Distributed Intrusion Detection System for Resource-Constrained Devices in Ad Hoc Networks

Adrian P. Lauf, Richard A. Peters, and William H. Robinson
Department of Electrical Engineering and Computer Science
Vanderbilt University, Nashville, TN 37235

Abstract

This paper describes the design and implementation of a hybrid two-stage intrusion detection system (IDS) for use with mobile ad-hoc networks. This system, called HybrIDS, classifies network interactions by mapping each behavior from its target operational scenario to a discrete label. The hybrid nature of our IDS is captured in the cooperative nature of two detection strategies. Our first detection strategy employs peak analysis and probability density functions to isolate deviance at the level of a single node. It can perform this analysis with zero prior knowledge of its operating environment; it requires no calibration data. In contrast, the secondary method relies on a cross-correlative component, which requires careful tuning of a detection threshold. Its primary advantage lies in its ability to detect multiple threats simultaneously. The first stage provides tuning and calibration information for the second stage. Our approach distributes the IDS among all connected network nodes, allowing each node to identify potential threats individually. The combined result can detect deviant nodes in a scalable manner, in the presence of a density of deviant nodes approaching 22%. Computational requirements are reduced to adapt optimally to embedded devices on an ad-hoc network.

1. Introduction

Distributed and mobile technologies pervade daily life in almost every way shape and form, and this creates an increasing need to protect not only systems, but their ever-critical data from malicious activity. [1] Because these technologies are further expanding in their abilities to intercommunicate, simple static methods are no longer adequate in providing security to these computational scenarios.

To this end, we can identify three quantifiable levels of protecting a system: 1.) Intrusion protection: preventing unauthorized access by means of encryption or another form of obfuscation. 2.) Intrusion detection: identifying intruders that have breached the mechanisms put in place to secure the networked system. 3.) Intrusion elimination: removing an intruder while providing minimal disruption to the legitimate network.

Much work exists in the field of intrusion protection; providing data integrity and confidentiality has long been provided by signature and encryption mechanisms, such as the industry-standard RSA encryption method, [2-4] which employs public-key cryptography. RSA provides the advantages of a block cipher encryption method and combines it with the principle that factoring large numbers is difficult even for advanced, sophisticated computing equipment. However, recent developments in distributed, password-defeating mechanisms as well as the advent of quantum computing, which in theory could instantly factor large numbers [5], are leaving RSA more vulnerable to attacks. Examples of other schemes succumbing to their inherent vulnerabilities are numerous; high-definition DVD content, such as that encrypted under the Advanced Access Content System (AACS) for digital rights management [6] was said to be completely impervious to attack, but was cracked within a few months of release [7].

This leads to the desire for the second security implementation: intrusion detection. This article will cover intrusion detection within the scope of an ad-hoc network scenario; it will not cover intrusion elimination. The classic model for an intrusion detection system, or IDS, involves the usage of a primary detection mechanism that is applied to a network or subnet. [8-14] In Figure 1, the brick wall, which can represent a firewall, houses the intrusion detection component. This component is filters away operational parameters related to its host system, and instead focuses on packet-level information, such as source, destination, frequency, routing mechanisms, and other protocol-specific pieces of information. As a direct result of this protocol-specific behavior, a traditional, centralized IDS requires large amounts of computational power [2, 15] to identify threats on a network. The need for complex traffic analysis and statistical pattern recognition cause the system to be impractical for scenarios not employing general-purpose computation.

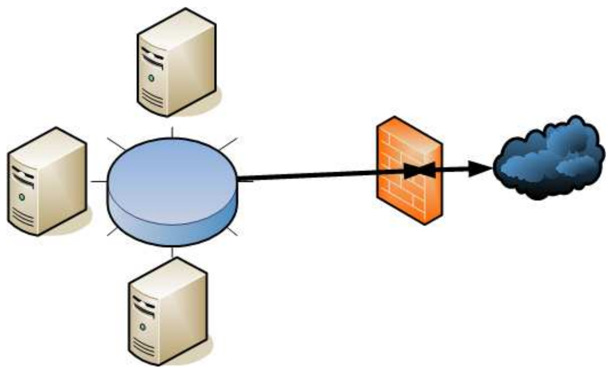


Figure 1 - An example of a Centralized IDS

We propose the development of a decentralized IDS approach that uses a combination of detection strategies to accomplish its goal effectively, while minimizing resource utilization. Figure 2 demonstrates that each node in the ad-hoc

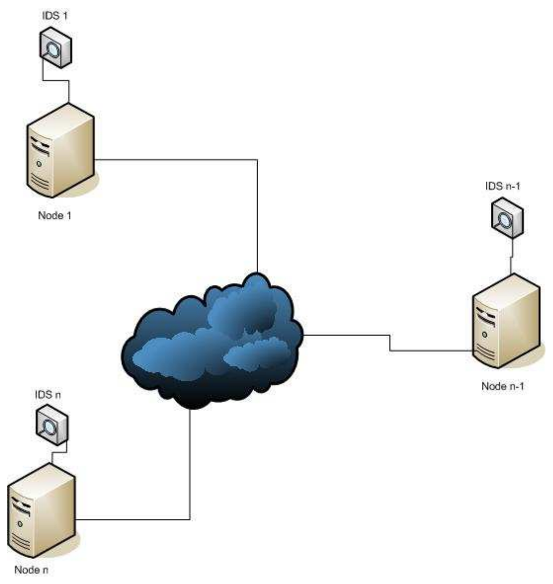


Figure 2 - An example of a distributed IDS

network contains a local copy of the IDS system. Furthermore, this IDS utilizes a set of abstraction principles that allows it to monitor behaviors related to different operational scenarios. Computational overhead is reduced by monitoring the system's behavior instead of individual network packets that may omit the context of an application's functionality. The IDS is called HybrIDS, and shall be referred to as such from this point forward.

To enable HybrIDS to adapt to multiple scenarios, an interface must be developed between the host system and IDS that still captures information about its target scenario. Therefore, we chose to abstract the system behavior to a level where only interactions among nodes are visible to the IDS This allows us to reduce requirements for computational intensity, which, depending on the target application, can be critical for power and hardware needs. In our scenario-driven tests, we were able to scale HybrIDS sub-linearly from network sizes ranging from 5 to 100 active nodes.

To reduce the load on the host system, the IDS can be integrated as separate hardware from the main processing platform. Our testbed platform was a 200MHz ARM9-based 32-bit RISC architecture equipped with 64MB of RAM and a small, embedded Linux platform featuring the Linux 2.6 kernel. However, for space-limited applications, and applications in which there also exists plenty of computing power, the IDS can be integrated directly into the host computing infrastructure. For reasons of portability, HybrIDS utilized the Java platform to write the IDS application itself.

