Abstract


Mobile Ad-hoc networking (MANET) is an important emerging technology. As recent several security incidents remind us, no open computer system is immune from intrusions. The routing protocols in ad-hoc networks are key components yet vulnerable and present special challenges to intrusion detection.

In this thesis, we propose an anomaly detection scheme for existing ad-hoc routing protocols. Our approach relies on information from local routing data and other reliable local sources. Our approach models the temporal/sequential characteristics of observations and uses entropy analysis for feature selection. Classification algorithms are used to compute anomaly detection models. We present case studies on DSR and DSDV protocols using the ns-2 simulator. The overall results thus far are very encouraging. We discuss how the available information from a routing protocol influences anomaly detection performance and attempt to provide guidelines on what features we need for anomaly detection.

Finally, we also discuss several challenging issues and propose our future work.

Keywords: Routing Security, Intrusion, Anomaly Detection, MANET, Classification
Anomaly Detection for Wireless Ad-Hoc Routing Protocols

by

Yi-an Huang

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

COMPUTER SCIENCE

Raleigh

2001

APPROVED BY:

Chair of Advisory Committee
Personal Biography

Yi-an Huang was born in Shanghai, China, on February 1, 1978. He obtained his Bachelor Degree in Computer Science in May 1999 at Fudan University. During his study in Fudan University, he took part in the research of *Audit Message Filter and Analysis System*, which is part of a campus security project. He also took internship in Microsoft Great Asian Support Center for half a year in 1998. He is now studying in the Master Program of Computer Science Department, North Carolina State University.
# Contents

List of Tables vi

List of Figures vii

1 Introduction 1

1.1 Wireless Ad-Hoc Environment .................................. 1

1.2 Vulnerabilities of Wireless Ad-Hoc Networks .................. 2

1.3 Intrusion Detection and Challenges of Ad-Hoc Networks ...... 5

2 Anomaly Detection 8

2.1 Background: Information Theory .............................. 8

2.1.1 Entropy .............................................. 8

2.1.2 Conditional Entropy .................................. 10

2.1.3 Relative Conditional Entropy ............................. 11

2.1.4 Information Gain and Classification ....................... 11

2.1.5 Information Cost ...................................... 14

2.2 Anomaly Detection Framework .................................. 14

2.3 Details of model design and implementation ................... 16
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1</td>
<td>Classifier</td>
<td>16</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Feature Selection and Entropy</td>
<td>18</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Post-Processing</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>Ad-Hoc Routing Protocols Overview</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>Source-Initiated On-Demand Routing</td>
<td>26</td>
</tr>
<tr>
<td>3.1.1</td>
<td>DSR</td>
<td>26</td>
</tr>
<tr>
<td>3.2</td>
<td>Table-driven Routing Protocols</td>
<td>27</td>
</tr>
<tr>
<td>3.2.1</td>
<td>DSDV</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>Anomaly Detection Study on Ad-Hoc Routing Protocols</td>
<td>30</td>
</tr>
<tr>
<td>4.1</td>
<td>Challenges and Attacks</td>
<td>30</td>
</tr>
<tr>
<td>4.2</td>
<td>Architecture</td>
<td>32</td>
</tr>
<tr>
<td>4.3</td>
<td>Case Studies</td>
<td>33</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Simulation Environment</td>
<td>33</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Feature Selection</td>
<td>36</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Scenario</td>
<td>36</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Models</td>
<td>37</td>
</tr>
<tr>
<td>4.3.5</td>
<td>Results</td>
<td>39</td>
</tr>
<tr>
<td>4.4</td>
<td>Discussion</td>
<td>47</td>
</tr>
<tr>
<td>5</td>
<td>Related Work</td>
<td>50</td>
</tr>
<tr>
<td>5.1</td>
<td>Intrusion Detection System</td>
<td>50</td>
</tr>
<tr>
<td>5.2</td>
<td>Information Theory</td>
<td>51</td>
</tr>
</tbody>
</table>
5.3 Ad-Hoc Routing security ........................................ 52

6 Conclusion and Future Work ........................................ 54

6.1 Conclusion .......................................................... 54

6.2 Future work .......................................................... 55

Bibliography .......................................................... 58
List of Tables

2.1 General structure of rule induction algorithms ............................................. 16

3.1 Constants used in the DSR simulation ...................................................... 28

3.2 Constants used in the DSDV simulation .................................................... 29

4.1 Parameters for Anomaly Detection Simulation ........................................... 35

4.2 Local Features on Ad-Hoc Protocols ....................................................... 37

4.3 Anomaly Detection on DSR with several experiments ................................. 45

4.4 Anomaly Detection on DSDV with several experiments ............................... 45

4.5 DSR: Accuracy on Traces with Different Running Times ............................ 47
List of Figures

1.1  Ad-Hoc Network Dynamic Topology ........................................ 4

4.1  Accuracy vs. Sequence Length ............................................... 40
Chapter 1

Introduction

A general introduction on ad-hoc environment and its security challenges is provided, together with comparison among different security approaches and why we pay particular attention on routing protocol study.

1.1 Wireless Ad-Hoc Environment

A wireless ad-hoc network consists of a collection of “peer” mobile nodes that are capable of communicating with each other without help from a fixed infrastructure. Interconnections between nodes are capable of changing on a continual and arbitrary basis. Nodes within each other’s radio range communicate directly via wireless links, while those that are far apart use other nodes as relays. Various protocols are designed to manage the new distributed communication platform on different layers.

Nodes usually share the same physical media; they transmit and acquire signals at the same frequency band, and follow the same hopping sequence or spreading code. The
data-link-layer functions manage the wireless link resources and coordinate medium access among neighboring nodes. The medium access control (MAC) protocol is essential to a wireless ad-hoc network because it allows mobile nodes to share a common broadcast channel. The network-layer functions maintain the multi-hop communication paths across the network; all nodes must function as routers that discover and maintain routes to other nodes in the network. Mobility and volatility are hidden from the applications so that any node can communicate with any other node as if everyone were in a fixed wired network. Applications of ad-hoc networks range from military tactical operations to civil rapid deployment such as emergency search-and-rescue missions, data collection/sensor networks, and instantaneous classroom/meeting room applications.

1.2 Vulnerabilities of Wireless Ad-Hoc Networks

As several recent high profile security incidents in the Internet community have shown us, no open computer network is immune from intrusions. Wireless ad-hoc network is particularly more vulnerable, due to its fundamental characteristics, such as open medium, dynamic topology, distributed cooperation, and constrained capability. Ironically, most of the features contributes to the fact that ad-hoc networks become useful and popular.

Open Medium  First of all, the use of wireless links renders a wireless ad-hoc network 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 ad-hoc
network can come from all directions and target at any nodes. Since 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. Damages caused 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 [ZL00].

For example, the current MAC protocols for wireless ad-hoc networks are all vulnerable. Although there are many 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 or 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.

**Dynamic Topology** Nodes are able to move towards arbitrary directions, at theoretically arbitrary speed (though a physical limit exists from system to system). Therefore, an ad-hoc network topology may change in an unpredictable manner. Some fundamental protocols in ad-hoc networks, like routing protocols, update information in reflect of topological
change, a potential threat can arise as intruders can attempt to update routing information maliciously by pretending to be some legitimate topological change. Due to the high volume of node movement, the legitimacy is hard to detect.

Figure 1.1 demonstrates an example on a topological change. The circles represent reachable regions within nodes’ radio range. Solid lines are wireless links. The scenario changes when node C is moving towards north-east. The original route from A to D is A through C through D. After the movement, the route is updated to A through B through C through D.

**Distributed Cooperation** Decision-making in ad-hoc networks is usually decentralized and many ad-hoc network algorithms rely on the cooperative participation of all nodes. The lack of centralized authority means that the adversaries can exploit this vulnerability for new types of attacks designed to break the cooperative algorithms.

Ad-hoc routing presents such vulnerability. Most ad-hoc routing protocols are inherently cooperative[RT99]. Unlike with a wired network, where extra protection can be
placed on routers and gateways, an adversary who hijacks an ad-hoc node could paralyze the entire wireless network by disseminating false routing information. Such false routing information could also result in messages from all nodes being fed to the compromised node, which is more hidden yet usually more dangerous.

**Constrained Capability**  Due to technology limits, most mobile hosts have limited network capacity, which, even worse, are effected by many physical factors, terrain, foliage, weather, building and structures. Disconnected operations [SKM+93] are very common in wireless network applications. In a multi-hop routing protocols, each hosts are responsible to serve as a router for many different hosts, which can easily be overloaded. The security issues here are similar to dynamic topology, where anomalies are hard to distinguish from normalcy.

For some nodes which rely on battery for power supply, energy conservation is also an important issue, which makes some computation-intensive security measures infeasible.

