*	10.0.1.2	08:4D:00:7D:C4	10.0.1.1	08:4D:00:7D:C1
PS 2:*PRQP*	10.0.1.5	08:4D:00:7D:C5	10.0.1.2	–
PS 3:*PRSP*	10.0.1.2	08:4D:00:7D:C2	10.0.1.5	08:4D:00:7D:C5
PS 4:*PRSP*	10.0.1.2	08:4D:00:7D:C4	10.0.1.5	08:4D:00:7D:C5

then MAC(D) is associated with IP(B). Table 3 lists the sequence of ARP packets for the example (in Fig. 3).

Now let us see the sequence of ARP traffic if the response in packet sequence 1 was genuine, i.e., IP(B)–MAC(B) was sent to A by B. For the ARP probe sent to verify IP(B)–MAC(B) (packet sequence 2) only B will respond to E with IP(B)–MAC(B) (packet sequence 3). Till T_{req} time the IDS will receive only one *PRSP* (unlike two in the case of attack). Once the DES detector (i.e., IDS) finds the reply to be genuine (by virtue of only one *PRSP*), details of *RSP* are stored in the Authenticated table. If more than one *PRSP*s are received within T_{req} with different MACs, details of *RSP* are stored in the Spoofed table. Following that, no probes will be sent for any ARP reply/request with source IP as IP(B). However, in the case of a Gratuitous request/reply, an ARP probe will be sent.

As discussed before, the IDS engine for detecting the ARP spoofing attacks would be constructed using a DES detector. So, initially the ARP events under normal and spoofed conditions are modeled using state-event based DES models and subsequently a DES detector is designed. Before we discuss the formal DES modeling framework we illustrate abstract state-event based models for ARP under normal and spoofed conditions. Fig. 4 shows the state based ARP model under normal conditions. Events listed before, namely receipt of *RQP*/*RSP*, sending of ARP probe request (*PRQP*) and receipt of ARP probe response (*PRSP*) are shown in bold with the transitions in the figure. Further, some elaboration of the events are also illustrated with the transitions. Fig. 5(a) and (b) show the model under request spoofing and response spoofing, respectively.

2.3. DES modeling

The DES model used for representing the LAN under normal and attack scenarios is a six tuple $\langle \Sigma, S, S_0, V, C, \Im \rangle$, where Σ is the set of events, S is the set of states, $S_0 \subseteq S$ is the set of initial states, V is the set of model variables, C is the set of clock variables and $\Im$ is the set of transitions. It may be noted that there is no final state as ARP traffic in LAN is a continuous (or renewal) process (never halts as long as the network is up). There are some states from which there is a transition to an initial state(s), representing renewal of the process; such states are termed as renewal states.

Each element v_i of V ($V = \{v_1, v_2, \ldots, v_n\}$) can take values from a domain D_i. The clock variables take values in non-negative reals $\mathbb{R}$. An invariant condition on a clock variable c, denoted as $\Phi(c)$, is a linear inequality or equality of the variable with a non-negative real.

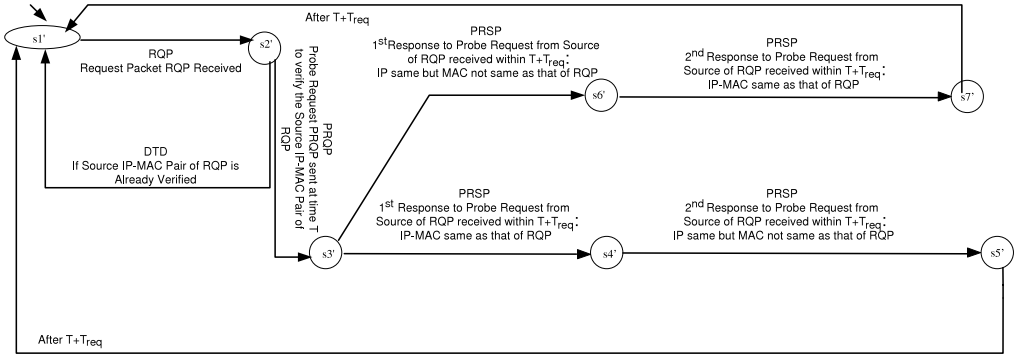
A transition $\tau \in \Im$ is a seven tuple $\langle s, s', \sigma, \phi(V), \Phi(C), \text{Reset}(C), \text{Assign}(V) \rangle$, where s is the source state, s' is the destination state, σ is an event (on which the transition is fired), $\phi(V)$ is Boolean conjunction of equalities of a subset of variables in V, $\Phi(C)$ is the invariant condition on a subset of clock variables, Reset(C) is a subset of clock variables to be reset and Assign(V) is a subset of model variables and assignments with values from their corresponding domains.[2]

Two attacks are considered namely, (i) Request spoofing: where the attacker sends an ARP request with falsified IP–MAC (source) pair in a query (to know the MAC address of some IP in the LAN), (ii) Response Spoofing: where the attacker sends a falsified IP–MAC (source) pair in a reply to some machine in the LAN. Fig. 6 shows the normal DES model of the LAN. Fig. 7(a) shows the DES model of the LAN under an ARP request spoofing attack and Fig. 7(b) is for an ARP response spoofing attack.

