Cross-Feature Analysis for Detecting Ad-Hoc Routing Anomalies

Yi-an Huang [1] * Wei Fan [2] Wenke Lee [1] Philip S. Yu [2]

[1]College of Computing, Georgia Institute of Technology, Atlanta, GA 30332

{yian,wenke}@cc.gatech.edu

[2]IBM T. J. Watson Research, Hawthorne, NY 10532

{weifan,psyu}@us.ibm.com

Abstract

With the proliferation of wireless devices, mobile ad-hoc networking (MANET) has become a very exciting and important technology. However, MANET is more vulnerable than wired networking. Existing security mechanisms designed for wired networks have to be redesigned in this new environment. In this paper, we discuss the problem of intrusion detection in MANET. The focus of our research is on techniques for automatically constructing anomaly detection models that are capable of detecting new (or unseen) attacks. We introduce a new data mining method that performs "cross-feature analysis" to capture the inter-feature correlation patterns in normal traffic. These patterns can be used as normal profiles to detect deviation (or anomalies) caused by attacks. We have implemented our method on a few well known ad-hoc routing protocols, namely, Dynamic Source Routing (DSR) and Ad-hoc On-Demand Distance Vector (AODV), and have conducted extensive experiments on the ns-2 simulator. The results show that the anomaly detection models automatically computed using our data mining method can effectively detect anomalies caused by typical routing intrusions.

1. Introduction

In recent years, with the rapid proliferation of wireless devices, the potentials and importance of mobile ad-hoc networking have become apparent. A mobile ad-hoc network is formed by a group of mobile wireless nodes often without the assistance of fixed or existing network infrastructure. The nodes must cooperate by forwarding packets so that nodes beyond radio ranges can communicate with each other.

With a striking similarity of the early days of Internet research, security issues in ad-hoc networking have not yet been adequately investigated in the current stage. MANET is much more vulnerable than wired (traditional) networking due to its limited physical security, volatile network topologies, power-constrained operations, intrinsic requirement of mutual trust among all nodes in underlying protocol design and lack of centralized monitoring and management point. There are recent research efforts, e.g., [8, 12], in providing various *prevention* schemes to secure the ad-hoc routing protocols, i.e., authentication and encryption schemes. However, the history of security research on the wired environments has taught us that we still need to deploy defense-in-depth or layered security mechanisms because security is a process (or a chain) that is as secure as its weakest link. For example, confidential data transmitted via an encrypted link can still be stolen from the end systems simply because of break-ins that exploit "weak passwords" or system software bugs, which are unlikely to be completely eliminated. Consequently, a second layer of security, i.e., *intrusion detection* and response is needed.

There are two major analytical techniques in intrusion detection, namely *misuse detection* and *anomaly detection*. *Misuse detection* uses the "signatures" of known attacks, and *anomaly detection* uses established normal profiles only to identify any unreasonable deviation from them as the result of some attack. Since MANET is still under heavy development and not many MANET-specific attacks have emerged, we believe that anomaly detection is the preferred technique in the current stage.

Our anomaly detection approach is based on data mining technologies because we are interested in *automatically* constructing detection models using logs (or trails) of system and network activity data. Some intrusion detection techniques suggested in literature use probabilistic analysis where the resulting models are not straightforward to be re-evaluated by human experts [6]. Some data mining models require temporal sequence from data stream [11], which is domain specific and highly inefficient when a large feature set is involved. The problem of anomaly detection in MANET involves a large feature set. It requires us to de-

*This work was completed when the author was a summer intern at IBM T. J. Watson Research Center.

velop new data mining approaches.

We have developed a new approach based on cross-feature analysis that we believe is suitable for MANET anomaly detection. We observe that strong feature correlation exists in normal behavior patterns. And such correlation can be used to detect deviations caused by abnormal (or intrusive) activities. For instance, consider a home network which is mainly composed of wirelessly connected home appliances and possibly a few human held wireless devices (such as PDAs). Networking controllers reside in all such nodes so that they can form an ad-hoc network. Naturally, we would expect that major portion of the established routing fabric remains stable for a long time since home appliances rarely change locations. We also require that some sensor device be installed on each node which can record useful statistics information. Let's imagine that one sensor finds out that the *packet dropping rate* increases dramatically without any noticeable change in the *change rate of routing entries*, it is highly likely something unusual has happened. The controller in the node may have been compromised to refuse forwarding incoming traffic while no route change actually takes place (which, if happens, may result in temporary packet dropping due to invalid stale routes). The relationship between the features *packet dropping rate* and *change rate of routing entries* can be captured by analyzing the normal patterns of historical data and be used later to detect (unseen) anomalies.

More formally, in the cross-feature analysis approach, we explore correlations between each feature and all other features. Thus, the anomaly detection problem can be transformed into a set of classification sub-problems, where each sub-problem chooses a different feature as a new class label and all other features from the original problem are used as the new set of features. The outputs of each classifier are then combined to provide an anomaly detector. In our study, we use two routing protocols, namely, DSR [10] and AODV [14], and collect trace logs of normal and abnormal data in the ns-2 simulator [5]. Our experiment results show that the anomaly detection models, trained on the normal traces using our data mining approaches, can identify different routing anomalies very effectively.