### 1.3 Intrusion Detection and Challenges of Ad-Hoc Networks

From the discussion of ad-hoc network specific challenges, we draw a conclusion that, although techniques designed for protecting wired networks are developed for years, the vast difference between the two networks makes it very difficult to apply them to an ad-hoc wireless network directly. 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 carry the private keys. Integrity validation using redundant information (from different nodes), such as those being used in secure routing \cite{SMGLA97, ZH99}, also relies on the trustworthiness of other nodes, which could likewise be a weak link for sophisticated attacks.

Therefore, we need a second wall of defense that provides the capability to detect intrusions and to alarm users. Intrusion detection techniques can provide such second defense for secure wireless ad-hoc networks.

Intrusion detection techniques can be categorized into misuse detection and anomaly detection. Misuse detection systems, e.g., IDIOT [KS95] and STAT [IKP95], use patterns of known attacks or weak spots of the system to match and identify known intrusions. Anomaly detection systems, for example, IDES \cite{LTG92}, flag observed activities that deviate significantly from the established normal usage profiles as anomalies, i.e., possible (known or new) intrusions. It is very obvious that the enemies, knowing that intrusion prevention and detection systems are installed in our networks, will attempt to develop and launch new types of attacks. In anticipation of these trends, we focus our research on the more difficult and important problem of anomaly detection.

Among those vulnerabilities in a wireless ad-hoc network, the initial focus of our research is in ad-hoc routing. This is because ad-hoc routing is obviously one of the most fundamental building blocks of an ad-hoc network. 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. Through the classification methodology that is to be described in this thesis, we can identify which routing protocol, with potentially all its routing table
information used, can result in better performing detection models. It will help answer the question “what information should be included in the routing table to make intrusion detection effective.” The results of this research can be valuable feedback to the ad-hoc networking community for better and more resilient routing protocols. Furthermore, 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 wireless networks.

To summarize, a wireless ad-hoc network has inherent vulnerabilities that are not easily preventable. To build a highly secure wireless ad-hoc network, we need to deploy intrusion detection and response techniques. The goal of this research is to develop new models and mechanisms to adapt intrusion detection techniques from their original applications in fixed wired network to this new wireless ad-hoc network environment. The technical challenge is to overcome the chaotic appearance in operations of wireless ad-hoc networks and discover intrinsic signs of intrusions.

The rest of the thesis is organized as follows. We discuss issues in building anomaly detection models for wireless ad-hoc networks in Chapter 2, then provide a general overview on ad-hoc routing protocols in Chapter 3. After that, case studies on two different ad-hoc routing protocols, DSR and DSDV, are presented in Chapter 4. We next discuss our findings from these experiments in Section 4.4. We then provide related work information in Chapter 5 and Chapter 6 concludes with a summary and an outline of future work.
Chapter 2

Anomaly Detection

2.1 Background: Information Theory

In this section, we discuss several information-theoretic measures (these concepts are covered in many texts on information theory, e.g. [CT91]). We explain how these measures characterize the regularity embedded in audit data and influence the performance of anomaly detection models. We also discuss how to use temporal sequences to improve information gains by applying information theory.

2.1.1 Entropy

Entropy, or Shannon-Wiener Index [SW64], is an important concept in information theory and communication theory. It measures the uncertainty (or impurity) of a collection of data items.

Definition 1 For a dataset $X$ where each data item belongs to a class $x \in C_X$, the entropy
of $X$ relative to this $|C_X|$-wise classification is defined as

$$H(X) = \sum_{x \in C_X} P(x) \log \frac{1}{P(x)}$$

where $P(x)$ is the probability of $x$ in $X$.

The typical interpretation of entropy is the number of bits required to encode (and transmit) the classification of a data item. A higher entropy means the class distribution is more varied, while a lower entropy means it is more uniform. A special case is if all data items belong to one class, then entropy is zero, which means no bit needs to be transmitted because the receiver ‘knows’ that there is only one outcome. The entropy value is larger when the class distribution is more even, i.e., when the data is more “impure”. For example, if the data items are evenly distributed in $|C_X|$ classes, then $\log|C_X|$ bits are required to encode a classification.

For anomaly detection, we can use entropy as a measure of the regularity of audit data. Each unique record in an audit dataset represents a class. The smaller entropy represents higher regularity in the audit dataset, which contains redundancies that helps to predict future values with existing data. The simple entropy model works happily when all data represents same class and the entropy is zero. Thus a single rule is sufficient to identity an anomaly. If the audit data contains many event types, then the entropy is greater than zero and a more complex model will be needed, using some other measures. But all these measures are based on the entropy concept.
2.1.2 Conditional Entropy

**Definition 2** The conditional entropy of $X$ given $Y$ is the entropy of the probability distribution $P(x|y)$, that is,

$$H(X|Y) = \sum_{x,y \in C_X, C_Y} P(x,y) \log \frac{1}{P(x|y)}$$

where $P(x,y)$ is the joint probability of $x$ and $y$ and $P(x|y)$ is the conditional probability of $x$ given $y$.

By definition, conditional entropy is a measure to specify how many number of bits to transmit a data item if we know the value of another data item. The more correlated two data items, the lower the conditional entropy. If $Y$ is completely determined by $X$, then $H(X|Y) = 0$.

An application of conditional entropy in anomaly detection is that we can explore a temporal sequential characteristic of audit data due to the temporal nature of user, program, and network activities.

Using the definition above, let $X$ be a collection of sequences where each is denoted as $(e_1, e_2, \ldots, e_{n-1}, e_n)$, and each $e_i$ is an audit event; and let $Y$ be the collection of subsequences where each is $(e_1, e_2, \ldots, e_k)$, and $k < n$, then the conditional entropy $H(X|Y)$ tells us how much uncertainty remains for the rest of audit events in a sequence $x$ after we have seen $y$, i.e., the first $k$ events of $x$ (note that since $y$ is always a subsequence of $x$ here, we have $P(x,y) = P(x)$).
2.1.3 Relative Conditional Entropy

**Definition 3** The relative entropy between two probability distributions \( p(x) \) and \( q(x) \) that are defined over the same \( x \in C_x \) is

\[
\text{relEntropy}(p|q) = \sum_{x \in C_x} p(x) \log \frac{p(x)}{q(x)}
\]

For anomaly detection, we often build a model using a training dataset and apply the model to the test dataset. Relative entropy measures the distance of the regularities between two datasets. These two datasets must have the same (or very similar) regularity for the anomaly detection model to attain high performance. Thus relative entropy is a useful measure to determine how general the model behaves.

When we use conditional entropy to measure the regularity of sequential dependencies, we can also use relative conditional entropy to measure the distance between two audit datasets.

**Definition 4** The relative conditional entropy between two probability distributions \( p(x|y) \) and \( q(x|y) \) that are defined over the same \( x \in C_x \) and \( y \in C_y \) is

\[
\text{relCondEntropy}(p|q) = \sum_{x,y \in C_x \times C_y} p(x, y) \log \frac{p(x|y)}{q(x|y)}
\]

Again, for anomaly detection, the smaller the relative conditional entropy, the better.

2.1.4 Information Gain and Classification

Intrusion detection can be cast as a classification problem: we wish to classify an audit event as belonging to the normal class, the abnormal class (in the case of anomaly detection) or a particular class of intrusion (in the case of misuse detection). Here assuming that
classifiers are used as anomaly detection models, we discuss how regularity of audit data influences the performance of anomaly detection models.

Given a training dataset where the records are defined by a set of features and each record belongs to a class, the goal of constructing a classifier is that after (selectively) applying a sequence of feature value tests, the dataset can be partitioned into “pure” subsets, i.e., each in a target class, so that the sequence of feature value tests can be used as the conditions in the classifier to determine the class of a new record (when its class is not yet known). In this process, all records in each final subset are considered as belonging to the majority class of the subset because for each record there can be only one classification outcome. It is obvious that the purer the final subsets, the more accurate the classifier. Therefore when constructing a classifier, a classification algorithm needs to search for features with high information gain [Mit97], which is the reduction of entropy when the dataset is partitioned according to the feature values.

**Definition 5** The information gain of attribute (i.e., feature) $A$ on dataset $X$ is

$$Gain(X, A) = H(X) - \sum_{v \in \text{Values}(A)} \frac{|X_v|}{|X|} H(X_v)$$

where $\text{Values}(A)$ is the set of possible values of $A$ and $X_v$ is the subset of $X$ where $A$ has value $v$.