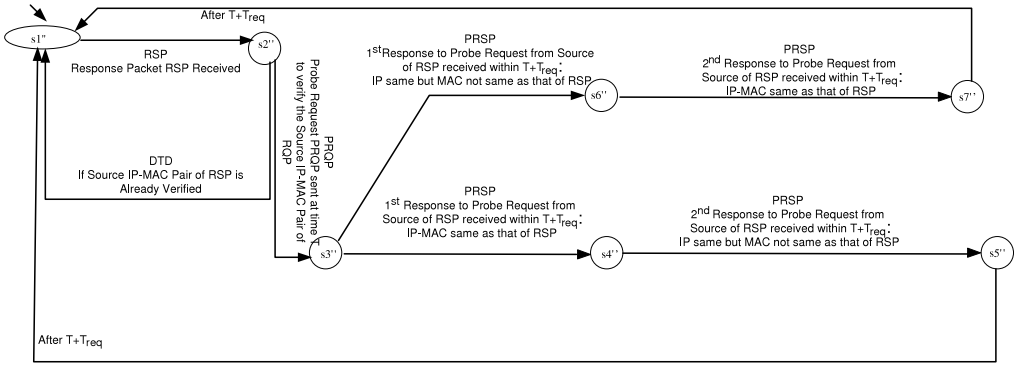
The terms in the DES required for modeling the LAN under normal and attack conditions are quantified as follows:

$\Sigma = \{RQP, RSP, PRQP, PRSP, DTD\}$. The set of states S is shown in Figs. 6 and 7. States with no primes correspond to normal situation and those with single and double primes denote request and response spoofing, respectively. Initial states (S_0) are also shown in Figs. 6 and 7. The model variable set is $V = \{IPS, MACS\}$; *IPS* has the domain as $D_1 = \{x.x.x.x | x \in \{1, 2, \ldots, 255\}\}$ and *MACS* has the domain as $D_2 = \{hh - hh - hh - hh - hh - hh | h \in Hex\}$. There is a clock variable y, which is used to determine if

[2] This DES model is obtained by taking features from Timed Automaton [17] and Extended Finite State Machine [18] and augmenting them with its original version [13].



(a) Request spoofing.



(b) Response spoofing.

Fig. 5. State based ARP model under request spoofing and response spoofing.

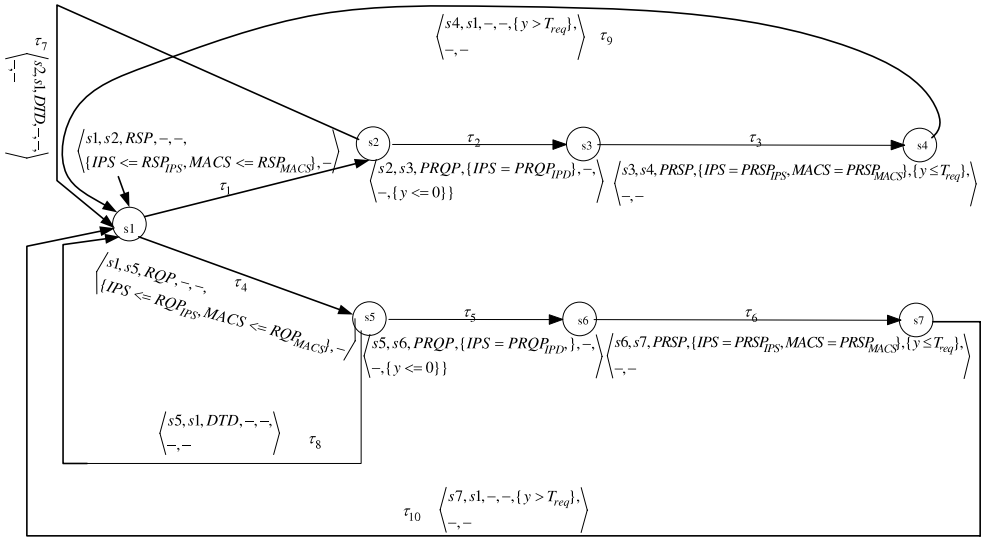
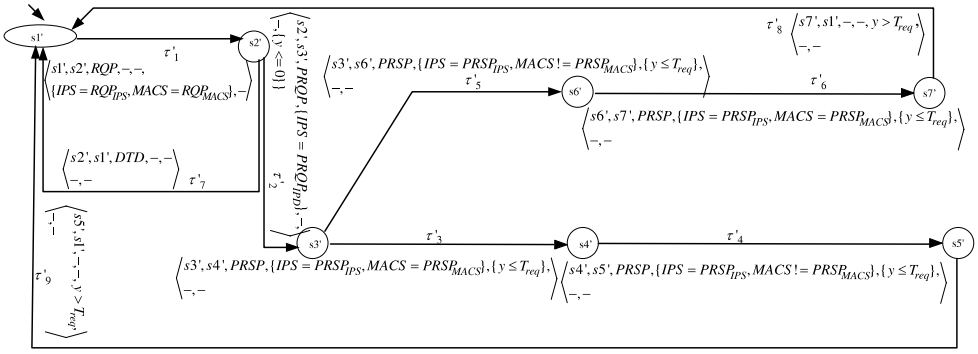


Fig. 6. DES model of LAN under normal conditions.

