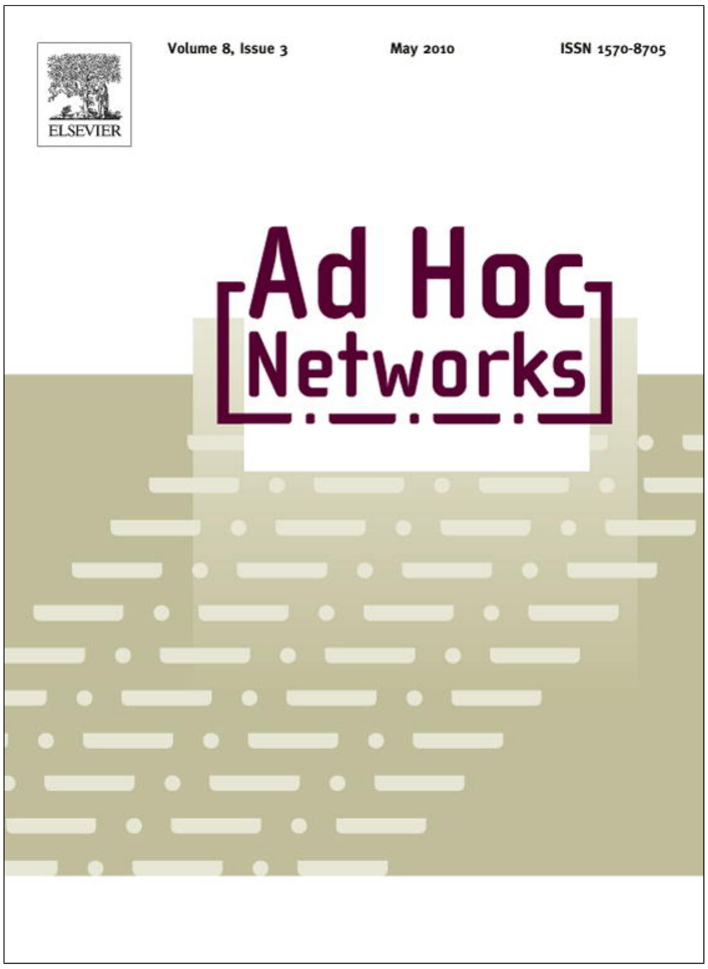

Volume 8, Issue 3
May 2010
ISSN 1570-8705
ELSEVIER
Ad Hoc Networks

Ad Hoc Networks 8 (2010) 253–266

Contents lists available at ScienceDirect

Ad Hoc Networks

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

A distributed intrusion detection system for resource-constrained devices in ad-hoc networks

Adrian P. Lauf *, Richard A. Peters, William H. Robinson

Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37235, United States

ARTICLE INFO

Article history:
Received 4 March 2008
Received in revised form 9 July 2009
Accepted 11 August 2009
Available online 18 August 2009

Keywords:
Embeddable
Machine learning
Intrusion
Detection
Scalability
Ad-hoc network
Highlevel Abstraction
Resource-constrained
Embedded

ABSTRACT

This paper describes the design and implementation of a two-stage intrusion detection system (IDS) for use with mobile ad-hoc networks. Our anomaly-based intrusion detection is provided by analyzing the context from the application-level interactions of networked nodes; each interaction corresponds to a specific function or behavior within the operational scenario of the network. A static set of behaviors is determined offline, and these behaviors are tracked dynamically during the operation of the network. During the first stage of the IDS, our detection strategy employs the analysis of global and local maxima in the probability density functions of the behaviors to isolate deviance at the granularity of a single node. This stage is used to capture the typical behavior of the network. The first stage also provides tuning and calibration for the second stage. During the second stage, a cross-correlative component is used to detect multiple threats simultaneously. 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 and can operate 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

Security, by definition, safeguards the assets of an individual or an organization against danger (e.g., theft, destruction, or unwarranted modification). Information security, from an embedded systems standpoint, becomes necessary in any case where information must be protected from agents (i.e., intruders) who are not authorized to view or to manipulate the content. Information security relies on preventive methods to protect content, using static techniques such as obfuscation of source, encryption of data for confidentiality [1], and verification of source integrity by means such as digital signatures [2]. This is particularly challenging in a highly networked environment, where communication and collaboration among devices serve as the backbone for system productivity [3]. Information security provides a measure of protection to the various communication protocols used among interconnected devices. However, some security techniques do not translate well to embedded systems, where constraints such as low-power, low-memory, and real-time operation may impact the computational capability of the system [4].

As a first line of defense, static security methods are steadfast and proven ways to protect information. Perhaps the biggest obstacle to providing complete information security using static methods is that a point of trust is always required; a trusted relationship must exist, whether it is in the transmission of a public key or a shared secret. Because of this trusted relationship, there always exists the possibility that the static method can be defeated despite the integrity and strength of the algorithm being used, however unlikely the scenario may be. Once broken, the static method is no longer able to ensure the

* Corresponding author. Tel.: +1 937 361 4021.
E-mail address: adrian.p.lauf@vanderbilt.edu (A.P. Lauf).

1570-8705/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2009.08.002

trustworthiness of the system. For example, RSA [2] provides 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 are leaving RSA more vulnerable to attacks, such as distributed, password-defeating mechanisms as well as the advent of quantum computing, which in theory could instantly factor large numbers [5]. As a second example, high-definition DVD content, 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].

To lessen the damage from such breaches, dynamic measures to security must be considered. In particular, intrusion detection provides the means by which anomalies in system behavior can be analyzed and graded on the threat they pose [8]. However, traditional intrusion detection systems typically employ packet-level analysis to determine whether a network has been compromised. The high computational overhead of packet-level analysis would strain the resources available in a network of embedded devices, where the constraints of power, memory, and timing hinder such a technique. Rather than using low-level information from all data packets individually, an intrusion detection system (IDS) can be implemented with high-level behavioral abstractions to allow each connected device (i.e., agent) to form a model of its system environment.

We describe the development of a decentralized intrusion detection approach that uses a combination of detection strategies while minimizing resource utilization. The work is an extension from [9]. Parallelism increases system scalability by removing the burden of analysis from one machine monitoring the entire collective (i.e., centralized) to multiple devices monitoring only their relevant interactions (i.e., decentralized). Within each node, two methods of intrusion detection are implemented, and ultimately combined to create a resultant IDS that delivers the capabilities of intrusion detection with low resource utilization and a desirable level of system accuracy. The Maxima Detection System (MDS) allows the characterization of either one or zero suspicious nodes. This is used to calibrate the sensitivity of the Cross-Correlative Detection System (CCDS), which is capable of detecting multiple intrusions. These methods are combined into a unified approach, which will be referred to as a Hybrid IDS (HybrIDS). HybrIDS can monitor a wide range of target applications because it scales well to a large ad-hoc network. Our testbed platform was a 200 MHz ARM9-based architecture equipped with 64 MB of RAM and a small, embedded Linux platform featuring the Linux 2.6 kernel. The IDS can be integrated directly into the host computing infrastructure, facilitating its implementation on resource-constrained devices. 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.

