Novel Immune-based Framework for Securing Ad hoc Networks

Yasir Abdelgadir Mohamed
Department of Computer and Information Sciences
Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 31750 Tronoh, MALAYSIA
+6053687423
yasir_eym@ieee.org

Azween B. Abdullah
+6053687507
azweenabdullah@petronas.com.my

ABSTRACT
One of the main security issues in mobile ad hoc networks (MANETs) is a malicious node that can falsify a route advertisement, overwhelm traffic without forwarding it, help to forward corrupted data and inject false or uncompleted information, and many other security problems. Mapping immune system mechanisms to networking security is the main objective of this paper which may significantly contribute in securing MANETs. In a step for providing secured and reliable broadband services, formal specification logic along with a novel immune-inspired security framework (I²MANETs) are introduced. The different immune components are synchronized with the framework through an agent that has the ability to replicate, monitor, detect, classify, and block/isolate the corrupted packets and/or nodes in a federated domain. The framework functions as the Human Immune System in first response, second response, adaptability, distributability, and survivability and other immune features and properties. Interoperability with different routing protocols is considered. The framework has been implemented in a real environment. Desired and achieved results are presented.

Keywords
Security, MANETs, specification logic, mobile agent

1. INTRODUCTION
Confidentiality has become the most important factor that determines interaction among people as well as network devices. The users’ fears arise from the wireless features of mobility and loosely connected infrastructure. In mobile ad hoc networks (MANETs), nodes are free to move as they wish, thus the network topology may change in an unpredictable way at unpredictable times. The wireless channels in mobile ad hoc networks have a broadcast nature with two consequences [1]: unless security is enforced with external mechanisms, communication remains insecure. Secondly, a high collision possibility is expected due to overhearing caused by the nodes in the transmitter communication range, and the contention of the shared medium, therefore, decreasing the bandwidth. Furthermore, the malicious node could introduce serious disruptions into the whole hybrid network. The decentralized nature of network control in MANETs makes them more prone to physical security attacks; hence, solutions should follow the same nature and must be distributed as each node at any time could set off the communication range of any other node. On the other hand, the immune system has many features that are desirable for the imperfect, uncontrolled, and open environment in which most computers currently exist. These include distributability, diversity, disposability, adaptability, autonomy, dynamic convergence, anomaly detection, multiple security layers, identification via behavior, no trusted components, and imperfect detection [2]. A wide variety of architectures for the computer immune system have been inspired from the foregoing features. In this research we are trying to emphasize more immune features, components, and processes that can help to contribute to one of the important areas; MANETs security

The rest of this paper is organized as follows: The related works are highlighted in section 2, while the framework upon which the specification is based is depicted in section 3, the logic is specified in section 4 followed by the conclusion and the future work.

2. BACKGROUND AND METHODOLOGY
In [3], a mechanism for detecting unauthorized and compromised nodes in a mobile ad hoc network has been proposed. The detection procedure depends on two phases. In the first phase, an authentication mechanism to determine the node’s identity takes place through using one of the authentication protocols. Phase two depends on an agent that is embedded into all nodes, knows the user’s standard profile, records deviations, gathers and analyzes and audits data locally, and passes a confidence interval to the neighbor.

In [4], a bio-inspired self-organized secure autonomous routing protocol has been proposed for securing wireless sensor networks. The goal is to minimize the initial broadcasting process to save the battery life time. It has been cited that the security enhancement is based on the human nerve barrier system in order to differentiate between a good and malicious node to decide whether it can join the domain or not. The routing mechanism is designed to use some metrics to make security decisions. This work is a proposal and no actual implementation or simulation experiments have been done.

