Intrusion Detection Techniques for Mobile Wireless Networks *

Yongguang Zhang
HRL Laboratories LLC, Malibu, California
E-mail: ygz@hrl.com

Wenke Lee
College of Computing, Georgia Institute of Technology
E-mail: wenke@cc.gatech.edu

Yi-An Huang
College of Computing, Georgia Institute of Technology
E-mail: yian@cc.gatech.edu

The rapid proliferation of wireless networks and mobile computing applications has changed the landscape of network security. The traditional way of protecting networks with firewalls and encryption software is no longer sufficient and effective. We need to search for new architecture and mechanisms to protect the wireless networks and mobile computing application.

In this paper, we examine the vulnerabilities of wireless networks and argue that we must include intrusion detection in the security architecture for mobile computing environment. We have developed such an architecture and evaluated a key mechanism in this architecture, anomaly detection for mobile ad-hoc network, through simulation experiments.

Keywords: intrusion detection, intrusion response, cooperative detection, anomaly detection, mobile ad-hoc networks.

1. Introduction

The rapid proliferation of wireless networks and mobile computing applications has changed the landscape of network security. The nature of mobility creates new vulnerabilities that do not exist in a fixed wired network, and yet many of the proven security measures turn out to be ineffective. Therefore, the traditional way of protecting networks with firewalls and encryption software is no longer sufficient. We need to develop new architecture and mechanisms to protect the wireless networks and mobile computing applications.

The implication of mobile computing on network security research can be further demonstrated by the follow case. Recently (Summer 2001) an Internet worm called Code Red has spread rapidly to infect many of the Windows-based server machines. To prevent this type of worm attacks from spreading into intranets, many companies rely on firewalls to protect the internal networks. However, there are multiple incidents that the Code Red worm has been caught from within the intranet, largely due to the use of mobile computers. As more and more business travelers are carrying laptops and more and more public venues (e.g. conferences) provide wireless Internet access, there are higher and higher chances that an inadequately protected laptop will be infected with worms. For example, in a recent IETF meeting, among the hundreds of attendees that carry laptops, a dozens have been detected to be infected with Code Red worm. When these laptops are later integrated back into their company networks, they can spread the worms from within and deem the firewalls useless in defending this worm.

1.1. Vulnerabilities of Mobile Wireless Networks

The nature of mobile computing environment makes it very vulnerable to an adversary's malicious attacks. First of all, the use of wireless links renders the network

* This paper was accepted for publication in ACM MONET Journal in 2002 and appear in this issue of ACM WINET due to editorial constraints.

susceptible to attacks ranging from passive eavesdropping to active interfering. Unlike wired networks where an adversary must gain physical access to the network wires or pass through several lines of defense at firewalls and gateways, attacks on a wireless network can come from all directions and target at any node. Damages can include leaking secret information, message contamination, and node impersonation. All these mean that a wireless ad-hoc network will not have a clear line of defense, and every node must be prepared for encounters with an adversary directly or indirectly.

Second, mobile nodes are autonomous units that are capable of roaming independently. This means that nodes with inadequate physical protection are receptive to being captured, compromised, and hijacked. Since tracking down a particular mobile node in a global scale network cannot be done easily, attacks by a compromised node from within the network are far more damaging and much harder to detect. Therefore, mobile nodes and the infrastructure must be prepared to operate in a mode that trusts no peer.

Third, decision-making in mobile computing environment is sometimes decentralized and some wireless network algorithms rely on the cooperative participation of all nodes and the infrastructure. The lack of centralized authority means that the adversaries can exploit this vulnerability for new types of attacks designed to break the cooperative algorithms.

For example, many of the current MAC protocols for wireless channel access are vulnerable. Although there are many types of MAC protocols, the basic working principles are similar. In a contention-based method, each node must compete for control of the transmission channel each time it sends a message. Nodes must strictly follow the pre-defined procedure to avoid collisions and to recover from them. In a contention-free method, each node must seek from all other nodes a unanimous promise of an exclusive use of the channel resource, on a one-time or recurring basis. Regardless of the type of MAC protocol, if a node behaves maliciously, the MAC protocol can break down in a scenario resembling a denial-of-service attack. Although such attacks are rare in wired networks because the physical networks and the MAC layer are isolated from the outside world by layer-3 gateways/firewalls, every mobile node is completely vulnerable in the wireless open medium.

Furthermore, mobile computing has introduced new type of computational and communication activities that seldom appear in fixed or wired environment. For example, mobile users tend to be stingy about communication due to slower links, limited bandwidth, higher cost, and battery power constraints; mechanisms like disconnected operations [24] and location-dependent operations only appear to mobile wireless environment. Unsurprisingly, security measures developed for wired network are likely inept to attacks that exploit these new applications.

Applications and services in a mobile wireless network can be a weak link as well. In these networks, there are often proxies and software agents running in base-stations and intermediate nodes to achieve performance gains through caching, content transcoding, or traffic shaping, etc. Potential attacks may target these proxies or agents to gain sensitive information or to mount DoS attacks, such as flushing the cache with bogus references, or having the content transcoder do useless and expensive computation.

To summarize, a mobile wireless network is vulnerable due to its features of open medium, dynamic changing network topology, cooperative algorithms, lack of centralized monitoring and management point, and lack of a clear line of defense. Future research is needed to address these vulnerabilities.

1.2. The Need for Intrusion Detection

Intrusion prevention measures, such as encryption and authentication, can be used in ad-hoc networks to reduce intrusions, but cannot eliminate them. For example, encryption and authentication cannot defend against compromised mobile nodes, which often carry the private keys. Integrity validation using redundant information (from different nodes), such as those being used in secure routing [25,27], also relies on the trustworthiness of other nodes, which could likewise be a weak link for sophisticated attacks.

The history of security research has taught us a valuable lesson – no matter how many intrusion prevention measures are inserted in a network, there are always some weak links that one could exploit to break in (just like the example at the beginning of this paper). Intrusion detection presents a second wall of defense and it is a necessity in any high-survivability network.

In summary, mobile computing environment has inherent vulnerabilities that are not easily preventable.

To secure mobile computing applications, we need to deploy intrusion detection and response techniques, and further research is necessary to adapt these techniques to the new environment, from their original applications in fixed wired network. In this paper, we focus on a particular type of mobile computing environment called mobile ad-hoc networks and propose a new model for intrusion detection and response for this environment. We will first give a background on intrusion detection, then present our new architecture, followed by an experimental study to evaluate its feasibility.