HybrIDS applies itself well to a wide range of target applications because it scales well to a large ad-hoc network. This gives it desirable properties of being able to monitor an extremely flexible size of network configurations without significantly increasing the computational load.

As a result, almost any ad-hoc network that possesses a discrete set of possible interactive behaviors can be a candidate target for HybrIDS integration.

HybrIDS provides a flexible method of intrusion detection. Rather than relying on a fixed detection strategy, HybrIDS utilizes a system of two intrusion detection schemes. They operate in a cooperative effort to increase detection efficiency and accuracy beyond the capabilities of each individual method. The detection scheme initially provides detection for a single anomaly without the need for training data, as the first phase prepares the second phase of the IDS. This phase also calibrates the multiple-anomaly-detection scheme. Our scenario-driven examples show that HybrIDS is capable of detecting deviant agents comprising up to 22% of the population of the nodes in an ad-hoc network. We can accomplish this in a scalable manner that allows for efficient identification of deviant nodes.

In Section 2 we will discuss two operational scenarios in which HybrIDS provides useful intrusion detection. Under Section 3, we provide details on how HybrIDS captures a model of its target system efficiently. Section 4 details the individual operating principles that comprise the hybrid nature of the IDS, followed by an explanation of their joint operation. Section 5 discusses HybrIDS in the context of the two operational scenarios, while Section 6 places HybrIDS in the context of other efforts in intrusion detection. Section 7 summarizes the paper and discusses future work.

2. Scenarios

The selection of a set of target applications of HybrIDS is perhaps as important as the nature of the system itself. Here we describe the rationale that enables a range of applications to benefit from the features offered in HybrIDS. The given examples are neither a complete list nor a basis for an explorative space of possible applications, but show potential applications of how HybrIDS adapts itself to its target's needs.

2.1 Modified ADS-B

The first scenario to consider is within the aviation sector. ADS-B, or the Automated Dependent Surveillance Broadcast system, provides a much-needed update to aircraft avionics. It provides aircraft and ground-based systems with the capability: (1) to manage and identify surrounding aircraft, (2) to evaluate traffic conditions, and (3) to provide collision avoidance (Airborne Collision Avoidance System) [16-19]. Because it is designed to operate within the civilian airspace domain, the broadcasts are made available to all aircraft and ground control mechanisms, regardless of the receiver's identity. ADS-B is currently, at the time of this writing, in extensive testing to determine its broader impact and benefits on the state of aviation today. Therefore, there is no accessible data on security breaches, security compromises, and possible misuse of the unencrypted, broadcasted data. However, it becomes clear that security concerns, in some form or another, will arise as a result.

To use this as an example candidate for intrusion detection in an ad-hoc network, we have extended the capabilities of ADS-B to operate based on node-to-node interaction requests. It is important to understand that ADS-B is intended for public, civilian use. Therefore, it does not serve well as an example system without first modifying some of its operational parameters. It is assumed that for first-level security purposes, some form of a security protocol exists, such as encryption and/or a shared secret method. Table 1 shows the unmodified ADS-B messages, which include details such as altitude, position, and identification parameters that allow the system to form a model of the airspace.

Table 1 - Tradition ADS-B broadcast contents

Broadcast Category	Broadcast Content	Statistical Distribution
1	Position	Symmetric (2Hz)
2	Velocity	Symmetric (2Hz)
3	Altitude	Symmetric (2Hz)
4	Aircraft ID	Symmetric (2Hz)
5	Rate of Climb	Symmetric (2Hz)

In addition to changing broadcasts to requests, the list of available requests was extended to 10 by adding information that might enable joint operations between manned or unmanned aircraft.

Table 2 - Modified ADS-B behavioral enumeration

Broadcast Category	Broadcast Content	Statistical Distribution
1	GPS Position Information	0.42
2	Altitude	0.20
3	Rate of Climb	0.15
4	Velocity Vector	0.10
5	Mission Update	0.08
6	Request Redirection	0.01
7	Mission Start	0.01
8	Mission End	0.01
9	Emergency Action	0.01
10	Leader/Follower change	0.01

In this updated model (Table 2), the aircraft identification request has been removed. For simplicity, it is assumed that sender/receiver information is ascertained during the network’s specific communication protocol. The third column indicates the probability that such a request will occur, thereby mapping a frequency distribution. For example, the networked aircraft would likely not issue many “Mission End” commands during normal operation, while a “GPS position” request would logically occur more frequently.

The modified ADS-B example contains four critical elements that make it an ideal candidate for the application of HybrIDS: 1.) A set of actions or behaviors that can be quantified before the network is in operation (Discrete Behaviors.) 2.) A statistical distribution of those actions (known or unknown). 3.) Independent functioning of the interconnected nodes. 4.) The capacity for each node to identify requests made of it by other nodes. These parameters map to a set of requirements for a low-power, embedded platform: 1.) High-level system abstraction. 2.) Minimized computational power requirement. 3.) Low resource utilization and overhead. The two lists indicate the level of *adaptability* and *conformity*; the former demonstrates how applicable and useful HybrIDS is within the system context, while the latter demonstrates the efficiency of its implementation. For instance, HybrIDS may adapt well to a target scenario, but certain elements such as size or speed requirements might limit its conformity. Further sections will clarify why HybrIDS conforms well to the ADS-B scenario.

2.2. Massively-distributed microrobotics

Another interesting target area involves the deployment of large numbers of autonomous robots that, in large numbers, can quantify and analyze a situation. These robots often enter areas where it would be unsafe or impractical for humans to exist. As an example, a “BallBot” or “Planetary Microbot” [20] was proposed as a solution for exploring crevasses on Mars. Many of them could be deployed – thousands at a time – and while many would be lost to the conditions of the environment, the remaining collective could garner valuable information about the surroundings. In this example, the robots have extremely small computing environments, and have minimal communication skills, qualifying them as deeply embedded systems [21]. While the need for security is not apparent for cooperative scientific research, other adaptations, for instance espionage, could well use capabilities for encryption and intrusion detection. A single external influence could cause the entire collective to fail if properly crafted.

2.3. Summary of scenarios

The ADS-B and distributed microrobotics examples highlight needs present in an ad-hoc network of homogeneous systems: security beyond simple encryption. Taking into account the requirements of adaptability and conformity, we will now explore how HybrIDS becomes a viable platform for intrusion detection in an ad-hoc network infrastructure.

3. System-level abstraction

To meet the requirements for adaptability, we must consider how HybrIDS can abstract a “world model” with which to understand its target application. The key to this lies in abstracting the possible space of node-to-node interactions, quantizing them ahead of time. For instance, in the modified ADS-B scheme, the “Broadcast Category” column in Table 2 represents a discrete listing of all the possible interactions that can occur. Not only are they listed discreetly, but they can also be seen as “mapped” to a label, albeit a simple integer numbering scheme.