In [5], a hybrid model for network intrusion detection has been presented. The Network Threat Recognition with Immune Inspired Anomaly Detection, or NetTRIIAD, model is divided into an Innate Layer and an Adaptive Layer. Two features are included in the NetTRIIAD antigen: address features (32 bits) and
protocol features (32 bits). The features are derived from IPv4, TCP, and UDP protocols. The danger model signal in NetTRIIAD includes two elements: single feature value and signal level value. In [6], a previous work on mobile ad hoc networks has been extended [7], [8] [9]. The sequences of protocol events are collected in two positions: at the nodes belonging to the route where the packet loss is observed, and during the time close to the packet loss time. Accordingly, they are considered as non-self antigens. Both negative selection and danger theory mechanisms are used to confirm the non-self detection. The system has the potential to be disposable, distributed, self-organized and lightweight, but has not been demonstrated in a realistic ad-hoc network yet.

3. SECURITY FRAMEWORK

3.1 Overview
The work in progress is a continuation of what has been done earlier. The four mobile agents' method had been proposed to secure the ad hoc domain [10]. Due to scalability and bandwidth constraints in ad hoc networks [11], the approach has been revised into a new one that proposed one agent with many roles and functions to secure the ad hoc domain [12]. In the following subsections, more progressive steps on the same concept have been derived.

3.2 Framework
As stated earlier and to synchronize the different immune processes, an agent with different roles and functions that maps different components, processes, and tasks has been introduced. The different studies have proved that the success in mapping more immune processes and features results in a robust security system for the concerned application [13]. In Table 1, some of the immune features have been mapped to the security protocol. This mapping is highlighted in the next section.

Table 1. Mapping immune system to the security framework

<table>
<thead>
<tr>
<th>Immunity</th>
<th>Ad hoc network Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body</td>
<td>Ad hoc Domain</td>
</tr>
<tr>
<td>B-cells</td>
<td>Immune Agent (IA)</td>
</tr>
<tr>
<td>T-cells</td>
<td>Detectors</td>
</tr>
<tr>
<td>Lymphocytes</td>
<td>Nodes</td>
</tr>
<tr>
<td>Memory Cells</td>
<td>Database inside the IA</td>
</tr>
<tr>
<td>Negative Selection</td>
<td>Match-delete process (secure environment)</td>
</tr>
<tr>
<td>Clonal selection</td>
<td>High scored detectors</td>
</tr>
<tr>
<td>Hyper mutation mask</td>
<td>Detectors cover</td>
</tr>
<tr>
<td>Self/ non-self</td>
<td>Frequently occurred/ abnormal pattern</td>
</tr>
<tr>
<td>Danger theory</td>
<td>Observation effects of non-classified pattern</td>
</tr>
<tr>
<td>Tolerisation</td>
<td>Inmatchability with the self patterns</td>
</tr>
<tr>
<td>Self-healing</td>
<td>Recovery process part</td>
</tr>
<tr>
<td>Distributability</td>
<td>Distributed IA copies</td>
</tr>
<tr>
<td>Adaptability</td>
<td>Updated database in the IA</td>
</tr>
<tr>
<td>APC(Antigen Presenting Cell)</td>
<td>The scanning process where the detectors can successfully disclose the non-self</td>
</tr>
</tbody>
</table>

The Main activities carried out by the Immune Agent (IA) to perform the security task can be identified in the following coloured Petri net (CPN) states and transitions illustrated in Figure 1.

![Image](354x623.png)

Figure 1. CPN model for main IA processes

The configuration process, which is the initial phase that leads up to the activation state, will be carried out in two phases; the first phase is in a secured environment which is a secured configuration phase while the second is an application phase. In the first phase, the IA is configured to monitor the traffic in a domain with no internet connection, which is an error free and secured environment. The network set-up sequence structure is depicted in Figure 2 below.

![Image](318x323.png)

Figure 2. Network setup sequence structure

During the secured configuration phase that will be conducted offline with no internet connection, three profiles will be created; self patterns, non-self, and detectors profiles. These profiles represent the basic database located in the IA, updated during the application phase and then distributed later to all copies inside the domain. The designation of the three profiles can be summarized as follows:

- **Self patterns’ profile.** All the protocol header events must be represented in binary strings that have a length of $l$ and are considered as a self pattern $i \in S_f$ (acceptable events). These strings are analogous to peptides in the immune system. To build the self patterns profile (GenesProfile), we aim to divide the protocol header into sub patterns, each of which is 8 bits in length that has been selected to control negative and positive faults. The self patterns can be represented as $S_f = \{s_{f1}, s_{f2}, ..., s_{fn}\}$.

