A HYBRID FRAMEWORK FOR INTRUSION DETECTION IN WIRELESS MESH NETWORKS

by

Muhammad Usama bin Aftab
B.Eng, Nadirshaw Eduljee Dinshaw University of Engineering and Technology, 2012

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

© Muhammad Usama bin Aftab, 2015
University of Victoria

All rights reserved. This dissertation may not be reproduced in whole or in part, by photocopying or other means, without the permission of the author.

A HYBRID FRAMEWORK FOR INTRUSION DETECTION IN WIRELESS MESH NETWORKS

by

Muhammad Usama bin Aftab
B.Eng, Nadirshaw Eduljee Dinshaw University of Engineering and Technology, 2012

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor
(Department of Electrical and Computer Engineering)

Dr. H. El Miligi, Departmental Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor
(Department of Electrical and Computer Engineering)

Dr. H. El Miligi, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Network security is an important domain in the field of computer engineering. Sensitive information flowing across computer networks is vulnerable to potential threats, therefore it is important to ensure their security. Wireless Mesh Networks (WMNs) are self-organized networks deployed in small proximity which have an wireless ad-hoc mesh topology. While they are cost effective and easy to deploy, they are extremely vulnerable to network intrusions due to no central switch or router. However, they can be secured using cryptographic techniques, firewalls or Demilitarized Zones (DMZs). Intrusion Detection Systems (IDSs) are used as a secondary line-of-defence in computer networks from possible intrusions. This thesis proposes a framework for a Hybrid Intrusion Detection System (HIDS) for WMN.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

In my graduate research, there are many people who directly or indirectly helped me. Therefore, I thank all those people who made an impact on me. I would like to specially thank:

My parents, who taught me how to speak my first words and how to be a better person. This thesis would not have been possible without their love and support and I cannot be thankful enough for all their sacrifices.

Fanie Collardeau, for always being there to support my research. Her advice always helped me to gain motivation and to put my efforts in the best way possible.

Prof. Aaron Gulliver, for guiding me in my thesis. Without his technical advice, i would not have been able to finish this thesis. His advice always helped me to successfully overcome the fear of research and gave me the determination to potentially apply for doctorate programme.

University of Victoria, for an extremely supportive environment which helped me to focus and dedicate my efforts to this research. Also for providing me with a prestigious fellowship which helped my work.

Darren Beckwith and Damien Hortsing, for their encouragement and support to help me focus on my thesis. Carmanah Technologies, to understand the importance of my research by providing me a friendly environment to work in.

Among the hills, when you sit in the cool shade of the white poplars, sharing the peace and serenity of distant fields and meadows - then let your heart say in silence, "God rests in reason". And when the storm comes, and the mighty wind shakes the forest, and thunder and lightning proclaim the majesty of the sky - then let your heart say in awe, "God moves in passion". And since you are a breath in God's sphere, and a leaf in God's forest, you too should rest in reason and move in passion.
- Kahlil Gibran

DEDICATION

I dedicate my work to my parents.

Chapter 1

Introduction

Wireless Mesh Network (WMN) is a growing domain in computer networking due to its easy deployment and cost effectiveness, but the lack of a practical IDS prevents it from being adopted widely. In WMN, security is achieved using cryptographic techniques borrowed from Mobile Ad-hoc Networks (MANETs), Wireless Sensor Networks (WSNs) and Wireless Local Area Networks (WLANs). Firewalls are used as a primary security solution, as they can restrict the access of unauthorized nodes into the network and provide a basic level of security. However, restricting unauthorized access does not guarantee security from possible intrusions and leaves the network vulnerable to attacks. If there is no secondary process to detect the presence of unauthorized nodes, intruders can find backdoors to gain access to the network and perform malicious activities. Thus, WMNs need a sophisticated IDS that can detect any malicious activities or intruders in the network and alert the proper authorities. In terms of working principle, intrusion detection can be categorized under two major methods [1]

1. Anomaly Detection (by detecting anomalies in the network)
2. Misuse or Signature Based Detection (by detecting attack patterns in the network)

Traditionally, networks have at least one centralized hub, switch or router connecting nodes and allowing them to communicate with each other, providing easy traffic monitoring. It also gives information about the network which helps the IDS to detect any possible intrusions in the network.

WMNs combine the advantages of ad-hoc networks and WLANs but are different from both [2]. The IDS model should exploit the features of WMN for implementing

a long-term solution for intrusion detection. Using a traditional IDS model might not be feasible for adequate intrusion detection. Hence, a hybrid technique is developed which exploits the best features of traditional IDS, while understanding the uniqueness of WMN to provide fast and better intrusion detection.

1.1 Motivation

Although previous research on intrusion detection for WMN is limited, this study takes motivation from two significant approaches available to design a new framework. These methods are significant because they show packet analysis as a main mechanism for intrusion detection.

1.1.1 Attack detection approach designed for BATMAN protocol

In [3], the authors provided an approach to detect network attacks for Better Approach To Mobile Adhoc Networking (BATMAN) protocol. This method consists of collecting and sorting traffic (pcap files) which provides routing events in the WMN using the Montimage Monitoring Tool (MMT). The results collected by MMT software were processed by an attack detection algorithm, where the number of inconsistencies were counted based on the disruptions found in the routing behaviour. Although, the authors claimed that their method could work with other routing protocols, no clear instructions were provided. In addition, the method has some problems:

- The method is not a stand-alone IDS; it provides ways to detect intrusions. All the IDS related problems such as processing cost and network overhead are not considered.
- The detection process strictly focuses on BATMAN packets. Hence, it has to be modified for any other routing protocol.
- It does not work in real-time: it first stores the data and then analyzes the packets separately.
- The method only describes the number of inconsistencies found by the algorithm but neglects the effects of mobility.

Attack detection approach presented for BATMAN is one of the few researches available for IDS in WMN. It also shows a difference between a theoretical approach and an implementable framework which is a consistent problem in the available literature.

1.1.2 Active probing based mechanism for IDS

In [4], a practical IDS was presented and tested on a test-bed consisting of 18 hosts. An active probing based IDS which exploited the classic fingerprinting mechanism was introduced and executed on an independent host. This Active Probing based Intrusion Detection System (NP-AIDS) or simply DogoIDS as described in [5] was designed for a separate IDS host which was prone to attacks and had additional processing capabilities as compare to a regular host. This DogoIDS host was completely portable and could move in the network. It keeps a distance of one hop from the target host before starting IDS process. It then probed the target host in the network for responses. Finally it matched the responses with response templates already saved in the database and analyzed the differences.

If the host was working properly, the response had to be identical to the one in the DogoIDS database. Significant differences between responses generated an alert. For each test, there were many probes, and in every probe there were many test packets whose responses were analyzed separately. The mechanism is explained in [4] in greater detail.

The model showed a practical way to detect intrusions that solves many problems other models did not answer. NP-AIDS, by exploiting the classic fingerprinting judged hosts based on the acquired responses. This model gave the foundation for this research as it explained various practical issues. After carefully analyzing the DogoIDS model, some open questions came up as listed below:

- The DogoIDS host works as a separate entity and never coordinates with the network for additional information. Thus, it lacks up-to-date information about the network.

- DogoIDS moves in the physical proximity (on a vehicle) and probes the hosts randomly. An intruder may go unnoticed because of this random behaviour.

- In [4], the increase in background traffic drastically increases the false alarm rate if the number of test packets are limited. In some cases, up to 5 test packets are

sent which increases the traffic if two or more IDS hosts are used for testing.

- DogoIDS works with one host at a time, hence a collective approach to test whole network simultaneously is not considered. It tests each host separately and decides their legitimacy.
- DogoIDS has to be one hop away from the target host when the IDS process starts, and it needs to be physically close to the intruder.
- The probing process occurs repeatedly without any mechanism to intelligently decide a pattern to test hosts. This method increases the traffic in the network without necessarily reducing the risk of intrusions.

DogoIDS is a significantly contribution in intrusion detection in WMN, but left open some issues. This thesis specially focuses on the preparation of a stable framework, which has the potential to turn into a practical model, while answering those issues.

1.2 Intrusion Detection for WMN

In the literature, some attack detection approaches and models are given that provide basic functionalities of intrusion detection for WMN. In [6], a reputation based model to detect intrusions in wireless mesh networks is given which works on reactive routing protocol. An intrusion detection system for OLSR based mesh networks is presented in [7] which shows descriptive experimental results. Some other examples of attack detection techniques are found in [8-11].

1.3 Contributions

In this thesis, a framework is presented for modelling IDS which is specially dedicated to WMN but can be extended to other applications. It exploits mesh topology to acquire information which is useful for the detection process.

The contributions of the proposed framework as an intrusion detector are:

1. A rule is developed to divide a WMN into zones.
2. A reputation algorithm is developed to quantify the trust of hosts.

3. A hybrid detection approach is given by using the best features from anomaly detection and misuse detection.

4. The dynamic behaviour of WMN is considered, which makes it scalable and potentially usable in other technologies like WSN, WLAN, MANET and Vehicular Ad-hoc Networks (VANET).

The results in this thesis will help in designing a practical IDS using the concepts described in the framework. It will also help researchers to decrease the false alarm rate and increase the chances of preventing intrusions by exploiting the uniqueness in mesh networks.

1.4 Thesis Outline

This section outlines the structure of thesis.

Chapter 2 describes the design of WMN and categorization of routing protocols. It also gives the description of common threats found in WMN.

Chapter 3 explains the key concepts used in this thesis such as reputation management and zoning algorithms. Hybrid framework for intrusion detection system is introduced in this chapter with detailed explanation and examples.

Chapter 4 contains the implementation of framework in OMNET++ using Network Description Language (NED) and C++. It also presents the performance evaluation of the proposed design by analyzing it through various tests including mobility, varying size and differing zones.

Chapter 5 discusses the future scope and scalability of the framework. It also presents ideas for potential work in intrusion detection systems in WMN. It contains concluding remarks and findings.

Chapter 2

Threats and Security Concerns in Wireless Mesh Networks

WMNs are self configurable networks joined in a mesh topology. Due to their architecture, they are used to access internet in small and large networks. They are conventionally different from traditional Local Area Networks (LAN) or WLAN. In this thesis, a host is defined as a wireless device which can be a desktop computer, laptop, PDA or smartphone.

2.1 Design features of WMN

In WMN, each host acts as a router to transmit traffic between peers if the destination host is not directly connected to the sender host. This formation of hosts is reliable because each host is connected to other hosts, giving alternate routes to any possible incoming packet. These networks are also self-organized, so if a host leaves the network, network will reconfigure itself for new potential routes for forwarding packets. As mentioned in [12], WMN has similar topology to mobile ad-hoc networks but it shows slow or no mobility. In WMN, the change in the topology occurs when a host leaves the network. WMN also does not show tremendous traffic abnormalities, because the use of the mesh network is normally under a controlled circumstances, revealed in [12].

