INTRUSION DETECTION IN MOBILE AD HOC NETWORKS

A Dissertation

by

BO SUN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2004

Major Subject: Computer Science

INTRUSION DETECTION IN MOBILE AD HOC NETWORKS

A Dissertation

by

BO SUN

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

Udo W. Pooch (Chair of Committee)	
Riccardo Bettati (Member)	Rabi N. Mahapatra (Member)
Michael T. Longnecker (Member)	Valerie E. Taylor (Head of Department)

May 2004

Major Subject: Computer Science

ABSTRACT

Intrusion Detection in Mobile Ad Hoc Networks. (May 2004)
Bo Sun, B.S., Nanjing University of Posts & Telecommunications;
M.S., Beijing University of Posts & Telecommunications
Chair of Advisory Committee: Dr. Udo W. Pooch

Most existent protocols, applications and services for Mobile Ad Hoc NETworks (MANETs) assume a cooperative and friendly network environment and do not accommodate security. Therefore, Intrusion Detection Systems (IDSs), serving as the second line of defense for information systems, are indispensable for MANETs with high security requirements.

Central to the research described in this dissertation is the proposed two-level nonoverlapping Zone-Based Intrusion Detection System (ZBIDS) which fit the unique requirement of MANETs. First, in the low-level of ZBIDS, I propose an intrusion detection agent model and present a Markov Chain based anomaly detection algorithm. Local and trusted communication activities such as routing table related features are periodically selected and formatted with minimum errors from raw data. A Markov Chain based normal profile is then constructed to capture the temporal dependency among network activities and accommodate the dynamic nature of raw data. A local detection model aggregating abnormal behaviors is constructed to reflect recent subject activities in order to achieve low false positive ratio and high detection ratio. A set of criteria to tune parameters is developed and the performance trade-off is discussed.

Second, I present a nonoverlapping Zone-based framework to manage locally generated alerts from a wider area. An alert data model conformed to the Intru-

sion Detection Message Exchange Format (IDMEF) is presented to suit the needs of MANETs. Furthermore, an aggregation algorithm utilizing attribute similarity from alert messages is proposed to integrate security related information from a wider area. In this way, the gateway nodes of ZBIDS can reduce false positive ratio, improve detection ratio, and present more diagnostic information about the attack.

Third, MANET IDSs need to consider mobility impact and adjust their behavior dynamically. I first demonstrate that nodes' moving speed, a commonly used parameter in tuning IDS performance, is not an effective metric for the performance measurement of MANET IDSs. A new feature - link change rate - is then proposed as a unified metric for local MANET IDSs to adaptively select normal profiles . Different mobility models are utilized to evaluate the performance of the adaptive mechanisms.

To my parents, brother, and wife

ACKNOWLEDGMENTS

This work would never have been done without the generous help and support that I received from numerous people throughout my doctoral study at TAMU. I would like to take this opportunity to express my sincerest thanks to these individuals.

I would like first to thank my advisor, Dr. Udo Pooch, for providing invaluable guidance in my research. He helped me to choose this wonderful research topic when I saw no hope in my Ph.D. study. His broad knowledge and deep insights help me to choose the correct methodology to carry out this research. Special thanks go to Dr. Riccardo Bettati for giving me invaluable advice to my Ph.D. study. I would also like to thank Dr. Rabi Mahapatra and Dr. Michael Longnecker for serving as my committee members.

I am especially grateful to Dr. Kui Wu, whose close cooperation with me enables me to solve many details of this research. I also wish to acknowledge all my colleagues at Texas A&M University, especially Jian Chen, Jie Rong, Yong Guan, Yong Xiong, Xinwen Fu, Dan Cheng, Jun Zheng, Cheng Shao, for their insightful discussions on research.

Finally, I would like to express my profound gratitude to my parents and brother for their tremendous help and spiritual support. I owe a lot to my wife, Xiangping Li, for her love, patience and endurance. Her understanding and encouragement are my momentum to complete the Ph.D. study.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

In this chapter, we first illustrate why Intrusion Detection Systems (IDSs) are necessary when we deploy Mobile Ad hoc NETworks (MANETs) in reality. We then discuss why traditional wired IDSs are not applicable to MANETs and the research challenges when we design MANET IDSs. Finally, an overview of the dissertation will conclude this chapter.

A. Introduction and Motivation

Unlike conventional cellular wireless mobile networks that rely on extensive infrastructure to support mobility, MANETs do not need expensive base stations or wired infrastructure. The absence of a fixed infrastructure requires mobile hosts in MANETs to cooperate with each other for message transmissions. To form such a cooperative self-configurable environment, every mobile host is supposed to be a friendly node and is willing to relay messages for others to their ultimate destinations. Global trustworthiness in all network nodes is the main fundamental security assumption in MANETs.

However, this assumption is not always true in reality. The nature of MANETs makes them very vulnerable to malicious attacks ranging from passive eavesdropping to active interfering. Most routing protocols only focus on providing efficient route discovery and maintenance functionality and pay little attention to routing security. Very few of them specify security measures from the very beginning. The nature of MANETs makes them very vulnerable to malicious attacks compared to traditional

---

The journal model is *IEEE Transactions on Computers.*

wired networks, because of the use of wireless links, the low degree of physical security of the mobile nodes, the dynamic topology, the limited power supply and the absence of central management point [1]. Some environments (such as the military tactical operations) have very stringent requirements on security, which make the deployment of security-related technologies necessary.

Intrusion prevention measures, such as encryption and authentication, can be used in MANETs to reduce intrusions, but cannot eliminate them. For example, a physically captured node that carries the private keys may allow the defeat of the authentication safeguards. The history of security research has demonstrated that no matter how many intrusion prevention measures are used, there are always some weak points in the system. In a network with high security requirements, it is necessary to deploy intrusion detection techniques. MANET IDSs, serving as the second wall of defense to protect MANETs, should operate together with prevention mechanisms (authentication, encryption etc.) to guarantee an environment with high-secure requirements. They should complement and integrate with other MANET security measures to provide a high-survivability network.

However, most of today's Intrusion Detection Systems (IDSs) focus on wired networks. The dramatic differences between MANETs and wired networks make it inapplicable to apply traditional wired ID technologies directly to MANETs. MANET does not have a fixed infrastructure. While most of today's wired IDSs, which rely on real-time traffic parse, filter, format and analysis, usually monitor the traffic at switches, routers, and gateways. The lack of such traffic concentration point makes traditional wired IDSs inapplicable on MANET platforms. Each node can only use the *partial* and *localized* communication activities as the available audit traces. There are also some characteristics in MANET such as disconnected operations [2], which seldom exist in wired networks. What's more, each mobile node has limited resources

(such as limited wireless bandwidth, computation ability and energy supply, etc.), which means MANET IDSs should have the property to be lightweight. All of these imply the inapplicability of wired IDSs on the MANET platform. Furthermore, in MANETs, it is very difficult for IDSs to tell the validity of some operations. For example, the reason that one node sends out falsified routing information could be because this node is compromised, or because the link is broken due to the physical movement of the node. All these suggest that an IDS of a different architecture needs to be developed to be applicable on the MANET platform [1].

B. Research Challenges

It is very challenging to design an intrusion detection system for mobile ad-hoc networks. The lack of fixed infrastructures and concentration points make it difficult to collect audit data for the entire network. However, detection models only relying on *partial* and *localized* information are difficult to achieve desirable performance. We also need to consider the scarce MANET resources (such as limited wireless bandwidth, computation ability and energy supply, etc.) when we design the IDS framework for MANETs. What's more, mobility makes the distinction between normalcy and anomalcy obscure. It is more difficult to distinguish false alarms and real intrusions. For example, a node that sends out falsified routing information could be because it has been compromised, or because of its arbitrary movement [1].

In summary, the following lists the research challenges in designing a viable intrusion detection system for mobile ad-hoc networks:

- What is the good intrusion detection framework for mobile ad-hoc networks?
- What are the statistical security features that could be used to construct detection models?

- What are the appropriate approach to construct detection models?

C. Overview of the Dissertation

It is very difficult to design a once-for-all intrusion detection system. Instead, an incremental enhancement strategy may be more feasible. A secure protocol should at least include mechanisms against known attack types. In addition, it should provide a scheme to easily add new security features in the future. Due to the importance of MANET routing protocols, we focus on the detection of attacks targeted at MANET routing protocols. Specifically, we use the *routing disruption attack* as the threat model throughout this dissertation. The general methodology is: we first establish the local detection model for MANET routing activities, then develop framework to facilitate the cooperation of IDS agents. The whole dissertation is organized as follows.

In Chapter II, we summarize the related work which has been done in wired and wireless intrusion detection systems. This includes alert aggregation and correlation systems which build on existing IDS prototypes in order to improve detection performance. We further introduce a few important existing prototypes and detection algorithms for wired networks, and demonstrate that very few research efforts have been devoted to MANET IDSs. We detail the characteristics of MANETs, especially their routing protocols, and describe why they are particularly vulnerable to attacks. The understanding of MANET characteristics is necessary for us to develop a proper MANET IDS system. We also summarize the existing prevention mechanisms for MANETs because they need to integrate with MANET IDSs to provide a highly survivable network. Note that the research described in this dissertation only focuses on the detection part, although intrusion response component is necessary in the system.

In Chapter III, we propose a nonoverlapping Zone-Based Intrusion Detection System (ZBIDS) that fits the requirements of mobile ad-hoc networks. In the system aspect of ZBIDS, an IDS agent is attached to each node. The network is logically divided into nonoverlapping zones which enable these agents to cooperate with each other to perform the intrusion detection task. Each IDS agent performs the functionalities of low level ZBIDS. It utilizes the *local* and *trusted* information and runs independently to monitor the node's local activities for abnormal behaviors and broadcast the locally generated alerts inside the zone. We present an internal model of the IDS agent and describe a Markov Chain based anomaly detection algorithm to construct its local detection engine. The details of feature selection, data collection, data preprocess, Markov Chain construction, classifier construction and parameter tuning are described. A simple approach utilizing *relative entropy* is adopted to demonstrate the effectiveness of selected features. In the high level of ZBIDS, the gateway nodes (also called interzone nodes, those nodes which have physical connections to different zones) of each zone are responsible for aggregating and correlating the locally generated alerts inside the zone in order to make the final decisions. An algorithm is presented to aggregate the locally generated alerts and to further improve the performance of ZBIDS. In ZBIDS, only gateway nodes can utilize alerts to generate the alarms. In this dissertation, we use alerts to denote the potential security breaches identified by local IDS agents, while alarms are finalized decisions made by ZBIDS to indicate an intrusion. An alert format compatible with the Intrusion Detection Message Exchange Format (IDMEF) [3] is presented to facilitate the interoperability of IDS agents.

In Chapter IV, based on Parsec [4] and GloMosim [5], extensive simulation is carried out in order to demonstrate the effectiveness of ZBIDS. Under Random Waypoint model and use the Dynamic Source Routing protocol (DSR) [6] as the

exemplary routing protocol, different sets of data, i.e., training data, test data, and attack data, are collected at different mobility levels. False positive ratio, detection ratio, and Mean Time to First Alarm ($MTFA$) are computed to measure the performance of ZBIDS. Detailed analysis of simulation results is provided. Proposed aggregation algorithm is also simulated to illustrate its better performance compared to local IDS.

In Chapter V, we describe our initial efforts in constructing an *adaptive* local MANET IDSs. How to effectively integrate mobility impact into MANET IDSs and take adaptiveness into consideration is very important. In Chapter V, focusing on the protection of MANET routing protocols, we first demonstrate that node moving speed, a most commonly used parameter in measuring MANET performance, is not desirable with respect to the performance measurement of local MANET IDSs. Then we propose the usage of a new feature - the link change rate, which could not only act as a unified metric in measuring MANET IDS performance, but also be used to facilitate local MANET IDSs to select normal profiles *adaptively*. We utilize different mobility models, Random Waypoint Model and Random Drunken Model, to study the performance of our proposed *adaptive* mechanism at different mobility levels. Simulation results show that our proposed *adaptive* mechanisms could provide IDSs which are less dependent on mobility models and keep roughly the same performance compared to non-adaptive mechanisms in terms of false positive ratio, detection ratio and $MTFA$. Detailed analysis of simulation results is also provided.

Chapter VI concludes this dissertation and lists important future work. Because not many research efforts have been devoted to MANET IDSs, this research only provides the initial effort in constructing a viable MANET IDS. Based on the detailed methodology of building ZBIDS described in this dissertation, Chapter VI summarizes important future directions in this respect.

CHAPTER II

RELATED WORK

Intrusion detection research on MANETs requires the discussion of intrusion detection systems (IDSs), alert aggregation and correlation systems and mobile ad-hoc networks. As we have mentioned, extensive research efforts have been devoted to wired IDSs, however, there are few work devoted to wireless IDSs. In recent years, several wired aggregation and correlation systems have emerged in order to improve the performance of wired IDSs. To the best of my knowledge, there are no work devoted to alert aggregation on MANET platform prior to the work described in this dissertation.

In this chapter, we first introduce the background knowledge of intrusion detection systems. This will cover wired IDSs, wired aggregation and correlation techniques, and wireless IDSs. We then introduce mobile ad hoc networks and their routing protocols. The understandings of these are necessary for us to construct a suitable MANET IDSs.

A. Intrusion Detection

Intrusion detection is a security technology that attempts to identify individuals who are trying to break into and misuse a system without authorization and those who have legitimate access to the system but are abusing their privileges [7]. The system protected is used to denote an information system being monitored by an intrusion detection system. It can be a host or a network equipment, such as a server, a firewall, a router, or a corporate network, etc [8].

An intrusion detection system (IDS) is a computer system that dynamically monitors the system and user actions in the network and computer systems in order

to detect intrusions. Because an information system can suffer from various kinds of security vulnerabilities, it is both technically difficult and economically costly to build and maintain a system which is not susceptible to attacks. Experience teaches us never to rely on a single defensive line or technique. IDSs, by analyzing the system and user operations in search of activity undesirable and suspicious, can effectively monitor and protect against threats. IDSs have been widely regarded as being part of the solution to protect today's computer systems.

Research on IDSs began with a report by Anderson [9] followed by Denning's seminal paper [10], which lays the foundation for most of the current intrusion detection prototypes. Since then, many research efforts have been devoted to wired IDSs. Numerous detection techniques and architecture for host machines and wired networks have been proposed. A good taxonomy of wired IDSs is presented in [8].

With the rapid proliferation of wireless networks and mobile computing applications, new vulnerabilities that do not exist in wired networks have appeared. Security poses a serious challenge in deploying wireless networks in reality. However, the vast difference between wired and wireless networks make traditional intrusion detection techniques inapplicable. Wireless IDSs, emerging as a new research topic, aim at developing new architecture and mechanisms to protect the wireless networks.

1. Wired Intrusion Detection Systems

Focusing mainly on network traffic data and computer audit data, there are two general approaches to detecting intrusions: misuse based intrusion detection (also referred to as knowledge-based detection, or detection by appearance) and anomaly based intrusion detection (also referred to as behavior-based detection or detection by behavior). They are complementary to each other for intrusion detection.

a. Misuse Based Intrusion Detection Systems

Misuse based IDSs operate based on a database of known attack signatures and system vulnerabilities. When IDS analyzer identifies an activity matching a signature that is stored in the database, an alarm is triggered. The advantages of misuse based IDSs include that they may have very low false alarm ratio. The triggered alarms are meaningful because the attack signatures contain the diagnostic information about the cause of the alarm. Disadvantages include that its completeness is not good because the attack signature databases and system vulnerabilities need to be kept up-to-date. This is a tedious task because new attacks and system vulnerabilities are detected on a daily basis. Careful analysis of the vulnerabilities is also time-consuming. Misuse detection based IDSs also face the generalization issues because most of the knowledge of the attacks is focused on the different versions of operating systems and applications.

There are several approaches in misuse detection. They differ in the representation as well as the matching algorithm employed to detect intrusion patterns. Here we simply list the main approaches:

- Expert system: Expert systems provide strategies and mechanisms for processing facts regarding the state of a given environment, and derive logical inferences from these facts. Audit events and the security policy are mapped to the facts that are recorded and evaluated by the system. During the process of mapping, a semantic meaning is attached to increase the abstraction level of the audit data. The expert system contains a set of rules that describe the attacks. These rules are triggered when certain activities that can satisfy their conditions happen. The execution speed of the expert system shell is usually poor because all of the audit data need to import into the shell as facts. Therefore, expert

system based IDSs only exist in research prototypes, as performance is more important in commercial products.

Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) [11] is an extension of the Intrusion Detection Expert System (IDES) [12] [13] and Next Generation Intrusion Detection System (NIDES) [14] by SRI International. EMERALD uses a rule-based expert system component for misuse detection. A forward-chaining rule-based expert system development toolset called the Production-Based Expert System Toolset (P-BEST) [15] is utilized to develop a modern generic signature-analysis engine. A chain of rules is established utilizing P-BEST to form the signature database.

- Pattern Recognition: In this approach, the encoding of known intrusion signatures as patterns (e.g., strings, a sequence of events, etc.) are matched against the audit data. The incoming events are searched to match the patterns representing intrusion scenarios. This method allows a very efficient implementation. Therefore, they are commonly used in commercial tools, such as RealSecure of Internet Security Systems [16].

- Colored Petri Nets: In this method, the signatures of the intrusions are modeled as a number of different states, which form Colored Petri Nets (CPNs). It has more generalities to represent the signatures and make it easy to write complex intrusion scenarios. However, it is very computationally expensive to try to manifest the misbehavior in the audit trail. Intrusion Detection In Our Time (IDIOT) is the one example that uses CPNs [17].

- State transition analysis: In this approach, an intrusion is represented as a sequence of actions performed by an intruder that leads from the initial state

to the target compromised state. State transition diagrams identify the steps and the requirements of the penetration. The states that make up the intrusion form a simple chain that has to be traversed from the beginning to the end. It is a technique proposed by Porras and Kemmerer [18], which is implemented in Ustat - a real-time intrusion detection system for UNIX [19].

b. Anomaly Based Intrusion Detection Systems

Anomaly based IDSs assume that an intrusion can be detected by observing a deviation from normal or expected behavior of the systems or users. Normalcy is defined by the previously observed subject behavior, which is usually created during a training phase. The normal profile is later compared with the current activity. If a deviation is observed, IDS flag the unusual activity and generate an alarm. The advantages of anomaly detection based IDSs include that they might be complete to detect attacks, i.e., they can detect attempts that try to exploit new and unforeseen vulnerabilities. They are also less system-dependent. Disadvantages include that they may have very high false alarm ratio and are more difficult to configure because the comprehensive knowledge of the expected behavior of the system is required. They usually require a periodic online learning process in order to build the up-to-date normal behavior profile. Anomaly detection approach is harder to implement, which make them inappropriate for commercial use.

Several anomaly detection techniques exist and differ in the representation of a normal profile and the inference of a deviation from the normal profile. The main approaches used in anomaly detection are:

- Statistics: Statistical-based anomaly detection techniques build a statistical profile (e.g., statistical distribution) of subject normal activities from historic

data by measuring a number of variables over time. Examples of the variables are the login/logoff times, the time duration of one session, the number of packets transmitted in this session, etc.

In EMERALD [11], the statistical algorithms employ four classes of measures to track subject activities: categorical, continuous, intensity, and event distribution. The profile is subdivided into short and long-term elements. A short-term profile may characterize the recent activity of the system, while a long-term profile is slowly adapted to the changes of the system activity. Many traffic perspectives are used to profile TCP/IP streams [11]. For example, all ICMP exchanges are parsed to analyze ICMP-specific transactions. The application-layer sessions from specific internal hosts to specific external hosts are analyzed for specific applications.

- Neural networks: The use of neural networks in IDS consists of three steps: learning the normal pattern of the system by collecting the training data; training the neural networks to identify the subject; applying the output of the neural networks to the observed activity to identify intrusions. Neural networks are computationally intensive, so they are not widely used in IDSs. Hyperview [20] is an example IDS that uses neural networks.

There are other anomaly detection techniques: anomaly detection techniques based on immunology [21] capture a large set of event sequences as the normal profile from historic data of subject normal activities, and use either negative selection or positive selection algorithms to detect the difference of incoming event sequences from event sequences in the normal profile [22]. Expert systems can also be used to implement anomaly detection techniques [13]. The IDSs can study the activities of the target system to form a set of rules to describe its normal behavior. Lee *et al.*

proposed to use datamining approach to construct intrusion detection models [23]. Anomaly detection techniques utilizing Chi-square test are also introduced in [24] and [25].

There are also anomaly detection techniques that use a first-order or high-order Markov model of event transitions to represent a normal profile [26] [27] [28] [29]. In [26], utilizing a Markov Chain model, Jha *et al.* proposed a general framework for constructing anomaly detectors. We modify it to fit MANET requirements. We define the *from_state* as the previous w ordered values of the categorized statistical measure and the *to_state* as the current statistical feature value. That is, we define the *from_state* as $\{X_i, X_{i+1}, \ldots, X_{i+w-1}\}$, and *to_state* as $\{X_{i+w}\}$ (w is a parameter that characterizes the Markov Chain). Therefore, we can consider the routing changes as a random process with stationary transition probabilities. We use the VQ algorithm [30] to preprocess the data and introduce "rare symbol" to construct a Markov Chain which would lead to better results in our environment. *Locality frame* is also used in the classifier construction. We also adopt a different approach to tune the parameters. Details of our methodology are described in Chapter III.

Besides misuse detection and anomaly detection, there is a new class of detection algorithm: specification-based techniques [31]. It combines the advantages of misuse detection and anomaly detection techniques. They detect attacks as deviations from a normal profile. Their approaches are based on manually developed specifications, thus avoiding the high rate of false alarms. However, the development of detailed specifications can be time-consuming.

2. Wireless Intrusion Detection Systems

Relatively few research efforts have been devoted to wireless IDSs. In [32], Kachirski *at al.* proposed a distributed intrusion detection system for ad hoc wireless networks

based on mobile agent technology. In [33], Samfat *et al.* proposed an Intrusion Detection Architecture for Mobile Networks (IDAMN). Its main functionality is to track and detect mobile intruders in real time. IDAMN includes two algorithms which model the behavior of users in terms of both telephony activities and migration patterns. In [34], a routing misbehavior in mobile ad hoc networks is identified: a node may misbehave by agreeing to forward packets and then failing to do so, because it is overloaded, selfish, malicious, or broken. The authors proposed to install extra facilities, *watchdog* and *pathrater*, to identify routing misbehavior in MANETs.

In the pioneer work of wireless intrusion detection research, Yongguang *et al.* [1] proposed a general intrusion detection and intrusion response architecture for MANETs. An agent is attached to each mobile node, and each node in the network participates in the intrusion detection and response. A majority-based distributed intrusion detection approach is proposed to facilitate the cooperation of neighboring nodes. Many of our ideas benefit from the research described in [1] [35]. In [36], a new data mining method that performs the "cross-feature" analysis to capture the inter-feature correlation patterns of MANET normal traffic is introduced to construct the normal profile. They focused on techniques for automatically constructing anomaly detection methods that are capable of detecting new attacks.

B. Wired Aggregation and Correlation Systems

Alert aggregation and correlation techniques are important to overcome the shortcomings manifested by IDSs. These shortcomings include the overwhelming alerts (*alert flooding*) generated by IDSs, high false positive ratio of the generated alerts, and the poor diagnosis information provided by the alerts. Existing alert aggregation and correlation systems have demonstrated promising approaches to analyze alerts

and generate more global and synthetic alerts.

Several alert aggregation and correlation techniques [37] [38] [39] [40] [41] [42] have been proposed to facilitate the analysis of intrusions. Based on the abundant attack scenarios, these approaches try to find the causal relationships between alerts and reveal the attack strategies. In [37], an aggregation and correlation component is built in Tivoli Enterprise Console. Cuppens *et al.* [38] [39] use *Lambda* language to specify attack scenarios and use Prolog predicates to correlate alerts based on IDMEF data model. In [40], a probabilistic method is used to correlate alerts using the attribute similarity among their features. Ning *et al.* [41] develop three utilities to facilitate the analysis of large sets of correlated alerts. In [42], a formal data model called $M2D2$ is proposed in order to make full use of the available information. The effectiveness of the proposed aggregation and correlation algorithms depends heavily on the information provided by the individual IDS.

C. Mobile Ad-Hoc Networks

Unlike conventional cellular wireless mobile networks that rely on extensive infrastructure to support mobility, a wireless Mobile Ad hoc NETwork (MANET) does not need expensive base stations or wired infrastructure. Nodes within the radio range of each other can communicate directly over the wireless links, while those that are far apart use other nodes as relays. In MANETs, each host must act as a router since routes are mostly *multihop*. Nodes in such a network move arbitrarily, thus the network topology changes frequently and unpredictably. Moreover, the wireless channel bandwidth is limited, and the mobile nodes operate on the constrained battery power which will eventually be exhausted.

Extensive research efforts have been devoted to various issues related to MANETS.

Because this research focuses on the protection of MANET routing protocols, here we briefly describe existing MANET routing protocols.

1. Routing Protocols of Mobile Ad-Hoc Networks

Many routing protocols have been proposed for MANETs. In general, these protocols could be divided into three categories: *proactive*, *reactive*, and *hybrid*. *Proactive* routing protocols (such as Destination-Sequenced Distance Vector routing protocol (DSDV) [43] and the Wireless Routing Protocol (WRP) [44]) waste limited bandwidth by continuously maintaining the complete routing information about the whole network. They react to topology changes, even if there is no traffic. They are also called *table-driven* methods. The protocols in this area differ in the number of tables maintained, the information each table contains as well as the details of how they are updated. *Reactive* routing protocols (such as Ad hoc On-demand Distance Vector routing protocol (AODV) [45], the Temporally Ordered Routing Algorithm (TORA) [46], and the Dynamic Source Routing protocol (DSR) [6]) are based on demand for data transmission. They can significantly reduce the routing overhead when the traffic is lightweight and the topology changes less dramatically, since they do not need to periodically update route information and do not need to find and maintain the routes when there is no traffic. The differences among *reactive* routing protocols lie in the implementation of the path discovery mechanism and optimizations to it. *Hybrid* methods combine *proactive* and *reactive* methods to find efficient routes. ZHLS [47] is one example of *hybrid* routing protocols. In ZHLS, the whole network is divided into nonoverlapping zones. ZHLS is proactive if the traffic destination is within the same zone of the source. It is reactive because a location search is needed to find the zone ID of the destination.

Fig. 1 is a categorization of existing routing protocols in MANETs. In the figure,

solid lines represent direct descendants while dotted lines depict logical descendants. Since new routing protocols are always being proposed for MANETs, we do not expect to include all of them here.

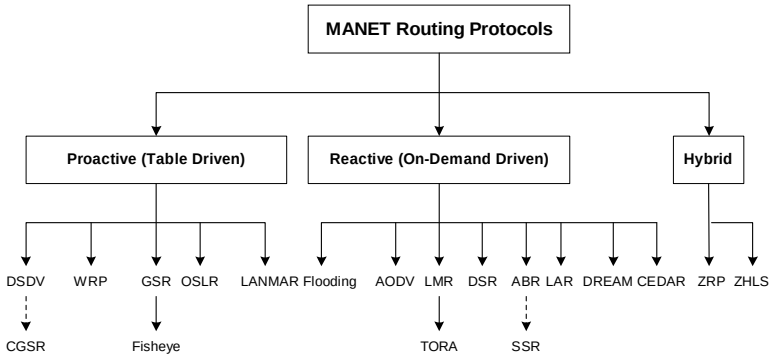


Fig. 1. A Classification of MANET Routing Protocols.