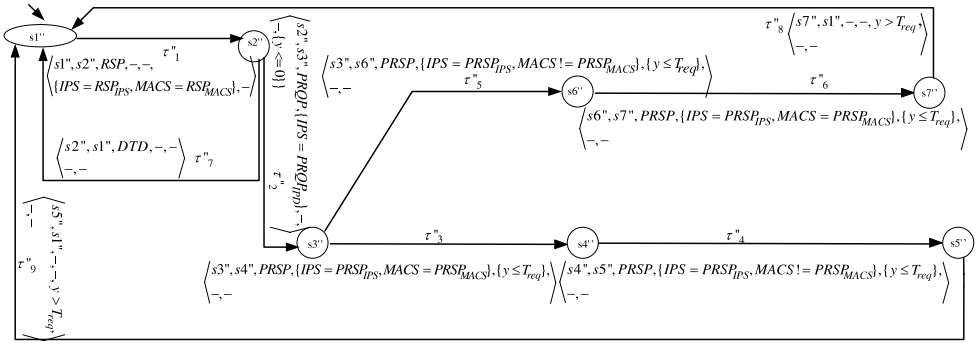
the probe responses have arrived within T_{req} time of sending the corresponding request. The transitions are shown in Figs. 6 and 7; like states, transitions without primes are for the normal model while single (double) primes are for request (response) spoofing. It may be noted that there are "—" for some fields in the tuple representing the transitions. If "—" is for $\phi(V)$ or $\Phi(C)$ then it represents a TRUE condition, while if the "—" is for Reset(C) or Assign(V) then it represents NO action (i.e., reset or assignment) is required.

To maintain brevity we give an overview of the DES model (designed above) for the request spoofing scenario (Fig. 7(a)).

The model moves to state $s2'$ on observation of an event *RQP* by transition $\tau 1'$. Enabling of $\tau 1'$ is only dependent on *RQP* and not on any model variables or clock invariant condition; this is depicted by "—" in the transition. Model variables *IPS* and *MACS* are assigned with source IP and source MAC of *RQP*, respectively. Clock variable y is not altered. Following that in state $s2'$, there are two options, either an ARP probe is sent ($\tau 2'$) or a *DTD* ($\tau 7'$) is received if the IP–MAC pair of the *RQP* is already verified and stored either in the Authenticated or Spoofed tables. Transition $\tau 2'$ is enabled on *PRQP* (sent to IP of the *RQP* under question); correspondence of *PRQP* with *RQP* is determined by checking model variable *IPS*



(a) Request spoofing.



(b) Response spoofing.

Fig. 7. DES model of LAN under request spoofing and response spoofing.

with $PRQP_{IPD}$. Also, clock variable y is reset. After transition $\tau 2'$, there are two options; (i) probe response from the attacker arrives ($\tau 3'$) having $PRSP_{MACS}$ the same as spoofed RQP_{MACS} or (ii) probe response from the normal host arrives ($\tau 5'$) having $PRSP_{MACS}$ not the same as spoofed RQP_{MACS}. It may be noted that the model does not capture which probe response ($PRSP$) is from normal and which is from an attacker. The model only denotes the fact that there are two responses with different MACs; $\tau 3'$, $\tau 4'$ and $\tau 5'$, $\tau 6'$ are the two combinations of arrival of the replies with different MACs. Further, these $PRSP$s are to arrive within T_{req} time after $PRQP$ is sent; this is checked by the clock invariant condition $y \leq T_{\text{req}}$ in transitions $\tau 3'$, $\tau 4'$, $\tau 5'$, $\tau 6'$. In a similar way all transitions can be explained.

The transitions from the renewal states occur after T_{req} time of sending the probes or on receipt of DTD. As discussed, it is assumed that all responses to a probe comes within T_{req} time and so processing needs to be done only on replies that come within that time interval. In the case when no probes are sent (by virtue of verification using tables), the model returns back to the initial state, as no probe responses are to be processed. The sequences of states from $s1'$ to a renewal state (not corresponding to DTD) in this model represents a combination of "RQP, $PRQP$, first $PRSP$, second $PRSP$". In all sequences two $PRSP$s are received with different MACs. Now, if we see a normal DES model (Fig. 6) similar state sequences can be observed but having only one $PRSP$ to the $PRQP$; for example, sequence $s1$, $s2$, $s3$, $s4$ represents the case when RQP is followed by $PRQP$ and a single $PRSP$ is received.

The DES models for the normal and response spoofing cases can be explained in a similar fashion.

2.4. Detector

After development of the DES model, a detector is designed which can identify whether a given sequence of events correspond to a normal or attack scenario. The detector is basically a kind of state estimator of the model which keeps updating the estimate using events of the system (here, LAN). Finally, the detector declares an attack when a state (of the detector) is reached whose estimate comprises states only from the attack model. Once normal or spoofing is determined, details of the packet being processed is recorded in the Authenticated or Spoofed tables. Before the detector is formally discussed the following definitions are introduced:

Definition 1 (*Measurement Equivalent Transitions and States*)**.** Two transitions $\tau_1 = \langle s_1, s_1', \sigma_1, \phi_1(V), \Phi_1(C), \text{Reset}_1(C), \text{Assign}_1(V)\rangle$ and $\tau_2 = \langle s_2, s_2', \sigma_2, \phi_2(V), \Phi_2(C), \text{Reset}_2(C), \text{Assign}_2(V)\rangle$ are equivalent if $\sigma_1 = \sigma_2$ (same event), $\phi_1(V) \equiv \phi_2(V)$ (same equalities over the same subset of variables in V), $\Phi_1(C) \equiv \Phi_2(C)$ (same invariant condition on the clock variables), $\text{Reset}_1(C) \equiv \text{Reset}_2(C)$ (same subset of clock variables are reset) and $\text{Assign}_1(V) \equiv \text{Assign}_2(V)$ (same subset of model variables with same assignment).