2. Intrusion Detection and the Challenges of Mobile Ad-Hoc Networks

2.1. Background on Intrusion Detection

When an intrusion (defined as "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource" [8]) takes place, intrusion prevention techniques, such as encryption and authentication (e.g., using passwords or biometrics), are usually the first line of defense. However, intrusion prevention alone is not sufficient because as systems become ever more complex, and as security is still often the after-thought, there are always exploitable weaknesses in the systems due to design and programming errors, or various "socially engineered" penetration techniques. For example, even though they were first reported many years ago, exploitable "buffer overflow" security holes, which can lead to an unauthorized root shell, still exist in some recent system softwares. Furthermore, as illustrated by the Distributed Denial-of-Services (DDoS) attacks launched against several major Internet sites where security measures were in place, the protocols and systems that are designed to provide services (to the public) are inherently subject to attacks such as DDoS. Intrusion detection can be used as a second wall to protect network systems because once an intrusion is detected, e.g., in the early stage of a DDoS attack, response can be put into place to minimize damages, gather evidence for prosecution, and even launch counter-attacks.

The primary assumptions of intrusion detection are: user and program activities are observable, for example via system auditing mechanisms; and more importantly, normal and intrusion activities have distinct behavior. Intrusion detection therefore involves capturing audit data and reasoning about the evidence in the data to determine whether the system is under attack. Based on the type of audit data used, intrusion detection systems (IDSs) can be categorized as network-based or host-based. A network-based IDS normally runs at the gateway of a network and "captures" and examines network packets that go through the network hardware interface. A host-based IDS relies on operating system audit data to monitor and analyze the events generated by programs or users on the host. Intrusion detection techniques can be categorized into *misuse detection* and *anomaly detection*.

Misuse detection systems, e.g., IDIOT [15] and STAT [9], use patterns of well-known attacks or weak spots of the system to match and identify known intrusions. For example, a signature rule for the "guessing password attack" can be "there are more than 4 failed login attempts within 2 minutes". The main advantage of misuse detection is that it can accurately and efficiently detect instances of known attacks. The main disadvantage is that it lacks the ability to detect the truly innovative (i.e., newly invented) attacks.

Anomaly detection (sub)systems, for example, the anomaly detector in IDES [18], flag observed activities that deviate significantly from the established normal usage profiles as anomalies, i.e., possible intrusions. For example, the normal profile of a user may contain the averaged frequencies of some system commands used in his or her login sessions. If for a session that is being monitored, the frequencies are significantly lower or higher, then an anomaly alarm will be raised. The main advantage of anomaly detection is that it does not require prior knowledge of intrusion and can thus detect new intrusions. The main disadvantage is that it may not be able to describe what the attack is and may have high false positive rate.

2.2. Problems of Current IDS Techniques

The vast difference between the fixed network where current intrusion detection research are taking place and the mobile ad-hoc network which is the focus of this paper makes it very difficult to apply intrusion detection techniques developed for one environment to another. The most important difference is perhaps that the latter does not have a fixed infrastructure, and today's network-based IDSs, which rely on real-time traffic analysis, can no longer function well in the new en-

vironment. Compared with wired networks where traffic monitoring is usually done at switches, routers and gateways, the mobile ad-hoc environment does not have such traffic concentration points where the IDS can collect audit data for the entire network. Therefore, at any one time, the only available audit trace will be limited to communication activities taking place within the radio range, and the intrusion detection algorithms must be made to work on this partial and localized information.

Another significant big difference is in the communication pattern in a mobile computing environment. As we have mentioned earlier, mobile users tend to be stingy about communication and often adopt new operation modes such as disconnected operations [24]. This suggests that the anomaly models for wired network cannot be used as is.

Furthermore, there may not be a clear separation between normalcy and anomaly in mobile environment. A node that sends out false routing information could be the one that has been compromised, or merely the one that is temporarily out of sync due to volatile physical movement. Intrusion detection may find it increasingly difficult to distinguish false alarms from real intrusions.

In summary, we must answer the following research questions in developing a viable intrusion detection system for mobile ad-hoc networks:

- What is a good system architecture for building intrusion detection and response systems that fits the features of mobile ad-hoc networks?
- What are the appropriate audit data sources? How do we detect anomaly based on partial, local audit traces – if they are the only reliable audit source?
- What is a good model of activities in a mobile computing environment that can separate anomaly when under attacks from the normalcy?

3. An Architecture for Intrusion Detection

Intrusion detection and response systems should be both distributed and cooperative to suite the needs of mobile ad-hoc networks. In our proposed architecture (Figure 1), every node in the mobile ad-hoc network participates in intrusion detection and response. Each node is responsible for detecting signs of intrusion locally and independently, but neighboring nodes can collaboratively investigate in a broader range.

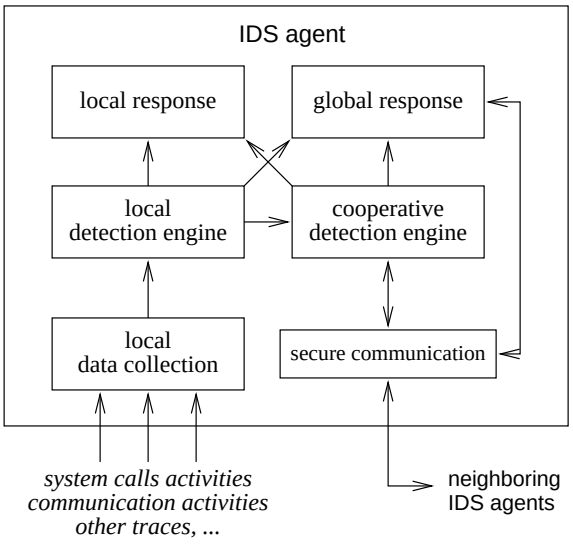

Figure 2. A Conceptual Model for an IDS Agent

In the systems aspect, individual IDS agents are placed on each and every node. Each IDS agent runs independently and monitors local activities (including user and systems activities, and communication activities within the radio range). It detects intrusion from local traces and initiates response. If anomaly is detected in the local data, or if the evidence is inconclusive and a broader search is warranted, neighboring IDS agents will cooperatively participate in global intrusion detection actions. These individual IDS agent collectively form the IDS system to defend the mobile ad-hoc network.

The internal of an IDS agent can be fairly complex, but conceptually it can be structured into six pieces (Figure 2). The data collection module is responsible for gathering local audit traces and activity logs. Next, the local detection engine will use these data to detect local anomaly. Detection methods that need broader data sets or that require collaborations among IDS agents will use the cooperative detection engine. Intrusion response actions are provided by both the local response and global response modules. The local response module triggers actions local to this mobile node, for example an IDS agent alerting the local user, while the global one coordinates actions among neighboring nodes, such as the IDS agents in the network electing a remedy action. Finally, a secure communication module provides a high-confidence communication channel among IDS agents.