2. Prevention Mechanisms in MANETs

MANET IDSs can only provide one layer of defense for MANETs. They should complement existing prevention techniques in order to provide a highly survivable system. Considerable research [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] has been devoted to MANET prevention mechanisms, especially focused on the routing layer. In order to better understand the role of IDSs in MANETs, we briefly summarize the prevention mechanisms.

Zhou and Haas [50], Dahill *et al.* [51] proposed to use asymmetric cryptography to secure on demand MANET routing protocols. However, its potential intensive computation may make the nodes in the network unable to verify the signatures quickly enough. Hu, Johnson and Perrig [52] proposed to use hash chains to secure DSDV and the use of Ariadne [58] with TESLA, an efficient broadcast authentication scheme that requires loose time synchronization. In [48], the authors proposed

Secure Routing Protocol (SRP) based on the assumption that there exists a *security association* and a *shared secret* key between the source node and the destination node. In [53], a new routing technique called Security-Aware ad hoc Routing (SAR) was introduced. It assumes a pre-established trust hierarchy and the network is classified into different trust and privilege levels. Nodes at each trust levels share symmetric encryption and decryption keys.

There are also some works devoted to protecting MANETs against specific attacks. In [59], Hu *et al.* introduced a *wormhole attack*, in which the attacker records a packet, or individual bits from a packet, at one location in the network, tunnels the packet to another location, and replays it there. They also introduced the general mechanism of *packet leashes* - *geographic leashes* and *temporal leashes* to detect wormhole attacks. In [60], *rushing attack* is introduced, and *Rushing Attack Prevention* (RAP) is developed to defend against this new attack.

CHAPTER III

ZONE-BASED INTRUSION DETECTION SYSTEMS

Intrusion Detection Systems (IDSs) are necessary in Mobile Ad-hoc NETworks (MANETs) with high-survivability requirements. Serving as the second wall of defense to protect MANETs, they complement intrusion prevention techniques to tackle the exploitable weaknesses of the system. Our objective is to design a suitable intrusion detection system which could meet the requirements of MANETs.

In this chapter, we describe a nonoverlapping Zone-Based Intrusion Detection System (ZBIDS). Section A provides the assumptions and the network model. Section B briefly introduces the Dynamic Source Routing (DSR) [6] protocol which is used as the example routing protocol throughout this research. It then details the threat model - the routing disruption attack. Section C presents the nonoverlapping Zone-Based Intrusion Detection System (ZBIDS), which mainly consists of two parts: the local intrusion detection agent and the nonoverlapping zone based framework. Section D provides the detailed descriptions of the local IDS agent. The functionality of each module is specified. Section E illustrates a Markov Chain based anomaly detection algorithm. Detailed procedures from feature selection, data preprocess, Markov Chain model construction, and the classifier construction are provided In order to achieve desirable performances, an approach to tune the parameters of the detection model is also presented. Section F depicts an aggregation algorithm used in ZBIDS. It works together with the local detection agent to form a complete intrusion detection system. An alert class hierarchy used by ZBIDS is also depicted. The proposed alert class hierarchy conforms to the Intrusion Detection Message Exchange Format (IDMEF) [3], which could facilitate the interoperability with other IDS systems. Section H concludes this chapter.

A. Assumptions and Network Model

1. Assumptions

All security systems must rely on some specific assumptions to guarantee their effectiveness. Our ZBIDS system is valid under the following assumptions.

We assume that the network can be divided into nonoverlapping zones. For example, each node can utilize a Global Positioning System or other methods [47] to find its physical location and determine its zone identity by mapping its physical location to a predefined zone map. The partitioning of the network could be based on simple geographic partitioning or other clustering algorithms [61]. We also assume that the zone partitioning mechanism is accurate and safe.

This research focuses on the protection of MANETs. Preventing and detecting attacks aimed at IDS itself will be another challenging research topic and is beyond the discussion of this dissertation. We assume the local IDS agent is tamper resistant. There are many software tamper resistance techniques [62] that are very hard to crack. Under these techniques, the attacker will not be able to reverse engineer any secrets from the "good" agents. The secret embedded in the software could be prevented from being extracted by the attacker. Under this assumption, we do not need to consider the security issues of the IDS agent itself. In addition, we assume that information exchange between IDS agents cannot be forged by an attacker. This excludes the possibility that some compromised node actively generates falsified alerts to disrupt the correct execution of the aggregation algorithm. When the local IDS is not tamper resistant, the compromised IDS has no incentive to send reports because this may result in the detection of attackers. Also, the alerts provided by noncompromised IDSs could dominate the information that gateway nodes collect. This could still enable the correct execution of the aggregation algorithm described

in Section F.

We assume that when initiating the attack, the attacker can use a fake address but does not change it dynamically. If an attacker changes its address quite often, a neighboring monitor mechanism can identify this misbehavior effectively.

Wireless communication is fundamentally untrusted. We do not place any trust assumptions on the communication infrastructure. We do not assume nodes using trusted hardware either. This model is still impractical in many situations and security problems under this assumption are much simpler. The network can be secured through a network-wide shared secret key for all message encryption and authentication.

2. Network Model

We model the network as an undirected graph G. A graph G = (V, E) consists of a set of n nodes (vertices) and a set of m node pairs (edges). The set of nodes, denoted by $V = \{1, 2, \ldots, n\}$, represents the network-enabled ad hoc devices; the set of edges, denoted by E, represents the wireless communication links. Each link (i, j) is bidirectional and connects node i and j. Link (i, j) is removed when the distance between node i and j is greater than the radio transmission range, while a new link is formed when their distance is less than or equal to the radio transmission range. The topology of G is constantly changing, as is the set of E. In this model, the neighbor set of a node v is defined as a set of those nodes that have links to v.

B. Threat Model

Since routing protocols are the cornerstone of MANETs, this research will focus on the detection of attacks targeted at MANET routing protocols, more specifically

on detecting one of the most important active attacks: *routing disruption* attacks. *Routing disruption attacks* are particularly harmful to the whole network. It is deemed as one of the most vicious attacks and has been studied broadly by other researchers. In this section, we use DSR as the exemplary routing protocol to model the behavior of the routing disruption attack.

1. Basic Operations of DSR

We use the Dynamic Source Routing (DSR) protocol [6] as our exemplary routing protocol throughout this research due to its popularity in MANET community. The DSR routing protocol also defines a number of optimizations (such as preventing route reply storm, path state and flow state mechanisms, piggybacking on route discoveries and gratuitous route errors etc.). For simplicity, we only use a basic version of DSR without optimizations throughout this research. With modest revision, however, our proposed ZBIDS is applicable to DSR with more complicated functionalities. Since routing protocols such as DSR have not been standardized and no routing protocol seems to be an obvious winner for ad hoc networks in a short time, we leave the work of expanding the proposed intrusion detection scheme as further work.

DSR uses the source routing approach (every data packet carries the whole path information in its header) to forward packets. Before a source node sends data packets, it must know the total path to the destination. Otherwise, it will initiate a route discovery procedure by flooding a Route REQuest (RREQ) message. The RREQ message carries the sequence of hops it passed through in the message header. Any nodes that have received the same RREQ message will not broadcast it again. Once an RREQ message reaches the destination node, the destination node will reply with a Route REPly (RREP) packet to the source. The RREP packet will carry the path information obtained from the RREQ packet. When the RREP packet traverses

backward to the source, the source and all traversed nodes will know the route to the destination. Each node uses a route cache to record the complete route to desired destinations. Route failure is detected by the failure of message transmissions. Such a failure will initiate a route error message to the source. When the source and the intermediate nodes receive the error message, they will erase all the paths that use the broken link from their route cache.

2. Routing Disruption Attack

Fig. 2 illustrates one example of the *routing disruption* attacks.

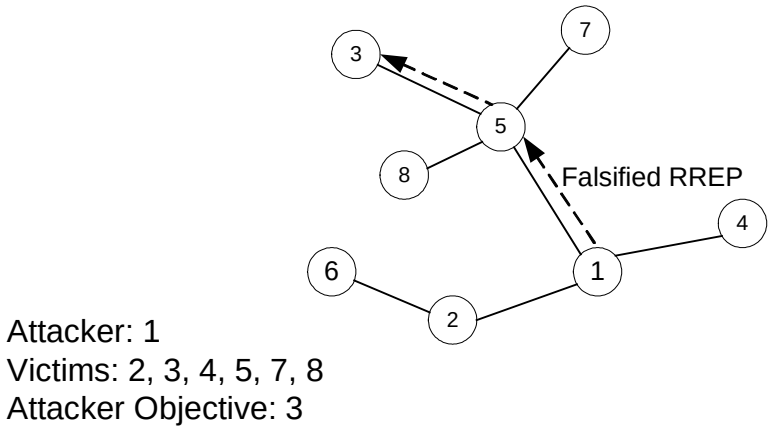


Fig. 2. An Example of the Routing Disruption Attack.

In the example, node 1 is the attacker and node 3 is the attack objective. Node 1 actively sends randomly-constructed, falsified RREP (Routing REPly) packets to node 3. The purpose is to effectively disrupt the routing logic of the victims, or even the whole network. The attacker may also try to induce the victim to form a short path to it, forming a routing black-hole. Since DSR uses source routing, the randomly constructed RREP needs to contain the path $1 \rightarrow 5 \rightarrow 3$ in order to reach

the destination. To do so, the attacker may initiate a *route discovery* procedure first to node 3, and based on the received RREP packet, the attacker can obtain a valid path. Thus, the path contained in the randomly constructed RREP may look like: $\{2, 4, 9, 7, 1, 5, 3\}$. Although the part $\{2, 4, 9, 7\}$ are randomly constructed, the valid partial path $\{1, 5, 3\}$ can guarantee that this fake RREP can reach node 3. When this fake RREP is unicasted to node 3, all nodes along the path (node 5) and their neighbors (for example, node $2, 4, 7, 8$) become victims due to the wireless broadcast nature and the enabled promiscuous-listening mode in the nodes. All victims will change their routing caches according to the newly received or promiscuously heard RREP packets. The victims could further disseminate this falsified routing information because it is assumed to reflect the current network topology. This attack can have a serious negative impact on the routing logic of the whole network.

The occurrence of this type of attack does not limit to the normal *route discovery* procedure. One obvious difference in the *routing disruption* attack between wired networks and MANETs is that: when a node moves, it is hard for the node to be targeted all the time. In the above example, the movement of node 3 may lead to the link break between node 5 and 3. Therefore node 3 may fail to receive falsified RREPs from node 1. This will lead to the phenomenon that a node is "partially" victimized during the whole intrusion session and it becomes more obvious with the increase of mobility.

Because mobility is arbitrary, it is very difficult to establish a mathematical model to characterize this kind of attack. One important assumption of intrusion detection is that normal and intrusive behaviors are distinct. If the attacker only sends one or two falsified routing control packets, it is very difficult for the victim to tell whether these falsified routing control packets are caused by mobility induced errors or generated by attackers, based only on local communication activities. Also,

sending only a few falsified routing control packets may not cause a serious impact on the whole network, considering the periodic route cache refreshment mechanism. What's more, it has been demonstrated that mobility can be utilized to propose new mechanisms to enhance security [63], thus reducing the attacker's ability to perform light attacks. Therefore, in the context of intrusion detection, we assume that an attacker has to send many falsified routing control packets in order to effectively disrupt the routing logic of the network.

C. Zone-Based Intrusion Detection System (ZBIDS)

In this section, we detail our proposed intrusion detection system - ZBIDS. From the system aspect, we attach an IDS agent to each mobile node. These IDS agents run independently and monitor local activities to detect abnormal behaviors. We choose to implement an anomaly detection algorithm because it is expected that more types of attacks will be launched against MANETs in the future. It is also difficult to obtain the complete trace of the attacks, which are often required in designing a misuse detection algorithm.

We logically divide the network into nonoverlapping zones to manage the locally generated alerts. By integrating the network information from a wider area, this management framework could reduce false positive ratio and improve detection ratio.

Therefore, the description of ZBIDS mainly consists of two parts: the overall network framework and the internal conceptual model of each IDS agent.

1. ZBIDS Framework

We adopt a zone-based intrusion detection framework because of the following considerations:

- Due to the dynamic nature of MANETs, *alert flooding* is expected in such an environment. Attacks are likely to generate multiple related alerts. By creating some alert concentration points, we can logically group related alerts together and reduce the false alarms generated for various reasons.

- Flat architecture is undesirable in managing the alerts. When the network becomes very large, scalability will be a serious problem. It is also unrealistic to have a centralized console in MANETs to manage all of the alerts because of the complicated mobility management and the issue of network reliability caused by the single point of failure.

A problem with a hierarchical approach in MANETs, however, is the cost of maintaining the hierarchy in face of mobility. When mobility is high, the introduction of the message overhead to create and maintain the hierarchy is unbearable.

We thus adopt a nonoverlapping zone-based framework because the communication overhead for creating and maintaining the topology is small [47]. It also requires little mobility management efforts. Actually, ZBIDS requires few extra control messages propagated within the zone in order to maintain the framework. Nevertheless, the selection of the zone size is critical and depends on factors such as node mobility, network density, transmission power and propagation characteristics, etc. The zone size should be neither too large nor too small. Large zone size compromises the advantage of using the hierarchical structure since the broadcast alerts may involve large communication overhead. Likewise, if the zone size is too small, the alert management nodes cannot collect enough information for aggregation.

The formation and the maintenance of zones are beyond the research topic in this dissertation. In a simple approach, the zones can be obtained based on geographic partitioning. Based on network connectivity, each node can be classified into one of

two categories: the interzone node (also called the gateway node) and the intrazone node. With the availability of GPS, it is possible for a mobile host to know its physical location. It can then determine its zone ID by mapping its physical location to a zone map, which has to be worked out at the design phase. By some locally broadcast mechanism (*Hello* messages, e.g.), each node can know the information of its neighbors. Therefore it can determine whether it is an interzone node or intrazone node. A node may change its role over time due to mobility. An example of ZBIDS is depicted in Fig. 3.

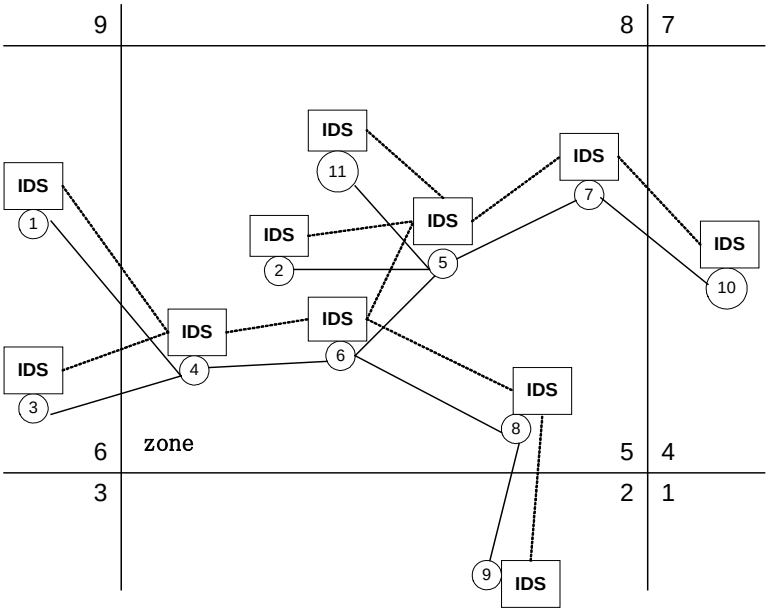


Fig. 3. The Zone Based IDS Framework for Mobile Ad Hoc Networks.

In Fig. 3, nodes 4, 7 and 8 are the gateway nodes of zone 5. Each mobile node is attached an agent, and all of these agents collaboratively perform the intrusion detection task. Each IDS agent runs independently to monitor its system activities, such as the user behavior, system behavior, radio communication activities, etc. and perform intrusion detection tasks locally. Intrazone nodes will report their locally

generated alerts to the gateway nodes in the same zone, and the gateway nodes will aggregate and correlate the received alerts. Gateway nodes in neighboring zones can further collaborate in order to perform intrusion detection tasks in a wider area.

Zhang *et al.* proposed an intrusion detection architecture [1], in which, only neighboring nodes can collaboratively cooperate. When the network becomes very large, scalability will become a serious problem. In order to solve this problem, we adopt the zone-based architecture and introduce the concept of intrazone and interzone nodes in MANET IDSs. There may exist many gateway nodes in a zone, thus avoiding the issue of single point of failure.

Intrusion detection must necessarily be deployed in various layers of networks. Certain attacks may be detected much earlier in the application layer, because it contains richer semantic information than the lower layer. For example, for a denial-of-service attack, the application layer may detect very quickly that a large number of incoming service connections have no actual operations; whereas the lower layers, which rely on information about the amount of network traffic (or the number of channel requests), may take longer to recognize the unusually high volume.

This research focuses on the attacks targeted at the routing layer, thus our IDS locates in the routing layer. It obtains data from routing caches to construct the classifier. Because of the distributed nature of ZBIDS, the communications among the IDS agents may rely on the underlying routing protocols.

In this research, we do not consider the following issues:

- We do not consider attacks targeted at the physical layer and Medium Access Control layer. We focus on the routing attack and use it as the threat model to develop our whole system. However, ZBIDS is general and can accommodate attacks targeted at other layers easily.

- We do not consider the formation and maintenance of zones. That is, we assume that the network can be divided into nonoverlapping zones and the zone partitioning mechanism is accurate and safe.

- We focus on the protection of MANETs. Preventing and detecting attacks aimed at IDS itself will be another challenging research topic and is beyond the discussion of this research. Therefore, we do not consider the security issues of IDS agent itself.

In the following sections, we describe the local detection model and aggregation algorithm used by the ZBIDS in detail.

D. Internal Model of the IDS Agent

The internal model of the IDS agent can be divided into the following components: the data collection module, the detection engine, the local aggregation and correlation engine (LACE), the global aggregation and correlation engine (GACE), and the intrusion response module. A diagram is given in Fig. 4.

The data collection module is mainly responsible for collecting the security related data from various audit sources. The detection engine will use the data which are parsed, filtered and formatted by the data collection module to perform intrusion detection locally. LACE will locally aggregate and correlate the detection results from different detection engines in the IDS agent. No detection model stands alone as a catch-all for network penetrations. In an environment with high security requirements, it is desirable to have multiple detection engines, which enable the use of different detection techniques. They will complement each other to improve the detection performance. The functionality of LACE is to combine the detection results of different local detection engines. The functionality of GACE depends on the type

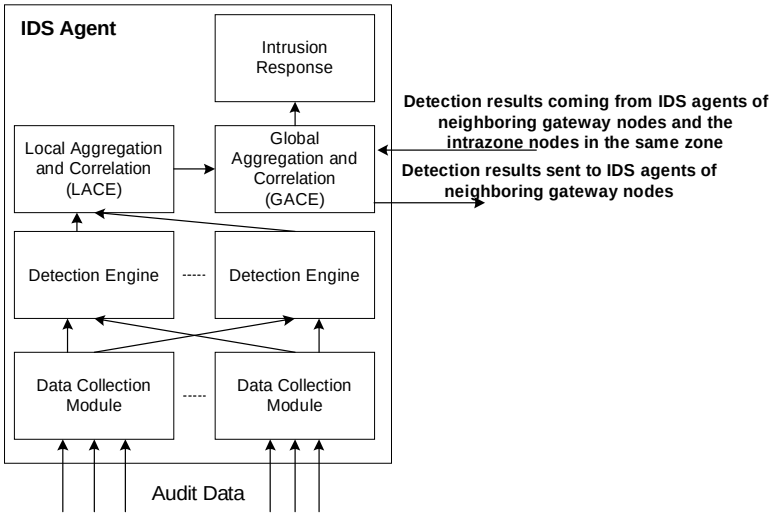


Fig. 4. Diagram of an IDS Agent.

of mobile nodes. If the node is an intrazone node, GACE is mainly responsible for transmitting the locally generated alerts to the gateway nodes in the same zone; if the node is a gateway node, GACE is to aggregate and correlate the detection results from the LACE of its own agent and the LACEs of the intrazone nodes in the same zone, and to cooperate with the GACEs of the gateway nodes with which it has physical connections. The intrusion response module is to handle the generated alarms. A detailed description of each module is as follows.

1. Data Collection Module

The functionality of the data collection module is to collect the security related data from various audit data sources and preprocess them to conform to the input format of the detection engines. There may exist many data collection modules in an IDS agent. Each module is responsible for collecting data from a particular data source. There are mainly two different data sources: network packets and host audit trails. Because

this research is focused on the routing attacks, the data source mainly consists of the routing activities, topological patterns and traffic changes, etc.

The data collection module consists of two layers: data filter layer and data preprocess layer. The functionality of the data filter layer is to select a particular security-related data source (the routing activities, e.g.) and generate the raw data. The data preprocess layer defines the representational data input format for the detection engine. It abstracts the raw data into a set of statistical variables in order to reflect the network status and periodically generates reports following the format required by the detection engine.

2. Detection Engine

Different detection techniques can be deployed in different detection engines in order to improve the detection performance. Misuse-based detection techniques operate based on the known attack scenarios and system vulnerabilities. Their main disadvantage is that they are only effective in detecting known attacks. It is expected that many new different types of attacks can be mounted in MANETs, so anomaly based detection techniques will play a main role in the MANET environment.

We try to avoid the use of computationally intensive detection techniques, such as the powerful Hidden Markov Model [64], Hotelling's T2 test [65], etc. which also require a large memory to store the computed matrix. Mobile nodes have limited power supply and storage space, thus excluding the possibility of such approaches.

Several types of anomaly detection techniques exist: string-based [21] [66], statistical-based [14] [13], and specification-based [31], etc. They differ in the format and the amount of available audit data as well as the modeling algorithms. An advantage of statistical-based anomaly detection techniques is their capability of explicitly representing and handling variations and noises involved in activities. In

the training process, the established normal profile of the subject must consider and represent variations of normal activities for distinguishing truly anomalous activities from expected variations of normal activities. Due to arbitrary mobility of nodes, MANETs are expected to demonstrate more dynamic activities. Therefore, we utilize a statistical based anomaly detection approach - the Markov Chain based anomaly detection algorithm to meet the challenge.

Details about how to construct the classifier are introduced in section E.

3. Local Aggregation and Correlation Engine (LACE)

Because different detection techniques can be deployed in the IDS agent, it is necessary for the LACE to aggregate and correlate the different detection results before transmitting them to the GACE.

The local IDS agents for a mobile node should be capable of operating in a standalone mode and detect attacks against the node. Since wireless ad hoc networks are constrained by bandwidth, energy consumption, and process capability, it is desirable to correlate the alert information on the local nodes first, as opposed to transmitting every alert across the network. The correlation could be very simple, for instance, based purely on the source address.

4. Global Aggregation and Correlation Engine (GACE)

The functionality of the GACE depends on the node types: if the node is a gateway node, its GACE utilizes the aggregation and correlation algorithm to combine the detection results from the IDS agents of intrazone nodes in the same zone and neighboring gateway nodes. If the node is an intrazone node, the functionality of the GACE is to distribute the outputs of its LACE to all of the gateway nodes in the same zone.

5. Intrusion Response Module

The countermeasures taken by the intrusion response module are different due to the different intrusions, network services, applications and confidence in the evidence. Possible countermeasures may include identifying the intruders, reinitiating the communication channels and excluding the compromised nodes from the networks.

This research focuses on intrusion detection, i.e., the detection of whether the network is under attack and the identification of the attackers. The functionality of intrusion response is simplified.

E. Anomaly Detection in Mobile Ad Hoc Networks

Detecting new attacks while keeping acceptably low false positive ratio is probably the most challenging and important problem in intrusion detection. In this section, we detail the procedure of constructing the detection model.

1. Outline of the Methodology

We collect the statistical features of interest, Percentage of the Change in Route entries (PCR) and Percentage of the Change in number of Hops (PCH), from the routing cache of mobile nodes, which reflect the mobility of the network, to construct a Markov Chain as the normal profile. The use of the Markov Chain can capture the temporal dependency among the network activities. It also takes into account their ordering property. *Vector quantization (VQ)* [30] approach is used in this process to convert the continuous raw audit data to categorized data items with minimum errors. The output of the *VQ* is then used to construct a Markov Chain model, which employs conditional probabilities in its transition probability matrix to represent the temporal profile of normal behavior. The Markov Chain model,

considering the ordering property, characterizes the normal changes of the routing caches with probabilities. It determines the probability of the next valid change, given the previous N changes. The Markov Chain model is then turned into a classifier, which serves as the detection algorithm. *Conditional entropy* is used in this process to determine the proper *window size* which parameterizes the Markov Chain model, thus avoiding the tedious trial-and-error process. The definition of *conditional entropy* and *window size* will be given in the following sections.

In the detection process, we define the distance of the current transition based on the transition probability matrix recorded in the Markov Chain model. A single deviation from the Markov Chain does not always correspond to the occurrence of an attack. An alert is generated only when the average distance over the near past is beyond some preset threshold value. The parameters used in the detection algorithm are tuned properly using the defined performance metrics.

We use an offline training process to generate the classifier. The general flowchart of using the offline training process to construct a classifier is illustrated in the right part of Fig. 5. The left part of Fig. 5 illustrates our corresponding strategy. The dotted line depicts their mapping relationships.

IDSs are classifiers. Their purposes are to *classify* an unknown activity to *normal* or *anomalous*. *Classifiers* are usually offline trained from collected data. Their purpose is to derive a set of rules which can be used during runtime. In the offline training phase, features of interest are first collected and preprocessed using training data whose classification is known *a priori*. The classification rules can then be derived correspondingly [67] [68]. In our system, we offline generate a Markov Chain to represent the subject normal behavior. Then we use normal data and abnormal data to determine proper threshold and generate the classifiers. We will elaborate on these issues in the remaining part of this chapter.

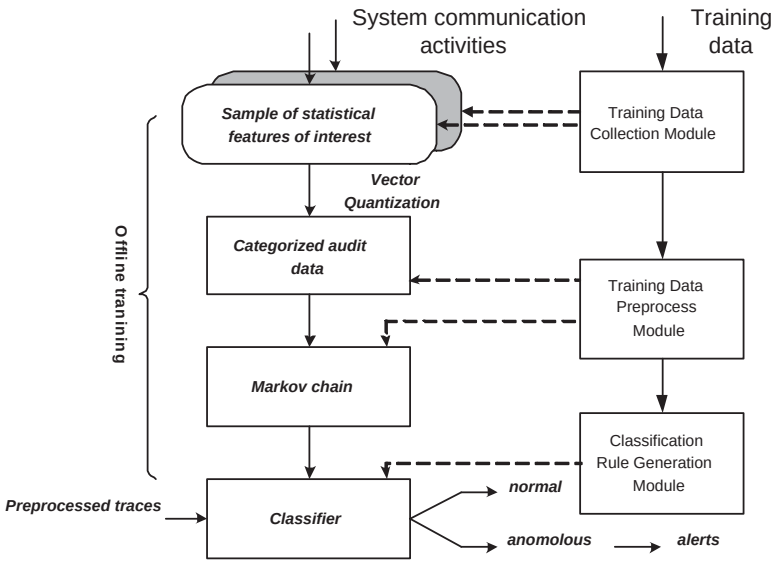


Fig. 5. General Strategy of Using the Markov Chain to Build the Classifier.