HybrIDS utilizes an integer labeling system to represent all interactions between nodes. The behaviors and interactions are specified before run time. While it is not necessary for the mappings to be identical across the nodes, it is useful to have them arranged as such for purposes of simulation and analysis.

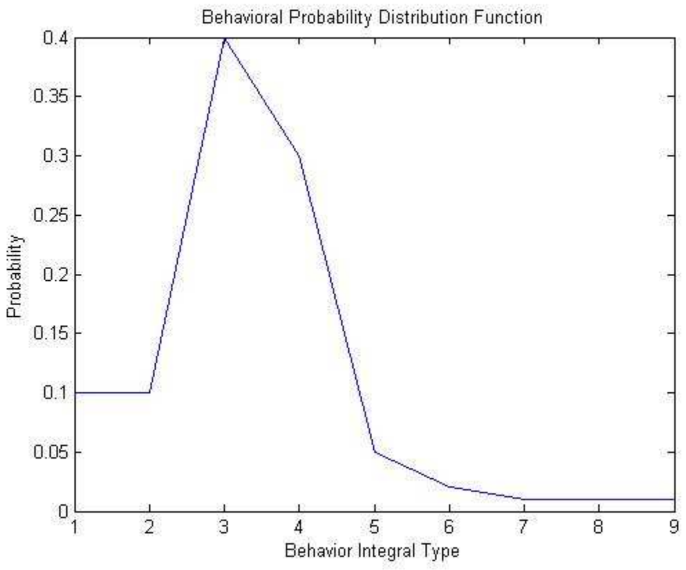
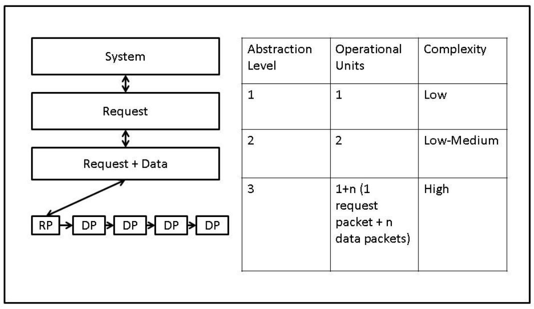


Figure 3 - Example Behavior Mapping

Each behavior mapping will inevitably experience some sort of probability distribution in terms of its occurrence. Figure 3 details a set of integral behavior types as well as their probability of occurrence. While different, the behaviors listed in Table 2 also represent a set of integer labels coordinated with a probability of occurrence. Together, these labels can generate a Probability Density Function (PDF) that forms a model of what kind of interactions a node will experience over time.

As a result, integer labeling of requests provides a simplified solution to understanding the dynamics of an ad-hoc network. One important note is that data itself is *not* used when characterizing input requests. This decision was



Abstraction Level	Operational Units	Complexity
1	1	Low
2	2	Low-Medium
3	1+n (1 request packet + n data packets)	High

Figure 4 - Levels of System Abstraction

made to reduce the complexity and processing requirements for the IDS. Rather, the level of abstraction is simply limited to the request types alone, as denoted by “Level 1” in Figure 4.

4. Reasons for a hybrid methodology

Because no one strategy is foolproof or completely efficient, we describe two techniques that, to some degree, compensate for each other’s weaknesses. The first strategy requires no training data, and can perform basic detection almost immediately upon starting. The second strategy can provide identification of multiple anomalies within the network. HybrIDS combines these strategies to provide an improved level of intrusion detection.

4.1 The Maxima Detection System (MDS)

The first detection method used by HybrIDS is called the Maxima Detection System (MDS). Its primary purpose is to rapidly identify potential threats. Under the hybrid detection scheme, its secondary purpose is to provide calibration information so that the secondary IDS phase, called CCIDS (Cross-Correlative Intrusion Detection System, introduced in the next section) can function more accurately. Its detection scheme analyzes peaks present in the PDF from statistics generated from requests made by other nodes.

It is important to understand how the IDS gathers its information before we can assess the performance of MDS when performing network behavior analysis. As mentioned before, interaction requests are classified with integer labels so that the system may be modeled in a high-level perspective. When a request is received by a node, the request and its source are recorded in a structure called a history table. This table contains column entries that map to the interaction labels and rows that map to other nodes monitored by the IDS. The IDS does not keep information about its host node. The recorded information is stored as counter values that indicate the number of requests received according to their classification, and from what node it originates. The history table is the most important information repository from which statistical models can be generated.

The statistical model itself is an aggregation of entries from the history table. Each table entry represents a counter value which can be averaged to form a PDF representing the request statistics of individual nodes. MDS forms an average “behavior profile” of its target scenario and associated network which is essentially a PDF consisting of the average of all the individual node PDFs. Figure 3 shows an example average PDF for a system. It is noted that the activities are ordered in such a way that a normalized distribution exists; this is a requirement for MDS to function properly. This ordering can take place in real time, or can be generated as a result of trial simulations. A Chi-Squared distribution is preferable for purposes of implementation.

Let γ be the number of nodes in the system and let β represent the number of behaviors present in the system. Let represent a matrix of dimensions $\gamma \times \beta$ containing the historically and temporally-updated probabilities of a certain behavior ξ. The mean PDF, φ is computed for each node in (1).

(1)

Following the averaging process, φ is then analyzed for peaks. Since the labels are ordered

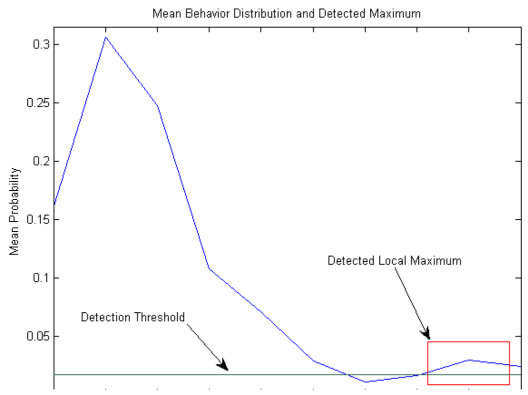


Figure 5 - Example peak detection

and will experience some form of a distribution, we can exclude the global maximum peak as normal activity. Following this exclusion, the PDF is analyzed for the presence of local maxima. Since a local maximum is a less-likely event for a normalized distribution, we assume that this indicates the presence of deviant behavior. Of course, during the duration of the scenario's operation, system behavior will change and the exactness of the distribution in the mean PDF will vary. For this reason, peak detection is limited by a sensitivity threshold. The indication of a local maxima and its threshold is shown in Figure 5.

This first stage in the detection process identifies only an interaction classification that is exhibiting an abnormal trend; it has not yet identified the source of this aberration. To complete identification, the history table is traversed to find the node that statistically has the greatest contribution to the establishment of the local maximum in the mean PDF. This reverse-mapping proves effective and fast; since the history table is comprised of counter values, its values can be easily normalized to find the source of the deviant behavior in terms of average contributions.

