Algorithmic Support for Distributed IDS in MANETs

Paulo M. Mafra¹, Joni da Silva Fraga¹, Altair O. Santin²
¹Automation and Systems Department, Federal University of Santa Catarina, Florianopolis, SC, Brazil
²Pontifical Catholic University of Parana (PUC-PR), Curitiba, PR, Brazil

Abstract—The intrusion detection systems (IDS) are usually designed to work on local networks. However, with the development of mobile networks and their applications, it became necessary to develop new architectures for IDSs to act on these networks in order to detect problems and ensure the correct operation of data communications and its applications. This paper presents a distributed IDS model for mobile ad hoc networks that can identify and punish those network nodes that have malicious behavior. In this paper we describe the proposed model, making a comparison with major efforts in the literature on distributed intrusion detection systems for mobile ad hoc networks and present the results of tests obtained with an implementation of the proposed model.

Keywords: Mobile Ad Hoc Networks; Intrusion Detection System; Distributed Algorithms;

1. Introduction

The last decade has witnessed a great evolution in communication technologies. Among these technologies are the Mobile Ad Hoc Networks, which form highly dynamic environments without the presence of concentrator units [1]. However, such technology is susceptible to a great variety of attacks by faulty or malicious units. The challenge that arises is to maintain the MANETs free from the activity of malicious or faulty nodes. In the face of the difficulty of avoiding the effects of malicious activities, mechanisms are necessary to at least minimize such effects. Intrusion detection systems (IDSs) can be used as one of these mechanisms [2]. The key question in these IDSs is to ensure that applications in mobile Ad Hoc environments can always evolve despite of failures, attacks by malicious entities or their own mobility. Works in this area deals mainly with problems applied to routing protocols in MANETs [2] and some works focus on problems of malicious behavior [2], [3], [4]. These works are limited to monitoring the communications in MANETs and the proposed IDS models usually have centralized elements and do not admit intrusions in their elements. There are no mechanisms to protect their own information.

The communication among entities in MANETs can be monitored by these entities in order to detect faulty or malicious behavior and to improve safety and reliability of these environments. In this paper we propose a secure and fully distributed IDS model for mobile Ad Hoc networks. We apply concepts of distributed systems and dependability.

The use of these concepts allows, within certain limits, the construction of an IDS less subject to restrictions than those present in other IDS models. The algorithms presented in this paper define an IDS with functions performed by groups of components (fully distributed executions) and with mechanisms and techniques that are tolerant to malicious activities of their own components. In section 2 we describe the IDS organization. In section 3 we introduced the algorithmic base that supports our secure IDS. In section 4 we present an analysis of the costs involved in the communication of IDS nodes and some results of performed tests in a simulator. Following (section 5), related work are presented. We finally present our conclusions.

2. Architecture Description

Communication protocols in MANETs depend upon collaboration from equipment which form these network nodes. As such, we also assume this collaborative environment. However, we do not limit this collaboration. I.e. all the network nodes participate in the intrusion detection. The proposed IDS model is distributed and should assume a hierarchical stratification in order to attend the diverse IDS functions. The differentiation of functions and their distribution among nodes in this collaborative environment establish two classes of nodes: the “leader nodes”, which perform higher level functions such as analysis; and the “collector nodes”, which assume lower level functionalities such as sensors, collecting data for future analysis. We also assume that all the network nodes possess at least 2f+1 neighbors, where f is the failure or intrusion limit that our algorithms should support¹.