The remainder of this paper is organized as follows. In Section 2 we discuss our network model assumptions and the proposed attack 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, and we describe the individual operating principles that comprise the multi-stage nature of the IDS, followed by an explanation of their joint operation. Section 4 discusses two operational scenarios to which HybrIDS may be applied, while Section 5 discusses the details of the implementation. Section 6 places HybrIDS in the context of other efforts in intrusion detection. Section 7 summarizes the paper and discusses future work.

2. Network and attack models

In this section we discuss how our proposed system can be integrated into the context of network architectures, and how threats to these networks might be established. In [10], the authors detail a number of possible attack scenarios that can occur on wireless ad-hoc networks, to which our system is targeted. These include passive attacks, such as listening/packet sniffing attacks, and active attacks, which seek to disrupt network traffic and system objectives, such as jamming, spoofing, man-in-the-middle attacks, and data modification attacks.

We make the assumption that our system will be deployed on a resource-constrained ad-hoc network, either mobile or static, that exhibits any form of "normal" operating behavior. On such networks, the application space will dictate behavioral patterns that can be analyzed and abstracted into a system model. This model is then referenced for anomaly detection. Rather than utilizing signature detection methods, HybrIDS focuses on linked statistical methods: two separate algorithms are implemented to address the challenge of intrusion detection by incorporating the context available within the application layer.

2.1. System-level abstraction to detect attacks

Our proposed methodology models the behaviors present in the application layer. This means that rather than concentrating on packet analysis, our system instead identifies patterns that occur naturally in the various real-time software applications on ad-hoc network nodes themselves. 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 and quantizing them ahead of time (i.e., offline). Not only are they listed discretely, but they can also be seen as "mapped" to a label, albeit a simple integer numbering scheme. In [10], the authors make the case for application-layer support of intrusion detection. Focusing on software applications allows an IDS to identify patterns based on interaction semantics that otherwise would be missed by packet analysis.

As an example, let us consider the difference between an application-layer IDS and a packet-monitoring IDS, both for an industrial process control system that monitors temperature and pressure, as well as actuate valves. Distributed nodes are responsible for maintaining correct operating conditions for the process. Supposing that an attacker were to compromise a node, how would the different IDS strategies detect or fail to detect a well-crafted attack? Let us assume that one node, responsible for measuring a temperature, gives regular feedback to another node, responsible for controlling a valve. Under normal operation, the valve actuation would oscillate in an expected sinusoidal manner, responding to the feedback control system parameters in the control loop. The packet monitoring system observes control inputs occurring at a regular rate. The application-layer IDS observes the requests to either open or close the valve.

The point at which the packet-inspection-based IDS fails is when the attacker crafts a spoofing attack designed to induce a system error; instead of allowing for normal feedback control, the attacker causes the temperature monitoring node to malfunction and constantly make incremental adjustments to close the valve. To the packet inspection system, packets of similar size and frequency are occurring at normal intervals, so an alarm is not raised. However, the application-layer IDS recognizes that an abnormal distribution of requests to close the valve are occurring based on its understanding of the network model, and can then trigger an alarm.

HybrIDS utilizes an integer labeling system to represent a static list of interactions among nodes. The behaviors and interactions are specified offline 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 simulation and analysis. Each behavior mapping will experience some sort of probability distribution in terms of its occurrence. Fig. 1 details a set of integral behavior types as well as their probability of occurrence, generated by plotting the frequencies of observed behaviors on a test system. The behaviors listed in Table 2 (Section 4) represent a different set of integer labels with its own probability of occurrence. 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 understand the dynamics of an ad-hoc network. One important note is that application data itself is *not* used when characterizing input requests. Instead, the level of abstraction is simply limited to the request types alone, as denoted by "Abstraction Level 1" in Fig. 2. This decision was made to reduce the complexity and processing requirements for the IDS. Lastly, HybrIDS does not interface directly with the network stack; instead, it interfaces with applications at the real-time kernel level to build its system model. While this requires greater integration with the applications running on the resource-constrained devices, it greatly simplifies analysis when compared to packet payload decomposition [11], and provides a secure system framework for future development. It also reduces computational overhead by allowing information that updates the IDS to be received in the form of software interrupts, application handles, or localized network connections.

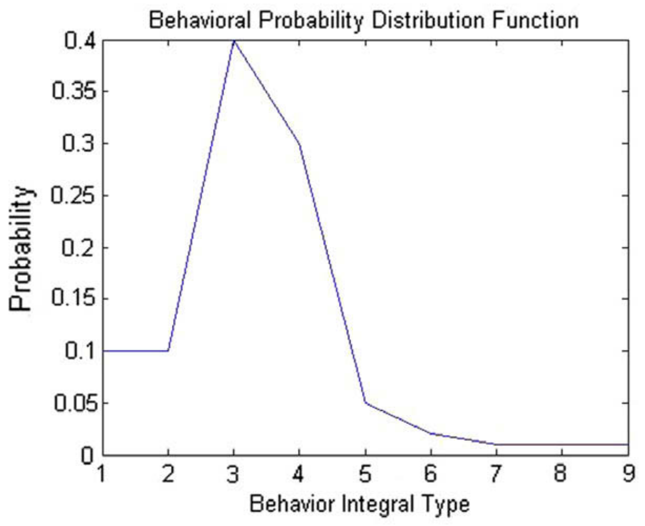
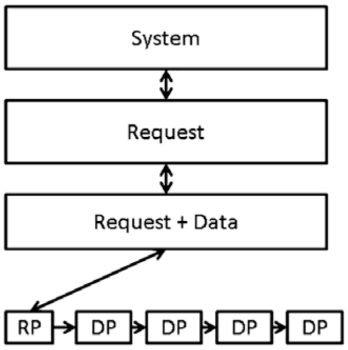


Fig. 1. Example behavior mapping statistical distribution.

2.2. The application layer

As mentioned in Section 2.1, analyzing the application-level behaviors of ad-hoc-networked devices provides a key advantage over conventional packet-inspecting intrusion detection systems, because it allows the identification of patterns by leveraging a richer amount of semantic information provided by the application itself. Some attacks can even be identified earlier using the semantic information of the application. In [11–13], the authors make the case for application-level intrusion detection systems, with an emphasis on analyzing the "application layer" by the connections made to various network-aware applications. Our approach is more fine-grained and analyzes application-specific interaction calls, which allows us to understand the application space in a more fundamental way. Rather than relying on connection information, our implementation is geared towards understanding how an application communicates with other networked nodes and what types of requests are being made. While hardware design and implementation may be geared towards a particular class of objectives and requirements, it is ultimately the application executing on the hardware platform that unifies the hardware resources to meet the intended requirements of a system. To this end, our proposed intrusion detection strategy is designed with a producer/consumer form of integration to maximize compatibility across hardware platforms. In addition, it allows for detailed monitoring of the semantic information available from the application domain.