Identification of deviant nodes can occur quickly because trends in the average PDF stabilize rapidly. Of course, the primary disadvantage to MDS is that it can at most identify only one deviant agent on the network. Significant statistical deviation in local maxima therefore provides the advantage of speed and reliability, which will be necessary for the eventual combination with CCIDS.

4.2 The Cross-Correlative Intrusion Detection System (CCIDS)

MDS relies on finding local peaks in a local IDS's average PDF. In order to expand the IDS's functionality, we would like to add features that increase the IDS's conformity to its target application. One of the most pressing requirements is the ability to detect multiple deviant nodes

– a capability not possible with MDS. To do this, we must consider a different method of analysis that can return multiple results. One promising technique utilizes cross-correlation to generate scores indicating the degree of correlation between node behaviors. Of course, any method has its disadvantages; cross-correlation is dependent on a threshold that determines whether a score is or is not statistically correlated to another score. However, when the threshold is appropriately set, cross-correlation can yield accurate results about multiple nodes that increase the IDS's conformity.

Our cross-correlative approach, called the Cross-Correlative Intrusion Detection System (CCIDS) once again draws on data from the history table. The scores are the result of cross-correlating an average PDF for the system with individual PDFs corresponding to each node. This yields a set of scores, one score per node. Let $\boldsymbol{\eta}$ represent a row-summed vector derived from the history table. The vector is then averaged. Let represent a transposed vector containing the *individual* averaged distribution of a behavior for a particular node number .
The scores from the resulting cross-correlation are obtained by . (2).

Once the scores are obtained, CCIDS must determine whether each score is normal or deviant. This is done by thresholding. Upon finding all the individual scores, an average score is generated. The threshold specifies by how much an individual score

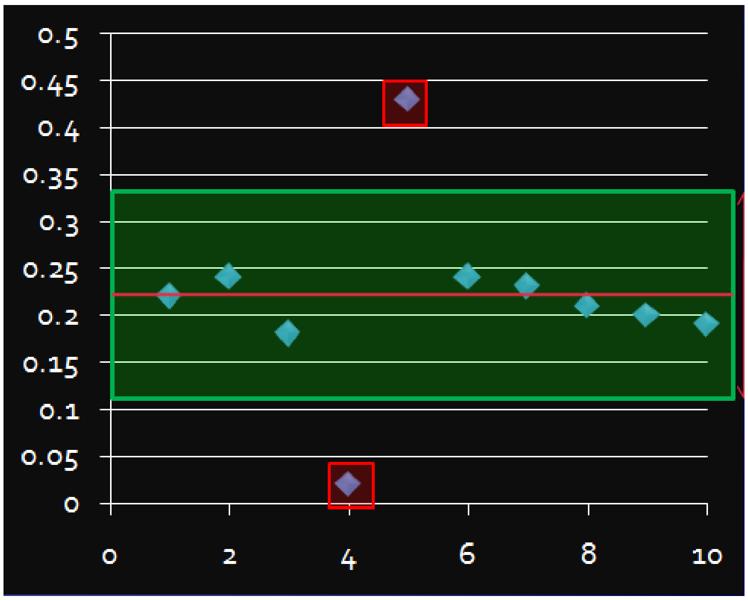


Figure 6 - Thresholding of scores

may vary from the average score before it is considered deviant. Figure 6 demonstrates a bounding box which represents the threshold. The red line denotes the mean score. Note that the threshold is a distance from the mean score line; it is an absolute value. Scores within the green bounding box are considered acceptable values, while those outside the box, highlighted in red, are considered deviant.

CCIDS's primary disadvantage is that the threshold is critically important in detection; improper setting will either include many false positives, or cause it to not detect any deviant nodes. Once properly set, CCIDS can run accurately with significant resistance to change in the system model; node behavior can change over time without significantly impacting the results of detected deviant nodes. However, setting the threshold properly must be done prior to runtime, or after sufficient calibration data has been found to find the optimum setting.

4.3 HybrIDS

The two detection methods introduced to this point, MDS and CCIDS have their respective strengths and weaknesses. MDS can detect one and only one deviant agent in a short period of time without needing any significant calibration. CCIDS can detect multiple deviant agents, but requires a period of calibration in order to set its threshold correctly. An improperly-set threshold would yield false positives, or no detection at all. We seek to combat the weaknesses by combining the strengths of each detection strategy. This combined approach is called HybrIDS.

One of the most important aspects of IDS conformity to a target is its ability to scale to larger networks. Adding more nodes to the ad-hoc network should minimally impact the efficiency of the IDS. To accomplish this, we have adjusted the execution rate of the most computationally-intensive portions of the IDS so that the execution time scales sub-linearly. IDS runs are based on two types of processing cycles: (1) data processing cycles, which are compute-intensive, and (2) data collection cycles, which do not execute analytical functions of the IDS. The method of controlling the number of data processing cycles relies on understanding that a larger collective of nodes will establish an average profile over a smaller amount of time than a small one. Therefore, less analysis is required to properly identify deviant nodes – whereas a network consisting of fewer nodes will require more computational analysis over the same amount of time.

Understanding scalability then allows us to issue data processing cycles (DPCs) variably between the data collection cycles (DCCs) such that more collection cycles will pass before analysis for a larger network size. As a result, as network size increases, IDS performance will scale effectively. In our studies, computational intensity varied sub-linearly dependent on the size of the network. We tested a range of network sizes, from 5 to 100 interconnected nodes, for both MDS and CCIDS.

The combination of the two detection strategies is key to the operation of HybrIDS. Each stage must operate in a manner that maximizes the efficiency of the IDS as a whole. This means limiting the run time of MDS to the point where it stabilizes and provides necessary calibration information for CCIDS. Before we can analyze transitioning, it is important to understand how MDS "calibrates" the CCIDS.

