2009 Eighth IEEE International Symposium on Network Computing and Applications

Generic Danger Detection for Mission Continuity

Richard Ford, William Allen, Katherine Hoffman, Attila Ondi
Department of Computer Sciences
Florida Institute of Technology
Melbourne, FL 32901
Email: <rford, wallen, hoffmank, aondi>@fit.edu

Marco Carvalho
Institute for Human and Machine Cognition
Pensacola, FL 32509
Email: mcarvalho@ihmc.us

Abstract— **Mobile Ad-hoc Networks (MANETs) have become the environment of choice for providing edge connectivity to mobile forces. In particular, next-generation military systems leverage MANET technology to provide information assets to troops. However, MANETs face a number of serious security exposures, which are a superset of traditional networks. In prior work, we have described BITSI, the Biologically-Inspired Tactical Security Infrastructure, which attempts to address these challenges. BITSI uses a variety of techniques inspired by biological systems to provide effect-based security that is centered upon mission enablement. One of these techniques is the application of Danger Theory to mission continuity. In this paper, we explore different ways of implementing danger detection within BITSI, and show how generic approaches that are low-cost both computationally and in terms of implementation can provide acceptable results**

Keywords-Danger Theory; Security; Damage Detection; MANETs; Mobile Ad-hoc Networks

I. INTRODUCTION

Mobile Ad-hoc Networks (MANETs) provide an important solution to data communications in environments where fixed network infrastructures are either unavailable or inadequate for the needs of the current mission. However, the flexibility and independence that allows a MANET to support mission-critical communications also deprives that MANET of the protections inherent in traditional wired networks. Additionally, the dynamic nature of MANETs raises security issues due to the difficulty of authenticating new users and of establishing the validity of software updates.

Increasingly, researchers have attempted to address security issues in dynamic environments by adapting biologically-inspired approaches to the detection of harmful behaviors. For example, Matzinger has proposed that the human immune system operates by combining self/non-self recognition with "danger signals" generated when damage is occurring [9]. While this approach is controversial in biology, it has proven a useful model for the design of artificial immune systems to provide computer security [1].

In [4] we described our Danger Theory-inspired approach to MANET security. This framework, known as the Biologically-Inspired Tactical Security Infrastructure (BITSI), uses a combination of reputation tracking and reinforcement learning to detect and respond to damage in an ad-hoc network environment without the need for centralized monitoring or control. Once damage is detected, BITSI attempts to identify the node(s) that is/are responsible for the damage and modifies the network's configuration to isolate, or at least reduce exposure to, the most likely sources of the damage. However, rather than simply blocking any suspect network traffic or hosts, BITSI's goal is to support mission continuity by reducing the risk of communication with harmful nodes while still allowing necessary interactions (e.g., if a misbehaving node were the only link between isolated units that link would be used as a "route of last resort").

One problem related to this approach is the need to reliably detect damage to the network or its components as soon as possible so that corrective action can be taken to reduce damage to the mission. In this paper we propose a novel approach that provides a general rule set for damage detection while reducing both the complexity of analysis and the cost in CPU cycles/network traffic needed to determine that damage has occurred. We demonstrate that this approach produces results that are sufficient for protection of the network and compare them with more traditional causal analysis techniques.

II. PRIOR WORK

Research in biologically-inspired approaches to security, such as Artificial Immune Systems (AIS) [8], is increasingly applied to the detection of malicious behaviors in dynamic environments. Loosely organized systems can fall prey to attacks when there is no simple way to distinguish friend from foe and little history of previous bad behavior. Early AIS work focused on differentiating between self and non-self [3], with the assumption that non-self was more likely to be harmful. However, one problem with this approach is that there is no clear evidence that self can not do harm, nor that non-self is always malicious.

A more recent approach based on Danger Theory [9], requires some measure of actual damage to an organism to trigger a defensive response. Thus, non-self that causes no harm is not (yet) a danger to the system and no assumption is made that known components are not capable of causing damage. Researchers have begun to apply Danger Theory-inspired approaches to intrusion detection in computer networks and systems [5] and, more specifically, to Mobile Ad-hoc Networks [10].