The rest of the paper is organized as follows. We briefly introduce a few MANET routing protocols used in our experiments and analyze different threats against these protocols. We then present a cross-feature analysis framework for anomaly detection. After that, we present case studies on MANET routing attack detection problem together with discussion and findings from these experiments. The paper concludes with a summary and an outline of future work.

2. Routing in MANET

We study two popular MANET routing protocols implemented in ns-2 in our case study. These protocols are Ad-hoc On-Demand Distance Vector Routing (AODV) and Dynamic Source Routing (DSR). There are some other MANET routing protocols such as ZRP [7]. We consider the selected protocols because first of all, they are already implemented in ns-2 and, secondly, they have been intensively studied in recent ad-hoc networking research due to their simplicity and competitive performance under high load and mobility. Both of them are on-demand (or reactive) protocols which only request for new routes if necessary (demanded by data traffic when no route is available or established yet).

2.1. DSR

DSR uses source routing to deliver packets through MANET. That is, the sender of a data packet finds a source route (i.e., a full path from the sender to the receiver) and includes it in the packet header. The intermediate nodes use this information to determine whether they should accept a packet and where to forward it. The protocol operates on two mechanisms: route discovery and route maintenance. Route discovery is used when the packet sender has not yet known the correct path to the packet destination. It works by broadcasting a `ROUTE REQUEST` message throughout the network in a controlled manner until it is answered by a `ROUTE REPLY` message from either the destination itself or an intermediate node that knows a valid path to it. For better performance, the source and intermediate routes save the route information in cache for future use. Furthermore, intermediate nodes can also learn new routes by eavesdropping to other route discovery messages taken place in the neighborhood. Finally, route maintenance mechanism is used to notify source and potentially trigger new route discovery events when changes in the network topology invalidate a cached route.

2.2. AODV

AODV, or *Ad hoc On-demand Distance Vector*, is another on-demand protocol, whose route discovery process is also reactive on an *as needed* basis. It differs from DSR in that it does not utilize source routes and instead refer to a route table stored in each node. A route table records all reachable nodes in the network with the following information: next hop, distance and a sequence number to maintain freshness. The protocol borrows ideas from distance vector based algorithms in wired networks. The route table is used to respond to `ROUTE REQUEST`s and later decide

the next hop for incoming data packets needed to be forwarded. Readers interested in the performance evaluation between these protocols can refer to [13] which presents an extensive study of measuring and comparing routing performance and requirements between AODV and DSR.

2.3. Attacks

We classify attacks on MANET routing protocols into the following two categories, based on the underlying routing functionality.

- **Route logic compromise**: This type of attacks involves those where incorrect routing control messages are injected into the network to subvert or damage route fabrics. They can be further divided into external attacks and internal attacks. External attacks are launched from the outside of the network, while internal attacks are initiated by one or more compromised node(s). For instance, *Black hole* is a classic routing attack where a malicious node advertises itself as having the shortest path to all nodes. It can be used as a denial-of-service attack where it can drop the packets later. Or it can be followed by traffic monitoring and analysis to find activity patterns of each node. Sometimes it becomes the first step of a man-in-the-middle attack. Another interesting attack is called *Update storm*. The malicious node deliberately floods the whole network with meaningless route discovery messages or `ROUTE REPLY` messages. The purpose is to exhaust the network bandwidth and effectively paralyze the network. The first attack is internal, while the second can be either external or internal.

- **Traffic distortion**: This type of attacks can snoop network traffic, manipulate or corrupt packet header or contents, block certain types of traffic or replay transmissions for some malicious purposes. One such example is *Packet dropping*. It does not actively change routing behavior as *Black hole* does. Instead it simply drops data or route packets when it feels necessary. Based on the frequency and selectiveness, it has the following variations. A **random dropping** attack drops packets randomly. A **constant dropping** attack drops packets all the time. A **periodic dropping** drops packets periodically to escape from being suspected. A **selective dropping** attack drops packets based on its destination or some other characteristics. It can be introduced to specifically create a partitioning of the entire network [17]. Another example is *Identity impersonation*. Attackers can impersonate another user to achieve various malicious goals. In a networked environment, it is important to correctly attribute user behavior with proper user identity. Pointing to an innocent individual as the culprit can be even worse than not finding any identity responsible at all. In particular, IP and MAC (Medium Access Control) addresses can be used as identification purposes. Unfortunately, these identities are easy to be forged during the transmission of data packets on network or link layers if the underlying communication channel is not secured.

The advantage of the above attack classification system is that since the functionalities of a router in any routing protocols generally involve both (a) establishing route path for future packet delivery, and (b) forwarding data packets based on established routes. Eventually, all attacks on routing protocols accomplish their goals by compromising at least one of the routing functionalities. Therefore, by collecting and analyzing proper routing and traffic related measures, we expect to detect possible attacks to the routing protocols. Note that the two types of attacks are not exclusive. It is not difficult to fabricate intrusions with combined attacks from both categories. However, most of these combined attacks can be partitioned into a few smaller sub-attacks, and each of them can be either route or traffic specific.