- **Detectors’ profile.** Based on the self patterns, the detectors’ profile is created. In our research, a random function generator is used. Each detector is then compared to the entire self patterns in the self profile; each has the same length as the self. The ones that match any of the existing self patterns will be discarded while the others are stored in the detectors profile. This is called the negative selection
The appropriate matching rule for the matching probability is driven from [16]; the matching probability ($P_m$) for 8-bits detector length is depicted in Figure 3. The $P_m$ equation can be written in the following form where ($L$) is the detector length while ($r$) is the threshold for the matching bits:

$$P_m = 2^{-r} \left(\frac{L - r}{2} + 1\right)$$

As depicted in the figure, as long as the threshold (matching bits) approaches the detector length value, the matching probability becomes high. Evidently, the detector’s length directly affects the detection process. A self pattern might be caught as malicious while a non-self might be treated as good; this is considered as false positive and false negative respectively, which can be reduced by controlling the detection threshold.

- **Non-self patterns’ profile**: Using the generated detectors, a classification to the incoming patterns could be achieved. Since the detectors are generated and trained not to match the self, the matching is expected to come about when non-self patterns are encountered, $N_f = \{n_{f1}, n_{f2},..., n_{fm}\}$.

Soon after the IA gets configured, the training phase in a secured environment starts. The three basic profiles (self, non-self, and detectors) will be created where the negative selection algorithm plays the significant role in validating the detectors’ profile. The profiles are then sent to the next phase where a less secure -but controlled- environment is deployed. The less secure environment here is needed to train the detectors to deal with the shapes of different attacks. Detectors’ production and training are similar to that which takes place in the bone marrow and thymus respectively in the immune system. Clonal selection (Figure 4) is one more immune mechanism that plays an important role in this phase. It is complementary to the role of negative selection. It explains how an immune response is mounted when a non-self antigenic pattern is recognized by a specific type of cell labeled as B-cell [16].
The B-cell proliferates when its receptors bind to pathogens scoring high affinity. The same concept is mapped to our framework by setting a score for the detectors; a detector will be cloned when it attains the score in detecting a certain number of non-self. Since the detector length in our work is set to 8-bits, the maximum tries number to clone a valid detector may not exceed the value depicted in equation (2) where n is the detector’s length, and M is the maximum tries to generate a valid cloned detector. The relation between M and n is illustrated in Figure 5.

\[ M = 2^n \log_2 2 \]  

(2)

![Figure 5. Maximum tries to generate a cloned valid detector](image)

Such a type of calculation is necessary to consider the space and time complexity for the inspired algorithms. The diagram affirms that the space complexity in a worst case is not linear to the number of bits in the detector length.

When the IA is perfectly configured and equipped with the different profiles, it will then be encapsulated and sent to the destination using the routing protocol packets. The next step is the application phase where an IA performs the security task. After the IA gets attached to the guest platform (destination node), it initially stores data recovery files in the agent’s local database and then monitors the incoming packets. The different profiles in the local database help to classify the incoming patterns, consequently a decision on how to treat the patterns will be made. The normal patterns will be permitted while the abnormal will be ignored/ blocked.

3.3 Mapping the danger theory mechanism

The detection probability becomes high because of the detector’s length, therefore, the false positive and false negative is expected to rate high as well. To reduce blocking the self patterns or allow the non-self patterns, the IA is configured to check the node’s system periodically during the communication process and to observe the effects of the incoming patterns of strings. The patterns that have negative effects will be blocked while the ones with positive effects will be received as usual. This concept has been derived from the danger theory concept of the immune system. It has a significant role in protecting the body against different attacks [17, 18]. There still remain the patterns that are not classified (neither self nor non-self) which will be classified as suspect patterns, and reclassified after the periodic system check results.