2.1 Topology and Component Description

The hierarchical topology of the model introduces the idea of clusters, as well as in [2], [3]. However, in our model, we consider each cluster to have various “leaders”. This various leaders form what we denominate as “leadership”. The leaders are chosen by its connectivity and available energy, but if it is necessary to maintain the number of 2f+1 leaders, any node could be chosen as a leader. These leaders

¹This limit f will be used in our approach of a threshold for the occurrence of abnormalities which, once they are not surpassed, the IDS and its distributed functions will continue to supply the correct and expected behavior. At this limit, malicious and faulty nodes present in the cluster and further, departures or churn during a predefined period are accounted for at a given instant named epoch_time.
which form a leadership in a cluster use “secure channels” to exchange information and alerts among themselves. The collecting nodes send a summary of the data collected to the leaders also through “secure channels”. The leadership which define the domains of a cluster are instituted based on \(2f + 1\) leaders. Leadership and, consequently, the corresponding cluster no longer exist as of the moment in which a leadership has less than \(2f + 1\) leaders.

The collecting nodes constitute the largest part of the cluster. In order to belong to a cluster, the collecting nodes need to possess secure channels with at least \(f + 1\) leaders of the cluster. The data collected by the collectors always considers its neighborhood, i.e., the collecting nodes capture the data from each neighbor and store the information in them in a table. This information may be, for example, the quantity of packets received and sent by the neighboring node \(i\). The leader nodes analyze the data sent by the collecting nodes related to the same cluster. Based on the analysis of this monitoring information, they make their decisions concerning malicious nodes. These decisions are in turn registered in local lists (ex. malicious nodes list) and later shared, compared, and synchronized with the remaining leader nodes which make up the same leadership which determines the existence of the corresponding cluster. Thus, it is not necessary consensus between the leaders of a cluster. Since the leadership has at least \(2f + 1\) leaders, the same choice of \(f + 1\) leaders about the behavior of a given node is sufficient for this decision to be considered by the leadership.

### 2.2 Cryptographic Mechanisms

IDS communication, which occurs among (collector - leader) and (leader - leader) cluster nodes makes use of cryptography. Encrypting messages done with the use of session keys and symmetric encryption. Key distribution involves the public keys (pair of asymmetric keys) for each participating node. The public key of this pair is signed by the node which is running it. All the existing routes from the node that is running it. We adapted the DSR algorithm to update the local routing cache, inserting \(f + 1\) disjoint routes to the leadership. This adapted algorithm (named RouteUpdate()) updates the list of neighbors and the list of leaders from the node that is running it. All the existing routes from \(i\) to a leader \(l\) are obtained (through the Route Discovery mechanism in the DSR). Once the routing cache is updated, the list of neighbors from \(i\) is generated. Each neighbor node is extracted from this routes.

### 3. System Dynamics

Collectors send data monitoring summaries to leadership. Such data is sent upon concluding a time period called the transmission time \((T_t)\). The leaders analyze the data sent from collectors at the end of each \((T_t)\). We call epoch the periods in which each cluster “freezes” its composition. In other words, during one epoch it is assumed that the composition of the cluster does not change. At each epoch there are several data collections (several \(T_t\)'s). The transmission time \((T_t)\) occurs \(n\) times in each epoch \((T_t = \text{epoch}/n)\).

![Temporal relation among updating rounds](image)

The changes which occur in the system during this period are not updated. I.e. possible system changes (within an epoch) such as faults, node entrances or departures, node exclusion, etc. are not taken into consideration in composing the cluster during that time. At the conclusion of each epoch, the cluster must synchronize itself. Thus, the updating round (UR) is initiated. These updating rounds, also present in [6], define a time period where cluster leaders exchange information in order to update their knowledge concerning the present state of the cluster. Therefore, in each synchronizing period (one updating round), the changes that occur in the cluster at last epoch are considered. Then, the composition for the new epoch is defined. During these updating rounds are also defined new roles for cluster nodes. The decisions taken during updating rounds will always depend upon the agreement of \(f + 1\) leaders. The ratio of these periods of time is illustrated in Figure 1. At the end of the UR a new epoch is initiated.