Our approach to MANET security, BITSI, takes a Danger Theory inspired approach to damage detection by focusing on damage to the mission instead of pre-defined damage to individual components of the system [4]. Thus, the goal of

978-0-7695-3698-9/09 $25.00 © 2009 IEEE
DOI 10.1109/NCA.2009.50

BITSI is to support mission continuity in the face of damage to one or more nodes in the MANET. As such, BITSI balances the overall impact of localized damage with the higher-level goals of the mission, at times tolerating a certain degree of harm when the source of that harm is necessary to achieve the mission's ultimate goals.

However, determining the source of damage in a dynamic environment is difficult. New nodes may arrive at any time, or nodes may move to a part of the network where they have not yet encountered their new neighbors. These new nodes may quickly become a necessary part of communications within their new location. How can a node's new neighbors determine if that node can be trusted? Malicious nodes certainly cannot be relied upon to warn their neighbors that they are about to do harm; often the only reliable indicator of a node's intent is its past actions. Thus, where possible, BITSI tracks the reputation of individual nodes to assess their trustworthiness and relays that reputation to other nodes in the network.

If a node behaves as expected, its reputation will be good and other nodes will trust it. If not, its negative reputation will spread and other nodes will avoid contact or, at least, interact with that node only when no other option is available. It is possible for false alarms or incorrect assumptions regarding a node's behavior to decrease a node's reputation unfairly. One solution to this problem is to allow forgiveness, increasing a node's reputation slowly as long as the node is not blamed for further damage. Since reputation is a collaborative effort, a false alarm generated by any one node will not reduce reputations drastically, but a pattern of "bad" reports from many nodes, or repeated from a single node, will accumulate over time, driving the reputation of the "bad" node down further and causing neighboring nodes to increasingly suspect that node as the source of the detected damage.

Identifying the actual source of damage in a dynamic environment, such a MANET, may be difficult. Nodes often do not know the real source of a network connection and can seldom know the exact path that multi-hop traffic takes from message to message. One approach [7], addresses this issue by introducing a collaborative filtering approach which assigns blame for detected damage based on collective reputation assessments rather than upon the observations of one node.

In [11] we discuss the use of reputation to adjust routing weights so that a node that appears to be causing damage can be avoided by future traffic. This approach has two distinct benefits. First, we can leverage the already-extant HSLS beaconing [2] to share routing reputations in the form of link costs. Each node adjusts the advertised cost of its outgoing links based on its own reputation. Second, this approach never completely disallows a route: historically unreliable links are still available to the network, they are simply less desirable due to their higher cost. Through simple experiments, we demonstrated that this approach works well in dynamic environments where node movements will frequently provide alternate routes around damaging nodes. If no new damage occurs, the weights gradually return to normal, forgiving the apparent cause of the damage. This approach both reduces the impact of false alarms and allows nodes which may have only been peripherally involved with the source of the damage to regain their "good" reputations over time.

III. BITSI-ENABLING AN APPLICATION

While our prior work has demonstrated that Danger Theory is an interesting metaphor for detecting and responding to attacks within a MANET, it did not directly address some of the challenges faced when defining danger and assigning cause and effect. Thus, while we demonstrated that the *underlying* technique has merit, the actual detection of damage and attribution of blame was left largely unexplored.

In this paper, we tackle this issue head on: how can a Danger-Theory inspired system actually handle damage attribution? In particular, we consider the task of "BITSI-enabling" an application, allowing it to communicate with a BITSI agent (kernel) on each node. Two approaches are explored: our previous "idealized" approach where an application is sophisticated enough to detect underlying damage cause (with some degree of probability), and a generic approach that requires little sophistication on the part of application designers.

Our hypothesis is that generic detection based upon relatively simple rules is almost as effective as highly-granular application-specific approaches, as well as being far simpler to implement and maintain.

A. Trial Implementation

Consider a simple application consisting of a sensor network designed to capture pictures when a particular sensor is triggered. In this system, connectivity is provided for by radios on each device, and the system uses the HSLS routing protocol [2]. A typical scenario is shown in Figure 1.