2. Detailed Descriptions

In the following, we first give some definitions related to information theory [69], then we describe how to construct the Markov Chain model and discuss how to use it to construct the classifier.

Definition 1 Entropy: Suppose X is a dataset, $C_x = \{C_x[1], C_x[2], \ldots, C_x[m]\}$ is a class set. Each data item of X belongs to a class $x \in C_x[i]$. Then the entropy of X related to this $|C_x|$-wise classification is defined as:

$$H(X) = \sum_{i=1}^{m} -P_i log P_i.$$

where P_i is the probability of x belonging to class $C_x[i]$.

Entropy is an important concept which is widely used in the field of communications. It can be interpreted as the number of bits required to encode the classification of a data item. It measures the uncertainty of a collection of data items. The lower

the *entropy*, the less varied the class distribution. If all data items belong to one class, its *entropy* is 0, which means that no bits need to be transmitted because the receiver knows that there is one class. The more varied the class distribution, the larger the *entropy*. When all of the data items are equally distributed over the m classes, its *entropy* is log$m$. In the context of anomaly detection, entropy could be used as a measure of the regularity of audit data. Here regularity refers to the sequential dependencies of sequences. Nevertheless, because we often need to predict the next state given the history information, *conditional entropy* is more useful in this respect.

Definition 2 Conditional Entropy: Suppose X and Y are two datasets, $C_x = \{C_x[1], C_x[2], \ldots, C_x[m]\}$ and $C_y = \{C_y[1], C_y[2], \ldots, C_y[n]\}$ are two class sets. Each data item of X belongs to a class $x \in C_x[i]$ and each data item of Y belongs to a class $y \in C_y[i]$. Then given Y and C_y, the entropy of X related to C_x is defined as:

$$H(X|Y) = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{ij} \log \frac{1}{P_{i|j}}$$

where P_{ij} is the probability of $x \in C_x[i]$ and $y \in C_y[j]$, $P_{i|j}$ is the probability of $x \in C_x[i]$ given $y \in C_y[j]$.

Conditional entropy describes the uncertainness of X given Y. The smaller the *conditional entropy*, the more correlated X and Y. If X can be determined by Y, $H(X|Y)$ is 0. In the context of anomaly detection, *conditional entropy* can be used to explore the temporal sequential characteristics of audit data due to the temporal nature of system activities.

More history information is desirable in achieving better detection performance. However, the more information we include in the detection model, the more data processing time required and the more complexity of the data model. *Window size* is an

important parameter to reflect how much information the data model includes. We choose the proper *window size* when the *conditional entropy* of the training data associated with the *window size* does not drop dramatically. We will have an experimental description of how to decide *window size* in the next chapter.

Definition 3 Relative Entropy: The relative entropy or *Kullback Leibler* distance between two probability mass function $p(x)$ and $q(x)$ is defined as

$$D(p \parallel q) = \sum p(x) log \frac{p(x)}{q(x)}$$

Relative entropy can be used to as a metric to measure the "distance" between two probability distributions, although it is not really a "distance". Relative entropy is always non-negative. In anomaly detection, we often build a model using the training data and apply the model to test and intrusion data. It it better that the "distance" between the training data and test data is small while the "distance" between the training data and the intrusion data is large in order to make the detection model achieve desirable performance. We will use relative entropy to demonstrate PCR and PCH are good candidate features.

Definition 4 Classifier: Suppose $\xi = \{a_1, a_2, \ldots, a_n\}$ is a set of symbols. ξ^* is a set of finite traces which only consist of the symbols in ξ. $\lambda = \{Normal, Anomalous\}$. Then in the context of intrusion detection, a classifier is a function $f : \xi^* \rightarrow \lambda$.

a. Feature Selection

Each intrusion detection approach is technically suited to identify a subset of the security violations to which the system is subject. The selection of statistical measures should be based on good understanding about the system itself as well as all possible attacks that may influence the system's normal behavior. Different attacks may be sensitive to different statistical features. Sometimes it requires domain expert knowl-

edge to help select good features. In the history of IDSs, people have used various features to construct detection models. They tend to define the normal behavior of a user, a program, or a network element. Since the ground-breaking discovery of S. Forrect [22], people find the short sequence of system calls of priviledged programs is stable in building the detection model, many research efforts have been focused on constructing different data models using the short sequence of system calls since then.

Designing IDS in MANETs needs to define new features, as MANETs are a new communication paradigm. These statistical features should be *reliable* in order to be trusted and used. Due to the distributed nature of MANETs, these features should also be *locally* collected. That is, they should be collected within the node itself or its communication activities. Because we focus on *routing disruption* attacks, we need to define features associated with the routing caches of mobile nodes in order to characterize their normal changes.

In MANETs, each mobile node can act as a router to relay data for other nodes. A routing table usually contains, at the minimum, the next hop to each destination and the distance to the destination (in terms of the number of hops). For a given application, we assume that the movement of each mobile node is independent and each node has its own specific behavior regarding movement. In typical mobile networks, nodes exhibit some degree of regularity, i.e., non-random behaviors, in their mobility patterns. For example, a car traveling on a road is likely to follow the path of the road and a tank traveling across a battlefield is likely to maintain its heading and speed for some period of time. As such, the mobile nodes tend to follow regular movement characteristics which are usually determined by specific MANET applications. As nodes in the network move in and out of wireless transmission range of one another, the routing cache of the nodes changes correspondingly, invalidating

Table I. Features Used

PCR	Percentage of the change in route entries
PCH	Percentage of the change in number of hops

old cached routing information or adding new routing entries: adding new routing entries when RREP packets are received; deleting mobility-induced routing entries when RERR packets are received; deleting expired routing cache entries when they are not used for a long time; updating the routing cache when a node promiscuously hears RREP/RERR/data packets, etc. Therefore, we define the normal updates of routing information as the normal profile.

Specifically, based on our experiment results, we use the features described in Table I that are sensitive to *routing disruption* attacks.

In DSR routing protocols, each entry contains a full path to the destination. Two routing entries are the same if the full paths contained in the route entries are the same. This includes the route destination and the hop-by-hop route comparison.

It is undesirable to consider only the route destination to distinguish two routes. According to the specification of DSR [6], for a given node, it is possible that there exist multiple routes to a destination (this is called *path redundancy*). If we only consider the route destination to distinguish two routes, this may not be desirable because it will only demonstrate a very small portion of the route cache changes.

Suppose for a given node, at time $t1$, there are $N1$ routing entries, the routing entry set is $S1$, and the sum of hops of all routing entries is $H1$; at time $t2$, there are $N2$ routing entries, the routing entry set is $S2$, and the sum of hops of all routing entries is $H2$. We define PCR and PCH as following:

- *PCR*: *PCR* is calculated as $(|S2 - S1| + |S1 - S2|)/|S1|$. $|S|$ indicates the number of elements in S. $(S2 - S1)$ means the newly increased routing entries during the time interval $(t2 - t1)$, and $(S1 - S2)$ means the deleted routing entries during $(t2 - t1)$. They together represent the changes of routing entries in $(t2 - t1)$.

- *PCH*: *PCH* is calculated as $(H2 - H1)/H1$. $(H2 - H1)$ indicates the changes of the sum of hops of all routing entries during the time interval $(t2 - t1)$.

For traces at each mobility level, we further measure the *relative entropy* between training data and test normal data (denoted as RE_{test} henceforth) and the *relative entropy* between training data and intrusion data (denoted as $RE_{intrusion}$ henceforth) in order to demonstrate the effectiveness of *PCR* and *PCH*. *Relative entropy* is a measure of the "distance" between two probability mass functions [69]. For anomaly detection, in order to achieve high performance, RE_{test} should be small (indicating the same or similar regularity between training data and test normal data) while $RE_{intrusion}$ should be large (indicating different regularity between training data and intrusion data) [70][71].

Intrusion detection needs to consider subject behavior in recent history. Therefore, for test data and intrusion data, we use the limited sample size which reflects its recent subject activities. For a sample of size n, if the number of occurrence of an item c is n_c, the probability of c is calculated as n_c/n. Because of the large amount of training data and limited test data (intrusion data) items, it is possible that some item appearing in training data does not appear in test data. Therefore, based on the definition of *relative entropy*, we cannot calculate the divergence of these two probability distributions because the denominator is 0. To cope with this problem, we adopt the popular Jelinek-Mercer smoothing method [72] to eliminate the zero-

frequency problem. That is, for item c, if its probability is 0 in test data (intrusion data), we use the following formula to modify its probability:

$$p_\lambda(c) = \lambda q(c) + (1 - \lambda)p(c), \qquad 0 \leq \lambda \leq 1.$$

where $q(c)$ is the probability of c in test data (intrusion data) for a given history, and $p(c)$ is the probability of c calculated from training data. λ is an interpolation coefficient which is applied as a balancing weight between the observed probability in test data (intrusion data) and the probability calculated from training data. Simulation results of *relative entropy* will be illustrated in section a.

We do not use features related to the physical movement of nodes (such as the *velocity*, *direction* and *moving distance*). In the context of our threat model, the attackers can still attack the network following their normal movements.

We have two alternatives when collecting raw feature values: *periodic* and *event-triggered*. We say an *event* happens each time the node receives a packet that triggers the update of the routing entry. For the *event-triggered* mechanism, the data collection module computes *PCR* and *PCH* each time an event happens. This mechanism could capture all route changes and therefore is more accurate to reflect the routing cache statistics. However, when data are collected this way, we may need to keep track of all *events* (RREP/RERR/data packets, etc.) and their time. The burst of *events* may make their processes a heavy burden for mobile nodes. This is further complicated by the time calculation. Therefore, it is difficult for *event-triggered* mechanism to meet the demands of MANET IDSs.

Therefore, we adopt a *periodic* mechanism. We collect the raw feature value every observation period, which determines the detection resolution. The data collection module of each IDS agent *periodically* collects data and preprocesses these statistical

measures into suitable formats. In this way, the temporal behavior of the routing activities can be represented as a discrete-time stochastic process.

b. Data Preprocess

The output of the data collection module contains only continuous data. However, the construction of the Markov Chain model requires discrete data. A suitable mechanism is thus needed to perform the data transformation.

One simple approach is to use n classes to represent raw audit data in n ranges. The range level can thus be simply defined based on the minimum and maximum value of raw data. However, this approach does not take into account the attributes of raw data and could possibly introduce high error rates. Too small number of bins will make the bins too "coarse", which will enable most values to reside in one or two bins. This will result in less sensitivity to intrusions. On the other hand, too big number of bins tend to be too "fine", so that many values would be associated with very small probabilities. The inappropriate selection of the number of bins would influence the performance of the IDS.

We thus propose to use the *Vector Quantization*(VQ) algorithm [30] to discretize the raw continuous data. VQ is a lossy data compression method based on the principle of block coding. In *VQ* algorithm, each input vector is mapped to one of a finite set of predetermined vectors. This set of predetermined vectors, called codevectors, is the codebook. Given a vector source (corresponding to raw training data) and a distortion measure (we use the commonly used *squared-error* distortion measure), it outputs a codebook and a partition of training data that will result in the smallest average distortion.

Let's assume training data consists of M source vectors:

$T = \{x_1, x_2, \ldots, x_M\}$.

In our context, they are the one-dimensional data (*PCR* or *PCH*) periodically obtained from the routing caches of the mobile nodes. M is assumed to be sufficiently large so that the statistical properties of the source are captured in the training sequence. Suppose each source vector is k-dimensional:

$x_m = (x_{m,1}, x_{m,2}, \ldots, x_{m,k}), \qquad m = 1, 2, \ldots, M.$

Let N be the number of codevectors which make up the codebook C,

$C = \{c_1, c_2, \ldots, c_N\}$,

Each codevector is also k-dimensional,

$c_n = (c_{n,1}, c_{n,2}, \ldots, c_{n,k}), \qquad n = 1, 2, \ldots, N.$

Let S_n be the encoding region associated with the codevector c_n and let

$P = \{S_1, S_2, \ldots, S_N\}$.

denote the partition of the space. If the source vector x_m is in the encoding region S_n, then its approximation (denoted by $Q(x_m)$) is

c_n: $Q(x_m) = c_n, \qquad if\ x_m \subseteq S_n.$

Because we use the commonly used *squared error* ($d(x, \overline{x}) = (x - \overline{x})^2$, where $\overline{x}$ is the approximation of x) as the distortion measure, the average distortion is given by:

$$D_{ave} = \frac{1}{Mk} \sum_{m=1}^{M} \parallel x_m - Q(x_m) \parallel^2,$$

where $||y|| = (y_1)^2 + (y_2)^2 + ... + (y_k)^2$.

The VQ algorithm can be described as: given T and N, find C and P such that D_{ave} is minimized. C and P, as the solution to the problem, must satisfy the following two criteria:

- Nearest Neighbor Condition: $S_n = \{x :\parallel x - c_n \parallel^2 \leq \parallel x - c_m \parallel^2, \qquad \forall m = 1, 2, \ldots, N\}$. This condition says that the encoding region S_n should consist of all source vectors that are closer to c_n than any of the other codevectors.

- Centroid Condition: $c_n = \sum_{x_m \subseteq S_n} X_m / \sum_{x_m \subseteq S_n} 1, \qquad n = 1, 2, \ldots, N.$ This condition says that the codevector c_n should be the average of all those training vectors that are in the encoding region S_n.

Unfortunately, designing a codebook that best represents the set of input vectors is NP-hard. We therefore resort to the suboptimal codebook design schemes - the Linde-Buzo-Gray (LBG) algorithm [73]. The LBG VQ algorithm works iteratively to find the codebook C and the partition S of training data to guarantee local optimality by comparing each input vector with all the codevectors.

After we obtain the codebook C, we need to find the mapped value for each raw data item. We find it in this way: for each data item x, we map the codevector in the codebook which has the smallest distance (measured as the $(x - Q(x))^2$) to x. In this way, the raw data are converted into categorized data.

In our case, we use the 1-dimensional input and construct separate codebooks for different features (PCR and PCH). The input raw data items are thus converted into categorized items suitable for the construction of the Markov Chain model.

Due to the dynamic nature of MANETs, nodes may move in an arbitrary manner. It is thus possible that the statistical feature could have unexpected sudden changes, leading to the small probability of some categorized data items. These categorized values are abnormal yet not malicious. They are undesirable in the Markov Chain construction process because they do not represent the general normal changes of the routing caches and introduce noise to training data. Their existence could introduce unnecessary states and lead to the distortion of the Markov state transition calculation. We thus take the following action to cope with this problem: for those data items whose probability is below some threshold, we convert them to a "rare" symbol. In this way, the noise in the training data could be reduced. We will detail

this in the next chapter.

The whole process of using VQ algorithm is illustrated in Fig. 6.

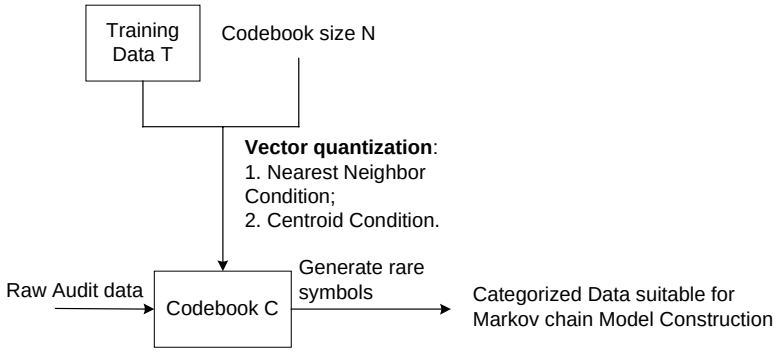


Fig. 6. Data Preprocess Using Vector Quantization.

c. Markov Chain Based Intrusion Detection

In [35], using the classification algorithm RIPPER [74], and the most discriminating feature values as the *concept*, classification rules can be obtained and used to construct normal profiles. This approach requires examples of abnormal behavior. However, it is difficult to obtain sufficient examples of malicious behavior that compromise system security. Leaving aside the practical difficulty of obtaining instances of hostile activities, there is an issue of coverage. The space of possible malicious behavior is potentially infinite. It would be difficult to demonstrate complete coverage of the space from a finite training corpus.

We utilize a Markov Chain based anomaly detection algorithm. It characterizes the normal behavior of the system and captures the characteristics of the temporal sequence of the system audit data by utilizing which *states* it moves between and with what probabilities.

That is, we define the *from_state* as $\{X_i, X_{i+1}, \ldots, X_{i+w-1}\}$, and *to_state* as

$\{X_{i+w}\}$ (w is a parameter that characterizes the Markov Chain). Therefore, we can consider the routing changes as a random process with stationary transition probabilities. We use the VQ algorithm to preprocess the data and introduce "rare symbol" to construct a Markov Chain which would lead to better results in our environment. *Locality frame* is also used in the classifier construction. We also adopt a different approach to tune the parameters.

A Markov Chain is a special type of discrete-time stochastic process. If a collection of random variables X_t (where the index t runs through $0, 1, \ldots$) has the property that:

- the probability distribution of the state at time t+1 only depends on the state at time t;
- the state transition from time t to time t+1 is independent of time;

The sequence of states X_t forms a Markov Chain.

In other words, $P(X_t = j|X_0 = i_0, X_1 = i_1, \ldots, X_{t-1} = i_{t-1}) = P(X_t = j|X_{t-1} = i_{t-1})$ and $P(X_t = i_t|X_{t-1} = i_{t-1}) = P(X_t = j|X_{t-1} = i) = p_{ij}$.

If the system has a finite number of states, the Markov Chain model describes a transition probability matrix:

$$\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1s} \\ p_{21} & p_{22} & \cdots & p_{2s} \\ \vdots & \vdots & \vdots & \vdots \\ p_{s1} & p_{s2} & \cdots & p_{ss} \end{bmatrix} \tag{3.1}$$

where s is the number of possible states and

$$\sum_{j=1}^{s} p_{ij} = 1. \tag{3.2}$$

Assume $(X_1, X_2, \ldots, X_t)$ gives a time-series representation of a given event sequence, where X_t denotes an event occurring at time t. We compute the transition probability matrix as follows:

$$p_{ij} = \frac{N_{ij}}{N_i} \tag{3.3}$$

where

- N_{ij} is the number of observation pairs X_t and X_{t+1} with X_t in state i and X_{t+1} in state j;
- N_i is the number of X_t in state i.

Normal profiles are used to characterize the normal behavior of the system. One of the difficulties using the Markov Chain to construct the normal profile is to define the states. Here we define the *from-state* as the previous N ordered values of the statistical measure and the *to-state* as the current statistical feature value. That is, we define the *from-state* as $\{X_i, X_{i+1}, \ldots, X_{i+w-1}\}$, and *to-state* as $\{X_{i+w}\}$ (w is a parameter that characterizes the Markov Chain.). If the transition probability $P[X_{n+1} = j | X_n = i]$ of the Markov Chain is the same for every n, we say that the Chain has *stationary transition probabilities* or has the *homogeneity* property [75] [76]. Therefore, we can consider the routing changes as a random process with stationary transition probabilities.

d. Construction of the Markov Chain Model

Before we describe how to construct the Markov Chain model, we first describe some related notations. Let ξ denote the set of symbols. A *trace* over ξ is a finite sequence of symbols. The set of finite traces over ξ is denoted as ξ^* and the set of traces

Table II. Variable Notations

ξ	the set of symbols	ξ^*	the set of finite traces over ξ
ξ_w	the set of traces of length w	$\|\psi\|$	the length of trace ψ
ζ	trace of test/anomalous data	ψ	a trace of training data
M	the set of traces formatted from raw audit data	w	the window size

of length w is denoted as ξ_w. Given a trace ψ over ξ, $|\psi|$ denotes its length. We summarize the notations of variables in Table II.

The size of ξ (the number of different symbols) is determined by the number of codevectors. Each codevector is represented by one symbol. Each *from-state* in the Markov Chain model is associated with a sequence of symbols, which is defined on $\xi \cup \{\phi\}$ and its length is w. Each tuple (s, s') is a *state* pair that represents the state transition from s to s'. All *states* are stored in one hash table H. The use of the hash table is to speed up the processing and is not crucial to the description of the algorithm. Each *state* and *transition* is associated with a *counter*, which indicates how many times this *state* or *transition* has occurred.

Each training trace is converted into a sequence of symbols over ξ. All of the sequences of symbols construct $M \in \xi^*$. The algorithm used to construct the Markov Chain model from M is illustrated in Fig. 7.

In Fig. 7, the initial *from_state* of the Markov Chain model is associated with a symbol sequence of length w consisting of the first w symbols of ψ. w is *window size*, which indicates the number of symbols associated with *from_state*. For each trace ψ, it sets *from_state* and *to_state*. If the *from_state* is not in hash table H, it is inserted into H and associated with a counter 1. If the *from_state* is already inserted into hash

```
Procedure Construct_Markov()

Input    D: raw data set;
Output   H: hash table that stores the states and the associated counters;

Begin

Use VQ algorithm to preprocess D;
Generate rare symbols, and convert D to a set of traces, denoted as M;

For each ψ ∈ M
{
        from_state is set to the first w symbols of ψ;
        shift ψ left by w positions;

        While (not reaching the end of ψ)
        {
                to_state is set to the first symbol in ψ;
                shift ψ left by one position;

                If (from_state ∉ H)
                {
                        from_state → H;
                        the counter of from_state in H is set to 1;
                }
                Else
                        increase the counter of from_state by 1;

                If (transition(from_state, to_state) ∉ H)
                {
                        (from_state, to_state) → H;

                        the counter of (from_state, to_state) is set to 1;
                }
                Else
                        increase the counter of (from_state, to_state) by 1.

                shift from_state left by one position;
                append to_state to the end of from_state;

        } /*for While*/
} /*for For*/

End.
```

Fig. 7. Pseudocode to Construct the Markov Chain Model.

table H, its associated counter is increased by 1. If the *transition(from_state, to_state)* is not in hash table H, it is inserted into H and associated with a counter 1. If the *transition(from_state, to_state)* is already inserted into hash table H, its associated counter is increased by 1. We can imagine H is constructed using a window of size w sliding through the trace ψ, each time by one position.

After all the traces in the training set have been processed, hash table H are constructed. They store the possible normal *states* and their *transitions* respectively. Each *state* and *transition* are associated with a positive integer. The probability of the transition (s, s') in H is calculated as:

$$P((s, s')) = \frac{N((s, s'))}{N(s)}$$

where $N((s, s'))$ is the counter associated with the *transition* (s, s'), and $N(s)$ is the counter associated with the *state* s. The higher probability the *transition* (s, s'), the more likely this *transition* is normal. An intrusive *transition* is expected to receive a low probability of support from the Markov Chain model of the normal profile.

Because *from-state* is denoted as $\{X_i, X_{i+1}, \ldots, X_{i+w-1}\}$, and *to-state* is denoted as $\{X_{i+w}\}$, we can see that the probability transition matrix is the same if we denote the *to-state* as $\{X_{i+1}, \ldots, X_{i+w-1}, X_{i+w}\}$. We can then depict a simple example in Fig. 8 to illustrate how the Markov Chain model is constructed from the training traces. Here we assume the *window size* is 3 and the set of input training traces are

$$\{xxxyz, xxxzy\}$$

Fig. 8 also shows the counters associated with the *states* and the *transitions*.

Based on the algorithm described in Fig. 7, the Markov Chain model can be denoted using a tuple $\{S, P, s_0\}$, where S is the set of all possible states, $P : (S \times S) \rightarrow$

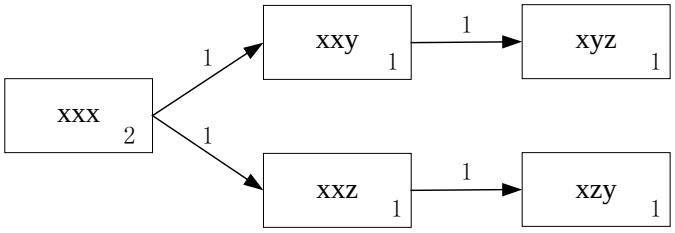


Fig. 8. A Simple Example to Illustrate How to Construct Makov Chains.

$\Re_+$ is the set of transition probabilities, $s_0 \in S$ is the initial state of the constructed Markov Chain model and $\Re_+$ denotes the set of non-negative real numbers. It is obvious that P satisfies the following:

$$\sum_{s' \in Next(s)} P((s, s')) = 1, \qquad \forall s \in S.$$

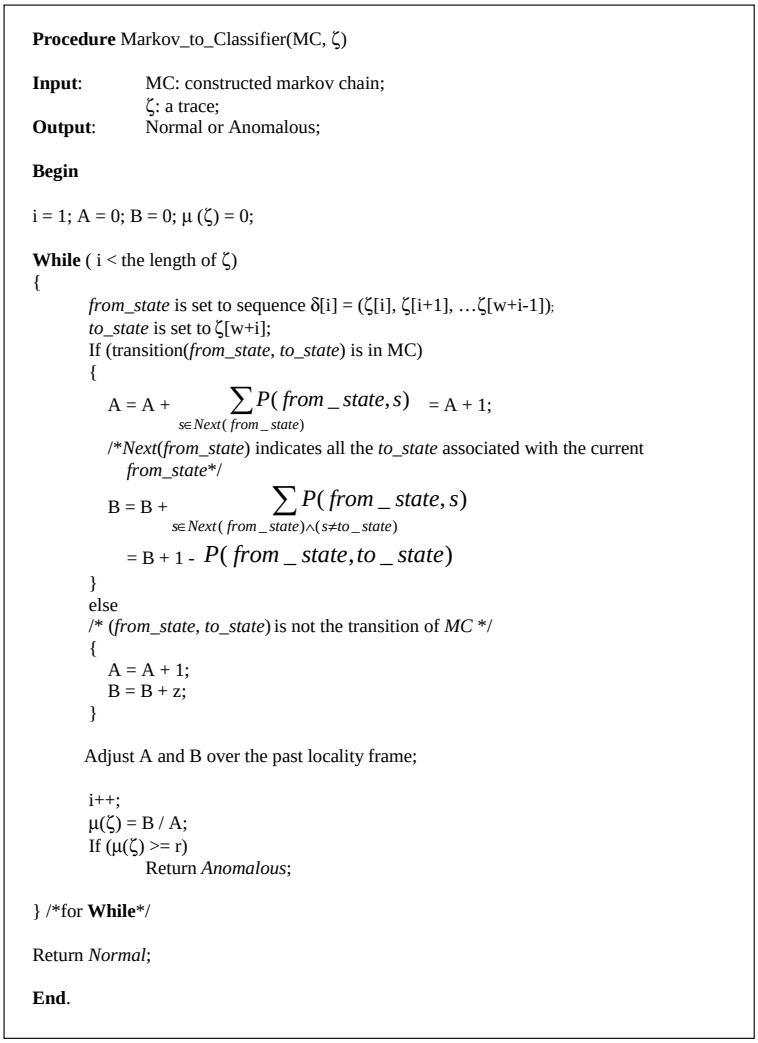
where $Next(s)$ denotes the set of *to_state* associated with s.

e. Constructing Classifiers Using the Markov Chain Model

Suppose $\zeta \in \xi^*$ is a trace converted from the raw system audit data. $l_\zeta = \{\delta_1, \delta_2, \ldots, \delta_{m-w+1}\}$ is the sequence of symbols corresponding to ζ, where $\delta_1 = \underbrace{[\zeta_1, \zeta_2, \ldots, \zeta_w]}_{w}$, $\delta_k = (\zeta_k, \zeta_{k+1}, \ldots, \zeta_{k+w-1})$, $\delta_k \in \xi_w$, $m = |\zeta|$. The algorithm using the Markov Chain model (MC) to construct the classifier is described in Fig. 9.