### 3.1 Route Discovery and Update

In this model we adopted the DSR routing protocol [7], an on demand routing protocol widely used in MANETs with a local routing cache. We adapted the DSR algorithm to update the local routing cache, inserting \(f + 1\) disjoint routes to the leadership. This adapted algorithm (named RouteUpdate()) also updates the list of neighbors and the list of leaders from the node that is running it. All the existing routes from \(i\) to a leader \(l\) are obtained (through the Route Discovery mechanism in the DSR). Once the routing cache is updated, the list of neighbors from \(i\) is generated. Each neighbor node is extracted from this routes. If the node that is running this...
algorithm is a leader, it attempts to get the maximum number of disjoint routes to other leaders as possible \((f+1)\) at least. Otherwise, it gets just \(f+1\) disjoint routes\(^2\).

### 3.2 Data Analysis

Data analysis is performed in each updating round. This analysis takes into account available data from neighbors of a node and is implemented based on the system Octopus IDS [8]. Each leader makes the analysis of each network node based on the data that it received from collectors. If the statistical analysis of \(f+1\) nodes points node \(i\) as suspect, then it is considered suspect by the corresponding leader.

This analysis is done by all the leaders present in a leadership. The comparison (through the exchange of encrypted and authenticated messages) among the results obtained in analysis from these leaders is made concrete as well as in the leadership. The results obtained by these comparisons will be considered by the cluster as a whole. With \(f+1\) leaders agreeing upon the analysis results, these results will be considered by the leadership. If there are at least \(2f+1\) leaders, the cluster will always decide upon any analysis result, even in the presence of \(f\) malicious leaders.

### 3.3 Data Dissemination

Each leader will only be connected to a cluster if it possesses routes to at least \(f+1\) leaders of that cluster, just as any node in the network. In order to the messages reach leadership, there must always be dissemination in leadership based on the correct leader. The algorithm 1 describes the steps of the dissemination protocol. In this algorithm, a node \(j\) disseminates a message \(msg_j\) in the \(Leadership_j\) (lines 3-6 of algorithm 1) which corresponds to its knowledge about the leaders that form the cluster’s leadership. Upon receiving the message \(msg_j\), each leader in turn sends it to its respective leadership knowledge (lines 7-11). In this algorithm, at least one correct leader is reached with each resend, which returns to disseminate the message once again, using the recursion of the protocol \(Disseminate()\) [9]. As each leader is connected to at least one correct leader and if the cluster leadership does not form disjointed graphs, then the majority of leaders will be reached through such dissemination.

### 3.4 Synchronizing periods: epoch times, and updating rounds

In order to deal with the dynamic aspects of the network and collecting data for the detection process, it was necessary to define times which determine the synchronization of the actions distributed throughout the system. As we work essentially with time periods, synchronizing the clocks is not necessary for the nodes to initiate synchronized operations.

\(^2\)The collector nodes need to reach just one correct leader that will disseminate the message through the leadership.

### Algorithm 1 Disseminate Protocol

- **On Initialization**: 
  - \(Received_k \rightarrow \{\}\)

- **On Disseminate\((msg_j, Leadership_j)\) at node \(j\)**
  - \% for all \(l_k \in Leadership_j\), do
    - \% \(msg_j\) is send to leaders \(l_k\) known by node \(j\)
    - \% \(send <DISSEMINATION, msg_j > \rightarrow l_k\)
  - end for

- **On Receive\(<DISSEMINATION, msg_j > \) at \(l_k\)**
  - \% \(msg_j\) to \(l_k\) is received from different leaders
  - \% \(send <DISSEMINATION, msg_j > \rightarrow l_k\)
  - \% \(msg_j\) locally delivered in \(l_k\)
  - \% \(msg_j\) send again a \(DISSEMINATION\)

Using their local clocks, the periods are controlled with their respective deadlines with timers which aid corresponding operational activation. The routine presented in Algorithm 2 is used to initiate a synchronized common activity among network nodes. It depends upon the course of the stipulated time and reception of at least \(f+1\) corresponding sync messages.