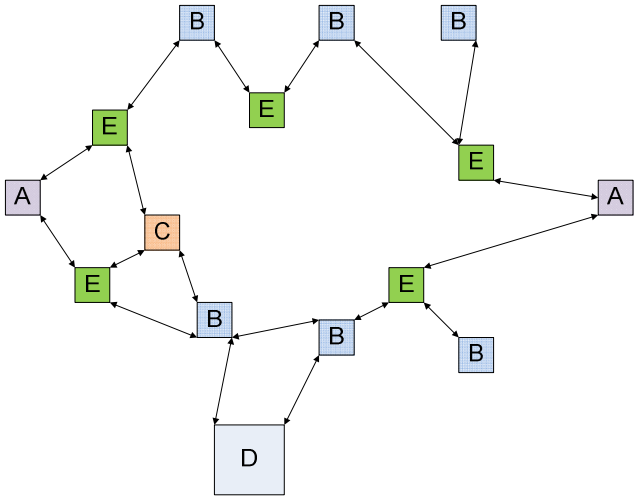

Figure 1. A typical sensor net application. Each node communicates via a wireless link; nodes route collaboratively using the HSLS protocol. Nodes of type A are trigger nodes; type B are cameras; type C are collation nodes; type D is the final delivery point. All nodes collaboratively route traffic, added by nodes (E) that exist to simply aid in connectivity.

In this figure, "normal" operation is carried out as follows. Sensors of type A are trigger nodes. These nodes detect that a vehicle is approaching, and request images from cameras of type B. These cameras record images at predetermined delays after receiving a trigger, and send their images to collation nodes of type C. At collation nodes, the images are collected, and an image set is returned to the command post (D) for analysis.

In this scenario, there are clearly two extremes in terms of detecting damage. First we attempted to implement our idealized approach. To do this, we created an expected state transition diagram for each node in the network. For example, consider the operation of node C. It may determine that it did not receive images from all sensors B, or that some of these images were unacceptably delayed. Examining the system, one can see that when C is missing images, one of the following failure modes is likely to have occurred:

- Sensor A did not request images from all the cameras as required
- Sensor A requested images, but the network dropped the traffic and so some cameras never received an image request
- Sensor B received an image request, but did not service it
- Sensor B received an image request, serviced it, but sent the image to the wrong collation point
- Sensor B received an image request, serviced it, and sent it to the correct collation point, but the network dropped the packet

Without further information, C is unable to determine why it did not receive the required images. However, C could conceivably issue queries to its own BITSI kernel to request state information. This approach would require each node to understand application semantics at the protocol level, as well as track state between multiple sessions. Furthermore, if such a request is made remotely but no response is received, C cannot determine the root cause of the problem – especially if the queried node is more than one hop away.

We have carefully considered implementing protocols that work hop-by-hop through the MANET. However, these approaches are slow and require considerable application-specific knowledge. Nevertheless, using next-hop queries of BITSI agents on each node, it is possible (if unwieldy) to determine with some level of certainty which nodes are causing mission failure.

Such an approach requires fairly sophisticated and voluminous logging of application traffic. In addition, changes to the underlying topology during execution can also cause nodes to reach false conclusions. However, due to the somewhat stochastic nature of the system, our experience is that these errors do not adversely affect the outcome over time.

Based on our efforts, it quickly became apparent that instrumenting a third-party application would be extremely time-consuming. Furthermore, as our instrumentation would be based on our intuition regarding potential vulnerabilities and failure modes, it would be of limited use in detecting exploits that work in unexpected ways.

Contrast this data-rich approach to a much simpler generic approach that knows only that *something* has failed. In response, the effected node modifies the reputation of *all* nodes that could have caused the problem. If each node keeps a buffer of recent packets, this approach can even be modified to deal with traffic spoofing, as the sending node for each packet received can be recorded. As damage reports are handled hop-to-hop, no special packet-tracing code needs to be added to the network stack. Furthermore, BITSI-enabled applications simply need to be able to detect deviation from previously-defined success metrics.