The producer/consumer relationship is defined by an intrusion detection monitoring component, which resides on a Java Virtual Machine (JVM) platform that can be executed on any host hardware equipped with a JVM compiled for the processor's instruction set architecture. This portability allows for the modular development of the IDS algorithms described in Section 3, such that the detection strategies are homogeneously implemented regardless of hardware concerns and considerations. However, because Java must exist in the virtual machine environment, gaining access to the maximum amount of information located at the kernel level of the operating system is impossible. The producer aspect of the relationship is filled by



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

Fig. 2. Levels of system abstraction.

specialized monitoring applications natively compiled in C for the target's host operating system environment. This allows for the monitoring of application-layer resources, interactions, and instantiations. A secure message-passing system exists to transfer data from the monitoring producer applications to the intrusion detection system.

Establishing this method allows us to view application-level interactions without modifying the application itself. Alternately, the model may be simplified by the virtue that our proposed solution can be used as a framework for providing intrusion detection services: applications may be designed to include reporting functions in their object class methods or function blocks. This flexibility is important to allow our detection strategy to be used in both existing systems (using the producer/consumer method) and in new systems (which can natively provide information to the IDS by design.) This framework, in addition to the application-level identification scope and the multi-stage methodologies proposed in Section 3, all represent contributions to the field.

2.3. The adversary model

We now present the adversary model space for which our proposed solution may be applicable. Because the application-layer focus permits our proposed system to be independent of specific implementation hardware and software, the adversary model is therefore specified by the target application itself. In general, our proposed system is able to identify *statistically anomalous* behaviors. These behaviors are identified as actions that are statistically variant from mean behavioral patterns that are observed over time. This is in contrast to signature-based systems, in which the intrusion detection mechanism identifies a threat based on observed or heuristically-determined patterns [8,14,15]. By working in the statistical anomaly space, we can more flexibly identify attacks than with using a signature-based method. This flexibility comes with a caveat – threats that are easily identified by a signature-based detection mechanism may take more time to be found by a statistical method, depending on the algorithm and how quickly statistical observations change.

The adversary model can be regarded as a single-point or multi-point attack on either a stationary or mobile ad-hoc network. Such an attack may be perpetrated by an individual with a general-purpose computing system, or by compromising selected nodes on the network itself. Jamming and spoofing attacks represent fundamental methods that can be used to disrupt ad-hoc networks (described earlier in Section 2). The jamming attack represents network challenges that require no authentication into the network, and thus can be perpetrated with equipment to disrupt communications. On the other end of the spectrum, a spoofing attack represents the need for trusted access to one or more network nodes; this attack can be more difficult to detect. The statistical behavior of a spoofed node may be less apparent than a node that does not need a trusted relationship with the network. We believe that spoofing methods will likely provide some of the greater challenges to ad-hoc network security in the near future, and thus focuses our efforts on efficiently identifying them through the use of linked statistical methods.

3. A multi-stage methodology for an application-layer IDS

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 can perform basic detection almost immediately upon starting (i.e., requires minimal warm-up time). The second strategy can provide identification of multiple anomalies within the network. HybrIDS combines these strategies to provide an improved level of intrusion detection. By providing a multi-stage approach, different attack types can be identified more easily as aspects of an attack (such as a denial of service attack) can be viewed as it progresses over time.

3.1. 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 a potential threat. Under the multi-stage detection scheme, its secondary purpose is to calibrate the secondary IDS phase, called CCDS (Cross-Correlative Detection System, introduced in the next section) so that it can function more accurately. MDS analyzes the peaks present in the PDF from statistics generated from requests made by other nodes.

MDS provides a way to visualize and identify statistical oddities in the observed interactions of the application layer. It isolates abnormal behaviors that manifest themselves as outlying peaks in a density function that do not correspond to an expected distribution. These outlying peaks tend to appear quickly for interactions that are statistically unlikely; this allows MDS to rapidly identify the presence of such suspicious interactions. This property makes MDS an excellent candidate to quickly assess the state of the network so that other, more detailed IDS strategies can be implemented and tuned to yield similar results.

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 can be modeled from a high-level perspective. When a request is received by a node, the request and its source are recorded in a history table. This table contains: (1) columns that map to the interaction labels and (2) 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 by classification, and from what node they originate. The history table is the repository from what node each request originates.

Also noteworthy is the construct needed for MDS to operate normally. Before integration into the application, the integer labels are organized to follow a predicted distribution that most logically approximates an average system behavior in terms of statistical frequency. The distribution should optimally reflect a chi-squared layout to allow for the detection of peaks not indicative of normal system behavior that may occur farther along the PDF. The accuracy of the initial predicted distribution determines the level of stringency required when setting the threshold to detect outlying peaks.

The statistical model is an aggregation of entries from the history table. Each table entry represents a counter value which can be averaged to form a PDF that represents the request statistics of individual nodes. MDS forms an average "behavior profile" of its host application and associated network, which is essentially a PDF consisting of the average of the set of individual node PDFs. Fig. 1 shows an example of a PDF generated by plotting event frequencies for a statistically representative system. Note that the types of requests are ordered in a manner to produce a normalized distribution; this ordering can take place in real time, or can be generated offline as a result of training data.

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

$$\varphi_\beta = \sum_{i=1}^{\gamma} \boldsymbol{\eta}_{\gamma\times\beta}. \tag{1}$$

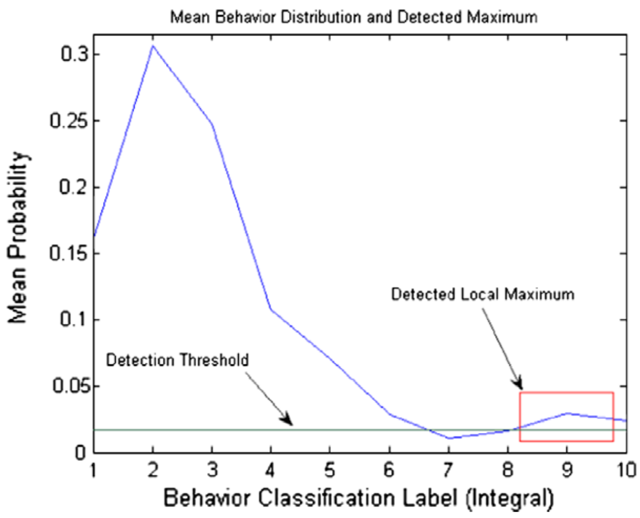


Fig. 3. Example peak detection.

Following the averaging process, ϕ is then analyzed for peaks. Since the labels are ordered and 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 lifetime of the application, system behavior will change, and the exact distribution in the mean PDF will vary. For this reason, peak detection is controlled by a sensitivity threshold. The indication of a local maxima and its threshold is shown in Fig. 3.

MDS identifies only an interaction that is exhibiting an abnormal trend; it has not yet identified the source of this aberration. To complete the identification, the history table is traversed to find the node that statistically has the greatest contribution to 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 normalized to find the most likely source of the deviant behavior in terms of average contributions.

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

3.2. The cross-correlative detection system (CCDS)

MDS relies on finding local peaks in an averaged PDF. In order to expand the functionality of the IDS, we need to add features that increase the conformity to a target application. One of the most pressing requirements is the ability to detect multiple deviant nodes, which was not possible with MDS. We must consider a different method of analysis that can return multiple results. One promising technique utilizes cross-correlation to generate numerical scores to indicate the degree of correlation among node behaviors. Of course, any method has its disadvantages;

cross-correlation depends 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 for multiple nodes.