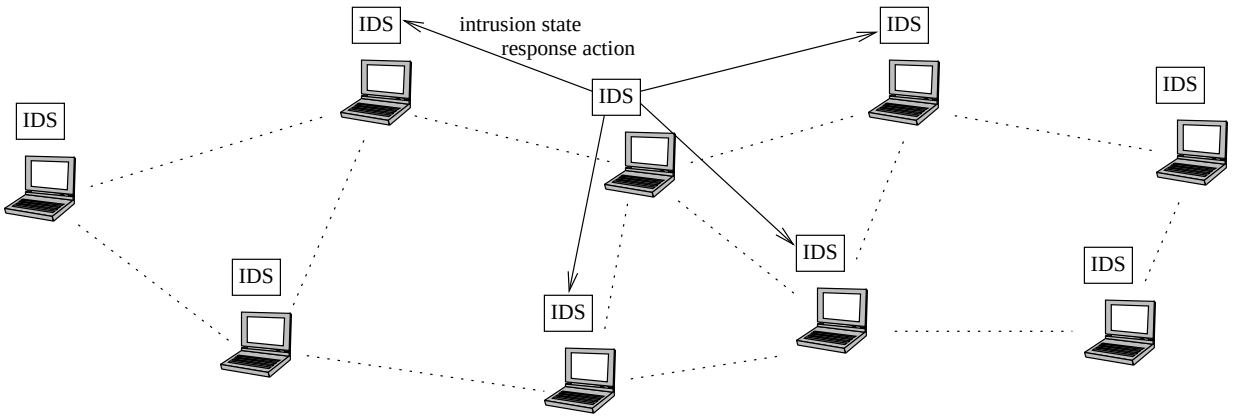

Figure 1. The IDS Architecture for Wireless Ad-Hoc Network

3.1. Data Collection

The first module, local data collection, gathers streams of real-time audit data from various sources. Depending on the intrusion detection algorithms, these useful data streams can include system and user activities within the mobile node, communication activities by this node, as well as communication activities within the radio range and observable by this node. Therefore, multiple data collection modules can coexist in one IDS agents to provide multiple audit streams for a multi-layer integrated intrusion detection method (Section 3.5).

3.2. Local Detection

The local detection engine analyzes the local data traces gathered by the local data collection module for evidence of anomalies. It can include both misuse detections or anomaly detection (Section 2.1). Because it is conceivable that the number of newly created attack types mounted on mobile computing environment will increase quickly as more and more network appliances become mobile and wireless, anomaly detection techniques will play a bigger role. We will have further discussion on anomaly detection in mobile wireless environment in Section 4.

3.3. Cooperative Detection

Any node that detects locally a known intrusion or anomaly with *strong* evidence (i.e., the detection rule triggered has a very high accuracy rate, historically), can determine independently that the network is under attack and can initiate a response. However, if a node detects an anomaly or intrusion with weak evidence, or the evidence is inconclusive but warrants broader investigation, it can initiate a cooperative global intrusion detection procedure. This procedure works by propagating the intrusion detection state information among neighboring nodes (or further downward if necessary).

The intrusion detection state information can range from a mere level-of-confidence value such as

- "With p% confidence, node A concludes from its local data that there is an intrusion"
- "With p% confidence, node A concludes from its local data and neighbor states that there is an intrusion"
- "With p% confidence, node A, B, C, ... collectively conclude that there is an intrusion"

to a more specific state that lists the suspects, like

- "With p% confidence, node A concludes from its local data that node X has been compromised"

or to a complicated record including the complete evidence.

As the next step, we can derive a distributed consensus algorithm to compute a new intrusion detection state for this node, using other nodes' state information received recently. The algorithm can include a weighted computation under the assumption that nearby nodes have greater effects than far away nodes, i.e., giving the immediate neighbors the highest values in evaluating the intrusion detection states.

For example, a majority-based distributed intrusion detection procedure can include the following steps:

- the node sends to neighboring node an "intrusion (or anomaly) state request";

- each node (including the initiation node) then propagates the state information, indicating the likelihood of an intrusion or anomaly, to its immediate neighbors;
- each node then determines whether the *majority* of the received reports indicate an intrusion or anomaly; if yes, then it concludes that the network is under attack;
- *any* node that detects an intrusion to the network can then initiate the response procedure.

The rationales behind this scheme are as follows. Audit data from other nodes cannot be trusted and should not be used because the compromised nodes can send falsified data. However, the compromised nodes have no incentives to send reports of intrusion/anomaly because the intrusion response may result in their expulsion from the network. Therefore, unless the *majority* of the nodes are compromised, in which case one of the legitimate nodes will probably be able to detect the intrusion with strong evidence and will respond, the above scheme can detect intrusion even when the evidence at individual nodes is weak.

A mobile network is highly dynamic because nodes can move in and out of the network. Therefore, while each node uses intrusion/anomaly reports from other nodes, it does not rely on fixed network topology or membership information in the distributed detection process. It is a simple *majority* voting scheme where *any* node that detects an intrusion can initiate a response.

3.4. Intrusion Response

The type of intrusion response for mobile ad-hoc networks depends on the type of intrusion, the type of network protocols and applications, and the confidence (or certainty) in the evidence. For example, here is a few likely response:

- Re-initializing communication channels between nodes (e.g, force re-key).
- Identifying the compromised nodes and re-organizing the network to preclude the promised nodes.

For example, the IDS agent can notify the end-user, who may in turn do his/her own investigation and take appropriate action. It can also send a "re-authentication" request to all nodes in the network to prompt the end-users to authenticate themselves (and hence their mobile nodes), using out-of-bound mechanisms (like, for example, visual contacts). Only the re-authenticated nodes, which may collectively negotiate a new communication channel, will recognize each other as legitimate. That is, the compromised/malicious nodes can be excluded.

3.5. Multi-Layer Integrated Intrusion Detection and Response

Traditionally, IDSs use data only from the lower layers: network-based IDSs analyze TCP/IP packet data and host-based IDSs analyze system call data. This is because in wired networks, application layer firewalls can effectively prevent many attacks, and application-specific modules, e.g., credit card fraud detection systems, have also been developed to guard the mission-critical services.

In the wireless networks, there are no firewalls to protect the services from attack. However, intrusion detection in the application layer is not only feasible, as discussed in the previous section, but also necessary. Certain attacks, for example, an attack that tries to create an unauthorized access "back-door" to a service, may seem perfectly legitimate to the lower layers, e.g., the MAC protocols. We also believe that some attacks may be detected much earlier in the application layer, because of the richer semantic information available, than in the lower layers. For example, for a DoS attack, the application layer may detect very quickly that a large number of incoming service connections have no actual operations or the operations don't make sense (and can be considered as errors); whereas the lower layers, which rely only on information about the amount of network traffic (or the number of channel requests), may take a longer while to recognize the unusually high volume.