If $\tau_1 \equiv \tau_2$ then the source states of the transitions are equivalent and so are the destination states, i.e., $s_1 \equiv s_2$ and $s_1' \equiv s_2'$.

The detector is represented as a directed graph $O = \langle Z, A\rangle$, where Z is the set of detector states, called O-*states*, and A is the set of detector transitions, called O-*transitions*. Henceforth, terms like transitions, states etc. of the DES are termed as model transitions, model states etc. to differentiate them from those of the detector. Each O-state $z \in Z$ comprises a subset of equivalent model states representing the uncertainty about the actual state and each O-transition $a \in A$ is a set of equivalent model transitions representing the uncertainty about the actual transition that occurs. The initial O-state comprises all initial model states. Following that, given any O-state z, the O-transitions emanating from z are obtained as follows. Let $\Im_z$ denote the set of model transitions from the model states $s \in z$. Let A_z be the set of all equivalence classes of $\Im_z$. For each $a \in A_z$, an O-transition is

created comprising all model transitions in a. Then the successor O-state for a is constructed as $z^+ = \{s|s$ is the destination model state of $\tau \in a\}$. Successors for O-states determining attack or normal scenarios are not created. This process is repeated till no new O-transition can be created.

Definition 2 (*Normal Certain O-State*)**.** An O-state, which contains only model states corresponding to the normal situation is called Normal certain O-state.

Definition 3 (*Attack Certain O-State*)**.** An O-state, which contains only model states corresponding to attacks is called Attack certain O-state.

Normal/attack certain O-states denote that the current (model) state estimate comprises only normal/attack states, thereby making a decision. Also as discussed before, all normal/attack certain O-states store the details of the packet being verified in the Authenticated/Spoofed tables. For a given normal or attack certain O-state z say, the packet details to be stored can be found in the event corresponding to the O-transition starting from the initial O-state and (in the path) leading to z.

Now we discuss the algorithm to construct the detector.

Algorithm 3: **Algorithm for construction of detector** O **for the DES model**

Input: DES models for Normal, Request Spoofing and Response Spoofing

Output: Detector for Request Spoofing and Response Spoofing attacks

Begin
$z1 \leftarrow S_0$;
$Z \leftarrow z1$;
$A \leftarrow \phi$;
for all $z \in Z$, which are not normal or attack certain **Do** {
  $\Im_z \leftarrow \{\tau | \tau \in \Im \wedge$ source state of $\tau \in z\}$;
  Find the set of all equivalent classes A_z, of $\Im_z$
  FOR ALL $a \in A_z$ {
    $z^+ = \{s|s$ is destination state of $\tau \in a\}$;
    $Z = Z \cup \{z^+\}$;
    IF z^+ is normal (attack) certain , {
      $E \leftarrow \sigma, \sigma$ is the event for model transition $\tau \in a$ where a is the O-transition emanating from $z1$ and in the path leading to z^+.
      Add E_{IPS}, E_{MACS}, E_{IPD}, E_{MACD} and Time of receipt of E to *AUTHT* (if z^+ is normal certain) else to *SPOOFT*
      } /* Entry of tables for certain states */
    $A = A \cup \{a\}$;
  } /* For all $a \in A_z$ */
} /* for $z \in Z$ which are not certain */
End

Fig. 8 illustrates the detector for the DES models in Figs. 6 and 7. Some of the initial steps for this detector are as follows.

- The initial state of the detector i.e., $z1$ comprises all initial model states $s1, s1', s1''$.
- $\Im_{z1} = \{\tau 1, \tau 1', \tau 1'', \tau 4\}$ which are all the outgoing model transitions from model states in $z1 = \{s1, s1', s1''\}$. Now, $A_{z1} = \{\{\tau 4, \tau 1'\}, \{\tau 1, \tau 1''\}\}$, as $\Im_{z1}$ is partitioned into two measurement equivalent classes namely, $\{\tau 4, \tau 1'\}$ and $\{\tau 1, \tau 1''\}$. Corresponding to $\{\tau 4, \tau 1'\}$ there is an O-transition $a7$ while corresponding to $\{\tau 1, \tau 1''\}$ there is another O-transition $a1$.
- The destination O-state corresponding to $a1$ is $z2 = \{s2, s2''\}$ as the destination model state for $\tau 1$ and $\tau 1''$ is $s2$ and $s2''$, respectively.

In the detector, states $z5, z6, z11, z12$ are attack certain O-states and $z7, z13$ are normal certain states. Now detection of an attack scenario and a normal scenario is discussed.