Clearly these two approaches have radically different properties. The more analytical approach may, for example, immediately infer the root cause of a problem and deal with it… assuming its reasoning is sound. In contrast, the generic approach has significant practical benefits in terms of implementation *if* the approach allows the network to "heal" around trouble spots. In the remainder of this paper, we consider the benefits, and implications of an implementation of generic danger detection.

Perhaps the most obvious benefit of the generic approach is that it treats applications as a black box in terms of functionality. That is, all that is needed for detecting danger is some method of determining if the application "succeeded" or "failed". With some applications this is relatively easy (for example, a web server succeeds if it returns a HTTP 200 return code, an application generates a SEGV and exits); with others, it can be hard. However, enablement does not require a complex cause and effect analysis, making BITSI compatibility quick and painless.

As some failure modes are likely to be difficult to determine automatically, a generic approach also lends itself to "human in the loop" decision making. For example, in the scenario discussed above, an operator may determine that a camera is misaligned, and therefore not capturing the desired information. A generic approach does not need to know *why* this condition occurred, simply that it did and so the camera is less desirable to that operator than it otherwise might have been. Given a choice, the system will therefore request images from an alternate source if available.

Finally, a very simple system that follows easily enumerable rules is attractive because of its computational simplicity. As stated in [12] MANETs are frequently made up of nodes that are CPU, memory and power constrained. As such, they are not typically suited for carrying out computationally-intensive tasks.

IV. Experimental Description

The following two experiments use BITSI to provide for mission continuity. Before describing the experiments in detail, we first consider the rules BITSI applies on each node.

Each BITSI node maintains a ring buffer of traffic it has handled. Each entry includes where the traffic was sent to and received from, if appropriate, along with a time stamp. This information is used to uniquely identify a packet in the buffer; the actual time is unimportant, and synchronization is unnecessary. The size of this buffer is configurable. The larger

the buffer size, the higher the chance that a BITSI node will be able to trace traffic back to its origin.

Each node also keeps track of two different sets of reputations. One set is related to routing (and is based upon how other nodes perceive traffic that has been passed through that node), and the other based on "service". This encompasses all services provided by a particular node, as well as all traffic originating from a particular node. While there is obviously some benefit to tracking reputation with respect to individual services/clients, our goal with the generic function was to make the system as simple as possible.

Each application can determine it is being damaged in one of two ways: either it is receiving unwanted traffic (junk) or it is failing to meet certain functional requirements. For example, if an application on a node receives traffic that is corrupted, it notes the packet is junk, and sends a reputation adjustment notice to the node that forwarded the packet to it. This node adjusts its routing reputation (in case it damaged the traffic during forwarding), and forwards the notice to its next hop upstream (as determined by the traffic stored in the ring buffer). When the reputation message gets to the node which originated the traffic, the node's service reputation is adjusted accordingly It is our assumption (as explained in [4]), that the BITSI component in all nodes resists reputation tampering.

Routing reputation is used to modify the broadcast route weights within HSLS. Thus, nodes that have a poor reputation with respect to routing will appear more costly. The impact of reputation on routing can be varied within the experimental environment.

One of the challenges with this setup is that there are a large number of parameters that have an impact on experiment design. Topology fundamentally dictates the functionality of the network. In particular, we note that in very sparse topologies where individual nodes have little or no choice in their service providers, there is little opportunity for self-organization to modify the system in a positive way. Similarly, in networks where nodes move quickly and the network topology is constantly changing (in comparison to HSLS update times), topology changes become the dominant factor in determining overall system performance.

In this paper, we will focus on one simple topology, and argue that the results presented generalize fairly well to a number of different scenarios. Our example topology is built to provide good connectivity between nodes, as well as a constant change in node position. Finally, each node generally has several choices of "next hop" when routing traffic.