The idea of this algorithm is to stipulate a period \(d\) and at the end of this period to start sending a \(sync\) message to the leadership (lines 4-7 of algorithm 2). Upon receiving this message, the leader saves it and checks how many \(sync\) messages he has received from different leaders. If the number of messages is greater than \(f+1\) then this leader sends again a \(sync\) message for the other leaders and collectors in order to force them to get in synchronizing period, if they still have not received \(f+1\) messages (lines 8-13). After this last send, the sender may start participating in the updating round (UR) (line 14). At this time we also synchronize collector nodes, synchronizing the transmission times (Tts) (line 17).

### Algorithm 2 Synchronizing() at node \(i\)

- **Require**: \(receive <sync_i >\) or period \(d\) elapsed in \(N\)

  - **Init**
    - \(\delta \leftarrow \text{time()}\) \% starts a new period counting \(d\)
    - \% \(\forall \leftarrow \text{time()}\) \% at the end of \(d\)
    - \% \(\forall \leftarrow \text{time()}\) \% if \(i\) is leader
  - **Do**
    - \% \(\text{send of messages } \text{sync} \text{ by } i\)
    - \% \(\delta \leftarrow \text{time()}\)

  - **Upon receive\((\text{sync}_i)\)**
    - \% \(\text{sync} \text{ messages received by } i\)
    - \% \(\text{send of messages } \text{sync} \text{ by } i\)
    - \% \(\text{call leaders to synchronize}\)
    - \% \(\forall \leftarrow \text{time()}\)

  - **Else if** \(\forall \leftarrow \text{time()}\) \% new synchronized \(U\text{R}\) is started

### 3.5 Transmission times

Algorithms for the activities corresponding to transmission time (Tt) periods and updating round (UR) periods were defined. In the \(\text{Tt}()\) algorithm, a deadline in which each
This view is composed by a set of lists (malicious nodes, where the cluster nodes update their views about the cluster. Upon receiving the message, the leader node verifies if the message is not older and saves it (lines 12-14). If the number of received messages in this (Tt) were greater than or equal to the number of collectors minus (lines 17-21). In such case, this suspected node is inserted received data if the node is suspect by most of its neighbors (lines 17-21). In such case, this suspected node is inserted into the list of suspects from that leader node (lines 19-20).

Algorithm 3 $T_t(monitoring\_data_i, T_{t_j})$

1: $Init:$
2: \( \gamma \leftarrow time() \)
3: \( T_{t_i} \leftarrow 0 \)
4: \( Collected\_data_i \leftarrow \emptyset \) % buffer for monitoring data
5: \( suspects\_List_i \leftarrow \emptyset \)
6: \( timeout \leftarrow p/3 \) % p is the value of one $T_t$
7: \( upon((time() - \gamma) \geq p) \) at node $i$ do % p is the period between two $T_t$
8: if \( (i \in Collectors) \) then
9: \( Disseminate(< monitoring\_data_i, T_{t_j} >, Leadership) \)
10: \( \gamma \leftarrow time() \)
11: end if

% leader $l$ receives data from collector $i$
12: \( upon receive(< monitoring\_data_i, T_{t_k} >) \) at $l : l \in Leadership$ do
13: if \( T_{t_k} = T_{t_l} \) then
14: \( Collected\_data_l \leftarrow Collected\_data_l \cup \{ < monitoring\_data_i, T_{t_k} > \} \)
15: if \( | Collected\_data_l | \geq 2f + 1 \) then
16: if \( | Collected\_data_l | \geq | Collectors | - f \lor (\gamma(t ime() - \gamma) \geq timeout) \) then
17: for all node $n \in Cluster(Leadership)$ do
18: if \( analysis_i^n \leftarrow Data\_Analysis(Collected\_data_i, id_n) \)
19: \( if analysis_i^n \in is\_suspect \) then
20: \( suspects\_List_i \leftarrow suspects\_List_i \cup \{ id_n \} \)
21: end for
22: \( T_{t_i} \leftarrow T_{t_i} + 1 \)
23: \( Collected\_data_i \leftarrow \emptyset \)
24: \( timeout \leftarrow EstimatedTime(timeout) \)
25: end if

In this system, for the purpose of estimating the time when the majority of messages sent by collectors arrive at leader nodes, we apply an adaptive timeout that can auto-adjust over time. This adaptive timeout is based on the timeout used in TCP protocol proposed by Jacobson [10]. In it, each sent message involves the sender estimating a time interval to receive of an acknowledgment from the destination. We call this algorithm $EstimatedTime()$. In this algorithm the observed error from last timeout was first calculated and the next value for the timeout then estimated.