3. Cross-Feature Analysis

The basic idea of a **cross-feature analysis** framework is to explore the correlation between one feature and all the other features, i.e., try to solve the classification problem $\{f_1, f_2, \ldots, f_{i-1}, f_{i+1}, \ldots, f_L\} \rightarrow f_i$ where $\{f_1, f_2, \ldots, f_L\}$ is the feature vector. Note that in the machine learning area, the terminology *class* in a classification system represents the task to be learned based on a set of features, and the *class label*s are all possible values a class can take. In the domain of intrusion detection, we would most likely to learn the system healthy status (the *class*) from known system information (the *features*), and *normal* or *abnormal* are both possible class labels.

A basic assumption for anomaly detection is that normal and abnormal events should be able to separate from each other based on their corresponding feature vectors. In other words, given a feature vector, we can tell whether the related event is normal or not without ambiguity. This assumption is reasonable since otherwise the feature set is not sufficient and must be redefined. Under this assumption, we can name a feature vector related to a normal event a **normal vector**, for short. Similarly, we call a feature vector not related to any normal events an **abnormal vector**. Here, we assume that all feature values are discrete. A generalized extension will be discussed later in the paper. We re-formulate the problem as follows. For all normal vectors, we choose one feature as the target to classify (which is called the **labeled feature**), and then compute a model using all normal

vectors to predict the chosen target feature value based on remaining features. In other words, we train a classification model $C_i : \{f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_L\} \rightarrow \{f_i\}$. For normal events, the prediction by C_i is very likely to be the same as the true value of the feature; however, for anomalies, this prediction is likely to be different. The reason is that C_i is trained from normal data, and their feature distribution and pattern are assumed to be different from those of anomalies. This implies that when normal vectors are tested against C_i, it has a higher probability for the true and predicted values of f_i to match. Such probability is significantly lower for abnormal vectors. Therefore, by evaluating the degree of result matching, we are more likely to find difference between normal and abnormal patterns. We name the model defined above a **sub-model with respect to** f_i. Obviously, relying on one sub-model with respect to one labeled feature is insufficient as we haven't considered the correlation among other features yet. Therefore the model building process is repeated for every feature and up to L sub-models are trained. Once done, we have accomplished the first step of our cross-feature analysis approach, i.e. the training procedure, which is summarized in Algorithm 1.

Data: feature vectors of training data $f_1, \ldots, f_L$;
Result: classifiers $C_1, \ldots, C_L$;
begin
  $\forall i$, train $C_i : \{f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_L\} \rightarrow f_i$;
  return $C_1, \ldots, C_L$;
end

Algorithm 1: Cross-Feature Analysis: Training Procedure

To generalize the framework to continuous features or discrete features with an infinite value space (e.g., the integer set), we should keep in mind that they cannot be used directly as class labels since only discrete (nominal) values are accepted. We can either discretize them based on frequency or use multiple linear regression. With multiple linear regression, we use the log distance, $|log(\frac{C_i(x)}{f_i(x)})|$, to measure the difference between the prediction and the true value, where $C_i(x)$ is the predicted value from sub-model with respect to f_i.

Once all sub-models have been trained, we analyze trace logs as follows. When an event is analyzed, we apply the feature vector to all sub-models, and count the number of models whose predictions match the true values of the labeled features. The count is then divided by L, so that the output, which is called the **average match count** throughout the paper, is normalized. We do not need all sub-models to match. In fact, what we need is a *decision threshold*. An event is classified as anomaly if and only if the *average match count* is below the threshold. Since it is hard to develop a perfect solution to determine the decision threshold for a general anomaly detection problem directly. and in practice, a small value of false alarm rate is often allowed, we can determine the threshold as follows: compute the *average match count* values on all normal events, and use a lower bound of output values with certain confidence level (which is one minus false alarm rate). As a summary, Algorithm 2 lists the strawman version of the test procedure. For convenience, $f_i(x)$ denotes the value of feature f_i belonging to event x. $[\![\pi]\!]$ returns 1 if the predicate π is true.

Data: classifiers $C_1, \ldots, C_L$, event $x = (f_1, \ldots, f_L)$, decision threshold θ;
Result: either normal or anomaly;
begin
  AvgMatchCount $\leftarrow \sum_i [\![C_i(x) = f_i(x)]\!]/L$;
  if *AvgMatchCount* $\geq \theta$ **then** return "normal";
  else return "anomaly";
end

Algorithm 2: Cross-Feature Analysis: Testing Procedure Using *Average Match Count*

One straightforward improvement to the strawman algorithm is to use probability instead of the 0-1 count, the probability values for every possible class are available from most inductive learners (e.g., decision trees, induction rules, naive Bayes, etc.) This approach can improve detection accuracy since a sub-model should be preferred where the labeled feature has stronger confidence to appear in normal data. Algorithm 2 can actually be regarded as a special case under the assumption that the predicted class is the only valid class and hence has a probability of 1.0, so the probability for the true class is either 1 (when the rule matches) or 0 (otherwise). More strictly, assume that $p(f_i(x)|x)$ is the estimated probability for the true class of the labeled feature, we define **average probability** as the average output value of probabilities associated with true classes over all classifiers. The optimized version is shown in Algorithm 3.

