DOCTORAT EN CO-ACCREDITATION
TELECOM SUDPARIS ET L'UNIVERSITE EVRY VAL D'ESSONNE

Spécialité : Informatique

Ecole doctorale : Sciences et Ingénierie

Présentée par
Anderson Morais

Pour obtenir le grade de
DOCTEUR DE TELECOM SUDPARIS

Distributed and Cooperative Intrusion Detection in Wireless Mesh Networks

Soutenue le 28 Novembre 2012

devant le jury composé de :

Directeur de thèse :	**Prof. Ana Cavalli**	**-**	**Télécom SudParis**
Rapporteurs :	**Prof. Patrick Sénac**	**-**	**ISAE et LAAS-CNRS**
	Prof. Marcelo Dias de Amorim	**-**	**LIP6-UPMC**
Examinateurs :	**Jean-Luc Richier**	**-**	**LIG**
	Michel Bourdellès	**-**	**THALES**
	Prof. Stéphane Maag	**-**	**Télécom SudParis**
	Prof. Nina Yevtushenko	**-**	**Tomsk State University**

Thèse n° 2012TELE0046

Acknowledgements

In the first place, I would like to thank my advisor, Prof. Ana Cavalli, for accepting me in her research group and providing me inestimable guidance throughout my doctoral study. I especially appreciated her precious advices and inputs during my research work.

Special thanks to Prof. Patrick Sénac and Prof. Marcelo Dias de Amorim for carefully reviewing my thesis manuscript and providing valuable comments. Also thanks to Jean-Luc Richier, Michel Bourdellès, Prof. Stéphane Maag, and Prof. Nina Yevtushenko for taking the time to check my thesis.

I also thank my friends in LOR department for their cherished friendship and the good time we passed in these three years: Anis Laouiti, Iksoon, Pramila, Pablo, Felipe, Mazen, Fayçal, Alessandra, Xiaoping, Khalifa, and Brigitte Laurent. With special thanks to friends Dario and Elisangela who helped me during my arrival in France.

Here I would like to express my sincerest thanks and deepest appreciation to my wife, Klicia, for her encouragement and support during this period. Without her company and assistance, I could not accomplish this mission.

Thanks are also extended to Prof. Eliane Martins who put me in contact with Prof. Ana Cavalli and encouraged me to start the PhD thesis.

Finally, I would like to express my profound gratitude to my family for their constant and generous support always believing in me. And I would like to thank Lord Jesus Christ for all spiritual strength, comfort in darkest hours, and constant presence.

Abstract

Wireless Mesh Network (WMN) is an emerging technology that is gaining importance among traditional wireless communication systems. WMNs are becoming popular as a way of providing Internet access in a cost-effective, easy, and fast manner. However, WMNs are particularly vulnerable to external and insider attacks due to their inherent attributes such as open communication medium and decentralized architecture. A compromised mesh device, for instance, can launch attacks by fabricating malicious routing information to disrupt the network routing operation, then putting the entire mesh network at risk. In WMNs, the mesh nodes cooperate and forward packets from each other in order to enlarge their communication range and reach nodes outside their transmission range. Nonetheless, internal malicious nodes may refuse to cooperate by intentionally dropping packets in order to violate the integrity of the network communication system.

In this research, we propose a complete distributed and cooperative intrusion detection system for efficient and effective detection of WMN attacks in real-time. Our intrusion detection mechanism is based on reliable exchange of network events and active cooperation between the participating nodes. In our distributed approach, Intrusion Detection Systems (IDSs) are independently placed at each mesh node to passively monitor the node routing behavior and concurrently monitor the neighborhood behavior. This is suitable since we do not have a centralized point of traffic concentration in the mesh network where a central IDS can be deployed to aggregate all of the traffic for analysis.

Based on that, we first implement a Routing Protocol Analyzer (RPA) that accuracy generates Routing Events from the observed traffic, which are then processed by the own node and exchanged between neighboring nodes. Second, we propose a practical Distributed Intrusion Detection Engine (DIDE) component, which periodically calculates accurate Misbehaving Metrics by making use of the generated Routing Events and pre-defined Routing Constraints that are extracted from the protocol behavior. The DIDE component is able to efficiently signalize malicious activities in real-time on the own node communication link or in the neighborhood with low computational overhead. Third, we propose a Cooperative Consensus Mechanism (CCM), which is triggered among the neighboring nodes if any malicious behavior is detected. The CCM module analyzes the

Misbehaving Metrics and shares Intrusion Detection Results among the neighbors to track down the source of intrusion, i.e., the malicious node, and decide on its culpability. In, addition, a threshold-based approach is developed to improve the analysis of Misbehaving Metrics so we can achieve a higher detection rate and lower false positive rate.

To validate our research, we implemented the distributed intrusion detection solution using a virtualized mesh network platform composed of virtual machines (VMs) interconnected, which represent the network topology. We also implemented several routing attacks that are representative of common WMN security threats to evaluate the performance of the intrusion detection mechanisms. Our experiments show very encouraging results on detecting different types of routing attacks and properly identifying the attacker nodes while incurring small computation cost.

Résumé

Les réseaux maillés sans fil (WMNs - Wireless Mesh Networks) sont une technologie émergente qui prend de l'importance parmi les traditionnels systèmes de communication sans fil. WMNs sont de plus en plus populaires comme un moyen de fournir l'accès à l'Internet d'une manière rentable, facile et rapide. Toutefois, WMNs sont particulièrement vulnérables à des attaques externes et internes en raison de leurs attributs inhérents tels que le moyen de communication ouverte et l'architecture décentralisée. Un appareil sans fil mesh compromis, par exemple, peut lancer des attaques en fabriquant des informations de routage malveillants pour perturber les opérations de routage du réseau, puis mettre l'ensemble du réseau mesh en danger. Dans WMNs, les nœuds mesh coopèrent et transmettent les paquets les uns des autres afin d'élargir leur gamme de communication et d'atteindre autres nœuds en dehors de leur sphère de transmission. Néanmoins, des nœuds malveillants internes peuvent refuser de coopérer en supprimant des paquets intentionnellement afin de violer l'intégrité du système de communication réseau.

Dans cette recherche, nous proposons un système complet de détection d'intrusion distribué et coopératif qui détecte efficacement et effectivement des attaques au WMN en temps réel. Notre mécanisme de détection d'intrusion est basé sur l'échange fiable des événements du réseau et la coopération active entre les nœuds participants. Dans notre approche distribuée, systèmes de détection d'intrusion (IDS - Intrusion Detection System,) sont indépendamment installé dans chaque nœud mesh pour surveiller passivement le comportement de routage du nœud et en même temps surveiller le comportement de son voisinage. Ceci est approprié puisque nous ne disposons pas d'un seul point de concentration du trafic dans le réseau mesh dans lequel un IDS centralisé peut être déployé pour analyser le trafic du réseau.

Sur cette base, nous avons d'abord développé un Analyseur de Protocole de Routage (APR) qui génère avec précision des événements de routage à partir du trafic observée, qui sont ensuite traités par le propre nœud et échangés entre les nœuds voisins. Deuxièmement, nous proposons un Mécanisme de Détection d'Intrusion Distribué (MDID) pratique, qui calcule périodiquement des Métriques de mal comportement précises en faisant usage des événements de routage générés et des Contraintes de Routage prédéfinies qui sont extraites

à partir du comportement du protocole. Le module de MDID est capable de signaler efficacement les activités malveillantes en temps réel sur le lien de communication du propre nœud ou dans le voisinage avec un coût de calcul bas. Troisièmement, nous proposons un Mécanisme de Consensus Coopérative, qui est déclenché parmi les nœuds voisins si tout comportement malveillant est détecté. Le Mécanisme de Consensus Coopérative analyse les Métriques de mal comportement et partage les Résultats de Détection d'Intrusion parmi les voisins pour traquer la source de l'intrusion, c'est à dire, le nœud malveillant, et décider de sa culpabilité. En, outre, une approche basée sur seuil a été développé pour améliorer l'analyse des Métriques de mal comportement afin que nous puissions atteindre un taux de détection plus élevé et un taux de faux positifs inférieur.

Pour valider notre recherche, nous avons mis en œuvre la solution de détection d'intrusion distribuée en utilisant une plate-forme de réseau mesh virtualisée composé de machines virtuelles (VM - Virtual Machines) interconnectés, qui représentent la topologie du réseau. Nous avons également implémenté plusieurs attaques de routage, qui sont représentatifs des menaces communes à la sécurité du WMN, pour évaluer la performance des mécanismes de détection d'intrusion. Nos expériences montrent des résultats très encourageants sur la détection des différents types d'attaques de routage et l'identification des nœuds attaquant alors que les coûts de calcul sont bas pour le nœud.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless Mesh Network (WMN) has emerged as a new technology to cope with the challenges of next-generation wireless networks, such as providing flexible, adaptive, and reconfigurable network architecture while offering inexpensive solutions to wireless Internet service providers (ISPs). ISPs are choosing WMNs to provide Internet connectivity to costumers as it allows a fast, easy, and cost-efficient network deployment.

WMNs are typically self-organized, self-configured, and self-healed networks [1] whereas the mesh nodes automatically establish a multi-hop ad hoc network and maintain the connectivity among the participating nodes. The mesh nodes transmit traffic from others nodes to reach a destination which they could not reach by themselves. This multi-hop routing strategy widens the wireless service area and enables the network self-managing and self-healing properties, e.g., if a mesh node can no longer operate or has a temporary loss of connection, its neighbors simply find an alternative route through one or more intermediate nodes and the network continues operating. Unlike traditional wireless networks, WMNs do not rely on dedicated wireless infrastructure, but the nodes count on each other to maintain the network fully connected and transmit traffic to destination. As a result, the network is very reliable and has a good wireless coverage since there is often more than one path between a source node and a destination node in the network.

WMN can be considered as a particular type of wireless ad hoc network. As Mobile ad hoc networks (MANETs), WMNs also offer improved services and lower infrastructure costs than conventional wireless networks since both architectures use multi-hop routing strategy. Therefore, MANETs and WMNs are similar, but MANETs also have to deal with the problems introduced by the constant mobility of nodes and frequent changes of network topology. There are basically two types of mesh nodes: mesh routers and mesh clients [2], as shown in Figure 1.1. The core nodes are the mesh routers, which constitute a wireless mesh backbone among them and offer multi-hop wireless Internet connectivity to the mesh

clients. As the mesh routers provide rich radio mesh connectivity, the up-front deployment cost and subsequent maintenance costs are significantly reduced. Mesh routers can be assigned as gateway routers, which are then connected to the high-speed backhaul network, i.e., to the Internet, in order to provide Internet connection to all nodes in the network. The mesh routers have limited or no mobility and perform the role of routing traffic received from stationary or mobile clients, who normally have no routing functionality, to the gateway router that is connected to the Internet. Mesh clients are often mobile wireless devices such as cell phones and laptops.

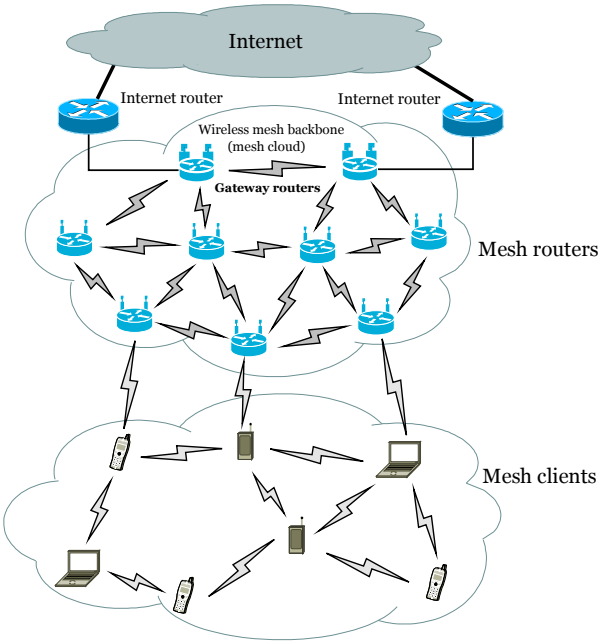


Figure 1.1: Wireless mesh network architecture.

WMNs generally use IEEE 802.11 technology to communicate between wireless mesh routers. Since 2004, IEEE 802.11 Task Group ‘S’ [6] has been developing an amendment to IEEE 802.11 standard to address the issue of multi-hop communication in WMNs defining how wireless mesh devices can interconnect to create a WLAN mesh network without having an Access Point (AP) between them. Wireless community networks and municipal wireless networks are good examples of real-life WMNs offering multi-hop paths between the wireless clients and gateway nodes. This type of network architecture provides low-cost Internet access via Wi-Fi technology to whole neighborhoods and a

municipal area by using inexpensive IEEE 802.11 wireless routers to deploy the WMN.

1.1 Motivation

Unlike wired networks, which possess sophisticated security defenses for gateways and routers to circumvent threats such as firewalls, antivirus, and complex cryptographic systems, in a wireless network environment, the radio nodes are continuously threatened by security intrusions. A WMN is highly vulnerable to several types of attacks due to its inherit characteristics such as shared wireless communication channel, lack of physical protection of nodes, absence of a centralized administration authority, high dependence of cooperation of all nodes, and devices with limited resources. The lack of fixed infrastructure and a reliable medium access mechanism make the mesh nodes potentially vulnerable to external attacks, such as passive eavesdropping and jamming attacks.

In WMNs, the nodes need to trust each other since a node depends on intermediate nodes to reach the other nodes in the network and eventually the Internet. Consequently, the routing mechanism of WMN assumes that each participating node provides accurate routing information and cooperatively forwards packets for its neighbors as specified by the routing protocol with no malicious intention. By taking advantage of this assumed trust, a malicious node can in practice adulterate routing paths of the network easily by disseminating incorrect routing information through the mesh network.

Insider attacks against the routing protocol, such as node impersonation and routing misbehavior attacks, are particularly a serious issue for the mesh network because they not only compromise the integrity of paths but also the availability of the entire mesh network. Attackers performing flooding and resource depletion attacks generate malicious packets, which uses valuable network and hardware resources and puts the mesh devices at risk. As a consequence, WMNs are exposed to a broad variety of active attacks, including routing loop attack, selfishness, black hole attack, wormhole attack, routing table poisoning, route manipulation or link spoofing attack, and Denial of Service (DoS) attacks.