The algorithm first initializes two real numbers A and B to 0. δ_i, the *from_state*, is obtained after repeatedly shift ζ left by one position from its beginning. If the *transition* $(from\_state, to\_state)$ exists in MC, A is increased by 1 and B increases by $F = 1 - P(from\_state, to\_state)$. Therefore, F sums up all of the probabilities of the transitions from the *from_state* that are not equal to the current $(from\_state, to\_state)$.

If the transition does not exist in MC, A is increased by 1 and B is increased by



Procedure Markov_to_Classifier(MC, ζ)

Input: MC: constructed markov chain;
ζ: a trace;
Output: Normal or Anomalous;

Begin

i = 1; A = 0; B = 0; μ (ζ) = 0;

While (i < the length of ζ)
{
from_state is set to sequence δ[i] = (ζ[i], ζ[i+1], …ζ[w+i-1]);
to_state is set to ζ[w+i];
If (transition(*from_state*, *to_state*) is in MC)
{
A = A + $\sum_{s \in Next(\ from\_state)} P(\ from\_state, s)$ = A + 1;
/**Next*(*from_state*) indicates all the *to_state* associated with the current *from_state**/
B = B + $\sum_{s \in Next(\ from\_state) \wedge (s \neq to\_state)} P(\ from\_state, s)$
= B + 1 - $P(\ from\_state, to\_state)$
}
else
/* (*from_state*, *to_state*) is not the transition of *MC* */
{
A = A + 1;
B = B + z;
}

Adjust A and B over the past locality frame;

i++;
μ(ζ) = B / A;
If (μ(ζ) >= r)
Return *Anomalous*;

} /*for **While**/

Return *Normal*;

End.

Fig. 9. Pseudocode of Using the Markov Chain Model to Build the Classifier.

the *penalized value* z. After each calculation, if B/A exceeds a preset threshold value r, ζ is considered containing malicious activities (return *Anomalous*). A and B are also updated after each calculation to accommodate the *locality frame* scheme. This requires the deletion of the oldest value, and the addition of the newest value. The usage of the hash table here could speed up the state and transition searching. A well designed hash table takes $O(1)$ time on average. Therefore, this algorithm can be executed efficiently.

Several parameters, such as the *penalized value* z and the *alert threshold* r, will be determined later through experiments. Here, a simple analysis of this algorithm is performed because it needs to be executed online in reality.

In Fig. 9, a hash table is used to maintain MC and its transition probability matrix. We skip the description of hash table and its hash function here. Interested readers could refer to [77]. In a hash table in which collisions are resolved by chaining, if the number of hash-table slots is at least proportional to the number of elements in the table and the hash function satisfies the assumption of simple uniform hashing, searching takes $O(1)$ time on average. In practice, *division method* (a key k interpreted as a natural number is mapped into one of m slots by taking the remainder of k divided by m) could be used as the hash function. Therefore, the time complexity of this algorithm depends on the length of the audit trace. We can see that given a well-designed hash table, this algorithm can be executed efficiently.

We derive our online measure r, or the *alert threshold*, from the number of mismatches occurring in a temporally local region, called a *locality frame*. At each point in our test trace, we check whether the current transition exists in the Markov Chain model, and keep track of the updated *alert signal* (defined as B/A) over the past *locality frame*. The B/A over the past *locality frame* are aggregated into the *alert signal*, as described in Fig. 9.

Different selections of A, B, and the alert threshold r will result in different detection models. Intuitively, the metric B/A measures how well the Markov Chain model predicts the trace ζ, e.g., a lower B/A indicates that the Markov Chain predicts the trace well. For simplicity, we will use

$$A = A + \sum_{s \in Next(from\_state)} P(from\_state, s) = A + 1;$$

$$B = B + \sum_{s \in Next(from\_state) \bigwedge s \neq to\_state} P(from\_state, s) = B + 1 - P(from\_state, to\_state);$$

where $Next(from\_state)$ denotes the set of to_state associated with this $from\_state$.

In this definition, B sums up all of the probabilities of the transitions from the $from\_state$ that are not equal to $(from\_state, to\_state)$. When analyzing the trace as in the **While** statement depicted in Fig. 9, if the *transition* $(from\_state, to\_state)$ has a low probability according to the constructed Markov Chain MC, it might be an anomaly. Since B adds up all the probabilities of the *transitions* from the $from\_state$ that are not equal to $(from\_state, to\_state)$, the small occurring probability of $(from\_state, to\_state)$ will lead to relatively large B. In other words, if a *transition* with a low probability occurs, B has a high value. Given the alert threshold r, this indicates the higher probability to generate an anomaly. There may exist different definitions of A and B to derive the "distance" of the current trace. This will be a future research question.

The performance of the classifier constructed from the Markov Chain model depends heavily on the parameter *window size* w and the *alert threshold* r. These parameters need to be tuned properly in order to achieve good performance. As the *window size* w increases, the algorithm depicted in Fig. 9 constructs a better model because it considers more historical data. However, the classical overfitting problem

will appear for a very large w because the constructed Markov Chain models the training data "too well". Given a hypothesis space H, a hypothesis $h \in H$ is said to **overfit** the training data if there exists some alternative hypothesis $h' \in H$, such that h has a smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances [78]. The possible random errors or noise in the training data could lead to overfitting. Overfitting is a significant practical difficulty for many learning approaches. As to the *alert threshold* r, the determination of its value will also lead to different performance of the classifier. We will have a detailed discussion of the parameter setting in Chapter IV.

We use the average distance over a *locality frame* to measure how well the trace matches the Markov Chain. This finally determines the *alert signal* of the trace in the near past. A *locality frame* is a length of a temporally local region over which the *alert signal* is determined. That is, only when the average distance over the past *locality frame* is larger than a predefined *alert threshold* at some point during the intrusion will the IDS generate an alert.

The local detection mechanism used here could aggregate the mismatch counts and is not sensitive to the trace length. Because of the dynamic nature of MANETs, it is expected to have many variations involved in its activities. Our approach is resilient to sudden abnormal changes, which is usually normal in MANET environments. Therefore our approach can avoid high false positive ratio due to the unexpected sudden changes of the statistical measures. Because real intrusions tend to produce anomalous sequences in temporally local clusters, our approach can still achieve high detection ratio.

The relation among *locality frame*, *window size* and the state transition is illustrated in Fig. 10.

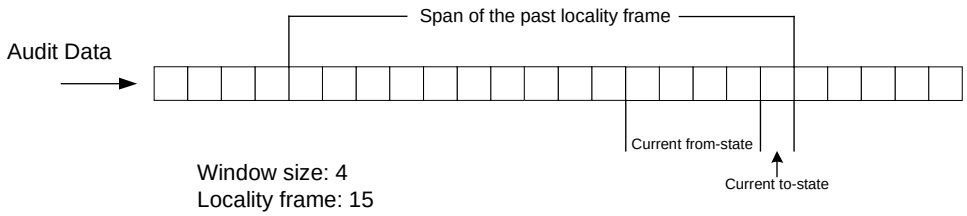


Fig. 10. Illustration Among Locality Frame, Window Size and State Transition.

3. Parameter Tuning

Performance tuning is detailed in the experiment part. Here we discuss the guidelines in deciding the suitable *window size* w, *penalized value* z and *alert threshold* r.

Tune the window size: Intrusion detection needs to consider the history of the subject activities because of their temporal nature. A proper *window size* needs to be determined. Different statistical measures will have different regularities under different mobility patterns. We use the *conditional entropy* to measure the regularity of the training data and help us to determine w.

Regularity refers to the sequential dependencies of sequences. Conditional entropy measures how much uncertainty remains for the current event given the previous N events. High-regularity data contain redundancies that facilitate predicting future events based on past events, while low-regularity impedes prediction.

Let X be a collection of sequences where each is denoted as $(e_1, e_2, \ldots e_n)$, and each e_i is an audit event. Let Y be the collection of subsequences where each is denoted as $(e_1, e_2, \ldots e_k)$, 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 (since y is always a subsequence of x, we have $P(x, y) = P(x)$.). For anomaly detection, the smaller the conditional entropy,

the better the detection model. For example, if we have a sequence of audit events of the same type, e.g., $X = \{aaaaaa, bbbbbb, ...\}$, then the conditional entropy is 0 and the event sequence is deterministic. If the conditional entropy is large, the audit sequence is not deterministic and is hard to model.

Intuitively, the more information the data model contains, the better the detection performance. However, if too much information is included, not only the data processing time, but also the model complexities will increase. Therefore, there is a trade-off in determining the proper *window size*. We choose the *window size* w when the *conditional entropy* parameterized by w does not drop dramatically. Audit data of different mobility scenarios will have different *conditional entropy* values given the same *window size*. This is one of the main reasons that different performance can be observed under different mobility scenarios in our later experiments.

Tune the *penalized value* and *alert threshold*: After deciding the *window size* which parameterizes the Markov Chain, two more important parameters are the *penalized value* z, which is used by the classifier when a transition is not found in the Markov Chain model, and the *alert threshold* r.

Let $\mu(\alpha) = B/A$ be the metric defined earlier. Intuitively, it measures the discrepancy between the Markov Chain model and the current trace. Smaller $\mu(\alpha)$ means a better fit of the current trace. For a normal trace $\alpha = \{\beta_1, \beta_2, \ldots\}$, the discrepancy $D_t(\alpha)$ over the *locality frame* with length L is:

$$D_t(\alpha) = \frac{\mu_L(\alpha) + \mu_{L+1}(\alpha) + \ldots + \mu_{|\alpha|}(\alpha)}{(|\alpha| - L + 1)} \tag{3.4}$$

where $\mu_i(\alpha)$ is the average μ over the locality frame $\{\beta_{i+1-L}, \beta_{i+2-L}, \ldots, \beta_i\}$, $\quad i \geq L$.

For a given normal trace set T_t, its discrepancy $D_t(T_t)$ is:

$$D_t(T_t) = \frac{\sum_{\alpha \subseteq T_t} D_t(\alpha)}{|T_t|} \tag{3.5}$$

where $|T_t|$ denotes the number of traces in T_t.

For a given trace α of intrusive activities, we define the discrepancy $D_a(\alpha)$ over the locality frame with length L as:

$$D_a(\alpha) = Max(\mu_L(\alpha), \mu_{L+1}(\alpha), \ldots, \mu_{|\alpha|}(\alpha)). \tag{3.6}$$

$\mu_i(\alpha)$ has the same meaning as before. We use Max here because it is possible that there are normal data mixed together with abnormal data due to the possible intermittent trigger of the attacks.

For a given trace set T_a of intrusive activities, its discrepancy $D_a(T_a)$ is:

$$D_a(T_a) = \frac{\sum_{\alpha \subseteq T_a} D_a(\alpha)}{|T_a|} \tag{3.7}$$

where $|T_a|$ denotes the number of traces in T_a.

We tune the *penalized value* z until the separation $(D_a(T_a) - D_t(T_t))$ between the anomalous traces and the test data is above a certain threshold. This makes the easy distinction between a normal trace and the trace of intrusive behavior.

After determining the *penalized value* z, a proper *alert threshold* value r should be decided. If the alert threshold is set too low, then on an intrusive trace, the classifier will generate an alert very quickly. However, a lower threshold also generates more false alarms on normal traces, resulting in a high false positive ratio. If the alert threshold is set too high, we can reduce the false positive ratio. However, we may risk failing to detect an attack and the average detection time may be high. Therefore, the value of the alert threshold is a trade-off. We use the following formula to decide r:

$$r = h_t * D_t(T_t) + h_a * D_a(T_a)$$

where $h_t > 0, h_a > 0$, and $h_t + h_a = 1$. Therefore, r is the weighted sum between $D_t(T_t)$ and $D_a(T_a)$. For simplicity, we set $r = (D_t(T_t) + D_a(T_a))/2$ in our later experiments.

Due to the nature of statistical approaches, it is difficult to provide diagnostic information to accurately analyze an attack. Therefore, the statistical procedure must provide necessary information that can be used to identify the reason of an alarm. In our implementation, each node records the number of routing control packets sent from each node and their associated probability in each period. This information could help to identify the reason of the attack and is necessary for the aggregation algorithm described later.

In this section, based on the attack model, we describe a Markov Chain based anomaly detector using PCR and PCH. It is a common sense that no detection approach is suitable to detect all penetrations. Therefore, our method is expected to complement with other approaches to address a comprehensive set of vulnerabilities.

F. Collaboration of IDS Agents

1. Introduction

In mobile ad hoc networks, the attackers can launch attacks when they are close to the victims. Thus, in general cases, through the local IDS agent attached to the node itself and/or neighboring nodes, these attackers can be detected. There are other kinds of attacks, however, which the attackers may launch far away from the victims. What's more, two or more attackers may collude to launch attacks that are more

complicated. In these situations, it is very difficult for the node itself to detect the attacks. In addition, the result based on the local IDS agent could lead to a very high false positive ratio.

We expect different approaches for local intrusion detection developed in the future. Due to the local nature of the audit data, they generally tend to generate many false positives and the alerts generated are trivial. Therefore, it is desirable to collaborate different IDS agents which aim at different attacks in different environments. Their approaches are complementary and the results could be aggregated to derive a global diagnosis of the attack. It is thus necessary to develop a framework to manage these alerts, analyze and aggregate the alerts in a wider area. In this section, we describe our work in designing and analyzing a nonoverlapping zone-based management framework that fits the requirement to aggregate the alerts in a wider area. We describe an aggregation algorithm that uses the similarity of attributes to aggregate the alerts.

Because we lack detailed analysis of MANET attacks in the literature and sophisticated attacks may make the situations very complex, we only consider the same occurrence of attacks. Specifically, we still target at the *routing disruption* attack.

The main objective of the aggregation algorithm and the zone-based framework is to reduce false positive ratio and increase detection ratio by aggregating local alerts. Global alerts could be generated to provide more diagnostic information of the attacks.

Alert aggregation is a new research area in intrusion detection systems. Most existing alert management systems develop their framework based on fully developed wired IDSs, and carry out experiments based on misused based IDS like Snort [79], e-Trust [80], etc. Because misuse based IDSs could provide more accurate and diagnostic information about the attack, this could greatly facilitate the alert aggregation

and correlation. The extensive attack database of wired network could also provide abundant attack scenarios for analysis.

The situation is different in MANETs. Due to the lack of detailed analysis of attacks in MANETs, alert aggregation is very challenging in MANET environment. Potential complex attack scenarios could make the aggregation very complicated. What's more, there are not many efforts that have been devoted to the local detection of MANET attacks. However, the distributed nature of MANETs makes alert aggregation an indispensable and integral part of MANET IDSs. In this section, we provide our initial work in this respect. We use our Markov Chain based anomaly detection model as the local detection model and the routing disruption attack as the attack model to demonstrate our aggregation algorithm.

2. Zone-Based Framework

In our nonoverlapping zone-based framework, it is necessary to maintain the stability of the zone connectivity for the cooperation of neighboring gateway nodes. If any two nodes are within the communication range, a physical link exists between them. If there is at least one physical link connecting any two zones, a virtual link between the two zones exists. When the number of physical connections between two zones decreases from 1 to 0 or increases from 0 to 1 due to the movement of nodes, the zone connectivity will change correspondingly. This can be mitigated by choosing the appropriate zone size. The selection of the suitable zone size needs to make the number of physical connections between two neighboring zones much bigger than zero, thus making the logical connection of two zones stable.

a. Collaboration Mechanism

There may exist two possible mechanisms for the gateway nodes to collaborate.

One is the *subscription-based* mechanism. It is not necessary that the gateway nodes collect all of the security related information from the IDSs of intrazone nodes in order to draw some conclusions. Based on its own status, the IDS of the gateway node can send a *subscription* message to its intrazone nodes to subscribe security related information. The *subscription* message could contain information that is related to the required data. The intrazone nodes can thus generate corresponding messages to fit the subscribed requirement. This mechanism introduces low communication overhead. However, the gateway node needs to carefully analyze the messages in order to determine what information is needed.

The other mechanism is the *local broadcast* mechanism. When the IDS of the intrazone node generates a local alert, it could locally propagate the detection results to the gateway nodes. When nothing is suspicious in the last period, there is no need for the local IDS to propagate security information. The neighboring gateway nodes could further collaborate through the transmission of the security-related information by the *beacon* messages. The rationale behind this mechanism is that: audit data from other nodes cannot be trusted and should not be used because the compromised node may send falsified data. However, the compromised nodes have no incentive to send reports of intrusion detection because this may result in their expulsion from the network. In this way, we avoid the use of global broadcast. It is also unnecessary to propagate local alerts inside the zone every period. All these strategies can result in less communication overhead. This is very desirable because message sending and receiving is very expensive in terms of energy. This mechanism could also enable gateway nodes to collect enough information to make final decisions. We thus adopt the *local broadcast* mechanism in our implementation.

In the zone-based intrusion detection framework, only gateway nodes could generate alarms. The local IDS attached to local nodes could only generate alerts based

on their local information and propagate these alerts inside the zone. The gateway nodes, having gathered the alert information periodically, could make better final decisions.

There may exist routing mechanisms enabling the intrazone node to disseminate the locally generated alerts to its gateway nodes in a more efficient manner. However, this is beyond the scope of this research.

3. Aggregation Mechanism

The purpose of the aggregation algorithm is to reduce false positive ratio and increase detection ratio by aggregating local alerts and present a broad view of the reported security issues. By grouping alerts together, aggregation will allow a better evaluation of the progress of the attack. In order to do so, we need the definition of a data model in the form of a class hierarchy to describe the alerts.

a. Class Hierarchy of the Alerts

The Intrusion Detection Working Group(IDWG) of the Internet Engineering Task Force (IETF) develops the Intrusion Detection Message Exchange Format(IDMEF)[3], which is intended as the standard to facilitate the inter-operability of commercial IDSs and research prototypes. Intrusion detection systems can use this format to generate the alert information. The IDMEF data model is an object-oriented representation of the alert data and is described using the Unified Modeling Language(UML) [81]. UML defines entities as classes, which consists of class name and class attributes, as depicted in Fig. 11. Currently, the IDMEF model uses only two of the relationship types defined by UML: *inheritance* and *aggregation*. *Inheritance* denotes a superclass/subclass type of relationship where the subclass inherits all the attributes, operations, and relationships of the superclass. *Aggregation* is a form of association

in which the whole is related to its parts. The implementation of the IDMEF data model uses the eXtensible Markup Language (XML) Document Type Definitions (DTD). We use the definition and implementation method recommended by IDWG to describe the alert classes used by ZBIDS.

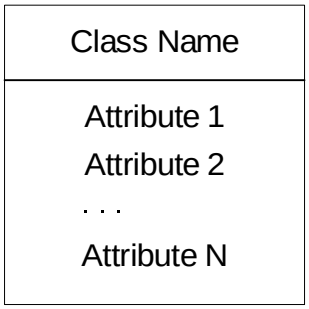


Fig. 11. Class Representation.

The IDMEF data model proposed by IDWG aims at wired IDSs. Its purpose is to define common data formats and data exchange procedures for sharing information of interest to intrusion detection systems. Due to the unique characteristics of MANETs and because we focus on the intrusion detection targeted at the network layer, we modify the IDMEF data model when designing the alert class. This includes adding some new classes (Zone class, for example) and attributes related to MANETs, deleting some unwanted classes (User class, Process class, etc.) and attributes, and modifying the definition of some classes and attributes (Location attribute, etc.). The alert class hierarchy for ZBIDS is depicted in Fig. 12 using the UML notation.

The alert class hierarchy depicted in Fig. 12 is general in ZBIDS. That is, it can be used as both the input and output of the LACE and GACE for better interoperability. When generating an alert, the detection engine formats it according to the class hierarchy depicted in Fig. 12. We also implement each alert class using

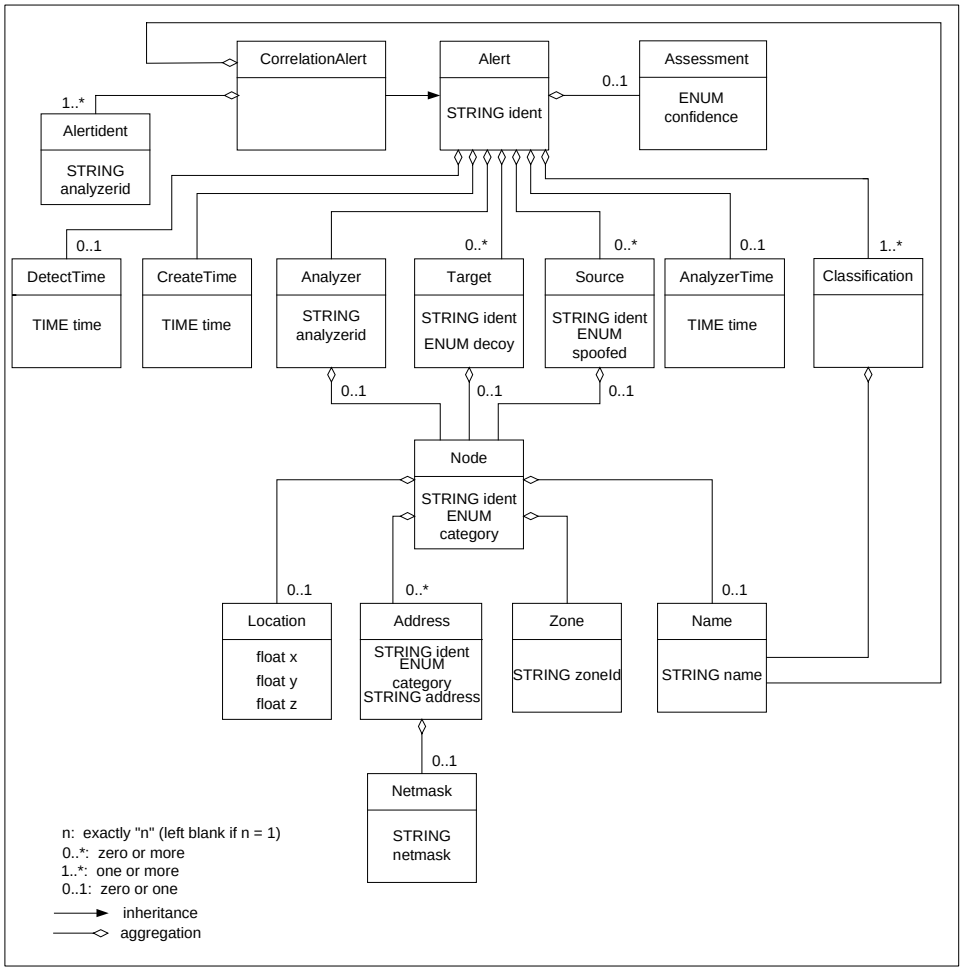


Fig. 12. The Alert Class Hierarchy of ZBIDS.

XML DTD, as shown in Appendix B. In this section, we generate all of our local alerts compliant with this format.

b. Aggregation Algorithm

The performance of the aggregation algorithm depends heavily on the performance of the local detection model, the amount of information and the accuracy of the information it provides. Existing aggregation algorithms assume the accurate information

provided by the local IDSs, and the aggregation algorithms are *alert-triggered* [37] [38] [39], i.e., the aggregation algorithms are triggered whenever a new alert is received. Our cases are different. First, due to the lack of misuse based MANET IDSs, we cannot assume the accurate identification of the attackers provided by the local IDSs. Second, the gateway nodes execute the aggregation algorithm periodically. At each time period, the aggregation algorithm aggregates the received local alerts and makes final decisions. If there is no alert received in the last period, no action is taken. This is computationally efficient since it avoids executing the algorithm every time an alert is received. The possible alert burst may crash the gateway nodes.

Old gateway nodes can locally broadcast historical records. This can lead to the quick learning of new gateway nodes and thus quick response to intrusion, but requires more bandwidth cost. Also, a new gateway node can obtain information quickly from local IDSs from their locally broadcasted alerts.

Each node has the LACE and GACE module. They use different sources as the alert inputs: the input of the LACE is the local detection engines, while the input of the GACE is either the local LACE (to intrazone nodes) or the intrazone nodes in the same zone and the neighboring gateway nodes (to gateway nodes).

When a local node detects an anomaly, it could generate an alert based on the proposed MANET IDMEF data model. This alert could contain the identification of the node, the alert classification, the time information, and the information of the routing control packets in the recent history that could contribute to the local alerts. The routing control packets in a given time interval is not sufficient for the intrusion detection. To our attack model, the local alert also includes the local history of the aggregated routing control packets, i.e., how many routing control packets are received and from which node the control packets are sent out. This could help the gateway node make the final decision.

Much information could be provided by the local alerts. The gateway node makes decisions mainly based on the following information from the local alerts:

- Classification similarity: In the IDMEF format, the attack type description is provided in the entity *classification*. In our context, two alerts are aggregated only if their classification fields are the same, indicating the same occurrence of attack. In our context, it should be "*Routing_Disruption*", which indicates the routing disruption attack.

- Time similarity: Each local alert provides the information of routing control packets in the past history. The entity *DetectTime* and *CreateTime* of IDMEF could be used to provide time information. *DetectTime* indicates the time when the attack happens, and $CreateTime$ indicates the time when the attack is detected. If the temporal difference between the $CreateTime$ of a newly received local alert and the time of the gateway node exceeds some predefined delay, this local alert is ignored.

- Source similarity: the *source* of the IDMEF data format indicates the possible sources of the attack. In the context of our attack model, it is the IP address of the attacker that actively propagates randomly constructed routing control packets.

Source similarity plays an important role in the alert aggregation. In the normal routing discovery procedure, if the intermediate nodes have the route to the destination, they could generate a RREP packet back to the traffic source. All nodes that receive the route packets modify its routing cache correspondingly. For each received or promiscuously heard RREP packet, the node records its source and destination IP addresses. If there is no attacker in the network, the distribution of these source

addresses in a given time period would be expected even, i.e., most of the time, there is no bias for a given source address.

We introduce a parameter $P_{routing\_abnormal}$. If the proportion of routing control packets from a certain address exceeds $P_{routing\_abnormal}$, it is abnormal and deserves further investigation. We experimently set $P_{routing\_abnormal}$ in our later experiments.

However, in normal cases, it is still possible that a node receives a high percentage of routing control packets from some certain node in a period. This is reasonable if we consider the route discovery procedure of DSR. This situation is enhanced by enabling promiscuous-listening mode and is the main reason to cause the false positive alarms of our aggregation algorithm.

When there exist attackers in the network, things are different. The attacker would send many falsified routing control packets into the network. The local IDSs of the victims, using the Markov Chain detection model described in the previous section, could generate the alerts and record the source and destination distribution of the routing control packets in the last period. The attacker's address would dominate the source distribution of the routing control packets. Having gathered this information in the last period, the gateway nodes could know the source address distribution of the routing control packets. If the probability of a particular source address exceeds some predefined threshold P, this address is then identified as the attacker's address. Note that an attacker cannot use different IP addresses to send out fake messages. Otherwise, it can be detected easily by its neighbors.