Given that there are vulnerabilities in multiple layers of mobile wireless networks and that an intrusion detection module needs to be placed at each layer on each node of a network, we need to coordinate the intrusion detection and response efforts. We use the following integration scheme:

- if a node detects an intrusion that affects the entire network, e.g., when it detects an attack on the ad hoc routing protocols, it initiates the re-authentication

process to exclude the compromised/malicious nodes from the network;

- if a node detects a (seemingly) local intrusion at a higher layer, e.g., when it detects attacks to one of its services, lower layers are notified. The detection modules there can then further investigate, e.g., by initiating the detection process on possible attacks on ad hoc routing protocols, and can respond to the attack by blocking access from the offending node(s) and notifying other nodes in the network of the incident.

In this approach, the intrusion detection module at each layer still needs to function properly, but detection on one layer can be initiated or aided by evidence from other layers. As a first cut of our experimental research, we allow the evidence to flow from one layer to its (next) lower layer by default, or to a specific lower layer based on the application environment.

The "augmented" versions of the detection model at a lower level are constructed as follows. In the "testing" process, the anomaly decision, i.e., either 1 for "yes" or 0 for "no" from the upper layer is inserted into the deviation score of the lower level, for example, (0.1, 0.1) now becomes (0.1, 0.1, 0). In other words, the deviation data also carries the extra information passed from the upper level. An anomaly detection model built from the augmented data therefore combines the bodies of evidence from the upper layers and the current layer and can make a more informed decision. The intrusion report sent to other node for cooperative detection also includes a vector of the information from the layers.

With these new changes, the lower layers now need more than one anomaly detection model: one that relies on the data of the current layer and therefore indirectly uses evidence from the lower layers, and the augmented one that also considers evidence from the upper layer.

Multi-layer integration enables us to analyze the attack scenario in its entirety and as a result, we can achieve better performance in terms of both higher true positive and lower false positive rates. For example, a likely attack scenario is that an adversary takes control of the mobile unit of a user (by physically disable him or her), and then uses some system commands to send falsified routing information. A detection module that monitors user behavior, e.g., via command usage, can detect this event and immediately (i.e., before further damage can be done) cause the detection module for the routing protocols to initiate the global detection and response, which can result in the exclusion of this compromised unit. As another example, suppose the users are responding to a fire alarm, which is a rare event and may thus cause a lot of unusual movements and hence updates to the routing tables. However, if there is no indication that a user or a system software has been compromised, each intrusion report sent to other nodes will have a "clean" vector of upper layer indicators, and thus the detection module for the routing protocols can conclude that the unusual updates may be legitimate.

4. Anomaly Detection in Mobile Ad-Hoc Networks

In this section, we discuss how to build anomaly detection models for mobile wireless networks. Detection based on activities in different network layers may differ in the format and the amount of available audit data as well as the modeling algorithms. However, we believe that the principle behind the approaches will be the same. To illustrate our approach, we focus our discussions on ad-hoc routing protocols.

4.1. Building an Anomaly Detection Model

Framework The basic premise for anomaly detection is that there is intrinsic and observable characteristic of normal behavior that is distinct from that of abnormal behavior. We use information-theoretic measures [5], namely, entropy and conditional entropy, to describe the characteristics of normal information flows and to use classification algorithms to build anomaly detection models. For example, we can use a classifier, trained using normal data, to predict what is *normally* the next event given the previous n events. In monitoring, when the actual event is not what the classifier has predicted, there is an anomaly. When constructing a classifier, features with high *information gain* (or reduction in entropy) [20] are needed. That is, a classifier needs feature value tests to partition the original (mixed and high entropy) dataset into pure (and low entropy) subsets, each ideally with one (correct) class of data.

Using this framework, we employ the following procedure for anomaly detection: a) select (or partition) audit data so that the normal dataset has low (conditional) entropy; b) perform appropriate data transformation according to the entropy measures (e.g., con-

structing new features with high information gain); c) compute classifier using training data; d) apply the classifier to test data; and e) post-process alarms to produce intrusion reports.

Attack Models In this study, we only consider attacks on routing protocols. In general, these attacks are in the following forms [26]:

1. *Route logic compromise* This type of attacks behaves by manipulating routing information, either externally by parsing false route messages or internally by maliciously changing routing cache information. In particular, we consider several special cases: (a) misrouting: forwarding a packet to an incorrect node; and (b) false message propagation: distributing a false route update.
2. *Traffic pattern distortion* This type of attacks changes default/normal traffic behavior: (a) packet dropping; (b) packet generation with faked source address; (c) corruption on packet contents; and (d) denial-of-service.

Notice that these two kinds of attacks can be combined together in a single intrusion. In our study, we implemented the following attacks in simulation: (1) falsifying route paths/route entry in a node's own cache; and (2) random packet dropping. The first is an abstraction of routing attacks because they resort to changing the routing information for malicious purposes. The second is simply traffic pattern distortion. Each intrusion session we simulated includes only one of these attack types. However, each execution trace can contain several intrusion sessions with different attack types.

Audit Data We suggest these two **local** data sources be used for anomaly detection: (1) local routing information, including cache entries and traffic statistics; and (2) position locator, or GPS, which we assume will not be compromised and can therefore reliably provide location and velocity information of nodes within the whole neighborhood. We use only local information because remote nodes can be compromised and their data cannot be trusted.

Feature Selection Feature selection is a critical step in building a detection model. Specifically, since we use classifiers as detectors, we need to select and/or construct features, from the available audit data, that have high information gain. The criteria of information gain is not *a priori*. We use an unsupervised method to construct the feature set. First, we constructed a large feature set to cover a wide range of behaviors. It is not efficient to run all experiments with all of these features. A small number of training runs can be conducted with the whole set of features on small audit data traces randomly chosen from a previously stored audit logs. For each training run, a corresponding model is built. The features that appear in the models and has weights not smaller than a minimum threshold are selected into the **essential feature set**. For different routing protocols and different scenarios, the essential feature set is different.

In practice, we expect the feature set needs to be updated after certain period, as the characteristics of routing behavior can change with time.

Classifier We use two classifiers in our study. One is a decision-tree equivalent classifier, RIPPER [4], a rule induction program. The other is a Support Vector Machine classifier, SVM Light [11]. RIPPER is a typical classifier in that it searches the given feature space and computes rules that separate data into appropriate (intended) classes. SVM Light instead pre-processes the data to represent patterns in much higher dimension than the given feature space. The heuristic is that with sufficiently high dimension, data can be separated by a hyperplane, thus achieving the goal of classification. SVM Light can produce a more accurate classifier than RIPPER when there are underlying complex patterns in the data that are not readily represented by the given set of features.