WMN consists of different nodes with distinct working characteristics. The classification is given below,

1. *Wireless Mesh Clients* are typically the mesh nodes such as laptops, computers,

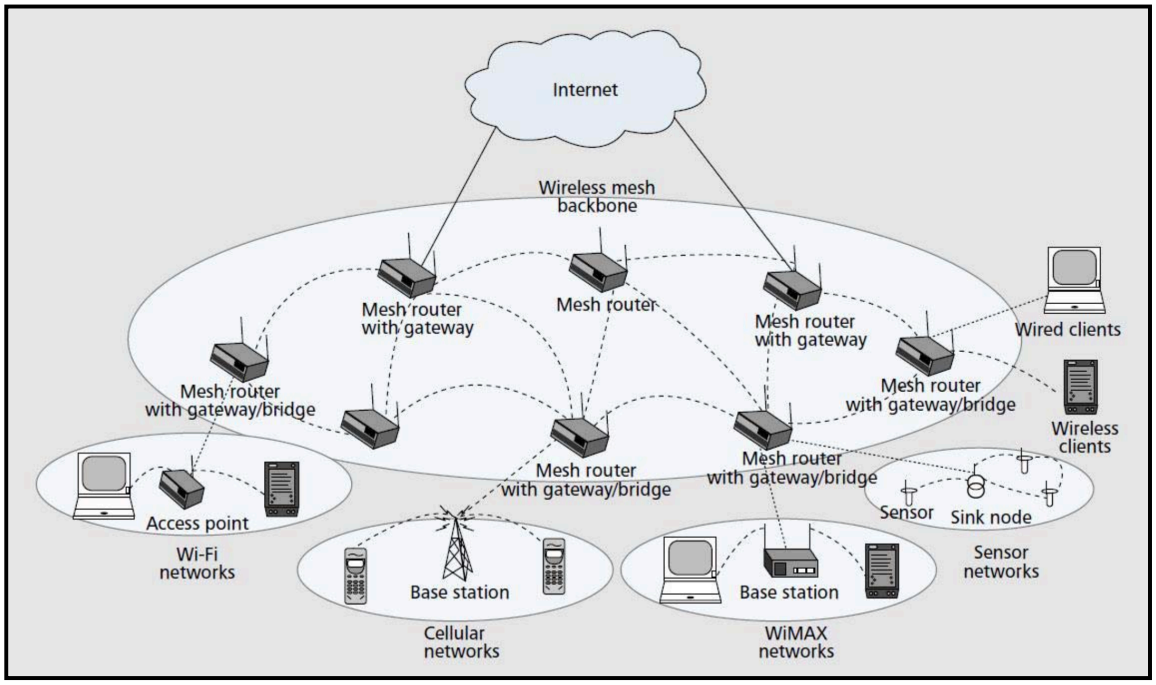

Figure 2.1: The wireless mesh network architecture [13].

tablets, mobile or wearable tech devices.

2. *Wireless Mesh Routers* are the middle point between Mesh Clients and Mesh Gateways and route the traffic in WMN.

3. *Wireless Mesh Gateways* are used to communicate with another WMN or to the internet.

2.2 Routing protocols in WMN

WMN are self-organized networks which do not require a central hub or switch. The core of this property lies in the routing protocol, which plays significant part in the execution. Many metrics have been proposed for developing routing algorithms in WMN, such as:

1. Number of hops (Hop Count)

2. Connection quality (Link Quality)

3. Network load (Load-Dependent Metrics)

4. Expected transmission count (calculating expected number of MAC layer transmissions for one delivery.)

Based on these metrics, there are many routing protocols developed in WMN that can be broadly classify in two kinds:

- *Proactive Routing Protocols* continuously maintain one or a number of routing tables that store routes at the given time. They help to recurrently send packets along the network to exchange and update information in neighbouring nodes.
- *Reactive Routing Protocols* that receive information only when data transmission is to be effected (on demand). Since they don't pre-calculate anything, they don't generate any additional traffic in the network but they take more time when the transmission happens.

Ad-hoc On-demand Distance Vector (AODV) and Dynamic Source Routing (DSR) are considered to be the core routing protocols used in WMN. They are reactive routing protocols and work on hop count metrics [14]. Since security is particularly dependent on the choice of the routing protocol, the most widely used protocol AODV is chosen.

2.2.1 Ad-hoc on-Demand Vector Routing Protocol (AODV)

AODV is a reactive routing protocol where a question-reply mechanism is used to determine the route between two nodes. Route Request (RREQ), Route Response (RREP) and Route Error (RERR) are used for this purpose. When source host wants to start the transmission it broadcasts RREQ packets. The intermediate hosts send the RREP packet if they know the route but if they don't know the route, they forward it until the RREP comes from the destination. If multiple RREPs come from different routes, the route from which RREP comes first is selected. If the intermediate hosts do not know the route, they reply back with RERR, this process is then repeated until RREP comes from the destination. Then the data is forwarded over the route.

2.3 Network attacks in WMN

In WMN, network attacks can be categorized as insider and outsider attacks. Insider attackers have access to one or more hosts in the network through breaking the

cryptographic keys and being authenticated by the network. Since they exist in the network as a normal host, it is difficult to detect their presence. On the other hand, outsider attacks are mostly done on the physical layer (e.g. jamming attacks). This research only deals with insiders attacks and the latter part is out of the scope. The classification of WMN network attacks with respect to the detection process is crucial. Insider attacks can be classified in the following types, as mentioned in [4]:

1. Local attacks (can be detected by gathering data locally and no previous information is needed).
2. Local correlation attacks (can be detected locally by gathering data).
3. Distributed correlation attacks (can only be detected by gathering information from distributed sources).

This classification gives various advantages while developing the IDS. It helps to keep track of the attacks covered by the system. Since there is no single hub/router in WMN, it is natural to make a distributed environment to detect attacks. By making a distributed architecture, local correlation attacks can also be detected easily. In latter sections, attacks that are considered in this research are given.

2.3.1 Packet dropping attacks

The most common and the easiest way to achieve the state of Denial of Service (DoS) in WMN is to drop all incoming packets. DoS condition is when the resources are made unavailable to the intended user. Attacker may control any host and drop every packet that going through it. It causes serious damage in the performance of the network.

2.3.2 Jellyfish attacks

Jellyfish attacks work with the correlation of Transmission Control Protocol (TCP) and maliciously drop some packets achieving DoS and effecting network throughput. These attacks, as discussed in [15], can further be classified in three types:

- Jellyfish periodic dropping attacks
- Jellyfish reordering attacks
- Jellyfish delay variance attacks

2.3.3 Backhole attacks

In blackhole attacks, a compromised host broadcasts the routing information claiming the best route to a particular destination. Other hosts, thus send their packets to the compromised host, believing it to be the best route where packets are selectively dropped achieving DoS.

2.3.4 Selective packet dropping attacks

This type of attack prevents certain hosts to avoid forwarding packets to a particular destination and could cause a DoS scenario over the network. This selective dropping can be done through probability, by total randomness or by selectively dropping certain types of packet to block a certain traffic.

Table 2.1: Types of network attacks in WMN

Attacks	**Primary focus**	**Category**
Packet dropping attacks	Denial of service	Data routing attack
Jellyfish attacks	Performance Degradation	Data routing attack
Blackhole attacks	Denial of service	Data routing attack
Message fabrication attacks	Creating false information for performance degradation and Quality of Service (QoS)	Data
Fuzzing attacks	Denial of service	Data
Man in the middle attacks	Violating data integrity	Data
Colluding mis-relay attacks	Denial of service	Data routing attack
Selfish attacks	Performance degradation	Data routing attack
Selective packet dropping	Performance degradation and denial of service	Data routing attack
Wormhole attacks	Denial of service	Data routing attack

2.4 Intrusion detection systems

The core concept of IDS lies in the importance of secondary defence mechanism. In computer networks, security is implemented on different levels to prevent the violation of security policies. Mostly, firewall is the first line of defence against any potential threat. Firewall restricts access to the network by blocking traffic from unreliable sources.

If the firewall is breached and the intruder managed to get access to the network, the IDS detects the violation and recognizes the possibility of intruder being inside the network. Most commonly, these systems are implemented in wired networks (LAN) but with the evolution of wireless networks such as MANETs, Ad-hoc networks and WMN, it is important to implement these systems for reliability. IDS can be classified in two categories:

- Anomaly based intrusion detection systems
- Misuse based intrusion detection systems

2.4.1 Anomaly based intrusion detection systems

Typically, an anomaly based model observes the network after carefully analyzing the traffic in normal scenarios. If the network shows any difference in behaviour or diverts from its usual conditions, it generates an alarm. This idea can be modified by only analyzing the abnormalities in certain conditions, according to the requirement. Since it studies the behaviour of the network, many anomaly based models can generate false alarms.

2.4.2 Misuse based intrusion detection systems

In misuse detection or signature based detection, an attack pattern or signature is identified and stored for future comparison with the network traffic. While anomaly based model works with abnormalities in the network, misuse based detection looks for attacks. As [16] claims, while misuse based detection is good for specified and well-known attacks, it cannot detect unusual and less-known attacks.

Chapter 3

Hybrid Intrusion Detection System (HIDS)

An Intrusion Detection System (IDS) monitors current processes in a particular environment and evaluates the state of the network by identifying anomalies or by verifying signatures. In the case of WMNs, there are many factors that make intrusion detection complicated as discussed in Chapter 2. Thus, it is important to exploit the features of the mesh topology to improve IDS. Conceptually, the goal is to analyze hosts in a network and to be able to tell if any behave inappropriately.

The IDS presented in this thesis is referred to as hybrid because it takes features from both anomaly based detection and misuse based detection. It detects anomalies by constantly exchanging messages between hosts but instead of pre-evaluating the network behaviour, it takes the previous reputation matrix as a signature for the next cycle of reputation management. A cycle represents one complete HIDS iteration of evaluating all hosts in the network. The hybrid intrusion detection framework takes advantage of the mesh topology by considering that each host already has a level of trust towards its neighbouring hosts as they are constantly exchanging traffic. If this trust value can be quantified, it can be used to detect intrusions.

3.1 Communication model and assumptions

Communication model in this thesis employs to traditional mesh network structures such as TCP/IP, MAC address and packet headers. All hosts are able to send/receive packets to all other hosts within their wireless range. Model follows AODV routing