Data: classifiers $C_1, \ldots, C_L$, event $x = (f_1, \ldots, f_L)$, decision threshold θ;
Result: either normal or anomaly;
begin
  AvgProbability $\leftarrow \sum_i p(f_i(x)|x)/L$;
  if *AvgProbability* $\geq \theta$ **then** return "normal";
  else return "anomaly";
end

Algorithm 3: Cross-Feature Analysis: Testing Procedure Using *Average Probability*

We now discuss in detail how the probability function can be calculated in some popular classification algorithms. Decision tree learners (such as C4.5 [16]) uses

a divide-and-conquer strategy to group examples with the same feature values until it reaches the leaves of the tree where it cannot distinguish the examples any further. Suppose that n is the total number of examples in a leaf node and n_i is the number of examples with class label ℓ_i in the same leaf. $p(\ell_i|x) = \frac{n_i}{n}$ is the probability that x is an instance of class ℓ_i. We calculate probability in a similar way for decision rule classifiers, e.g. RIPPER [4]. For naive Bayes classifiers (one such implementation (NBC) is publicly available at http://fuzzy.cs.uni-magdeburg.de/∼borgelt/software.html), we assume that a_j's are the feature values of x, $p(\ell_i)$ is the prior probability or frequency of class ℓ_i in the training data, and $p(a_j|\ell_i)$ is the prior probability to observe feature attribute value a_j given class label ℓ_i, then the score $n(\ell_i|x)$ for class label ℓ_i is: $n(\ell_i|x) = p(\ell_i)\prod_j p(a_j|\ell_i)$ and the probability is calculated on the basis of $n(\ell_i|x)$ as $p(\ell_i|x) = \frac{n(\ell_i|x)}{\sum_k n(\ell_k|x)}$.

An Illustrative Example We use a simplified example to demonstrate our framework. Consider an ad-hoc network organized by two nodes. Packets can only be delivered from one end to the other if they are within each other's transmission range. We define the following three features. 1) Is the *other node reachable*? 2) Is there any *packet delivered during last 5 seconds*, and 3) is there any *packet cached for delivery during last 5 seconds*? For simplicity, we assume all features are binary valued, i.e., either **True** or **False**. All normal events are enumerated in Table 1. We then construct three sub-models with respect to each feature, shown in Table 2. The "Probability" columns here denote the probability associated with predicted classes. We use an illustrative classifier in this example that works as follows. If only one class is seen in all normal events where other non-labeled features have been assigned with a particular set of values, the single class is selected as the predicted class with the associated probability of 1.0. If both classes are seen, label *True* is always selected with the associated probability of 0.5. If none are seen (which means the combination of the other two feature values never appears in normal data), we select the label which appears more in other rules, with the associated probability of 0.5. To compute the probability for the true class, we use the probability associated with the predicted class if it matches, or one minus the associated probability if it does not. For example, the situation when a route is viable, no data is cached and no data is therefore delivered is a normal case. We apply the corresponding feature vector, $\{True, False, False\}$, into all three sub-models and all match the predicted classes. But the first sub-model with respect to the feature "Reachable?" has a probability of 0.5 only, which is obvious since when no data is delivered, it does not matter whether the route is up or not. The average match count is then calculated as $(1 + 1 + 1)/3 = 1$, and the average probability is $(1 + 1 + 0.5)/3 = 0.83$. Suppose we use a threshold of 0.5, then both values tell that the event is normal, which is right. A complete list of the *average match counts* and *average probabilities* for all possible events (both normal and abnormal) is shown in Table 3. Note that we use abbreviations here where AMC stands for the *Average match count*, and AP is the *Average probability*. The results clearly show that given a threshold of 0.5, both Algorithm 2 and 3 work well to separate normal and abnormal events, while Algorithm 3 works better as it achieves perfect accuracy (Algorithm 2 has one false alarm with the input $\{False, False, False\}$).

Table 1. Normal events in the 2-node network example

Reachable?	Delivered?	Cached?
True	True	True
True	False	False
False	False	True
False	False	False

Table 2. Cross-Feature models

Delivered?	Cached?	Reachable?	Probability
True	True	True	1.0
False	False	True	0.5
False	True	False	1.0
True	False	True	0.5

(a) Sub-model with respect to 'Reachable?'

Reachable?	Cached?	Delivered?	Probability
True	True	True	1.0
True	False	False	1.0
False	True	False	1.0
False	False	False	1.0

(b) Sub-model with respect to 'Delivered?'

Reachable?	Delivered?	Cached?	Probability
True	True	True	1.0
True	False	False	1.0
False	False	True	0.5
False	True	True	0.5

(c) Sub-model with respect to 'Cached?'

Table 3. Outcome from both normal and abnormal events

Reachable?	Delivered?	Cached?	Class	AMC	AP
True	True	True	Normal	1	1
True	False	False	Normal	1	0.83
False	False	True	Normal	1	0.83
False	False	False	Normal	0.33	0.67
True	True	False	Abnormal	0.33	0.17
True	False	True	Abnormal	0	0
False	True	True	Abnormal	0.33	0.17
False	True	False	Abnormal	0	0.33

4. Experimental Studies

In order to study how our data mining framework can be used to construct anomaly detection models for MANET routing, we have conducted the following simulation experiments.

4.1. Experiment Set-up

We use the Network Simulator ns-2 to run MANET simulations. As a simulation project developed in ISI/USC, ns-2 is one of the most widely used wired and wireless network simulators nowadays.