Many research efforts have been conducted to conceive security methods for the existing routing protocols of WMNs. In this case, many secure routing protocols have been designed such as ARAN [3], Ariadne [4], and SAODV [5]. Most of these approaches rely on key management, public key infrastructure (PKI), and encryption technologies to protect

the integrity and authenticity of routing messages as well as prevent unauthorized nodes from joining the network. Outsider attacks can be avoided by applying cryptographic techniques, including authentication and encryption mechanisms, in which the transmitted messages are protected from modifications. Nevertheless, a compromised device owning a legitimate key, or even a non-authenticated mobile node that have access to the network, can generate malicious traffic, intercept and modify packets, and refuse to forward traffic by discarding packets received from neighboring nodes. Furthermore, cryptographic schemes usually require high computational overhead, which consume a significant portion of hardware resources, such as CPU and memory, and can be exploited by DoS attacks.

Malicious nodes disrupt the integrity of routes and compromise the whole network operation when they propagate fake routing messages to manipulate the network routes, or behave in a selfish manner by dropping legitimate routing and/or data packets that are expected to be relayed by the node. Selfish nodes can choose to drop packets in a selective or random fashion, or drop all the packets without forwarding any traffic. Hence, message fabrication and packet dropping attacking techniques represent a serious threat to the normal routing functioning and data transmission of WMNs.

Satisfactory security solutions and attack detection mechanisms in this emerging wireless network environment have not been sufficiently studied. Therefore, a distributed and cooperative Intrusion Detection System (IDS) is essential for wireless ad-hoc networks to defend against malicious behavior, to secure the routing of WMN, and to cope with the limitations of cryptographic systems. IDSs, which have been successfully used in wired networks to detect attacks, can provide an appropriate second line of defense to identify malicious traffic and misbehaving nodes in hostile wireless environments.

Intrusion detection task comprises data collection by monitoring the various system activities such as system logs or captured packets from the network, and subsequent analysis of the data to identify possible security breaches such as intrusions and misuses. However, designing IDS for WMNs is a difficult task because of the intrinsic properties and physical limitations of WMNs. First, since all nodes are expected to be well-behaved and highly cooperative, malicious nodes can exploit this assumption to launch different attacks targeting any node in the mesh network. Then, the IDS design must cover the largest possible number of attacks that can be executed by misbehaving insider nodes.

Second, detecting insider attacks in a mesh network requires intense collaboration between the neighboring nodes and a broader view of the network activity in order to track down the source of intrusion, i.e., each node needs sufficient monitoring information from the neighbors to detect intrusions. A distributed intrusion detection approach is more convenient than exclusively relying on local observation of individual nodes since the node can only see a portion of the network traffic: the packets it sends or receives. Therefore, local data auditing is intrinsically inaccurate, and cannot detect some type of routing attacks, for instance, byzantine attacks, due to the absence of a centralized point of traffic concentration in the ad-hoc network topology.

Moreover, the communication links between nodes are more unstable in a wireless environment than a wired network because of interferences and congested channels, then causing loss of messages between nodes. Therefore, the intrusion detection mechanism must take into account the link quality in terms of packet loss rate in order to have a minimum amount of traffic information to analyze, and maintain a precise intrusion detection accuracy with reduced false positive and false negative rates.

In addition, mesh devices, such as netbooks and mobile phones, commonly have constrained hardware resources, such as memory, CPU, and battery power, and restricted bandwidth usage. Then, this diversity of mesh nodes with scarce resources may affect the effectiveness and efficiency of the IDS they support. For instance, a node can silently drop packets to conserve resources then encumbering the IDS task of distinguishing overloaded nodes from selfish nodes. Thus, the IDS should be efficient in the matter of producing minimum communication overhead and low computational overhead during the intrusion detection process. Therefore, intrusion detection for WMNs remains a complex and challenging issue for security researchers.

1.2 Contributions

The main contribution of this work is the implementation of a complete distributed and cooperative intrusion detection solution for WMNs. We focus on efficiently detecting insider intrusions in real-time by collaboratively monitoring the network activity of the mesh nodes and by exchanging suspected traffic information. We also study routing attacks against the nodes of the WMN, which reveal security breaches of the routing protocol. In

particular, the key contributions of this dissertation include as follow.

In a first part, we present routing manipulation attacks against Better Approach To Mobile Ad hoc Network (BATMAN) [7], a proactive routing protocol especially designed for WMNs but also applicable to MANETs. We provide analysis of routing manipulation attacks and demonstrate the attack feasibility in detail. We show how a unique compromised node is capable of manipulating the routes of the rest of the mesh nodes and redirect the traffic to the compromised node. In order to validate our analysis, we implemented the routing manipulate attack in an emulated mesh network environment composed of virtual machines (VMs), which represent the mesh nodes executing BATMAN protocol. We perform routing attacks by using a network traffic generator, which emulates the malicious node behavior that fabricates routing packets to mislead the target nodes. Then, we perform extensive experiments on emulating routing manipulation attacks to check BATMAN security mechanism. We also present an efficient mechanism to detect routing manipulation attacks and identify the source of attack, i.e., the malicious node, which makes use of the BATMAN traces produced by each mesh node.

Therefore, we provide insights into how to emulate routing attacks against the real routing protocol implementation, in our case we utilize BATMAN protocol, in a controlled virtualized mesh network environment since most of previous ad-hoc network security approaches, including intrusion detection solutions for WMNs, have been validated using simulation-based studies and network simulators. Our analysis and experimental results of routing attack emulation expose the security vulnerabilities of BATMAN protocol.

In the second part of the thesis, we propose a distributed and cooperative intrusion detection approach for WMNs. In our approach, an IDS tool runs individually on each node to passively monitor the network traffic and eventually detect routing attacks in collaboration with the neighboring nodes. We implement specific traffic filters for the IDS tool in order to select the appropriate routing packets being transmitted on the node network interface. Then, each node IDS tool concurrently analyses the collected routing traffic and generates routing protocol events, which are processed by the own node and subsequently exchanged with direct neighbors to minimize the message overhead when monitoring the neighborhood behavior. For that purpose, we developed a Routing Protocol Analyzer (RPA) module which examines each routing packet and produces respective Routing

Events according to the protocol behavior. In this work, we make use of BATMAN routing protocol as case study, and therefore BATMAN specification is used to define the Routing Events. However, the RPA module is quite customizable and flexible, and can easily accommodate other routing protocols as well.

After, we implement the core part of the distributed intrusion detection architecture, the Distributed Intrusion Detection Engine (DIDE) component. This module processes the Routing Events produced by the RPA module and also the Routing Events received from neighboring nodes to cooperatively detect routing misbehavior intrusions. We define Routing Constraints which are extracted from the routing protocol behavior and are integrated into DIDE component. Again, we use BATMAN specification to formulate the Routing Constraints, nevertheless additional routing protocols are similarly supported. We also implement part of the routing attack detection algorithms in the DIDE. Based upon the Routing Constraints, the DIDE component periodically calculates Misbehaving Metrics. The Misbehaving Metrics indicate if possible malicious routing behavior is occurring on the node network interface or in the neighborhood, i.e., if the node or a neighbor is originating malicious routing traffic, for instance, the malicious node is transmitting fake routing packets to its neighbors. Our collaborative intrusion detection procedure, which is carried out by the DIDE component, is executed in real-time by the IDS tool having low computational overhead and reasonable message overhead, and it is quite efficient with regard to hardware resources consumption.

We propose a Cooperative Consensus Mechanism (CCM) module that is triggered by the node IDS if any evidence of malicious routing information is found by the IDS agent. The CCM module inspects the Misbehaving Metrics provided by the DIDE component, such as rate of inconsistent routing packets received by the node, and confirms if abnormal routing behavior detected by DIDE is a routing attack or not based on thresholds. Then, the node IDS warns the neighboring nodes about the detected routing intrusion. Before making the final decision, the node IDS consults its neighbors. Thus, the CCM module shares Intrusion Detection Results with neighboring CCM modules to confirm that its neighbors diagnosed the same routing attack to finally make a collective decision about the suspicious node. If the majority of nodes detected the intrusion, a consensus is reached among the neighboring nodes on the suspected node. The proposed Consensus Algorithm coordinated

among neighboring nodes is able to efficiently detect malicious routing traffic in a neighborhood and the source of the intrusion with good precision in real-time.

Furthermore, we propose a threshold-based scheme that is used along with CCM module to enhance the accuracy of analysis of Misbehaving Metrics. The threshold mechanism takes into consideration the rate of packet loss between communication links of nodes to differentiate attack detection rate from false alarm rate in order to achieve a better true positive rate and low false positives. Our threshold-based mechanism results in smaller false positives and also provides better efficiency in terms of intrusion detection rate.

We demonstrate the efficacy of our distributed intrusion detection approach by applying a virtualized mesh network environment that consists of VMs interconnected. The VMs connected with each other are equivalent to mesh nodes executing our distributed intrusion detection solution in a mesh topology. In this platform, we represent the adversary nodes by using VMs which execute attack injector tools. We employ these attacking tools to implement the most common routing misbehavior attacking techniques such as routing message fabrication and packet dropping attack variants. In that manner, we satisfactorily evaluate the performance of our distributed intrusion detection algorithms. Our experimental results show that our distributed intrusion detection method achieves high detection rate for different types of message fabrication and packet dropping attacks with no false negatives and low false positive rate.

We show the our IDS implementation presents good effectiveness and high level of reliability in the role of detecting different routing attacks, and yet incurs small computation overhead, i.e., CPU and memory overhead, and acceptable communication overhead for the mesh devices. Then, our intrusion detection solution achieves scalability having good performance in terms resources consumption with practical assumptions.

1.3 Results

The constant pursuit of publishing the obtained results conducted the development of this work. Therefore, as soon as we obtained preliminary results on the intrusion detection approach we published the first article in a relevant conference. Thus, the listing below shows in ascending order by date of publication the articles which have been resulted from this thesis work and provides a quick description about each one.

- A. Morais, and A. Cavalli, *Route Manipulation Attack in Wireless Mesh Networks*, In Proceedings of the 25th IEEE International Conference on Advanced Information Networking and Applications (AINA 2011), pp. 501-508, March 2011, Biopolis, Singapore.

  This paper presents detailed security analysis of routing manipulation attacks against Better Approach To Mobile Ad hoc Network (BATMAN) routing protocol. In this paper, we demonstrate the attack practicability through a virtualized network environment where we perform several attacks by using an attack injector tool.

- A. Morais, and A. Cavalli, *Detection of Attacks in Wireless Mesh Networks*, In Proceedings of 5th Latin-American Symposium on Dependable Computing (LADC 2011), pp. 45-54, April 2011, Sao Jose dos Campos, Brazil.

  This paper analyzes routing attacks against the proactive routing protocol Better Approach To Mobile Ad hoc Network (BATMAN). We demonstrate the impact of these attacks by using an emulated mesh network platform that consists of virtual nodes executing BATMAN protocol. Then, we show how to detect this type of attack by making use of BATMAN traces produced by the mesh nodes.

- A. Morais, and A. Cavalli, *A Distributed Intrusion Detection Scheme for Wireless Ad Hoc Networks*, In Proceedings of 27th Annual ACM Symposium on Applied Computing (SAC'12), pp. 556-562, March 25-29, 2012, Riva del Garda, Italy.

  This paper introduces the distributed intrusion detection approach which is based upon exchange of events and cooperation between the mesh nodes. The proposed mechanism relies on non-intrusive traffic monitoring at each node, which generates network events. These events are processed by the intrusion detection system that is able to signalize malicious activities in real-time. We demonstrate the approach efficiency for detecting misbehaving nodes through the virtualized network environment that consists of virtual nodes interconnected.

- A. Morais, and A. Cavalli, *An Event-Based Packet Dropping Detection Scheme for Wireless Mesh Networks*, In Proceedings of 4th International Symposium on Cyberspace Safety and Security (CSS 2012), pp. 309-323, December 12-13, 2012, Melbourne, Australia.

  In this paper, a distributed and cooperative approach for detection of packet dropping attacks is provided. An intrusion detector placed at each node monitors the node behavior and shares routing events and detection results with neighboring nodes. Based on these events, each node is capable of detecting malicious dropping behavior on the own node or in the neighborhood. A virtualized mesh network environment is used to implement the distributed packet drooping detection approach and to execute packet-dropping attacks.

1.4 Organization of the Dissertation

The rest of the dissertation is organized as follows. In Chapter 2, we discuss related concepts of intrusion detection in wireless ad-hoc networks and WMNs, and compare our work with other related work. Chapter 3 presents the routing manipulation attack against BATMAN routing protocol and provides a trace-based method to detect this type of routing attack and malicious nodes in the WMN. Chapter 4 describes the proposed distributed and cooperative intrusion detection architecture and its main modules for collaborative detection of routing attacks and the corresponding adversarial nodes. In this chapter, we focus on detecting message fabrication attacks variations. Chapter 5 applies the distributed intrusion detection architecture for detection of packet dropping attack variants. In Chapters 3, 4 and 5, we demonstrate the effectiveness of our attack detection and distributed intrusion detection mechanism in a virtualized mesh network environment through extensive experimental evaluation. Finally, Chapter 6 concludes this work and points toward some future research directions.

Chapter 2

Background and Related Work

Many studies have been focused on providing security schemes to WMNs. Cryptography tools such as authentication services and access control mechanisms can improve the level of security of WMNs. However, these attack prevention approaches alone cannot avoid all possible attacks, such as internal attackers. Therefore, it is indispensable to have additional security schemes to deal with misbehaving mesh nodes that have a valid key and access rights. In this case, various intrusion detection approaches for WMNs and wireless ad-hoc networks have come to existence. In this chapter, we discuss background and related work to our research.

2.1 Intrusion Detection

Intrusion refers to an incident of unauthorized access to data or a computing service. Intrusion Detection System (IDS) is a device or software application that collects data by monitoring the different system activities, and subsequently analyzes the gathered data to detect possible unauthorized activities, such as malicious activities or security policy violations. The collected data can be audit logs created by the operating system or packets captured from the network interface. When an intrusion is detected, a passive or active response is triggered by the IDS according to the response policy.

IDSs are primarily focused on identifying potential security threats, logging information related to the observed events, producing security reports, and notifying security administrators of suspected events so that the attack can be contained to minimize the system damage and the attacker can be properly identified. The accuracy of the IDS is usually estimated in terms of false positives, i.e., false alarms triggered by the IDS when no attack has happened, and false negatives, i.e., attacks not detected by the IDS. Then, IDSs seek to reduce both false positive rate and false negative rate to improve the IDS performance and its efficiency. IDSs can be classified into two categories according to the

monitored events, i.e., the collected audit data.

- Host-based IDS (HIDS): HIDS monitors and analyzes internal events of the computing system such as system calls, application logs, and file system activities to identify intrusions. Basically, HIDSs checks the dynamic behavior and the state of the computer system, which is stored in RAM, in the file system, or in log files.