If all features have low information gain, then the classifier will have poor performance because after the original dataset is partitioned, the subsets still have large entropy, i.e., they are still “impure”. Therefore, for anomaly detection (and intrusion detection in general), the higher the information gain of the features, the better.
When the regularity of sequential dependencies is used directly in the anomaly detection model, there is a direct connection between conditional entropy and information gain. For example, suppose we have a classifier that uses the first $n-1$ audit events to classify (i.e., predict) what normally the $n$th event should be. In this case, the first $n-1$ events are used as the features and the $n$th event as the class. Since all $n-1$ features can be used in the classifier, for simplicity in our discussion, we can “collapse” all of them into a single feature $A$. Then for $Gain(X, A)$, the second term in the formula is essentially the conditional entropy of the length $n$ sequence given the length $n-1$ subsequence (prefix). Therefore, when we model a sequence, the smaller the conditional entropy, the higher the information gain, and hence the better detection performance of the model.

When we model a complex subject, e.g., network traffics, we often need not only information pertaining to the current event but also sequential (or temporal) information on previous events. Conditional entropy can be used in the feature construction process to suggest what features can be added so that the feature set contains information on both current and previous events. For example, suppose that in a timestamped audit dataset, each record initially is defined as $<t, f_1, f_2, \ldots, f_n, c>$ where $t$ is the timestamp, each $f_i$ is a feature, e.g., the duration of the current connection, number of bytes sent, etc., and $c$ is the class label. Suppose that we use the service of a connection as its class, that is, we want to model how each service normally behaves. If there is strong regularity, i.e., low conditional entropy, on the sequence of services (or the combination of service and other features), we can add features that express this regularity. One way is to add features that act as place holders for the services of previous connections (that fall within a time window), i.e., each connection record includes the names of some previous services, $s_{i-1}, s_{i-2}, \ldots$.
etc. Alternatively, to reduce the total number of features (and hence the complexities of the model), we can use some statistical feature(s), e.g., within the past $n$ seconds, the percentage of the services that are the same as the current one, to approximate the regularity information. In [LS98], we showed that these temporal and statistical features usually have high information gain, and hence a better model can be built when these features are added to the audit data.

### 2.1.5 Information Cost

Intuitively, the more information we have, the better the detection performance. However, there is always a cost for any gain. For intrusion detection, one important goal is to detect intrusions as early as possible so that appropriate responses can be carried out effectively. We can define information cost as the average time for processing an audit record and checking against the detection model. When we include more information, we not only increase the data processing time, we often increase the model complexities as well. Therefore, there needs to be a trade-off between detection performance and cost. For example, the simple measure $\text{Accuracy}/\text{Cost}$ may be used to determine the “optimal” amount of information to be used in the model.

### 2.2 Anomaly Detection Framework

The basic premise for anomaly detection is that there is intrinsic and observable characteristic (or regularity) of normal behavior that is distinct from that of abnormal behavior. The task of developing an anomaly detector is to first study the characteristic of normal
audit data and then build a model that best utilizes the characteristic. We use information-
theoretic measures \[CT91\], namely, entropy and conditional entropy, to describe the char-
acteristics of normal information flows and to use classification algorithms to build anomaly
detection models \[LX01\].

We can use classifiers as anomaly detection models. For example, we use a classifier,
trained using normal data, to predict what is \textit{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. Given a record described by a set of features (or attributes), a classifier
determines the class label of the record. When constructing a classifier, the algorithm
searches for features with high \textit{information gain} (or reduction in entropy) \[Mit97\]. That is,
a classifier needs feature value tests to partition the original dataset (with mixed classes and
hence high entropy) into pure subsets (each ideally with one class and hence low entropy).

Using this framework, we employ following procedure for anomaly detection,

- select (or partition) audit data so that the normal dataset has entropy (or conditional
  entropy) as low as possible

- perform appropriate data transformation according to the entropy measures (e.g.,
  constructing new features with high information gain);

- apply classification algorithm on training data to learn a classifier;

- apply the classifier on test data; and

- post-process alarms to produce intrusion reports;

- verify relative conditional entropy on sequences from both traces.
2.3 Details of model design and implementation

2.3.1 Classifier

We attempt to make use of two types of classifiers, to learn the efficiency of different types of classifiers on our problem.

Our selection set includes a rule based classifier, RIPPER [Coh95], a fast effective rule induction tool widely known, and a Support Vector Machine classifier, SVM Light [Joa99].

Table 2.1: General structure of rule induction algorithms

Input: ES is the training set.

Procedure Rule Induction(ES)

Let RS = ∅.

For each class C

Let ⊕ = {E ∈ ES | Class(E) = C}.
Let ⊖ = {E ∈ ES | Class(E) ≠ C}.

Repeat

Let R = Find_Best_Rule(C, ⊕, ⊖).
Let ⊕ = ⊕ - {E ∈ ⊕ | R covers E}.
Let RS = RS ∪ {R}.

Until ⊕ = ∅ or R = Nil.

Return RS.

Function Find_Best_Rule(C, ⊕, ⊖)

Let Body = True.

Let R be the rule; Body ⇒ C.

Repeat

For each possible antecedent A

Let BA = Body ∧ A.
Let na = #{E ∈ ⊕ | E satisfies BA}.
Let nb = #{E ∈ ⊖ | E satisfies BA}.

Let Body = BA that maximizes some heuristic H(na, nb).

Until no antecedent causes a significant improvement H(na, nb).

Return R, or Nil if Body = True.

Rule induction algorithms typically employ a set covering or “separate and conquer” approach to induction. This strategy derives its name from the fact that it forms a class definition by constructing a rule that covers many positives examples, and few or no negative
ones, then “separate out” the newly covered examples and start again on the remainder. It is summarized in Table 2.1. A rule is composed of a consequent and a body part. The consequent part is the predicted class. The body is a conjunction of antecedents, each antecedent being a condition involving a single attribute. For symbolic attributes, this condition is a simple equality test; in some systems, negation and disjunction of values are possible. For numeric attributes, the condition is typically inclusion in a one-sided interval. A rule is said to cover an example, and conversely the example is said to satisfy it, if all the conditions in the rule are true for the example. Given a class, its members in the training set are called positive examples, and the remainder are negative examples. [Dom96]

RIPPER algorithm is based on one of the recent rule learning algorithms, Incremental Reduced Error Pruning (IREP) which can efficiently handle large noisy datasets, and then add certain iterations of optimization. This algorithm is efficient in both time and space requirement. It is comparably much faster than other well-known rule based algorithms and never uses data structures larger than the data set. It is also scalable in that it is reported to scale nearly linearly with the number of examples in a noisy dataset. [Coh95]

Then we learn that a new category of machine learning method, the Support Vector Machine [Sch97] method, can also be used for classification job. The SVM has attracted lot of an increasing interest in late years, which uses the Mercer kernels for efficiently performing computations in high-dimensional spaces. SVM pre-processes the data to represent patterns in much higher dimension, the heuristic of which is that with sufficiently high dimension, data, which is usually non-linearly related to input space, can be separated by a hyper plane, thus achieving the goal of classification. The SVMs approximately perform Structural Risk Minimization principle to minimize the average error committed on
independent examples randomly drawn from the same distribution.

In reality, we find out that 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. The disadvantage of SVM, is that to achieve good performance, it sometimes need to pick up a proper set of kernel and optimization parameters carefully. Also, some useful features (such as multi-valued class classification) are still under development.

The input attributes are formed using information theoretic measures described before. We illustrate how it works in Section 2.3.2.

2.3.2 Feature Selection and Entropy

Feature selection is highly challenging, and it plays a critical part in differentiating behaviors of a normal scenario and an abnormal case. Because the major attacks on ad-hoc routing protocols are the false routing information generated by a comprised node and malicious traffic distortion, we can define trace data to describe the route updates and flow statistics. Besides, within a highly dynamic environment, we also consider the mobility effects on location adjustment.

Moreover, since a decision based classifier is used, we have to carefully choose the features with high information gain, so that some features may be pruned, in order to provide a classifier potentially more fit for future data set, also, increases human understandability of rules being produced.

Since a major difficulty in our ad-hoc routing research is the high mobility of whole
system, i.e., even a static host (without message exchange and any movement at all) may still experience route change, because of the movement on other hosts. Limited by computation capability, it is still not feasible yet for us to calculate the whole network’s topology and store it locally, hence the situation leads to high false alarm to detect a real intrusion instead of a random anomaly. The real problem here is the information gain from single observation may not be sufficient to achieve an accurate model.

In order to increase information gain, we can apply conditional entropy measure to enhance our feature model. The method is inspired from system call sequence study [LSC97].

Our method is therefore to consider a temporal sequence of events, where each event records a snapshot of local information. Hopefully, when length of the sequence is long enough, the last event can be induced using multiple events from past $n - 1$ timestamps. i.e., given a recorded sequence,

$$S_n = E_{-n}, E_{-n+1}, ..., E_{-1}, E_0$$

where $E_0$ is the last event, while $S_{n-1} = E_{-n}$ to $E_{-1}$ are $n - 1$ events precedes it. If the conditional entropy given by $H(S_n | S_{n-1})$ is small enough, the normalcy of last event can be set up by remembering last $n - 1$ events. Hence, we can have the sequence, instead of single event, to represent current observation.