3.6 Updating rounds

An updating round (UR) is defined as a period of time where the cluster nodes update their views about the cluster. This view is composed by a set of lists (malicious nodes, leaders, collectors, etc.). During these URs, the information about suspected nodes is shared among cluster leaders. In an UR node entrances and departures are also processed. Following, we explain how UR() algorithm works.

In UR() algorithm the list of suspects generated from last epoch during (Tt) is sent to other leaders through the Disseminate() protocol. Upon receiving this list, the leader node checks if the message is not older. If the number of received messages is greater than or equal to $2f + 1$, then the leader node executes the function $identifies\_Suspect()$ in which an analysis of each node is performed searching for nodes reported to be suspicious by at least $f + 1$ leaders. These nodes are included in the malicious node’s list of the leader. In the sequence, the leader node disseminates a message asking for the election of a new coordinator. These messages are saved and when the number of messages reaches $2f + 1$, a leader is elected as coordinator.

The coordinator then processes the node entrance and departure requests in the cluster, generates a new view of the cluster (with new nodes) and spreads this view to all the nodes in the cluster. Upon receiving the new view of the cluster, if the node is a leader, it checks the message validity. If the message is valid then the node updates its view starting a new epoch. If it is not valid, a message asking for new coordinator is spread throughout leadership again. If the node that received the new view is a collector, the message signature is verified and its list of neighbors is updated.

3.7 Identification, entrance, and departure of nodes

The node identification process in the network should be secure enough so that it cannot create multiple identities. Thus, it is possible to prevent attacks like the Sybil [11]. In our model, node identification is carried out with the use of certificates. A certifying authority (CA) considered to be known and reliable by the network nodes generates a certificate for the public key for each node when it joins the system. The role of the CA in our model is assumed by a certifying entity\(^3\). With this model, we may define the user and his/her equipment. There will not be users using multiple equipment on the network.

Algorithm 4 presents the steps involved to insert a new node within a cluster. Upon entrance, a new node $i$ must broadcast an entrance request message (REQIN). A neighbor, upon receiving REQIN, disseminate it to the leadership. This message informs its identification (node id) and its credential (its public key in certificate form) (lines 2-9 of algorithm 4). The new node $i$, upon receiving the new view of the leadership verifies the signature and initializes its view of the cluster, assuming the role assigned to it by the

\(^3\)This certifying entity will not necessarily need to be an official PKI. It may be a system management commission, an administrator, etc.
leadership. The list of neighbors will be formed by the intersection of the list of neighbors, the list of collectors and the list of leaders of the new view, excluding the nodes that were identified as malicious (lines 10-23).

Algorithm 5 Departure of node $i$

1: (On Departure()) at node $i$
2: Disseminate(<REQOUT, id$_i$, cred, tUR$_i$>, Leadership$_i$)
3: upon receive <REQOUT, id$_i$, cred, tUR$_i$> at node $l$ do
4: if $l$ ∈ Leadership$_i$ then
5: leaving_nodes$_i$ ← leaving_nodes$_i$ ∪ {<REQOUT, id$_i$, cred, >}
6: end if
7: end if
8: % node $i$ waits a response from leadership in next UR
9: upon receive <view, tUR$_i$> at node $i$ do
10: if VerifiedSignature(certi, : certi ∈ view.certificates) then
11: role = leader
12: else if $i$ ∈ view.Col then
13: role = collector
14: end if
15: if $i$ ∈ view.Lead then
16: Leadership$_i$ ← view.Lead
17: Collectors$_i$ ← view.Col
18: RouteUpdate()
19: CN$_i$ ← listNeighbors$_i$
20: CN$_i$ ← CN$_i$ ∩ (view.Col ∪ view.Lead ∪ view.black_list)
21: tUR$_i$ ← tUR$_i$
22: end if
23: end if