- Network-based (NIDS): NIDS monitors the network traffic. This can be achieved by putting the network interface into promiscuous mode, so all the packets are captured and are subsequently examined to detect malicious activity.

An example of HIDS is OSSEC [8]. OSSEC is an open-source HIDS that performs log analysis, checking of file integrity, policy monitoring, rootkit detection, time-based alerting, and active response. OSSEC is composed of a central manager that stores file integrity checking databases, logs, events, and system auditing entries, and it receives information from agents, syslog, databases, and from agentless devices. Agents are programs installed on systems in which they collect information and forward it to the central manager for analysis and correlation. Agentless perform file integrity monitoring on systems without an agent installed, for instance, firewalls, switches, and routers.

A popular NIDS is the open-source Snort [9]. Snort can operate as a packet sniffer, which reads packets traversing the network link, and display them on the console, as a packet logger to dump complete packets into a log file for later analysis, or as NIDS which monitors the network traffic and checks the packets against a set of rules defined by the user. A Snort rule set is a file of attack signatures with respective actions that tells Snort what to do in case packets match the rule criteria. It works basically as an attack pattern matching technique. However, Snort does not provide communication mechanisms which allows Snort instances deployed in different nodes to share attack detection information. As a result, Snort does not support distributed intrusion detection and collaboration between the participating nodes to identify malicious nodes in a consensual manner.

Bro [10] is a representative NIDS that passively watches traffic on the network link for detecting network intruders. Bro is divided into an event engine that transforms the filtered

network traffic stream into a series of higher-level events, and a policy script interpreter that runs these events by performing event handlers, which are written in a specialized Bro language. Event handlers are used to express security policies, and are capable of updating the global state information, synthesizing new events, scheduling events, recording information to disk files, and generating alerts.

Since Bro provides a flexible and extensible monitoring architecture, we take advantage of Bro structure to implement our distributed intrusion detection architecture for WMNs, as explained in Section 4.2. Then, we apply Bro monitoring engine in the proposed intrusion detection solution to capture and filter network traffic. In addition, we employ Bro enriched language to implement a set of attack detection algorithms for identifying malicious routing behavior. Moreover, we make use of Bro communication mechanism to maintain intrusion detectors connected and implement the scheme of exchange of network events and intrusion detection results between nodes, so we can monitor local vicinities independently and perform collaborative diagnosis of routing attacks in the neighborhood. We provide more details on our distributed intrusion detection architecture in Section 4.2.

More recently, Suricata [11], an open-source NIDS developed by Open Information Security Foundation (OISF) was released. The novelty of Suricata is it employs a multi-threaded architecture allowing the detection engine to take advantage of multiple core and multiple processor host architecture to increase the speed and efficiency of traffic analysis using hardware acceleration. As Snort, Suricata is a rule-based IDS then it matches a set of pre-defined rules to the captured traffic to recognize attacks. Nonetheless, Suricata presents the same architectural limitations as Snort by not providing a cooperative scheme between nodes to share intrusion data for mutually detection of attacks.

IDSs are also classified depending on the detection engine, i.e., the detection technique used to identify malicious behavior, into three main categories as follows.

- Signature-based or misuse IDS: The IDS compares the collected traffic with a pre-defined set of attack patterns, known as signatures/rules, to identify intrusions. If an intrusion pattern matches, then that attack is detected. The advantage of signature-based detection is its accuracy. For that reason, signature-based IDSs are widely preferred for commercial use. One limitation of this technique is it can only detect

known attacks, not being able to detect new attacks or even variants of known attacks, then presenting many false negatives. Therefore, IDS signatures must be constantly updated to not leave the system vulnerable to new attacks, which can take a significant time.

- Anomaly-based IDS [12]: The IDS uses profiles of normal behavior of the system, usually defined statistically through automated training, to match against the actual system activity, and it signalizes any significant deviation as anomalous behavior, but not necessarily malicious. The underlying assumption is that malicious activities will considerably deviate from normal behavior. The advantage of anomaly-based IDSs is they are able to detect unknown attacks without prior experience. The major drawback is the high false positive rate since normal behavior has large deviations and it changes over time. Thus, false alarms must be investigated by the system administrator, which is a time consuming job.

- Specification-based IDS [13]: The IDS relies on a set of constraints or rules that specify the correct behavior of a program/protocol, and monitor the actual behavior of the program/protocol in execution regarding these constraints. The intrusions, which generally make the program/protocol behave in an incorrect manner, will be identified without exact knowledge about the nature of the intrusion. This technique combines the strengths of anomaly-based detection and signature-based detection techniques, and it is able to detect known and unknown attacks with low false positives. However, this approach cannot detect some type of attacks, such as DoS attacks because they do not directly violate the program/protocol specification.

Snort and Suricata NIDS are good examples of signature-based IDS, which rely exclusively on rule-based detection to inspect the network traffic. Bro policy script engine also incorporates a signature matching facility that searches for specific traffic content as Snort pattern matching. However, for Bro, these signatures are expressed as regular expressions rather than fixed strings.

2.2 Intrusion Detection in Wireless Mesh Networks

The current IDS tools for wired networks cannot be directly applied to WMNs. Building IDS for inspecting the mesh network activity is not an easy task because of the intrinsic characteristics and physical restrictions of WMNs. However, an IDS for WMNs is essential to secure the network routing functionality and to cope with the limitations of cryptographic systems, then providing a second line of defense. Consequently, researchers have been working hard to develop new IDSs for WMNs or adapt the existing IDSs to suit the characteristics of WMNs. Most of the research in intrusion detection pertains to MANETs because WMN is a relatively recent development. Nonetheless, in principle, all of the intrusion detection approaches of MANETs can be applicable to WMNs.

There have been few signature-based detection approaches proposed for WMNs and little research on attack pattern matching techniques for this type of network. The constant need of updating the signature database is a difficult process in mesh network architectures and each node IDS signature database has to be individually updated each time. Attack databases cannot be easily updated without a central administration.

Anomaly-based detection engines for WMNs rely on particular models of node expected behavior, that are mainly based on statistical approaches, and label those nodes that deviate from the behavior model as malicious nodes. However, there is no clear separation between normal and abnormal network activities in mobile wireless scenarios. Since nodes can join or leave the network arbitrarily and communication links can be quite unreliable, e.g., with significant packet losses, then fluctuations on the transmission of routing information can be attributed to a malicious node, or a node that has a poor wireless connection, or a mobile node moving independently. As a consequence, the problem of false positives becomes worse for the IDS in this environment with different scenarios.

A number of specification-based intrusion detection approaches have been proposed to detect attacks against Ad hoc On-Demand Distance Vector (AODV) and Optimized Link State Routing Protocol (OLSR) traditional routing protocols. Tseng et al. [14] proposed the first specification-based intrusion detection approach for MANETs. The approach uses Finite State Machines (FSMs) for specifying the correct routing behavior of AODV. Network Monitors (NMs), which are assumed to cover all nodes, employ these FSMs for detecting run-time violation of AODV operations, especially during the route discovery

process. NMs analyze request-reply flows, i.e., Route Request (RREQ) and Route Reply (RREP) messages, to identify attacks on AODV. NMs can ask neighbors about previous messages or other node information. However, the approach does not take into account natural packet loss present in wireless networks and cannot distinguish between lost messages and packets dropped on purpose. Moreover, some of the assumptions used in this work are not very realistic, e.g., MAC addresses and IP addresses of all nodes are registered in the NMs and remain unchanged, and MAC addresses cannot be forged. In addition, the approach requires adding additional field in the protocol original message RREQ to keep track of the RREQ path, which certainly will cause incompatibility issues with other nodes executing the unmodified routing protocol version.

Huang et al [15] propose to model AODV protocol behavior by decomposing it into normal basic events and modeling these events by using Extended Finite State Automaton (EFSA). The EFSA can detect anomalous basic events, such as fabrication of routing messages or modification of routing messages, which are direct violations of the protocol specification. They also use statistics on the states and transitions of EFSAs for building a detection model to identify anomalous basic events that are temporal and statistical in nature, i.e., not violate directly the specification, such as flooding of routing messages. The approach combines the advantages of specification-based and statistical-based intrusion detection. However, the protocol behavior is only analyzed from the point of view of a single node with no communication between neighboring nodes for sharing collected data/detection results, which makes the intrusion detection process inaccurate due to the distributed and cooperative nature of WMNs and the constant instability of wireless links.

Orset et al. [16] provide a formal specification of correct behavior of OLSR protocol by applying Extended Finite State Machines (EFSM) in order to detect run-time violations of protocol implementation, i.e., some typical OLSR attacks and conformance anomalies. The approach is similar to EFSA-based detection approach and therefore it has the same limitations as it, i.e., the approach only monitors the correct behavior of each node separately and do not considers the point of view of neighboring nodes for cooperative detection of attacks. Moreover, the approach does not provide additional methods, for instance, statistical schemes, to circumvent the main limitation of specification-based IDS: detect attacks that do not immediately violate the protocol specification, e.g., message

flooding attacks. In addition, building detailed formal specification for complex routing protocols can be a very time consuming job.

Our IDS approach is mainly based on specification-based intrusion detection since we specify Routing Constraints, which define the correct protocol behavior, so we can detect malicious behavior that violate these constraints. The reason for using Routing Constraints is they are optimized to cover the most common routing attacking methods of WMNs: message fabrication, packet dropping, and message flooding. However, we essentially rely on routing events, that are defined based upon the protocol specification, and are generated as a result of the analysis of each routing message. By making use of routing events together with Routing Constraints, we can identify most of attacks of WMNs in a reliable and collaborative way since routing events are analyzed in parallel by the node and its neighboring nodes to compute misbehaving metrics. Furthermore, we integrate a threshold-based mechanism to distinguish between accidental packet loss of links and malicious packet dropping. Hence, our intrusion detection approach provides better detection accuracy with no false negatives while limiting the number of false positives. Moreover, the approach is able to identify any new attack that makes use of these common attacking methods without the need of updating the intrusion detection engine.

IDSs can also be classified according to the architecture employed for intrusion detection in the network environment as follows.

- Stand-alone IDS: In this architecture, the IDS is installed at each node and strictly utilizes node's local audit data for detecting network intruders without collaborating with other nodes. Nevertheless, the fact that the IDS architecture relies exclusively on local audit data to recognize malicious behaviors the IDS performance is limited in respect of detection accuracy and type of attacks that the IDS is able to detect because of the distributed nature of WMNs. This architecture is more suitable for flat network infrastructure.

- Distributed and cooperative IDS: To resolve the problem of absence of a single point of traffic concentration in WMNs, where the IDS can be deployed for traffic analysis, distributed and cooperative IDS architecture is proposed. Similarly to

stand-alone architecture, a IDS installed in each node processes node's audit data locally but they also communicate with other nodes to exchange audit data and/or detection results, examine the neighbors behavior, make agreed decisions about inconclusive intrusion detection and suspected nodes, and provide responses to attacks. This IDS architecture is more appropriate for flat networks.

- Hierarchical IDS: This architecture adopts a multilayer approach that divides the network into clusters or zones. Some nodes are selected, based on specific criteria, to perform as cluster-heads and have more responsibilities and roles in the intrusion detection, such as communication with other clusters, that are different from those of nodes that are part of the cluster, i.e., the cluster members. Cluster members usually execute lightweight IDSs that only analyze local audit data, while cluster-heads run a more complete intrusion detection engine that function as a second layer of IDS using audit data from the cluster members to carry out global detection and response. Hierarchical IDSs perform somehow cooperatively to detect intrusions. Hierarchical architecture is preferred for multi-layered networks. However, the process of creating and maintaining clusters often imposes extra processing load and communication overhead to the network nodes.

Snort, Bro, and Suricata IDSs have per default a stand-alone IDS architecture. However, Bro provides a communication system for exchanging intrusion detection information that can be perfectly applied for distributed and cooperative detection, then making Bro a distributed and cooperative IDS as well. That is the main reason why we employ Bro for implementing our distributed intrusion detection solution for WMNs.

Since nodes in WMNs have partial view of network data, a distributed and cooperative IDS architecture is preferable than stand-alone architecture in order to provide sufficient traffic data to each node's detection engine to reach the correct decision about detected intrusions and malicious nodes. Moreover, relying on local data auditing is essentially inaccurate and cannot detect some specific attacks, such as coordinated/collusion and byzantine attacks, due to the absence of a central point of traffic concentration in the WMN topology. Detection of internal attacks implies strong collaboration between neighboring

nodes and wider view of network operation in order to locate the source of attack, i.e., each node IDS requires enough traffic information from neighboring IDSs to diagnose such intrusions. Therefore, the majority of the most recent intrusion detection approaches for WMNs relies on distributed and cooperative IDS architecture.

2.3 Distributed and Cooperative Intrusion Detection

Since the network data is forward by the nodes of WMN in a cooperative manner which is analogous to the mode that Internet routers relay IP packets, intrusion detection in Internet routing environment is directly related to intrusion detection in wireless ad-hoc networks. WATCHERS [17] is one of the first intrusion detection approaches proposed for Internet routing environments, which makes use of a distributed intrusion detection scheme running concurrently and independently in each router. Subsequent intrusion detection approaches proposed for ad-hoc networks and WMNs adopt similar ideas from WATCHERS. In the approach, routers check incoming packets to identify any routing anomalies. In addition, each router keeps track of the amount of data received and transmitted to neighboring routers. A link-state routing protocol is assumed since each router is aware of other routers and the network topology. Then, routers periodically exchange their respective collected metrics by using a flooding protocol, and start the analysis phase for doing the comparisons. The purpose is detecting in a distributed manner misbehaving routers, which can be compromised by an attacker and are selectively dropping packets above a given threshold, or can be malfunctioning and are misrouting too many packets. If misbehaving routers are detected, the other routers will change their routing tables to avoid forwarding packets through these bad routers. However, computational costs can be high in this type of Internet environment since each router must keep packet metrics for all of its neighboring routers and share this information by flooding. In addition, certain conditions must be respected, for instance, the majority of routers must be good, and each good or bad router must be connected to at least one good router.

Zhang et al. [18] proposed one of the earliest distributed and cooperative intrusion detection architecture for ad-hoc networks. In the architecture, IDS agents are placed individually on each node. The IDS agent model is composed of six modules. The local data collection module gathers audit data, such as communication activities within the node

radio range. Then, this collected data is analyzed by the local detection engine module for evidence of anomalies. If an anomaly is detected with strong evidence, the IDS agent can independently determine that the network is under attack and can initiate a response action by triggering the local response module, which raise an alert, or starting the global response module. If an anomaly is detected with inconclusive evidence, IDS agents request the cooperation of neighboring IDS agents for global detection using the cooperative detection engine module, which communicates to other agents through a secure communication module to exchange relevant data. However, the authors do not present the details of the cooperative detection algorithm, i.e., which type of data is exchanged. In addition, they only evaluate the local detection engine module for detecting intrusions locally.