The selection of proper sequence length is a trade-off between high accuracy and low information cost. Because we need to predict the $n$th event, using conditional entropy $H(X | Y)$, where $X$ is the sequence of $n$ events, and $Y$ is the subsequence of $n - 1$ events, excluding $E_0$, as specified in [LX01], a longer sequence length always gives smaller conditional entropy, however, information cost (computation cost on processing audit data and
checking against the detection model, given in Section 2.1.5) also increases linearly. There is a need to call for such a trade-off. For simplicity, in this thesis, we use a simple threshold scheme. The shortest sequence length with conditional entropy no larger than a given threshold is used. From the real results given in Section 4.3.4, trade-off is well handled with this scheme.

Two classification scheme can then be used to train a model.

2.3.2.1 Bi-class scheme

Firstly, if both normal data and abnormal data with tag are available, we use the conventional scheme with a class label “normal” or “abnormal”, which originally was related with current observation, and which is now related with the whole sequence. Since we explore ‘more’ high gained features from past observations, we are expecting to see finer results from the scheme.

For instance, assume we provide two features related with each event. Assume feature 1 is traffic volume, feature 2 is routing update level (0 means no routing update in the time between previous event and current event). The original input data set now looks like this, (in RIPPER format, so last entries are class labels)

```
100, 2, normal.
200, 4, normal.
400, 2, abnormal.
400, 1, abnormal.
```

And after we sequence the original data with a length of two,

```
x, y, 100, 2, normal.
100, 2, 200, 4, normal.
200, 4, 400, 2, abnormal.
400, 2, 400, 1, abnormal.
```
x, y are features related with the event before our first example event.

2.3.2.2 Uni-class scheme

We could also do classification even if only normal data are present. In this scheme, we choose current observation $E_0$ as decision class, and the subsequence $S_{n-1}$, not $S_n$, is used as attributes instead.

Our example shown uses a trace of pure normal data.

100, 2, normal.
200, 4, normal.
400, 3, normal.
400, 5, normal.

Again, we use a sequence of length two and the routing update count of current event is used as class label.

x, y, 100, 2.
100, 2, 200, 4.
200, 4, 400, 3.
400, 3, 400, 5.

Here, only three features are counted as attributes. Another is regarded as class label. The number of class labels, in this example, is decided by possible routing updates values. Assuming a maximum routing update level is defined as 5, we have six possible class labels: 0, 1, 2, 3, 4, 5.

2.3.2.3 Decision Making

The decision on which models is used depends on several factors.

Multiple Class Should multiple class classifier be used?
In pure normal data scheme, we have to deal with multiple value classes. Some classifiers are designed to work better on binary classes. In our two classifiers, SVM_Light currently only works on binary classes. Thus it simply can not be used for this scheme yet. Even in those classifiers multiple valued class can be used, the performance often drops heavily.

**Train Set Size** How big the normal train set should have?

Since we depends on the integrity of normal data to cover all normal data patterns, the self-training methods tend to use a much larger data set. Though an upper bound can be estimated, a more feasible way to find out if the data set is enough is to do validation on some test set, if, e.g., false alarm rate is high, then it is possibly necessary to increase sampling time. Here, a trial-and-error approach may still necessary.

**Abnormal Data** Is abnormal data always available?

Our answer is no. Sometimes only normal data can be obtained from routine audit log, and sometimes intrusion data are rather incomplete in contrast to normal data. It is improper to build a classifier using these intrusion data far less than normal data, and the result is that over-fitting on intrusion signals turns out to be unacceptably huge. Thus, the restriction on using intrusion data in real audit log is rather strict. However, in our model, we only restrict ourselves to several types of known intrusions, and use them only on both training data and test data. Hence, we can accumulate ‘enough’ intrusion data for building up the Bi-Class model.

Overall, we use Bi-Class model for our case study, and both classifiers can be used in this case.
After we ‘cook’ data by adding features of past n events, we have classifiers working to build a proper model. We then use the model to do predictions on a trace log, where normalcy labels are no longer given. The prediction stream is then further processed to achieve higher performance, by a procedure called post-processing.

2.3.3 Post-Processing

A voting scheme with sliding window is used to post-process predictions derived from classifier, in order to clear spurious prediction errors. (see discussion in Section 4.4)

- Choose a parameter $l$ and let window size be $2l + 1$

- For a region scanned by the window, if more abnormal predictions are made than normal predictions, i.e., number of ”abnormal” > $l$, then the region is marked ”abnormal”

- Count every observation within an abnormal window as “abnormal”

- Shift the sliding window with one window size, i.e., $2l + 1$, and repeat the procedure until the whole trace is processed

- Count all continual abnormal regions as one intrusion session

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. However, should we allow an extreme large window, intrusions with short duration time may happily not be reported at all.
By choosing a good window size, we can avoid high false positive rate and still have high detection rate. The parameter \( l \) should be selected to maximize the ratio detection rate to false positive rate. In our study, we found results from \( l = 3 \) with corresponding window size 7, shows up to be the best option.

The procedure can be demonstrated using following example. Suppose following results are a piece from prediction streams from one classifier,

```
abnormal, abnormal, normal, abnormal, normal,
abnormal, abnormal, abnormal, abnormal, abnormal,
abnormal, normal(1), abnormal, normal, normal,
normal, normal, abnormal, normal, normal,
normal.
```

The real intrusion session ends by the event marked with (1).

Assume a sliding window starts from the beginning of our given piece, there are 5 of 7 in the first window, thus we judge that the whole window is composed of abnormal data. Similarly, we judge the following two windows are composed of abnormal and normal data respectively. Thus, the post-processed predictions look like this,

```
abnormal, abnormal, ABNORMAL, abnormal, ABNORMAL,
abnormal, abnormal, abnormal, abnormal, abnormal,
abnormal, ABNORMAL(1), abnormal(2), ABNORMAL(3), normal,
normal, normal, NORMAL, normal, normal,
normal.
```

And our reported intrusion session ends by the event marked with (3). Capitalized predictions are those being changed in this procedure. The reported boundary between intrusion session and normal data are very close to the actual one.
Chapter 3

Ad-Hoc Routing Protocols Overview

Since the advent of DARPA packet radio networks in the early 1970s, numerous protocols have been developed for ad-hoc mobile networks. Such protocols must deal with those typical limitations of these networks, which include high power consumption, low bandwidth, and high error rates. Routing in wireless ad-hoc network is a key element for realistic and efficient use by network applications on upper layers.

These routing protocols may generally be categorized as:

1. table-driven, and

2. source-initiated on-demand driven.

Despite being designed for the same type of underlying network, the characteristics of each of these protocols are quite distinct. The following sections describe the protocols and categorize them according to their characteristics.
3.1 Source-Initiated On-Demand Routing

Source-initiated on-demand routing creates routes only when desired by the source node. When a node requires a route to a destination, it initiates a route discovery process within the network. This process is completed once a route is found or all possible route permutations have been examined. Once a route has been established, it is maintained by some form of route maintenance procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired.

3.1.1 DSR

DSR [Joh94, JM96, BJM98] uses source routing rather than hop-by-hop routing, with each packet to be routed carrying in its header the complete, ordered list of nodes through which the packet must pass. The key advantage of source routing is that intermediate nodes do not need to maintain up-to-date routing information in order to route the packets they forward, since the packets themselves already contain all the routing decisions. This fact, coupled with the on-demand nature of the protocol, eliminates the need for the periodic route advertisement and neighbor detection packets present in other protocols.

The DSR protocol consists of two mechanisms: Route Discovery and Route Maintenance. Route Discovery is the mechanism by which a node $S$ wishes to send a packet to a destination $D$ obtains a source route to $D$. To perform a Route Discovery, the source node $S$ broadcasts a ROUTE REQUEST packet that is flooded through the network in a controlled manner and is answered by a ROUTE REPLY packet from either the destination node or another node that knows a route to the destination. To reduce the cost of Route
Discovery, each node maintains a cache of source routes it has learned or overheard, which it aggressively uses to limit the frequency and propagation of ROUTE REQUESTs.

Route Maintenance is the mechanism by which a packet’s sender $S$ detects if the network topology has changed such that it can no longer use its route to the destination $D$ because two nodes listed in the route have moved out of range of each other. When Route Maintenance indicates a source route is broken, $S$ is notified with a ROUTE ERROR packet. The sender $S$ can then attempt to use any other route to $D$ already in its cache or can invoke Route Discovery again to find a new route.

For better performance, if destination $D$ would receive Route Request packet, which means a feasible path really exists, the source node will finally receive response, it then caches the path for future use, and send the packet with the entire source route specified in its header. Intermediate nodes use the source route to determine if it should accept the packet and to whom the packet should be forwarded.