Our test topology consists of one client and one server fixed at opposite edges of a rectangular area that is 300x100 units in size. Nodes move from waypoint to waypoint at a constant speed of 1 unit per second. In order to provide for end-to-end connectivity, the moving nodes are broken up into four groups: attackers, and 3 groups of defenders. The defenders are localized into one group of 5 confined to the area bounded by (0,0) and (100,100), one group of 10 confined between (100,0) and (200,100) and a final group of 5 limited to the box (200,0) and (300,100). Attackers, when introduced to the network, may roam anywhere in the area bounded by (0,0) and (300,100). The setup is shown graphically in Figure 2.

This topology was loaded into our discrete time step simulator, Hephaestus. This simulator has been extensively compared to more granular simulators such as NS2 (see [11] for a more complete example).

Using this topology, we attempt to make requests from one fixed node to the other. Responses are sent within the same time step. With no attackers present, 196 service requests are made; 178 of these requests are correctly responded to. Dropped requests are the result of routing errors, where a route that is no longer available is still marked as viable in the routing table of a node.

In the first experiment, we add attackers one by one to this topology. Attackers participate in traffic routing, but differ from defenders in that they deliberately corrupt server replies. Thus, as attackers are added to the topology, the number of server requests correctly responded to drops.

The experiment relies on several other parameters that have a profound impact on outcome. Potential variables under our control are:

- The size of the buffer of traffic stored on each BITSI node
- The amount of background traffic, and how it flows through the network
- How BITSI nodes are forgiven for being implicated in prior damage events
- How much reputation change is made for each damage event
- How reputation modifies the cost of routes as broadcast by HSLS

Each of these parameters changes the overall behavior of the system. For example, if the forgiveness value is set too low, many good nodes will end up with a poor reputation; conversely, if the forgiveness rate is too high, the system never learns. Correct values are determined by experiment; machine learning may be used in the future.

Buffer sizes and background traffic are examined in the second experiment, and their impact is discussed there. Finally, the level of damage also impacts the rate at which reputation is adjusted. If being involved in one damaged flow makes one node's opinion of another drop to zero, the system will not learn effectively. Similarly, if the change in reputation is small, the system will take a large amount of time to learn.

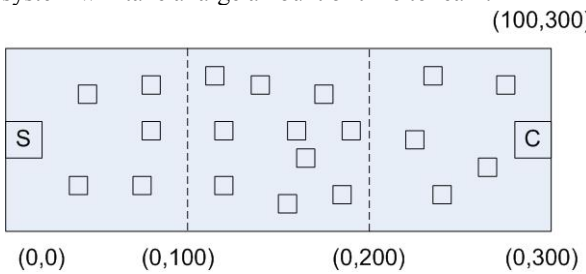

Figure 2. Network configuration for experiments.

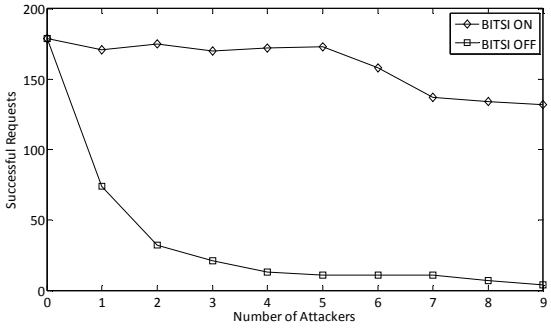

Figure 3. Successful server requests and responses as a function of the number of attackers. The upper line shows the performance of the system with BITSI enabled; the lower with BITSI disabled.

Fig. 3 shows the "successful" end-to-end server requests and responses as a function of the number of attackers. The upper line shows the effect of allowing BITSI to adjust routing weights based on reputation.

Experiment 2 introduces background traffic to the system. As noted above, the more background traffic in the system, the larger each traffic buffer must be in order to trace damaged traffic back to its source. This is shown in tabular form in Table 1, and graphically in Figure 4.