Post-processing Given an execution trace, we first apply a detector to examine each observation. Then a post-processing scheme is used to examine the predictions and generate intrusion reports: (1) choose a parameter l and let the window size be $2l + 1$; (2) for a region covered by the current window, if there are more abnormal predictions than normal predictions, i.e, the number of abnormal observations is greater than l, then the region is marked as "abnormal"; (3) label every observation within an abnormal region as "abnormal" and every observation within a normal region as "normal"; (4) shift the sliding window with one window size, i.e, $2l + 1$, and repeat (2) and (3) until the whole trace is processed; (5) count all continuous abnormal regions as

one intrusion session. In our experiments, we use $l = 3$.

The intuition here is that a detection model can make spurious errors and these false alarms should be filtered out. In contrast, a true intrusion session has "locality", i.e., it tends to result in many alarms within a short time window. Therefore, these alarms can be grouped into a single intrusion report.

4.2. Detecting Abnormal Updates to Routing Tables

The main requirement of an anomaly detection model, and intrusion detection systems in general, is low false positive rate, calculated as the percentage of normalcy variations detected as anomalies, and high true positive rate, calculated as the percentage of anomalies detected. We need to first determine the trace data to be used that will bear evidence of normalcy or anomaly. For ad-hoc routing protocols, since the main concern is that the false routing information generated by a compromised node will be disseminated to and used by other nodes, we can define the trace data to describe, for each node, the normal (i.e., legitimate) updates of routing information.

A routing table usually contains, at the minimum, the next hop to each destination node and the distance (number of hops). A legitimate change in the routing table can be caused by the physical movement(s) of node(s) or network membership changes. For a node, its own movement and the change in its own routing table are the only *reliable* information that it can trust. Hence, we use data on the node's physical movements and the corresponding change in its routing table as the basis of the trace data. The physical movement is measured mainly by distance and velocity (this data can be obtained by a built-in GPS device). The routing table change is measured mainly by the percentage of changed routes (PCR), the (positive or negative) percentage of changes in the sum of hops of all the routes (PCH), and the percentage of newly added routes. We use percentages as measurements because of the dynamic nature of mobile networks (i.e., the number of nodes/routes is not fixed). Table 1 shows some fictional trace data for a node.

Table 1
Sample Trace Data for Ad-Hoc Routing

Distance	Velocity	PCR	PCH	...
0.01	0.1	20	15	...
10	20	80	50	...
0.02	0.1	0	0	...
...	...	...	...	...

During the "training" process, where a diversity of normal situations are simulated, the trace data is gathered for each node. The trace data sets of all nodes in the training network are then aggregated into a single data set, which describes all normal changes in routing tables for all the nodes. A detection model which is learned from this aggregated data set will therefore be capable of operating on any node in the network.

A normal profile on the trace data in effect specifies the correlation of physical movements of the node and the changes in the routing table. We can use the following scheme to compute the normal profile:

- denote PCR the *class* (i.e. *concept*), and distance, velocity, and PCH, etc. the *features* describing the concept;
- use n classes to represent the PCR values in n ranges, for example, we can use 10 classes each representing 10 percentage points – that is, the trace data belongs to n classes;
- apply a classification algorithm to the data to learn a classifier for PCR;
- repeat the above for PCH, that is, learn a classifier for PCH;

A classification algorithm, e.g., RIPPER [4], can use the most discriminating feature values to describe each concept. For example, when using PCR as the concept, RIPPER can output classification rules in the form of: "if (distance $\leq$ 0.01 AND PCH $\leq$ 20 AND ...) then PCR = 2; else if ...". Each classification rule (an "if") has a "confidence" value, calculated as the percentage of records that match both the rule condition and rule conclusion out of those that match the rule condition. The classification rules for PCR and PCH together describe what are the (normal) conditions that correlate with the (amount of) routing table changes. We use these rules as the normal profiles.

Checking an observed trace data record (that records a routing table update) with the profile involves applying the classification rules to the record. A misclassification, e.g., when the rules say it is "PCR = 3" but in fact it is "PCR = 5", is counted as a violation. We can use the "confidence" of the violated rule as the "deviation score" of the record. In the "testing" process, the

Table 2
Sample Deviation Data

PCR deviation	PCH deviation	Class
0.0	0.0	normal
0.1	0.0	normal
0.2	0.2	normal
0.9	0.5	abnormal
0.3	0.1	normal
...	...	...

deviation scores are recorded. For example, if abnormal data is available, we can have deviation data like those shown in Table 2. We can then apply a classification algorithm to compute a classifier, a detection model, that uses the deviation scores to distinguish abnormal from normal.

If abnormal data is not available, we can compute the normal *clusters* of the deviation scores, where each score pair is represented by a point (*PCR deviation*, *PCH deviation*) in the two-dimensional space, e.g., (0.0, 0.0), (0.2, 0.2), (0.3, 0.1), etc. The "outliers", i.e., those that do not belong to any normal cluster, can then be considered as anomalies. Clustering is often referred to as "un-supervised learning" because the target clusters are not known *a priori*. Its disadvantage is that the computation (i.e., the formation) of clusters is very time consuming. If the application environment allows a *tolerable* false alarm rate, e.g., 2%, then the clustering algorithm can be parameterized to terminate when sufficient, e.g., greater than 98%, points are in proper clusters.

A poor performance of the anomaly detection model, e.g., a higher than acceptable false alarm rate, indicates that the data gathering (including both "training" and "testing" processes) is not sufficient, and/or the features and the modeling algorithms need to be refined. Therefore, repeated trials may be needed before a good anomaly detection model is produced.

In the discussion thus far, we have used only the *minimal* routing table information in the anomaly detection model to illustrate our approach, which can be applied to all routing protocols. For a specific protocol, we can use additional routing table information and include new features in the detection model to improve the performance. For example, for DSR ad-hoc routing protocol [13,19], we can add source route information (the complete, ordered sequence of network hops leading to the destination). We can also add predictive features according to the "temporal and statistical" patterns among the routing table updates, following the similar *feature construction* process we used to build intrusion detection models for wired networks [17]. For example, for a wired TCP/IP network, a "SYN-flood" DoS attack has a pattern which indicates that a lot of half-open connections are attempted against a service in a short time span. Accordingly, a feature, "for the past 2 seconds, the percentage of connections to the same service that are half-open" was constructed and had been proved to be highly predictive. Similarly, in a mobile network, if an intrusion results in a large number of routing table updates, we can add a feature that measures the frequency (how often) the updates take place.