As stated previously, MDS identifies at most one suspected deviant node. In contrast, CCIDS is capable of detecting several deviant nodes simultaneously. If the threshold is improperly set, many of those detected nodes may be false positives. To reduce this probability and to set the threshold properly, the IDS enters a transition period following a stabilization period for MDS. The threshold for CCIDS initially is set to represent a 100% deviation from the average score – a wide margin of deviance that most likely would not catch any positives, false or otherwise. During the transition, both MDS and CCIDS run simultaneously and both return sets of suspected nodes. These sets are compared to see if the deviant agent set from CCIDS matches the one entry from MDS. Initially, except in rare scenarios, nothing will be returned, and the threshold for CCIDS is lowered. Successive comparisons and threshold adjustment proceed until the deviant node set comparison returns a match. This indicates a corroboration of the scenario by both MDS and CCIDS. At this point, the transition period

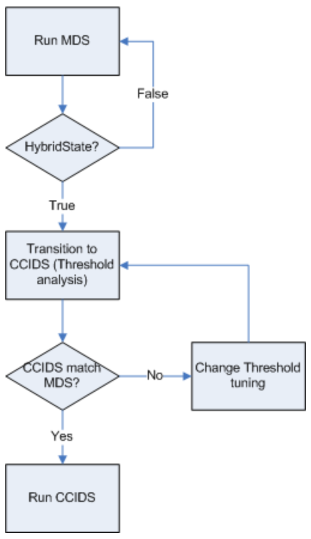


Figure 7 - Transition Algorithm

ends, and CCIDS, now properly calibrated, continues as the primary means of detection. Figure 7 shows this algorithm as a flowchart. In the figure, the “HybridState” decision item must be true in order to begin the transition process. Performance metrics can establish the conformity of HybrIDS on an ad-hoc target application. Based on experimental results, the most significant impact on IDS performance was provided by the percent pervasion of deviant nodes on the ad-hoc network. Pervasion is defined as the total percentage of nodes known to exhibit deviant behavior within the collective. Since scale was shown to have little impact on HybrIDS’s efficiency, performance was measured in number of processing cycles required to stabilize an accurate result in terms of percent pervasion. Tests were based on a fixed number of 10 behaviors (appropriate to the scenario), with generated data conforming to the modified ADS-B model shown in Section 2.1. The results are shown on a surface plot in Figure 8, which represents varying number of nodes versus a varying percentage of pervasion. The vertical axis shows the number of processing cycles required before identification was successful. This number includes MDS, transition, and CCIDS processing cycles.

We can see that an increase of pervasion affects the stabilization time of the IDS. This makes sense because the increase in occurrence of deviant behavior becomes more common, which in turn has a greater impact on the score separation found by CCIDS. Since individual scores are compared to an average, the average will begin to manifest elements of the deviant behavior in a more determiental way. As a result, more transition cycles are required to adjust the threshold more stringently, and more CCIDS processing is required, since more data is required to find the correct number of deviant agents.

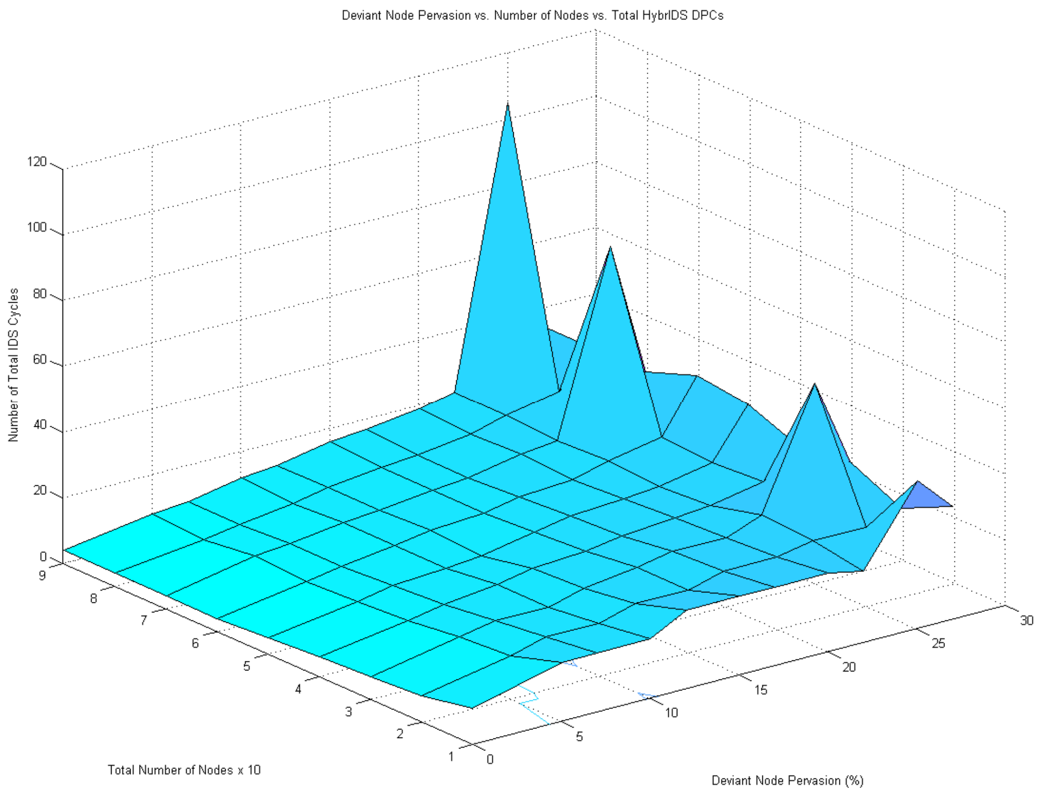


Figure 8 - HybrIDS performance

According to data collected for the modified ADS-B scenario, HybrIDS is capable of converging on a list of suspect nodes, provided that the pervasion of deviant nodes is less than or equal to 22%. HybrIDS will converge on results with pervasions exceeding this limit, but predictability becomes difficult, impacting the conformity of the IDS to an embedded application. This is an important constraint, since most real-time systems have little to no tolerance for non-determinacy in their operations. With these limitations in mind, the adaptability of various scenarios to use HybrIDS becomes more apparent.

5. Analyzing Scenarios: Modified ADS-B

Let us refer now to the two scenarios described previously to explore adaptability and conformity. We will cover the four requirements for adaptability and the three requirements for conformity. The modified ADS-B scheme will be analyzed first, followed by the BallBot scheme.

The modified ADS-B scheme meets the Discrete Behavior requirement easily. According to our scheme, there is a fixed set of 10 behaviors that defines the operational characteristics of the system. These behaviors are listed in Table 2. The second requirement, i.e., a statistical distribution of actions, is also met. Because ADS-B functions are, in general, the same for all aircraft, there will be an inherent distribution of the occurrence of behaviors.

Independent operation of nodes is also guaranteed. While the nodes may coordinate information, no one aircraft may take control of another. The last requirement for adaptability, of knowing a received request, is provided in that each transmission contains a sender along with the receipt of request information. We make an important assumption that identity spoofing is rendered either impossible or impractical by the communications protocol. All nodes are "presumed innocent until proven guilty" by the first phase (i.e., MDS) of the IDS.

Turning our attention now to conformity, we now demonstrate that HybrIDS can efficiently meet the criteria for conforming to a target scenario. The requirement for a high level of system abstraction is implicitly demonstrated by Table 2 in that all interactions between the aircraft have been abstracted to a request type. HybrIDS utilizes these integer mappings at face value. There is no notion of data or associated parameters that might complicate the understanding of the system itself. Computational power requirement is mitigated inherently by the distribution and control of data processing cycles. HybrIDS uses situational awareness to reduce the need for large and complex computations. Therefore, we can utilize less powerful hardware that can reduce the power, cost and thermal requirements of its host system.