Our approach, called the Cross-Correlative Detection System (CCDS) once again draws on data from the history table. The scores (i.e., one score per node) provide the cross-correlation of an averaged PDF for the system with the individual PDFs corresponding to each node. Let $\boldsymbol{\eta}$ represent a row-summed vector derived from the history table. The vector is then averaged. Let γ_i represent a transposed vector containing the *individual* averaged distribution of a behavior for a particular node number. $i \in m$ The scores σ_i from the resulting cross-correlation are obtained by the equation

$$\sigma_i = \boldsymbol{\eta} \cdot \gamma_i \forall i \in m. \tag{2}$$

Once the scores are obtained, CCDS 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 the range that an individual score may vary from the average score before it is considered deviant. Fig. 4 demonstrates a bounding box which represents the threshold. The middle line denotes the mean score. Note that the threshold is a certain distance from the mean score line; it is an absolute value. Scores within the bounding box are considered acceptable values, while those outside the box, which are highlighted, are considered deviant.

CCDS provides statistical comparisons through the vector cross-correlation of node interactions across the application layer. System activities that exceed a defined threshold are considered in violation of normal system activity. CCDS permits a long-term, stable understanding of the network's world model and can provide an easy way to identify normal and deviant behavior. The score-based method also allows an indirect quantization of acceptable behaviors within a system such that system design can take advantage of this detection; building a system with CCDS in mind allows for a "correct-by-design" approach to application-layer behaviors.

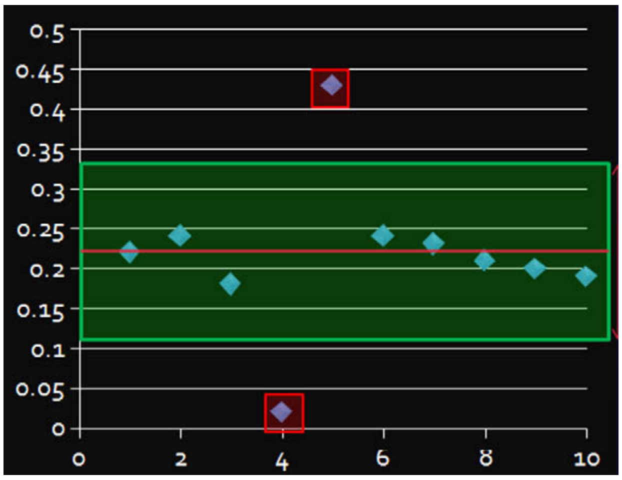


Fig. 4. Thresholding of scores.

The primary disadvantage of CCDS 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, CCDS 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.

3.3. HybrIDS

The two detection methods, MDS and CCDS, have their respective strengths and weaknesses. MDS can detect exactly one deviant agent in a short period of time without significant calibration. CCDS can detect multiple deviant agents, but requires a period of calibration in order to set its threshold correctly. An improper threshold would yield either false positives or no detection at all. We seek to combat the weaknesses of each method by combining the strengths of each detection strategy (Table 1). This combined approach is called HybrIDS. At heart, it represents the merging of the two IDS mechanisms described above; since each is fitted with different statistical methods of identifying intrusions, their merging provides a more complete assessment of the network security condition. In short, MDS calibrates CCDS for optimal detection efficiency, permitting threshold adjustment of the CCDS module to identify problem nodes effectively.

3.3.1. Scalability

An important aspect of the IDS 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. Two types of processing cycles are used: (1) data collection cycles (DCC), which do not execute analytical functions of the IDS, and (2) data processing cycles (DPC), which are compute-intensive. The method to control the number of data processing cycles relies on an understanding that a larger collection of nodes will establish an average profile over a shorter amount of time than required by a smaller collection of nodes. Therefore, given a fixed amount of time, less computational analysis is required for correct identification of deviant nodes in a larger network, whereas a network consisting of fewer nodes will require more computational analysis.

Understanding scalability then allows us to issue DPCs variably between the DCCs such that more collection cycles will pass before the analysis of a larger network. As a result, as network size increases, IDS performance will scale effectively. In our studies, computational intensity varied sub-linearly depending on the size of the network. We tested a range of network sizes, from 5 to 100 interconnected nodes, for both MDS and CCDS. We found that the implementation of a successful detection strategy while minimizing operational duty cycle became the greatest challenge in designing the system; this fact was critical when deciding how the two methods should be fused in a resource-constrained network. Since our target platform may feature nodes with limited computational resources, it is imperative that the detection phases, tuning times, and computationally-intensive portions of the system operate as efficiently as possible.

3.3.2. *Calibration within HybrIDS*

As stated previously, MDS identifies at most one suspected deviant node. This represents a recorded value that deviates significantly from the expected normal distribution of events. It is reasonable to assume that most ad-hoc operational scenarios will follow a stochastic model; we can create a normalized distribution for expected outcomes based upon offline training from the application. MDS relies on this preset distribution to identify abnormal occurrences. In contrast, CCDS 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 the stabilization of MDS. Fig. 5 shows this algorithm as a flowchart. The "HybridState" decision must be true to begin the transition process. The threshold for CCDS initially is set to represent a 100% deviation from the average score – a wide margin that most likely would not identify any positives, false or otherwise. During the transition, both MDS and CCDS run simultaneously, and both return suspected nodes. A comparison is made to determine whether the set of deviant agents from CCDS includes the one entry from MDS. Initially, except in rare scenarios, nothing will be returned for CCDS, and the corresponding

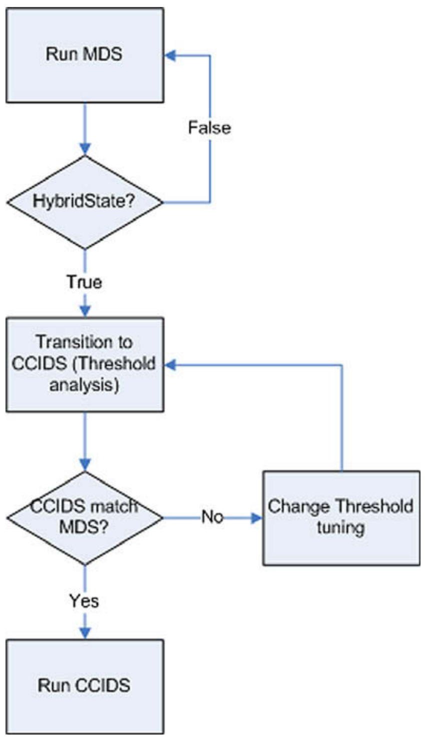


Fig. 5. Transition algorithm.

threshold for CCDS is reduced. Successive comparisons and threshold adjustments proceed until there is a match, indicating agreement between MDS and CCDS. At this point, the transition period ends, and CCDS, now properly calibrated, continues as the primary means of detection.

4. Scenarios for the experimental testbed

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 not intended as a complete list, but show potential applications of how HybrIDS adapts itself different scenarios.