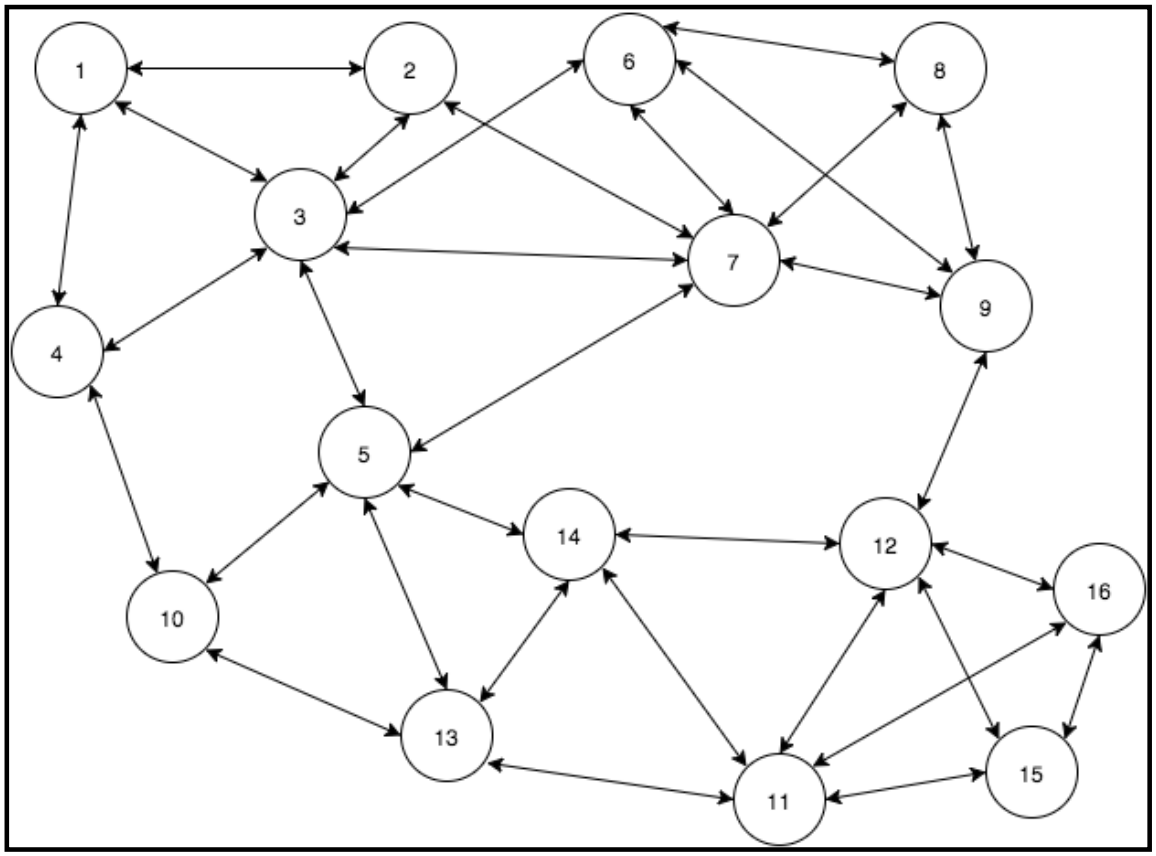

Figure 3.1: A 16 host wireless mesh network

protocol which is reactive (See Chapter 2). For each transmission, a RREQ packet is broadcast and data is transmitted via the best route acquired. If one host is not in the transmission range of another host, intermediate hosts are able to relay packets as described in the AODV protocol. Another important assumption in the communication model is that the HIDS host has knowledge of all hosts in the network through firewall. HIDS also has a view of the complete network; it can roam freely within the network and it can contact any host at any time.

3.2 Attacker model

This section describes the capabilities and limitations of the attacker model followed in this research. Over the time, many attacker models for different types of networks have been introduced. This research takes inspiration from the attacker models provided in [5] and [17] for MANETs. Network attacks mentioned in Section 2.3 are

based on packet dropping, hence individual attacks are not modelled in this thesis. Total and selective packet dropping is implemented to evaluate the proposed HIDS in Chapter 4. The packet dropping attack is effective for all hosts at a given time.

3.2.1 Controlling capabilities

In this thesis, the byzantine behaviour [18] is assumed for the intruder to control multiple legitimate hosts. Byzantine behaviour is when an attacker takes full control of one or more legitimate hosts in a network and behaves arbitrarily to disrupt network performance. Only insider attackers are considered in this research. Thus, routing attacks but not necessarily outsider attacks are detected by the proposed IDS.

3.2.2 Communication model

An attacker is able to communicate with any host in the network. It can perform any routing behaviour such as send, receive, forward or drop packets. It behaves according to the routing protocol, hence only AODV route packets are considered in this thesis. Attacker uses the same hardware and radio capabilities as normal hosts in the network.

3.3 Working principle

The process of HIDS relies on the mesh topology of WMN, because it quantifies one host's opinion about another. A host can have a very large number of mesh connections, making the study of the network intractable. It is likewise impossible to keep track of all host's opinions for every other host. For example, Figure 3.1 shows a 16 host network connected in the mesh topology. Host 7 is connected to six other hosts, so there are 6 different opinions calculated for host 7 which may contain redundancies. Studying many connections to a single host will take more time and processing cycles. Thus, it is important to come up with a methodology to divide the network in an appropriate way that decreases redundancy while keeping it reliable. The process of HIDS is divided in two broad phases, as shown in Figure 3.2.

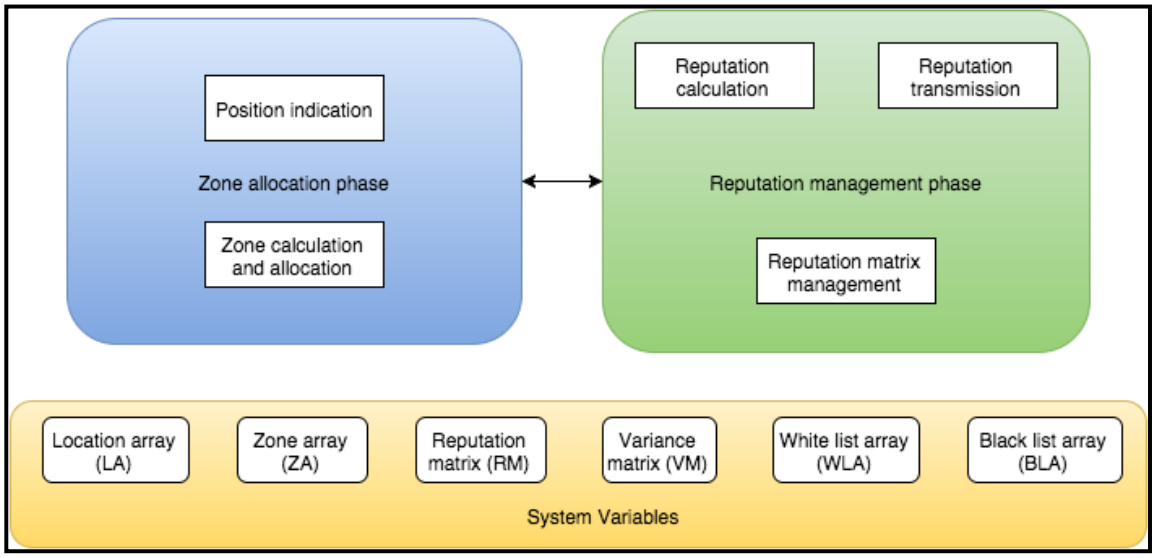

Figure 3.2: The HIDS model with system variables

3.3.1 Zone allocation phase

The network is divided into zones to decrease redundancy, cost, complexity and resources required for reputation management. This thesis introduces an algorithm to divide a network so that every node is part of a zone. Algorithm 1 is responsible to execute zone allocation phase where the HIDS first queries all hosts in the mesh network to report their current position (see Figure 3.3). It then receives the location messages from hosts and starts allocating hosts in zones based on their geographical coordinates. Hosts that are close to each other are zoned together as long as they lie under maximum host count. The effect of increasing or decreasing the number of hosts in a zone is discussed with great detail in the Chapter 4. The radius of a zone (R_{avg}) is set to 1.5x average wireless range of the devices in the network. This keeps the zone size moderate with respect to the total network size and reduces the false alarm rate. This factor is for evaluation purposes only and the study of varying the zone size is beyond the scope of this thesis.

Figure 3.5 provides a flowchart of the execution of Algorithm 1. To understand Algorithm 1, consider the 16 host mesh network shown in Figure 3.4 where host 3 is to be assigned in a zone. $\_ST\_STAMP$ is the starting time stamp and $\_CL\_STAMP$ is the current cycle time stamp of HIDS. $\_AC\_STAMP$ is the time stamp to record the database update time. Intrusion detection is a continuous process hence presented in a *while(true)* loop which runs indefinitely. HIDS queries each host in the network

```
message inet::GenericAppMsg;

// Query Message:
// This is a query message sent by HIDS host
//
message QueryMessage extends inet::GenericAppMsg {
    string nodeName;
    double queryString;
}
message inet::GenericAppMsg;

// Location Message:
// This is a location message from normal hosts
//
message LocationMessage extends inet::GenericAppMsg {
    string nodeName;
    double locationX;
    double locationY;
    double locationZ;
    double rangeRadius;
}
```

Figure 3.3: Query message and location message as used in the simulations

to send their location information. Once all hosts have reported their positions to the HIDS, it stores the information in Location Array (LA). In Algorithm 1, *for* loop for *totalHost* indicates that the process of zone allocation is run for all hosts (which in this case is for host 3). It is important that a host is not put in two different zones simultaneously. Thus, algorithm confirms if the current host is already allocated to any zone by verifying *host.isAssigned*. It executes another *for* loop for *remainingHosts* to analyze the location of other hosts. If the current zone is not full, it starts the process of calculating smallest distance between host 3 and the remaining hosts. It determines host 1, 2 and 4 as the nearest hosts and mark them as $isAssigned = true$. Algorithm then verifies if the current zone is full by comparing *currentZoneHostCount* with the *maximumCount* and set *isZoneFull* variable. It then breaks further processing of the loop for host 3. On the other hand if the zone is already full, it puts the zone in Zone Array (ZA) and continues the process for host 3. Once all hosts are assigned, HIDS transmits LA and ZA to the hosts and waits for the next cycle in a *while* loop by comparing the difference of current time and $\_CL\_STAMP$ with *threshold* time. *threshold* is the waiting time until next zoning phase. It also waits until a new connection arrives which sets *isNewConnection*.

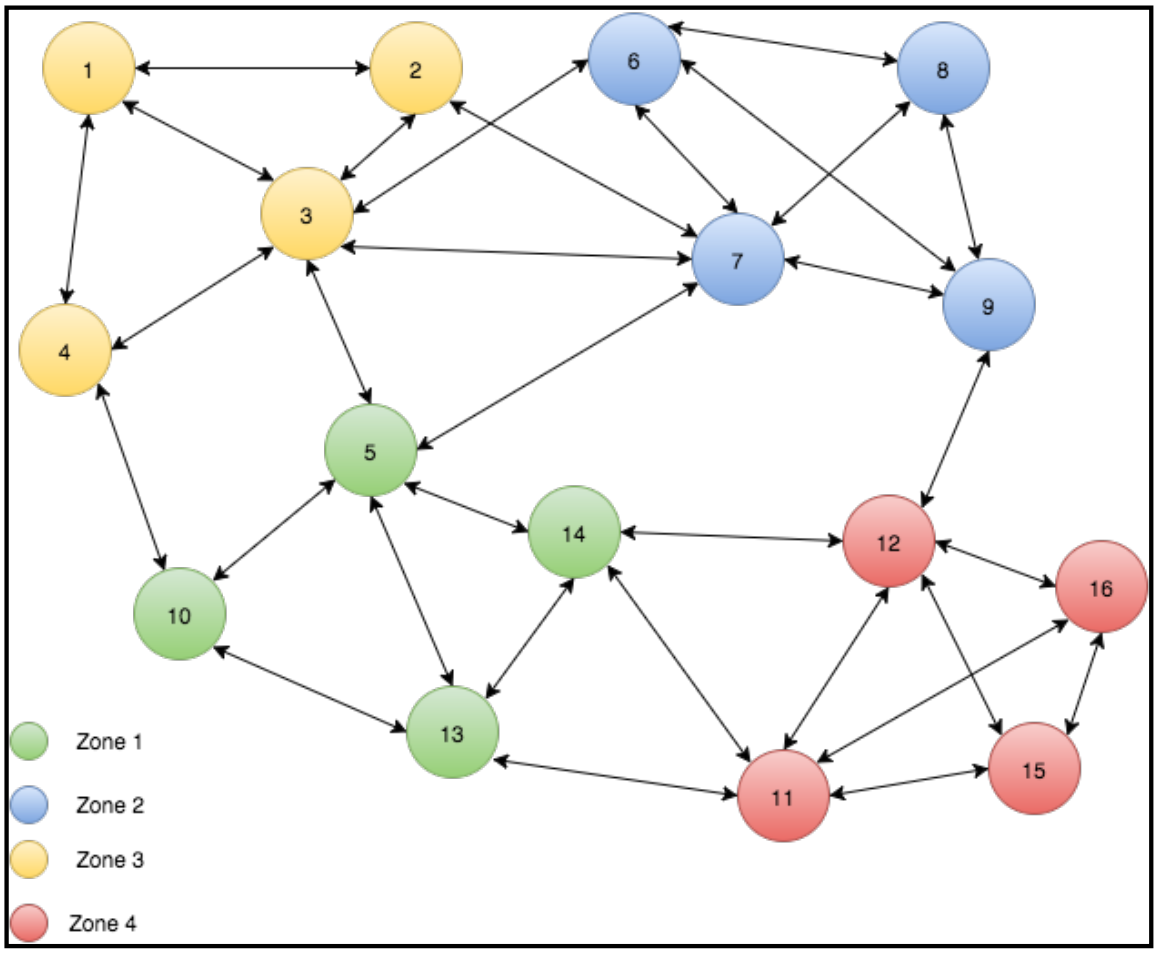

Figure 3.4: A 16 host mesh network divided in 4 zones with 4 hosts per zone configuration