Finally, resource utilization is met through a constrained use of program resources. For portability reasons, HybrIDS is written in Java, and can run on any system implementing or emulating the Java 1.5 Runtime Environment. Java includes garbage collection routines and intelligent memory management, which is useful for conservation. Despite this, garbage collection can affect determinacy, which can severely impact responsiveness and performance in a real-time system. To combat this problem, all relevant memory structures persist throughout the entire period of HybrIDS's execution. In a typical configuration, utilizing 35 nodes with 10 behavior types for the ADS-B scenario, maximum memory usage by the largest data structure does not exceed 2.7 kilobytes.

In our test configuration, we utilized a 200MHz ARM9-based development board with 64MB of RAM and an embedded Linux operating system kernel. A specially-designed, cross-compiled Java runtime environment, called JamVM v. 1.5.0 was loaded in order to execute the HybrIDS application, which was packaged as a JAR file. JamVM itself is does not fully comply with the Java 1.5 standard due to licensing issues, but is completely compatible in terms of the execution of HybrIDS. It also features a compact memory footprint and is optimized for embedded systems applications.

To assess the overall memory footprint, the JVM was run with a simple Java application containing nothing more than a thread sleep command. Any memory overhead from the application would therefore be negligible. The JVM itself required 4.3MB of memory to run, while HybrIDS, due to various program and control structures, required less than 750 KB of RAM. Together, JVM and IDS required approximately 5 MB of application memory, which can be seen in Figure 9. This conforms easily to the requirements of all but the most deeply embedded system architectures. Approximately 2.4 MB of RAM was utilized by the operating system kernel, issuing a total requirement of 7.4 MB for the system as a whole.

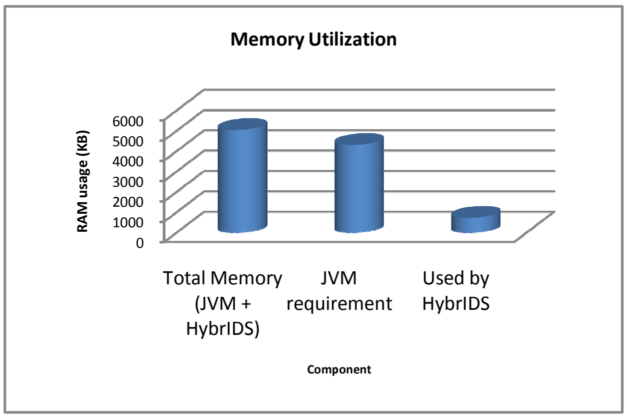


Figure 9 - Memory utilization

Viewing the different aspects of adaptability and conformity, it becomes apparent that HybrIDS is a good match for the modified ADS-B scenario. It is capable of quickly and thoroughly identifying multiple deviant agents in the scenario's ad-hoc network, and does so utilizing minimal system resources. Therefore, we recommend HybrIDS as a possible solution to intrusion detection needs for ADS-B and any possible derivatives.

5.2 Analyzing scenarios: Microrobotics

In the modified ADS-B scenario, we do not see much host-based restriction of HybrIDS in terms of memory and processing requirements. CPU processing capability, power, and memory are not greatly curtailed by a large system such as an aircraft. In contrast, the BallBot features a much smaller processing environment. While the exact constraints are unknown, it is assumed that a processing system available to a mobile robot only a few centimeters in size will be restricted to sensors and communications. Remaining space will be used by power supply and mobility, which severely limits the possibilities of running complex operations. We can assume that operational memory space will shift from megabytes to a few kilobytes in order to conserve power. Processing, if any, would exist via an 8 or 16-bit microcontroller.

Given these restrictions, Java would not make a good implementation choice. Memory and processing overhead from the JVM would be simply too great to allow for execution to take place. While a C or assembly-based implementation has neither been written nor tested, it is possible to re-factor the code into a minimalist language to bypass overhead incurred by high-level implementations. We believe this that the BallBot scenario is future fertile ground for our IDS technology. Given the scalability results for the modified ADS-B scenario, which shows sub-linear growth as nodes are added, it is easy in principle to extend the ad-hoc network size to include thousands of nodes. Figure 10 demonstrates a projected analysis of the number of processing cycles required to identify deviant nodes. We have utilized individual logarithmic trends for each pervasion percentage. While not exact, the logarithmic-like nature demonstrates that adding more nodes reduces the number of DPCs as discussed earlier. Furthermore, the flexibility of the IDS monitoring method assures that each node monitors only other nodes with which it actively interacts. This further reduces the overhead. For example, a 1000-node network does not necessarily imply that each node is monitoring 999 other nodes; it may be monitoring as few as 10 or as many as 500, depending on the relationships it establishes with the network's constituent nodes.

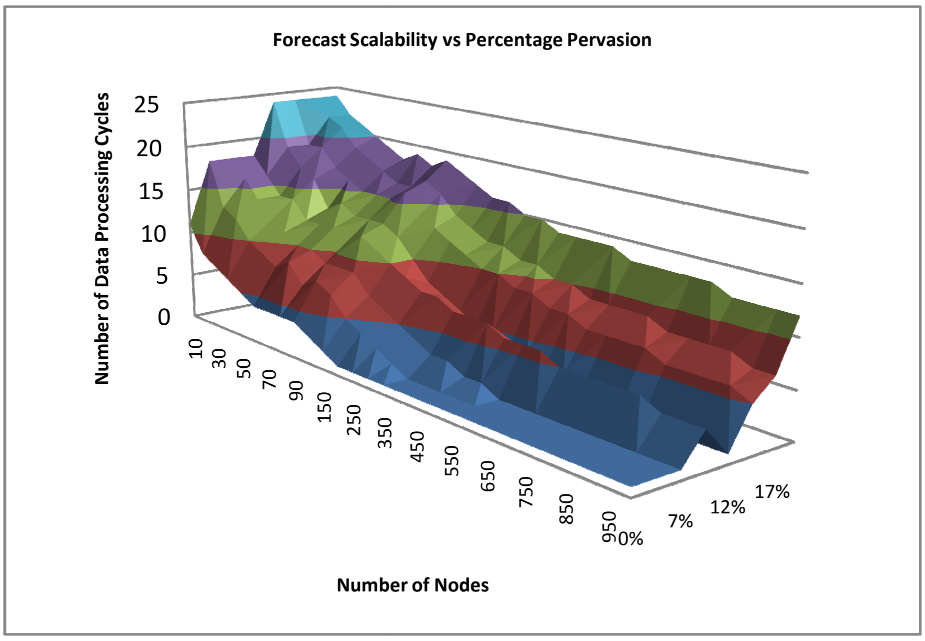


Figure 10 - Projected Scalability Surface Plot

As processing hardware and power supplies advance in their capabilities, HybrIDS could eventually be integrated into this ultra-small-scale computing environment with relative ease. The four requirements of adaptability would be met by intelligent distinctions between sensor-based data and shared requests or commands. For the BallBot case, conformity is the greatest challenge. Fitting the computing environment efficiently into such a minimalist system presents challenges that at the time of this writing may not have a solution. Despite this, we are confident that the scalability and performance aspects of HybrIDS, as demonstrated in the modified ADS-B model, shows promise of providing enhanced security that a hybrid, embedded IDS can provide.