Attack detection scenario

Let the detector reach state $z6$ by the sequence $z1, z2, z3$. In all these states (i.e., $z1, z2, z3$) the attack or normal condition cannot be determined as the state estimate comprises one normal model state and one attack model state, e.g., $z2$ has $s2$ and $s2''$. $z6$ being an attack certain O-state, reached by occurrence of $a5$ (by virtue of τ_5''), declares an attack. It may be observed from Fig. 7(b) that τ_5'' corresponds to the *PRSP* for a *PRQP* (τ_2'') within T_{req} time, whose source MAC ($PRSP_{MACS}$) is different from the source MAC of the ARP response (RSP_{MACS}) being verified. Now we put the details of the packet being verified in *SPOOFT*. $a1$ is the O-transition emanating from $z1$ and is in the path leading to $z6$. The event corresponding to $a1$ (same as τ_1/τ_1'') is *RSP*. So, in $z6$ we put RSP_{IPS}, RSP_{MACS}, RSP_{IPD}, RSP_{MACD} and time of receipt in *SPOOFT*.

Normal identification scenario

Let the detector reach state $z7$ by the sequence $z1, z2, z3, z4$ (involving no normal/attack certain states). $z7$ being an normal certain O-state, reached by occurrence of $a6$ (by virtue of τ_9), declares the packet being verified as normal. It may be observed from Fig. 6 that τ_9 corresponds to the fact that only one *PRSP* (τ_3) has arrived for the *PRQP* (τ_2) within T_{req} time, whose source MAC ($PRSP_{MACS}$) is the same as the source MAC of the ARP response (RSP_{MACS}) being verified. Now, similar to the attack detection scenario, RSP_{IPS}, RSP_{MACS}, RSP_{IPD}, RSP_{MACD} and time of receipt are updated in the Authenticated table.

In a similar way, the whole detector can be explained.

For the example discussed in Section 2.2, the Authenticated table is blank and the Spoofed table is as shown in Table 2.

Request/response spoofing attacks discussed above can lead to other attacks like man-in-the-middle and denial of service. To verify the occurrences of such cases, once request/response spoofing is determined by the detector, the "Man-in-the-middle detector" algorithm and "Denial of service detector" algorithm are called. These two algorithms are detailed in the next sub-section.

2.5. Detection of man-in-the-middle and denial of service using Authenticated and Spoofed tables

An attacker D performs a man-in-the-middle attack between two hosts A and B say, if D can sniff all traffic from A to B and vice versa. To perform this attack D needs to send two spoofed replies

(i) to A with IP(B)–MAC(D) (i.e., $RSP1 : RSP1_{IPS} = \text{IP(B)}$, $RSP1_{MACS} = \text{MAC(D)}$, $RSP1_{IPD} = \text{IP(A)}$, $RSP1_{MACD} = \text{MAC(A)}$) and
(ii) to B with IP(A)–MAC(D) (i.e., $RSP2 : RSP2_{IPS} = \text{IP(A)}$, $RSP2_{MACS} = \text{MAC(D)}$, $RSP2_{IPD} = \text{IP(B)}$, $RSP2_{MACD} = \text{MAC(B)}$). These replies arrive within T_{MiTM} time window.

Due to *RSP*1, all traffic A wants to send B will go to D. D will then forward them to B, thereby sniffing traffic from A to B. Similarly, due to *RSP*2, D can sniff all traffic from B to A. So, for the successful man-in-the-middle attack there are two spoofed responses where the source IP–destination IP pair of one (spoofed response) is flipped in the other and both have the same source MAC address.

Algorithm 4, discussed below, detects if spoofing (already declared by the detector and details stored in the Spoofed table) leads to a man-in-the-middle attack. This algorithm analyzes all the entries in the Spoofed table to check if within T_{MiTM} there are two responses with flipped source IP–destination IP and same source MAC address.

The algorithm first determines a subset of entries of the Spoofed table whose source MAC matches the source MAC of the response under question. Also, only those entries of the Spoofed table are

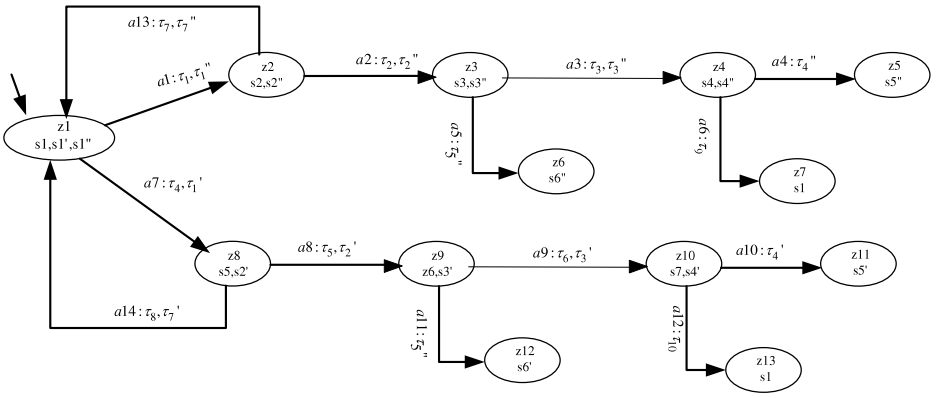


Fig. 8. Detector for the DES model of Figs. 6 and 7.

considered which have arrived within T_{MiTM} time of the arrival of the response being verified. Thus, we obtain a subset of the Spoofed table as $SPOOFT'$. Then, if there is an entry in $SPOOFT'$ with flipped source IP–destination IP (with respect to the response being verified), a man-in-the-middle attack is detected.