The whole process is cooperative. A malicious node $M$ can break in any of these links to fail down the protocol, for example, $M$ reports false Route Reply message even it does not have a path to $D$, $M$ can stop forwarding Route Reply from other nodes or modify the packet for its own good.

The constants we used in DSR simulation are listed in Table 3.1.

### 3.2 Table-driven Routing Protocols

A different approach from source-initiated on-demand routing is the table-driven routing protocols. This type of routing attempts to maintain consistent, up-to-date routing informa-
Table 3.1: Constants used in the DSR simulation

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Time between retransmitted ROUTE REQUESTs (exponentially backed off)</td>
<td>500 ms</td>
</tr>
<tr>
<td>Size of source route header carrying n addresses</td>
<td>4n + 4 bytes</td>
</tr>
<tr>
<td>Timeout for non-propagating search</td>
<td>30 ms</td>
</tr>
<tr>
<td>Time to hold packets awaiting routes</td>
<td>30</td>
</tr>
<tr>
<td>Max rate for sending gratuitous REPL Ys for a route</td>
<td>1/s</td>
</tr>
</tbody>
</table>

tion from each node to every other node in the network. These protocols require each node to maintain one or more tables to store routing information, and they respond to changes in network topology by propagating updates throughout the network in order to maintain a consistent network view. The areas where they differ are the number of necessary routing-related tables and the methods by which changes in network structure are broadcast. The following sections discuss some of the existing table-driven ad-hoc routing protocols.

3.2.1 DSDV

DSDV [PB94] is a hop-by-hop distance vector routing protocol requiring each node to periodically broadcast routing updates. The key advantage of DSDV over traditional distance vector protocols is that it guarantees loop-freedom.

It uses a table-driven approach by maintaining a table to all destinations currently seen, listing the “next hop” for each reachable destination. DSDV tags each route with a sequence number and considers a route R more favorable than $R^0$ if R has a greater sequence number, or if the two routes have equal sequence numbers but R has a lower metric. Each node in the network advertises a monotonically increasing even sequence number for itself. When a node B decides that its route to a destination D has broken, it advertises the route to D
with an infinite metric and a sequence number one greater than its sequence number for the route that has broken (making an odd sequence number). This causes any node A routing packets through B to incorporate the infinite-metric route into its routing table until node A hears a route to D with a higher sequence number. The use of sequence number guarantees loop-free.

DSDV uses two kinds of update schemes, namely, full update and partial update. The adoption of latter is in expectation to alleviate traffic load on periodic routing broadcast. Implementation on ns-2 uses a small fluctuation on time for packet updates, to avoid synchronization on updates from different hosts.

The constants we used in DSDV simulation are listed in Table 3.2. The first two of them are modified from default setting (which is 15s and 3, respectively), to provide a more promptly data collection rate.

<table>
<thead>
<tr>
<th>Table 3.2: Constants used in the DSDV simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Periodic route update interval</td>
</tr>
<tr>
<td>Periodic updates missed before link declared broken</td>
</tr>
<tr>
<td>Initial triggered update weighted settling time</td>
</tr>
<tr>
<td>Weighted settling time weighting factor</td>
</tr>
<tr>
<td>Route advertisement aggregation time</td>
</tr>
<tr>
<td>Maximum packets buffered per node per destination</td>
</tr>
</tbody>
</table>
Chapter 4

Anomaly Detection Study on Ad-Hoc Routing Protocols

4.1 Challenges and Attacks

To show how better our anomaly detection model works, we would like to work on one of the challenging field in security field, the wireless ad-hoc networks. In particular, we hope to concentrate our work on securing routing protocols, which are still immature and under rapid development. Because of high dynamics and other limits shown before, the design of ad-hoc routing protocols is more complicated and usually a nice piece of trade-off among multiple factors, which include improving routing optimum, minimizing traffic volume and restricting power use. Though a lot of new protocols have been proposed and implemented, we understand that security issues are rarely concerned, or even so, hardly practical. The anomaly detection study, then, can help to build an detection and alarm agent locally with few modification on current implementations.
We now only consider attacks on routing protocol level. An attack from different layers may adopt different or similar approaches. For example, an transport layer attack, especially TCP specific, may behave similarly with what widely happens in wired scenario.

In general, we consider routing attacks act in one of several forms. [Ven00].

1. **Route logic compromise**

   This type of attack 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: Forward a packet to an incorrect node,

   (b) False message propagation: to distribute a false route update.

2. **Traffic pattern distortion**

   This type of attack changes default behavior on the traffic patterns, which includes:

   (a) Packet dropping;

   (b) Packet Generation with faked source address;

   (c) Corruption on packet contents;

   (d) Denial of Service

   In another words, they are actually different attempts in order to tamper on control (in first case) and data channels (in latter case). Notice that it is not uncommon to combine two kinds of attacks together, as every routing agent should perform two functionalities, choosing an optimal route for packet forwarding and actually forward packets, in an efficient
way. Things may become even worse, as the routing control messages may well consume
a huge portion of traffic volume in ad-hoc environment, either by careless protocol design,
or by external attacks. Hence the boundary between two types of attack is not always clear.
If no good detection methods can be relied on, such an intrusion can be launched easily.

For simulation purpose, we implemented following attacks:

1. Falsifying route paths/route entry in its own cache

2. Random packet dropping

The first attack is routing specific, which does (random) corruption on internal routing table
entries. This is an abstraction on all protocol attacks as they finally resort to change the
routing logic for their own favor. Another is traffic pattern specific, which randomly drops
packets. For simplicity, each intrusion session is limited to one type of them. However, one
trace of running log can contain several intrusion sessions with different types of attack.

4.2 Architecture

In the context, we suggest that two kinds of information source are reliable and therefore
suitable for our feature selections.

(1) Local Routing Information, including cache entries and traffic statistics;

(2) Position Locator, or GPS. We assume GPS will not be distorted and therefore reliable
to provide location and velocity information within whole neighborhood.

Why should we only depend on limited resources? The reason lies on several aspects.
Firstly, a single host cannot collect the global information, only communication activities
within its radio range is observable. Secondly, a single mobile host is usually resource and power restricted, which implies capability to collect and analyze huge volume of data from various sources is not always feasible. Lastly, in case when intrusion may actually already happen nearby, to exchange information with others may be eavesdropped and could be potentially misleading if message exchanges are trusted blindly. In response to those potential threats, we start up by building a local intrusion detection agent, with information collected merely from trustable sources.

4.3 Case Studies

In this section we describe the experiments on two specific ad-hoc wireless protocols, which are Dynamic Source Routing (DSR) protocol and Destination-Sequenced Distance-Vector Routing (DSDV) protocol. They are picked as they represent two different types of ad-hoc wireless routing protocols, proactive and on-demand. We then show how our anomaly detection methods can be applied to different types of protocols and demonstrate the effects of different parameters on effectiveness of classification. Finally, we make a comparison on regularity and accuracy of models built between two protocols and try to explain the difference.

4.3.1 Simulation Environment

We used the wireless networks simulation from the Network Simulator ns-2\textsuperscript{1} [FeV00].

\textsuperscript{1}release 2.1b7a, December 2000

Ns-2 is a discrete event simulator targeted at networking research. Ns-2 provides sub-
stantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks.

Ns-2 began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. Ns-2 has also included substantial contributions from other researchers, including wireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems.

It includes simulation for wireless ad-hoc network infrastructure, popular wireless ad-hoc routing protocols (DSR, DSDV, TORA, ADOV), and mobility scenario and traffic pattern generation scripts. In order to incorporate with our simulation, we first made the following changes.

**Changes on ns-2 code**  The DSDV code contains in ns-2 distribution seems to have a problem which may leads to infinite loop on handling packets. The original code, under some condition, incorrectly waits for sending queue to be cleaned while some packets are in fact fed back into the same queue. A patch has been made and submitted to ns mail list for review.

We also checked codes of routing agents, so that it may not use information from a global topological collector (the God), which is not always feasible in a real scenario. Those information are only collected locally.

We use node movement and communication pattern generation scripts from CMU. However, some parts of those scripts are rewritten to allow a more flexible parameter selection. Especially, radio range can now be specified as a parameter upon scenario creation. The communication pattern generator also conforms to a strict event count, instead of a
random value, which may provides high variability in traffic flows.