6. Related Work

There exists a significant body of research on intrusion detection. Much of this work focuses on the development of IDSs that exist on static or non-ad-hoc networks. Some of our research relates closely to the work by Xiaoqiang et al., who specify methods for using cross-correlation in anomaly detection [12]. Irdis and Shamugam demonstrate how techniques of artificial intelligence (AI) may be applied to the field of intrusion detection. Our strategy does not use AI in order to reduce dependence on any one algorithm or system [8]. Dwen-Ren et al. specify the use of a hybrid intrusion detection system [22]. Their approach does not explicitly cover the ad-hoc network implementation, and the hybrid techniques are based on the use of centralized analysis alternating between a series of fuzzy logic classification mechanisms. Other references explore the application of reputation systems, such as that presented by Buchegger et al. [23]. A reputation system describes a method of reliably routing data. Useful for the context of connected routers, a reputation system seeks to identify transmission nodes that are either aiding or impeding proper transmission of information. The identification is performed by "reputation" information which is gathered from the observations of other nodes over time.

The authors of works [15, 23-39] specify the usage of intrusion detection within the field of ad-hoc networks. This field is, as expected, more compact and specific, as general networking techniques and computing power is no longer as applicable as in the traditional static-network-based IDS scheme. Marchang and Datta [33] introduce a collaborative IDS scheme in which message-passing between nodes serve to build collective information among nodes that are either directly connected, or within a one-hop-route of each other. Their approach is more central to communications and routing, while our work focuses on behavior-based IDS techniques, which are not used to determine routing for data between nodes.

The "Layered Intrusion Detection Framework" proposed by Komninos and Douligeris [31] delineates a specialized mobile ad-hoc IDS technique in which nodes assume different roles (e.g., alert, detection, and collection). This differs from our approach in which data collection and processing is done on each host node; there is no coordinated action between IDS instances.

This allows for greater application independence and allows the host systems to operate independently of information from other systems. It also reduces the possibility of failure and reduces computational needs.

Patwardhan et al. implement a threshold-based IDS [36] that works with routing on an ad-hoc network infrastructure through the use of “watchdog” nodes. This is a specialized case of an IDS that utilizes nodes serving a specific purpose. Our system implements a general-purpose detection scheme that is not dependent on special-function nodes. HybrIDS is also not a solution for the purposes of data routing and/or finding optimal, non-compromised routes between nodes.

Komninos, Vergados and Douligeris describe a method, which in principle, is similar in some respects to methods present to HybrIDS. They describe a two-phase process for detecting intrusion on an ad-hoc network, one of which requires zero-knowledge [32]. This is similar to the MDS phase in HybrIDS, which requires no training data. However, their framework is more specialized and deals with the management and distribution of encryption keys, which is a lower level of abstraction than HybrIDS proposes.

Given the literature relevant to mobile, ad-hoc-network-based IDSs, we believe HybrIDS sets itself apart both in purpose and in its detection approach to provide a unique perspective to the intrusion detection landscape. Because of its inherent scalability and low resource consumption, and portable codebase, our IDS is applicable to a different set of possible target scenarios than that featured in the related works presented in this section.

7. Conclusion and Future Work

In this article we have demonstrated the need and applicability of an embeddable IDS within the context of an ad-hoc network. We have shown how HybrIDS integrates well with scenarios that require scalability and computationally efficient implementations. The IDS compensates for weaknesses in detection methodologies by providing a unique two-stage anomaly identification strategy.

The first strategy, MDS, performs peak analysis to determine what possible deviance exists among networked nodes, with zero knowledge of the host system. The primary advantage of this method is that it quickly establishes a “lock” on the most likely potential deviant node. MDS balances data collection and data processing, thereby lowering the required power profile.

A secondary IDS strategy, called CCIDS, provides multiple-anomaly detection. Its primary strength comes from the use of cross-correlative operators. Because of the nature of the cross-correlation strategy, we must carefully choose a detection threshold with which the IDS can successfully detect deviant nodes without introducing false positives. Inherently, this requires prior operational knowledge of the operational scenario.

HybrIDS addresses the single-detection problem of MDS along with the threshold issue present with CCIDS by using the detection strategies in tandem. MDS is used as a calibration instrument to tune the threshold used in CCIDS in an accurate and automated fashion. Together, the two methods can provide a capable model of the activities in a distributed mobile ad-hoc network and perform analysis with minimal impact on computational resources.

We have shown two different scenarios, with small to large network sizes, and analyzed how HybrIDS can take advantage of both situations. In the modified ADS-B scenario, where more computational power might be available, HybrIDS shows rapid convergence on a set of possibly deviant nodes. It can identify deviant nodes up to a density of 22% of the interconnected nodes. In contrast, the microrobotics scenario presents an altogether different challenge of integrating an IDS into an extremely limited computing scenario. While current technology may limit its implementation, a scalability forecast shows that HybrIDS will indeed perform well once resources match computational requirements. It is our belief that HybrIDS addresses the intrusion detection needs of a large and scalable array of mobile ad-hoc networks. The two detection strategies implemented provide flexibility and accuracy to situations that otherwise may not have benefitted from the use of an IDS.

Future work includes the correlation of performance requirements for various network sizes. The data processing cycle is a relative indicator of the amount of work required. We will supplement this information with of the distribution of integer and floating point instructions required. In addition to computation, we will also analyze the power requirements using the embedded ARM9 processing platform. Further work is also being done to model trial runs on large distributed networking test beds, using the Vanderbilt University Institute for Software Integrated Systems (ISIS) Generic Modeling Environment (GME) [40]. Our goal is to dynamically render network diagrams from the viewpoint of multiple connected nodes running HybrIDS. This will provide a better perspective on how each node perceives its world model, as well as how different instantiations of HybrIDS identify threats within a distributed network.

8. Acknowledgments

This work was supported by TRUST (The Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422) and the following organizations: AFOSR (#FA9550-06-1-0244) Cisco, British

Telecom, ESCHER, HP, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia and United Technologies.