The output of the application phase will be the updates for the different profiles needed for securing the domain. These updates will be multicasted to all IA copies inside the domain for an improved performance in the future.

3.4 Routing protocol adaptation

Our immune security framework depends on the routing protocol deployed to establish the connection between the sender and the receiver. Since there are many routing protocols in the wireless ad hoc networks, a standard mechanism has to be set to generalize this framework to all routing protocols. As illustrated in Figure 6 and as a description to the protocol adaptation part which appeared in Figure 4, the routing protocol will be examined to determine whether a reactive protocol or a reactive it is.

![Figure 6. Routing protocol adaptation](image)
• **IA distributor**: In contrast to the holder, the distributor has the authority to establish a new federated domain using its own license in case it leaves the federated domain. A node \( x_i \) could be in one of four scenarios:

I. A new node \( x_i \) that intends to join a federated domain \( x_i \rightarrow \chi (T) \) to be either a holder or a distributor according to the conditions that it accepts the license.
II. A node \( x_i \) intends to leave the federated domain \( \chi (T) \setminus x_i \).
III. A distributor node that leaves a federated domain and wants to establish a new domain.
IV. A node \( x_i \) in a federated domain needs to communicate with a node in another federated domain: \( x_i \leftrightarrow x_n (T) \).

The four scenarios can be demonstrated in Figure 7 below.

Figure 7. Different scenarios for the federated domains
Before the specification logic being explained, the different notations used in the logic are clarified in Table 2.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta )</td>
<td>System components</td>
</tr>
<tr>
<td>( \Omega )</td>
<td>Classification process</td>
</tr>
<tr>
<td>( \delta )</td>
<td>Detector</td>
</tr>
<tr>
<td>( \varepsilon )</td>
<td>A change in system</td>
</tr>
<tr>
<td>( \iota )</td>
<td>Connection between two in a secure environment</td>
</tr>
<tr>
<td>( \eta )</td>
<td>Run periodic system check</td>
</tr>
<tr>
<td>( S_r )</td>
<td>Temporary file: suspect patterns file</td>
</tr>
<tr>
<td>( \tau )</td>
<td>A pattern</td>
</tr>
<tr>
<td>( \leftrightarrow )</td>
<td>Block process (BLCK)</td>
</tr>
<tr>
<td>( \iota )</td>
<td>Recovery process (REV)</td>
</tr>
<tr>
<td>( \Phi )</td>
<td>IA activation</td>
</tr>
<tr>
<td>PCKs</td>
<td>packets</td>
</tr>
<tr>
<td>( \rho )</td>
<td>Clone process</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>Checking process</td>
</tr>
<tr>
<td>( U )</td>
<td>Set of all patterns (self + nonself)</td>
</tr>
<tr>
<td>( \Leftrightarrow )</td>
<td>Connection terminated</td>
</tr>
<tr>
<td>( \chi )</td>
<td>Nodes inside a domain</td>
</tr>
<tr>
<td>( \psi )</td>
<td>Detectors set</td>
</tr>
<tr>
<td>( \omega )</td>
<td>Monitoring phase activation</td>
</tr>
<tr>
<td>( s_r )</td>
<td>Self patterns</td>
</tr>
<tr>
<td>( \leftrightarrow )</td>
<td>Save process</td>
</tr>
<tr>
<td>( \leftrightarrow )</td>
<td>Match process</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>Immune agent license</td>
</tr>
<tr>
<td>DrP</td>
<td>Detector Profile</td>
</tr>
<tr>
<td>CS</td>
<td>Connection establishment</td>
</tr>
<tr>
<td>( \Xi )</td>
<td>Broadcast process</td>
</tr>
<tr>
<td>RVR, BLCK</td>
<td>Recover and block processes</td>
</tr>
</tbody>
</table>

1. **Configure Immune Agent (IA):** Training phase:

\[ \text{IA} := < S_f , N_f , DrP, \delta, RVR, \text{BLCK}> \]

(In immune agent configured with the necessary components; self, non-self, and detectors profiles, detection, recovery, and blocking functions)