Our objective in this study is to lead to a better understanding of the important and challenging issues in intrusion detection for ad-hoc routing protocols. First, using a given set of training, testing, and evaluation scenarios, and modeling algorithms (e.g., with RIPPER as the classification algorithm for protocol trace data and "nearest neighbor" as the clustering algorithm for deviation scores), we can identify which routing protocol, with potentially all its routing table information used, can result in better performing detection models. This will help answer the question "what information should be included in the routing table to make intrusion detection effective." This finding can be used to design more robust routing protocols. Next, using a given routing protocol, we can explore the feature space and algorithm space to find the best performing model. This will give insight to the general practices of building intrusion detection for mobile networks.

4.3. Detecting Abnormal Activities in Other Layers

Anomaly detection for other layers of the wireless networks, e.g., the MAC protocols, the applications and services, etc., follows a similar approach. For example, the trace data for MAC protocols can contain the following features: for the past s seconds, the total number of channel requests, the total number of nodes making the requests, the largest, the mean, and the smallest of all the requests, etc. The *class* can be the range (in the number) of the current requests by a node. A classifier on this trace data describes the normal context (i.e. history) of a request. An anomaly detection model can

then be computed, as a classifier or clusters, from the deviation data.

Similarly, at the mobile application layer, the trace data can use the service as the *class* (i.e., one class for each service), and can contain the following features: for the past s seconds, the total number of requests to the same service, the number of different services requested, the average duration of the service, the number of nodes that requested (any) service, the total number of service errors, etc. A classifier on the trace data then describes for each service the normal behaviors of its requests.

Many attacks generate different statistical patterns than normal requests. Since the features described above are designed to capture the statistical behavior of the requests, the attacks, when examined using the feature values, will have large deviations than the normal requests. For example, compared with normal requests to MAC or an application-level service, DoS attacks via resource exhaustion normally involve a huge number of requests in a very short period of time; a DDoS has the additional tweak that it comes from many different nodes.

5. Experimental Results

To study the feasibility of our security architecture, we have implemented anomaly detection in a network simulator and conducted a series of experiments to evaluate its effectiveness. We choose three specific ad-hoc wireless protocols as the subjects of our study. They are Dynamic Source Routing (DSR) protocol [12,13,3], Ad-hoc On-Demand Distance Vector Routing (AODV) protocol [22], and Destination-Sequenced Distance-Vector Routing (DSDV) protocol [21], They are selected as they represent different types of ad-hoc wireless routing protocols, proactive and on-demand. We now show how our anomaly detection methods can be applied to these protocols and demonstrate the effectiveness of our models can be used on other different scenarios.

We used the wireless networks simulation software, from *Network Simulator ns-2*[1] [6]. It includes simulation for wireless ad-hoc network infrastructure, popular wireless ad-hoc routing protocols (DSR, DSDV, AODV and others), and mobility scenario and traffic pattern generation.

[1] release 2.1b7a, December 2000

5.1. Features Selection

The decision to pick features rely on several factors. It should reflect information from several sources, i.e, from traffic pattern, from routing change, and from topological movement. In order to compare among different protocols, we use a similar feature set for all. Generally we consider same sets for traffic and topological information, but allow slight deviation to make maximum utilization of routing information. Even under the same measure, different protocols interpret it in a slight different manner. For instance, PCH is the percentage of change in number of total intermediate hops from all source routes cached in DSR, but the percentage of change of sum of metrics to all reachable destinations in DSDV and AODV.

We used the following features in Table 3 to build models on ad-hoc routing protocols. Notice features are collected from three sources, route cache, traffic pattern, and movement information of the host. All features are locally collected.

To simplify the modeling task, all quantities are discretized. All features other than velocity and distance are separated into six levels, from 0%-20%,20%-40%,..., 80%-100%, and higher. Velocity and distance features are discretized into 10 levels.

5.2. Models

We then build models using two classification algorithms, the traditional induction based classifier RIPPER and a new SVM classifier SVM_Light.

First, models are trained off-line using training data from one of our simulations with pure normal data with running time of 100,000 seconds.

5.3. Data

To test our models, we used five different test scripts to generate traces. *normal* is a normal trace, *100k-rt* and *10k-rt* are traces with intrusions on routing logic and with running time as 100,000 and 10,000 seconds respectively. *100k-tf* and *10k-tf* are traces with distortion on traffic patterns. *10k-rt* and *10k-tf* contain at most 10 intrusion sessions, while *100k-rt* and *100k-tf* contain at most 100 intrusion sessions. All results are filtered through post-processing procedure. For each result, we run ten times and report its average and error under 95% confidence level.

Table 3
Local Features on Ad-Hoc Protocols

Features	Explanation	Features	Explanation
PSTC	% of Significant Traffic Change		
VELOCITY	Velocity	DISTANCE	Distance from last log
RDC	Relative Distance Change		
PCR	% of Change in Route entries	PCH	% of Change in number of Hops
DSDV Specific			
PCB	% of Change of Bad (unreachable) destinations	PCU	% of Change of Updated destinations
PCS	% of Change of Stale destinations		
DSR S/pecific			
PCB	% of Change of Bad routes	PCU	% of Change of Updated routes
PCS	% of Change of Stale routes		
AODV Specific			
PCB	% of Change of Bad routes	PCU	% of Change of Updated routes
PCS	% of Change of Stale routes		

Table 4
Detection performance on DSR with five test scripts

Trace	RIPPER		SVM_Light	
	Detection Rate	False Alarm Rate	Detection Rate	False Alarm Rate
normal	N/A	1.39 ± 0.98%	N/A	0.0710 ± 0.053%
100k-rt	90.7 ± 3.24%	15.3 ± 4.08%	99.1 ± 0.16%	0.0667 ± 0.002%
100k-tf	85.2 ± 2.38%	13.7 ± 4.30%	99.1 ± 0.09%	0.0556 ± 0.022%
10k-rt	90.9 ± 3.07%	9.56 ± 4.27%	99.1 ± 0.37%	0.0360 ± 0.042%
10k-tf	89.8 ± 4.23%	10.12 ± 5.53%	99.0 ± 0.33%	0.0533 ± 0.065%