Marti et al. [19] proposed watchdogs and pathraters. A watchdog runs on each node to monitor the behavior of neighboring nodes by listening promiscuously the neighbors' transmissions. When a node forwards a packet, then the node's watchdog verifies that the next node in the path also forwards the packet. Since it is assumed Dynamic Source Routing (DSR) as routing protocol, the watchdog is aware of which route the packet will follow through the network. If the watchdog detects a neighboring node is dropping packets more than a certain threshold bandwidth, the node is considered as misbehaving one and a message is sent to the source node. The pathrater mechanism avoids routing packets through the detected misbehaving nodes. Each node maintains a rating for all the other nodes between 0 and 1. Then, it calculates a path metric by averaging the nodes ratings in the path. A neutral rating value of 0.5 is initially given for each node. The rating of nodes on all actively used paths is periodically incremented, but if a node is suspected of misbehaving, it will have its rating drastically decreased. But, the approach has serious limitations, e.g., since the watchdog must have knowledge of the route it is limited to source routing protocols. In addition, the watchdog cannot detect a misbehaving node in the presence of ambiguous and receiver collisions, neighbors with limited transmission power, false misbehaving report (blackmail attack) [41], collusion between nodes (coordinated attacks), and partial dropping, i.e., a node dropping packets below the threshold.

Kachirski and Guha [20] introduced a multi-sensor intrusion detection system based on mobile agent concept. The IDS is divided into three levels of mobile agents (sensors) with certain functions: monitoring, decision-making, and action. Mobile agents, which monitor

the network packets, are only executed in certain nodes that are chosen by vote within a cluster. In addition, these nodes, i.e., the cluster-heads, execute decision-making agents, which make collective decisions about network intrusions and also collect host-based intrusion information from the cluster members. Every node executes an action agent that is responsible for resolving intrusion situation on the node, e.g., locking out a node. The advantages of this structure are reduced communication cost and the security analysis computation is restricted to a few key nodes. Nonetheless, the feasibility of proposed multi-sensor IDS, via simulation or implementation, is not studied for attack detection scenarios with malicious nodes. Therefore, we cannot verify the approach efficiency.

Sterne et al. [21] proposed a general cooperative intrusion detection architecture for MANETs. The architecture is organized as a dynamic hierarchy using multiple-layering clustering. Intrusion detection observations are obtained from leaf nodes and cluster-heads, which are responsible for collecting data by promiscuous monitoring or direct participant reporting. Then, leaf nodes report the detection metrics to cluster-heads periodically. These cluster-heads can be part of a second cluster, then reporting respective detection data to their cluster-heads. The intrusion detection data is incrementally analyzed, consolidated, correlated, and summarized as it flows upward towards the nodes in the top of the hierarchy. The infrastructure also allows dissemination of intrusion response. The proposed architecture is well formulated and it claims reducing communication overhead through hierarchy. However, the practicability of the approach is not demonstrated via simulation or implementation, so it is difficult to estimate the cost of operation and maintenance of the architecture. Moreover, the approach does not differentiate between link packets loss and malicious packet dropping, it can only detect a significant number of packets dropped.

Yang et al. [22] introduced SCAN, a network layer security solution for detecting and reacting to misbehaving nodes. In SCAN, each node monitors the routing and packet forwarding behavior of its neighboring nodes by promiscuously overhearing the neighbor transmissions, and independently detects any misbehavior in its own neighborhood. A distributed consensus scheme is implemented, in which a node is convicted of being malicious only after multiple neighbors have reached that consensus. Nonetheless, SCAN has the same disadvantages of Marti's watchdogs approach since SCAN approach is limited to AODV routing protocol that is similar to source routing protocol. Detection of routing

misbehavior is only possible by adding new fields to AODV RREQ and RREP routing messages so each node can maintain part of the routing table of its neighbors. In addition, SCAN cannot detect packet forwarding misbehavior in the presence of neighbors with weak signal transmission, and is not able to distinguish packet loss due to collision from intentional packet dropping. Another limitation of SCAN is the need for multiple neighboring nodes to collaborate in order to make a decision about a suspicious node.

Razak et al. [23] proposed a cooperative two-tier IDS architecture. In the first-tier, a local-level IDS collects local audit data by overhearing network activities in promiscuous mode and analyze these data firstly using a signature-based detection engine and then an anomaly-based detection engine. If the IDS at first-tier cannot decide whether the suspicious activity is malicious, the second-tier of the architecture is triggered. The second-tier IDS of the node applies a global detection mechanism, which gathers audit data from neighboring nodes and examines these data by using signature-based and anomaly-based detection, similarly to the first-tier. The second-tier IDS maintains a list of friends to ensure that nodes exchanging audit data with the IDS are trustful. This friendship mechanism reduces the problem of false data transmission, i.e., blackmail attacks, since only trustful nodes are allowed to send audit data to the second-tier IDS. However, the exchange of audit data, the use of multiple detection engines for local and global detection, and the use of trust management increases the IDS complexity and imposes a considerable computational and communication overhead for the nodes that have constrained resources. In addition, the accuracy of the IDS in detecting common routing attacks is not evaluated.

Komninos and Douligeris [24] proposed a layered intrusion detection framework (LIDF) to detect compromised nodes in ad-hoc networks. LIDF consists of collection, detection, and alert modules, which operates locally in every node in the network. The collection module collects audit data at OSI link layer, for checking one-hop connectivity and frame transmission, and at network layer, for checking routing and data packet forwarding. The detection module processes the most relevant audit data collected from these different layers to perform anomaly-based layered intrusion detection. Once malicious activity is detected on a node, the alert module is responsible for notifying the neighboring nodes which can request additional audit data from neighbors in order to reach a more precise decision regarding the suspicious node. However, the approach does not

specify in which attack scenarios the cooperative detection scheme is triggered. In addition, the cooperative scheme requires that each node have at least three neighbors for functioning in reliable way, but it is vulnerable to blackmail attacks. The authors claim the approach is able to detect malicious packet dropping, but details of packet dropping detection mechanism are not given neither experimental evaluation is presented.

The recent approach [25] proposed by Zhang et al. introduces an anomaly-based detection system, termed RADAR, to detect and handle anomalous nodes in WMNs. Monitor agents running on each mesh node gather local traffic observations, which are fed to a reputation management system to calculate trust values for deriving the node reputation. These agents exchange their opinions periodically or as required. Managers, which can be deployed on mesh routers, run a dual anomaly detection engine, which uses observations from the agents as input, i.e., the calculated node reputation that represents the node behavior, in order to measure deviations between normal and anomalous behavior for each node. Response policies handle the anomalies reported by the two detection engines. However, the authors have not specified when the nodes decide to request the neighbors' cooperation for exchanging detection information, e.g., the trust values of node, and how this cooperation is achieved. Moreover, authors are vague about how monitor agents and managers are deployed in the network, i.e., what type of the IDS architecture is employed. Consequently, we cannot estimate the communication and computational overhead added by the intrusion detection architecture. Nevertheless, the use of two detection engines at each node increases the processing overhead.

More recently, Saxena et al. [26] propose a hierarchical architecture for detecting selfish behavior in WMNs and enforcing cooperation among mesh routers. The architecture adopts a decentralized detection scheme by dividing the mesh routers into clusters. Monitoring Agents (MAs) installed on specific mesh routers watch the behavior of neighboring mesh routers in the cluster by collecting periodic traffic reports from the mesh routers and sending them to Sink Agents (SAs) hosted at the mesh gateways. SAs process the reports and detect the presence of selfish mesh routers. Malicious packet dropping is detected by counting the packets that the mesh router was supposed to forward and the packets that it actually forwarded. A reputation mechanism is used to punish selfish mesh routers. As opposed to the preceding approaches, the IDS architecture considers the

wireless link quality to differentiate between intentional packet dropping and packet loss due to poor link quality in order to decrease the number of false positives. The authors claim the hierarchical reporting architecture reduces the communication overhead. However, the constant transmission of traffic reports from mesh routers to MAs in the cluster and from MAs to SAs incurs an extra communication overhead to these nodes. In addition, mesh routers elected as cluster-heads and mesh gateways are unfairly overloaded, which imposes a high processing cost since these nodes must frequently process traffic reports, and they are vulnerable to man in the middle attack and blackmail attacks. Moreover, cluster-heads may become critical point of failures in the IDS architecture.

2.4 Conclusion

The carried analysis reveals the most important strengths and weaknesses of current cooperative and distributed IDSs for ad-hoc networks. It is hard to judge which IDS approach is the best solution since none of them are complete and they primarily focus on addressing specific issues of wireless ad-hoc networks. A key point in the existing approaches is that they have been exclusively validated using simulation-based studies or theoretical models, then lacking experience in real WMNs. Simulation is useful to estimate the performance of intrusion detection engines, which are complex to implement as IDS tools and difficult to deploy at nodes in an ad-hoc network infrastructure. However, simulation cannot fully represent the implementation details and performance requirements of the IDS capturing network traffic, analyzing the collected data, and exchanging intrusion detection information with neighboring nodes, all that at real-time in the mesh device with constrained hardware resources. As a result, these approaches can be shown to be unpractical in real WMN scenarios due to unforeseen factors.

The majority of the approaches do not demonstrate the viability of the cooperative detection scheme where neighbors are supposed to share audit data and/or detection results in order to make collective/local decisions about malicious traffic and suspicious nodes. Monitoring the behavior of neighboring nodes by overhearing transmissions in promiscuous mode is not reliable since wireless interferences and periods of congestion are frequent between nodes in this type of environment. However, no approach, except the recent work [26], has taken into consideration the effect of the wireless link quality, more

precisely packet loss, on the performance of the intrusion detection engine, that certainly affects the number of false positives and accuracy of detection rate. Existing solutions are not well suited for WMNs as they assume the existence of reliable end-to-end communication path between every pair of nodes in network infrastructure. While this assumption is typically valid in wired networks, it generally does not hold in WMNs.

Therefore, we have decided to validate our approach on a virtualized mesh network platform composed of VMs interconnected by virtual switches that represent the wireless links. The virtual switch allows emulating packet transmission degradations, e.g., packet loss rate and loss burst. The VMs possess hardware configuration equivalent to a common mesh device and the unmodified routing protocol implementation is installed on each VM. Each VM executes our distributed intrusion detection solution as a IDS tool, which independently monitor the network interface and actively exchange traffic information and detection results with neighboring VMs for concurrent monitoring of neighborhood behavior and collaborative detection of attacks. This virtualized environment allows demonstrating how a cooperative and distributed IDS architecture can be applied to detect routing attacks and malicious nodes in a more realistic WMN scenario. To the best of our knowledge, there is no similar cooperative IDS architecture reported in the open literature.

Chapter 3

Attack Analysis

In this chapter, we introduce routing manipulation attack targeting BATMAN routing protocol. We provide comprehensive analysis of routing manipulation attack and illustrate the attack viability. We demonstrate that a malicious node is capable of adulterating the routing table of target mesh nodes in order to redirect the network traffic from these nodes to the malicious node. This is achieved by simply fabricating routing packets from the malicious node which are then diffused through the entire network.

We first perform analysis on the impact of routing manipulation attack on the node's routing behavior in detail, and show that this attack can result in severe consequences to the data routing that depends not only on the network topology but also on the location and strategy of the attacker node. Then, we implement routing manipulate attack in a virtual mesh network environment consisting of Virtual Machines (VMs) which correspond to the mesh nodes. Routing attacks are emulated by a network traffic generator in a VM, which represents the malicious node. We carry out extensive experiments on routing attack emulation in different example scenarios. We then propose a precise mechanism that unequivocally identifies routing manipulation attack and the attacker node by using nodes' traces that are based on BATMAN protocol behavior. Our systematic analysis and experimental evaluation with routing attacks reveals security vulnerabilities of BATMAN. These breaches can be exploited by an attacker to degrade the network performance.

The rest of the chapter is organized as follows. Sections 3.1 and 3.2 describe the general system model and attacker model under which we study routing attacks in WMNs. In Section 3.3, we introduce BATMAN routing protocol. In Section 3.4, we provide a description of routing manipulation attack. In Section 3.5, we present the experiments with the routing attack. In Section 3.6, we describe the proposed method to detect the routing attack and the malicious node. Finally, Section 3.7 concludes this chapter.

3.1 System Model

A WMN is comprised of a set of static wireless mesh routers and a set of wireless mesh clients. Each mesh router is equipped with one IEEE 802.11g wireless card operating in ad-hoc mode. Since the transmission power of a mesh router is not sufficient to cover the entire deployment area, thus, to achieve more coverage with the same transmission power, mesh routers form a multi-hop wireless network and forward traffic for each other providing a rich multi-hop radio connectivity for the mesh clients.

Mesh nodes use externally mounted omnidirectional antennas to broadcast wireless transmission in all directions. Omnidirectional antennas are largely applied for radio broadcasting antennas in mobile devices that use radio and for the deployment of wireless networks. Compared to directional antennas, omnidirectional antennas are very easy to install, and we do not need to care about the antenna position since it radiates and receives equally well in all directions. In addition, with mesh nodes equipped with omnidirectional antennas, it is possible to overhear transmissions from other nodes on the compass.

Each mesh client is attached to a mesh router and communicates with other mesh clients via the multi-hop network established among the mesh routers. A mobile mesh client may join the network, and its state is maintained by each mesh node in the network using the client announcement mechanism of the routing protocol [36] in order to make this client reachable from any given point in the mesh network when it moves from one mesh node to the next, i.e., a roaming client. However, to keep the wireless scenario simpler, we assume all mesh nodes in the network are static without mobility.

The network interface card of each mesh node is identified by unique MAC address, and the card is set into promiscuous mode, so we can capture all packets for subsequent analysis. Mesh nodes are identified by their MAC address and IP address. We assume all mesh nodes, including attacker nodes, have the same transmission power, and all wireless links have bidirectional, i.e., symmetric, communication capability. All nodes have minimum computational capabilities and enough hardware resources compared to any wireless mesh router with limited resources. We assume any node may be malicious and any packet sent by a given malicious node is broadcasted to all neighboring nodes.

We assume the network card of nodes does not fail, and consequently the mesh nodes do not present a faulty behavior. However, the communication channels between nodes are

susceptible to faults due to performance impairments such as packet loss caused by congestion, interference, collision, and packet corruption.