Parameter Selection We apply the random way-point model in ns-2 to emulate node mobility patterns with a topology of 1000m by 1000m. We use both TCP and UDP/CBR (Constant Bit Rate) as underlying transport protocols, and different protocols are used separately in different experiments. The maximum number of connections is set to be 100, traffic rate is 0.25 packets per second, the pause time between movements is 10s and the maximum movement speed is 20.0m/s. These settings are typical adhoc settings with adequate mobility and data load overhead, and are used throughout our experiments. We use one running trace of normal data as training set. For evaluation purposes, we use several other traces with normal data only, and a few traces composed with *black hole* and *packet dropping* attacks, started at 2500s and 5000s respectively. All traces have a run time of 10000 seconds with route statistics logged every 5 seconds (see Table 4).

Feature Construction The features (used in classification) constructed in our experiments belong to two categories, non-traffic related and traffic related. All non-traffic related features are detailed in Table 4 and the meaning of each feature is further explained in the "Notes" column. These features capture the basic view of network topology and route fabric update frequency. They are calculated based on the scenario and mobility scripts and the trace log file.

All traffic related features are collected under the following considerations. Packets come from different layers and different sources. For example, it can be a TCP data packet delivered from the originator where the feature is collected. It can also be a route control message packet (for instance, a `ROUTE REQUEST` message, used in AODV and DSR), which is being forwarded at the observed node. We can then define the first two aspects of a traffic feature as, *packet type*, which can be data specific and route specific (including different route messages used in AODV and DSR), and *flow direction*, which can take one of the following values, `received` (observed at destinations), `sent` (observed at sources), `forwarded` (observed at intermediate routers) or `dropped` (observed at routers where no route is available for the packet). We do, however, exclude the combination that data packets can be forwarded or dropped since it would never appear in a real ns-2 trace log. Routing protocols in MANET usually encapsulate data packets by adding particular headers with routing information at the source node and unpack them at the destination. Therefore all activities (including forwarding and dropping) during the transmission process only involve "route" packets. Also, we need to evaluate both short-term and long-term traffic patterns. In our experiments, we sample data in three predetermined *sampling periods*, 5 seconds, 1 minute and 15 minutes. Finally, for each traffic pattern, we choose two typical *statistics measures* widely used in literature, namely, the packet count and the standard deviation of inter-packet intervals. Overall, a traffic feature can be defined as a vector $<$ packet type, flow direction, sampling periods, statistics measures $>$. All dimensions and allowed values for each dimension are defined in Table 5. For instance, the feature to compute the standard deviation of inter-packet intervals of received `ROUTE REQUEST` packets every 5 seconds can be encoded as $< 2, 0, 0, 1 >$. Overall, we have $(6 \times 4 - 2) \times 3 \times 2 = 132$ traffic features, where 6, 4, 3, 2 are the number of packet types, flow directions, sampling periods and statistics measures, respectively.

For all continuous features or discrete features with infinite value space, we discretize them using a frequency-bucket scheme. We divide the value space of a continuous feature into a fixed number of continuous ranges (buckets), so that the frequencies of occurrences of feature values dropped in all buckets are equal. Then a continuous feature can be replaced by the index of its corresponding bucket. This approach guarantees that the chances of appearance of all possible labels (after discretization) in a feature are approximately the same. A pre-filtering process using a small random subset of normal vectors is necessary to retrieve the frequency distribution of all continuous features. In our experiments, we choose the bucket number to be 5.

Table 4. Topology and route related features

Features	Notes
time	ignored in classifi cation. Only for reference
velocity	node movement velocity (scalar)
route add count	routes newly added via route discovery
route removal count	stale routes being removed
route fi nd count	routes in cache with no need to re-discovery
route notice count	routes added via overhearing
route repair count	broken routes currently under repair
total route change	route change rate within the period
average route length	average length of active routes

Table 5. Traffic related features

Dimension	Values
Packet type	data, route (all), `ROUTE REQUEST`, `ROUTE REPLY`, `ROUTE ERROR` and `HELLO` messages
Flow direction	received, sent, forwarded and dropped
Sampling periods	5, 60 and 900 seconds
Statistics measures	count and standard deviation of inter-packet intervals

Intrusion Simulation We choose to implement the following intrusion scripts to study the problem of anomaly detection in MANET.

- Black hole attack

  It is a routing specific attack which is implemented by the following means. For DSR, a compromised host H_c broadcasts bogus `ROUTE REQUEST` messages with selected source and destination and a fake sequence number with maximum allowed value. The source route field in the fabricated `ROUTE REQUEST` specifies a one-hop route from the source to H_c as if the host were the immediate neighbor who forwarded the source's first `REQUEST`. Those messages are not harmful themselves except for the overhead due to flooding. The attack is really accomplished by the side effect that a neighbor overhearing a bogus `REQUEST` message would take the source route recorded in the message, reverse it, mistakenly assume the reversed source route could be a better route to the source, and override existing valid route(s) to the source by the faked route. If the attacker generates multiple bogus `REQUEST` messages where each message specifies a different source and all sources (except for the attacker itself) are covered, then all traffic flows, no matter where their destinations could be, will be forwarded to the compromised node. This is very similar in concept to an astronomical object, i.e., a region in space in which the pull of gravity is so strong that nothing can escape, as a corollary to Einstein's general theory of relativity.