4.1. *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. At the time of this writing, ADS-B is currently in extensive testing to determine its broader impact and benefits on the state of modern aviation. Therefore, there is no accessible data on security breaches, security compromises, or possible misuse of the unencrypted, broadcasted data. However, the lack of data does not eliminate security as a concern.

Table 2 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. Note that ADS-B is intended for public, civilian use. To use ADS-B as an example in this paper, we have extended its capabilities to model the operation of a network with node-to-node requests. 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
Individual method advantages and disadvantages.

Detection method	Advantages	Disadvantages
MDS	• Fast detection startup time • Low computational overhead	• Single node anomaly • identification • Unstable long-term detection
CCDS	• Multiple anomaly detection • Long-term detection stability	• Requires careful initial calibration • Somewhat more computationally-intensive

Table 2
Tradition ADS-B broadcast contents.

Broadcast category	Broadcast content	Statistical distribution
1	Position	Symmetric (2 Hz)
2	Velocity	Symmetric (2 Hz)
3	Altitude	Symmetric (2 Hz)
4	Aircraft ID	Symmetric (2 Hz)
5	Rate of climb	Symmetric (2 Hz)

Table 3
Modified ADS-B behavioral enumeration.

Request category	Request 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 addition to changing broadcasts to requests, the list of available requests was extended to ten by adding information that might enable joint operations between manned or unmanned aircraft. In this updated model (Table 3), the aircraft identification request has been removed. For simplicity, it is assumed that sender/receiver information is ascertained during the network's communication protocol. Table 3 also provides the statistical distribution from our model, which indicates the probability that each request will occur. 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. When considering that this loosely-structured ad-hoc network scenario provides information that is critical in coordinating flight paths and providing safe operation of manned or unmanned aircraft, it becomes apparent why it is an ideal target for attacks. The threat of collisions between aircraft operating together or simply sharing the same airspace leads the list of potential attack objectives. From a practicality standpoint, the most likely attack type would manifest itself as a jamming scenario. This is because it is difficult to interface with a group of moving aircraft; there may be insufficient time to crack encryption algorithms or establish an effective injection exploit with a ground-based attack. Simply jamming the communications or operation of a single node in the collective could be enough to cause catastrophic failure of either the mission, or provide a major lapse in collision avoidance protocols.

Another scenario warranting scrutiny is that of a spoofing attack designed to simulate command and control functionality originating from ground operations. Because most if not all situations in our modified ADS-B approach will require some form of ground-based interaction (whether fully autonomous or not), it is possible for an attacker to simulate or overpower ground communications in an effort to divert, disable, destroy or cause destruction with the series of networked aircraft using methods such as collisions, alterations to system objectives, and redirections. This can be achieved by successfully defeating the existing encryption methods and injecting commands from the attacking remote ground station.

Lastly, we consider the effects of defeating network security between aircraft in a non-jamming method by executing either replay attacks or misdirection to achieve a malicious objective. All of the situations shown here are for illustrative purposes only. We will focus primarily on jamming and injection attacks to demonstrate HybrIDS's detection capabilities.

4.2. Massively-distributed microrobotics

Another interesting application involves the deployment of large numbers of autonomous robots that 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. In the example, small, spherical, self-propelled devices with a power supply, primitive communication capabilities, and data gathering tools are capable of research tasks. In such a system, hundreds or thousands of the devices are deployed in order to gather data and provide detailed analysis. The robots have extremely small computing and communication resources, which are characteristics of deeply embedded systems [21].

The experiment is designed with a threshold number for device failures. It is not expected that all nodes will operate properly or be able to communicate useful data to the experiment. However, these conclusions are based on the lack of a foreign attack influence on the microrobotics device network. Should an attack, such as a jamming attack, manifest itself during deployment, the results could be catastrophic, particularly if the data collecting experiment relies on a distributed command and control mechanism. In such a scheme, a number of microrobots may be tasked with providing uplink communications. If those nodes were selectively jammed, it would terminate the uplink, leaving the cluster effectively isolated and without recourse of returning findings, with their finite powers supplies fading rapidly.

We also propose that lessons learned about attacking the system described in the microrobotics scenario can be applied to terrestrial multi-node deployments in which there exist devices with dedicated sampling instrumentation and distributed command and control structures. Such a system could apply to cooperative, multi-device marine research, or non-human-accessible natural formations, or even apply to defense and military ground-based vehicles incorporating a distributed command structure. The threat of targeted jamming attacks to disrupt uplinks still exists.

4.3. Common scenario characteristics

The modified ADS-B example and the distributed microrobotics example contain 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 operation of the interconnected nodes, and (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, and (3) low resource utilization and overhead. The two lists indicate the level of *adaptability* and *conformity*; the former demonstrates the applicability and usefulness of HybrIDS within the system context, while the latter demonstrates the efficiency of its implementation. For instance, HybrIDS may adapt well to an application, 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.

The ADS-B and distributed microrobotics examples highlight the 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.

5. Results and discussion from the experimental testbed

This section describes the method to generate data and attack the applications described in Section 4. In addition, each application is discussed to explore adaptability and conformity. We will cover the four requirements for adaptability and the three requirements for conformity.

5.1. Data generation and attack methods

Our operational and attack data is drawn from the modified ADS-B scenario listed in Section 4.1. We evaluated the potential damage of different attacks in a networked set of autonomous aircraft. Our first assumption is that an attack designed to sever ground communications would only be minimally efficient; any well-designed autonomous system is equipped to deal with an uplink communications fault for a certain period of time. We then assumed that jamming a single isolated node, while having potentially devastating consequences, would still be restricted to affecting one or two nodes in the system, unless device interdependence is considered. However, a well-designed ad-hoc network system should be capable of handling such contingencies and reorganize accordingly.

With this in mind, we decided to investigate malformed instruction injection. If done properly, this type of attack can affect the entire network, while being executed in a clandestine manner. Data generation was performed in a MATLAB script that simulated mission data based on a probability density function, with a certain amount of randomized skew to add a more realistic error variable. To simulate our injection attack, we injected ”emergency action” commands at irregular intervals, as well as “mission end” commands. Such commands are unlikely to manifest themselves under normal operations. The variance of the abnormal commands was randomized depending on the simulation. In general, these commands exhibited a 10% deviation from normal levels, which was enough to be identified accurately by MDS, and subsequently CCDS following the threshold tuning phase.

Based on experimental results, the percent pervasion of deviant nodes within the ad-hoc network created the most significant impact on IDS performance. Pervasion is defined as the total percentage of nodes known to exhibit deviant behavior within the collective. In Fig. 6, the number of nodes is increased along with the percentage of pervasion. The vertical axis shows the number of processing cycles required before identification was successful. Fig. 6 shows that an increase of pervasion affects the stabilization time of the IDS. As the increase in deviant behavior becomes more common, it reduces the separation of the scores calculated by CCDS. Since individual scores are compared to an average, the average will begin to manifest elements of the deviant behavior in a more determeintal way. As a result, more transition cycles are required to adjust the threshold more stringently, and more CCDS processing is required, since more data is required to find the correct number of deviant agents.