We assume that each node authorized to be part of the mesh network owns a pair of public and private keys and a client certificate that binds its public key to a unique user identifier. We assume that each mesh node can authenticate its messages by making use of security mechanisms such as digital signatures or message authentication codes (MAC). We also assume that each node has an authenticated communication channel with its neighboring nodes, which assures confidentiality, authenticity, and integrity of exchanged messages. These security techniques prevent external attacks, such as spoofing attacks and message injection attacks from outsider nodes.

3.2 Attacker Model

Normally, adversarial nodes can be either external attackers, e.g., a malicious wireless card trying to interfere with the transmissions, or insider attackers, e.g., a legitimate mesh node compromised by the attacker. We assume that adversarial nodes are insider attackers and have full control of the compromised node. The adversarial node has access to the cryptographic keys kept by the node, i.e., public and private keys. The adversarial node can perform passive attacks, such as eavesdropping, and active attacks, such as fabricating, modifying, or dropping packets. Adversarial nodes can carry out attacks either individually or coordinated with other adversarial nodes, i.e., collusion.

In this work, we focus on attacks at network layer, which is responsible for packet forwarding and routing operations. We assume that attackers do not exploit vulnerabilities of physical layer and data link layer, i.e., MAC layer. Attacks on the physical layer can be avoided by using anti-jamming broadcast communication techniques such as uncoordinated direct-sequence spread spectrum (UDSSS) [27] or delayed seed-disclosure direct-sequence spread spectrum (DSD-DSSS) [28]. And the security of MAC layer can be enforced by employing secure MAC protocols such as IEEE 802.11i [29], implemented as Wi-Fi Protected Access II (WPA2).

3.2.1 Attacking Methods

The adversarial node utilizes specific attacking methods to achieve its objectives. The attacker goals are: (i) to disrupt the routing service by violating the integrity of routing tables of mesh nodes and also routing messages; (ii) to cause DoS on data delivery, i.e., disrupt the data forwarding service; and (iii) to violate the network service and nodes availability. We analyzed common attacking methods for routing protocols of WMNs and summarized them as four basic attacking methods.

1. Message Fabrication: The attacking node uses this method to illegally inject forged routing messages into the network in order to violate the routing table of target nodes and consequently the network routing functionality. Usually, the injected messages follow the routing protocol specification, i.e., the message content is legitimate and messages interval and sequence are respected. The consequences are redirection of routes, corruption of routing tables, routing loops, and impersonification/spoofing of nodes.

2. Packet Dropping: The malicious node drops selectively/randomly forwarded routing messages and/or received routing messages in order to violate the routing table of target nodes and the routing service. The consequences are isolation of nodes from rest of network, invalidation of routes, and selfish behavior of nodes not serving as relay point to other nodes, and therefore DoS on the transmission of routing packets. Alternatively, the attacker can drop forwarded and/or received data packets to disrupt the integrity of data delivery service. For instance, in black hole attack, all data traffic is redirected to the malicious node i.e., the black hole, which do not forward any data traffic at all, then discarding all data packets. The consequence is obviously DoS on data delivery service for the affected nodes.

3. Message Flooding: This attacking method uses the same technique as message fabrication but with some essential differences, which makes it much more aggressive. The fabricated messages, that are injected by one or more malicious nodes, can contain one or more different source addresses, which can be randomly

generated, and the interval between each forged message is very short, e.g., some milliseconds. Therefore, the entire mesh network is flooded with an excessive number of forged routing messages and/or data packets that violate the network availability and integrity of routing tables. The consequence of this attacking method is DoS on nodes, where a node is prevented from receiving and sending data packets to other nodes, overflow of routing tables, consumption of valuable network resources, congestion of communication channels, and sleep deprivation in case of reactive routing protocols.

4. Message Modification: In this attacking method, also known as man in the middle attack, the attacker adulterate either the content of routing messages exchanged between nodes to disrupt the routing service integrity, or the content of data packets to disrupt the data transmission service. The consequences are corruption of routing tables and DoS on distribution of data packets for the concerned nodes.

In this work, we do not consider message modification attacks because we assume the cryptographic system ensures authenticity and integrity of exchanged messages. However, man in middle attacks can be easily identified by our intrusion detection approach. Since each node IDS shares routing information, which contain key routing parameters, with neighboring nodes for checking the routing behavior of neighbors, then any modification on these key routing parameters by the attacker would be promptly detected by the IDS.

We also do not take into consideration message flooding attacks since tracking down the source of the intrusion for certain types of attacks might be a difficult task if the attack is carried out through intermediaries nodes. For example, distributed denial of service (DDoS) attack is basically a message flooding attack. However, the malicious flooding traffic comes from innocent nodes that are simply rebroadcasting the phony messages received from neighboring nodes, as expected by the protocol behavior, that are received from other neighbors, and that were primarily originated by the malicious nodes. Thus, the IDS would obviously identify these intermediate nodes as the flooding nodes but it cannot trace the attack further back to the remote source nodes that firstly created the malicious traffic since the IDS communication is limited to local neighborhoods.

We do not consider routing loop attacks, where the attacker introduces a loop in the route path between two nodes by forging routing messages from a legitimate node in order to alter the route of target nodes, then violating the routing table of nodes and the normal routing operation. This attack causes data packets from nodes to flow between two or more nodes without being forwarded to the correct destination. Since routing loop attack is based on message fabrication technique, our solution can suitably identify this attack.

We also do not consider wormhole attacks. In this attack, two malicious nodes collude to create a virtual link between each other, i.e., a hidden tunnel that is alleged to be the best path between two points in the network, to secretly transmit packets from other nodes. As a consequence, intermediate nodes in the correct path between these two network points are unable to participate in the network operations since routes through these node are permanently denied, and then the attacker can drop all packets or conditionally forward packets from the other nodes. Several works [30][31] address this type of attack. However, with our intrusion detection solution is possible to detect such wormholes since attackers basically employ message fabrication and/or packet dropping attacking methods to violate the routing table of target nodes so that these nodes choose the route passing through the malicious node as best option to the destination. In addition, if the two malicious nodes are neighbors, the intrusion detection approach precisely detects the wormhole attack.

Moreover, coordinated attacks, such as byzantine attacks, which are carried out by a group of attackers, can be detected by using our solution since there is high cooperation between nodes IDSs for analyzing the routing behavior of local neighborhoods. This type of attack essentially use message fabrication or message flooding attacking methods to concurrently compromise the integrity and availability of network operations and nodes.

The aforementioned attacking methods are realistic and cover most of the attacking techniques of WMNs. In addition, they can be implemented in our virtualized mesh network platform without great difficulty. Routing manipulation attack, which is based on message fabrication attacking method, is detailed in this chapter. Detection of message fabrication attacking method is studied in Chapter 4, and packet dropping attacking method detection is introduced in Chapter 5.

3.3 BATMAN

In this work, we make use of Better Approach To Mobile Ad hoc Networking (BATMAN) [7], a proactive routing protocol for wireless ad-hoc mesh networks. We opted for BATMAN as routing protocol since results from practical experiments with BATMAN and other routing protocols [32][33], such as OLSR, have shown that BATMAN outperforms these protocols on most of performance metrics such as route trip delay (in terms of number of hops), route convergence latency, and packet loss rate. BATMAN achieves higher level of stability and packet delivery ratio presenting a lower route change frequency compared to traditional routing protocols. However, BATMAN presents a higher message overhead to achieve the lowest packet loss. To overcome this problem, BATMAN designers are developing a new scheme to optimize BATMAN message flooding algorithm [34].

3.3.1 Protocol Overview

The majority of routing protocols employed in WMNs were designed based on traditional routing protocols from ad-hoc networks, which were conceived taking into consideration the high mobility of nodes. Common routing protocols such as OLSR, DSR, and AODV are good examples, which were developed built on the assumption that the network is constantly modifying its topology because of nodes mobility and signal losses over the wireless communication medium. Mesh networks can be seen as a special type of ad-hoc network where little or no mobility is expected and only occasional route fluctuations happen. BATMAN protocol attempts to maximize traffic throughput and minimize packet delay in the network instead of just keeping basic connectivity among the nodes.

BATMAN routing protocol was designed as a response to OLSR drawbacks. A community wireless network known as Freifunk in Berlin (Germany) noticed that OLSR suffered from many performance limitations when the mesh network grew very large, e.g., more than 300 nodes [35]. These limitations include routes going up and down regularly due to routing tables being unnecessary flushed because of routing loops, and it took several seconds, for a small router device, to recalculate the whole topology graph in a mesh network with hundreds of nodes. Then, they decided to develop a new routing algorithm for large static WMNs to fulfill these requirements.

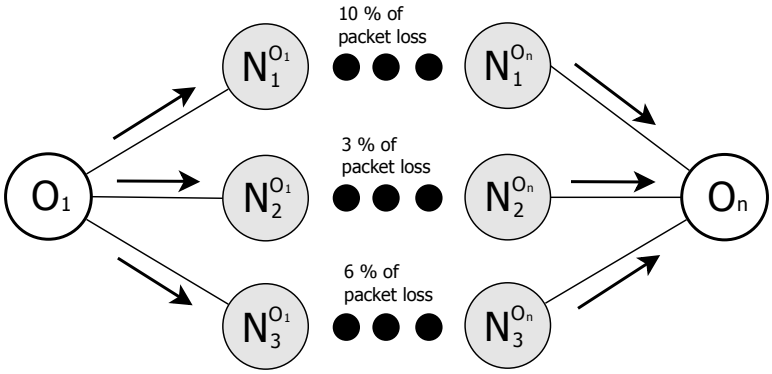
BATMAN algorithm's approach is to share the information of best paths between nodes in the network to all participating nodes. Each node keeps only the information of the best next-hop for every destination in the WMN. BATMAN algorithm is only concerned with learning about the best next-hop through which it can find the best path since every node in the path knows the best next-hop it has to use to send messages. Therefore, it is not necessary to find or calculate the complete route for all destination nodes in the mesh network. BATMAN implements a flooding mechanism based on sequence numbers which avoids propagating contradicting topology information and limits the amount of topology messages flooded in the mesh network. BATMAN algorithm is designed to deal with unreliable links of the WMN by performing statistical analysis of packet loss and propagation speed in order to select the most reliable route upon the next-hop.

3.3.2 Flooding Mechanism

The BATMAN routing algorithm [7] is detailed as follows.

1. Each node (also known as Originator) periodically broadcasts hello messages, referred as Originator Messages (OGMs), to tell its neighbors about its existence. An OGM contains: the Originator Address, the Source Address, the Previous Sender address, a Time To Live (TTL) value, a unique Sequence Number value, a Transmission Quality (TQ) value, Translation Table (TT) count, and Gateway Flags.
2. Following a neighboring node receives an OGM: it modifies the Source Address to its own address and rebroadcast this OGM in accordance to BATMAN forwarding rules to tell its neighboring nodes about the existence of the node that originated the OGM, and so on and so forth.
3. Hence, the mesh network is flooded with OGMs until every node has received it at least once, or until they got lost because of the packet loss of wireless links, or until their TTL value has expired. Upon receiving a self-originated OGM, the Originator performs a bidirectional link check to verify that the detected link to a given neighbor can be used in both directions.
4. The Sequence Number value of an OGM is utilized to verify the message freshness,

i.e., to distinguish between new OGMs and duplicated OGMs in order to guarantee that each OGM is only counted once. The amount of OGMs, i.e., the total number of Sequence Numbers, received from a given Originator via each neighboring node is used as a metric to estimate the route quality. Thus, BATMAN chooses as best next-hop to this Originator, the neighboring node from which it has received the highest amount of OGMs within a Sliding Window (a packet count metric).



(a) Mesh network.

Node	O_1	$N_1^{O_1}$	$N_2^{O_1}$	$N_3^{O_1}$	$N_1^{O_n}$	$N_2^{O_n}$	$N_3^{O_n}$
Broadcasted OGMs	100	-	-	-	-	-	-
Rebroadcasted OGMs	-	100	100	100	-	-	-
Received OGMs	-	-	-	-	90	**97**	94

(b) OGMs broadcasted by node O_1.

Originator	Next-hop	Potential next-hop
O_1	$N_2^{O_n}$	$N_3^{O_n}$

(c) Routing table of node O_n.

Figure 3.1: BATMAN flooding scheme.

OGMs that travel on bad routes where the wireless link quality is poor will experience packet loss or delay on their path through the mesh network. On the contrary, OGMs that follow good routes will be transmitted faster and more reliable. To find out which is the best route to a destination node, BATMAN counts the number of Sequence Numbers

originated by that node and received from its different neighboring nodes. Thus, BATMAN chooses as next hop the mesh nodes in the path to the destination node which possess the best transmission quality. BATMAN makes use of this information to maintain a routing table containing the best next-hop towards every Originator in the mesh network.

BATMAN flooding mechanism is illustrated in Figure 3.1. In Figure 3.1 (a), node O_n will know about the existence of node O_1 in distance by receiving the OGMs originated by node O_1 which are rebroadcasted by its single hop neighboring nodes $N_1^{O_1}$, $N_2^{O_1}$, and $N_3^{O_1}$. Since node O_n has more than one neighboring node, it has to decide which neighbor it will select as best next-hop to send data to distant node O_1. This decision is taken based on the number of OGMs node O_n has received faster and more reliable via one of its single hop neighboring nodes $N_1^{O_n}$, $N_2^{O_n}$, and $N_3^{O_n}$, as illustrated in Figure 3.1 (b). Then, node O_n chooses node $N_2^{O_n}$ as its current best next-hop, i.e., best ranking neighbor, towards node O_1 because node $N_2^{O_n}$ has forwarded the most number of OGMs originated by node O_1. Finally, node O_n updates its routing table accordingly, as seen in Figure 3.1 (c).

3.3.3 BATMAN Advanced

BATMAN advanced, also known as *batman-adv* [36], is the implementation of BATMAN routing protocol for MAC Layer, i.e., OSI Layer 2, in form of Linux kernel module. The majority of ad-hoc network routing protocols are designed for Layer 3, i.e., the network layer, which means they use UDP packets for exchanging routing information and also have to manipulate the kernel routing table for updating routing information.

In *batman-adv*, the routing information and data traffic are transmitted using raw Ethernet frames. *Batman-adv* encapsulates and forwards all data traffic until it reaches the destination, thus emulating a virtual network switch of all participating mesh nodes. Hence, all nodes seem to be link local, i.e., they are neither aware of the network's topology nor affected by network changes. The advantages of this design are: the user can execute other protocols, such as IPv4, IPv6, and DHCP, on top of *batman-adv* protocol, nodes can participate in the mesh network without the need of having an IP address, mobile/non-mesh nodes can easily be attached to the mesh network keeping their connectivity, and the data flow is optimized through the node mesh interface.