  For AODV, we choose same destination as well as the source in each bogus message, which is allowed in this protocol. Similarly, the attack script fabricates a source sequence number to be the maximal allowed value, and claims that the compromised host is the next hop from the source node.

  One comment is that these attacks should be temporary as bogus route updates with maximum sequence number will be expired eventually. An attacker has to inject these bogus update messages periodically to defeat this. However, we observe that the current implementation doesn't recover completely even after the attack has completely stopped. The system still reports abnormal sometime even new updates with sequence numbers within normal ranges have already been accepted. The exact reason is still under our investigation.

- Selective packet dropping attack

  Under this traffic specific attack, packets are dropped based on its destination on the compromised host. The destination address can be specified as a parameter to the attack script.

For each type of intrusion, we don't actually turn on the intrusion behavior all the time as otherwise it could become an obvious target to be detected. Instead, we introduce a simple on-off model where intrusion sessions are inserted periodically. For the sake of simplicity, we assume the duration of each intrusion session and the gap between two adjacent intrusion sessions are the same. The value of duration is specified as a script parameter. We haven't yet fully investigated the detection performance when a different set of period values is used. For instance, some attacks such as Denial-of-Service, the attacker may not need to hide its identity at all, therefore the period can take a value as long as the whole running time (so that no off period actually takes place). The study on parameter selection remains our future work. We summarize the details of implemented intrusions and parameters in Table 6.

Table 6. Details of simulated MANET intrusions

Script	Description	Parameters
Black hole	Broadcast bogus routes to all nodes	duration
Selective Packet Dropping	Drop all packets to some specifi c node	duration destination

4.2. Experimental Results

With a combination of AODV vs. DSR and TCP vs. UDP/CBR, we have conducted all four scenarios in our experiments. In this paper, we only discuss in details the results on AODV with TCP. The results on other scenarios, i.e., are similar. In a longer version of this paper (which is available from the first author's homepage at http://www.cc.gatech.edu/∼yian), results on other scenarios

are also presented. Also, we choose several classifiers using different classification algorithms for evaluation purposes. These classifiers are C4.5, RIPPER, and NBC. Note that all results discussed in the paper are collected on one node only for brevity. Similar results and performance have been verified on other nodes of the simulated network throughout our experiments.

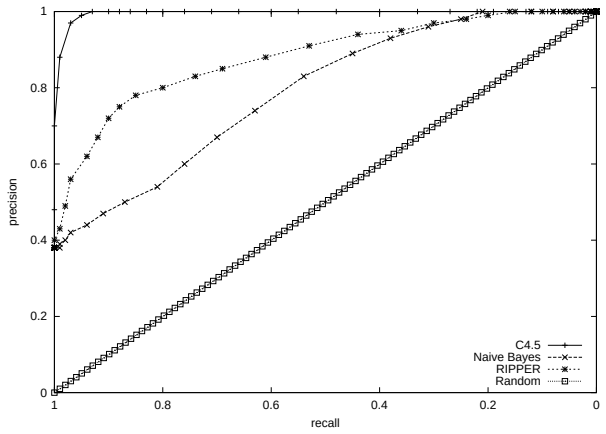

Figure 1. Recall-Precision curves using average probability

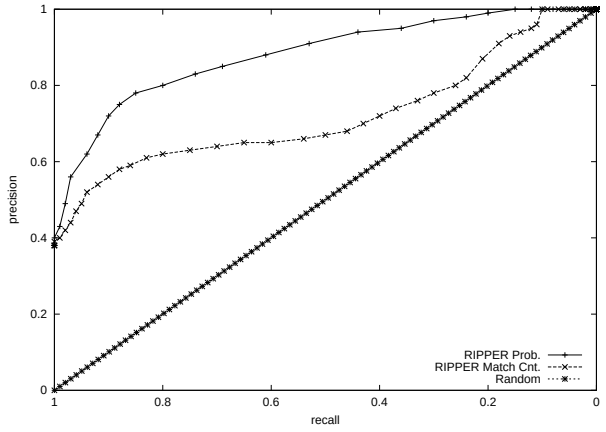

Figure 2. Average match count vs. average probability with RIPPER

As we suggested earlier, a decision threshold can be computed by calculating the lower bound of output values from normal events. We here show detection results when different values of threshold are used by using a recall-precision curve and explain how an optimal threshold values can be achieved empirically in this way.

Recall is a measure of the fraction of known positive examples that are correctly classified. Precision is a measure of the fraction of the classified positive examples that are truly positive. If I denotes intrusions and A denotes alarms (i.e., classified as positive), then recall rate is $p(A|I)$ and the precision rate is $p(I|A)$. The better (higher) a recall rate is, the more abnormal instances are captured. The better (higher) a precision rate is, the more chance that alarms are really intrusions. It is hard to achieve perfect results for both measures at the same time in practice. Some trade-off has to be made based on other criteria. In a recall-precision curve, the x-axis is the recall rate and the y-axis is the precision rate. Conventionally, we draw the x-axis from 1 to 0 so that the theoretically optimal point (i.e., both values of recall and precision are 1.0) appears in the top left corner. In our experiments, we obtain various operation points in the recall-precision space by varying decision thresholds. A larger (smaller) threshold implies more (less) examples are classified as positive, then the recall rate may go up (down) since more (less) anomalies have chance to be classified correctly. On the other hand, the precision rate may go down (up) since more (less) normal events are also classified as alarms. The 45-degree diagonal of the recall-precision curve is the result of "random guess", where anomalies and normal connections have equal probability to be classified as anomalies. The closer the curve follows the left border and then the top border, the more accurate a proposed method is. In practice, recall and precision carry different costs. The choice of decision threshold is to minimize the total loss.