Table 5
Detection performance on DSDV with five test scripts

Trace	RIPPER		SVM_Light	
	Detection Rate	False Alarm Rate	Detection Rate	False Alarm Rate
normal	N/A	5.37 ± 3.10%	N/A	6.01 ± 1.41%
100k-rt	88.34 ± 5.03%	23.8 ± 7.41%	86.0 ± 0.96%	26.3 ± 5.49%
100k-tf	90.61 ± 2.99%	24.1 ± 6.70%	85.6 ± 0.83%	25.5 ± 2.05%
10k-rt	87.93 ± 4.31%	15.8 ± 4.32%	85.3 ± 4.82%	20.5 ± 10.0%
10k-tf	85.23 ± 3.28%	14.5 ± 4.87%	84.4 ± 0.60%	23.4 ± 5.78%

The results in Table 4, 5 and 6 are the detection rates and false alarms rates in terms of individual records (not intrusion sessions). It is interesting to observe that DSR with its results by SVM outperforms the other two a lot,

Table 6
Detection performance on AODV with five test scripts

Trace	RIPPER		SVM_Light	
	Detection Rate	False Alarm Rate	Detection Rate	False Alarm Rate
normal	N/A	1.45 ± 0.72%	N/A	2.36 ± 1.07%
100k-rt	91.71 ± 3.23%	20.2 ± 6.27%	95.3 ± 0.79%	1.27 ± 0.38%
100k-tf	88.48 ± 4.14%	17.8 ± 5.10%	93.9 ± 0.72%	2.06 ± 0.63%
10k-rt	92.36 ± 3.79%	14.4 ± 4.87%	94.7 ± 0.51%	3.28 ± 0.93%
10k-tf	89.91 ± 5.31%	15.7 ± 3.39%	97.1 ± 0.32%	3.57 ± 0.79%

while the DSDV is the worst. Why do the performances over different protocols vary so much? We will discuss this issue in Section 5.4.

As a matter of fact, RIPPER performs very poorly, which is not the case when it is applied in similar tasks in wired networks. As we know, RIPPER is a rule based classifier, where rules are composed by expressions in first-order feature space. This can be an strong indication that quasi-linear anomaly detection analysis used in traditional intrusion detection systems can not be used in ad hoc networks, where high mobility defeats such effort. Our experiments show that SVM has better performance, we will conduct more study to gain insights on how SVM is able to catch the dynamic characteristics.

We then used DSR/SVM for further experiments where we tested our model with tests larger than the training set, from 200,000 seconds to 1,000,000 seconds. The results are shown in Table 7. Detection and false alarm rates on individual records and intrusion sessions (after post-processing) are listed.

The results show that though the model has been trained with a much smaller trace, it has already converged satisfyingly, so that it is effective even for a much longer trace. Almost all intrusion sessions are detected, the false alarm rate is also very low, the average of them is about 0.125%.

5.4. Discussion

The experiment results demonstrate that an anomaly detection approach can work well on different wireless ad-hoc networks. That is, the normal behavior of a routing protocol can be established and used to detect anomalies.

First, it is important to point out that we use a post-processing scheme to remove some spurious error during normal use period. Errors are unavoidable in normal traces but we assume that they should not happen frequently. In contrast, "multiple" disorders are usually recorded during deliberate intrusion. By choosing a good window size, we can avoid high false positive rate and still have high detection rate. The issue of spurious error can lead to a debate in the intrusion detection research community on how to detect an intrusion that relies on single "maneuver". For example, using network connection data, anomaly detection can be very effective against multi-connection-based port scan and DDoS attacks, but not so for a single-connection-based buffer-overflow attack. However, using system call trace generated by a running program, anomaly detection models can be very effective against buffer-overflow attacks [7,16]. This shows that there are some natural limits on detection capabilities, depending on at which layer the data is collected. Thus, for the routing protocols layer, we also believe that with the cooperation of IDS on other layers, the overall anomaly detection performance can be improved.

In this experiment, we also find a few system parameters that may change the normal behavior heavily. One of them is the mobility level – if the model is classified using values from another mobility level, the alarm rate can be much higher. This can be solved by randomizing the mobility level in the experiment. However, the current ns-2 code does not yet support this feature. It nevertheless teaches us an important lesson that a good anomaly detection model should collect all possible value combinations and normal scenarios. We plan to develop schemes to cluster and classify the normal scenarios so that we can build specific anomaly detection models for each type of normal scenarios.

Table 7
DSR: Accuracy on Traces with Different Running Times

Running Time	Sequence Detection Rate	Sequence False Alarm Rate	Intrusion Session Detection Rate	Intrusion Session False Alarm Rate
200000	99.1 ± 0.11%	0.0830 ± 0.052%	100.0 ± 0%	0.187 ± 0.52%
300000	99.1 ± 0.12%	0.0684 ± 0.024%	100.0 ± 0%	0.000 ± 0.00%
400000	99.1 ± 0.08%	0.0703 ± 0.028%	100.0 ± 0%	0.168 ± 0.30%
500000	99.2 ± 0.11%	0.0663 ± 0.026%	100.0 ± 0%	0.124 ± 0.22%
600000	99.1 ± 0.09%	0.0785 ± 0.018%	100.0 ± 0%	0.338 ± 0.34%
700000	99.1 ± 0.12%	0.0819 ± 0.027%	100.0 ± 0%	0.000 ± 0.00%
800000	99.1 ± 0.13%	0.0747 ± 0.026%	100.0 ± 0%	0.126 ± 0.24%
900000	99.1 ± 0.06%	0.0640 ± 0.018%	100.0 ± 0%	0.115 ± 0.22%
1000000	99.0 ± 0.15%	0.0755 ± 0.021%	100.0 ± 0%	0.070 ± 0.19%

Having done the experiments on three ad-hoc routing protocols, we now attempt to answer this question – which type of protocol is "better" for anomaly detection. Our solution tends to prefer DSR and AODV, even in the first look its route update is not as "regular" as DSDV. After detail analysis of these protocols, we believe that anomaly detection works better on a routing protocol in which a degree of redundancy exists within its infrastructure. DSR embeds a whole source route in each packet dispatched, hence making it harder to hide the intrusion by faking a few routing information. We call this a path redundancy. Further, the DSR and AODV route update depends on traffic demand, which makes it possible to establish relationships between routing activities and traffic pattern. We call this a pattern redundancy. DSDV, in contrast, has a very weak correlation between control traffic and data traffic, even when we preserve the traffic feature. Note that DSR and AODV are both on-demand protocols. We therefore believe that those types of redundancy have helped on-demand protocols to have a better performance.

The next question, naturally, is about what information we need in a general routing protocol, and whether we can add criteria into security consideration on new protocol design. We believe that a high correlation among *changes* of three types of information is preferred: **traffic flow**, **routing activities** and **topological patterns**. The topological pattern, which is noticeably dynamic in the ad-hoc environment, should be more valuable if it is used by a protocol when making route decisions. For example, new routing protocols such as Location-Aided Routing protocol [14] that attempt to utilize topological information may be more advantageous. However, we have not found their implementation on ns-2, thus we cannot conduct an experiment at present time. We can conjecture that it should improve the accuracy of our current model.