3.4 Routing Manipulation Attack

This section describes the routing manipulation attack against BATMAN, where a malicious node disrupts the network routes by violating the integrity of nodes' routing tables. We use a state transition diagram to describe this attack, where the malicious node manipulates the routes of mesh nodes in order to divert their traffic to itself.

3.4.1 Attack Methods

A malicious node can fabricate and broadcast OGMs, for a legitimate mesh node, with continuous valid Sequence Numbers, which it actually did not receive, so it can control the routes of target nodes in order to divert these routes to pass through the malicious node to a given destination [7]. As routing decisions are based on statistical analysis of the amount of OGMs received instead of information contained in the packets, the attacker only needs to generate a number of OGMs that are numerous enough to redirect the routes of target nodes to the compromised node. In other words, the attacker has to continuously win the neighbor ranking of target nodes towards the destination, thus overriding the number of legitimate OGMs broadcasted by the destination node. The attacker can apply two methods to carry out the attack:

1) **Attacking Method 1**: Update the neighbor ranking range of target nodes.
   In this method, the attacker creates phony OGMs for the Originator containing Sequence Number values out of the actual Sliding Window size, i.e., the range of Sequence Number received so far for that Originator. Thus, target nodes will presume that the Originator has reinitiated and will update their neighbor ranking range of Sequence Numbers for the Originator. The attacker has to constantly generate OGMs with valid Sequence Numbers to persuade all target nodes to update their neighbor ranking range every time they receive the phony OGMs. If the Sliding Window of the link, via which the attacker's OGMs have been received, has the most Sequence Numbers, then this link is considered to be the best link towards the Originator, which certainly goes through the compromised node.

2) **Attacking Method 2**: Not update the neighbor ranking range of target nodes.
In this method, the attacker generates phony OGMs for the Originator with Sequence Number values a few counts ahead of the current legitimate Sequence Numbers broadcasted by the Originator. These fake OGMs will be preferred in the neighbor ranking range of target nodes. However, the neighbor ranking range for the Originator is not updated by the target nodes. Anyhow, the malicious node has to broadcast a sufficient number of fake OGMs to win the neighbor ranking range of target nodes, so they choose this link as best next hop.

3.4.2 Example of Attack Scenario

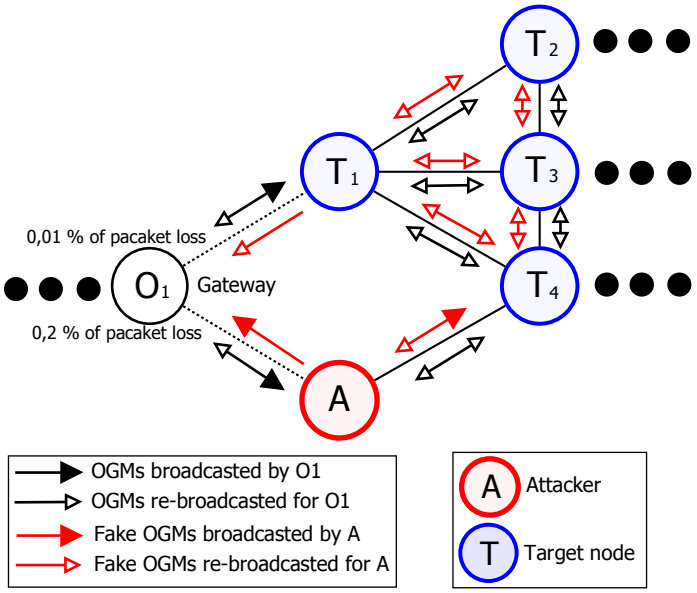


Figure 3.2: Example of route manipulation attack.

Route manipulation attack is exemplified in Figure 3.2. We assume that node A is the attacker, and nodes T_1, T_2, T_3, and T_4 are the target nodes. Nodes T_1 and A are the neighboring nodes of gateway node O_1. As the link between nodes O_1 and T_1 has a packet loss rate lower than the one between nodes O_1 and A, then target nodes T_2, T_3, and T_4 prefer a route via node T_1 toward node O_1. Thus, the compromised node A generates and broadcasts OGMs for Originator O_1 with continued valid Sequence Number values, therefore applying Attacking Method 2, in order to redirect the routes of target nodes T_2,

T_3, and T_4 toward node O_1 to node A. The broadcasting rate of fake OGMs used by the malicious node A is higher than BATMAN default rate defined for the nodes, than attacker node A always wins the neighbor ranking of target nodes towards the destination node O_1.

In accordance with BATMAN algorithm, the target nodes will choose compromised node A as the best next-hop to node O_1 because node A has rebroadcasted the most number of OGMs for Originator node O_1. Since node A is the only routing point for the target nodes to reach gateway node O_1, it can perform the following misuses without being noticed:

- Discard data packets or OGMs that go through it. For example, if node A drops all OGMs broadcasted by target node T_4 and at the same time node T_1 crashes, then node T_4's route entry will be deleted from the routing table of node O_1. Consequently, node T_4 will be unable to exchange data traffic with gateway node O_1 and it will lose Internet connection being disconnected.
- Modify data packets or OGMs that pass through it. For example, node A can modify the destination address of data packets from node T_4 in order to misguide the other mesh nodes to forward node T_4's packets to wrong destinations.
- Delete some of the content of data packets or OGMs. For instance, node A can delete the Originator Address of OGMs broadcasted by node T_4 so that the mesh nodes cannot find a route to node T_4, i.e., the route entry of node T_4 is permanently removed from the routing tables of other nodes.

3.4.3 Attack Model

Figure 3.3 illustrates the extended state machine diagram for route manipulation attack. The state machine is triggered when an OGM is received by the malicious node, and then the machine moves to initial state "OGM received" and executes "process OGM" action. Thus, the node processes the OGM having two possibilities, i.e., two transitions can be activated by "OGM processed" trigger and both return to initial state. The first transition has its guard condition evaluated as true if the OGM's Originator Address is the same as the node's address. That means the OGM was initially originated by the own node, so "Drop OGM" action is executed by this transition. In the second transition, the guard condition evaluates to true if the OGM's Originator Address is not the same as to the node's address.

That implies that the OGM was originated by a different node, so the transition executes "Rebroadcast OGM" action. If the compromised node executes a malicious action, i.e., "Attacker action" trigger is fired, then a third transition of "OGM received" state moves the machine to "OGM broadcasted" state. That means "Fabricate and broadcast OGM" action of transition is executed and the guard condition of "OGM's Originator Address is not the same as the node's address" is evaluated to true.

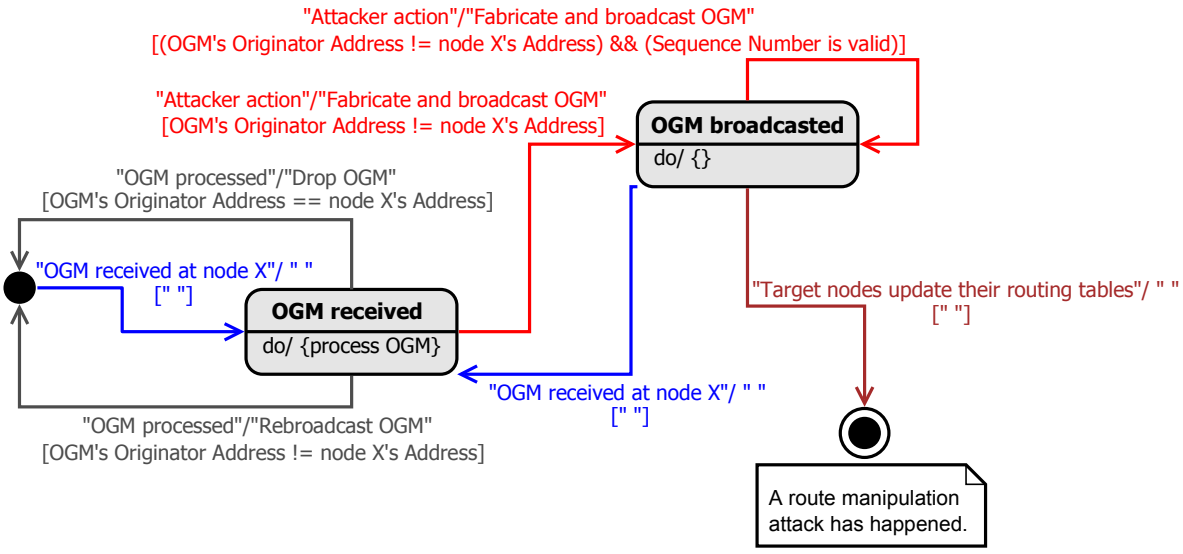


Figure 3.3: State machine for routing manipulation attack.

In "OGM broadcasted" state, if the attacker continues fabricating and broadcasting OGMs with the Originator Address different from the node's address, the state transition is constantly fired, which basically goes back to "OGM broadcasted" state. Finally, if the target nodes update their routing tables' best next-hop for the Originator that is being broadcasted by the fake OGMs of attacker, then the second transition of "OGM broadcasted" state is fired. Thus, the state machine goes to final state, which implies that route manipulation attack has been succeeded.

3.5 Experiment and Results

In this section, we present the experimental results concerning the emulation of routing manipulation attack targeting BATMAN routing protocol. The attack is implemented in our virtualized mesh network platform. We performed several experiments by using the

attacking methods described in Section 3.4.1.

3.5.1 Experiment Environment

The routing manipulation attack was implemented in a virtualized network environment, which emulates the mesh network topology. The mesh network is emulated by VMs executing BATMAN protocol, which represent the mesh nodes. We performed a series of experiments in order to verify the damage caused by the attack in the routing service of nodes. The following virtualized architecture setup has been employed:

- A main machine consisting of Intel(R) Xeon(R) CPU W3550 @ 3.06GHz with 12.0 GB of RAM, executing 64-bit Ubuntu Linux OS with Kernel 2.6.35, is used to run QEMU emulator [37] instances for virtualization of individual nodes.
- Each QEMU instance, executing a 32-bit Ubuntu Linux OS with kernel 2.6.35, is attached to a virtual switch (*vde_switch* tool [38] with some modifications). QEMU instances are connected with each other by using *wirefilter* tool [38], which emulates bi-directional communication links between nodes.
- A QEMU instance consists of standard PC 32Bit computer architecture with one CPU, that is the default 32Bit CPU type of QEMU, and 256 MB of RAM memory, supporting Linux OS. We assume each mesh node in the network has the same hardware configuration as the QEMU instance, particularly a Wi-Fi router.
- The virtual network configuration employs a fixed topology, where each virtual node is connected to a virtual switch instance, which only allows packets to be sent if they come from the virtual neighbors of node. In that way, virtual links between nodes are properly emulated. Link impairments, such as packet loss and packet delay, are emulated by applying *wirefilter* tool.

Batman-adv v.2010.1.0 [36] is compiled from source code and loaded into the Linux kernel of each virtual node as kernel module. BATMAN operates on the network interface of each virtualized node. We assume each node has only one network interface card which is used by BATMAN routing protocol. Originator Interval default value of 1000 ms, i.e., a value in milliseconds that determines how often BATMAN broadcast OGMs, is employed.

We use *batman-adv* event logging and debugging mechanism in order to obtain BATMAN related debugging information. Debugging information refers to decisions made by BATMAN about routing, flooding, and message forwarding operations. Then, all debugging information is saved to log files which are synchronized with a global clock, i.e., the time of the main machine that launches the VMs. We assume that each node clock is synchronized with the global clock of main machine. During collection of debugging data at each virtual node, the average CPU usage of main machine was about 40% and average main memory usage was around 60%. Then, there was no performance degradation and resource contention, which could interfere in the process of results collection.

3.5.2 Attack Implementation

We implemented the attack using packETH tool [39], an Ethernet packet generator. For that, we added new functionalities to the tool which allows fabricating and broadcasting OGMs, with continuous valid Sequence Number values, on the virtual node's network interface. Since the tool does not allow updating Sequence Number field, that is 32 bit, in an OGM packet, we added this feature to the tool interface so we can define initial Sequence Number values and increment the value in conformance with BATMAN flooding mechanism. In that way, we are able to emulate routing manipulation attacks by executing the modified tool on the compromised node.

We emulate the routing manipulation attack in two different network mesh scenarios. In Scenario 1, we add one attacker node in a smaller virtual mesh network. In Scenario 2, we add two attackers in a more complex mesh network topology.

3.5.3 Attack Emulation in Scenario 1

The emulated mesh network for this scenario has seven mesh nodes, as illustrated in Figure 3.4. Scenario 1 comprises one attacker A_2, one gateway node O_1, and five target nodes T_3, T_4, T_5, and T_6. Attacker A_2 generates and broadcasts phony OGMs for node O_1 to divert the routes of target nodes toward node O_1 to malicious node A_2.

The duration of each experiment is 20 minutes. All virtual nodes take about 4 minutes to completely initialize, which includes configuring the virtual connections, i.e., setting up links with neighboring nodes, and establishing routes with the other nodes. After that,

malicious node A_2 starts broadcasting the forged OGMs.

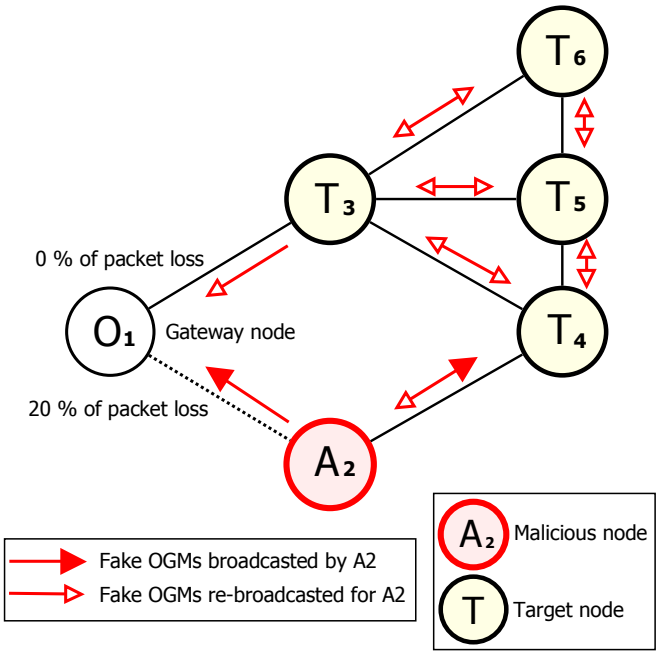


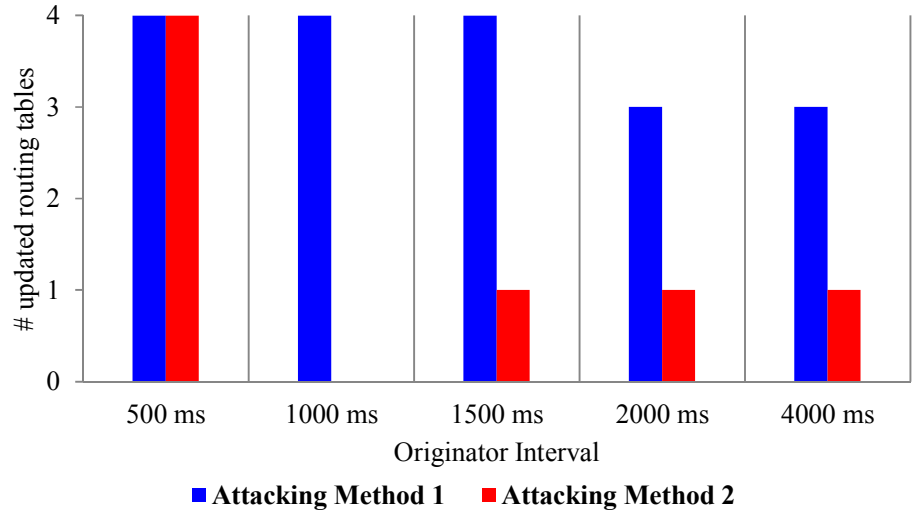
Figure 3.4: Mesh network topology for Scenario 1.