**Simulation Script—the xMobile script** In order to generate simulation under different environment settings, we provides a script with many of parameters in simulation adjustable, given by a parameter file. All those parameters are listed in Table 4.1. The output of script is a trace file with a detailed log of routing information, the interval of logging is also adjustable.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Descriptions</th>
<th>Default values</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol</td>
<td>routing protocol: dsdv or dsr</td>
<td>dsdv</td>
</tr>
<tr>
<td>nn</td>
<td>number of nodes</td>
<td>10</td>
</tr>
<tr>
<td>stop</td>
<td>stop time</td>
<td>100000</td>
</tr>
<tr>
<td>tx,ty</td>
<td>topology</td>
<td>1000,1000</td>
</tr>
<tr>
<td>events</td>
<td>number of traffic events</td>
<td>10</td>
</tr>
<tr>
<td>flow</td>
<td>type of traffic; cbr or tcp</td>
<td>cbr</td>
</tr>
<tr>
<td>rate</td>
<td>traffic rate</td>
<td>20.0</td>
</tr>
<tr>
<td>pause</td>
<td>pause time in a given location</td>
<td>10</td>
</tr>
<tr>
<td>speed</td>
<td>maximum movement speed</td>
<td>20.0</td>
</tr>
</tbody>
</table>
| intrpOn,intrpOff | coeff for off-on model used in intrusion generation, if intrpOn=0, disable intrusion model | 1000,10000
| intrpMethod    | 0 for routing corruption                          | 2                            |
|                | 1 for traffic distortion,2 for random              |                              |
| radius         | detectable radius range                           | 330                          |
| interval       | logging interval                                  | 5                            |

Default traffic parameters are selected to reflect a moderate traffic pattern, compared with total running time. Some other location parameters, such as speed, size of whole region, also chosen to present an environment as typical as possible. For instance, the maximum speed of 20.0 is sufficient to show the ‘dynamic’ node topology, but not too dynamic to announce most route caching mechanism completely futile. Thus, we are able to show
the effectiveness of different protocols under ‘usual’ situation. The choice of radius, is also restricted by physician medium constraint.

4.3.2 Feature 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 both. Generally we consider same sets for traffic and topological information, but allow slight deviation to make maximum utilization of routing information. Even under measures of the same name, different protocols interpret them 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.

We adopt following features in Table 4.2 to build a model on DSR/DSDV protocols. Notice features are collected from three sources, Route Cache, traffic pattern, and movement information of the host. All features are locally collected.

4.3.3 Scenario

A trace is first created under default scenario, i.e., 10 data sources, the running time is 100000 seconds, radio range is 330. All nodes are required to move in a rectangular area of 1000 unit by 1000 unit. Their maximum speed is 20 unit per second.
Table 4.2: Local Features on Ad-Hoc Protocols

<table>
<thead>
<tr>
<th>Features</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSTC</td>
<td>% of Significant Traffic Change</td>
</tr>
<tr>
<td>VELOCITY</td>
<td>Velocity</td>
</tr>
<tr>
<td>DISTANCE</td>
<td>Distance from last log</td>
</tr>
<tr>
<td>RDC</td>
<td>Relative Distance Change</td>
</tr>
<tr>
<td>PCR</td>
<td>% of Change in Route entries</td>
</tr>
<tr>
<td>PCH</td>
<td>% of Change in number of Hops</td>
</tr>
<tr>
<td>LINK_BAD</td>
<td>% of Change in Bad Links</td>
</tr>
<tr>
<td>RT_ADD</td>
<td>% of Route newly added</td>
</tr>
<tr>
<td>DSR Specific</td>
<td></td>
</tr>
<tr>
<td>RT_FIND</td>
<td>% of Route newly being found</td>
</tr>
<tr>
<td>RT_NOTICE</td>
<td>% of Route noticed</td>
</tr>
<tr>
<td>DSDV Specific</td>
<td></td>
</tr>
<tr>
<td>MT_ADD</td>
<td>% of Hops Added</td>
</tr>
<tr>
<td>RT_CHG</td>
<td>% of Route Change</td>
</tr>
<tr>
<td>MT_CHG</td>
<td>% of Hops Change</td>
</tr>
</tbody>
</table>

We also use 10 UDP traffic with constant-bit-rate (CBR) with random source and destination. Each data packet is 512 bytes in length, and the transmission rate is fixed on 20 packets/s or 10k bytes/s. A maximum of 10000 packets are transmitted within one flow.

4.3.4 Models

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

First, two traces are created from our simulation script xMobile. The only difference between these two traces is that one disables the intrusion mode (i.e., it is pure normal) while the other enables it by choosing parameter intrpOn=1000 and intrpOff=10000, which means an intrusion could last as long as 1000 seconds while the interval between two con-
secutive intrusions is 10000 seconds. We use the random (2) for the intrpMethod parameter so as we can have random intrusions in the trace.

Next, from each trace we extracted observation points as training data. Each record in the training data is a sequence of observation points, \( O_0, O_{-1}, \ldots, O_{-n} \), where each \( O_{i-1} \) is the observation preceding \( O_i \) in timestamp order. Each \( O_i \) is described by a set of features \( < f_1, f_2, \ldots, f_m > \) described in Table 4.2. 50\% of the training data comes from the normal trace and the other 50\% from the intrusion trace. Each record is labeled as “normal” or “abnormal”.

From the discussion on sequence length determination in Section 2.3.2, we determine sequence length \( n \) by a simple threshold. We choose \( n \) from 2 and stop when a threshold on conditional entropy is met. The reason we do not choose \( n \) starting from 1 as the scripts we use to be parsed on to SVM.Light classifier requires a multiple feature set, and the length of which must not be one. Besides that, a longer sequence length is unnecessary, as the computation cost (information cost) grows in proportional to the sequence length. For illustration purpose, we continue to calculate corresponding accuracy when a longer sequence is used till a large enough value (which is determined in advance, we use 10) is reached. By computing conditional entropy and using a threshold of 0.1, we chose \( n = 2 \) for DSR, and \( n = 8 \) for DSDV. (In comparsion of the actual bounded sequence \( n = 1 \) for DSR, our computation cost is increased a little bit only.) In Figure 4.1 ,we discover that The DSR results already contain high regularity features and thus a small sequence length is sufficient. The situation changes for results from DSDV. As sequence length increases, conditional entropy drops, detection performance improves dramatically, while the false alarm rate is still bounded in a very low level. From the threshold we use, sequence longer
than what we choose achieve no significantly higher accuracy. The false alarm rate is low
when a smaller sequence length is chosen, unsurprisingly, as, normal behavior in a short
sequence is more possible to be fully enumerated and remembered. However, the detection
rate is much lower, since information gain is small, anomalies are still hard to distinguish
from normalcy.

After the model is trained, we chooses several test scripts with different parameter
settings, the predicted result is then filtered by the post processing scheme suggested above.
Here we choose l=3.

4.3.5 Results

We now illustrate the input and output on each steps. The results are modified slightly to
accommodate the format requirement.

The following results are generated automatically from traffic pattern generation script.

```
# nodes: 10, max conn: 10, send rate: 20.0, seed: 0.0
#
# 9 connecting to 6 at time 6683.1309519163942
# set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(0)
$ns_ attach-agent $node_(6) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 10.0
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 6683.1309519163942 "$cbr_(0) start"
#
# 7 connecting to 8 at time 6995.6966475563577
# set udp_(1) [new Agent/UDP]
# traffic 2...
#
# 6 connecting to 0 at time 5843.1785860299969
# traffic 3...
# more traffic
```

39
Figure 4.1: Accuracy vs. Sequence Length
The following results are generated automatically from connection pattern generation script.

```plaintext
# Total sources/connections: 10/10

# nodes: 10, pause: 100.00, max speed: 20.00 max x = 1000.00, max y: 1000.00, max range: 330.00

$node_(0) set X_ 701.823032750873
$node_(0) set Y_ 539.71206519323
$node_(0) set Z_ 0.000000000000 # always zero in ns-2
$node_(1) set X_ 940.702469010870
$node_(1) set Y_ 386.397500216308
$node_(1) set Z_ 0.000000000000
# initial locations of every node...
$node_(9) set X_ 184.007984662033
$node_(9) set Y_ 622.198377992804
$node_(9) set Z_ 0.000000000000
$god_ set-dist 01 1 # node 0 and node 1 are neighbors
$god_ set-dist 0 2 16777215 # no routes available between node 0 and node 2
$god_ set-dist 0 3 2 # a route of 2 hops can connect 0 to 3
$god_ set-dist 0 4 3
$god_ set-dist 0 5 16777215
$god_ set-dist 0 6 2
$god_ set-dist 0 7 1
$god_ set-dist 0 8 2
$god_ set-dist 0 9 16777215
$god_ set-dist 1 2 16777215
$god_ set-dist 1 3 1
$god_ set-dist 1 4 2
# ...
$god_ set-dist 2 3 16777215
$god_ set-dist 2 4 16777215
# ...
$ns_ at 100.000000000000 "$node_(0) setdest 288.139477025474 760.190622729259 \ 10.475937704999"
# At 100s, node 0 starts moving towards (288,760) with a speed of 10.48.
$ns_ at 100.000000000000 "$node_(1) setdest 454.247079045864 530.657926793446 \ 15.355217634996"

# more movements...
$ns_ at 103.796066720490 "$god_ set-dist 0 8 1"
# At 103s, node 0 and node 8 can communicate directly
$ns_ at 105.904567673590 "$god_ set-dist 1 8 1"
# ...
$ns_ at 107.208667592116 "$god_ set-dist 0 9 2"
$ns_ at 107.208667592116 "$god_ set-dist 1 5 2"
# ...
$ns_ at 133.043177270314 "$node_(1) setdest 454.247079045864 530.657926793446 \ 0.000000000000"
# ...
$ns_ at 138.57544023982 "$god_ set-dist 1 7 2"
# ...
$ns_ at 99994.098544225428 "$god_ set-dist 4 8 1"
# Destination Unreachables: 2044
# Route Changes: 9291
# Link Changes: 1938
# Node | Route Changes | Link Changes
```