Table 1. System behavior for 6 attackers with a variety of buffer sizes

Buffer Size	Successful Requests	Failed Traces	Total Damage Reports
2	34	53	54
4	21	59	62
8	57	23	28
16	65	16	23
32	76	8	14
64	73	6	12
128	77	1	11

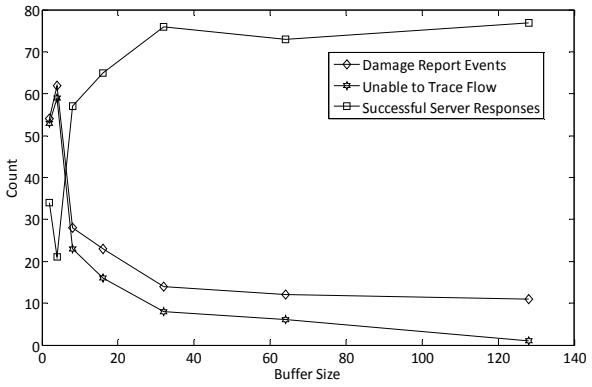

Figure 4. Graphical representation of incomplete damage report events as a function of buffer size.

V. Results & Discussion

Overall, the results shown are encouraging, and clearly demonstrate the utility of generic Danger detection. Note that in Fig. 3 BITSI is simply detecting the arrival of corrupted traffic. Furthermore, only one node is participating in the damage reporting process. This one-dimensional view of the network is probably a worst case performance for BITSI. The more nodes (and hence different views and routes) that participate in the system, the more quickly and accurately it learns.

In addition to this, the topology chosen is probably suboptimal from a detection perspective. Corrupted flows will almost always involve good nodes, as the two flow endpoints are distant in the network. Informally, we have found that when BITSI nodes have access to a greater variety of network views, their performance improves. Thus, while BITSI can protect networks that are entirely static, attackers are more reliably identified when nodes move freely.

This observation has greater impact when we consider Experiment 2. As the client/server node pair we are monitoring are at the edges of the overall experimental bounding box, these nodes seldom forward traffic for other nodes. Thus, they generally know the last of hop of any traffic they send. This explains the somewhat counterintuitive rise in performance as the buffer size decreases from 4 to 2. Although BITSI's performance *should be* worse, it improves. This is because the change to the next hop from the server fortuitously pushes further traffic to a route that has few failures.

This leads us to consider another important point. It is not the raw counts of failed traces that matters, as this is misleading. As BITSI reconfigures the routing weights in real time, the number of failures (and therefore proportionally the number of failed attempts to completely trace traffic) drops. Thus, we expect the percentage of failed traces to decrease as buffer size increase – this is, indeed, how the system functions (buffer size two has a failure rate of 98%; the corresponding rate for a buffer size of four is 95%).

Figure 4's data clearly illustrates that the system does not need to be able to perfectly trace traffic in order to be useful. While buffers are filled with benign traffic, appropriate values for forgiveness and damage still allow BITSI to build up a picture of the system over time. This resilience to failure is one of the things that make the BITSI approach so attractive. Even when presented with imperfect information (as is likely to be the case in a real MANET), the system preserves the mission.

One important area that we expect to impact implementation is the nature of the forgiveness function – that is, how reputations slowly get restored in the absence of new negative reports. In the experiment shown above forgiveness is implemented as a simple linear function. However, our intuition is that a more complex function that weighs the consensus of node opinions will be more powerful, especially with respect to routing reputation.

Experiment 2 is notable in that the *mission* is preserved in the presence of failure/attack. As client nodes select routes based upon past experiences with each resource, the system quickly adapts to failures. Again, no sophisticated learning

models are applied: as such, our intuition is that performance could be considerably improved by such approaches. However, this is not without risk, as added complexity can lead to unexpected behavior in loosely coupled systems.

It is important to stress that outcome is highly topology-dependent. For example, if a client node can only reach one camera via a path that contains an attacker, there is no way in which BITSI can reconfigure the system to provide service. Similarly, if all reachable cameras are unresponsive, BITSI cannot help as no viable options exist. This observation should be unsurprising as it is entirely in keeping with the biological metaphor: redundancy directly impacts BITSI's ability to work around damage. In other experiments, we have observed that higher node densities tend to function more efficiently. In our further work, we will present the data shown here for a variety of different scenarios. However, our experience is that the overall behavior shown in this paper is typical of the many different topologies we have explored; we have chosen to focus on just one for the sake of simplicity.