We now discuss how to decide P. The selection of P depends on attack intensity, attacking time, node placement, etc. If the threshold P is low, the gateway nodes could identify the attack more accurately, thus achieving higher detection ratios. However, this could lead to high false positive ratios. If the threshold P is high, the gateway nodes could miss the attack, but reduce the false positive ratio. We propose

a simple approach to decide P in the following way.

In normal cases, for a given gateway node, if local alerts are received in a given time period, we first pick those source addresses whose aggregated probability is larger than $P_{routing\_abnormal}$. We denote these probabilities as P_{t_i} $(i = 1, 2, \ldots, n_t)$.

Suppose for a given gateway node G, it has m_t time periods in which it receives local alerts, we compute the average of $P_{t_i}, (i = 1, 2, \ldots, n_t)$ over these m_t periods as:

$$P_{G_t} = \sum_{j=1}^{m_t} \sum_{i=1}^{n_t} P_{t_i}/m_t.$$

P_{G_t} represents, to gateway node G, the irregularity of the source address distribution of the routing control packets when the system is at normal status. Given a test trace, we compute its average over all gateway nodes:

$$P_{test} = \sum_{\forall \ gateway \ nodes} \frac{P_{G_t}}{the \ number \ of \ gateway \ nodes}.$$

Given the trace of intrusive activities, we first compute the attack address distributions contained in the routing control packets. We denote these probabilities as P_{a_i} $(i = 1, 2, \ldots, n_a)$.

Suppose for a given gateway node G, it has m_a time periods in which it receives local alerts, we compute the average of $P_{a_i}, (i = 1, 2, \ldots, n_a)$ over these m_a periods as:

$$P_{G_a} = \sum_{j=1}^{m_a} \sum_{i=1}^{n_a} P_{a_i}/m_a.$$

P_{G_a} represents the source address distribution of the routing control packets in the gateway node G during the attack time. Given the trace of intrusive activities, we compute the average of P_{G_a} over all gateway nodes:

$$P_{attack} = \sum_{\forall\ gateway\ nodes} \frac{P_{G_a}}{the\ number\ of\ gateway\ nodes}.$$

We set P as:

$$P = h_t * P_{test} + h_a * P_{attack},$$

where $h_t > 0$, $h_a > 0$, and $h_t + h_a = 1$. By adjusting h_t and h_a, we can get a better trade-off between the false positive ratio and detection ratio. In our later simulation study, h_t an h_a are set equal.

The pseudocode of setting P could be depicted in Fig. 13.

Having decided P, we can now describe the algorithm that a gateway node uses to determine whether it should generate alarms in a given time period. When a gateway node receives locally broadcast alerts in some period, it first sums up the aggregated probabilities of those source addresses whose probability is larger than $P_{routing\_abnormal}$. If the resultant value is less than P, the gateway node will not generate alarms. Otherwise, the gateway node will generate alarms and provide attacker information based on the probability distribution of source addresses.

It is possible that the detection sensitivity of the aggregation algorithm will decline with the increase of the size of the attack group. Especially when the attackers collude to attack at the same time and the attack objectives overlap. In this kind of situation, for the attack objectives, there is no single address that dominates the probability distribution of the source addresses of the routing control packets. This could impact the detection ratio of the aggregation algorithm. However, it is still possible that the attack victims do not overlap completely, whose attack address probability distributions contribute to the effectiveness of our aggregation algorithm. One example of this is depicted in Fig. 14.

Procedure Determine_P()
Input: test trace, attack trace, $P_{routing\_abnormal}$
Output: P
Begin
Test trace:
For each gateway node G
  $P_{Gt} = 0$;
  For each time interval of G that receives local alerts
    For all P_{ti}
      If $P_{ti} > P_{routing\_abnormal}$
      then $P_{Gt} = P_{Gt} + P_{ti}$
    End For
  End For
  /* m_t is the number of time intervals of G that receives local alerts */
  $P_{test\_sum} = P_{test\_sum} + P_{Gt}/ m_t$
End For
/* N_{test} is the number of gateway nodes that receive local alerts */

$$P_{test} = \frac{P_{test\_sum}}{N_{test}}$$

Attack trace:
For each gateway node G
  For each time interval of G that receives local alerts
    Compute the sum of the probability of attacker source addresses P_{Ga}
  End For
  /* m_a is the number of time intervals of G that receives local alerts */
  $P_{attack\_sum} = P_{Ga} / m_a$
End For
/* N_{attack} is the number of gateway nodes that receive local alerts */

$$P_{attack} = \frac{P_{attack\_sum}}{N_{attack}}$$

$P = h_t * P_{test} + h_a * P_{attack.}$
END

Fig. 13. Pseudocode of How to Decide P.

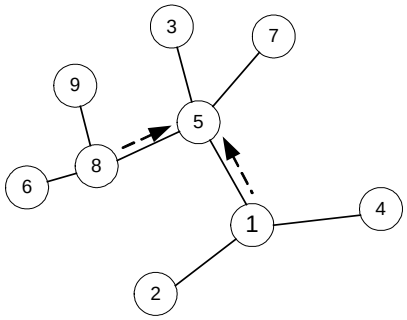


Fig. 14. One Example of Two Attackers.

In this example, node 1 and 8 are attackers, and the attack objective is node 5. The probability distributions of 1 and 8 provided by node 5 would make it difficult for the gateway node to detect the attackers. However, since the victims do not overlap (the victims of node 8 are 5, 6, and 9, while the victims of node 1 are 2, 4, and 5), they all provide information of 1 and 8. This could contribute to the detection of the attack.

G. The Relation between Intrusion Detection and Intrusion Prevention

In MANETs, intrusion prevention and intrusion detection techniques need to complement each other to guarantee a highly secure environment. They play different roles in different status of the network.

Intrusion prevention measures, such as encryption and authentication, are more useful in preventing outside attacks. Considerable research has been done in preventing the misbehavior at the network layer.

Once the node is compromised, however, intrusion prevention measures will have little effect in protecting the network. At this time, the role of intrusion detection is more important. In mobile ad hoc networks, it is much easier to gain physical

possession of the node. When a node is compromised, the attacker owns all its cryptography key information. Therefore, encryption and authentication cannot defend against a trusted but malicious user.

Intrusion detection research assumes that subject activities are observable, and normal and intrusive activities have distinct behavior. Therefore, by identifying the different behavior manifested by attackers, intrusion detection systems, serving as the second wall of defense, could provide a complementary security mechanism to MANETs.

Intrusion detection and intrusion prevention are not totally separated. For example, it is possible that, in ZBIDS, the local IDS encrypts the generated local alert and broadcast them to the gateway nodes using the network-wide shared secret among the local IDS agents. This could prevent some compromised nodes from fabricating alerts and enable the correct functionality of GACE.

H. Summary

In this chapter, we describe a Markov Chain based anomaly detection algorithm and an aggregation algorithm for MANETs. Based on the locally collected statistical measures that reflect the mobility of the network, a Markov Chain is constructed to act as the normal profile, which is then used to build a classifier. Ordering property is considered and the transition probability is used to define the distance of the trace and the normal profile. The aggregation algorithm could further reduce the false positive ratio and increase the detection ratio, which are demonstrated in the next chapter.

CHAPTER IV

PERFORMANCE EVALUATION OF ZBIDS

In order to study the feasibility and effectiveness of our detection algorithm and validate our detection model, we carried out extensive simulation experiments using various mobility scenarios. In this chapter, we first describe our simulation approach and then present the simulation results of local IDS and ZBIDS respectively.

A. Simulation Model

1. Simulation Platform and Parameter Settings

We use a simulation model based on Parsec [4] and GloMoSim [5] to investigate the performance of the proposed approaches. We choose DSR as the routing protocol and the parameters used in the simulation are described in Appendix III.

Specifically, in our simulation, the channel capacity of mobile hosts is set to the same value: 2 Mbps. We assume all nodes have the same transmission range of 250 meters. A free space propagation model with a threshold cutoff is used as the channel model. In the free space model, the power of a signal attenuates as $1/r^2$, where r is the distance between mobile hosts. In the radio model, capture effects are taken into account. We use the Distributed Coordination Function (DCF) of IEEE 802.11 for wireless LANs as the MAC layer protocol. It has the functionality to notify the network layer about link failures. We enable the promiscuous receive mode of nodes, which enables every received packets delivered to the network layer.

In the simulation, 30 mobile nodes move in a 1000 meter X 500 meter rectangular region. Compared with a square region, the rectangular region can enlarge the average route length so that we can easily observe the performance difference in different

scenarios. We assume each node moves independently with the same average speed. Except otherwise indicated, the mobility model we use in this chapter is the Random Waypoint model. In this mobility model, a node randomly selects a destination from the physical terrain. It moves in the direction of the destination in a speed uniformly chosen between the minimal speed and the maximal speed. After it reaches its destination, the node stays there for a pause time and then moves again. In our simulation, the minimal speed is 3 m/s, and the maximal speed is 5 m/s. We change the pause time from 30 seconds to 900 seconds to investigate the performance influence at different mobilities.

8 source-destination pairs are selected randomly to generate Constant Bit Rate (CBR) traffic as the background traffic. The interval time for data transmission is 0.25 second. The size of all data packets is set to 512 bytes. A packet is dropped when no acknowledgement is received after seven retransmissions or when there is no buffer to hold the packet. The buffer size is set to 128 packets. All traffic is generated, and the statistical data are collected after a warm-up time of 300 seconds in order to give the nodes sufficient time to finish the initialization process. When we simulated a routing disruption attack, the attacker is uniformly chosen from the 30 nodes.

2. Data Sets

The hardware we use to collect the data sets is SGI Origin 2000 with a 32-processor, 8 GB distributed-shared-memory. The operating system of it is IRIX 6.5.

In general, three kinds of data need to be generated: training data, testing data, and intrusion data. We execute the application in as many normal modes as possible while tracing its behavior. Specifically, we use different pause time (30S, 150S, 300S, 600S, 900S) to represent different mobility scenarios. At each mobility level, we randomly select 4 different seeds. For a given mobility scenario and a given random

seed, we run the simulation 400 minutes in order to get the normal data. In this way, we get $5 \times 4 = 20$ normal data traces. For each of the data trace, we collect (PCR and PCH) feature values every 3 seconds after a warm-up time period of 300 seconds. We mix the same-mobility normal data of different nodes and different random seeds together to generate the normal trace for a given mobility. From each normal data trace, we use its last 40 minutes part as the testing data. The rest of the normal data are used as the training data, and they are used to construct the Markov Chain model and the classifier. Thus, altogether we have 20 training data and testing data traces. For each mobility scenario, we have 4 training data and testing data respectively. Each training data trace has 7074 data items and each testing data trace has 786 data items.

Because this research focus on the routing disruption attacks, we also simulate this type of attack in order to get data of intrusive behaviors. Under the same mobility scenario, we let the simulation run 10 minutes. For each run, we let the routing attack script start at 500S, and the attack lasts 60S.

Training data are used to construct the local Markov Chain model. Testing data and intrusion data are used to tune the parameters of the local classifier and the aggregation algorithm. In order to evaluate the performance of ZBIDS, we further generate a different set of normal and abnormal data.

Due to the mobility of nodes, it is possible that a node is only a "partial" victim during the whole intrusion session. That is, the node only receives or promiscuously hears part of the falsified routing control packets because of mobility-caused link breakage. We use data traces of all of the victims, including both the "partial" victims and "full" victims, to represent the intrusive behavior.

We use the LBG-VQ algorithm [73] to discretize the raw continuous data in order to construct the Markov Chain model. Given each mobility scenario, feature

value *PCR* and *PCH* are used respectively to form one-dimensional vector sources. A proper selection of the codebook size is important. If it is selected too small, too many errors will be introduced in the data transformation process because the bin size would be too large. If it is selected too large, the states in the constructed Markov Chain model will be huge. Through experiments, we categorize the data under different mobility into 32 distinct values (the size of the codebook is 32) and find this value provides a good trade-off.

The dynamic nature of MANETs makes it possible that some categorized routing change values have a very small probability. For example, it is possible that a node receives many RREPs and/or RERRs in some data collection period, which in turn, trigger an abnormal but not malicious routing cache changes. The existence of these kinds of data is undesirable in the construction of the Markov Chain model and should be filtered out. We deal with this problem in the following way: if the probability of one data item is less than a very small value, such as 0.01 (this value is observed through simulation), we convert it into a "rare" symbol. In this way, the abnormal but not malicious routing cache change values are "aggregated" into a common symbol.

The construction of the Markov Chain model takes a few hours for each mobility level. Generally, the higher the mobility, the longer the training time, and the more states we will obtain. When the mobility is low, we expect less changes of the routing cache. In fact, when the pause time is very large (900S, for example), for a long time, the routing table changes are 0. This also speeds up the construction of the detection model.

When constructing the classifier using the Markov Chain model, we experimentally set the size of the locality frame to 40. This corresponds to the data history in the last 120S.

3. Performance Metrics

We use three metrics to evaluate the performance of our local detection model. In addition to the traditional *accuracy* (false positive ratio and detection ratio) measurements, the *Mean Time to the First Alarm* (MTFA) and the *communication overhead* are also considered.

- **False positive ratio**: As to local IDSs, it is reported for normal data not used during offline training process and is computed from dividing the total number of false alerts by the total number of transactions in the normal data.

  Traditional wired IDSs are often built using the short sequence of system calls of priviledged programs or constructed connection information. Therefore, they can treat one program execution or one connection as a trace, decide whether each trace is normal or abnormal, and compute the *false positive ratio* correspondingly.

  We are focusing on the network layer and there is no concept of connection here. Therefore, we treat the subject (here is the routing table) activities over the past *locality frame* as one trace.

  In the process of scanning a test trace, when the *alert signal* of the past *locality frame* is above the tuned *alert threshold*, we count it as one *false alert*. Making a single decision as to whether a normal trace appears anomalous or not is not sufficient, especially for long traces. We thus define *false positive ratio* as the percentage of decisions in which normal data are flagged as anomalous.

  That is, for a normal trace α, its length is denoted as $|\alpha|$. The length of the *locality frame* is denoted as L. Let $alert(\alpha)$ denote the number of alerts that the local IDS generates over α. Then *false positive ratio* of α is defined as:

$$\frac{alert(\alpha)}{|\alpha| - L + 1}.$$

Let T_t denote the normal trace set. We compute the *false positive ratio* of each trace $\alpha \in T_t$, denoted as $f_1, f_2, \ldots, f_n$. n is the number of traces in T_t. Then *false positive ratio* of T_t is defined as :

$$\frac{f_1 + f_2 + \ldots + f_n}{n}.$$

As to the aggregation algorithm, it is defined as the percentage of decisions in which normal alert aggregations are flagged as anomalous. Gateway nodes need to execute the aggregation algorithm periodically. For a gateway node, at each t_i when it receives local alerts, it needs to make a decision (whether to generate an alarm or not). When there are no attackers in the network, suppose one gateway node makes n decisions, and generate m alarms, its *false positive ratio* is defined as $\frac{m}{n}$. The *false positive ratio* of the aggregation algorithm is the average of the *false positive ratio* of all gateway nodes.

- **Detection ratio**: As to local IDSs, it is reported for traces of intrusive behavior and is computed from dividing the total number of correct detections by the total number of victims in the anomalous data. Any above *alert-threshold* signal anywhere in the intrusive traces counts as a correct detection of the intrusion.

  Let T_a denote the intrusive trace set. The number of traces in T_a is denoted as $|T_a|$. For each trace $\alpha \in T_a$, the local IDS needs to make a decision whether it is normal or abnormal. Let $alert(T_a)$ denote the number of alerts that the local IDS generates over T_a. *Detection Ratio* is defined as:

$$\frac{alert(T_a)}{|T_a|}.$$

As to the aggregation algorithm, it is reported for traces of intrusive behavior and is computed from dividing the total number of gateway nodes raising correct alarms by the total number of gateway nodes which should raise alarms in the anomalous data.

When there are attackers in the network, if a gateway node receives local alerts from some victim's IDS, it should generate alarms. Suppose there are n gateway nodes which receive local alerts from some victim's IDS, and m of them generate alarms, *detection ratio* is defined as $\frac{m}{n}$.

- **MTFA**: This metric is defined over anomalous traces and measures how fast the classifier detects the attack. It is desirable that the IDS detect the attack as quickly as possible. Given an anomalous trace ξ, suppose the attack starts location is L_a, our IDS generates its first alert after scanning the L_d-th symbol, then the $MTFA$ corresponding to ξ normalized by the length (denoted as L) of the locality frame is given by $MTFA(\xi) = (L_d - L_a)/L$. We measure MTFA over the anomalous trace set T_a as:

$$\frac{\sum_{\xi \in T_a} MTFA(\xi)}{|T_a|}.$$

- **Communication Overhead**: The communication overhead is computed as the number of transmission of local alerts in a given time period for one node. It is mainly introduced by propagating the local alerts of intrazone nodes in ZBIDS.

For each mobility level, if the variance of one performance metric (Detection ratio, MTFA, for example) is small, we only calculate its average result and do not draw its confidence interval. If its variance (false positive ratio, for example) is large, we randomly select 10 traces - that is, 10 runs of the simulation, and calculate the average result as well as the 95% confidence interval. The confidence interval is determined by the following formula:

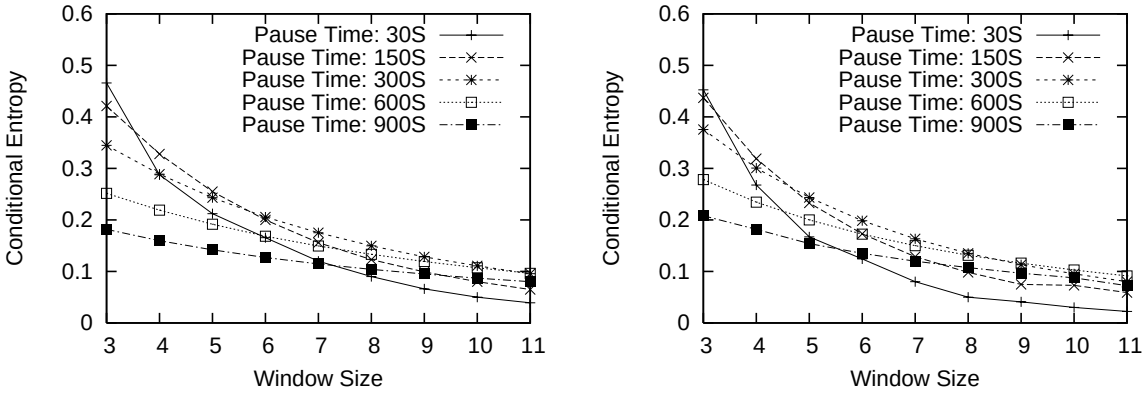
$$P(\bar{X} - \frac{bs}{\sqrt{n-1}} < \mu < \bar{X} + \frac{bs}{\sqrt{n-1}}) = 0.95$$

where X is the random variable, n is the number 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 and the probability distribution (we use t-distribution). Then the interval $[\bar{X} - \frac{bs}{\sqrt{n-1}}, \bar{X} + \frac{bs}{\sqrt{n-1}}]$ is 95% confidence interval.

For the detection ratio and false positive ratio, we also use different *alarm threshold*, namely, 0.8*r, r, and 1.2*r, to watch the different performance values.

4. Parameter Tuning

We compute the conditional entropy of the normal data at different mobility levels in order to decide the proper *window size* w that characterizes the Markov Chain model. We change w from 3 to 12 in order to determine the desirable w for different statistical measures. To compute $H(X|Y)$ (suppose Y is in the form $(e_1, e_2, \ldots, e_w)$ and X is in the form $(e_1, e_2, \ldots, e_w, e_{w+1})$) for training data at the same mobility level and the given w, two scans of the training trace are required. The first scan records the unique appearance of y and its probability distribution, while the second scan computes the marginal probability distribution of x given y. In this way, we can get the conditional probability $P(y|x)$. Based on Definition 2 in Chapter III, we can then compute the *conditional entropy* of training data. From the conditional entropy of



(a) Feature PCH (b) Feature PCR

Fig. 15. Conditional Entropy of Feature *PCH* and *PCR*.

different measures, we decide the value of w so that the conditional entropy does not drop dramatically.

We use the determined w to construct the Markov Chain model. Two more important parameters are the *penalized value* z when a transition is not found in the Markov Chain model and the *alarm threshold* r. We experimentally determine them in the way as we have stated in the previous chapter. We tune the parameters until $(D_a(T_a) - D_t(T_t)) \geq 1$. Typical value of r is between 1 and 2.

B. Simulation Results of Local IDSs

1. Conditional Entropy

Fig. 15 illustrates the *conditional entropy* of feature *PCH* and *PCR* of training data at different mobility levels when the sequence length varies from 3 to 11 with an increase of 1. Each line here represents training data at the same mobility level. We can see from the simulation results that audit data under different mobility has different regularity. Audit data under high mobility is more irregular because of more

unexpected changes, and this is the reason that will lead to high false positive ratios in detection.

We can see that the *conditional entropy* drops as the *window size* increases. This is because the larger the *window size*, the more information is included in the detection model, thus the less uncertainty remained in the audit data. We can also observe that the *conditional entropy* does not drop dramatically when the *window size* reaches 4 or 5. This motivates us that if we set the *window size* to 4 or 5, the predicted state is highly deterministic if we consider the trade-off involved with much longer *window size* values.

We also notice that when the *window size* is set to 4 or 5, with the decrease of mobility (i.e., the increase of the pause time), the *conditional entropy* also decreases. This implies that the data of lower mobility is more regular compared to data of higher mobility. Actually, a closer look at the raw data shows that when mobility is very low (the pause time is 900S, for example), the network topology is relatively stable. Therefore many data items are 0, indicating no changes of the routing caches. This is one of the main reasons why the performance of MANET IDSs of lower mobility is better than that of higher mobility in terms of defined performance metrics.

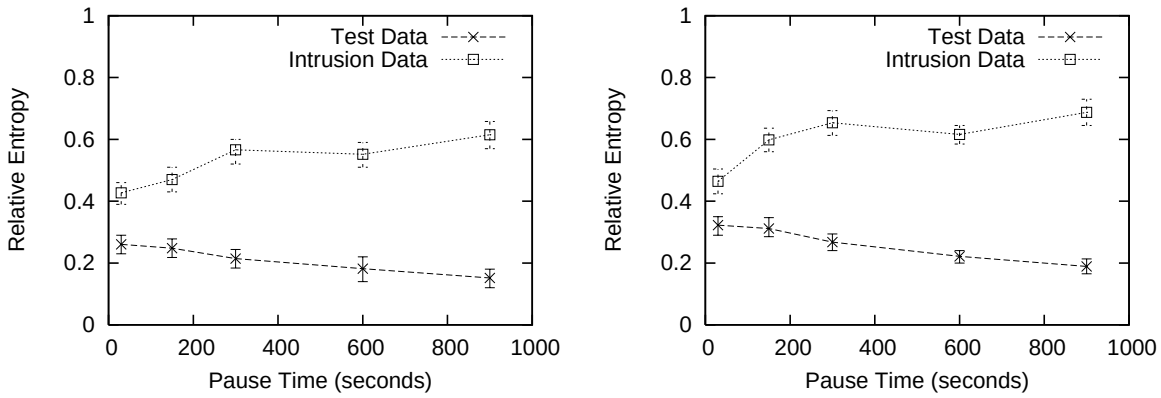
Therefore, in the following, we set w to 4 and 5 respectively to build the Markov Chain model and the classifier and watch their performance. Note that we have performed the simulation when w is set to 6 and 7. The simulation results fit the general trend and discussion described in the following.

a. Relative Entropy

In order to demonstrate the potential effectiveness of the adopted features, we measure the RE_{test} and $RE_{intrusion}$ of PCH and PCR using two different mobility models, the Random Waypoint model and the Random Drunken model. In the Random

Drunken mobility model, each node moves independently with the same average speed. Each node moves continuously within the region without pausing at any location. It changes direction, randomly chosen, after every unit of distance.

Fig. 16 and Fig. 17 illustrate RE_{test} and $RE_{intrusion}$ of feature *PCH* and *PCR* using the Random Waypoint model and the Random Drunken model respectively. We set λ of Jelinek-Mercer smoothing method to be 0.9, which is a commonly used constant.

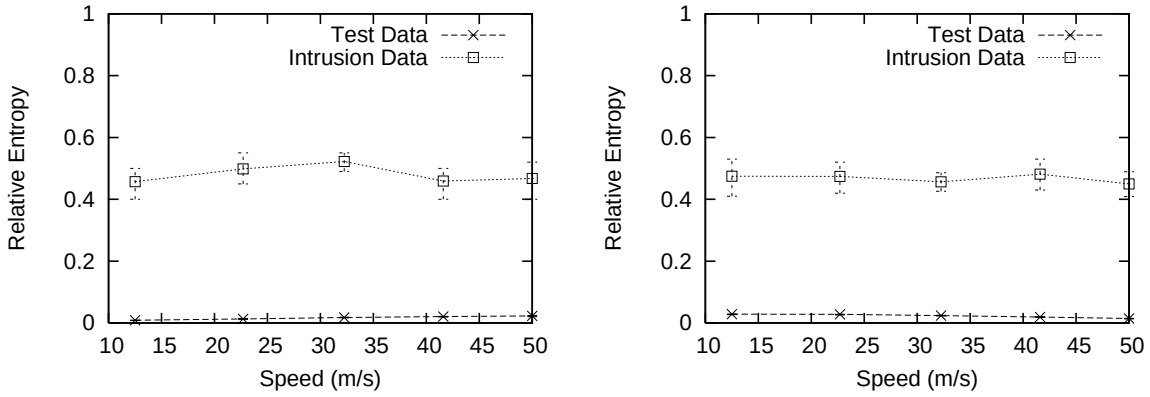


(a) Feature PCH. (b) Feature PCR.

Fig. 16. Relative Entropy of Feature *PCH* and *PCR* Using the Random Waypoint Model.

We can see from simulation results that RE_{test} is smaller than $RE_{intrusion}$. This suggests that *PCH* and *PCR* are suitable features and can be used to construct anomaly detection models.

We can also see that audit data at different mobility has different RE_{test}. Audit data under high mobility is more irregular, therefore, when mobility is high, RE_{test} is larger and the difference between RE_{test} and $RE_{intrusion}$ is smaller. This explains from one aspect why the performance of anomaly detection under high mobility is worse.



(a) Feature PCH. (b) Feature PCR.

Fig. 17. Relative Entropy of Feature *PCH* and *PCR* Using the Random Drunken Model.

2. False Positive Ratio

False positive ratio of local IDSs using different features (PCR and PCH) is illustrated in Fig. 18, Fig. 19, Fig. 20, and Fig. 21. As we can see, for each plot, the false positive ratio increases with the decrease of the *alarm threshold* r. This is as what we have expected. When the *alarm threshold* decreases, it is easier for the *alarm signal* of the normal trace to exceed r, thus generating alarms.

We can also see that in each figure, the false positive ratio decreases with the decrease of mobility, because as we have shown in the previous section, the trace of lower mobility demonstrates higher regularity. When mobility is low, their normal routing table changes are less dramatic and have less unexpected values. This makes it easier and more accurate for the Markov Chain model to characterize its normal behavior. Due to low mobility, the testing trace also has less unexpected changes. This would contribute to the low false positive ratio.