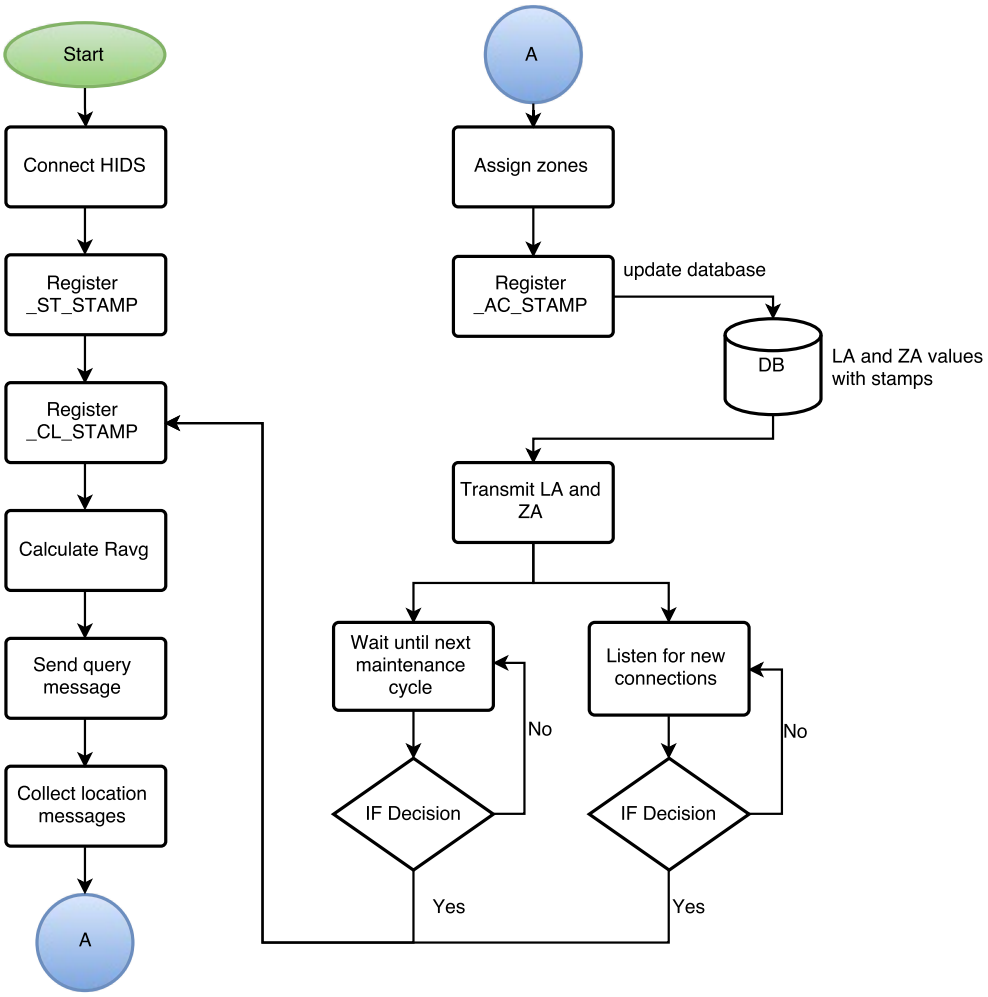

Figure 3.5: Zone allocation flowchart

Algorithm 1: Zone allocation algorithm

```
input  : R_avg,n, maximumCount = 3, 5 and 10
output: LA, ZA
initialization;
_ST_STAMP ← currenttime;
while true do
    _CL_STAMP ← currenttime;
    Send query message;
    LA ← collect location messages
    R_avg ← R_avg × 1.5;
    forall the host ∈ totalHosts do
        if host.isAssigned == false then
            forall the R ∈ remainingHosts do
                compareSmallestDistance();
            end
            if isZoneFull == false then
                host.isAssigned ← true;
                R.isAssigned ← true;
                if currentZoneHostCount > maximumCount then
                    isZoneFull ← true;
                end
                break;
            else
                ZA ← zone;
                isZoneFull ← false;
                continue for the same host;
            end
        end
    end
    _AC_STAMP ← currenttime;
    transmit LA and ZA;
    while (currentTime−_CL_STAMP < threshold) ||!isNewConnection do
        wait();
    end
end
```

3.3.2 Reputation management phase

Reputation management is a way to quantify a host's opinion about it's neighbours within a zone. In WMN, unlike a normal computer network, a host acts as a router and forwards incoming traffic to the hosts directly connected to it. Those hosts can either accept the traffic (if packets are meant to be for that host) or forward them with the best possible route, depending on the metric followed by the routing protocol. An intruder can selectively drop, manipulate, change or simply reject all incoming traffic. Thus, each host must ensure that the packet forwarded is reaching to its destination.

The fundamental concept behind reputation management is that an opinion is obtained as to whether a host is behaving appropriately. Many types of mesh network attacks including jellyfish attacks, blackhole attacks and selective or total packet dropping attacks can be detected by calculating reputations. This thesis proposes a way to quantify the reputations of hosts over time. The reputation management phase has two algorithms:

1. Algorithm 2 - Reputation algorithm at individual host
2. Algorithm 3 - Reputation algorithm at HIDS

The reputation of node y is calculated at node x as:

$$R_{x,y} = \frac{\sum_{t=0}^{t} P_{fwd}}{\sum_{t=0}^{t} P_{total} - \sum_{t=0}^{t} P_j}$$

Reputation is a ratio between total packets forwarded (P_{fwd}) over the total number of packets that had to be forwarded. The denominator is a difference between total packet sent to the host y (P_{total}) and packets for which host y was the destination (P_j). The difference thus gives total count of packets host y is responsible to forward.

To understand Algorithm 2 (refer Figure 3.7), consider host 5, 10, 13 and 14 sorted in zone 1. Figure 3.4 shows that host 10 has two directly connected hosts. It calculates their reputations and reports back to the HIDS. Similarly, host 5 has three directly connected hosts, host 14 which has two directly connected hosts and host 13 has three directly connected hosts. So, host 5 calculates $R_{5,10}, R_{5,13}$ and $R_{5,14}$. The equation is executed by host 5 three times by sending a test message to host 10, 13 and 14 respectively. Algorithm 2 is distributed in two threads where the first thread is dedicated to calculate and manage reputations and the second thread is dedicated to listen HIDS calls for removing flagged hosts from the routing table.

Reputation management is an indefinite process hence both threads run in a *while(true)* loop. HIDS stores current time in $\_LCOFG\_STAMP$ which is the last configuration time stamp. Host 5 makes a dummy message and send it to host 10, 13 and 14 seperately. It takes the *totalPacketCount* and saves it in P_{total}. Algorithm then runs a for loop for all received packets to save them in a buffer. It calculates the sum of all received packets and save it in P_{fwd}. The calculated reputations $R_{5,10}, R_{5,13}$ and $R_{5,14}$ are transmitted to the HIDS in the form of *DummyMessageForReputation* presented in Figure 3.6. The thread waits for the next cycle by comparing the difference between *currentTime* and $\_LCOFG\_STAMP$ with *threshold* which is the duration between two cycles.

Thread II listens to incoming HIDS calls for routing table. If HIDS call arrives, algorithm sets *callReceived* variable and delete the flagged host from routing table and wait until next call arrives.

```
cplusplus {{
#include "GenericAppMsg_m.h"
}}
message inet::GenericAppMsg;

message DummyMessageForReputation extends inet::GenericAppMsg {
    long reputation;
    string testerNode;
    string testeeNode;
    string messageString;
    double sentTime;
    double recieveTime;
}
```

Figure 3.6: A dummy reputation message as used in simulation

Algorithm 3 refers to HIDS algorithm where individual hosts send back their calculated reputations to HIDS. A visual representation of Algorithm 3 is presented in Figure 3.8. Thread I shows the information gathering where $\_RMS\_STAMP$ is set with the current time indicating the last reputation management stamp. The thread runs in an infinite loop where it sets the variable *reputationArrives* if reputation is arrived from a host. It then receives all incoming messages and starts Thread II which is responsible to put this reputation in a reputation matrix. $\_LIGT\_STAMP$ is set at this point representing last variance calculation stamp. Reputation matrix is given by $R_{t=i}[totalHostCount, totalHostCount]$, indicating that if the mesh network has 30 hosts, the matrix will be of $R_{t=i}[30, 30]$, where $i = 0, 1, 2, 3, \ldots, k$ where k is

the count of reputation management cycles. Once the matrix is saved, variance is calculated for the reputations of individual hosts to study the change of reputation from its mean value. Sudden change of reputation for any host will raise an alarm. The variance is calculated by the following formula:

$$V_{(xy)k} = \frac{1}{n}\sum_{i=0}^{n}(R_{(xy)i} - \overline{R}_{(xy)i})^2$$

$V_{(xy)k}$ shows the variance for the reputation $R_{(xy)}$ at time instance k. $k+1$ and k are two consecutive cycles for reputation management. If the change in variance of $k+1$ and k is greater than threshold value, it will generate an alarm.

$$\Delta V_{xy} \geq threshold \rightarrow alarm$$

The threshold depends on the probability of packet dropping in the network and is correlated to the amount of background traffic. Given a typical level of background traffic a threshold of 0.1 is used for illustration purposes in this thesis. Once a host is flagged, HIDS tells other hosts in the zone to avoid using routes that include the potentially compromised host. Refer to Figure 3.4, if host 7 is told that host 8 is compromised, it avoids all routes associated to host 8 and updates its routing tables. Host 6 and host 9 will do the same, because they are in the same zone. HIDS maintains an array of all flagged hosts indicated as CA (Corrupted host Array) with their relative zones.