Algorithm 4: **Man-in-the-middle detector**
Input : *RSP* - Spoofed response to be verified, Spoofed table
Output: Status
Begin
$SPOOFT' = \{SPOOFT[i] | \forall i\ (SPOOFT_{MACS}[i] = RSP_{MACS})$ && (time of arrival of *RSP* - $SPOOFT_{Ti}[i]) \leq T_{MiTM}\ \}$
IF $(SPOOFT'_{IPS}[j] = RSP_{IPD})$ && $(SPOOFT'_{IPD}[j] = RSP_{IPS})$, (for any j, $1 \leq j \leq SPOOFT'_{MAX}$)
Status= *MiTM* and "attacker is *MACS*"
End

Algorithm 5, discussed below, detects if any denial of service is being attempted on some IP in the LAN. A denial of service attack is said to occur in a host if the number of ARP replies against the IP of the host under question, within a time interval (δ), exceeds a threshold DoS_{Th}. The host will be busy in processing the replies thereby making it unavailable for catering to other normal activities.

This algorithm is executed when the detector determines a *RSP* to be spoofed, to verify if it is involved in denial of service attacks. The algorithm finds all responses from the Authenticated table and Spoofed table which are having the destination IP as RSP_{IPD} and have arrived within δ time of arrival of the *RSP*; the table capturing this information is termed as *DOST*. If the number of such replies is greater than DoS_{Th} (i.e., $|DOST| > \mathrm{DoS}_{Th}$) denial of service is detected.

Algorithm 5: **Denial of service detector**
Input : *RSP*
Output: Status
$AUTHT' = \{AUTHT[i] | \forall i\ (AUTHT_{IPD}[i] = RSP_{IPD})$ &&
(time of arrival of *RSP* - $AUTHT_{Ti}[i]) \leq \delta\ \}$
$SPOOFT' = \{SPOOFT[i] | \forall i\ (SPOOFT_{IPD}[i] = RSP_{IPD})$ &&
(time of arrival of *RSP* - $SPOOFT_{Ti}[i]) \leq \delta\ \}$
$DOST = AUTHT' \cup SPOOFT'$
IF $|DOST| > DoS_{Th}$, Status is Denial of service for RSP_{IPD}

2.6. Analysis of extra ARP traffic

As the scheme uses active probing, some extra ARP traffic is generated. In a switched LAN let there be n hosts among which one is the attacker. If the attacker does not launch any attack then only $2 \cdot n$ extra ARP packets are generated by the active probing mechanism. This is explained as follows. If there is no attack then only once is the IP–MAC pair for each host to be validated which requires one ARP probe being sent and one reply to the probe is received. This checking would generally happen when each host starts up and sends a gratuitous ARP packet. Once IP–MAC is validated and stored in the Authenticated table, no probes are to be sent for other ARP packets with these IP–MAC pairs. Now, if the attacker sends a spoofed packet then the number of extra ARP packets is upper bounded by $3 \cdot k$, where k is the number of spoofed packets; this is elaborated as follows. With each spoofed ARP packet, the 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. However, sometimes no extra traffic is present, if the information about the spoofed packet is found in the Authenticated/Spoofed tables.

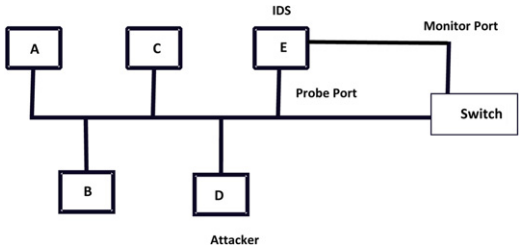


Fig. 9. Configuration of the test bed.

3. Implementation, experimental results and comparison

The test bed created for the experiments consists of 5 machines running different operating systems. The machines were named alphabetically ranging from A–E. Machines A–E had the following OSs in execution: Windows XP, Ubuntu, Windows 2000, Backtrack 4 and Backtrack 4, respectively. Machine D with Backtrack 4 was the attacker and machine E had the detector in execution (i.e., the IDS). These machines were connected in a LAN with a CISCO catalyst 3560 G series switch [19] with port mirroring facility. The configuration of the test bed is shown in Fig. 9. E has two LAN cards–one for data collection in the LAN through port mirroring and the second is exclusively used for sending/receiving ARP probe requests/replies. It may be noted that IP–MAC for the second card corresponds to that of the IDS. Attack generation tools Ettercap, Cain and Abel were deployed in machine D and several scenarios of spoofing MAC addresses were attempted.

The basic block diagram for the (software) implementation of the IDS is given in Fig. 10. As shown in the figure, first the packets obtained by the mirrored port are sniffed by the "packet snipper" module. The "ARP request handler" or "ARP response handler" module is executed depending on whether an ARP request or response packet is snipped. The ARP request handler and ARP response handler are implemented in C language as per the steps discussed in Algorithms 1 and 2, respectively.