According to data collected for the modified ADS-B scenario, HybrIDS is capable of converging accurately on a list of suspect nodes, provided that the pervasion of deviant nodes is less than or equal to 22%. HybrIDS can converge on results with pervasions exceeding this limit, but the length of time necessary becomes difficult to predict. This is an important constraint, since non-determinacy impacts the operation of a real-time system. With these limitations in mind, the adaptability of various scenarios to use HybrIDS becomes more apparent.

5.2. Scenario discussion: Modified ADS-B

Viewing the different aspects of adaptability and conformity from Section 4.3, HybrIDS is a good match for the modified ADS-B scenario. It is capable of identifying multiple deviant agents in the scenario's ad-hoc network, and does so utilizing minimal system resources. The modified ADS-B scheme meets the discrete behavior requirement. There is a fixed set of 10 behaviors that defines the operational characteristics of the model (Table 3). A statistical distribution of actions is also present in the model because ADS-B functions are similar for all aircraft. 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 (i.e., identifying requests) is provided in that each transmission contains a sender along with the receipt of request information. We assume that identity spoofing is rendered either impossible or impractical by the communications protocol.

The requirement for high-level system abstraction is implicit from Table 3 because all interactions between the aircraft have been abstracted offline to a request type. HybrIDS utilizes integer mappings to represent

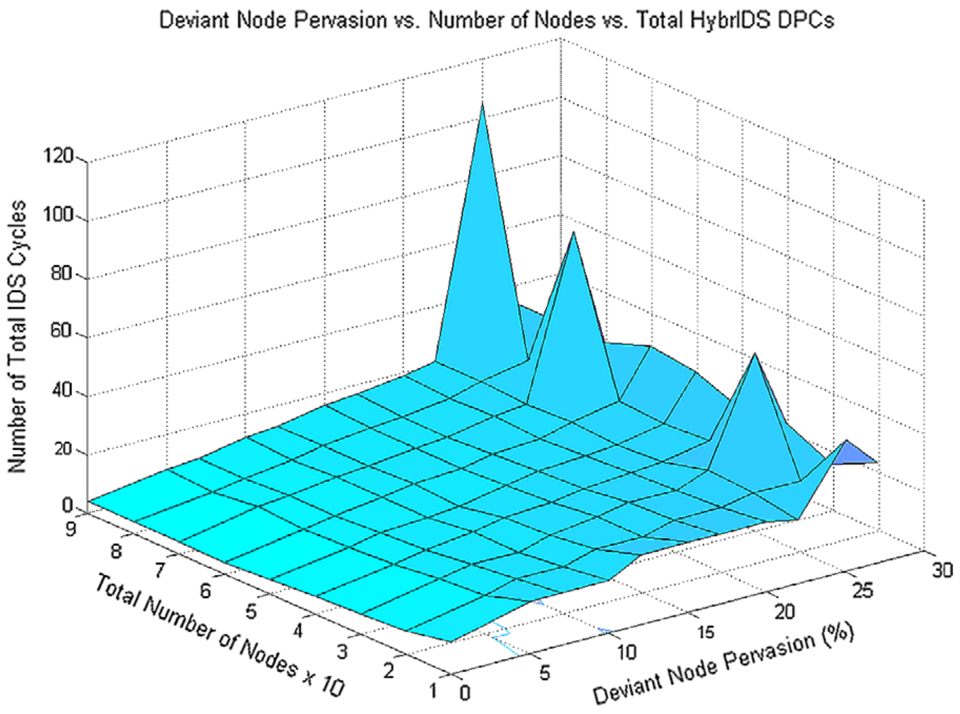


Fig. 6. HybrIDS performance for the modified ADS-B scenario.

each possible interaction. The computational power requirement is reduced by the distribution and control of data processing cycles. HybrIDS uses situational awareness to reduce the need for large and complex computations. Therefore, it can use less-powerful hardware to 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. However, 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 (KB).

In our test configuration, we utilized a 200 MHz ARM9-based development board with 64 MB 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. 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 first run with a simple Java application containing only a thread sleep command. Any memory overhead from the application would therefore be negligible. The JVM itself required 4.3 MB of memory, while HybrIDS required less than 750 KB of RAM due to various program and control structures. Together, JVM and HybrIDS required approximately 5 MB of application memory (Fig. 7). This meets the requirements of all but the most deeply embedded system architectures. Approximately 2.4 MB of RAM was utilized by the operating system kernel, requiring 7.4 MB for the system as a whole.

5.3. Scenario discussion: 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

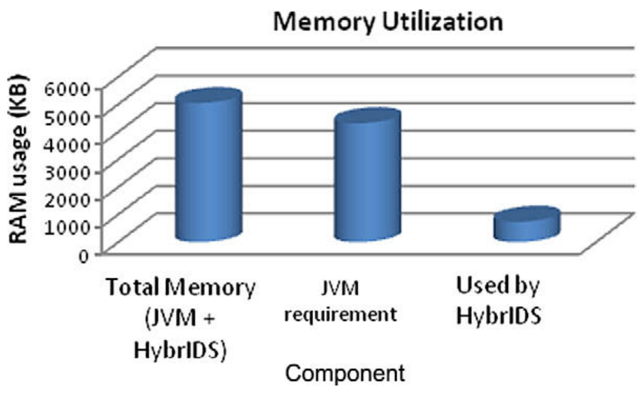


Fig. 7. Memory utilization.

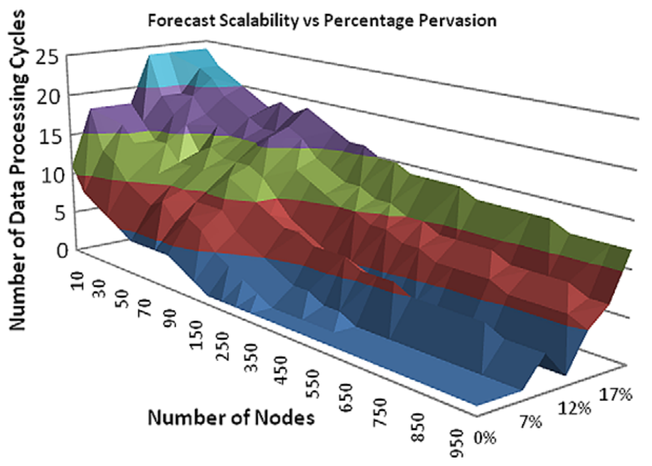


Fig. 8. Projected scalability surface plot.

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. The remaining space will be used by power supply and mobility, which severely limits the possibilities of running complex operations. We assume that operational memory size will shift from megabytes to a few kilobytes in order to conserve power. Processing, if any, would exist via an 8-bit or 16-bit microcontroller.

Given these restrictions, Java would not make a good implementation choice. Memory and processing overhead from the JVM would be too great to allow for execution to take place. While a C or assembly-based implementation of HybrIDS has not been developed, it is possible to re-factor the code into a minimalist language to bypass overhead incurred by high-level implementations. We believe 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. Fig. 8 projects the number of processing cycles required to identify deviant nodes for large networks. 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. Furthermore, the flexibility of HybrIDS 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 monitor just a few or many nodes, depending on the communication patterns it establishes within the network.