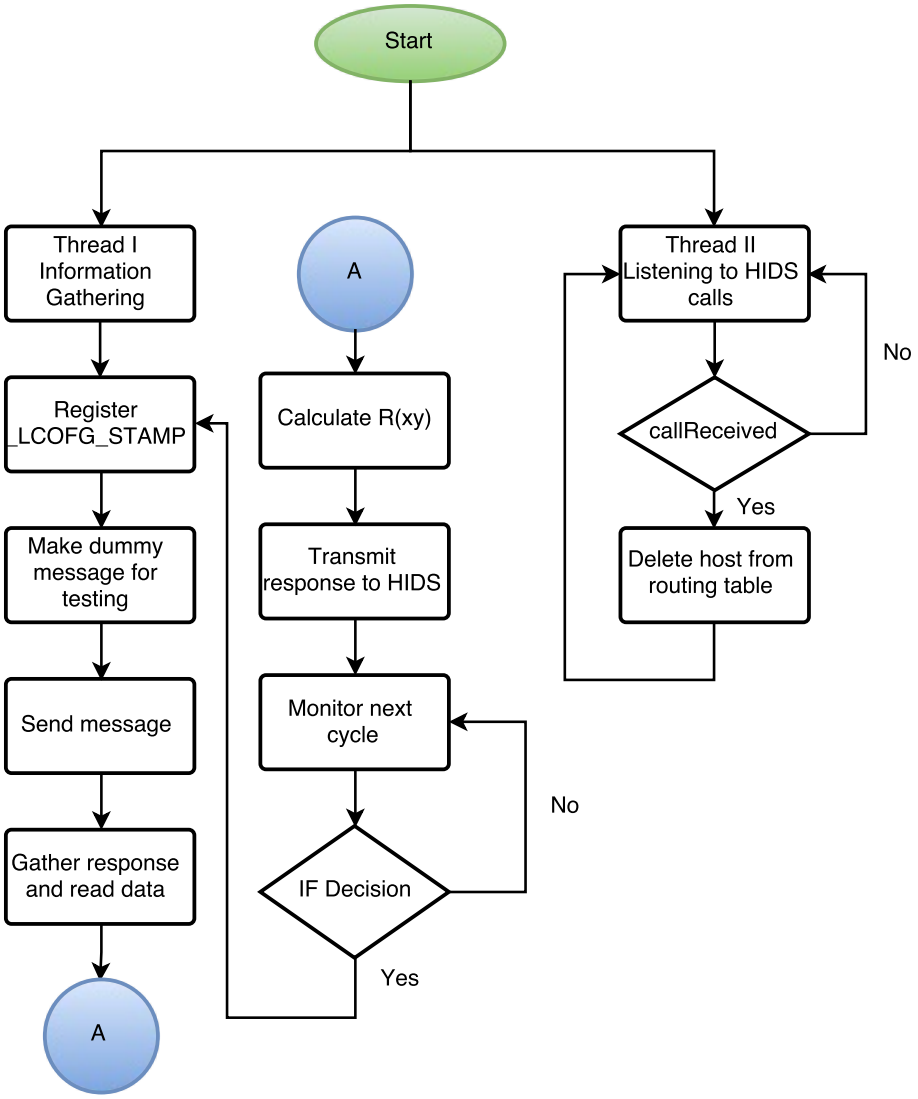

Figure 3.7: Host reputation flowchart

Algorithm 2: Reputation algorithm at host

input : $ZA_{relevantZone}$
output: R_{xy}
initialization;
start Thread I and Thread II;
Thread I: Information gathering
while *true* **do**
  $\_LCOFG\_STAMP \leftarrow currenttime$;
  makeDummyMessageForTesting();
  sendMessage();
  $P_{total} \leftarrow$ totalPacketCount;
  forall the $packet \in packets$ **do**
    gatherResponseAndReadData();
    totalPacketRecieve++;
  end
  $P_{fwd} \leftarrow$ totalPacketRecieve;
  $R_{x,y} \leftarrow \frac{\sum_{t=0}^{t} P_{fwd}}{\sum_{t=0}^{t} P_{total} - \sum_{t=0}^{t} P_j}$
  transmit $R_{x,y}$ to HIDS;
  $P_{fwd} \leftarrow$null; $P_{total} \leftarrow$null;
  while $(currentTime - \_LCOFG\_STAMP < threshold)$ **do**
    wait();
  end
end
Thread II: Listening to HIDS calls
while *true* **do**
  if $callReceived == true$ **then**
    deleteFlaggedNodeFromRoutingTable();
    $callReceived \leftarrow$ false;
  else
    wait();
  end
end

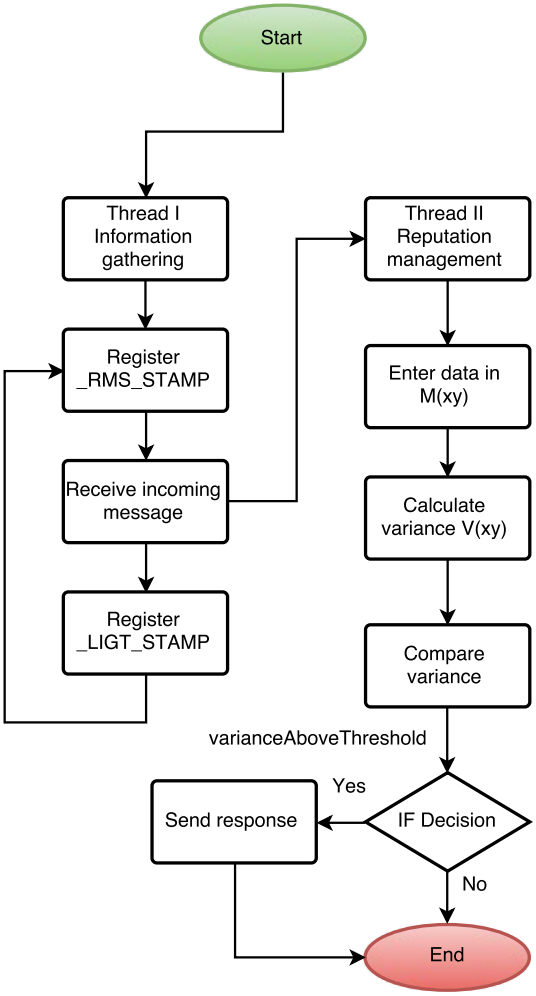

Figure 3.8: HIDS reputation flowchart

Algorithm 3: Reputation algorithm at HIDS

input : R_{xy}
output: $flag$
initialization;
start Thread I;
Thread I: Information gathering
$\_RMS\_STAMP \leftarrow currenttime$;
while *true* **do**
  if *reputationArrives* **then**
    receiveIncomingMessages();
    start Thread II;
    $\_LIGT\_STAMP \leftarrow currenttime$;
  else
    wait();
  end
end
Thread II: Reputation Management
$V_{(xy)k} \leftarrow \frac{1}{n} \sum_{i=0}^{n} (R_{(xy)i} - \overline{R}_{(xy)i})^2$;
comparingVariance();
if $varianceAboveThreshold == true$ **then**
  $CA \leftarrow flaggedhost$;
  sendResponse();
end

Chapter 4

Implementation and Performance Evaluation

In this chapter, the HIDS framework is simulated to test the performance of the proposed algorithms. This chapter contains an overview of the network simulation environment and performance analysis done for both phases presented in the previous chapter. The results and their implications are discussed.

4.1 Network simulators for WMN

In [19], authors suggested that due to practicality and scalability, testbeds are more popular in research on wireless networks. For performance evaluation, some practical testbeds exist for WMNs such as Roofnet, MeshNet and ORBIT. Although some researchers prefer testbeds for experimentation, the importance of simulation cannot be neglected. Testbeds have physical limitations, hardware costs and limited scalability, while simulations are easily expandable, have almost no cost and no physical limitations. Many simulators contain a wireless module for routing, MAC and physical layer protocols. In Table 4.1, the pros and cons of different network simulators are given from a WMN perspective.

Table 4.1: Comparison of major network simulators for WMN

	NS-2	**NS-3**	**OPNET**	**OMNET++**	**Qualnet**
Programming language	oTcl C++	Python C++	C C++	C++	Parsec
Documentation and support	Excellent	Excellent	Excellent	Good	Good
Scalability	Medium	Medium large	Medium	Large	Very large
Software price	Free	Free	Proprietary	Free	Proprietary
Graphical support	No	Limited	Yes	Yes	Yes

Open-source and proprietary simulators are considered for performance evaluation of HIDS because they allow testing in different scenarios without hardware and maintenance cost of testbeds. Omnet++ is chosen to simulate HIDS because:

- Graphical support is given to visually analyze network behaviour.
- Distributed under open-source license.
- Provides global community to troubleshoot problems and documentation.
- Scalable for large wireless networks with the support of C++.

4.1.1 Omnet++

Omnet++ is an open-source network simulator distributed under the GNU Academic Public License. Omnet++ is free for non-commercial use such as research and study purposes. Eclipse IDE with C++ can be downloaded online for Linux, Microsoft and Mac OS X. While it is easily configurable on Linux and Microsoft machines, gdb-debug is not available for OS X Yosemite and El-Capitan, resulting in debug problems. It is a known problem and will be solved in near future.

INET is an open-source OMNET++ model suite for wired and wireless network. INET v3.1.1 is used in this research which contains modules for WMN such as physical layer and routing layer modules. Although Omnet++ comes with basic examples in INET, knowledge of C++ and Network Description (NED) language is required.

4.2 Simulation setup and environment

Omnet++ gives a detailed level of customization to develop networks, especially with INET where many physical, MAC and network layer models are available to setup the environment. Since, HIDS mainly focuses on intrusion detection, dynamic MAC and IP allocation functionalities of Omnet++ were used to save time to select network details. The simulation parameters are given in Table 4.2.

Table 4.2: System and simulation parameters for HIDS

Parameters	Value/Description
System parameters	
Operating system	Mac OS X El Capitan v10.11.1
Processor	2.4 GHz Intel Core i5
Memory	8GB 1600 MHz DDR3
Storage	Macintosh HD - Solid State Device (SSD)
Simulation parameters	
Number of hosts	15-60 hosts depending on simulation
Number of fixed hosts	3 hosts (one HIDS host, two client hosts)
Proximity size	1000m x 1000m
Mesh protocol	AODV routing protocol
Tx power	20 dBm
Maximum communication range	300m
Data rate	Upto 2Mbit/s
Traffic type	TCP packets
Mobility	Static mobility, slow mobility (at 5m/sec), fast mobility (at 10m/sec)

An OMNET project contains all the files and modules developed for the simulation. The mobility module RandomWPMobility in INET v3.1.1 was used to create travel paths for hosts. It also chooses random speeds if the speed is not defined in .ini file of OMNET project. When a host reaches its target position, a new position is randomly selected and the host starts to move in that direction, as long as it lies within the constraints of physical proximity of the network.

Three parameters are used here as IDS metrics:

- Bytes transferred (amount of data transferring during execution).
- Execution time (Omnet++ execution time for one cycle of the HIDS).

- False Positive Response Rate - FPR (rate of false alarms during the execution of HIDS).

4.3 Implementation of HIDS in Omnet++

HIDS is implemented in Omnet using C++ and NED. A client-server architecture is followed for exchanging location and reputation messages. As described in the previous section, INET comes with pre-built modules that can be used for the development of wireless networks. WMN is created using *AODVRouter* host array and a fixed HIDS host as shown in Figure 4.1. *IPv4NetworkConfigurator* is used to self configure IPv4 addresses in WMN and *RoutingTableRecorder* is used to delete routes as mentioned in Algorithm 3.

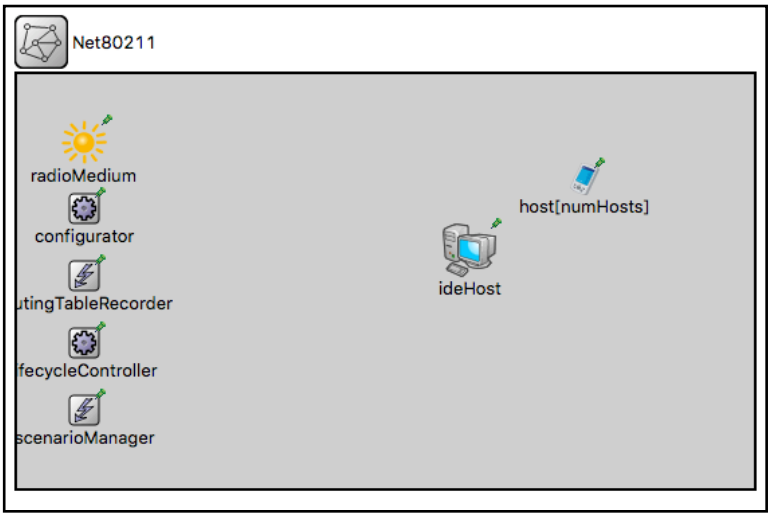

Figure 4.1: Omnet++ WMN schematic as used in the simulation

Figure 4.1 is a graphical representation of the structure of .ned file which works with omnetpp.ini file developed in Omnet++ project. A description of these files is given below:

- .ned files consist of the variables of modules such as AODV router module, IPv4 configurator module, Routing table module and other physical layer modules.
- omnetpp.ini file describes the parameters of modules defined in .ned file such as bitrate, mac header length and mobility method. It also connects C++ classes to modules.