41
The output from ns-2 is a trace of internal routing information and packet deliveries from specific protocols. Notice that we do not explicitly output location information from ns-2. It is analyzed and extracted from movement pattern traces. Following is a piece from output for DSDV protocol.

```
M 0.0 nn:10 x:1000 y:1000 rp:DSDV
M 0.0 sc:seq-dsv-10-100000-330-100-20-cbr-10-0.1.sc cp:seq-dsv-10-100000-330-
100-20-cbr-10-0.1.cp seed:0.0
M 0.0 prop:Propagation/TwoRayGround ant:Antenna/OmniAntenna
# _0_ is the source id. One entry in routing table, destination/hop/metric is \
O/O/O, which means a plain entry which always exists.
SRC 1.0000 _0_ cache-summary 1 0 0 0 normal
SRC 1.0000 _1_ cache-summary 2 1 1 0 2 2 1 normal
SRC 1.0000 _2_ cache-summary 1 2 2 0 normal
SRC 1.0000 _3_ cache-summary 2 1 1 1 3 3 0 normal
SRC 1.0000 _4_ cache-summary 1 4 4 0 normal
SRC 1.0000 _5_ cache-summary 1 5 5 0 normal
SRC 1.0000 _6_ cache-summary 1 6 6 0 normal
SRC 1.0000 _7_ cache-summary 1 7 7 0 normal
SRC 1.0000 _8_ cache-summary 1 8 8 0 normal
SRC 1.0000 _9_ cache-summary 1 9 9 0 normal
SRC 2.0000 _0_ cache-summary 1 0 0 0 normal
SRC 2.0000 _1_ cache-summary 2 1 1 0 2 2 1 normal
SRC 2.0000 _2_ cache-summary 1 2 2 0 normal
SRC 2.0000 _3_ cache-summary 2 1 1 1 3 3 0 normal
SRC 2.0000 _4_ cache-summary 1 4 4 0 normal
SRC 2.0000 _5_ cache-summary 1 5 5 0 normal
SRC 2.0000 _6_ cache-summary 1 6 6 0 normal
SRC 2.0000 _7_ cache-summary 1 7 7 0 normal
SRC 2.0000 _8_ cache-summary 1 8 8 0 normal
SRC 2.0000 _9_ cache-summary 1 9 9 0 normal
SRC 3.0000 _0_ cache-summary 1 0 0 0 normal
SRC 3.0000 _1_ cache-summary 2 1 1 0 2 2 1 normal
SRC 3.0000 _2_ cache-summary 1 2 2 0 normal
SRC 3.0000 _3_ cache-summary 2 1 1 1 3 3 0 normal
SRC 3.0000 _4_ cache-summary 1 4 4 0 normal
SRC 3.0000 _5_ cache-summary 1 5 5 0 normal
SRC 3.0000 _6_ cache-summary 1 6 6 0 normal
SRC 3.0000 _7_ cache-summary 1 7 7 0 normal
SRC 3.0000 _8_ cache-summary 1 8 8 0 normal
SRC 3.0000 _9_ cache-summary 1 9 9 0 normal
SRC 4.0000 _0_ cache-summary 1 0 0 0 normal
SRC 4.0000 _1_ cache-summary 2 1 1 0 2 2 1 normal
SRC 4.0000 _2_ cache-summary 1 2 2 0 normal
SRC 4.0000 _3_ cache-summary 2 1 1 1 3 3 0 normal
SRC 4.0000 _4_ cache-summary 1 4 4 0 normal
```
By combining ns-2 trace output and movement pattern trace, we analyze those features listed in Table 4.2 on each node. Temporal sequences on those features are then constructed.

As we use Bi-Class model, we preserve their original class labels.

The input for RIPPER on DSR protocol resembles this,

```
# NODE, PSTC1, VELOCITY1, DISTANCE1, RDC1, PCR1, PCH1, ..., PSTC2, VELOCITY2, DISTANCE2, ..., class.
1, 0, 15.36, 15.36, 0, 0, 0, ..., 0, 10.48, 10.48, 10.48, ..., normal.
0, 0, 10.48, 10.48, 0, 0, 0, ..., 0, 10.48, 10.48, 10.48, ..., normal.
#...
0, 70, 8.44, 8.44, 3, 2, 3, ..., 0, 8.44, 8.44, 8.44, ..., abnormal.
#...
```

The following rule sets are extracted from output of RIPPER on DSR protocol on a smaller dataset (otherwise, the related features shown on the right are much longer).

Final hypothesis is:
abnormal :- RT_FIND1<=0, VELOCITY1>=9.45, VELOCITY2<=12.38, VELOCITY2>=7.72 (35/3).
abnormal :- RT_FIND1<=0, LINK_BAD2<=0.05, VELOCITY2>=4.21, VELOCITY2<=6.93 (93/23).
...
abnormal :- LINK_BAD1<=0, RT_FIND2<=0.05, VELOCITY2<=7.48, VELOCITY2>=6.93 (24/1).
default normal (654/53).

For example, the first rule says if RT_FIND from first observation is not positive (actually it means zero as no negative values are given), velocity from first observation no larger than 9.45 and velocity from second observation is within the range from 7.72 to 12.38, then the event is an abnormal event. The rule matches 35 sequences but misses 3.

For SVM_Light, the input format using a scarce scheme and can be converted from ripper format. The output of SVM_Light is composed a set of support vectors and much harder to read by human.

SVM-light Version V3.50
0 # kernel type
3 # kernel parameter -d
1 # kernel parameter -g
1 # kernel parameter -s
1 # kernel parameter -r
empty # kernel parameter -u
208 # highest feature index
9726 # number of training documents
2819 # number of support vectors plus 1
-0.77013767 # threshold b

# Support Vector I
-0.051018815811832894657751324984929 1:0.0524 2:0.0524 22:0.0524 23:0.0524
43:0.0524 44:0.0524 65:0.0524 85:0.0524 86:0.05
24 106:0.0524 107:0.0524 127:0.0524 128:0.0524 148:0.0524 149:0.0524
169:0.0041999999 170:0.0041999999 192:0.0099999998 194:0.029999
999 199:0.029999999 200:0.0099999998 202:0.029999999 204:0.029999999
# Support Vector II
# ...

We first choose five different test scripts. normal is a normal trace, 100k-rt is a trace with same running time as trace data, and does attacks by creating randomly new source routes directly in route table. 100k-tf is another trace with distortion on traffic patterns. 10k-rt and 10k-tf use same preference, but with running time of 10,000 seconds. For each trace, we run 10 times and calculate average results as well as 95% confidence intervals.
The confidence interval is determined by the following formula

$$\Pr \left( \bar{X} - \frac{bs}{\sqrt{n-1}} < \mu < \bar{X} + \frac{bs}{\sqrt{n-1}} \right) = 0.95$$ (4.1)

where $X$ is the random variable, $n$ is the count of running times, $\bar{X}$ is the expected value of $X$, $s$ is the standard deviation of $X$. $b$ is a number determined by $n$ (10) and probability (0.95), which is available for lookup from any statistics textbook. The value in our case is 2.262. Then the interval $[\bar{X} - \frac{bs}{\sqrt{n-1}}, \bar{X} + \frac{bs}{\sqrt{n-1}}]$ is a 95 percent confidence interval.

Table 4.3: Anomaly Detection on DSR with several experiments

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

Table 4.4: Anomaly Detection on DSDV with several experiments

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

The results in Table 4.3 and 4.4 are detection rates and false alarms rates in terms of individual records (not intrusion sessions). The results from DSR are much better than DSDV because DSR is preferred model, built by SVM, outperform DSDV’s RIPPER and
SVM models. the greatest misclassification rate using RIPPER is 15%, while the SVM method can achieve only 0.07% error.

Secondly, traffic distortion tends to have a worse performance than routing information distortion, which shows our model may not contain adequate data to train all traffic patterns. This is likely due to our feature selection only reflects significant traffic pattern change from router’s view. We will still see, however, in DSR, strong correlation among traffic pattern and routing changes help to alleviate the problem. A better model, of course, should introduce a complete flow-monitoring scheme as a strong signal generator for intrusion detection. Such a scheme, however, is distributed cooperative in nature. We will study such a cooperative system, by using the basic local model we described in this thesis as a module, in our future work.