Algorithm 5 presents the steps that involve the departure of a node from the system. In this algorithm, the node that want to leave the cluster, disseminate a message REQOUT to the leadership. Upon receiving the message, each leader adds the message in a list of leaving nodes. In the next UR, when the node that want to quit receive the new view, it checks the signature and can then leave the system. The entrances and departures of the system are always considered during periods of updating rounds. Otherwise, the node will be considered as a malicious node.

4. Results and corresponding analysis

4.1 Communication Costs

The communication costs of the algorithms proposed in this model depend principally upon the size of the cluster and its leadership. As such, we calculate these costs in terms of the messages sent from each algorithm. Table 1 presents these costs for each algorithm, in which: $n$ is the number of nodes; $l$ is the number of leaders in the cluster; $c$ represents the number of collectors in the cluster. We adopted the limit of $f = (l-1)/2$. In other words, the number of malicious or faulty nodes is half minus one of the cluster’s leader nodes (it is the limit of anomalies that the IDS supports to operate properly). In this calculation, we also assume the costs of the cryptographic mechanisms, presented in [6].

<table>
<thead>
<tr>
<th>Table 1: Communication costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entrance of a node in the network</td>
</tr>
<tr>
<td>Transmission time (Tt)</td>
</tr>
<tr>
<td>Updating Round (TUR)</td>
</tr>
</tbody>
</table>

In the entrance process of a node in the network (Algorithm 4), this node sends a request to his neighbors ($c$ in the worst case). Upon receiving the request, each neighbor node resends the request to the other leaders of leadership using the protocol Disseminate(). The cost of this protocol is given by $D = (f + 1) * (f + 1)$ where $f = (l-1)/2$. In this calculation, we arrive at $D = (l^2 + 2l + 1)/4$. Thus, the total cost for a node gets into the network is then $(c + D)$, or $c + (l^2 + 2l + 1)/4$. In calculating the costs for transmission time (Tt), each collecting node sends a message with monitoring data to $f + 1$ leaders and the first leader to receive the message executes the Disseminate() protocol. As such, we arrive at $c * (f + 1) + D$. Through substitution, we arrive at $cl/2 - c/2 + (l^2 + 2l + 1)/4$. In calculating an updating round, each leader disseminates, into the leadership, a message containing the result of its analysis concerning each cluster node $(l * D)$ or $(l^3 + 2l^2 + 1)/4$.

Necessary messages for 100-node network

Fig. 2: Necessary messages by the number of leaders

In Figure 2, we present an example of the costs (in terms of messages) for the entrance operations of a network node, transmission time, and updating rounds in a 100-node network forming merely one cluster. With the use of communication cost information obtained, it is always possible to find more adequate values for the number of leaders in a cluster. In this example, the ratio between adequate leaders and collectors for a 100-node network

4The role of a new node is defined based on the need of leaders to maintain the cluster, node’s connectivity and available energy.
should be approximately 20 leaders. From the graph, we note that as of 20 leaders, the ratio between the number of necessary messages for updating round operations (UR) grow exponentially. This same behavior is not noticed when the considered algorithms are transmission time (Tt) and entrance of a new node into the network. This algorithms are not as dependent on their costs in terms of messages as the updating round regarding the size assumed for the leadership.