1.1 **Building self/non-self profiles**:

\[ S/N \text{prf} := < \text{PCKs}, \chi_i , \varepsilon_i > \]

(IA monitors the packets within a traffic between two nodes, no internet)

\[ \text{IA}: \varepsilon_i \leftrightarrow [\omega] \quad \varepsilon_i : = \{a_0 , a_1, \ldots , a_n\}; \]

(Save the frequently occurring patterns as self patterns, each pattern has the fixed length \( l \) and consists of a string of Boolean)

\[ \text{IA}: \varepsilon_i \not\in [\omega] \rightarrow \{N_f \} \]

(Each generated detector matched with the entire self patterns, either deleted or saved, negative selection test)

\[ \text{IA}: \rho = F (\psi) \]

(Pass the detectors profile to clone function)

\[ \rho : \omega \text{PCKs} \parallel \chi_i \leftrightarrow \chi_j \]  

(Clone process run in a live environment)

\[ \rho : \forall \delta; \delta_i \rightarrow \{\delta_0 , \delta_1 , \ldots , \delta_n\} \]

(Each generated detector matched with the entire self patterns, either deleted or saved, negative selection test)

\[ \text{IA}: \rho = F (\psi) \]

(Pass the detectors profile to clone function)

\[ \rho : \omega \text{PCKs} \parallel \chi_i \leftrightarrow \chi_j \]  

(Clone process run in a live environment)

\[ \rho : \forall \delta; \delta_i \rightarrow \{\delta_0 , \delta_1 , \ldots , \delta_n\} \]

(Each generated detector matched with the entire self patterns, either deleted or saved, negative selection test)
3.2 Blocking process (BLCK):

(3.2.1) $\delta_{i+1}:iblocked$ gradually.

(3.2.2) $\forall (i+1)$ will broadcast the new pattern(s) to all IAs inside the domain.

IA: $\forall (i+1) \rightarrow \mathcal{N}_i$.

4. RESULTS AND DISCUSSION

This work has been implemented in a wireless environment that contains one mobile node and two stationary nodes connected through an access point and wireless adapters. For the prototyping purpose, the protocol used was IEEE 802.11g and the operating system was Windows XP. The code is written in JAVA and MySQL-front and MYSQL Command Line Client were used for the output. In the offline (secured) phase, where the domain is isolated from other networks (including internet), self-profile and detectors-profile have been created. A sample of the captured patterns table is shown in Table 3.

Table 3. Sample of incoming patterns profile

<table>
<thead>
<tr>
<th>PATTERN_ID</th>
<th>PATTERN</th>
<th>TYPE</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>fa9cfe</td>
<td>GE</td>
<td>192.168.1.3</td>
</tr>
<tr>
<td>2</td>
<td>16f8cd0</td>
<td>SU</td>
<td>192.168.1.3</td>
</tr>
<tr>
<td>3</td>
<td>1bd4722</td>
<td>SU</td>
<td>192.168.1.3</td>
</tr>
<tr>
<td>4</td>
<td>1b104d2</td>
<td>GE</td>
<td>192.168.1.5</td>
</tr>
<tr>
<td>5</td>
<td>a9835</td>
<td>GE</td>
<td>192.168.1.3</td>
</tr>
<tr>
<td>6</td>
<td>80a59</td>
<td>SU</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>7</td>
<td>1c672d0</td>
<td>SU</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>8</td>
<td>1050169</td>
<td>SU</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>9</td>
<td>7a84e4</td>
<td>SD</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>10</td>
<td>c3b895</td>
<td>SF</td>
<td>192.168.1.5</td>
</tr>
</tbody>
</table>

The incoming patterns profile contains the following types of patterns:

- **Self patterns (SF):** Incoming patterns that do not match with detectors hence classified as self patterns.
- **Genes patterns (GE):** Incoming patterns that match with the frequently occurring patterns that are captured and stored as pure self.
- **Self patterns from cloned detectors (SD):** Patterns that are classified using the cloned detectors.
- **Nonself patterns (NS):** the patterns other than the genes (GE) and detectors are classified as nonself if it affect the NTFS files.
- **Suspect patterns (SU):** The incoming pattern is firstly classified as suspect (SU) and then all the NTFS files in the C drive are checked to verify that no alteration had been by the suspect patterns.