We applied the two routing manipulation attack techniques introduced in Section 3.4.1.

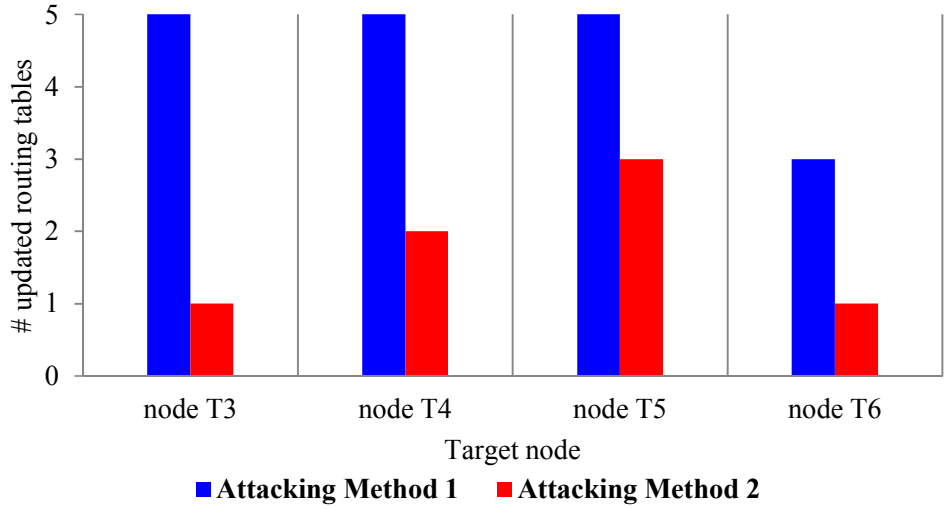
1. **Attacking Method 1**: Update the neighbor ranking range of target nodes.
   The attacker generates OGMs with Sequence Numbers values at least twice the values of Sequence Numbers of OGMs broadcasted by node O_1, e.g., when the attacker launches the attack if current Sequence Number value of node O_1's OGM is 500, then the attacker generates OGMs with Sequence Numbers values starting from 1000.
2. **Attacking Method 2**: Not update the neighbor ranking range of target nodes.
   The attacker generates OGMs with Sequence Number values close to current Sequence Number values broadcasted by node O_1, i.e., the attacker's Sequence Number values are a few counts ahead of node O_1's Sequence Number values.

Furthermore, we take into account different Originator Interval values for broadcasting the phony OGMs of attacker, so we can verify the effectiveness of the attacking methods. We used Originator Interval values of 500 ms, 1000 ms, 1500 ms, 2000 ms, and 4000 ms.

Before starting the attack, we record the routing tables of target nodes in order to check Originator O_1's route entry. In the routing tables, route entry of Originator O_1 points to a route via node T_3 since packet loss rate of link between nodes O_1 and T_3 is lower than the one between nodes O_1 and A_2. We also record the routing tables of target nodes after each attack execution in order to check if node O_1's route entry was updated in the routing tables of target nodes which points to a route via node A_2 as a consequence of phony OGMs broadcasted by the attacker in the network.



(a) Routing table updates concerning Originator Interval values.



(b) Routing table updates concerning the target nodes.

Figure 3.5: Routing table updates for Scenario 1.

Figure 3.5 shows the experiment results obtained for Scenario 1. In Figure 3.5 (a), for Attacking Method 1 and Originator Interval values of 500 ms, 1000 ms, and 1500 ms, we can notice that the number of OGMs generated by attacker node A_2 easily win the neighbor ranking of all target nodes T_3, T_4, T_5, and T_6. Consequently, target nodes are forced to update their routing tables, as presented in Figure 3.5 (b). Nevertheless, the routing table of node T_6 is not updated for Originator Interval values of 2000 ms and 4000 ms because the Originator Interval value of the phony OGMs broadcasted by the attacker is higher than node O_1's default Originator Interval of 1000 ms . That means the number of phony OGMs fabricated by the attacker cannot override the number of legitimate OGMs of node O_1. Thereby, node T_6 does not update its current best next-hop towards node O_1.

For Attacking Method 2 and Originator Interval value of 500 ms, we can notice all target nodes T_3, T_4, T_5, and T_6 update their routing tables, as presented in Figure 3.5 (a). Since the Originator Interval of attacker is lower than default Originator Interval value of node O_1, the phony OGMs continuously win the neighbor ranking of target nodes. However, none of routing tables of target nodes is updated for Originator Interval of 1000 ms because the number of phony OGMs fabricated by the attacker is not enough to surpass the number of legitimate OGMs broadcasted by node O_1. Concerning the other Originator Interval values, at least one routing table is updated by the target node since the phony Sequence Numbers generated by the attacker differ more than the Sliding Window size for Originator O_1 in the target node.

The experiments results show that a compromised node can efficiently violate the routing table of any mesh node in the network by simply broadcasting a sufficient number of fake OGMs for a given legitimate mesh node. So the routes of target nodes toward the legitimate mesh node will be redirected to the compromised node. To make sure that the forged OGMs will always overcome the legitimate OGMs, the attacker has to execute Attacking Method 1 with an Originator Interval value lower than 1500 ms. A second option is to execute Attacking Method 2 with an Originator Interval value lower than 500 ms, i.e., the attacker has to send fake OGMs at a high broadcasting rate.

3.5.4 Attack Emulation in Scenario 2

The emulated mesh network topology used in this scenario is illustrated in Figure 3.6. The

mesh scenario comprises eleven nodes: two attackers A_4 and A_9, two gateway nodes O_1 and O_{10}, and seven target nodes T_2, T_3, T_5, T_6, T_7, T_8, and T_{11}. Attacker A_4 generates and broadcasts fake OGMs for node O_1, and attacker A_9 generates and broadcasts fake OGMs for node O_{10} in order to divert the routes of target nodes toward nodes O_1 and O_{10} to malicious nodes A_4 and A_9 respectively.

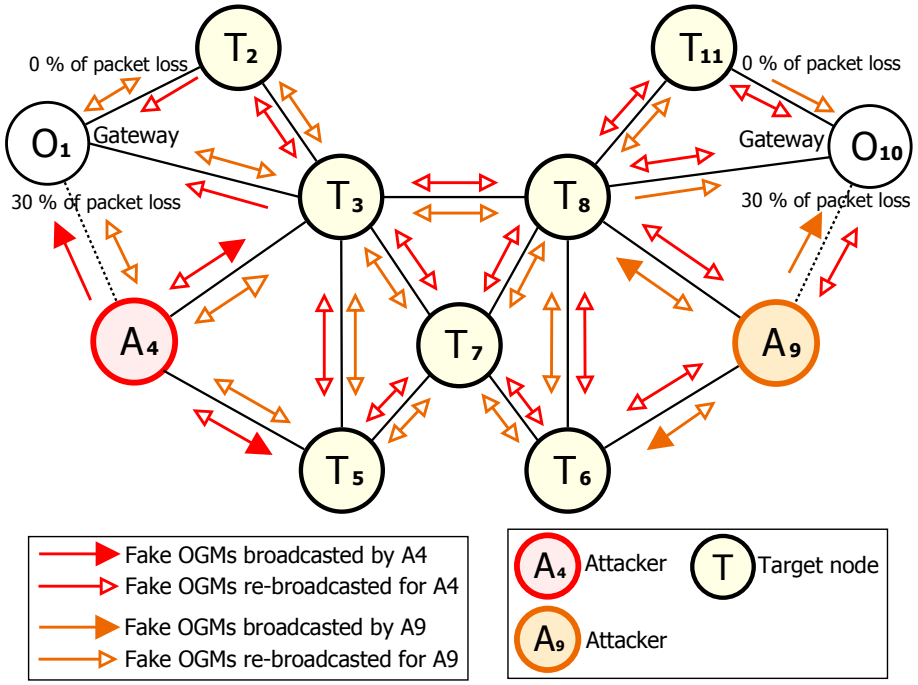
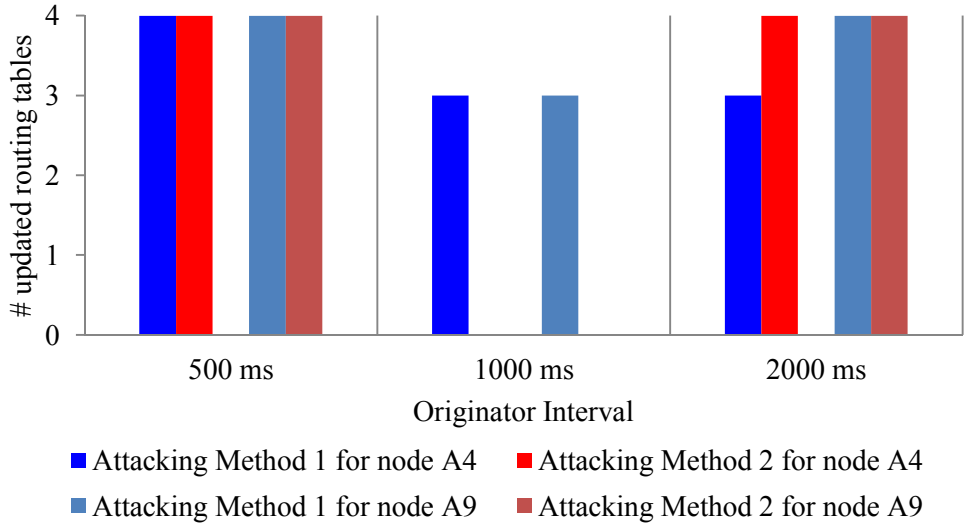


Figure 3.6: Mesh network topology for Scenario 2.

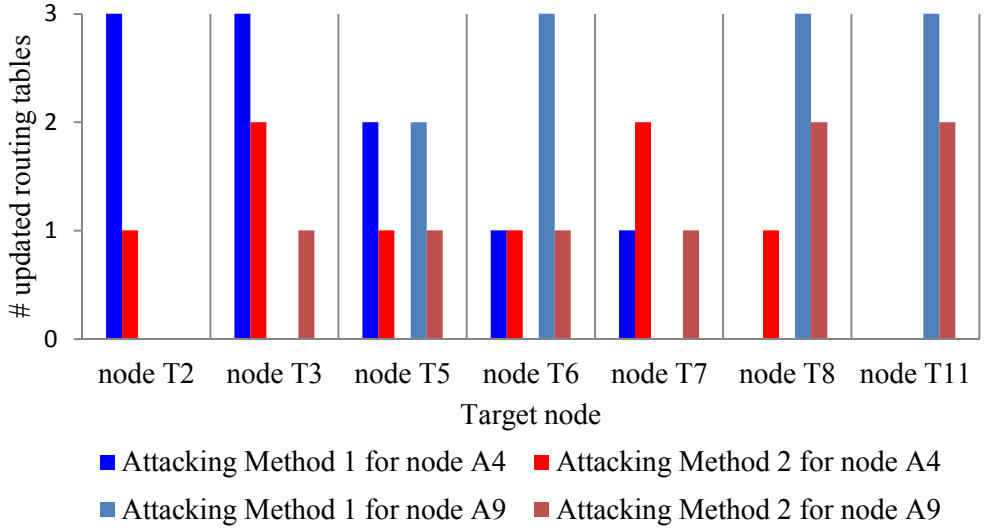
The duration of each experiment is 20 minutes. After 5 minutes of network initial set-up for all the mesh nodes, malicious nodes A_4 and A_9 start to generate their respective phony OGMs. In this scenario, we also apply the two methods of routing manipulation attack introduced in Section 3.4.1. The attackers A_4 and A_9 applies the same strategy, for Attacking Methods 1 and 2, as performed for Scenario 1 in Section 3.5.3 when creating and broadcasting fake OGMs for Originators O_1 and O_{10} respectively. We consider Originator Interval values of 500 ms, 1000 ms, and 2000 ms for both attacking methods.

Before executing the attacks, we record the route entries of Originators O_1 and O_{10} in the routing tables of target nodes. The routes entries point to paths via nodes T_2 and T_{11} because these nodes have the best links, i.e., the lower packet loss rate, to gateway nodes

O_1 and O_{10}, respectively. After executing the attacks, we check if the route entries of Originators O_1 and O_{10} have been updated in the routing tables of target nodes which point to paths via nodes A_4 and A_9 respectively as a result of the fake OGMs broadcasted by malicious nodes A_4 and A_9.



(a) Routing table updates concerning Originator Interval values.



(b) Routing table updates concerning the target nodes.

Figure 3.7: Routing table updates for Scenario 2.

Figure 3.7 shows the experiment results obtained for Scenario 2. In Figure 3.7 (a), for

Attacking Method 1 and Originator Interval values of 500 ms, 1000 ms, and 2000 ms, we can observe that most of target nodes update their routing tables because of the fake OGMs fabricated by attackers A_4 and A_9. In Figure 7 (b), attacker A_4 forces target nodes T_2, T_3, and T_5 to update their best ranking neighbor since these target nodes are the only one-hop neighbors of gateway node O_1 and malicious node A_4. However, attacker A_9 has more influence on the routing table of target nodes T_6, T_8, and T_{11} because these target nodes are one-hop neighbors of gateway node O_{10} and malicious node A_9.