4.2 Implementation and tests

The principle objective of the tests performed in this work was to verify the simulated limits the system would support through the Omnet++ version 4.1 simulator with a Mixim version 1.1 wireless network module. This tests were performed in a 300 x 400 meter rectangular environment. The period for transmission time was established at 60 seconds, and each epoch’s time was set at 300 seconds. These amounts were obtained based on simulator tests, in such a way as to not saturate the system with messages generated during transmission times and updating rounds. The total simulation time was limited to 6010 seconds (20 URs). This total time was found to be sufficient to observe the proposed model’s results. The node mobility rate in mps (meters per second) was defined at 2.0 mps. These values are similar to those employed in [12] where tests with MANETs were performed.

The malicious activity was defined considering that nodes with malicious behavior fulfill their functions in randomly routing messages. In other words, sometimes they transmit, other times they omit messages in routing. This behavior in our simulations follows a uniform distribution, in which 80% of the cases of the messages are discarded by the malicious nodes. In the other 20%, they participated in routing correctly. We chose this type of behavior as it is more difficult to detect than simply a node which discards all the messages it receives, or than nodes which retransmit only to similar (routing messages merely to a list of nodes which possess the same behavior pattern).

Table 2: Results of tests

<table>
<thead>
<tr>
<th>Observed Feature</th>
<th>5% dep.</th>
<th>10% dep.</th>
<th>15% dep.</th>
<th>20% dep.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of messages</td>
<td>5.00%</td>
<td>9.00%</td>
<td>14.53%</td>
<td>15.91%</td>
</tr>
<tr>
<td>Disseminated msg</td>
<td>100%</td>
<td>100%</td>
<td>93.66%</td>
<td>92.00%</td>
</tr>
<tr>
<td>Detection rate</td>
<td>100%</td>
<td>100%</td>
<td>97.13%</td>
<td>97%</td>
</tr>
</tbody>
</table>

In the second and third tests, two distinct conditions were observed, respectively, without malicious nodes and with 10% of the malicious nodes (therefore, respecting the 2f + 1 leader limit in leadership). In these tests, we adopted a fixed entrance of nodes at 10% to each updating round, but varying the departure in 5%, 10%, 15%, and 20%. With these tests, we observed the system’s behavior with the node entrances and departures. The results of these tests are presented in Table 3 and Table 4.

Table 3: Results of tests without malicious nodes

<table>
<thead>
<tr>
<th>Observed Feature</th>
<th>5% dep.</th>
<th>10% dep.</th>
<th>15% dep.</th>
<th>20% dep.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of messages</td>
<td>5.00%</td>
<td>9.00%</td>
<td>14.53%</td>
<td>15.91%</td>
</tr>
<tr>
<td>Disseminated msg</td>
<td>100%</td>
<td>100%</td>
<td>93.66%</td>
<td>92.00%</td>
</tr>
<tr>
<td>Detection rate</td>
<td>100%</td>
<td>100%</td>
<td>97.13%</td>
<td>97%</td>
</tr>
</tbody>
</table>

In the first test, three distinct conditions were observed: without malicious nodes; 10%, 20%, and 30% of the malicious nodes respecting the 2f + 1 leadership limit; and with 20% and 30% of the malicious nodes, but keeping the leadership set at 10% (surpassing the f limit). The results of this test are presented in Table 2. This test showed us that even with the occasioned loss of messages through network problems, node mobility, or malicious activity, the message dissemination rate among leaders remained very close to 100% (even when the f limit was surpassed). Consequently, the malicious activity detection rate was rather high. As such, even exceeding the limits supported by the presently proposed model, the system continues to work without the guarantee of message delivery.