During the test period it has been found that the count for genes patterns (GE), self from detectors (SD) patterns, self patterns, suspect patterns, and nonself patterns were 32283, 6821, 954412, 111795, and 5518 respectively. The five data types are shown in
The patterns are firstly classified as a suspect and shown in the screen shot in Figure 9. After a change, alteration or a modification to one of the NTFS files in the system for three counts it is then reclassified as nonself.

Figure 8. In the same figure the generated detectors used in the classification besides the high scored detectors cloned are depicted.

![Pattern Reclassification](image)

**Figure 8. Incoming patterns and detectors profiles**

The patterns are firstly classified as a suspect and shown in the screen shot in Figure 9. After a change, alteration or a modification to one of the NTFS files in the system for three counts it is then reclassified as nonself.

![Pattern Reclassification](image)

**Figure 9. Pattern reclassification**

As depicted in the figure, regardless of source address, the same incoming pattern will be captured, e.g. pattern 17a29a1 had been classified during the first three counts as a suspect pattern where it has been assigned id 43437 sent from 160.0.53.202, the same pattern is captured again with different id 171509 as nonself sent from 160.0.53.20 when a change in one of the system files has been discovered. The pattern that is classified as a nonself will be broadcasted to the rest of the nodes so as to be treated properly as a nonself in future. Figure 10 depicts that the same pattern is classified by different nodes as a nonself.

![Different Node Classifications](image)

**Figure 10. A pattern gets same classification by different nodes**

The entire incoming patterns that have not been classified will be stored as suspect patterns (SU). If it has been encountered more than three times and at the same time it doesn’t affect NTFS system files, it is then considered as a self as shown in Figure 11. Some patterns initially classified as suspect, but their behavior indicate that it is not malicious and hence they are reclassified as self.

![Reclassified Patterns](image)

**Figure 11. A suspect pattern reclassified as a self**

The corrupt patterns that harm the NTFS files are classified as nonself and it will be blocked. The time at which damage has been checked is declared at pattern_damage_node_time column in the in Figure 12. As an example, pattern fcfc67 has been encountered as nonself with three different pattern id incoming from different three nodes. A search in the block_patterns table shows that the same pattern is then blocked.

![Blocked Patterns](image)

**Figure 12. Blocked (MNB) nonself patterns**

In Figure 12, pattern fcfc67 had been sent from different sources, however, since it was classified as nonself by one node, the classification was broadcasted to all domain and the same pattern is then blocked. Since the application phase was conducted in the domain after connecting to the internet, the only blocked nodes were the ones that exist inside the domain. Some malicious nodes send malicious patterns through the gateway which cannot be blocked; otherwise this would block the internet. The source node that keeps sending malicious will be blocked as depicted in Figure 13.

![Blocked Source Node](image)

**Figure 13. Malicious node blocked**
5. CONCLUSION AND FUTURE WORK
The success in mapping immune features and properties depicted in Table 1 above is expected to contribute to the ad hoc security field with a new auto anomaly detection mechanism, memory mechanism for better future reaction, self isolation for the malicious nodes resulting in a trusty communication environment, and nodes’ survivability. The immune-based security framework for MANETs has been presented. An intelligent agent that contains three profiles, self cells profile, non-self profile, and detectors’ profile is created. Replicas of the agent are distributed to all nodes inside the domain upon connection. A combination of negative selection, clonal selection, and danger theory mechanisms have been mapped for a self-organized system. The framework resolves some limitations which appeared in the previous works including our prior model. The major well thought-out issues are scalability and the bandwidth conservation, which mainly characterize the ad hoc networks. The work was implemented in static environment and the mobility so far has not been tested and will be conducted in a simulation experiments later. We plan, as a future work, to generalize this protocol by applying it to an application domain that has more nodes and much complicated scenarios as a verification.

REFERENCES