Comparing Fig. 18 and Fig. 19, we can see the slight increase of the false positive ratio shown in Fig. 19, which corresponds to *window size* 5. This could be explained

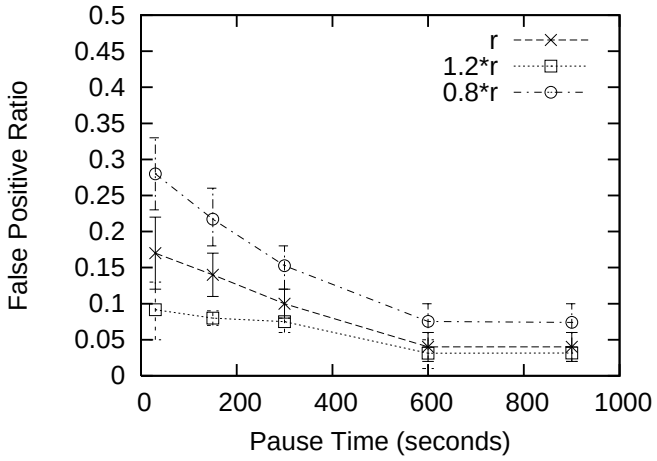


Fig. 18. False Positive Ratio of the Local IDS Constructed Based on Feature *PCH* When Window Size Is Set to 4.

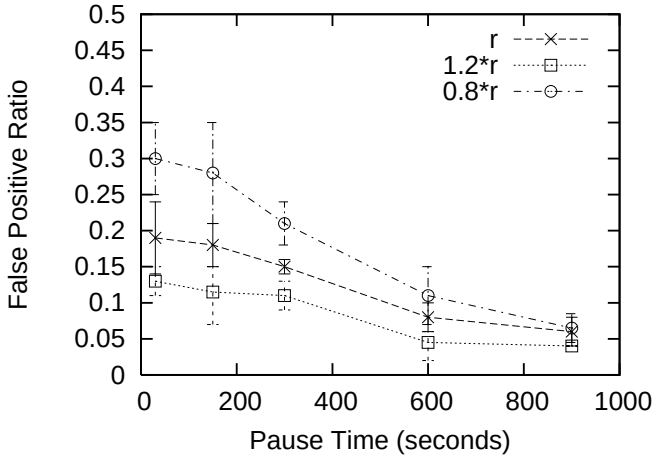


Fig. 19. False Positive Ratio of the Local IDS Constructed Based on Feature *PCH* When Window Size Is Set to 5.

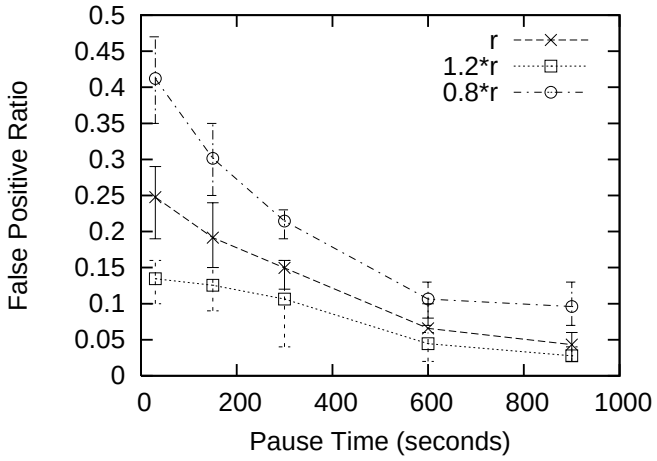


Fig. 20. False Positive Ratio of the Local IDS Constructed Based on Feature *PCR* When Window Size Is Set to 4.

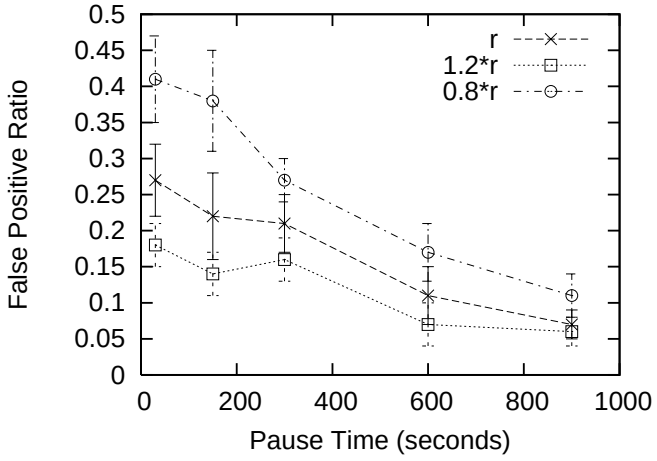


Fig. 21. False Positive Ratio of the Local IDS Constructed Based on Feature *PCR* When Window Size Is Set to 5.

as: with the increase of the *window size*, the Markov Chain model characterizes the normal behavior of routing cache changes more accurately because more history information is included. This will result in a larger probability to generate false alerts because the small fluctuation of the normal behavior could lead to false alerts generated by the classifier with larger window size. That is, the detector with larger window size is more sensitive to unexpected abnormal changes. The same is true when we compare Fig. 20 and Fig. 21.

Comparing Fig. 18 and Fig. 20, we can see that the classifier constructed using the feature *PCR* results in a larger false positive ratio compared to the classifier constructed using the feature *PCH*. This demonstrates that *PCH* is better than *PCR* in terms of false positive ratios. Because each entry of the DSR routing cache contains a full path to the destination, *PCH* considers not only the change of the number of routes, but also the change of the length of each routing entry. Therefore, *PCH* contains more information compared to *PCR*. This makes it more accurate to be utilized to characterize the normal behavior of routing behavior. The same is true when we compare Fig. 19 and Fig. 21.

3. MTFA

Simulation results of $MTFA$ of local IDSs are depicted in Fig. 22, Fig. 23, Fig. 24 and Fig. 25.

As we can see, with the increase of the *alarm threshold* r, $MTFA$ increases. Because with a larger *alarm threshold*, the detector needs a longer malicious trace in the current *locality frame* to make the *alarm signal* exceed the *alarm threshold*. This will result in a larger MTFA. However, from simulation results, the trend of $MTFA$ is not very obvious with respect to the change of mobility.

Comparing Fig. 22 and Fig. 23, we can see the slight increase of *MTFA* shown

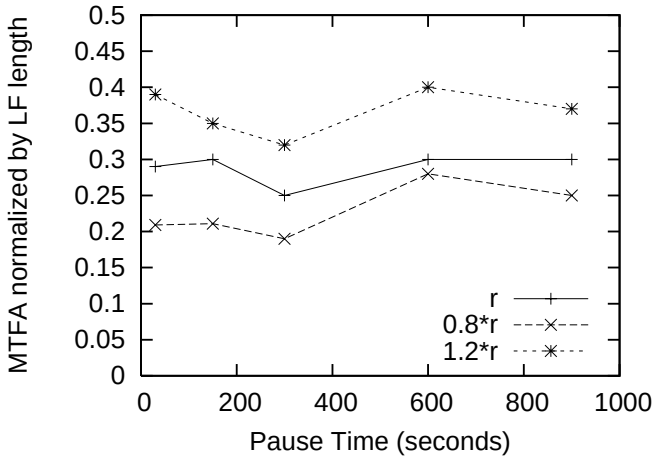


Fig. 22. MTFA of the Local IDS Constructed Based on Feature *PCH* When Window Size Is Set to 4.

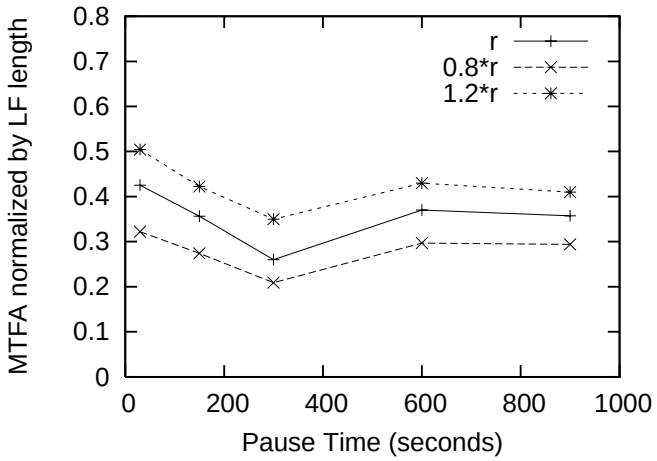


Fig. 23. MTFA of the Local IDS Constructed Based on Feature *PCH* When Window Size Is Set to 5.

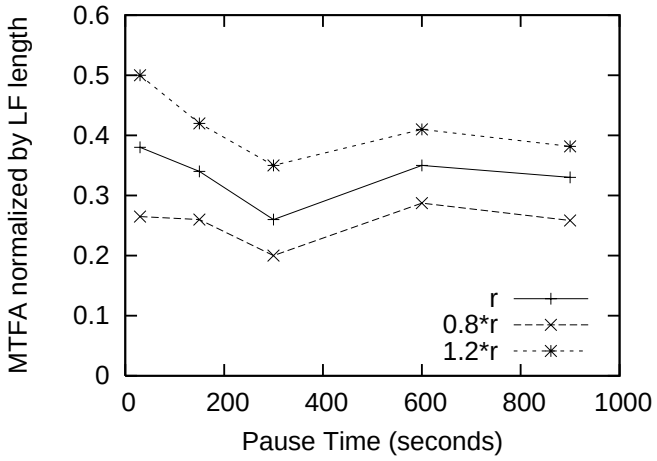


Fig. 24. MTFA of the Local IDS Constructed Based on Feature *PCR* When Window Size Is Set to 4.

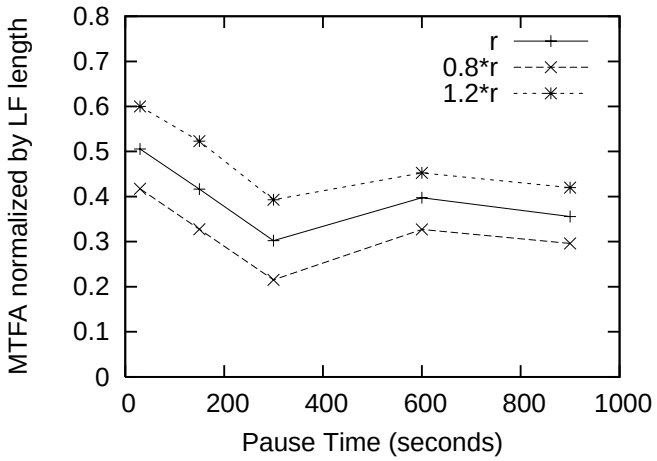


Fig. 25. MTFA of the Local IDS Constructed Based on Feature *PCR* When Window Size Is Set to 5.

in Fig. 23, which corresponds to *window size* 5. This could be explained as: When w increases, the Markov Chain model could characterize the behavior more accurately. A transition which is valid in the Markov Chain model with a small window size could become invalid when w becomes larger. This could contribute to a larger r in the context of our parameter tuning approach. Therefore it needs a longer history for the *alert signal* to exceed r. The same is true if we compare Fig. 24 and Fig. 25.

Comparing Fig. 22 and Fig. 24, we can observe that MTFA of *PCR* shows a larger value compared to that of *PCH*. Normal profile constructed using *PCH* makes the distinction between the normal behavior and the abnormal behavior easier compared to that constructed using *PCR*. The same is true if we compare Fig. 23 and Fig. 25.

4. Detection Ratio

Simulation results of the detection ratio of local IDSs are illustrated in Fig. 26, Fig. 27, Fig. 28 and Fig. 29.

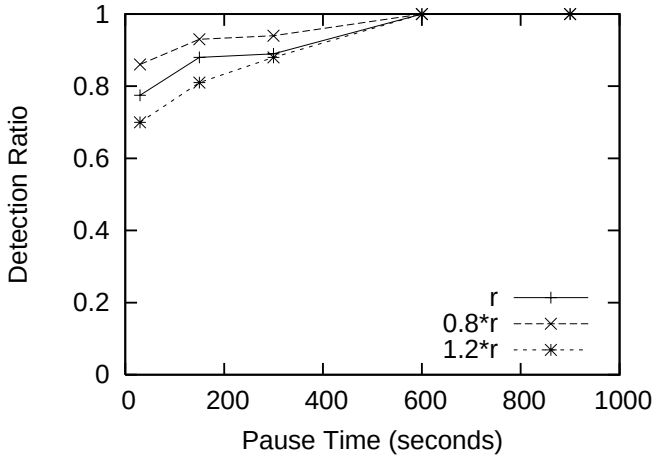


Fig. 26. Detection Ratio of the Local IDS Constructed Based on *PCH* When Window Size Is Set to 4.

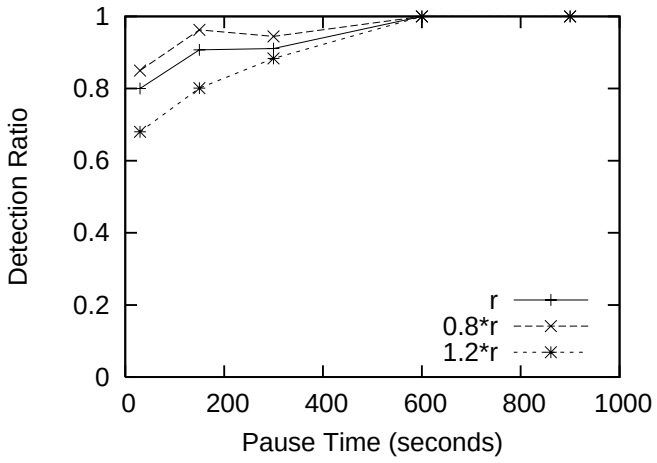


Fig. 27. Detection Ratio of the Local IDS Constructed Based on *PCH* When Window Size Is Set to 5.

As we can see, in each figure, the detection ratio increases with the decrease of the *alarm threshold* r. Because when the *alarm threshold* decreases, it is easier for the *alarm signal* of the normal trace to exceed it, thus generating alarms. When mobility is low, routing table changes are less dramatic and has less unexpected changes. Thus it is easier for the classifier to identify the abnormal behavior.

We also observe that in each figure, the detection ratio increases with the decrease of mobility. As we have shown in the previous section, the trace of lower mobility demonstrates higher regularity. When the mobility is low, their routing table changes are less dramatic and has less unexpected changes. Thus it is easier for the classifier to identify the abnormal behavior.

When mobility is high, the detection ratio is relatively low. This is mainly caused by "partial" victims. We use the data traces of all of the victims, including the "partial" victims, as the data of intrusive behavior. Some "partial" victims only receive a few falsified routing control packets during the whole intrusion session.

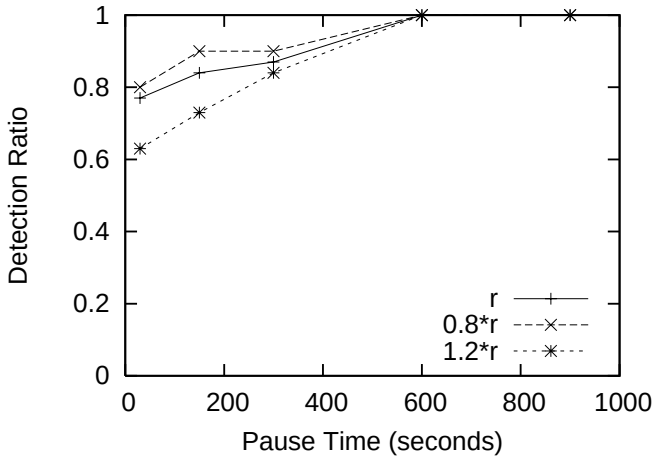


Fig. 28. Detection Ratio of the Local IDS Constructed Based on *PCR* When Window Size Is Set to 4.

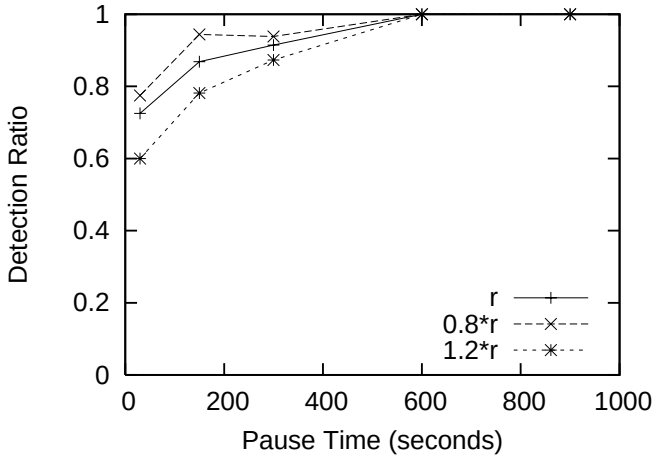


Fig. 29. Detection Ratio of the Local IDS Constructed Based on *PCR* When Window Size Is Set to 5.

It is thus very difficult for the detector to distinguish between normalcy caused by temporary mobility-induced errors and anomaly caused by one or two falsified routing control packets. The situation becomes worse when mobility is high because of the quick link breakage. When mobility is low, however, the local IDSs can achieve very good detection ratios.

Comparing Fig. 26 and Fig. 27, when w increases, the detection ratio corresponding to the same mobility also increases slightly (except at low mobility level, when the detection ratios are already very high). However, there exist some exception points. The slight increase of the detection ratio could be explained as: When w increases, the Markov Chain model could characterize the normal behavior more accurately and thus detect more subtle abnormal changes of the routing caches. This could contribute to the increase of the detection ratio. In our detection model, we notice that an abnormal transition that is not a valid transition in the Markov Chain model with a small window size is not a valid transition in the Markov Chain model with larger window size either. However, note that an abnormal transition that is not a valid transition in the Markov Chain model with a larger w could be a valid transition in the Markov Chain with a smaller w. The slight decrease of the detection ratio could be explained as: When w increases, the *alert signal* could also increase. This would make it more difficult to detect the attackers. Therefore, we observe a trade-off here. The same is true if we compare Fig. 28 and Fig. 29.

Comparing Fig. 26 and Fig. 28, the classifier constructed using *PCH* results in a larger detection ratio compared to the classifier using *PCR*. This demonstrates that *PCH* is better than *PCR* in terms of detection ratios. The reason is similar to when we come to the issue of false positive ratios. Normal profile constructed using *PCH* contains more information of DSR routing caches, and is thus more accurate to characterize routing activities. The same is true if we compare Fig. 27 and Fig. 29.

5. Discussion

Simulation results of the local IDSs demonstrate that it is viable to construct a Markov Chain model to characterize the normal behavior of routing cache changes and build a classifier based on the data model. This classifier works well in environment with low mobility.

The detection algorithm aggregates the mismatch counts in the recent *locality frame* and thus is not sensitive to a single mismatch. This is important in MANET environment because of the variations involved in MANET activities due to its dynamic nature. In this way, we could avoid high false positive ratios, especially in environment with low mobility.

We could also see that the feature *PCH* is better than *PCR*. Simulation results show that classifier constructed using *PCH* demonstrates better performance than that constructed using *PCR* in terms of both false positive ratios and detection ratios. This is because feature *PCH* considers not only the number of routing entries, but also the content of each routing entry. Therefore it is easier and more accurate to characterize the normal behavior of routing cache changes.

A Receiver Operating Characteristic (ROC) [82] curve plots pairs of false positive ratio and detection ratio as points when various signal thresholds are used. We do not plot the ROC curve of the local IDSs as most researchers have done ([24], [25], [23], [22], [**?**]) when describing the performance of wired IDSs, because we find the performance of the local IDSs is sensitive to mobility. Actually, at the same mobility, there is a gradual trade-off between false positive ratios and detection ratios: the false positive ratios increase with the increase of detection ratios.

C. Simulation Results of ZBIDS

We notice in the previous section that local IDSs constructed using feature *PCH* demonstrates better performance results than those constructed using feature *PCR*. Therefore, in this section, we use the classifier constructed using *PCH* as the local IDS to evaluate the performance of ZBIDS. The *window size* of the local IDS is set to 4.

1. False Positive Ratio

We can see that the performance of local IDSs at high mobility level is not desirable. In particular, its false positive ratio is still high. For a realistic detection system, it is important that the false alarm ratio remains low. Simulation results of ZBIDS in this section illustrate that ZBIDS and the aggregation algorithm are effective in reducing false positive ratios.

We compute the false positive ratio of the aggregation algorithm based on the same test data used by the local Markov detection model for the purpose of comparison. If in the last time period, the gateway node receives no local alerts, it will take no action. The false positive ratio is then computed from dividing the total number of false alarms by the total number of decisions made by the gateway node.

As shown in Fig. 30, the aggregation algorithm achieves much lower false positive ratios compared to that of the local IDS. The local detection module could only use the information of local communication activities to detect possible intrusions. Due to mobility, it is very often that there could be unexpected changes of the routing activities, which will lead to the generation of false alerts by local IDSs. The gateway nodes, on the other hand, by aggregating the local alerts and routing control packets in the zone, could know what is happening in a wider area of the network. By

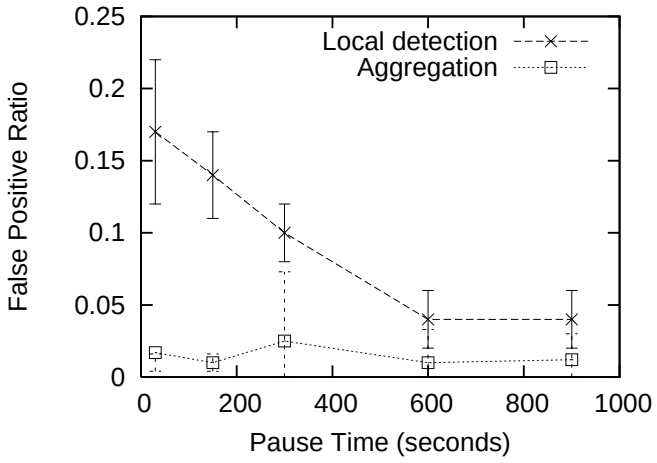


Fig. 30. False Positive Ratio of Local IDS and ZBIDS.

analyzing the probability of aggregated routing control packet source addresses, it could effectively eliminate the sudden unexpected yet normal activities of MANETs. In this way, many false alerts could be suppressed.

2. Detection Ratio

We compute the detection ratio of the aggregation algorithm based on the same attack data used by the local detection model. We measure the detection ratio from dividing the number of gateway nodes that actually generate alarms by the number of gateway nodes that should generate alarms. The result is illustrated in Fig. 31.

As we can see, the aggregation algorithm achieves better detection ratios compared to that of local IDSs. This is because the existence of "partial" victims is the main reason leading to the degradation of the detection ratio of local IDS. However, it is possible that in the same time period, the "partial victims" and some "full victims" of the same attacker coexist in the same zone, the information provided by the "full victims" could make the gateway nodes detect the attackers. In this way, the

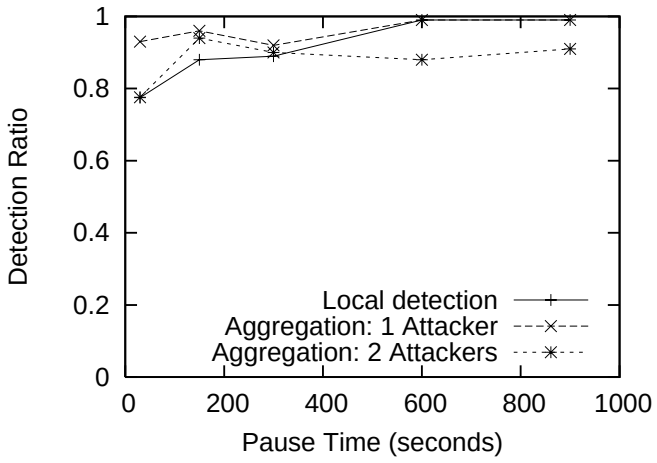


Fig. 31. Detection Ratio of Local IDS and ZBIDS.

detection ratio is improved.

The arbitrary mobility could cause the routing control packets generated in an unexpected way. The routing control packets generated by the attacker may dominate in some time period. In the mean time, it is still possible that the routing control packets caused by the attacker only have a lower portion, for example, other routing control packets caused by the normal routing discovery procedure could cause a burst. This could depress the generation of true alarms. We can see that this phenomenon is more obvious with the increase of the mobility.

The existence of many attackers may lead to the decrease of the detection ratio because this may lead to the decrease of the probability of the attacker source address. This situation is worse when several attackers attack the same victims at the same time. However, if the attackers do not collude, it is likely that different attackers have different victims and their attack time does not overlap. In this case, our ZBIDS can still achieve high detection ratio.

3. Communication Overhead

The extra communication overhead introduced by ZBIDS is caused by propagating the local alerts of intrazone nodes. We measure the communication overhead as the number of transmission of local alerts in a given time period for one node. For a given local alert, it will be transmitted once by all nodes in the same zone.

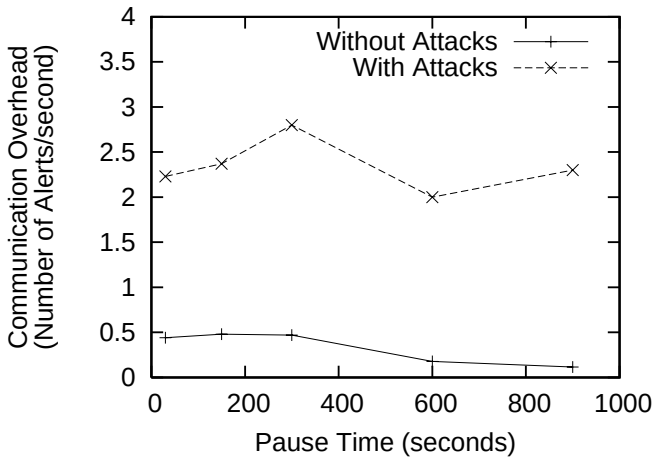


Fig. 32. Communication Overhead of ZBIDS.

As shown in Fig. 32, we can see that, when there are attacks in the network, the communication overhead is higher because of the increased number of generated local alerts. We can also see that, when there are no attacks in the network, the communication overhead decreases with the decrease of mobility. This is because when the mobility is low, local IDSs demonstrate better performance in terms of false positive ratios, thus reduces the number of alerts locally propagated in the zone.

Although extra communication overhead is unavoidable, Fig. 32 shows that the overhead is trivial. In average, each node only needs to send about 2 to 3 alerts per second when there are attacks in the network.

4. Global View of Attacks

Due to the adoption of ZBIDS, the gateway nodes could provide a wider view of the attack that is happening in the network. Expressed in our modified MANET class hierarchy, an example of one possible aggregated alert is depicted in Fig. 33.

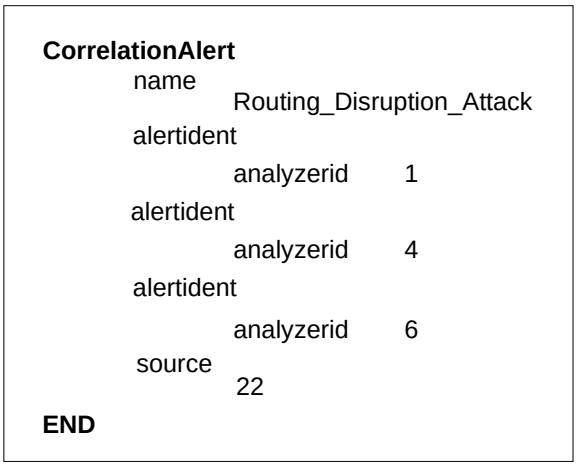


Fig. 33. An Example of One Aggregated Alert.