- .cc and .h files describe the working of custom modules developed in the simulation such as, implementing Algorithm 1, 2 and 3 described in Chapter 3 for HIDS.

INET simplifies the implementation of IP or MAC layers in Omnet++, but it is not sufficient to transfer custom messages as needed by HIDS. To solve this problem, TCP layer applications were developed for HIDS and hosts. These applications work in coordination with each other, where one application sends data to a TCP connection (presented in the algorithm in Chapter 3), while the other application waits for a new message to arrive and sets the boolean variables *isNewConnection, callReceived* and *reputationArrives*.

TCP layer applications work as server or client based on the type of message and direction of information flow. These applications are developed as Omnet++ modules which run C++ classes written for simulation. Following steps are taken to develop a module in Omnet++:

- Create a .ned file defining module by extending ITCPApp.
- Write the C++ source and header files and linking it to the module created in .ned file using the *Define_Module()* function

Table 4.3: Omnet++ terminologies

Terminology	**Description**
gdb-debug	Standard debugger for GNU operating systems.
INET	An open-source OMNET++ model suite for wireless networks.
RandomWPMobility	A random mobility module given in INET suite.
ITCPApp	A generic interface given in INET to implement TCP application.
TCPGenericSrvApp	An example module for server application built using ITCPApp.
TCPBasicClientApp	An example module for client application built using ITCPApp.
localPort	.net file parameter for local port to receive incoming connections.
connectPort	.ned file parameter to describe a port for outgoing connections.
thinkTime	.ned file parameter to describe a think time before transmission.
reconnectInterval	.ned file parameter to describe a time before a link is reconnected.
requestLength	.ned file parameter to describe the length of request.
replyLength	.ned file parameter to describe the length of reply requested.
rcvdPk	.ned file parameter to describe the incoming physical layer signal.
sentPk	.ned file parameter to describe the outgoing physical layer signal.
GenericAppMsg	An interface given in INET to make custom TCP messages.

4.3.1 Defining TCP applications using ITCP interface

ITCPApp is a generic interface given in INET that uses socket programming to send and receive data from the TCP layer. Six default applications are given in INET source code (as shown in Figure 4.2), which can be modified as needed. In the case of HIDS, an application needs to act as server and client simultaneously, *TCPGenericSrvApp* and *TCPBasicClientApp* modules are merged to allow this behaviour. The new ITCPApps are named as *TCPPeerHIDS* and *TCPPeerHost* for their respective use. As shown in Figure 4.1, *TCPPeerHIDS* works at *ideHost* whereas *TCPPeerHost* works at individual hosts as described in *host[]* array.

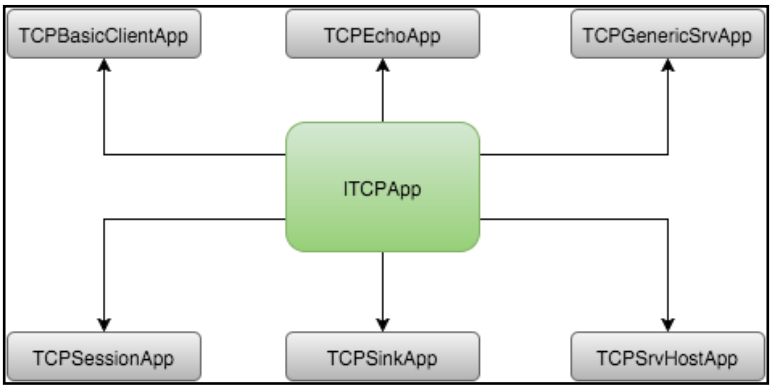

Figure 4.2: Default TCP applications found in INET

By using ITCPApp interface, some important parameters can be set in .ned files including *localPort, connectPort, thinkTime, idleInterval, reconnectInterval, requestLength* and *replyLength*. TCP applications developed for HIDS use *rcvdPk* and *sentPk* signals to handle incoming and outgoing packets on TCP connection. These variables are also used for calculating end-to-end delay and bytes transferred metrics. While working with ITCPApp, *dataTransferMode* should be set as *object* to ensure that the data transfer refers to a custom message object instead of byte stream.

4.3.2 Defining custom messages using *GenericAppMsg*

GenericAppMsg is an INET interface used to define custom messages. *ReputationMessage*, *QueryMessage*, *LocationMessage* and *MiscMessage* were developed using *GenericAppMsg* to transfer information across the network. TCP applications were developed to transfer these messages over TCP sockets. When Omnet++ project is built, it generates a C++ class and header for each custom message. These classes

are then referred in modules when desired message needs to be transferred. A short description of each custom message is given below:

- *ReputationMessage* is responsible for transferring reputation from hosts to HIDS.
- *QueryMessage* is used to query hosts when HIDS process needs to start.
- *LocationMessage* is responsible to convey location information from hosts to HIDS.
- *MiscMessage* is used for miscellaneous purposes like transferring zone information, flagging and pinging a host.

4.4 Simulation results and discussion

Simulation results and discussion are presented in this section. The algorithms developed in Chapter 3 were simulated in Omnet++ individually, to compute bytes transferred, execution time (time for one HIDS cycle) and False Positive Response (FPR). HIDS is implemented and simulated in an environment where the effects are measured by changing:

- Mobility (static, slow at 5 m/s and fast at 10 m/s)
- Number of hosts per zone
- Number of zones

The effect of mobility and increasing the number of zones is studied by taking different number of hosts/zone, i.e. taking 3 hosts/zone, 5 hosts/zone and 10 hosts/zone. For each scenario, whole HIDS process is executed (Algorithms 1, 2 and 3 in Chapter 3). It is assumed that all hosts within a zone are connected to each other providing a scenario where the maximum number of reputations within the zone are calculated and transmitted to the HIDS.

4.4.1 Performance with 3 hosts/zone

3 hosts/zone is a small scenario where hosts do not calculate or manage a high number of reputations. In this section, 15, 20 and 30 hosts are organized in 3 hosts/zone configuration. Each host has to calculate and maintain two reputations. These reputations are then forwarded to HIDS and kept in the matrix of $R_{t=i}[15, 15]$, $R_{t=i}[20, 20]$ and $R_{t=i}[30, 30]$ respectively. More information about these matrix is given in the next section. Results acquired from 3 hosts/zone scenario by varying total number of hosts are given in Table 4.4.

Table 4.4: Results for static, slow and fast mobility for 3 hosts/zone

Mobility	**Hosts**	**Zones**	**Execution time (min)**	**Bytes transferred (MB)**
Static	15	5	1.93	0.41
	20	7	4.20	0.76
	30	10	7.23	1.45
Slow	15	5	2.85	0.46
	20	7	5.99	1.05
	30	10	9.14	2.08
Fast	15	5	4.92	0.62
	20	7	8.62	1.60
	30	10	13.66	3.16

Discussion

HIDS shows consistent results while tested in 3 hosts/zone configuration. In Table 4.4, execution time and bytes transferred are increasing while changing the number of hosts from 15 to 30 in the network. Slow mobility is not significantly different from fast mobility in terms of execution time or bytes transferred. For example, the difference between static and slow mobility for 15 hosts is 0.92 min with 44.55 KB increase in data. This increase shows 47.3% rise in execution time and 10.5% rise in bytes transferred. The results of static and fast mobility are most different for 15 hosts, where a difference of 2.99 min is noticed. The execution time difference between static and fast mobility is due to constant exchange of routing traffic between hosts because they perform handover in the network. Handover causes route reconfiguration in AODV resulting in increased execution time. In this configuration, execution time and bytes transferred are mainly increased by total number of hosts. The results are

comparable to DogoIDS [3], where 10 s are taken to process 7 MB of data to test one host. On the other hand, the execution time of HIDS is 2.6 s to test one host by transferring 0.02 MB. This is a significant difference as it shows the advantage of considering mesh topology to study the whole mesh network together.

4.4.2 Performance with 5 hosts/zone

The number of hosts per zone is increased to evaluate HIDS in a more dense environment. In this section, total number of hosts are 15, 20 and 30 hosts. There are 5 hosts/zone so each host needs to calculate and transmit four reputations. These reputations are then forwarded to HIDS where they are kept in matrices $R_{t=i}[15, 15]$, $R_{t=i}[20, 20]$ and $R_{t=i}[30, 30]$ respectively. This configuration results in more packets being transferred within a zone and causes more data transfer between hosts and the HIDS. The results acquired by the HIDS are given in Table 4.5.

Table 4.5: Results for static, slow and fast mobility for 5 hosts/zone

Mobility	**Hosts**	**Zones**	**Execution time (min)**	**Bytes transferred (MB)**
Static	15	3	3.22	0.49
	20	4	4.65	0.81
	30	6	11.42	2.31
Slow	15	3	4.16	0.56
	20	4	6.45	1.23
	30	6	16.60	4.05
Fast	15	3	5.274	0.71
	20	4	9.88	1.93
	30	6	21.50	5.13

Discussion

Three mobilities are considered to study the performance of HIDS. Results show that the execution time and bytes transferred increase when the number of hosts increases from 15 to 30. The mobility does not significantly influence execution time and causes a difference of 0.94 min which makes 29.1% increase (moving from static to slow motion of 5 m/sec while measured for 15 hosts). Similarly, bytes transferred also shows minor differences and a difference of only 71.17 KB of data is recorded giving a percentage increase of 14.2%(moving from static to slow mobility while measured

for 15 hosts). As found in the previous section, the effect of mobility on execution time and bytes transferred does not change abruptly showing the stable behaviour of HIDS. However, those results are not replicated when the effect is measured going from static to fast mobility with 30 hosts. It doubles the execution time of the algorithm. Similarly, the bytes transferred increases while going from static to fast mobility with 30 hosts. To conclude, the effect of mobility on the HIDS is not prominent on execution time or bytes transferred for 15 hosts but with four reputations per host to be calculated, it increases the execution time and the amount of bytes transferred for 30 hosts.

4.4.3 Performance with 10 hosts/zone

This section describes the worst case scenario where mesh network contains 30, 40 and 60 hosts in a 10 hosts/zone configuration. Each host has to calculate nine reputations. These reputations then go to HIDS where they are sorted and put in a matrix of $R_{t=i}[30, 30]$, $R_{t=i}[40, 40]$ and $R_{t=i}[60, 60]$ respectively. The results of this configuration are given in Table 4.6.