6. Related Work

There have been a lot of studies on security prevention measures for infrastructure-based wireless networks (such as [2,23]), but there is little work on the aspect of intrusion detection. We have argued in this paper that intrusion detection is extremely important for mobile computing environment.

On the prevention side, general approaches such as key generation and management have been used in a distributed manner to insure the authenticity and integrity of routing information. Zhou and Haas [27] introduced a routing protocol independent distributed key management service. This approach uses redundancies in the network topology to provide reliable key management. The key idea is to use key sharing with a maximum threshold ratio of compromised nodes to total nodes. Binkley [1] reported experiments on authentication of MAC and IP layers. Jacobs and Corson [10] proposed an authentication architecture where the emphasis is to build a hierarchy of trust in order to authenticate IMEP messages. The difficulties in realizing all these proactive schemes are: first, cryptography

is relatively expensive on mobile hosts, where computational capability is comparatively restricted; second, since there is no central authority that can be depended upon, authentication is more difficult to implement; third, these schemes are only useful to prevent intruders from outside (external attacks) and are not useful when an internal node is compromised (internal attack).

On the detection and response side, Smith et al. [25] suggested methods to secure distance-vector routing protocols. Extra information of a predecessor in a path to a destination is added into each entry of the routing table. Using this piece of new information, a path-reversal technique (by following the predecessor link) can be used to verify the correctness of a path. Such mechanisms usually come with a high cost and are avoided in wired network because routers are usually well protected. However in a mobile ad-hoc network, because each node acts as a router and is not as secure, this kind of information that helps intrusion detection is very valuable.

7. Conclusion

We have argued that any secure network will have vulnerability that an adversary could exploit. This is especially true for mobile wireless networks. Intrusion detection can compliment intrusion prevention techniques (such as encryption, authentication, secure MAC, secure routing, etc.) to secure the mobile computing environment. However, new techniques must be developed to make intrusion detection work better for wireless networks.

Through our continuing investigation, we have shown that an architecture for better intrusion detection in mobile computing environment should be distributed and cooperative. Anomaly detection is a critical component of the overall intrusion detection and response mechanism. Trace analysis and anomaly detection should be done locally in each node and possibly through cooperation with all nodes in the network. Further, intrusion detection should take place in all networking layers in an integrated cross-layer manner.

We focused our research on ad-hoc routing protocols because they are the foundation of a mobile ad-hoc network. We proposed to use anomaly detection models constructed using information available from the routing protocols for intrusion detection purposes. We applied RIPPER and SVM Light to compute classifiers as anomaly detectors. We have implemented this model and have conducted simulations to evaluate its performance. Finally, we showed that these detectors in general have good detection performance.

There are some interesting findings. In particular, we noted some disparity in security performance among different types of routing protocols. We claimed that protocols with strong correlation among changes of different types of information, i.e, location, traffic, and routing message, tend to have better detection performance. More specifically, on-demand protocols usually work better than table-driven protocols because the behavior of on-demand protocols reflects the correlation between traffic pattern and routing message flows.

References

[1] James Binkley. Authenticated ad hoc routing at the link layer for mobile systems. Technical Report 96-3, Portland State University, Computer Science, 1996.

[2] A. Boukerche and Mirela Sechi Moretti Annoni Notare. Neural fraud detection in mobile phone operations. In *Proceeding of the IPDPS 2000 Workshops, Cancun, Mexico*, pages 636–644, May 1-5 2000.

[3] J. Broch, D. Johnson, and D. Maltz. The dynamic source routing protocol for mobile adhocnetworks. Internet draftdraft-ietf-manet-dsr-01.txt, December 1998.

[4] William W. Cohen. Fast effective rule induction. In *Proc. 12th International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[5] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[6] K. Fall and e Varadhan. *The ns Manual (formerly ns Notes and Documentation)*, 2000.

[7] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for Unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 120–128, Los Alamitos, CA, 1996. IEEE Computer Society Press.

[8] R. Heady, G. Luger, A. Maccabe, and M. Servilla. The architecture of a network level intrusion detection system. Technical report, Computer Science Department, University of New Mexico, August 1990.

[9] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, March 1995.

[10] S. Jacobs and M. S. Corson. MANET authentication architecture. Internet draftdraft-jacobs-imep-auth-arch-01.txt, expired 2000, February 1999.

[11] T. Joachims. *Making large-scale SVM learning practical*, chapter 11. MIT-Press, 1999.

[12] D. Johnson. Routing in ad hoc networks of mobile hosts. In

Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, U.S., 1994.

[13] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Tomasz Imielinski and Hank Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[14] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *ACM/Baltzer Wireless Networks (WINET) journal*, Vol 6-4 - Extended version of the Mobicom'98 paper., 2000.

[15] S. Kumar and E. H. Spafford. A software architecture to support misuse intrusion detection. In *Proceedings of the 18th National Information Security Conference*, pages 194–204, 1995.

[16] W. Lee and S. J. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, January 1998.

[17] W. Lee, S. J. Stolfo, and K. W. Mok. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, May 1999.

[18] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neumann, H. Javitz, A. Valdes, and T. Garvey. A real-time intrusion detection expert system (IDES) - final technical report. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, February 1992.

[19] D. A. Maltz, J. Broch, J. Jetcheva, and D. B. Johnson. The effects of on-demand behavior in routing protocols for multihop wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, August 1999.

[20] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[21] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[22] C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[23] S. Jacobs S. Glass, T. Hiller and C. Perkins. Mobile IP authentication, authorization, and accounting requirements. Request for Comments 2977, Internet Engineering Task Force, October 2000.

[24] M. Satyanarayanan, J. J. Kistler, L. B. Mummert, M. R. Ebling, P. Kumar, and Q. Lu. Experiences with disconnected operation in a mobile environment. In *Proceedings of USENIX Symposium on Mobile and Location Independent Computing*, pages 11–28, Cambridge, Massachusetts, August 1993.

[25] B. R. Smith, S. Murthy, and J.J. Garcia-Luna-Aceves. Securing distance-vector routing protocols. In *Proceedings of Internet Society Symposium on Network and Distributed System Security*, pages 85–92, San Diego, California, February 1997.

[26] Lakshmi Venkatraman. Secured routing protocol for ad-hoc networks. Master's thesis, University of Cincinnati, OH, March 2000.

[27] L. Zhou and Z. J. Haas. Securing ah hoc networks. *IEEE Network*, 13(6):24–30, Nov/Dec 1999.