The recall-precision curves with three classifiers (C4.5, RIPPER and NBC) are shown in Figure 1 based on average probability measures. We find out that results from the three classifiers are quite different. Quantitatively, we can use the area between a curve and the "random guess" diagonal line, or AUC (Area Under the Curve), to evaluate the accuracy degree of the corresponding classifier. Using the measure, C4.5 shows almost perfect performance as its curve is very close to the left and top borders, which is far much better than those of the other two classifiers. Experiments using average match counts are conducted which are not shown here due to space limit. We do, however, observe that RIPPER improves performance dramatically when we use the average probability instead of the average match count. The improvement is demonstrated in Figure 2. Similar effects do not seem to appear in the other two cases. In Figure 1, RIPPER can be seen as the second best classifier. The best performance point, which appears in the C4.5 curve, is (0.99, 0.97). Here a simplified criterion is used, i.e., best performance point occurs with the closest distance to the optimal point (1, 1).

We show the curves of average probability values from both normal and abnormal traces in Figure 3. Here only C4.5 results are shown. Since we use multiple traces in all test categories (normal and abnormal), the averaged outcome from each category is used. In the figure, we observe

that almost identical curves for both normal and abnormal traces are shown during the initial 2500 seconds because no intrusions start before 2500s in our set-up. After that, it can be easily identified that normal traces have an nearly flat curve, which implies that our work has successfully modelled most of the normal events. On the other hand, abnormal traces show an oscillating curve which ranges from 0.75 to 0.95, indicating that abnormal activities have been detected by significant difference from normal activities in our approach. At 5000s, the curve drops even further. It is an obvious sign that the amount of abnormal activities has become larger (which is true since a second intrusion is introduced at that time).

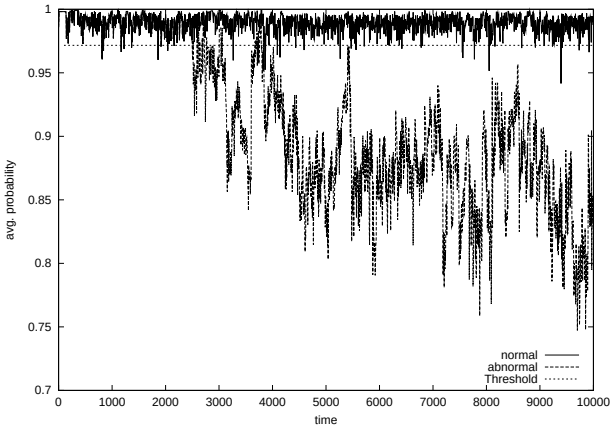

Figure 3. Average probability: normal vs. abnormal traces

We then plot the output average probability values using density distribution curves in Figure 4, which can help us to find out the optimal decision threshold. In the figure, decision threshold is represented by a vertical line. The right of the threshold line is considered normal. The other side is abnormal. Here we see the decision threshold is about 0.97 (based on the best performance point we have obtained from the recall-precision curve).

The results so far come from set-up with mixed intrusions from different categories (routing and traffic) in the same trace. Readers may wonder what would happen if a different set of intrusion combination is used. Figure 5 shows the detection results of two other intrusion sessions. One is composed of *black hole* attacks only, and the other is composed of *selective packet dropping* attacks only. Each trace consists of three intrusions (of same type) which start on 2500s, 5000s and 7500s respectively. Each individual intrusion lasts for 100 seconds. The result from the mixed intrusion scenario is also provided here. Different intrusion types show slightly different patterns but each of them can still be easily separated from normal traces by the threshold line. Areas under the normal curve while to the left of the

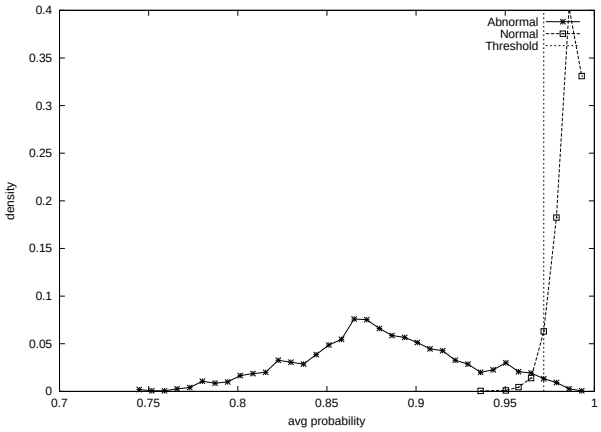

Figure 4. Average probability density distribution: normal vs. abnormal traces

threshold line (false alarms) and under the intrusive curves while to the right of the threshold line (anomalies mistakenly accepted) are all very small.