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 may not have a solution, at the time of this writing. Despite this, we are confident that the scalability and performance aspects of HybrIDS show the promise of providing enhanced security within resource-constrained networks.

6. Related work

There exists a significant body of research on anomaly-based intrusion detection [8,11,22]. Much of this work focuses upon intrusion detection 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 [23]. 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 [24]. Dwen-Ren et al. specify the use of a hybrid intrusion detection system [25]. 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.

Buchegger and Le Boudec introduce and detail different methods used to form the basis for reputation systems, applicable to the ad-hoc network scenario [26–28]. Their work focuses primarily on developing a system of node-based reputations to determine optimal and safe strategies to route data among the nodes. In their work, they show a variety of different trust-based, reputation-building mechanisms by which nodes that previously have not interacted with each other can determine whether or not the nodes are trustworthy. This reputation is based upon prior accumulated information about each respective node. Reputations must be propagated among nodes to form the basis for the group-wide consensus about the trustworthiness of the interconnected nodes. This is more applicable to an open, heterogeneous system where nodes join and leave the network. Our work focuses more on a closed, homogeneous system where the normal behavior of node interaction is well-defined.

Intrusion detection has also been developed for ad-hoc networks [14,22,29–44]. In this field, general networking techniques and computing power are no longer as applicable as in the traditional static-network-based IDS scheme. Marchang and Datta [38] introduce a collaborative IDS scheme in which message-passing builds collective information among nodes that are either directly connected, or within a one-hop-route of each other. Their approach, like that done at HRL Laboratories and Georgia Tech [12,13,45] is more central to communications and routing, while our work focuses on behavior-based IDS techniques at the application layer, which are not used to determine data routing among nodes.

The "Layered Intrusion Detection Framework" [36] 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 are performed 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 [41] 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 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 [37]. 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 proposed by HybrIDS.

In [11], Like et al. describe an application-layer intrusion detection framework that gathers information on the packet payload – which can loosely be interpreted as an application-layer analysis of the intrusion. However, this still requires individual packet decomposition, and is therefore not ideally suited for resource-constrained device networks. In contrast, HybrIDS interfaces directly with applications on the kernel level of the host device node.

Given the literature relevant to intrusion detection in mobile, ad-hoc-networks, we believe HybrIDS sets itself apart in its approach to provide detection capability from the semantic information of the network's application. Because of its scalability, low resource consumption, and portable codebase, HybrIDS is applicable to a different set of applications than those discussed in this section. It also does not exclude the use of traditional techniques, such as packet analysis, which can provide additional facets to network intrusion detection systems.

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-based 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 node which is most-likely deviant. MDS balances data collection and data processing, thereby lowering requirements for system power. A secondary IDS strategy, called CCDS, detects multiple anomalies. Its primary strength comes from the use of cross-correlative operators. Because of the nature of the cross-correlation strategy, we must choose a detection threshold that allows the IDS to detect deviant nodes successfully without introducing false positives. Inherently, this requires prior operational knowledge of the operational scenario.

HybrIDS addresses the limitation of MDS along with the calibration of CCDS by using the detection strategies in tandem. MDS is used as a calibration instrument to tune the threshold used in CCDS 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 can perform analysis with minimal impact on computational resources.

We have shown two different scenarios, with small and large network sizes, and have analyzed how HybrIDS can take advantage of both situations. In the modified ADS-B scenario, where more computational power is available, HybrIDS shows rapid convergence on a set of deviant nodes. It can identify deviant nodes up to a density of 22% of the interconnected nodes. In contrast, the microrobotics scenario presents a different challenge of integrating an IDS into a scenario with extremely limited computation. While current technology may limit its implementation, a scalability forecast shows that HybrIDS will perform well once resources match computational requirements.

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) [46]. 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.

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.

References

[1] T. Wollinger, J. Guajardo, C. Paar, Cryptography in embedded systems: an overview, in: Embedded World 2003 Exhibition and Conference, Nuernberg, Germany, 2003.

[2] R.L. Rivest, A. Shamir, L.M. Adleman, A method for obtaining digital signatures and public-key cryptosystems (Rev. Ed.), vol. 15, Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge, 1977. 1p.

[3] L. Zhou, Z.J. Haas, Securing ad hoc networks, IEEE Network 13 (1999) 24–30.