We then used DSR/SVM for more 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 4.5. Detection and false alarm rates on individual records and intrusion sessions (after post-processing) are listed. The results show that our model is very steady in performance and provides high accuracy even in a much bigger data set.

Are our results good enough? Considering alarm signals are further treated by a human operator. One convention defined for goodness is that no more than one false alarm signal can be allowed in one hour, which means $1/86400$ false alarm rate in our case, or 0.001%. Therefore, our results are only preliminary and cannot satisfy the standard yet.
4.4 Discussion

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

The issue of spurious error leads 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 attack [FHSL96, LS98]. It shows there are some natural limits on detection capabilities, depending on data at which layer you collect. Thus, in our routing protocol level, we also believe amendment can be made for above situation when cooperation of IDS on different layers is available. Interestingly, intrusion in medium access control (MAC) layer can also happen and [ZL00] suggests to
apply similar anomaly detection approaches on MAC layer.

In this experiment, we also find a few system parameters that may influence normal behavior heavily. The mobility level is one of them – 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, so it remains a future work for us. It nevertheless teaches us an important lesson that a good anomaly detection model should collect every possible value combinations.

Having done the experiment on two 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, even in the first look its route update is not as “regular” as DSDV. After detailed analysis of the two 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 complete 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, DSR 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 very weak correlation between control traffic and data traffic, even when we preserve the traffic feature, it is almost ignored by rule-based classifiers like RIPPER. We therefore believe that those types of redundancy should contribute to DSR’s superior performance.

The next question, naturally, is 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 protocol with high correlation among changes of three different 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 referred by a protocol when making route decisions. For example, new routing protocols such as Location-Aided Routing protocol [KV00], which attempts to utilize topological information, should have been more advantageous. However, we have not found their implementation on ns-2, thus we cannot conduct an experiment immediately. We, however, conjecture that it should improve the accuracy of our current model.
Chapter 5

Related Work

5.1 Intrusion Detection System

Anomaly Detection is an important research area in intrusion detection, while misuse pattern recognition systems do well on recognizing existing attacks, they have difficulties in discovering new attacks or even minor variations of existing ones. Anomaly detection does not have this drawback. In earlier systems [AFV95, Sma88, Labe], anomaly detection accumulate statistics from system measures, e.g., the CPU usage, the number of shell commands used etc. In several recent studies, learning-based approaches were applied to build anomaly detection models using system call data of or privileged programs [FHSL96, GS99, WFP99]. Such detection helps to improve the scale of observation and can find attacks which left few hints on global system behaviors. Lane et al. [LB99] proposed a learning algorithm for analyzing user shell command history to detect anomalies. The algorithm attempts to address the ”concept drift” problem, i.e., when the normal user behavior changes. While these systems all have some degree of success, they were
developed for a particular kind of environment. The fundamental question of “how to build and evaluate anomaly detection in general” has not been adequately addressed. As a result, the approaches developed in these studies may not be applicable to other environments.

Research have begun to develop principles and theories for intrusion detection. Axelsson [Axe00] pointed out that the established field of detection and estimation theory bears similarities with the IDS domain. Therefore, the results from detection and estimation, which have been found applicable to a wide range of problems, many be used in the IDS domain.

### 5.2 Information Theory

The most related work is by Maxion et al. [MT00], where the relationship between data regularity and detection performance of anomaly detection model was studied. The study, based on synthetic train data, suggests that the current practice of deploying a particular anomaly detection system across different environments is perhaps flawed and should be reconsidered, as the detection performance varies when the regularity of test data set varies. One of our study [LX01] also confirmed this finding in that we showed that the expected detection performance can be attained only when the relative conditional entropy between the training and test datasets is small.
Security issues for infrastructured wireless systems [BN00, SGP00, Moh96] have been well studied, but there are still only a few security schemes proposed for the MANETS.

Two types of security schemes are proposed for ad-hoc routing networks. One type belongs to proactive schemes, generally, standard schemes such as key generation and management service is used in a distribute manner among individual host participated in authentication, so as to insure routing information’s authenticity and integrity not being tampered.

Zhou and Haas [ZH99] have introduced a distributed key management service, which is routing protocol independent. The proposal applies redundancies in the network topology to provide reliable key management. The key idea is to use key sharing with the assumption that the ratio between nodes compromised to total nodes can be bounded. One theorem they proved is that if the upper limit for the number of compromised server nodes can be set to $t < 1$, at least $n = (3t + 1)$ nodes are needed to enable the scheme. The architecture does require that the underlying routing protocol can manage multiple routes. Since no center authority is required, the proposal implements a distributed trust model.

Experiment also has been performed by J. Binkley [Bin96] when authentication of MAC and IP address exchange has been attempted. Jacobs and Corson [JC99] has also proposed an authentication architecture where the emphasis is building a hierarchy of trust relationship in order to authenticate IMEP messages security. The proposed scheme details the formats of messages, together with protocols that achieve authentication. The architecture can accommodate different authentication schemes.
The difficulty related with such proactive schemes is that, first, cryptography is relatively computationally expensive on mobile hosts, where computational capability is comparatively restricted. second, since no center authority can be depended on, the authentication is more difficult to implement, third, it is only useful to prevent intruders from outside (external attack). If an internal node is compromised (internal attack), such schemes no longer work.

Another type of schemes is to detect intrusions from existing information and responds promptly. Intrusion detection schemes can detect both external and internal attacks. For example, Smith et al. [SMGLA97] suggests methods to secure distance-vector routing protocols are proposed, extra information of a predecessor in a path to a destination is added into each entry in 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 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.

An architecture for distributed and cooperative intrusion has been presented in our previous work [ZL00]. In this architecture, each node participates and is responsible for detecting anomalies locally and independently, but neighboring nodes could collaborate, with each node running an Intrusion Detection System agent. If an anomaly is detected in local data, neighboring IDS cooperatively participate in global intrusion detection. We expect to continue our work based on this paper and eventually nuild a distributed and cooperative intrusion detection system for wireless ad-hoc networks.
Chapter 6

Conclusion and Future Work

6.1 Conclusion

Routing protocols for wireless ad-hoc networks are designed to meet the design requirements of distributed routing and dynamic network topology. Security concerns are often an after-thought. We pointed out these protocols are vulnerable and require security protection. We proposed to use anomaly detection models constructed using information available from those protocols for intrusion detection purposes. One advantage using anomaly detection scheme is that it requires less modification on current routing protocols. We suggested that we should use only local data sources because remote nodes can be compromised and can not be trusted blindly. We used the heuristic that there are temporal (sequential) patterns in the observations and computed information-theoretic measures (e.g., conditional entropy) for feature selection. We applied two different tools, RIPPER and SVM,Light, to compute classifiers as anomaly detectors. 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. We also list some comments for potential security benefit in future protocol design.

6.2 Future work

There are some limitations in our study that we will address, an outline of proposed work in future is presented as well.

- More Routing Protocols

  We plan to study other routing protocols (in addition to DSR and DSDV) to better understand the relationship between routing information and detection performance. We notice that different protocols prefer different classification algorithms when detection performance is concerned. We will further investigate this phenomenon.

  We have planed to make extensive study on other different protocols, to justify our conclusion from the comparative study between two typical protocols.

- Information Measures

  We have not studied the behavior of sequences of multiple features thoroughly. The preliminary research we currently used in pure normal data model virtually pick one
of the features randomly in the last observation as the predicted class. As we ob-
served correlations exist among different features as well, not only among temporal
sequences, we have planed to study how to utilize two type of regularities more effi-
ciently.

- Data set Size Requirement

we have not studied how to estimate when the training data set is “sufficiently” com-
plete when only pure normal data are available. It is intuitive to derive that a much
larger data set is required. However, we should carefully study the physical data set
size requirement and hence determine the condition under which our model can be
applied in real-time applications.

- Parameter Space Partition

Second, our model is built using traces from ns-2 and is thus influenced by its system
parameters. It seems unrealistic that we can build one model, from a particular set
of parameters, for anomaly detection in general situations. We need a method to
partition the wide range of parameter settings into categories, possibly according to
some typical application-specific scenarios, and build a model for each category.

- Global Cooperative Study

Our current research restricts to local routing level. From our previous discussion, we
have noticed that integration among different layers and different hosts is necessary
to detect certain type of intrusions.

To evolve into a more powerful detection system, we will try some preliminary host
cooperation schemes first, where each host uses our anomaly detection approach locally. We then expect to build a complete intrusion detection system on top of wireless ad-hoc networks.
Bibliography


Los Alamos National Laboratory. Wisdom and sense guidebook. Los Alamos National Laboratory, Undated.