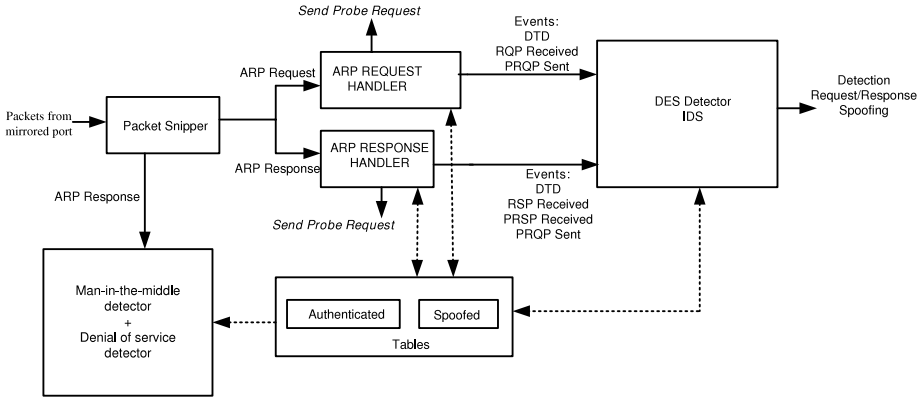


Fig. 10. Block diagram of the implementation of the IDS.

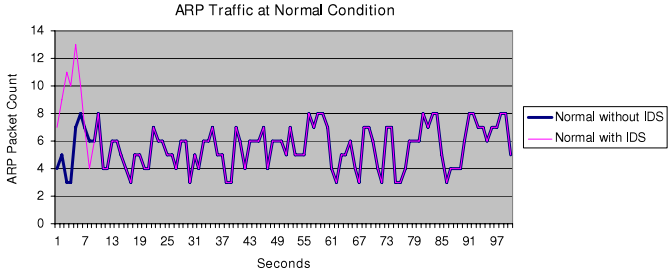


Fig. 11. ARP traffic at normal condition.

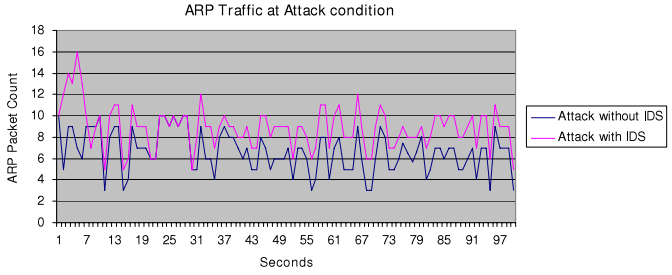


Fig. 12. ARP traffic at spoofed condition.

These modules send ARP probe requests to verify IP–MAC pairs whose genuineness is not yet determined. Further, these modules also generate events when, (i) IP–MAC is already verified to be genuine/spoofed (*DTD*), (ii) request packet is sent, (iii) probe request packet is sent, (iv) probe response packet is received and (v) response packet is received. These events are given to the "DES detector" module. The detector is implemented as a timed automaton [17]. The events, model variables and clock variables of the detector timed automaton are the same as that of the DES model. In the implementation, enabling of the *O*-transitions (based on event, equalities over the subset of model variables and invariant condition on the clock variables), assignment of model variables and resetting of clocks are the same as any one of the model transitions contained in the corresponding *O*-transition. All the attack certain *O*-states give an output of 1, and other states output 0. For each ARP request or response packet received in the snipper an event is generated, which in turn spawns an instance of the detector at state $z1$ and is added to an active list. On the arrival of the next *RQP*, *RSP*, *PRQP*, *PRSP* or *DTD* events, it is given to all instances of detectors in the active list to their corresponding present state. Each of them can make a transition on these events if the enabling condition is satisfied. Any detector that reaches an attack certain or normal certain *O*-state is deleted from the list. Thus, when monitoring ARP behavior with active probes, we create many instances of the detector machine, each of which traces a path; those detecting two *PRSP*s with different MACs to an *PRQP* within T_{req}, flags an alarm for spoofing.

In Fig. 10 it may also be noted that we have two tables namely the Authenticated table and Spoofed table. The "ARP request handler" and "ARP response handler" modules look into these tables to see if a given IP–MAC pair is already tested for genuineness/spoofed. Also, these two modules and the DES detector log data into these tables for packets as and when the genuineness of IP–MAC pairs (of the packets) is verified. Finally, the IDS also has a module termed as "Man-in-the-middle detector + Denial of service detector". This module reads data from the Spoofed table and determines if spoofing has led to other attacks like man-in-the-middle or denial of service. This module is implemented in C as per the steps discussed in Algorithms 4 and 5.

The values of extra ARP traffic (discussed before in terms of n and k) were measured experientially. Figs. 11 and 12 show the amount of ARP traffic generated for 4 scenarios, during the first 100 s of startup of the hosts in the test bed. The number of ARP packets generated in the LAN between t and $t+1$ seconds is plotted against the $(t+1)$th second.

The first case is of normal operation in the absence of the IDS. The second case is when the IDS is running and there are no attacks generated in the network. It may be observed from Fig. 11 that ARP traffic with IDS running is slightly more (at normal conditions) compared to that without IDS mainly during the first few seconds.

Table 4
Comparison of ARP attack detection mechanisms.

Attacks	Proposed	Active [11]	Colasoft [5]	Arpdefender [20]
Malformed packets	Y	Y	N	N
ARP request spoofing	Y	Y	Y	Y
ARP response spoofing	Y	Y	Y	Y
ARP man-in-the-middle	Y	N	N	Y
ARP DoS	Y	N	Y	N

This is because, under normal conditions, active probes need to be sent to verify IP–MAC pairs only once when they occur the first time in the network traffic. All ARP requests/responses sent for periodic updates of ARP caches maintain their IP–MAC pairings (same as those sent at startup); no probes are required for already verified pairs. The third case is when 100 spoofed IP–MAC pairs were injected into the LAN and IDS is not running. The fourth case is when the same 100 spoofed IP–MAC pairs were injected into the LAN with IDS running. From Fig. 12 it may be noted that most of the time a little extra traffic is generated by the proposed IDS.