Table 4.6: Results for static, slow and fast mobility for 10 hosts/zone

Mobility	**Hosts**	**Zones**	**Execution time (min)**	**Bytes transferred (MB)**
Static	30	3	13.03	2.09
	40	4	21.50	4.50
	60	6	41.18	7.95
Slow	30	3	18.59	2.86
	40	4	26.93	4.61
	60	6	53.36	11.99
Fast	30	3	25.84	5.89
	40	4	32.87	8.11
	60	6	62.44	13.48

Discussion

The results show minor changes in execution time and bytes transferred while moving from static to slow. The difference in execution time is 5.59 min (42.9% increase) for 30 hosts, while a difference of 12.18 min (29.5% increase) is observed for 60 hosts in 10 hosts/zone configuration. Moving from static to fast mobility with 60 hosts increases

in execution time and bytes transferred. The difference in execution time is increased by 21.26 min (150% increase) and data increases by 5.53 MB. In conclusion, increase in hosts shows increase in execution time and bytes transferred with fixed mobility. On the other hand, changing mobility by keeping the same number of hosts shows small change in execution time and byte transferred suggesting that HIDS is less effected by mobility. The delay in execution time and increase in bytes transferred is seen because each host takes more processing and transferring time to compute and transmit 9 reputations. This scenario is adopted to study the behaviour of HIDS when mobility is an important factor, where all hosts start moving from 0 to 10 m/s. While this change in speed is not observed normally in mesh networks, it is simulated to study the ability of HIDS to handle big networks with high mobility. 10 hosts/zone configuration is the worst-case scenario for this research, thus it is comparable to the worst-case scenario of DogoIDS where a delay of 16 s is observed for one host to transfer 30 MB. HIDS takes 25.80 s to transfer 71.33 KB. In HIDS, time taken for the execution is more than DogoIDS but it gives significantly less network overhead by transferring 71.33 KB as compare to 30 MB.

4.4.4 Reputation matrix management

After collecting all reputations, HIDS starts sorting them in matrix and compares the matrix with old matrices. The reputation matrix is presented as $R_{t=i}[n, n]$, where n=*totalNumberOfHostInTheNetwork* and $i = 0, 1, \ldots, k$. Where k is total instances for reputation cycles. In this research, three reputation matrices are studied where all reputations are collected and compared for $k = 0$, $k = 1$ and $k = 2$ instances. This section describes reputation matrices of 3 hosts/zone configuration for static mobility. Total number of hosts are 15, which gives reputation matrices of $R_{t=0}[15, 15]$, $R_{t=1}[15, 15]$ and $R_{t=2}[15, 15]$.

Table 4.8 shows the normal behaviour of the network where all hosts are forwarding packets correctly. The reputation is calculated by the formula given in Chapter 3

$$R_{x,y} = \frac{\sum_{t=0}^{t} P_{fwd}}{\sum_{t=0}^{t} P_{total} - \sum_{t=0}^{t} P_j}$$

Ideally, if a host is successfully forwarding all packets, the reputation calculated is 1 which is considered as a normal behaviour. Table 4.8 shows that reputation is being managed at HIDS and hosts are showing $R_{x,y} = 1.0$ for all other hosts present

in their zone. For example, reputation of host 10 is managed by host 6 and host 7 hence in Table 4.8, reputation matrix shows $R_{6,10} = 1.0$ and $R_{7,10} = 1.0$. For the rest of $R_{x,10}$ the value of reputation is zero showing that those hosts do not belong to the same zone.

The attacker model as described in Chapter 2 is followed to simulate incidental packet dropping in Omnet++ where host 7 drops some traffic due to traffic or wireless losses. This incidental dropping is represented in Table 4.9, where the reputation ($R_{6,7}$ and $R_{10,7}$) is reduced to 0.86. This change in reputation will generate an alarm if the variance calculated by HIDS of instances $R_{t=0}[15, 15]$ and $R_{t=1}[15, 15]$ is greater than the threshold. In this section, the threshold is set as 0.1 and the variance of $R_{t=0}[15, 15]$ and $R_{t=1}[15, 15]$ is equal to 0.0093 which causes no alarm.

Table 4.10 shows an attack of total packet dropping, where attacker took control of host 7 to drop all traffic. Since the attacker is not replying back any packets, the reputation ($R_{6,7}$ and $R_{10,7}$) is 0.0. This shows a sudden increase in the variance where $V_{(6,7)2} = 0.37$, $V_{(6,7)2}$ is showing the variance at $k = 2$ for $R_{6,7}$. Same effect is noticed at $R_{10,7}$ because traffic coming from all hosts is being dropped due to the attack. In this case, the variance is exceeding threshold level 0.1 which causes an alarm.

The HIDS processes every reputation separately, so even if an intruder selectively drops packets of only one host, it will be detected by the HIDS. Taking the previous example, suppose $R_{6,7}$ and $R_{10,7}$ show different results ($R_{6,7} = 0.01$ and $R_{10,7}$=0.89). It is assumed that host 7 is showing abnormal behaviour by rejecting packets coming from host 6. This is a case of selective packet dropping by host 7. Therefore, an alarm is generated identifying the behaviour of host 7 towards host 6. The reputation management process yields similar results when tested with 5 hosts/zone and 10 hosts/zone.

4.4.5 False Positive Response Rate (FPR)

The False Positive Response (FPR) is defined as a response generated by IDS in which normal behaviour is classified as attack. If a reputation message does not reach the intrusion detection system and the next cycle of reputation management begins, HIDS considers it as an alert. This behaviour can happen in two scenarios:

- A reputation packet is dropped and never reaches the HIDS.
- A host leaves the mesh network without the HIDS knowing.

Since HIDS depends on exchanging messages between hosts, it takes care of routing related ambiguities. The first scenario, where the reputation packet never reaches HIDS is unlikely because HIDS uses TCP packets. TCP protocol makes sure that a message is successfully transmitted from one end of the network to another end. If packets drop, receiver host does not send back ACK signal to the sender hence packets are re-transmitted. TCP mechanism takes longer to run as it takes care of every packet window sent and received but it is a better way to transmit and receive messages as it ensures less or no packet drop.

The HIDS has access to the routing information of the mesh network. Hence the second scenario where a host leaves without the knowledge of HIDS is unlikely to occur because every change of route is monitored by HIDS. In Chapter 2, it is defined that HIDS monitors the network and keeps track of the hosts. If a host leaves the network, HIDS removes row and column in the reputation matrix. This ensures that in the next cycle, the corresponding reputation of that host is not calculated.

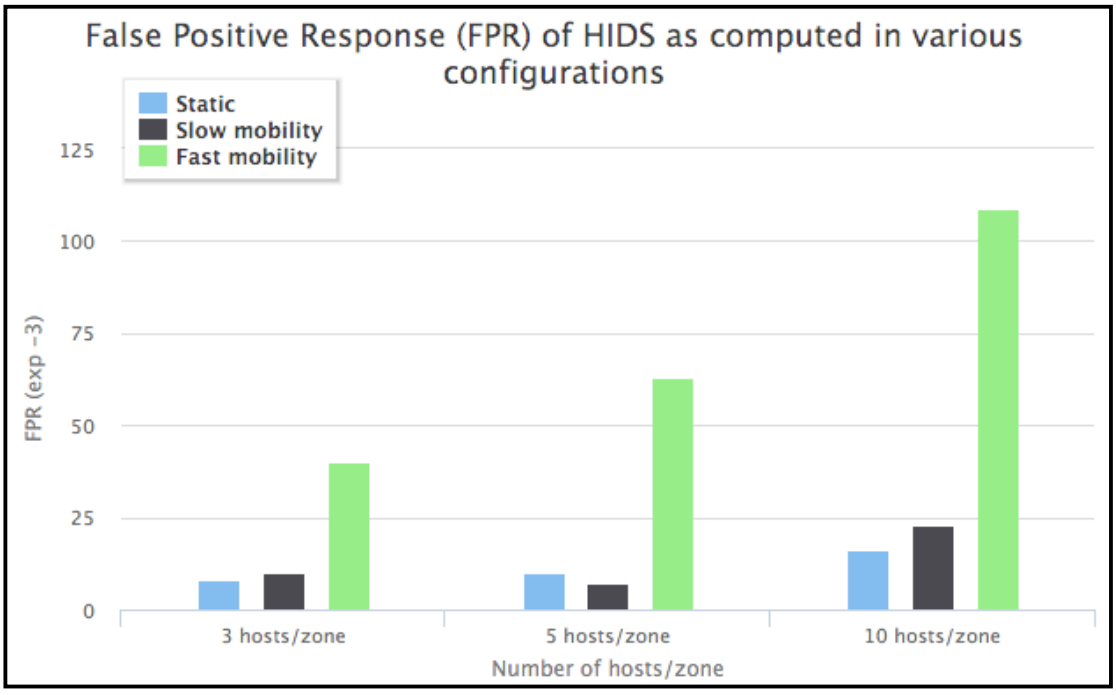

Figure 4.3: FPR for HIDS in 3 hosts/zone, 5 hosts/zone and 10 hosts/zone configurations

In Figure 4.3, the FPR is insignificant for static and slow mobilities for all configurations. For fast mobility, the FPR of 0.047 was observed for 3 hosts/zone, 0.067 for 5 hosts/zones and 0.109 for 10 hosts/zone. Fast mobility causes a higher FPR because

there are more packet losses in fast mobility with the higher number of handovers. Thus, it is possible that the HIDS process will start the next reputation management cycle before a host delivers the reputation. In addition, an increase in FPR is observed when every host in the network moves with a speed of 10 m/s. This mobility scenario is not observed normally in WMN and was adopted to study HIDS behaviour in an extreme scenarios.

DogoIDS analyses one host at a time with static mobility. In [3], FPR calculated for the worst-case scenario is 0.12 for one host. On the other hand, HIDS is giving significantly low FPR by showing 0.013 in the worst case scenario while testing 30 hosts at a given time. This shows that HIDS has a lower FPR than DogoIDS. Table 4.7 gives a comparison between DogoIDS and HIDS. This comparison is when the hosts have no mobility.

Table 4.7: Comparison between HIDS and DogoIDS

	HIDS	**Dogo IDS**
Execution time (best case)	2.6 s to test 1 host	10 s to test 1 host
Execution time (worst case)	25.8 s to test 1 host	16 s to test 1 host
Bytes transferred (best case)	0.02 MB to test 1 host	7 MB to test 1 host
Bytes transferred (worst case)	0.07 MB to test 1 host	30 MB to test 1 host
False positive response (worst case)	0.013 to test all hosts	0.12 to test 1 host

Host evaluated															
Host evaluated by	**host 1**	**host 2**	**host 3**	**host 4**	**host 5**	**host 6**	**host 7**	**host 8**	**host 9**	**host 10**	**host 11**	**host 12**	**host 13**	**host 14**	**host 15**
host 1	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
host 2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
host 5	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
host 6	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
host 7	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
host 8	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 9	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 10	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
host 12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
host 13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
host 14	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
host 15	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

Table 4.8: 15 host reputation matrix calculated at t = 0 for 3 hosts/zone configuration

Host evaluated															
Host evaluated by	**host 1**	**host 2**	**host 3**	**host 4**	**host 5**	**host 6**	**host 7**	**host 8**	**host 9**	**host 10**	**host 11**	**host 12**	**host 13**	**host 14**	**host 15**
host 1	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
host 2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
host 5	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
host 6	0.0	0.0	0.0	0.0	0.0	0.0	**0.86**	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
host 7	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
host 8	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 9	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 10	0.0	0.0	0.0	0.0	0.0	1.0	**0.86**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
host 12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
host 13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
host 14	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
host 15	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

Table 4.9: 15 host reputation matrix calculated at t = 1 for 3 hosts/zone configuration when host 7 is showing selective packet dropping behaviour

Host evaluated															
Host evaluated by	**host 1**	**host 2**	**host 3**	**host 4**	**host 5**	**host 6**	**host 7**	**host 8**	**host 9**	**host 10**	**host 11**	**host 12**	**host 13**	**host 14**	**host 15**
host 1	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
host 2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
host 5	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
host 6	0.0	0.0	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
host 7	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
host 8	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 9	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 10	0.0	0.0	0.0	0.0	0.0	1.0	**0.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
host 11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
host 12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
host 13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
host 14	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
host 15	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

Table 4.10: 15 host reputation matrix calculated at t = 2 for 3 hosts/zone configuration when host 7 is showing total packet dropping behaviour

Chapter 5

Conclusion and Future Work

This research presented a framework to detect intrusions in Wireless Mesh Network (WMN) when first line-of-defence (e.g. firewall) failed to provide security. Hybrid Intrusion Detection System (HIDS) offered a distributed architecture monitored by a centralized HIDS host. Each host was responsible to manage the reputations of other hosts in its zone. Framework provided a concept of using zoning technique to analyze the network for possible intrusions. Dividing the network in zones is important in WMN because WMN does not have centralized routers or switch to monitor traffic. For the performance analysis of HIDS, intensive testing was done on the framework. The effect of mobility was tested by changing three sets of hosts/zone configuration and increasing number of hosts from 15 to 60. This chapter analyzes results of performance analyses in Chapter 4 and the future scope of HIDS framework. It also recommends the best way to use HIDS in a mesh networks.