Table 4: Results of tests with 10% of malicious nodes

<table>
<thead>
<tr>
<th>Observed Feature</th>
<th>5% dep.</th>
<th>10% dep.</th>
<th>15% dep.</th>
<th>20% dep.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of messages</td>
<td>11.94%</td>
<td>11.71%</td>
<td>29.46%</td>
<td>22.71%</td>
</tr>
<tr>
<td>Disseminated msg</td>
<td>100%</td>
<td>100%</td>
<td>93.32%</td>
<td>93.65%</td>
</tr>
<tr>
<td>Detection rate</td>
<td>100%</td>
<td>100%</td>
<td>97.07%</td>
<td>96.73%</td>
</tr>
</tbody>
</table>

Through these tests, we were able to observe that system message loss was greater with 10% malicious nodes than without any. However, the rate of messages spread throughout leadership was similar (with 10% maliciousness, and without any). We also observe that with a departure rate greater than its entrance rate, some messages were not widespread among leadership. This may be explained by the low network density over time. With such density, the nodes possess fewer connections. As a result, some messages may not arrive to any correct leader (the probability of this happening increases with reduced network density). In the detection rate factor, one observes behavior similar to message diffusion rates. In the same manner, if some messages are not spread throughout leadership, the detection system will end up erring. Finally, we judge the set of tests to be satisfactory and that it evidences the limits and effectiveness of our proposals.
5. Related Studies

Intrusion detection systems for MANETs were proposed in [2]. These systems use clusters to collaboratively detect intrusions. Each cluster possesses a leader which monitors all the traffic within its cluster. These studies have not used cryptography in message exchange, thus making it possible for various types of attacks to occur in the intrusion detection process. Nor have these systems assumed their leaders were alone in their clusters, with malicious behavior.

In another IDS [13], the node which detects suspect activity requests opinions from its neighbors concerning this suspect activity. After analyzing each neighbor’s vote, the node makes a decision and informs it to the participating nodes who voted. However, this voting mechanism is vulnerable to message violation from and collusion with malicious nodes. In another study [14], a node hierarchy organizational model was developed on various levels, where the lowest level collects the data and the higher levels correlate the data sent to them. This study, to the contrary of the others cited here, permits the detection of several malicious nodes at the same time. However, the malicious nodes may only belong to the lower levels of the proposed hierarchy. In our proposal, any node may have malicious behavior, whether leaders or collectors. The only limitation in our model is that the number of malicious nodes cannot exceed f.

Studies concerning IDS for MANETs show that the majority of the systems proposed are capable of identifying few types of attacks or some routing protocol problems for these networks [4]. In our proposal we adopted a detection model based on anomalies. Thus we are able to identify and neutralize a large set of types of attacks and routing problems described in literature. Just as the architectures presented in [2], [14], our model also assumes a hierarchical stratification. In these models, the hierarchical topology introduces the idea of clusters. The majority of studies in literature do not deal with the entrance aspects, departure aspects, or node mobility within the network. In no related study were we able to find simulated test results or real environment test results. In [2] a time period was established for the network to reorganize itself, in which the leaders could be re-elected through a voting process. Merely some of the IDS presented ([15], [14]), indicate the use of cryptographic mechanisms to secure properties such as authenticity, confidentiality, and the integrity of messages exchanged between the IDS nodes.

The greatest contribution of this study, separating it from others present in literature, is the use of distributed systems concepts and dependability concepts applied to an IDS model for MANETs. The use of these concepts permitted - within certain limits - the development of a model less subject to restrictions. The proposed system is able to deal with various faulty or malicious nodes without there being interference in the network’s normal behavior. IDSs normally developed for MANETs, in the literature do not have mechanisms to protect their own information and do not tolerate intrusions into its various components. The approach introduced in this paper aims to build an IDS infrastructure that tolerates malicious actions. Beyond this, our system is able to identify a large number of different attacks or variations of known attacks.

6. Conclusions and future study

In this paper, we presented our efforts to develop an IDS model for dynamic environments together with distributed algorithms that support this model. This proposal is centered on a hierarchical malicious behavior detection model for MANETs. This model follows the concepts of dynamic distributed systems, permitting the presence of various non-malicious entities. We presented the complexity of the proposed algorithms in terms of messages. The proposed model permits the correct functioning of the network while the faulty or malicious node limit is not exceeded. However, these tests showed us that even with the f limit exceeded, the system continues to function. In such a case, there is no guarantee that our algorithms always work correctly.

References