[4] P. Kocher, et al., Security as a new dimension in embedded system design, in: 41st Design Automation Conference, 2004.
[5] C. Day, Quantum computing is exciting and important-really!, Computing in Science and Engineering 9 (2) (2007) p. 104.
[6] T.S. Perry, DVD copy protection: take 2, IEEE Spectrum 42 (1) (2005) 38–39.
[7] N. Farrell, AACS decrypted, The Inquirer, 2006.
[8] A. Patcha, J.-M. Park, An overview of anomaly detection techniques: existing solutions and latest technological trends, Computer Networks 51 (12) (2007) 3448–3470.
[9] A.P. Lauf, R.A. Peters, W.H. Robinson, Intelligent intrusion detection: a behavior-based approach, in: The 21st Advanced Information Networking and Applications: Symposium for Embedded Computing, Niagara Falls, Canada, 2007.
[10] B. Wu et al., A Survey on attacks and countermeasures in mobile ad hoc networks, in: X.S.Y. Xiao, D.-Z. Du (Eds.), Wireless/Mobile Network Security, Springer, 2006.
[11] Z. Like, B.W. Gregory, Analysis of payload based application level network anomaly detection, in: 40th Annual Hawaii International Conference on System Sciences, HICSS 2007.
[12] Y. Zhang, W. Lee, Intrusion detection in wireless ad hoc networks, in: Proceedings of the sixth annual international conference on Mobile computing and networking, ACM: Boston, MA, United States, 2000.
[13] Y. Zhang, W. Lee, Y.-A. Huang, Intrusion detection techniques for mobile wireless networks, Wireless Networks 9 (5) (2003).
[14] D. Hongmei, et al. Agent-based cooperative anomaly detection for wireless ad hoc networks, in: 12th International Conference on Parallel and Distributed Systems, ICPADS, 2006.
[15] A. Yamada, et al., Intrusion detection system to detect variant attacks using learning algorithms with automatic generation of training data, in: International Conference on Information Technology: Coding and Computing, ITCC, 2005.
[16] C. Daskalakis, P. Martone, Alternative surveillance technology for the Gulf of Mexico, in: The 23rd Digital Avionics Systems Conference, DASC 04, 2004.
[17] W.H. Harman, ADS-B airborne measurements in Frankfurt, in: The 21st Digital Avionics Systems Conference, 2002.
[18] D.S. Hicok, D. Lee, Application of ADS-B for airport surface surveillance, in: The 17th AIAA/IEEE/SAE Digital Avionics Systems Conference, 1998.
[19] R. Nichols, et al. Testing of traffic information service broadcast (TIS-B) and ADS-B at Memphis International Airport, in: The 21st Digital Avionics Systems Conference, 2002.
[20] Swarming for Success, in: Astrobiology Magazine. NASA Ames., 2005
[21] K. Walther, J. Nolte, A flexible scheduling framework for deeply embedded systems, in: 21st International Conference on Advanced Information Networking and Applications Workshops, AINAW '07, 2007.
[22] J.B.D. Cabrera, C. Gutierrez, R.K. Mehra, Infrastructures and algorithms for distributed anomaly-based intrusion detection in mobile ad hoc networks, in: IEEE Conference on Military Communications, MILCOM, 2005.
[23] Z. Xiaoqiang, Z. Zhongliang, F. Pingzhi, Intrusion detection based on cross-correlation of system call sequences, in: 17th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 05, 2005.
[24] N.B. Idris, B. Shanmugam, Artificial intelligence techniques applied to intrusion detection, in: INDICON2005, 2005.
[25] T. Dwen-Ren, T. Wen-Pin, C. Chi-Fang, A hybrid intelligent intrusion detection system to recognize novel attacks, in: 37th IEEE Annual 2003 International Carnahan Conference on Security Technology, 2003.
[26] S. Buchegger, J.Y. Le Boudec, Nodes bearing grudges: towards routing security, fairness, and robustness in mobile ad hoc networks, in: 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, 2002.
[27] S. Buchegger, J.Y. Le Boudec, Performance analysis of the CONFIDANT protocol, in: Third ACM International Symposium on Mobile Ad Hoc Networking and Computing, Lausanne, Switzerland, 2002.
[28] S. Buchegger, J.Y. Le Boudec, Self-policing mobile ad hoc networks by reputation systems, in: IEEE Communications Magazine, 2005, pp. 101–107.
[29] P. Brutch, C. Ko, Challenges in intrusion detection for wireless ad hoc networks, in: Symposium on Applications and the Internet Workshops, 2003.
[30] S. Buchegger, J.Y. Le Boudee, Self-policing mobile ad hoc networks by reputation systems, Communications Magazine IEEE 43 (7) (2005) 101–107.
[31] D.R. Choudhary, et al., Computationally and resource efficient group key agreement for ad hoc sensor networks, in: Second International Conference on Communication Systems Software and Middleware, COMSWARE, 2007.
[32] O.M. Erdem, Efficient self-organized key management for mobile ad hoc networks, in: IEEE Global Telecommunications Conference, GLOBECOM '04, 2004.
[33] V. Guyot, Using WEP in ad hoc networks, in: IFIP International Conference on Wireless and Optical Communications Networks, 2006.
[34] O. Kachirski, R. Guha, Effective intrusion detection using multiple sensors in wireless ad hoc networks, in: 36th Annual Hawaii International Conference on System Sciences, 2003.
[35] O. Kachirski, R. Guha, Intrusion detection using mobile agents in wireless ad hoc networks, in: IEEE Workshop on Knowledge Media Networking, 2002.
[36] N. Komninos, C. Douligeris, LIDF: layered intrusion detection framework for ad hoc networks, Ad Hoc Networks 7 (2009) 171–182.
[37] N. Komninos, D. Vergados, C. Douligeris, Detecting unauthorized and compromised nodes in mobile ad hoc networks, Ad Hoc Networks 5 (2007) 289–298.
[38] N. Marchang, R. Datta, Collaborative techniques for intrusion detection in mobile ad hoc networks. Ad Hoc Networks, in press.
[39] A. Mishra, K. Nadkarni, A. Patcha, Intrusion detection in wireless ad hoc networks, IEEE Wireless Communications 11 (1) (2004) 48–60.
[40] J. Mundinger, J.Y. Le Boudec, Analysis of a reputation system for mobile ad hoc networks with liars, in: Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WIOPT, 2005.
[41] A. Patwardhan, J. Parker, M. Iorga, A. Joshi, T. Karygiannis, Y. Yesha, Threshold-based intrusion detection in ad hoc networks and secure AODV, Ad Hoc Networks 6 (2008) 578–599.
[42] S.A. Razak, S.M. Furnell, N.L. Clarke, P.J. Brooke, Friend-assisted intrusion detection and response mechanisms for mobile ad hoc networks, Ad Hoc Networks 6 (2008) 1151–1167.
[43] D. Watkins, C. Scott, Methodology for evaluating the effectiveness of intrusion detection in tactical mobile ad hoc networks, in: IEEE Conference on Wireless Communications and Networking Conference, WCNC, 2004.
[44] W. Xia, Intrusion detection techniques in wireless ad hoc networks, in: 30th Annual International Computer Software and Applications Conference, COMPSAC '06, 2006.
[45] Y.-A. Huang, W. Lee, Attack analysis and detection for ad hoc routing protocols, in: Recent Advances in Intrusion Detection, Springer, Berlin/Heidelberg, 2004, pp. 125–145.
[46] Generic Modeling Environment, 2000–2007, Vanderbilt University, Nashville.

Adrian P. Lauf is a Ph.D. Candidate in the Department of Electrical Engineering and Computer Science at Vanderbilt University. He received his B.E. in Computer Engineering at Vanderbilt University in 2005, his M.S. in Electrical Engineering in 2007 at Vanderbilt University, and is finishing his Ph.D. in Electrical Engineering. His research thrust is harbored within the Institute for Software Integrated Systems, a research institute part of the Electrical Engineering and Computer Science department at Vanderbilt University. As part of a group called TRUST (Team for Research in Ubiquitous Secure Technologies), he researches embedded systems security on mobile ad-hoc networks (MANETs). His current work encompasses task-to-resource reallocation for networks with compromised or disabled nodes, and the associated integration of intrusion detection technology. He is a student member of the IEEE.

Richard A. Peters is an Associate Professor of Electrical Engineering in the Department of Electrical Engineering and Computer Science. He is a member of the Center for Intelligent Systems where he directs research on the humanoid robot, ISAC, and on various mobile robots at the Cognitive Robotics Laboratory. He is also has worked with members of the Department of Psychology Cognitive Sciences Group, and the Department of Physics and Astronomy. He is a member of the NASA Robonaut research team and is the Chief Technology Officer of Universal Robotics.

William H. Robinson is an Assistant Professor in the Department of Electrical Engineering and Computer Science at Vanderbilt University. He received his B.S. in electrical engineering from Florida Agricultural and Mechanical University in 1996 and his M.S. in electrical engineering from the Georgia Institute of Technology (Georgia Tech) in 1998. He received his Ph.D. in electrical and computer engineering from Georgia Tech in 2003. His research explores hardware and software tradeoffs to improve system performance, system reliability, and system security. Topics of interest include computer architecture design, integrated circuit (IC) design, rapid prototyping using field-programmable gate arrays (FPGAs), secure hardware platform design, radiation-hardening-by-design for digital circuitry, and mitigation of single event effects. Dr. Robinson's major honors include a National Science Foundation CAREER Award and selection for DARPA's Computer Science Study Panel, both in 2008. Dr. Robinson is a Senior Member of the IEEE and participates in the Computer Society and the Education Society. He is a Senior Member of the Association for Computing Machinery (ACM), and has additional memberships in the American Society of Engineering Educators (ASEE) and the National Society of Black Engineers (NSBE).