Table 4 presents the types of LAN attacks generated and detected successfully by the proposed scheme. Also, in the table the capabilities of other LAN attack detecting tools for these attacks are reported.

4. Conclusion and discussions

In this paper a DES detector based IDS for detecting ARP request and response spoofing is proposed. Most of the schemes reported for detecting ARP attacks add stringent constraints like making IP–MAC pairs static, patching of OS in all the hosts, modifying the standard address resolution protocol etc. Further, each scheme can detect only a few types of ARP related attacks. In this paper it is shown how the proposed DES based IDS for ARP attacks detects a near complete class of ARP related attacks yet maintaining the standard of the address resolution protocol. Further, this being mainly a software based approach, it requires only a switch with port mirroring facility as additional hardware.

The scheme is based on the failure detection theory of DES. In the theory first, two types of DES models are required to be generated—(i) capturing the normal condition of the system and (ii) capturing the system under abnormal (failure) condition. Following that a DES detector is designed using the models, which can determine online if the system is under normal or failure conditions. This DES based scheme is adapted for design of the IDS for ARP attacks after some modifications. In traditional systems (where DES is applied) failures can be distinguished from normal behavior by a difference in the sequence of states. However in ARP there is no such difference in the sequence of states in the case of an attack compared to normal condition. So for applying DES theory to ARP attack detection an active probing mechanism is used. The active probing technique sends an ARP probe request to the source IP of any ARP request/response packet if the genuineness of the IP–MAC pair of the request/response packet is not yet known. It may be noted that this active probing mechanism does not violate the standard address resolution protocol. With this probing mechanism there is a difference in packet sequence (normal compared to attack) which is used for DES model development. Further, new parameters namely time and model variables were also augmented in the basic DES framework for efficient modeling of ARP. Finally the detector was designed which was implemented as the IDS for detecting ARP attacks.

The IDS was deployed in a test bed and results showed successful detection of a large class of ARP related attacks namely, malformed packets, request spoofing, response spoofing, man-in-the-middle, denial of service etc. Also, experimental results illustrated that the amount of extra ARP traffic that is generated due to the active probing mechanism is minimal.

At present the scheme can only detect the attacks. In other words, in the 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 a differentiating 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. 1st ed. John Wiley & Sons, Ltd.; 2003.
[2] Kozierok CM. TCP/IP guide. 1st ed. No Starch Press; 2005.
[3] Cisco 6500 catalyst switches. http://www.cisco.com/en/us/products/hw/switches/ps708/.
[4] Lockhart A. Network security hacks. 1st ed. O'Reilly Media, Inc.; 2007 [Chapter 6].
[5] Colasoft capsa. http://www.colasoft.com.
[6] Roesch M. Snort—lightweight intrusion detection for networks. In: USENIX conference on system administration. 1999. p. 229–38.
[7] Abad CL, Bonilla RI. An analysis on the schemes for detecting and preventing ARP cache poisoning attacks. In: International conference on distributed computing systems. IEEE; 2007. p. 60–7.
[8] Hsiao H-W, Lin CS, Chang S-Y. Constructing an ARP attack detection system with SNMP traffic data mining. In: International conference on electronic commerce. ACM; 2009. p. 341–5.
[9] Gouda MG, Huang C-T. A secure address resolution protocol. Computer Networks 2003;41(1):57–71.
[10] Lootah W, Enck W, McDaniel P. Tarp: ticket-based address resolution protocol. In: Annual conference on computer security applications. IEEE; 2005. p. 106–16.
[11] Ramachandran V, Nandi S. Detecting ARP spoofing: an active technique. In: International conference on information systems security. Springer Verlag; 2005. p. 239–50.
[12] Trabelsi Z, Shuaib K. Man in the middle intrusion detection. In: Globecom. IEEE; 2006. p. 1–6.
[13] Cassandras CG, Lafortune S. Introduction to discrete event systems. 1st ed. Kluwer Academic Publishers; 1999.
[14] Sekar R, Bendre M, Dhurjati D, Bollineni P. A fast automaton-based method for detecting anomalous program behaviors. In: Symposium on security and privacy. IEEE; 2001. p. 144–55.
[15] Whittaker S-J, Zulkernine M, Rudie K. Towards incorporating discrete-event systems in secure software development. In: International conference on availability, reliability and security. IEEE; 2008. p. 1188–95.
[16] Hubballi H, Biswas S, Roopa S, Ratti R, Nandi S, Barbhuiya FA, et al. A DES approach to intrusion detection system for ARP spoofing attacks. In: Mediterranean conference on control and automation. IEEE; 2010. p. 1–6.
[17] Alur R, Dill DL. A theory of timed automata. In: Theoretical computer science. Elsevier; 1993. p. 183–235.
[18] Cheng K-T, Krishnakumar AS. Automatic functional test generation using the extended finite state machine model. In: International design automation conference. IEEE, ACM; 1993. p. 86–91.
[19] Cisco 3560 catalyst switches. http://www.cisco.com/en/us/products/hw/switches/ps5528/.
[20] Arpdefender. http://www.arpdefender.com.