This example shows that the local IDSs attached to node 1, 4 and 6 generate local alerts and these alerts are aggregated into an *CorrelationAlert* by the gateway nodes. We can conclude from the *CorrelationAlert* that these nodes are the victims of the routing disruption attack. *Source* indicates the identification of the attacker: node 22. This makes it easier to track the offending mobile node.

5. Discussion

Simulation results of the aggregation algorithm demonstrate that ZBIDS could achieve better performance compared to that of local IDSs. Specifically, it could reduce false positive ratios at high mobility, which is desirable in practical environment. We can

also conclude from the simulation that our aggregation algorithm can further improve the detection ratio and provide a global view of the attacks.

D. Summary

Based on the simulation results, we can say that the normal routing behavior can be established and used to construct the detectors.

First, the Markov Chain based local anomaly detection model works well in low mobility environment. In fact, when the mobility is low, we observe less dynamic routing behavior in terms of the utilized features. This makes the simple Markov Chain model accurate to characterize its behavior and demonstrate effective performance.

Second, we observe from our simulation that different classifiers should be constructed under different mobility level. This implies that training data of all mobility levels should be collected in order to achieve desirable performance.

Third, using the aggregation algorithm under the zone based framework, we could reduce the false positive ratio to an acceptable level, especially at high mobility levels. Therefore, the Markov Chain based local anomaly detection model and the aggregation algorithm under the zone based framework complement each other to make a complete MANET IDS.

CHAPTER V

TOWARDS ADAPTIVE INTRUSION DETECTION IN MOBILE AD HOC NETWORKS

One of the main difficulties in building anomaly-based MANET IDSs is how to consider mobility impacts when we design detection engines. This is especially important because most dynamics in MANETs are caused by mobility. MANET IDSs without properly considering mobility are prone to cause high false positive ratio, rendering the IDSs useless. Most previous work on MANET IDSs adopts mobile speed or node pause time to capture the influence of mobility on detection algorithms. We have observed that mobile speed alone is not an accurate measurement. The extraction of a common feature among different mobility models is necessary for tuning system parameters in detection engines.

In this chapter, utilizing different mobility models, we first demonstrate that moving speed, a common parameter in measuring the performance of MANETs, is not desirable in measuring the performance of local MANET IDSs when we consider different applications. We then propose an effective feature for IDSs, link change rate, to dynamically reflect different mobility environment. Suitable normal profiles and proper threshold can then be adaptively selected by each local IDSs through periodically measuring its local link change rate. Utilizing the Markov Chain anomaly detection model as an exemplary MANET IDS described in Chapter III, we demonstrate the effectiveness of our proposed adaptive mechanisms under different mobility models.

The main contribution described in this chapter is to propose a unified measurement to capture the impact of mobility on intrusion detection engines and an effective adaptive mechanism to integrate the above measurement into local MANET

IDSs for the purpose of abstracting normal/abnormal profiles. Most previous work on building IDS for MANETs adopts mobile speed to evaluate the performance of IDS and to tune parameters for feature selection. However, we observe that without taking into account particular mobility models, mobile speed alone cannot tell IDS how fast the link changes are and the parameters setting based on mobile speed will not be accurate. At the end of this Chapter, detailed simulation study is provided.

The rest of the chapter is organized as follows. In Section A, we study the behavior of the local IDS under different mobility models and demonstrate that mobile speed, which is a commonly used metric for most existing IDSs, is not a good measurement for deciding system parameters of IDSs. Based on this observation, we propose an accurate and unified metric. Section B presents adaptive mechanisms which can be integrated into MANET local IDS agent. Section C provides the simulation model and the detailed simulation results.

A. IDS Behavior under Different Mobility Models

1. Different Mobility Models

Two mobility models, the Random Waypoint model and the Random Drunken model, were simulated. In the Random Waypoint mobility model, each node randomly selects a destination in the simulated area and a speed from a uniform distribution of specified speeds. The node then travels to its selected destination at the selected speed. The transit from one position to another position is called a movement epoch. On arriving at the destination, it is stationary for a given pause time. After that, a new movement epoch begins: the node resumes its movement to a newly selected destination with a newly selected speed.

In the Random Drunken mobility model, each node moves independently with

the same average speed. Each node moves continuously within the region without pausing at any location. It changes direction, randomly chosen, after every unit of distance.

2. Simulation Platform and Parameter Settings

We use a simulation model based on GloMoSim [5]. The simulations parameters are the same as described in Table III. The same simulation platform is used throughout the simulation process for the purpose of comparison. In the Random Waypoint model, the *pause time* was set to 0 seconds. In each movement epoch, the speed was uniformly chosen between the *minimum speed* and the *maximum speed*. The {*minimum speed*, *maximum speed*} pair is set to different values in order to measure the impact of speed on IDS performance. In the Random Drunken model, the movement granularity was set to 1 meter, that is, each node randomly re-selects a direction every meter. The node's speed is controlled by the *mobility interval time*, which indicates how long it takes for a node to travel 1 meter. For example, a *mobility interval time* of $0.1S$ is equivalent to $10m/s$.

3. Performance Metrics

We use the following metrics throughout the simulation in order to investigate the impact of different mobility models on the performance of local IDSs.

- **False positive ratio**: It is defined as the percentage of decisions in which normal data are flagged as anomalous.
- **Detection ratio**: It is reported for traces of intrusive behavior and is computed from dividing the total number of correct detections by the total number of victims in the anomalous data.

- **MTFA**: It is defined over anomalous traces and measures how fast the classifier detects the attack. Given an anomalous trace ξ, if we suppose the attack start location is L_a and our IDS generates its first alarm after scanning the L_d-th symbol, then the $MTFA$ corresponding to ξ normalized by the length (denoted as L) of the locality frame is given by $MTFA(\xi) = (L_d - L_a)/L$.

They are the same as the performance metrics of local IDS described in chapter IV.

4. Speed Is Not a Good Metric

In order to investigate the impact of different mobility models on the performance of local MANET IDSs, we use the same parameters (the same number of discretized output of VQ algorithm, "rare symbol" conversion threshold, window size, length of short-term subject activity, penalized value, etc.) to tune *alert threshold* of IDSs under different mobility models. Given a mobility model, the same amount of training data, test data, and abnormal data at different mobility levels are collected using the same procedure in order to build the classifier. A different set of data is collected to evaluate the performance of the classifier. Detailed procedure is described in previous chapters.

a. False Positive Ratio

When using moving speed as the parameter, false positive ratio of the Random Waypoint model and Random Drunken model is shown in Fig. 34. We use relatively larger speed (small mobility interval time) in the Random Drunken model because we observe that when speed is small in the Random Drunken model, the link changes are very small and the routing tables are quite stable [83].

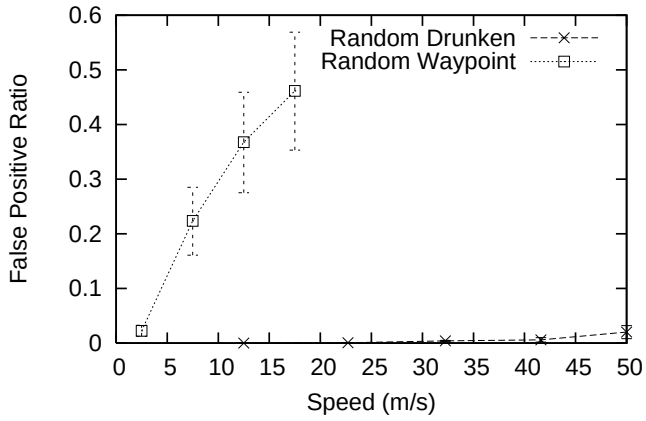


Fig. 34. False Positive Ratio When Using Moving Speed as the Parameter.

In both Fig. 34, we can see that with the increase of speed, the false positive ratio increases. This is more obvious for Random Waypoint model. With the increase of moving speed, no matter what mobility models we use, the node routing tables will have more changes. Therefore, the trace will demonstrate lower regularity, which results in the higher false positive ratio.

We can see that although the moving speed of the Random Drunken model is larger compared to that of the Random Waypoint model, its false positive ratio is much smaller. This is because given the same moving speed, Random Drunken model will not generate as many link breakages as Random Waypoint model does. However, routing table changes are impacted directly by link changes, not node moving speed. If a group of nodes move in the same direction, it is possible that although they move at a very high speed, their routing tables experience small changes. This demonstrates that speed is not a good metric in measuring false positive ratio when we consider different mobility models. Setting IDS parameters based solely on moving speed is likely to be incorrect.

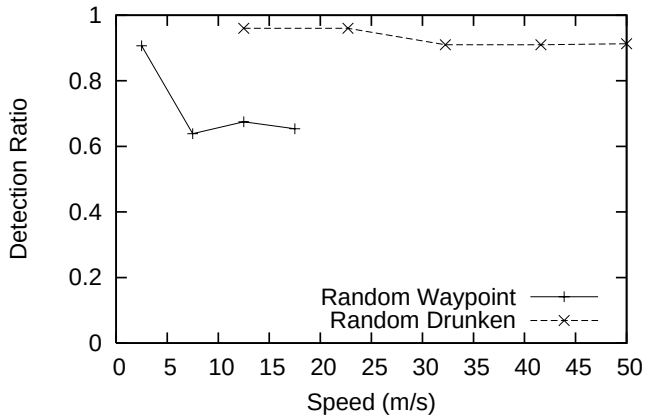


Fig. 35. Detection ratio When Using Moving Speed as a Parameter.

b. Detection Ratio

When using moving speed as the parameter, detection ratio of the Random Waypoint model and Random Drunken model is shown in Fig. 35. From Fig. 35, we can see that in both mobility models, detection ratio decreases with the increases of speed. When mobility is low, routing table changes are less dramatic and has less unexpected changes. Therefore, abnormal behavior tends to have a larger distance from normal profiles, and it is easier for the classifier to identify the abnormal behavior. Also, the phenomenon of "partial victims" is more obvious at high mobility, resulting in the decrease of the detection ratio.

We observe that the overall detection ratio of the Random Drunken model is higher than that of the Random Waypoint model, even if the nodes' moving speed is higher in the Random Drunken model. The reason is similar: network topology is much more stable in the Random Drunken model than in the Random Waypoint model at the same moving speed. This demonstrates that speed is not an accurate metric in measuring detection ratio.

c. MTFA

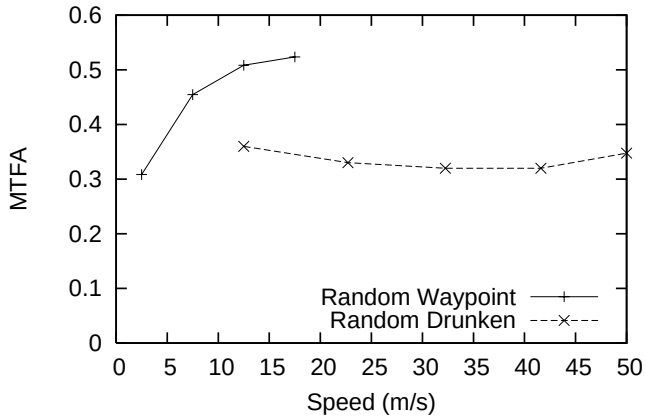


Fig. 36. MTFA When Using Moving Speed as a Parameter.

The results of $MTFA$ of the Random Waypoint model and the Random Drunken model are shown in Fig. 36. For Random Waypoint model, its $MTFA$ increases with the increase of nodes' moving speed. This is because larger moving speed could lead to a larger *alert threshold*, therefore leading to larger $MTFA$. We can also see that although the nodes' moving speed is larger in Random Drunken model, its $MTFA$ is smaller than that of the Random Waypoint model. This again demonstrates that speed is not a good metric in measuring the performance of IDS.

5. A Unified Metric

We have illustrated that node moving speed is not a good parameter in measuring the performance of MANET IDSs. Our purpose is to find a unified metric which is independent of mobility models and could be used to measure MANET IDS performance. Because routing table changes are directly impacted by link changes, we further measured the link change rate of different mobility models under the same scenarios.

We now define link change rate. Suppose for a given node, at time $t1$, its neighbor set is $N1$; at time $t2$, its neighbor set is $N2$. Link change rate is defined as:

$$(|N2 - N1| + |N1 - N2|)/|t2 - t1|$$

$|N2-N1|$ means the number of new neighbors during the time interval of $(t2-t1)$, and $|N1 - N2|$ means the number of neighbors that moved away during the interval of $(t2 - t1)$. They together represent the number of neighbor changes in $(t2 - t1)$. Link change rate can be *locally* collected by each node.

For a given mobility model and a given mobility level represented by {*minimum speed*, *maximum speed*} pair, we compute the average node link change rate. Using the computed link change rate, we combine the MANET IDS performance over different mobility models. The result is illustrated in Fig. 37, Fig. 38, and Fig. 39.

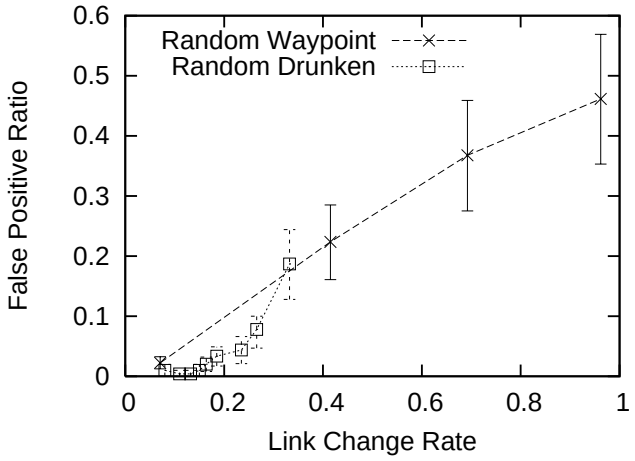


Fig. 37. False Positive Ratio When Using Link Change Rate as a Parameter.

As shown in Fig. 37, with the increase of link change rate, the false positive ratio and MTFA increase, and the detection ratio decreases. Fig. 37 demonstrates that if parameter settings of IDS are based on the link change rate, the performance of IDS will be less independent of mobility model. Compared with nodes moving speed,

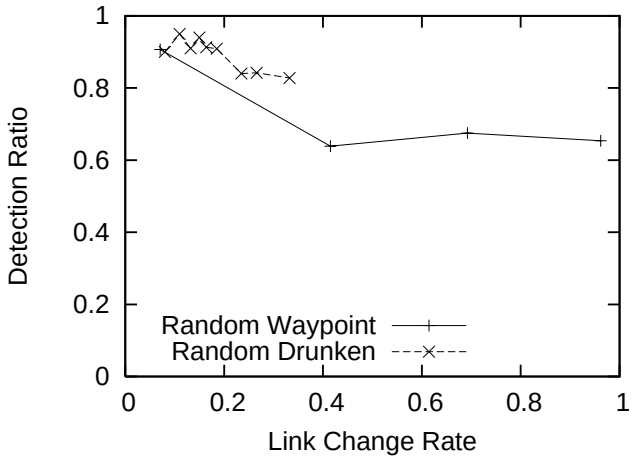


Fig. 38. Detection Ratio When Using Link Change Rate as a Parameter.

link change rates can be used more accurately to measure routing table changes. A larger link change rate implies a more dynamic environment, which makes it more difficult to differentiate normal and abnormal behavior. From Fig. 38, we can see that for the same link change rate, the differences of detection ratio among different models do not have big gap. From Fig. 39, we can that the MTFA increases with the increase of link change rate. All these suggest that the performance of IDS will be less independent of mobility model.

B. Adaptive IDS

1. Adaptive Mechanism

The fact that link change rate can be used to reflect MANET dynamics independent of mobility models motivates us to investigate adaptive mechanisms that utilize link change rate as a security feature and integrate it into our IDS model. For an effective anomaly-based intrusion detection system, an important requirement is that the constructed profiles should be *adaptive*. *Adaptive* profiles can account for normal network

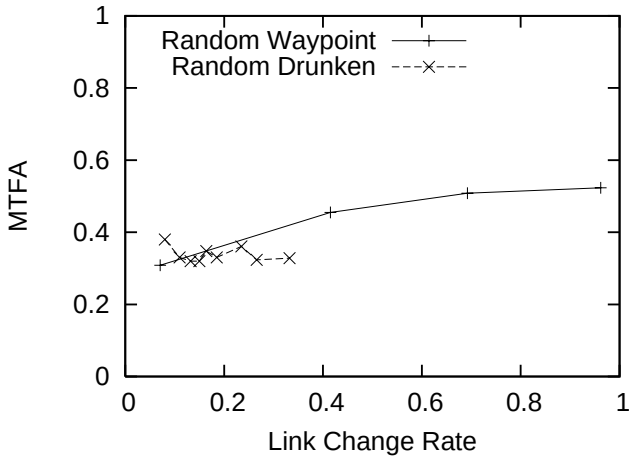


Fig. 39. MTFA When Using Link Change Rate as a Parameter.

changes to avoid raising false alarms. This is especially important in MANETs given its dynamic environments, where different mobility levels will need different normal profiles.

We introduce *adaptive* mechanisms into our systems by adjusting the transition matrix characterized by Markov Chain and the detection threshold through learning its environments locally. Each node measures its *link change rate* periodically, based on the measured link change rate in the recent history, each local IDS can adjust the parameter settings of Markov Chain and the detection threshold. We have demonstrated that different mobility scenarios will need different profiles and different thresholds. Link change rate could provide a unified metric independent of different mobility models and can be used to adjust the behavior of intrusion detection systems.

We take the following procedures to construct our adaptive MANET IDS, as illustrated in Fig. 40.

- **Offline training**: Using different mobility models, we first collect the routing activities at different mobility levels. Following our existing offline training approach to construct the Markov Chain based anomaly detection model, we

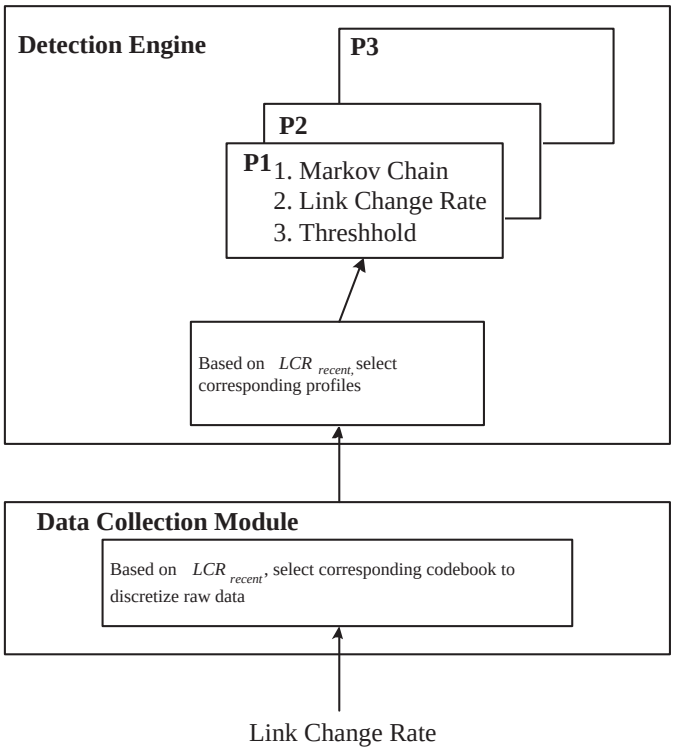


Fig. 40. Adaptive Mechanism.

compute the detection threshold at different mobility levels. We further compute the average link change rate at each mobility level.

- **Online selection**: The data collection module of each IDS agent periodically collects its local link information and computes its link change rate over the recent history, denoted as LCR_{recent}. Based on LCR_{recent}, the data preprocess module discretizes the raw data and selects the corresponding codebook whose link change rate has the smallest Euclidean distance to LCR_{recent}. LCR_{recent} is also reported to detection engine, which can select the normal profile whose link change rate has the smallest Euclidean distance to LCR_{recent}. This process is summarized in Fig. 41.

Procedure Select_Adaptively()
Input: periodically calculated link change rate
Output: normal profile adaptive to mobility
Begin
For each local IDS at time t
Compute the link change rate over the recent history, denoted as LCR_{recent};

Compute the Euclidean distance between LCR_{recent} and each link change rate stored in normal profiles;

Select the normal profile whose link change rate has the smallest Euclidean distance to LCR_{recent};

Use the adaptively selected Markov Chain to calculate the *alert signal* of recent routing activities.

Based on calculated *alert signal* and adaptively selected *alert threshold*, decide whether to generate alert or not.
END For
END

Fig. 41. Pseudocode to *Adaptively* Select Normal Profiles.

2. Measurement of Link Change Rate

The effective implementation of our proposed adaptive IDS depends greatly on the accurate measurement of link change rate. Link change information may be obtained directly from some routing protocols, such as Ad hoc On Demand Distance Vector (AODV) routing, that use beacon messages to detect neighbors as well as link breakage. In this case, our proposed IDS can re-use this information to save bandwidth.

In case there is no existing mechanism to provide link change information, we utilize periodical beacon signal sent by each node to measure link changes. The bandwidth cost of beacon messages is small. Nevertheless, introducing additional bandwidth cost may not be desirable in building an effective IDS. Hence we need to minimize this communication cost without degrading the measurement accuracy

largely. The accuracy of the measurement relies on how fast each node sends out beacon messages. The shorter the interval time of beacon messages, the more accurate the measurement. But sending out beacon messages too often may consume too much energy even if beacon messages are very short.

Although mobile speed is not an accurate metric for tuning IDS parameters as stated before, mobile speed provides us with good heuristics on estimating link change rate: high speed causes more link changes in general. Therefore, a fast node should send out beacon messages more frequently than a slow node. By differentiating beacon messages interval time based on nodes' moving speed, communication cost can be reduced.

Nevertheless, we should make the above measurement strategy independent of mobility models. In order to achieve this goal, we propose a learning approach in adopting beacon message interval time. Initially, a short beacon interval time is used for each node in order to obtain accurate link change information. If a node observes that there is no link change during several beacon interval times, it can increase the beacon interval time to a larger value. When a node increases its moving speed or finds too many link changes within one beacon interval time, it should decrease its interval time. Since our IDS does not require link change information in a very small time granularity (for example, providing link change rate every 50 ms is meaningless), the above learning strategy can reduce communication cost effectively without degrading the detection performance of our IDS.

Based on the above consideration, our local detection agent is energy efficient because it requires very small communication cost. Although our IDS needs to calculate feature values, the calculation is neither complex nor intensive. Also, compared to communication, calculation usually consumes much less energy. For example, the ratio of energy spent in sending one bit versus executing one instruction ranges from

220 to 2290 in different architectures [84].

C. Simulation Study of Adaptive IDSs

We use the same simulation model, as described in Section 2, to perform simulation study for the purpose of comparison. The result is illustrated in Fig. 42, Fig. 43, and Fig. 44.

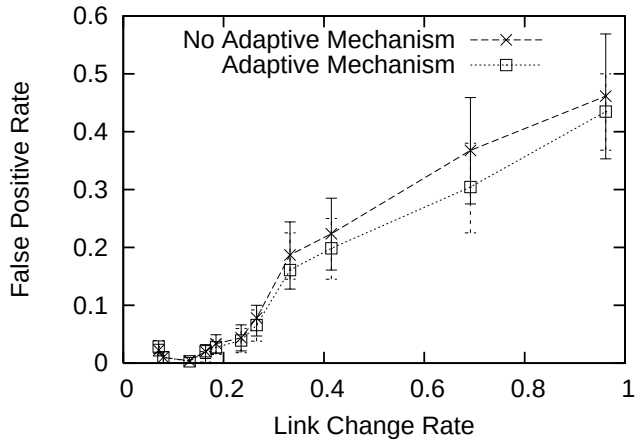


Fig. 42. False Positive Ratio of Adaptive Local IDS.

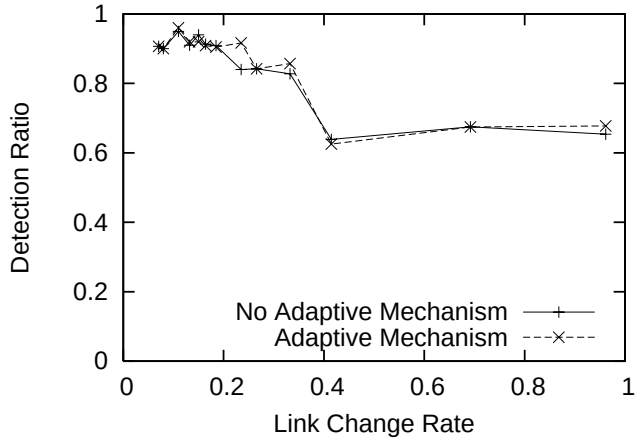


Fig. 43. Detection Ratio of Adaptive Local IDS.

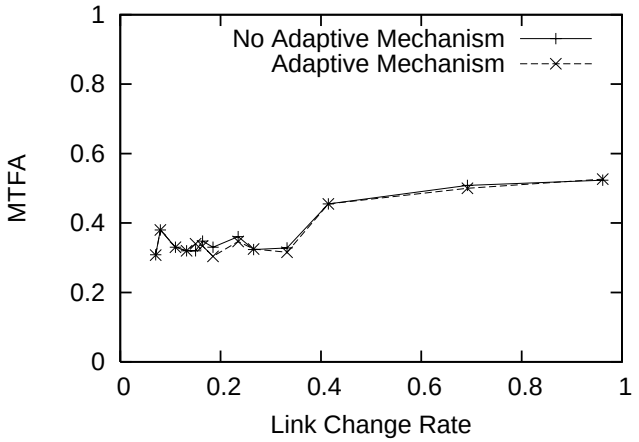


Fig. 44. MTFA of Adaptive Local IDS.

From Fig. 42, we can see that at the same link change rate, the false positive ratio of adaptive IDS is comparable to that of IDS not using adaptive mechanism. Adaptive mechanisms take into consideration mobility-caused dynamics and can change normal profiles correspondingly. We can also see that detection ratio of adaptive mechanisms and non-adaptive mechanisms does not show much difference, as illustrated in Fig. 43. When attacks happen in the network, abnormal routing table changes will not expect to follow any normal profiles. That is, the introduce of the *adaptive* mechanism will not enable the abnormal change caused by the attack to be found in any normal profiles. This will lead to the increase of the same *penalized value*. Therefore, adaptive mechanisms will not help in improving detection ratio. Because of the similar reason, we also observe that $MTFA$ of adaptive mechanisms and non-adaptive mechanisms does not show much difference, as illustrated in Fig. 44. This illustrates that the main benefit of the adaptive mechanism is to provide IDSs which are less dependent on mobility models and keep roughly the same performance compared to non-adaptive mechanisms in terms of false positive ratio, detection ratio and $MTFA$.