5.1 Effect of mobility on HIDS

HIDS showed consistent behaviour in mobile scenarios where hosts moved with a certain speed. The effect was studied in three scenarios where static, slow and fast mobilities were considered to analyze performance. Although for fast mobility (i.e. 10 m/sec), HIDS showed stable execution time and bytes transferred, it showed higher FPR compared to static or slow mobility. Increase in FPR was not considered as high error rate, because even in the worst scenario during which fast mobility was chosen for 10 hosts/zone configuration, its value 0.109 was less than 0.12 as shown by DogoIDS [4].

WMN is normally implemented in a space where mobility is not abrupt or continuous. Hence it is unlikely to have such strong mobility conditions in a practical WMN. For static and slow mobility, HIDS showed less execution time, bytes transferred and FPR. This observation concluded that HIDS would work best in relatively less mobile mesh network, hence it is not suitable for VANET.

5.2 Effect of increasing number of hosts/zone

Distributed environment shared the responsibility of calculating reputations and forwarding them to HIDS host. In this research, a key assumption was that in a zone, every host was connected to every other hosts. Thus, for the configuration of 5 hosts/zone, each host had to calculate four reputations and for 10 hosts/zone, each host had to calculate and transfer nine reputations.

Theoretically, if a host needed to manage more reputations it should take more execution time and transferred more data across the network. Chapter 4 supported this theory where increasing number of hosts in a zone was increasing execution time and data transferred. On the other hand, FPR was not increasing drastically which showed that by putting more hosts in a zone the chances of false alarm were still negligible. The recommended configuration for HIDS framework is to use less hosts in a zone and create more zones. This will result in less execution time, data transferred and FPR. On the other hand, increasing number of hosts per zone gives security benefits. More hosts in a zone results in more reputation calculations, which improves the chances of attack detection.

5.3 Effect of increasing number of hosts

Conventionally, WMN is a small network providing connectivity to a floor, a small hospital or a company. WMN is not comparable to WLAN or VANET where the size is an important factor to deploy IDS. Still, network size was considered as an important factor for this research and HIDS was analyzed by varying sizes from 15 to 60 hosts. This parameter helped to study performance metrics of HIDS in a bigger network. Results in Chapter 3 indicated that by increasing number of hosts execution time increased. Data transferred also increased when more hosts were attached to the mesh network. In this research, up to 60 hosts were taken to study the behaviour of HIDS while configured in a 10 hosts/zone configuration. Increasing number of host

did not effect FPR which showed stability of HIDS in a larger network. If the HIDS is used in a bigger network, the number of hosts/zone should be kept low to decrease execution time and data transferred and keeping low FPR.

5.4 Future work

HIDS framework was developed to give a better intrusion detection mechanism to WMN. Performance analysis and discussion presented in this thesis show that HIDS gave practical execution time, amount of data transferred and FPR which are important factors to study the practicality of an IDS. The only disadvantage of this framework is that it gives more FPR in fast mobility. Mobility is not an issue in WMN because the relative positions of hosts do not change often but it can be improved to cater some other network types.

5.4.1 HIDS in VANETs and WSNs

Modern automotive industry is reshaping the future of VANET where infotainment systems connect to internet to provide various functionalities. Mobility in VANETS is extensive as vehicles navigate through the network rapidly. In this case, HIDS may not be suitable as it shows more FPR in fast mobility yet this issue can be solved by developing algorithms for HIDS positioning. FPR is caused by the relative motion of hosts and their inability to deliver reputation messages in time. Research in this area and developing algorithm to ensure the message deliverance can expand HIDS functionalities to secure VANET.

HIDS functionality can be extended to cater WSN networks. WSN nodes are spatially distributed wireless sensors to study the behaviour of the environment. These autonomous sensors have limited processing capacity, hence HIDS is not suitable for the processing but it can be adapted by:

- replacing the use of TCP with UDP without compromising on the reputation message delivery.
- modifying AODV protocol to include reputation management at network layer.
- including cross layer information transfer to handle route management efficiently.

5.4.2 Extending HIDS functionalities

HIDS framework presented in this thesis is limited to route manipulation and packet dropping attacks. In future, the functionality of HIDS can be extended to detect packet modification attacks. Intruder can modify the content of packets while routing it to another host. This scenario is difficult to detect in a WMN because there is no centralized switch or router to monitor traffic. Since HIDS has an overview of the network, it is easier to implement the detection of packet modification attacks on a HIDS host. HIDS framework does not recognize packet dropping attacks, if happening on a single host (outside the zone) and not collectively. Thus, algorithm changes can be made to recognize this scenario.

In [4], DogoIDS gives a probing mechanism to identify attacks in a WMN. This mechanism can be included in HIDS to confirm the presence of an intruder. Once HIDS flags a host and prohibits all routes from that host, probing mechanism can help to confirm the intruder and reduce the FPR. This is a two-way attack detection mechanism where the performance of the network is not compromised while reducing the probability of false alerts. Results showed that HIDS is not good for fast mobility, hence this extension can be helpful to implement HIDS in VANETs where extensive mobility is present and low FPR is required.

5.4.3 Extending HIDS testing scenarios

HIDS was tested for three metrics as described in the previous chapter. By implementing a real-world attack, the False Negative Response (FNR) can also be tested to find the efficiency of HIDS. FNR describes the situation where no alarm is raised when an attack takes place.

Bibliography

[1] M. Tomasek, M. Cajkovsky, and B. Mados. *Intrusion detection system based on system behavior.* International Symposium on Applied Machine Intelligence and Informatics, pp. 271-275, 2012.

[2] H. Li, M. Xu, and Y. Li. *The research of frame and key technologies for intrusion detection systems in IEEE 802.11-based wireless mesh networks.* International Conference on Complex, Intelligent and Software Intensive Systems, pp. 455-460, 2008.

[3] F. B. Abreu, A. Morais, A. Cavalli, B. Wehbi and E. M. de Oca. *An effective attack detection approach in wireless mesh networks.* International Conference on Advanced Information Networking and Application Workshop, pp. 1450-1455, 2013.

[4] R. D. do Carmo. *Active intrusion detection for wireless multi-hop networks.* PhD dissertation, Technische Universitat Darmstadt, 2014.

[5] R. D. do Carmo and M. Hollick. *Analyzing active probing for practical intrusion detection in wireless multi-hop networks.* Conference on Wireless On-demand Network Systems and Services, pp. 77-80, 2014.

[6] Z. Zhang and F. Abdesselam. *RADAR: a reputation-based scheme for detecting anomalous nodes in wireless mesh networks.* Wireless Communications and Networking Conference (WCNC), pp. 2621 - 2626, 2008.

[7] S. Misra, P. V. Krishna, and K. I. Abraham. *Adaptive link-state routing and intrusion detection in wireless mesh networks.* Institution of Information and Technology Vol. 4, pp. 374-390, 2010.

[8] Y. Li. *A reputation system for wireless mesh network using multi-path routing protocol.* International Performance Computing and Communications Conference (IPCCC), pp. 1-6, 2011.

[9] E. W. T. Ferreira, G. A. Carrijo and B. Bhargava. *Intrusion detection in wireless mesh networks using a hybrid approach.* International Conference on Distributed Computing Systems Workshops, pp. 451-454, 2009.

[10] D. Makaroff, P. Smith, N. J. P. Race and D. Hutchison. *Intrusion detection systems for community wireless mesh networks.* Mobile Ad Hoc and Sensor Systems (MASS), pp. 610-616, 2008.

[11] D. Bansal, S. Sofat and P. Kumar. *Distributed cross layer approach for detecting multilayer attacks in wireless multi-hop networks.* IEEE Symposium on Computer and Informative (ISCI), pp. 692-698, 2011.

[12] A. Raniwala and T. Chiueh. *Architecture and algorithm for an IEEE 802.11-based multi-channel wireless mesh network.* Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies. , pp. 2223-2234, 2005.

[13] M. Eslami and O. Karimi. *A survey of wireless mesh networks: architecture, specification and challenges.* Control and System Graduate Research Colloquium, pp. 219-222, 2014.

[14] P. Owczarek and P. Zweirzykowski. *Routing protocols in wireless mesh networks - A comparison and classification.* Poznan University of Technology, Faculty of Electronics and Telecommunications. 2012.

[15] I. Aad, J. P. Hubaux and E.W. Knightly. *Impact of denial of service attacks on ad-hoc networks.* IEEE/ACM Transactions on Networking, pp. 791-801, 2008.

[16] P. Garcia, J. Diaz, G. Macia and E. Vazquez. *Anomaly-based network intrusion detection: techniques, systems and challenges.* Computer and Security, pp. 18-28, 2009.

[17] J. Cordasco and S. Wetzel. *An attacker model for MANET routing security.* 2nd ACM Conference on Wireless Network Security (WiSec'09), pp. 87-91, 2009.

[18] L. Zhang and G. Ding. *Byzantine attack and defense in cognitive radio networks: A survey*. IEEE Communication Survey and Tutorials, vol. 17, pp. 1340-1344, 2015.

[19] K. Tan, D. Wu, A. Chan and P. Mohapatra. *Comparing simulation tools and experimental testbeds for wireless mesh networks*. IEEE World of Wireless Mobile and Multimedia network (WoWMoM), pp. 1-9, 2010.

Glossary

WMN	Wireless Mesh Network
DMZ	Demilitarized Zone
IDS	Intrusion Detection System
MANET	Mobile Ad-hoc Network
WSN	Wireless Sensor Network
WLAN	Wireless Local Area Network
BATMAN	Better Approach To Mobile Ad-hoc Networking
MMT	Montimage Monitoring Tool
NP-AIDS	Active Probing based Network Intrusion Detection System
VANET	Vehicular Ad-hoc Network
NED	Network Description
DSL	Digital Subscriber Line
MAC	Media Access Control
AODV	Ad-hoc On-demand Distance Vector
DSR	Dynamic Source Routing
RREQ	Route Request
RREP	Route Reply
RERR	Route Error
DoS	Denial of Service
TCP	Transmission Control Protocol
HIDS	Hybrid Intrusion Detection System
LA	Location Array
ZA	Zone Array
FPR	False Positive Response
UDP	User Datagram Protocol