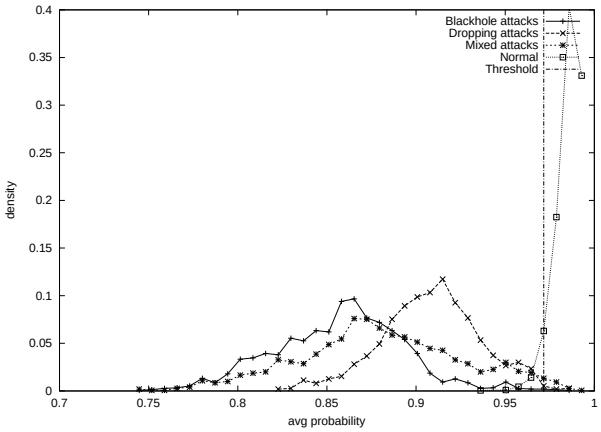

Figure 5. Average probability density distribution: different intrusion scenarios

5. Related Work

For intrusion prevention in MANET, general approaches such as key generation and management have been used in a distributed manner to ensure the authenticity and integrity of routing information. Binkley [2] reported experiments on authentication of MAC and IP layers. Hubaux et al. [9] proposed to use a PGP-like scheme to bootstrap trust relationships. Perrig et al. [15] studied the problem of authenticating broadcast in sensor networks. Their approach is to achieve asymmetry (which is required for authentication) through a delayed disclosure of symmetry keys. It re-

quires that the sensor nodes and the base station be time-synchronized. These prevention or proactive schemes are in general difficult to defend against internal attacks or passive attacks.

Researchers are also starting to study intrusion detection and response technique for MANET. For example, the CONFIDANT [3] system has a monitor on each mobile node for observations, reputation records for first-hand and trusted second-hand observations, trust records to control trust given to received warnings, and a path manager for nodes to adapt their behavior according to reputation. However, the protocol targets specifically at the unfairness problem and the major concern is that individual node may deny forwarding packets, which restricts the protocol from being used against more general attacks.

Early approaches [1] of anomaly detection used statistical measures of system features, e.g., CPU usage, to build normal profiles. Ghosh and Schwartzbard [6] proposed using a neural network to learn a profile of normality. To the best of our knowledge, we are the first to investigate how inter-feature correlations can help to build anomaly detection models.

6. Conclusion and Future Work

We present in this paper a new data mining approach based on cross-feature analysis for anomaly detection in MANET routing. The results show that this approach is very useful when strong inter-feature correlation can be extracted automatically in the normal system data. We find that our resulting model is fairly comprehensible by human experts who can further improve it.

Future Work Although the framework is studied in the background of MANET routing, we believe that it is a *general* anomaly detection approach for network intrusion problem as well as a few financial fraud detection problems where only normal data could be trusted. We are currently applying the framework to some other data sets. Initial experiments using credit card fraud detection have revealed promising results.

We are developing technologies to reduce computational cost, where a less number of models would be involved in the combination process and each of them could be simplified with a reduced feature set. We are currently studying approaches based on both correlation analysis and factor analysis.

References

[1] D. Anderson, T. Frivold, and A. Valdes. Next-generation intrusion detection expert system (NIDES): A summary. Technical Report SRI-CSL-95-07, Computer Science Laboratory, SRI International, Menlo Park, California, May 1995.

[2] J. Binkley and W. Trost. Authenticated ad hoc routing at the link layer for mobile systems. *Wireless Networks*, 7(2):139–145, 2001.

[3] S. Buchegger and J. L. Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403 – 410, Canary Islands, Spain, Jan. 2002. IEEE Computer Society.

[4] W. W. Cohen. Fast effective rule induction. In *Machine Learning: the 12th International Conference*, Lake Taho, CA, 1995. Morgan Kaufmann.

[5] K. Fall and e Varadhan. *The ns Manual (formerly ns Notes and Documentation)*, 2000. Online reference: http://www.isi.edu/nsnam/ns/ns-documentation.html.

[6] A. K. Ghosh and A. Schwartzbard. A study in using neural networks for anomaly and misuse detection. In *8th USENIX Security Symposium*, 1999.

[7] Z. Haas and M. R. Pearlman. The zone routing protocol (ZRP) for ad hoc networks. Internet draft draft-ietf-manet-zone-zrp-04.txt, expired 2003, July 2000.

[8] Y. Hu, A. Perrig, and D. B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking* (MobiCom 2002), Sept. 2002.

[9] J. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceeding of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Long Beach, CA, 2001.

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

[11] T. Lane and C. E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, 2(3):295–331, 1999.

[12] P. Papadimitratos and Z. J. Hass. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, San Antonio, TX, Jan. 2002.

[13] C. Perkins, E. M. Royer, S. R. Das, and M. K. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE Personal Communications Magazine special issue on Ad hoc Networking*, pages 16–28, Feb. 2001.

[14] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, Feb. 1999.

[15] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. SPINS: security protocols for sensor networks. In *Mobile Computing and Networking*, pages 189–199, 2001.

[16] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[17] F. Wang, B. Vetter, and S. F. Wu. Secure routing protocols: Theory and practice. Technical report, North Carolina State University, 2000.