9. References

1. Kocher, P., et al. *Security as a new dimension in embedded system design*. in *Design Automation Conference, 2004. Proceedings. 41st*. 2004.
2. He, G. and S.R. Tate. *Efficient Authenticated Key-Exchange for Devices with a Trusted Manager*. in *Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on*. 2006.
3. Rivest, R.L., A. Shamir, and L.M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*. Rev. ed. 1977, Cambridge: Massachusetts Institute of Technology, Laboratory for Computer Science. 15, [1] p.
4. Yang, Q., et al. *An embedded RSA processor for encryption and decryption*. in *ASIC, 2001. Proceedings. 4th International Conference on*. 2001.
5. Day, C., *Quantum Computing Is Exciting and Important--Really!* Computing in Science & Engineering, 2007. **9**(2): p. 104-104.
6. Perry, T.S., *DVD copy protection: take 2.* Spectrum, IEEE, 2005. **42**(1): p. 38-39.
7. Farrell, N. (2006) *AACS decrypted*. The Inquirer **Volume**,
8. Idris, N.B. and B. Shanmugam. *Artificial Intelligence Techniques Applied to Intrusion Detection*. in *INDICON, 2005 Annual IEEE*. 2005.
9. Keum-Chang, L. and L. Mikhailov. *Intelligent intrusion detection system*. in *Intelligent Systems, 2004. Proceedings. 2004 2nd International IEEE Conference*. 2004.
10. Naess, E., et al. *Configurable middleware-level intrusion detection for embedded systems*. in *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*. 2005.
11. Siraj, A., S.M. Bridges, and R.B. Vaughn. *Fuzzy cognitive maps for decision support in an intelligent intrusion detection system*. in *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*. 2001.
12. Xiaoqiang, Z., Z. Zhongliang, and F. Pingzhi. *Intrusion detection based on cross-correlation of system call sequences*. in *Tools with Artificial Intelligence, 2005. ICTAI 05. 17th IEEE International Conference on*. 2005.
13. Yamada, A., et al. *Intrusion detection system to detect variant attacks using learning algorithms with automatic generation of training data*. in *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*. 2005.
14. Zhenwei, Y., J.J.P. Tsai, and T. Weigert, *An Automatically Tuning Intrusion Detection System*. Systems, Man and Cybernetics, Part B, IEEE Transactions on, 2007. **37**(2): p. 373-384.
15. Choudhary, D.R., et al. *Computationally and Resource Efficient Group Key Agreement for Ad Hoc Sensor Networks*. in *Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference on*. 2007.
16. Daskalakis, C. and P. Martone. *Alternative surveillance technology for the Gulf of Mexico*. in *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*. 2004.
17. Harman, W.H. *ADS-B airborne measurements in Frankfurt*. in *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*. 2002.
18. Hicok, D.S. and D. Lee. *Application of ADS-B for airport surface surveillance*. in *Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE*. 1998.

19. Nichols, R., et al. *Testing of traffic information service broadcast (TIS-B) and ADS-B at Memphis International Airport*. in *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*. 2002.
20. *Swarming for Success*, in *Astrobiology Magazine*. 2005, NASA Ames.
21. Walther, K. and J. Nolte. *A Flexible Scheduling Framework for Deeply Embedded Systems*. in *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*. 2007.
22. Dwen-Ren, T., T. Wen-Pin, and C. Chi-Fang. *A hybrid intelligent intrusion detection system to recognize novel attacks*. in *Security Technology, 2003. Proceedings. IEEE 37th Annual 2003 International Carnahan Conference on*. 2003.
23. Buchegger, S. and J.Y. Le Boudee, *Self-policing mobile ad hoc networks by reputation systems.* Communications Magazine, IEEE, 2005. **43**(7): p. 101-107.
24. Brutch, P. and C. Ko. *Challenges in intrusion detection for wireless ad-hoc networks*. in *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on*. 2003.
25. Cabrera, J.B.D., C. Gutierrez, and R.K. Mehra. *Infrastructures and algorithms for distributed anomaly-based intrusion detection in mobile ad-hoc networks*. in *Military Communications Conference, 2005. MILCOM 2005. IEEE*. 2005.
26. Erdem, O.M. *Efficient self-organized key management for mobile ad hoc networks*. in *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*. 2004.
27. Guyot, V. *Using WEP in ad-hoc networks*. in *Wireless and Optical Communications Networks, 2006 IFIP International Conference on*. 2006.
28. Hongmei, D., et al. *Agent-based cooperative anomaly detection for wireless ad hoc networks*. in *Parallel and Distributed Systems, 2006. ICPADS 2006. 12th International Conference on*. 2006.
29. Kachirski, O. and R. Guha. *Effective intrusion detection using multiple sensors in wireless ad hoc networks*. in *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*. 2003.
30. Kachirski, O. and R. Guha. *Intrusion detection using mobile agents in wireless ad hoc networks*. in *Knowledge Media Networking, 2002. Proceedings. IEEE Workshop on*. 2002.
31. Komninos, N. and C. Douligeris, *LIDF: Layered Intrusion Detection Framework for Ad-Hoc Networks.* Ad Hoc Networks. **In Press, Accepted Manuscript**.
32. Komninos, N., D. Vergados, and C. Douligeris, *Detecting unauthorized and compromised nodes in mobile ad hoc networks.* Ad Hoc Networks, 2007. **5**(3): p. 289-298.
33. Marchang, N. and R. Datta, *Collaborative techniques for intrusion detection in mobile ad-hoc networks.* Ad Hoc Networks. **In Press, Corrected Proof**.
34. Mishra, A., K. Nadkarni, and A. Patcha, *Intrusion detection in wireless ad hoc networks.* Wireless Communications, IEEE [see also IEEE Personal Communications], 2004. **11**(1): p. 48-60.
35. Mundinger, J. and J.Y. Le Boudec. *Analysis of a reputation system for mobile ad-hoc networks with liars*. in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, 2005. WIOPT 2005. Third International Symposium on*. 2005.
36. Patwardhan, A., et al., *Threshold-based intrusion detection in ad hoc networks and secure AODV.* Ad Hoc Networks. **In Press, Corrected Proof**.

37. Razak, S.A., et al., *Friend-assisted intrusion detection and response mechanisms for mobile ad hoc networks.* Ad Hoc Networks. **In Press, Corrected Proof**.
38. Watkins, D. and C. Scott. *Methodology for evaluating the effectiveness of intrusion detection in tactical mobile ad-hoc networks*. in *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*. 2004.
39. Xia, W. *Intrusion Detection Techniques in Wireless Ad Hoc Networks*. in *Computer Software and Applications Conference, 2006. COMPSAC '06. 30th Annual International*. 2006.
40. *Generic Modeling Environment*. 2000-2007, Vanderbilt University: Nashville.