Because the Random Waypoint model tends to generate a link change rate which

is larger than that of the Random Drunken model, we further generate test data with relatively smaller link change rate of the Random Waypoint model and apply the corresponding normal profile of the Random Drunken model to it. Simulation results demonstrate similar performance in terms of the false positive ratio, detection ratio and $MTFA$. This again shows that the adaptive mechanism provides IDSs which are less dependent on mobility models.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

In this chapter, we summarize the research, review the contributions, and discuss important future work.

A. Conclusions

This research describes a nonoverlapping Zone-Based Intrusion Detection System (ZBIDS) for mobile ad-hoc networks. It consists of the detailed description of the local IDS agent and the nonoverlapping zone-based framework for the IDS agents to collaborate. They complement each other to make a complete intrusion detection system for MANETs. Because of the importance of routing protocols in MANETs, we use the routing disruption attack as the threat model to illustrate the effectiveness of ZBIDS. Simulation results illustrate that ZBIDS can achieve acceptable false positive ratio and detection ratio.

The first question in intrusion detection research is what statistical features of interest should be used to construct the model. In the history, short sequences of system calls used by priviledged processes can be used to effectively distinguish normal and intrusive behavior and utilized to construct host based IDSs. *Tcpdump* data could be used to construct network based IDS. However, these features are not suitable to be used to construct detection models to guard against MANET routing attacks. Based on the experiment results, we utilize the percentage of the change in route entries and the percentage of the change in number of hops as the features to capture the representative behaviors of MANET network activities.

By collecting the statistical features of interest from the routing cache of mobile nodes periodically and utilizing Vector Quantization algorithm (VQ) to convert them

into discretized data, we utilize a Markov Chain model to capture their temporal dependency and construct the normal profile. A classifier is then constructed based on the probability transition matrix. Considering the dynamic activities of MANETs, we use the average deviation over the past locality frame as the distance measure. In the process of constructing the classifier, conditional entropy is used to determine the proper window size, and other parameters are also properly tuned to consider the performance trade-off.

We have conducted extensive simulations to evaluate the performance of ZBIDS. Simulation results demonstrated that the proposed local Markov Chain based anomaly detection algorithm could have relatively high false positive ratio when mobility is high. Deploying local IDSs alone could also lead to alert flooding problem. Therefore, we further propose the nonoverlapping zone-based framework and an aggregation algorithm. By collecting the security related information expressed in the proposed MANET IDMEF data model from a wider area, the gateway nodes could reduce the false positive ratio and improve the detection ratio by utilizing the aggregation algorithm. A global view of the attack happening in the network could also be provided, which will better facilitate the diagnosis process.

MANET IDSs need to take into consideration mobility impacts in order to achieve desirable performance. Utilizing our Markov Chain based local MANET IDS, we first demonstrate that node moving speed, a commonly used parameter in measuring MANET performance, is not desirable to benchmark the performance of MANET IDSs when we consider different mobility models. We then propose the usage of a new feature, the link change rate, to act as a unified metric in measuring MANET IDS performance. We further demonstrate how to utilize link change rate to build *adaptive* mechanisms into the detection model. Suitable normal profile and proper detection threshold can be dynamically selected by each local IDSs through periodi-

cally measuring its local link change rate. Simulation results show that our proposed adaptive mechanisms are effective in lowering false positive ratio and independent of mobility models.

B. Thesis Contributions

We recap the thesis contributions:

- We utilize the percentage of the change in route entries and the percentage of the change in number of hops as the features to capture the representative behaviors of the MANET network activities and construct a Markov Chain based anomaly detection algorithm.
- We propose a nonoverlapping Zone Based Intrusion Detection System (ZBIDS) that fits the unique requirement of MANETs. In this framework, each node is attached an IDS agent that acts as a local detection module. ZBIDS creates alert management points by introducing a two-tier logical hierarchy. This framework could avoid heavy communication overhead and single point of failure. We propose an aggregation algorithm which could further improve the detection performance. All these local IDSs collectively form a complete MANET IDS to protect the mobile ad hoc network.
- We propose to integrate *adaptive* mechanisms into local MANET IDSs. By utilizing link change rate, a unified metric which could reflect the dynamics of MANETs, the *adaptive* mechanism could dynamically select proper normal profile and detection threshold.

C. Future Work

Up to now, not many research efforts have been devoted to MANET IDSs. This thesis provides our initial work in this respect. As a very new and promising research area, there are several interesting and important future directions:

- We use DSR as the example routing protocol throughout this research to study the performance of ZBIDS. One of the important future work is to study MANET IDS performance under other popular routing protocols (both reactive and proactive).

- Focusing on MANET routing protocols and using the routing disruption attack as the threat model, we develop our ZBIDS to the full. However, because of the difficulty to design a once-for-all security solution, we have thus far done very little study on the protocol analysis at other MANET layers (Medium Access Control layer, Application layer, etc.) and other attack models. We believe it is necessary to carry out research in these aspects in order to guard against intrusions in a high-secure environment. For example, because of the richer semantic information available in the application layer, a Denial-of-Service attack may be detected earlier by the application-layer IDS. Therefore, intrusion detection module needs to be placed at each layer of the node and coordinated to detect the attack more accurately.

- In order to provide better detection performance, it is necessary to analyze and categorize MANET attack models and system vulnerabilities. Existing research work lacks the detailed analysis in this respect. However, sufficient research into the attack scenarios is necessary in several respects.

  First, this could help to identify more useful features that could be used to char-

acterize the normal behavior of MANETs from various aspects. Defining good features is one of the most important steps in building an effective detection model. Possible other features may be constructed from traffic patterns and topological changes. Their characteristics could be utilized to construct better features.

Second, further defined features could also facilitate the construction of misuse based intrusion detection systems. Misuse based IDSs operate based on a database of known attack signatures and system vulnerabilities. Their low false positive ratios are very attractive in practice. However, misuse based IDSs are impossible without the understanding of comprehensive attack scenarios. An attack language needs to be defined to represent attack scenarios. Based on these, suitable rule-based IDSs for MANETs could be constructed. It needs to integrate with anomaly-based IDS to provide better performance.

Third, the cooperation of local IDS agents are required in MANET IDSs due to their distributed nature. Aggregation and correlation of the security related information are needed in order to improve the detection performance. How to cooperate and how to design the aggregation and correlation algorithm also depend on the extensive knowledge of attack scenarios. This is a really challenging issue when the attack model becomes very complex.

- Appropriate intrusion response techniques are needed in order to protect the network system. How to cooperate the intrusion detection and response modules and how to respond to the identified attacks effectively deserve further research.

REFERENCES

[1] Y. Zhang and W. Lee, "Intrusion Detection in Wireless Ad Hoc Networks," *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (ACM MobiCom'00),* Boston, MA, pp. 275-283, Aug. 2000.

[2] M. Satyanarayanan, J. J. Kistler, L. B. Mummert, M. R. Ebling, P. Kumar, and Q. Lu, "Experiences with Disconnected Operation in a Mobile Environment," *Proceedings of USENIX Symposium on Mobile and Location Independent Computing,* Cambridge, MA, pp. 11-28, Aug. 1993.

[3] D. Curry and H. Debar, "Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition," *Internet Draft,* June 2002, http://www.potaroo.net/ietf/ids/draft-ietf-idwg-idmef-xml-11.txt.

[4] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B. Park, and H. Song, "Parsec: A Parallel Simulation Environment for Complex Systems," *IEEE Computer,* vol. 31, no. 10, pp. 77-85, Oct. 1998.

[5] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks," *Proceedings of the 12th Workshop on Parallel and Distributed Simulations (PADS '98),* Banff, Canada, pp. 154-161, May 26-29, 1998.

[6] J. Broch, D. Johnson, and D. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad hoc Networks," *IETF Internet Draft,* Feb. 2002, http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt.

[7] B. Mukherjee, T.L. Heberlein, and K.N. Levitt, "Network Intrusion Detection," *IEEE Network,* vol. 8, no. 3, pp. 26-41, 1994.

[8] H. Debar, M. Dacier, and A. Wespi, "A Revised Taxonomy for Intrusion-Detection Systems," *Annales des Telecommunications,* vol. 55, pp. 361-378, 2000.

[9] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance," Technical Report, James P. Anderson Co., Fort Washington, PA, April, 1980.

[10] D. E. Denning, "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering,* vol. 13, no. 7, pp. 222-232, Feb. 1987.

[11] P. Porras and A. Valdes, "Live Traffic Analysis of TCP/IP Gateways," *Proceedings of the 1998 ISOC Symposium on Network and Distributed System Security (NDSS'98),* San Diego, CA, March 1998.

[12] T. F. Lunt, R. Jagannathan, R. Lee, S. Listgarten, D. L. Edwards, P. G. Neumann, H. S. Javitz, and A. Valdes, "IDES: The Enhanced Prototype C a Real-time Intrusion-Detection Expert System," Technical Report SRI-CSL-88-12, SRI International, Menlo Park, CA, Oct. 1988.

[13] H.S. Javitz and A. Valdes, "The SRI Statistical Anomaly Detector," *Proceedings of 1991 IEEE Symposium on Research in Security and Privacy,* pp. 316-326, May 1991.

[14] R. Jagannathan, T. Lunt, D. Anderson, C. Dodd, F. Gilham, C. Jalali, H. Javitz, P. Neumann, A. Tamaru, and A. Valdes, "System Design Document: Next-Generation Intrusion Detection Expert System (NIDES)," Technical Report A007/A008/A009/A011/A012/ A014, SRI International, 333, Ravenswood Avenue, Menlo Park, CA, March 1993.

[15] U. Lindqvist, and P.A. Porras, "Detecting Computer and Network Misuse through the Production-Based Expert System Toolset (P-BEST)," *Proceedings of the 1999 IEEE Symposium on Security and Privacy,* Oakland, CA, pp. 146-161, May 9-12, 1999.

[16] Internet Security Systems, "RealSecure Network Protection," Nov. 2003, Available at
http : *//www.iss.net/products_services/enterprise_protection/rsnetwork.*

[17] S. Kumar and E. Spafford, "A Pattern Matching Model for Misuse Intrusion Detection," *Proceedings of the 17th National Computer Security Conference,* pp. 11-21, Oct. 1994.

[18] P.A. Porras and R. Kemmerer, "Penetration State Transition Analysis C a Rule-Based Intrusion Detection Approach," *Proceedings of the 8th Annual Computer Security Application Conference,* pp. 220-229, Nov. 1992.

[19] K. Ilgun, "Ustat: A Real-time Intrusion Detection System for Unix," *Proceedings of IEEE Symposium on Research in Security and Privacy,* Oakland, CA, pp. 16-28, May, 1993.

[20] H. Debar, M. Becker and D. Siboni, "A Neural Network Component for an Intrusion Detection System," *Proceedings of 1992 IEEE Symposium on Research in Security and Privacy,* Oakland, CA, pp. 240-250, May, 1992.

[21] S. Forrest, S.A. Hofmeyr, and A. Somayaji, "Computer Immunology," *Communications of the ACM,* vol. 40, no. 10, pp. 88-96, Oct. 1997.

[22] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting Intrusions Using System Calls: Alternative Data Models," *Proceedings of 1999 IEEE Symposium on*

Research in Security and Privacy, Oakland, CA, pp. 133-145, May 1999.

[23] W. Lee, S. J. Stolfo, and K. W. Mok, "A Data Mining Framework for Building Intrusion Detection Models," *Proceedings of the 1999 IEEE Symposium on Security and Privacy,* Oakland, CA, pp. 120-132, May 1999.

[24] N. Ye, X. Li, Q. Chen, S. M. Emran, and M. Xu, "Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 31, no. 4, pp. 266-274, 2001.

[25] N. Ye, S. M. Emran, Q. Chen, and S. Vilbert, "Multivariate Statistical Analysis of Audit Trails for Host-based Intrusion Detection," *IEEE Transactions on Computers,* vol. 51. no. 7, pp. 810-820, 2002.

[26] S. Jha, K. Tan, and R.A. Maxion, "Markov Chains, Classifiers, and Intrusion Detection," *Proceedings of the 14th IEEE Computer Security Foundations Workshop,* Cape Breton, Nova Scotia, Canada, pp. 206-219, 2001.

[27] N. Ye, "A Markov Chain Model of Temporal Behavior for Anomaly Detection," *Proceedings of 2000 IEEE Workshop on Information Assurance and Security,* United States Military Academy, West Point, NY, pp. 171-174, June 6-7, 2000.

[28] N. Ye, T. Ehiabor, and Y. Zhang, "First-order versus High-order Stochastic Models for Computer Intrusion Detection," *Quality and Reliability Engineering International,* vol. 18, no. 3, pp. 243-250, 2002.

[29] M. Nassehi, "Anomaly Detection for Markov Models," Technical Report Tech Report RZ 3011(#93057), IBM Research Division, Zurich Research Laboratory, March 1998.

[30] R. M. Gray, "Vector Quantization," *IEEE ASSP Magazine,* pp. 4-29, April 1984.

[31] C. Ko, G. Fink, and K. Levitt, "Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-Based Approach," *Proceedings of 1997 IEEE Symposium on Security and Privacy,* Oakland, CA, pp. 134-144, May 1997.

[32] O. Kachirski, and R. Guha, "Intrusion Detection using Mobile Agents in Wireless Ad Hoc Networks," *Proceedings of IEEE Workshop on Knowledge Media Networking,* pp. 153-158, 2002.

[33] D. Samfat, and R. Molva, "IDAMN: An Intrusion Detection Architecture for Mobile Networks," *IEEE Journal on Selected Areas in Communications,* vol. 15, no. 7 , pp. 1373-1380, Sept. 1997.

[34] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (ACM MobiCom'00)*, Boston, MA, pp. 255-265, Aug. 2000.

[35] Y.A. Huang, "Anomaly Detection for Wireless Ad Hoc Routing Protocols," Master thesis, North Carolina State University, 2001.

[36] Y. Huang, W. Fan, W. Lee, and P. S. Yu, "Cross-Feature Analysis for Detecting Ad-hoc Routing Anomalies," *Proceedings of the 23rd International Conference on Distributed Computing Systems*, Providence, RI, pp. 478-487, May 2003.

[37] H. Debar and A. Wespi, "Aggregation and Correlation of Intrusion-Detection Alerts," *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID'01),* Davis, CA, pp. 85-103, 2001.

[38] F. Cuppens, "Managing Alerts in a Multi-Intrusion Detection Environment,"

17th Annual Computer Security Applications Conference, New Orleans, Louisiana, Dec. 10-14, 2001.

[39] F. Cuppens, and A. Mige, "Alert Correlation in a Cooperative Intrusion Detection Framework," *Proceedings of 2002 IEEE Symposium on Security and Privacy,* Berkeley, CA, pp. 202-215, 2002.

[40] A. Valdes and K. Skinner, "Probabilistic Alert Correlation," *Proceedings of the 4th International Workshop on the Recent Advances in Intrusion Detection (RAID'01),* Davis, CA, Oct. 2001.

[41] P. Ning, Y. Cui, and D. S. Reeves, "Analyzing Intensive Intrusion Alerts via Correlation," *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID'02),* LNCS 2516, Zurich, Switzerland, pp.74-94, Oct. 2002.

[42] B. Morin, L. M, H. Debar, and M. Ducass, "M2D2: A Formal Data Model for IDS Alert Correlation," *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID'02),* LNCS 2516, Zurich, Switzerland, pp. 115-137, Oct. 16-18, 2002.

[43] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *ACM Computer Communication Review,* vol. 24, London, UK, pp. 234-244, Oct., 1994.

[44] Murthy and J. J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," *ACM Mobile Networks and Applications Journal,* vol. 1, no. 2, pp. 183-197, 1996.

[45] C. E. Perkins, and E. M. Royer, "Ad hoc On-Demand Distance Vector (AODV)

Routing," *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99),* New Orleans, LA, pp. 90-100, Feb. 1999.

[46] V. D. Park and M. S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," *Proceedings of the 9th IEEE International Conference on Computer Communications and Networking (INFOCOM'97),* Kobe, Japan, pp. 1405-1413, Apr. 1997.

[47] M. Joa-Ng and I. Lu, "A Peer-to-Peer Zone-Based Two-Level Link State Routing for Mobile Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications,* vol. 17, no. 8, pp. 1415-1425, Aug. 1999.

[48] P. Papadimitratos and Z.J. Haas, "Secure Routing for Mobile Ad Hoc Networks," *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'02),* San Antonio, TX, pp. 27-31, Jan. 2002.

[49] J. Lundberg, "Routing Security in Ad Hoc Networks," *Tik-110.501 Seminar on Network Security 2000,* Helsinki, Finland, 2000.

[50] L. Zhou, and Z. Haas, "Securing Ad Hoc Networks," *IEEE Network Magazine*, vol. 13, no. 6, pp. 24-30, November/December, 1999.

[51] B. Dahill, B. N. Levine, E. Royer, and C. Shields, "A Secure Routing Protocol for Ad Hoc Networks," Technical Report UM-CS-2001-037, Department of Computer Science, University of Massachusetts, Aug. 2001.

[52] Y. Hu, D. B. Johnson, and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks," *4th IEEE Workshop on Mobile*

Computing Systems and Applications (WMCSA'02), Calicoon, NY, pp. 3-13, June 2002.

[53] S. Yi, R. Naldurg, and R. Kravets, "Security Aware Ad Hoc Routing for Wireless Networks," Technical Report, UIUCDCS-R-2001-2241, Department of Computer Science, University of Illinois at Urbana-Champaign, Aug. 2001.

[54] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An On-Demand Secure Routing Protocol Resilent to Byzantine Failures," *Proceedings of the 1st ACM Workshop on Wireless Security (WiSe'02),* Atlanta, GA, pp. 21-30, Sept. 28, 2002.

[55] M. G. Zapata, and N. Asokan, "Securing Ad Hoc Routing Protocols," *Proceedings of the 1st ACM Workshop on Wireless Security (WiSe'02),* Atlanta, GA, pp. 1-10, Sept. 28, 2002.

[56] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks," *Proceedings of the IEEE 9th International Conference on Network Protocols (ICNP'01)*, Riverside, CA, pp. 251-260, Nov. 11-14, 2001.

[57] H. Yang, X. Meng, and S. Lu, "Self-Organized Network Layer Security in Mobile Ad Hoc Networks," *Proceedings of the 1st ACM Workshop on Wireless Security (WiSe'02),* Atlanta, GA, pp. 11-20, Sept. 28, 2002.

[58] Y. Hu, A. Perrig, and D. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," *Proceedings of the 8th ACM International Conference on Mobile Computing and Networking (MobiCom'02),* Atlanta, GA, pp. 12-23, Sept. 23 - 28, 2002.

[59] Y. Hu, A. Perrig, and D.B. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks," *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03),* vol. 3, San Francisco, CA, pp. 1976-1986, April 2003.

[60] Y. Hu, A. Perrig, and D.B. Johnson, "Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols," *Proceedings of the 2003 ACM Workshop on Wireless Security (WiSe'03) in Conjunction with Mobicom'03,* San Diego, CA, pp. 30-40, Sept. 2003.

[61] R. Sivakumar, P. Sinha, and V. Bharghavan, "CEDAR: A Core-Extraction Distributed Ad Hoc Routing Algorithm," *IEEE Journal on Selected Areas in Communications,* Special Issue on Wireless Ad Hoc Networks, vol. 17, no. 8, pp.1454-1465, Aug. 1999.

[62] H. Goto, M. Mambo, H. Shizuya, and Y. Watanabe, "Evaluation of Tamper-Resistant Software Deviating from Structured Programming Rules," *Information Security and Privacy (ACISP'01),* Lecture Notes in Computer Science 2119, Springer-Verlag, 2001.

[63] S. Capkun, J. P. Hubaux and L. Buttyn, "Mobility Helps Security in Ad Hoc Networks," *Proceedings of the 4th ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC'03),* Annapolis, MD, pp. 46-56, June 2003.

[64] L. R. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of IEEE,* vol. 77, no. 2, pp. 257-286, 1989.

[65] T. P. Ryan, *Statistical Methods for Quality Improvement*, New York:John Wiley & Sons, 1989.

[66] A. K. Ghosh, A. Schwartzbart, and M. Schatz, "Learning Program Behavior Profiles for Intrusion Detection," *Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring,* Santa Clara, CA, pp. 51-62, April 1999.

[67] R. O. Duda and P. E. Hart, *Pattern Classification*, New York:John Wiley & Sons, 2001.

[68] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol: 22, no. 1, pp. 4-37, 2000.

[69] T. Cover and J. Thomas, *Elements of Information Theory*, John Wiley & Sons, 1991.

[70] K. Tan, and R. Maxion, ""Why 6? Defining the Operational Limits of Stide, an Anomaly-based Intrusion Detector," *Proceedings 2002 IEEE Symposium on Security and Privacy,* Oakland, CA, pp. 173-186, May 12-15, 2002.

[71] W. Lee, and D. Xiang, "Information-Theoretic Measures for Anomaly Detection," *Proceedings of 2001 IEEE Symposium on Security and Privacy,* Oakland, CA, pp. 130-143, May 13-16, 2001.

[72] C. Zhai and J. Lafferty, "A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval," in *Proceedings of the 24 th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001),* New Orleans, LA, pp. 334-342, 2001.

[73] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications,* pp. 702-710, Jan. 1980.

[74] W. W. Cohen. "Fast Effective Rule Induction," *Proceedings of the 12th International Conference on Machine Learning,* Lake Tahoe, CA:Morgan Kaufmann, pp. 115-123, 1995.

[75] A. B. Clarke, and R. L. Disney, *Probability and Random Processes: A First Course with Applications*, New York:John Wiley & Sons, 1985.

[76] R. Durrell, *Probability: Theory and Examples*, Pacific Grove, CA: Wadsworth & Brooks, Second Edition, 1995.

[77] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, New York:McGraw Hill, 1990.

[78] T. M. Mitchell, *Machine Learning*, New York:McGraw-Hill, 1997.

[79] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," *Proceedings of LISA '99: 13th Systems Administration Conference,* Seattle, Washington, pp. 229-238, Nov. 7-12, 1999.

[80] Computer Associates, "E-Trust Intrusion Detection," 2003, http://www3.ca.com/Solutions/Product.asp?ID=163.

[81] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Model*, Reading,MA:Addison-Wesley, 1998.

[82] B.H. Kantowitz and R.D. Sorkin, *Human Factors: Understanding People-System Relationships*, New York:John Wiley & Sons, 1983.

[83] K. Wu, and J. Harms, "Performance Study of Proactive Flow Handoff for Mobile Ad Hoc Networks," *Proceedings of the 9th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'01),* Cincinnati, OH, pp. 99-107, Aug. 2001.

[84] V. Raghunathan, C. Schurgers, S. Park, and M.B. Srivastava, "Energy-Aware Wireless Microsensor Networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 40-50, March 2002.

APPENDIX A

SIMULATION PARAMETERS USED IN CHAPTER III

Table III. Simulation Parameters

Parameters	Values
Channel capacity	2Mbps
Channel model	Free space propagation model with a threshold cutoff
Transmission range	250m
MAC layer	Distributed Coordination Function of IEEE 802.11
Number of nodes	30
Moving region	1000m X 500m
Mobility model	Random waypoint model
Minimum speed	3m/s
Maximum speed	5m/s
Traffic pairs	8 pairs with CBR traffic
Interval transmission time	0.25s
Data packet size	512bytes
Data collection interval time	3 seconds

APPENDIX B

THE XML DTD DESCRIPTION OF THE ALERT CLASS HIERARCHY OF ZBIDS

The occurrence indicators

? the content may appear either once or not al all

* the content may appear one or more times or not at all

+ the content must appear at least once, and may appear more than once

[none] the content must appear exactly once

(1)**Alert**

<!ELEMENT Alert (

DetectTime?, CreateTime, Analyzer, Source*, Target*, Assessment?, AnalyzerTime?, Classification+, Assessment?

)>

<!ATTLIST Alert

ident CDATA '0'

>

(2)**DetectTime**

<!ELEMENT DetectTime (#PCDATA)

>

<!ATTLIST DetectTime

time CDATA #REQUIRED

>

(3)**CreateTime**

<!ELEMENT CreateTime (#PCDATA)

>

<!ATTLIST CreatetTime

time CDATA #REQUIRED

>

(4)**Analyzer**

<!ELEMENT Analyzer (

Node?

)>

<!ATTLIST Analyzer

analyzerid CDATA '0'

>

(5)**Source**

<!ENTITY %attvals.yesno "

(unknown | yes | no)

">

<!ELEMENT Source (

Node?

)>

<!ATTLIST Source

ident CDATA '0'

spoofed %attvals.yesno 'unknown'

>

(6)**Target**

<!ENTITY %attvals.yesno "

(unknown | yes | no)

">

<!ELEMENT Target (
Node?
)>
<!ATTLIST Target
ident CDATA '0'
decoy %attvals.yesno 'unknown'
>

(7)**AnalyzerTime**

<!ELEMENT AnalyzerTime (#PCDATA)
>
<!ATTLIST AnalyzerTime
time CDATA #REQUIRED
>

(8)**Classification**

<!ELEMENT Classification
(name) >

(9)**Assessment**

<!ENTITY %attvals.confidence "
(low | medium | high | numeric)
">
<!ELEMENT Assessment (#PCDATA | EMPTY)*
>
<!ATTLIST Assessment
confidence %attvals.confidence 'numeric'
>

(10)**Node**

```
<!ENTITY %attvals.nodecategory “
   (Intra-zone | Gateway)
”>
<!ELEMENT        Node                (
   Location?, (Name | Address), Address*, Zone*
)>
<!ATTLIST        Node
   ident    CDATA    ‘0’
   category    % attvals.nodecategory    ‘Intra-zone’
>
```

(11)**Location**

```
<!ELEMENT  Location  (#PCDATA)
>
<!ATTLIST  Location
   x    CDATA    #REQUIRED
   y    CDATA    #REQUIRED
   z    CDATA    #IMPLIED
>
```

(12)**Address**

```
<!ENTITY %attvals.addresscategory “
   (unknown | ipv4-addr | ipv6-addr)
”>
<!ELEMENT  Address  (#PCDATA)
>
<!ATTLIST  Address
   ident    CDATA    ‘0’
```

category %attvals.addresscategory 'unknown'

address CDATA #REQUIRED

>

(13)**Zone**

<!ELEMENT Zone (#PCDATA)

>

<!ATTLIST Zone

zoneid CDATA #REQUIRED

>

(14)**Name**

<!ELEMENT Name (#PCDATA)

>

<!ATTLIST Name

name CDATA '0'

>

(15)**Netmask**

<!ELEMENT Netmask (#PCDATA)

>

<!ATTLIST Netmask

netmask CDATA '0'

>

(14)**CorrelationAlert**

<!ELEMENT CorrelationAlert (

name, Alertident+

)>

(16)**Alertident**

```
<!ELEMENT  Alertident  (#PCDATA)
>
<!ATTLIST  Alertident
   analyzerid   CDATA   #IMPLIED
>
```

VITA

Bo Sun, was born in 1974 in Xuzhou, P.R. China. He received his B.E and M.E. diplomas respectively from Computer Communications Department, Nanjing University of Posts & Telecommunications in 1996 and Beijing University of Posts & Telecommunications in 1999. He is going to receive the Ph.D. degree in Computer Science at Texas A&M University in May 2004.

His permanent mailing address is: Xuzhou Jianguo Xilu #127 Unit 2, Apt. 606, Jiangsu, P.R.China, 221006.

The typist for this thesis was Bo Sun.