VI. Future Research & Conclusion

The experiments described in this paper represent a naïve view of cause and effect. At one extreme, each node has a complete understanding of potential causes; at the other, nodes infer nothing about cause and effect. Instead, each node modifies reputation based on other nodes' involvement in a particular transaction. While our results show that a simplistic approach is effective, it is unlikely to be optimal. Instead, we believe that a more reasoned approach is to attempt to "learn" cause and effect based upon different inputs.

In [6], an approach using "Hierarchical Temporal Memory" has been proposed. This approach is attractive, as it is capable of learning in an unsupervised and unstructured way based upon different inputs. Furthermore, it is particularly adept at learning patterns that are ordered temporally. Our intuition is that applying a machine-learning approach to cause/effect analysis will provide for a richer response environment.

Overall, this work demonstrates that "BITSI-enablement" of an application can be carried out in a relatively simple manner. Furthermore, by detecting local damage in a generic way, a defender is not required to encyclopedically codify failure modes; instead, a more holistic approach can be taken.

We are currently implementing an end-to-end demonstration of BITSI designed to execute on real systems. This system will support the features described here, and will be tested using a variety of different attack scenarios. Our hope is that by actually experimenting on BITSI in a "live" environment we can demonstrate the effectiveness of generic Danger Detection as well as discover real-world artifacts that our simulator is incapable of producing.

References

[1] Aickelin, U., Greensmith, J. and Twycross, J., "Immune system approaches to intrusion detection - a review," Proc. 3rd International Conference on Artificial Immune Systems (ICARIS 2004), pages 316-329, Catania, Italy, 2004.

[2] BBN, "Hazy Sighted Link State (HSLS) routing: A scalable link state algorithm," BBN Technical Memorandum No. 1301, August 2001, Revised March 2003.

[3] Forrest, S., Hofmeyr, S., Somayaji, A., and Longstaff, T., "A sense of self for unix processes," Proc.1996 IEEE Symp., Research in Security and Privacy, pp. 120-128, IEEE Computer Society Press, 1996.

[4] Ford, R., Carvalho, M. and Allen, W., "BITSI: A biologically-inspired adaptive defense framework", Proc. Adaptive and Resilient Computer Security Workshop (2007)

[5] Greensmith, J., Twycross, J. and Aickelin, U., "Dendritic cells for anomaly detection," IEEE Congress on Evolutionary Computation, 2006.

[6] Hawkins, J. and George, D., "Hierarchical temporal memory: concepts, theory, and terminology," Numenta Inc., 2006

[7] Hoffman K., Ondi A., Ford R., Carvalho M., Brown D., Allen W., Marin G., "Danger theory and collaborative filtering in MANETs," Journal in Computer Virology, ISSN 1772-9890, Springer Paris, 2008

[8] Kephart, J.O., Sorkin, G., Swimmer, M., and White, S.R., "Blueprint for a computer immune system," Proc. International Virus Bulletin Conference, Virus Bulletin PLC, San Francisco, CA, 1997.

[9] Matzinger, P., "Tolerance, danger and the extended family," Annual Review of Immunology, volume 12, pp. 991-1045, 1994.

[10] Mundinger, J. and Boudec, J-Y. L., "Analysis of a reputation system for Mobile Ad-Hoc Networks with liars," Perform. Eval., volume 65, numbers 3-4, 2008.

[11] Ondi, A., Allen, W., Ford, R., Marin, G., Carvalho, M., and Perez, C., "MANET simulation and security: new wineskins for new wine?", presented at ACM Southeast Conference, Auburn AL, 2008

[12] Sterne D., Balasubramanyam P., Carman D., Wilson B., Talpade, R., Ko C., Balapari R., Tseng C-Y., Bowen T., Levitt K., and Rowe J., "A general cooperative intrusion detection architecture for MANETs," Proc. 3rd IEEE International Workshop on Information Assurance (IWIA'05), pp. 57-70, IEEE Computer Society, Washington, D.C., 2005