In Figure 3.7 (b), we can observe attacker A_4's fake OGMs had a limited influence on target nodes T_8 and T_{11}, and attacker A_9's fake OGMs did not affect target nodes T_2, T_3, and T_7 then not updating their routing tables. The reason is: since fake OGMs of attackers A_4 and A_9 go through the best ranking neighbors, i.e., nodes T_3 and T_8, which are the same best next hops of gateway nodes O_1 and O_{10}, then there is no need for these target nodes to update their best ranking neighbor. For the same reason, the routing table of target node T_7 is updated the smallest number of times. On the contrary, target nodes T_5 and T_6 are the only nodes impacted by both attackers A_4 and A_9 since they have more neighboring nodes to select as best next hop in the neighbor ranking.

In Attacking Method 2 and using Originator Interval value of 500 ms, we can see that all of target nodes update their routing tables, as illustrated in Figure 3.7 (a). Since the number of fake OGMs broadcasted by attackers A_4 and A_9 is higher than the number of legitimate OGMs of nodes O_1 and O_{10}, then the attackers continuously win the neighbor ranking of target nodes as occurred in Attacking Method 1. None of the routing tables is updated for Originator Interval value of 1000 ms since the fake OGMs are not enough to exceed the number of legitimate OGMs. Target nodes update their routing tables for Originator Interval value of 2000 ms because the fake Sequence Numbers sent by attackers differ more than the Sliding Window size of target nodes.

From the experiments results, we can notice that attackers can easily modify the routes of target nodes and violate their routing tables by employing Attacking Method 1 using Originator Interval values of 500 ms, 1000 ms, and 1500 ms, or optionally employing Attacking Method 2 using Originator Interval values of 500 ms, and 1500 ms.

3.6 Attack Detection

In this section, we present an efficient attack detection scheme that is based on analysis of BATMAN traces collected from different mesh nodes. The detection approach relies on tracing routing messages flooded by each BATMAN node through the mesh network. The detection scheme assumes that each node collect sufficient routing-related information so it can validate the consistency of related routing packets by using constraints.

We implemented the attack detection mechanism into BATMAN Advanced Control and Management (*batctl*) tool [36] so we are able to detect routing manipulation attacks. *Batctl* is a BATMAN tool used to configure and debug *batman-adv* kernel module. It also offers a Layer 2 version of *ping*, *traceroute*, and *tcpdump* tools since all nodes participating in the virtual switched network are completely transparent for all protocols above Layer 2. Then, common network diagnosis tools, which are mainly designed for Layer 3, do not work as expected. *Batctl* tool analyses BATMAN traces as follow.

1. *Batctl* makes use of *batman-adv* debugging output, i.e., log files of BATMAN nodes, in order to create a small database of all Sequence Numbers broadcasted by all mesh nodes in the network.
2. Thus, the tool employs this database, for instance, to trace all Sequence Numbers of a given Originator and build a tree structure of every Sequence Number broadcasted by this Originator node. The tree structure describes the path a Sequence Number travel through the network for the moment it is broadcasted by the Originator, then rebroadcasted by the neighboring nodes of the Originator, and so on, until every node has receive it at least once.
3. The root node of the tree is the Originator of Sequence Number, the child nodes of root node are the neighboring nodes of Originator that rebroadcasted the Sequence Number, and so on.

For example, to detect whether a compromised node A is broadcasting forged Sequence Numbers for a legitimate node O_n in the network, we follow the procedure.

1) Firstly, we apply *batctl* tool to create a tree structure for each Sequence Number

broadcasted by node O_n.

2) After, we check in each tree structure if the following constraint is violated:
   - $C1$: if any child node is equal to the root node, i.e., the own Originator O_n.

   If we find such mismatch, there are two explanations.
   i. The first one means a legitimate OGM of node O_n was broadcasted back to node O_n, what is not possible according to BATMAN flooding mechanism described in Section 3.3.2.
   ii. The other possibility is the compromised node A fabricated and broadcasted an OGM for O_n and this OGM was broadcasted back to node O_n, which is more realistic.

We implemented constraint $C1$ into *batctl* tool in order to check for this type of inconsistency when creating the tree structure for Sequence Numbers of a given Originator and also count the number of inconsistencies found for each Originator, i.e., the number of times constraint $C1$ is violated. Then, we executed the modified tool version with the log files captured from all mesh nodes of Scenario 1 in Section 5.1 in order to trace the Sequence Numbers of all nodes but specially Originator O_1, and with log files of all nodes of Scenario 2 in Section 5.2 to particularly trace the Sequence Numbers of Originators O_1 and O_{10}. These Originators are the ones the attackers fabricate phony OGMs for.

3.6.1 Results for Scenario 1

We use the improved *batctl* tool to trace all of the Sequence Numbers broadcasted by each Originator in Scenario 1 and also counts the number of mismatches found for each node. Figure 3.8 presents the attack detection results for mesh nodes of Figure 3.4 when attacker A_2 broadcasts fake OGMs with Originator Interval of 500 ms.

For both Attacking Methods 1 and 2, we can note Originator O_1 has the highest rate of inconsistencies detected in its Sequence Number trees, i.e., 43% and 33% respectively. In Attacking Method 1, 43% of OGMs broadcasted for Originator O_1 in the network are malicious, and in Attacking Method 2, 33% of OGMs broadcasted for Originator O_1 are malicious. As next step, we have to look into Sequence Number trees of Originator O_1 to verify who is supposedly broadcasting fake Sequence Numbers for node O_1.

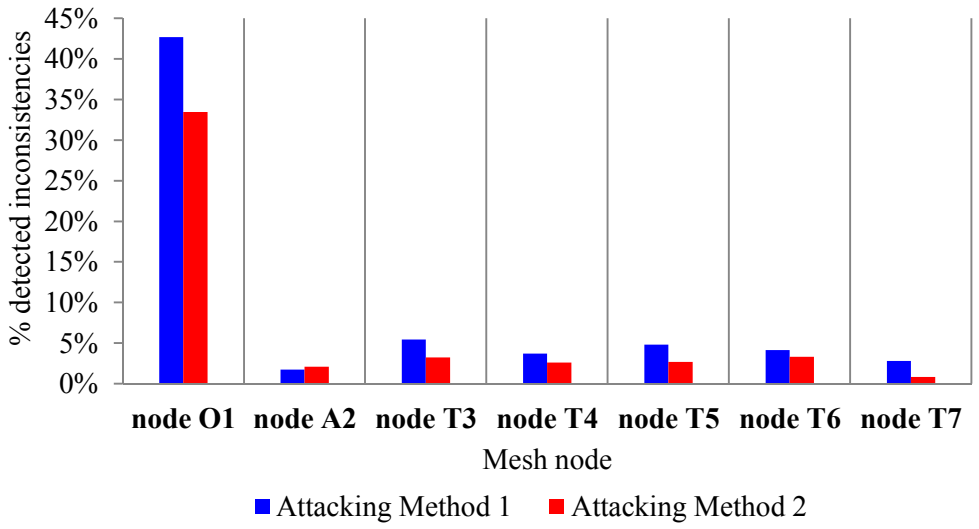


Figure 3.8: Detection results for Scenario 1.

In Figure 3.8, we can note that concerning the other mesh nodes the rate of detected inconsistencies is low having values below 6%. In fact, these inconsistencies do not properly correspond to attack attempts, and they can be considered as false positives. These inconsistencies occur at the beginning and at the end of each experiment. They are caused by the lack of synchronization between the mesh nodes at the time the logging mechanism of each node starts and stops saving BATMAN debugging information. For instance, at the time node T_3 stops recording debugging output, its log file contains more information about the broadcasted Sequence Numbers of node O_1 than the log files of own node O_1 and node A_2 which stopped capturing debugging output earlier. Therefore, *batctl* tool is not able trace the complete path of certain Sequence Numbers in the log files of these mesh nodes since it is missing information from other nodes. Then, *batctl* tool is obliged to build bogus Sequence Number trees which cause these incorrect inconsistencies, i.e., false positives.

Figure 3.9 illustrates an example of inconsistency detected for Originator O_1 in Scenario 1. The picture shows the Sequence Number tree of a phony OGM broadcasted by attacker A_2 for Originator O_1. In Figure 3.9, the compromised node A_2 generates and sends a Sequence Number for node O_1, which is the root of the tree. Thus, this Sequence Number is broadcasted on the interface of nodes A_2, O_1, and T_4 that are the child nodes of root node. At this time, the tool promptly detects an inconsistency since it violates constraint $C1$, i.e., "the child node O_1 is equal to root node O_1".

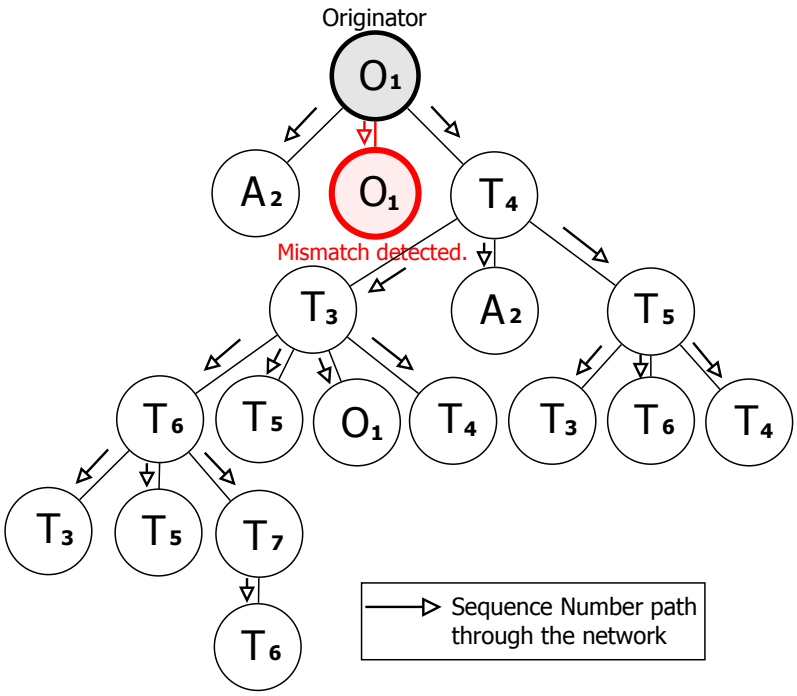


Figure 3.9: Sequence Number tree for node O_1.

We can remark that the source of attack is node A_2 since this phony OGM of Originator O_1 is broadcasted at the same time to nodes A_2, O_1, and T_4 and according to the network topology of Figure 3.4 the only one-hop neighbors of node A_2 are nodes O_1 and T_4, and the inconsistency was found in the Sequence Number tree of node O_1. Therefore, by using elimination we conclude that node A_2 is the malicious node.

Figure 3.10 shows the attack detection results for node O_1 at different Originator Interval values. We can observe that for Attacking Methods 1 and 2 the rate of detected inconsistencies decreases as the Originator Interval value increases. This decrease in the rate of detected inconsistencies is considered a normal behavior since if Originator Interval increases then the broadcasting rate of fake Sequence Numbers decreases accordingly, what means that the attacker broadcasts less malicious OGMs.

Nonetheless, in Attacking Method 2, the detection rate for Originator Intervals values of 1000 ms, 1500 ms, 2000 ms, and 4000 ms are even lower than the respective ones of Attacking Method 1. Since the values of Sequence Numbers of fake OGMs are close to values of Sequence Numbers of legitimate OGMs, legitimate OGMs of node O_1 overlap with fake OGMs broadcasted by the attacker. Then, *batctl* tool cannot correctly trace the

path that fake Sequence Numbers follow in the network, so making it difficult for the tool to identify inconsistencies in the Sequence Number trees.

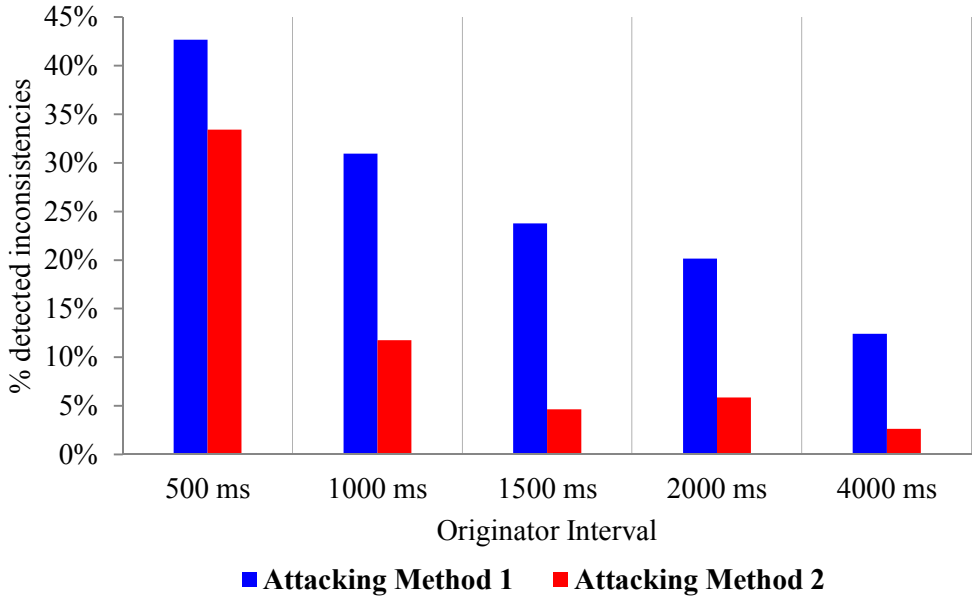


Figure 3.10: Detection results for node O_1.

The modified version of *Batctl* tool, for detection of routing manipulation attacks, had a good performance in the role of detecting fake OGMs broadcasted by a malicious node in the mesh network with low false positives and high detection rate. The tool was able to detect most of fake OGMs in Attacking Method 1 with different Originator Intervals, and most of fake OGMs in Attacking Method 2 when Originator Interval is 500 ms. In those cases, route manipulation attack is more effective in role of redirecting the routes of target nodes, as presented in the results of Figure 3.5.

3.6.2 Results for Scenario 2

Figure 3.11 presents the attack detection results for the mesh nodes of Figure 3.6 when attackers A_4 and A_9 broadcast phony OGMs with Originator Interval of 1000 ms. In Figure 3.11, we can see nodes O_1 and O_{10} have the highest rate of detected inconsistencies in their Sequence Number trees. We found that 26% of Sequence Numbers broadcasted for Originator O_1 are malicious, i.e., they are invalid, and 31% of Sequence Numbers

broadcasted for Originator O_{10} are malicious. These detection rates are close to the detection rate of Originator O_1 in Scenario 1 shown in Figure 3.10.

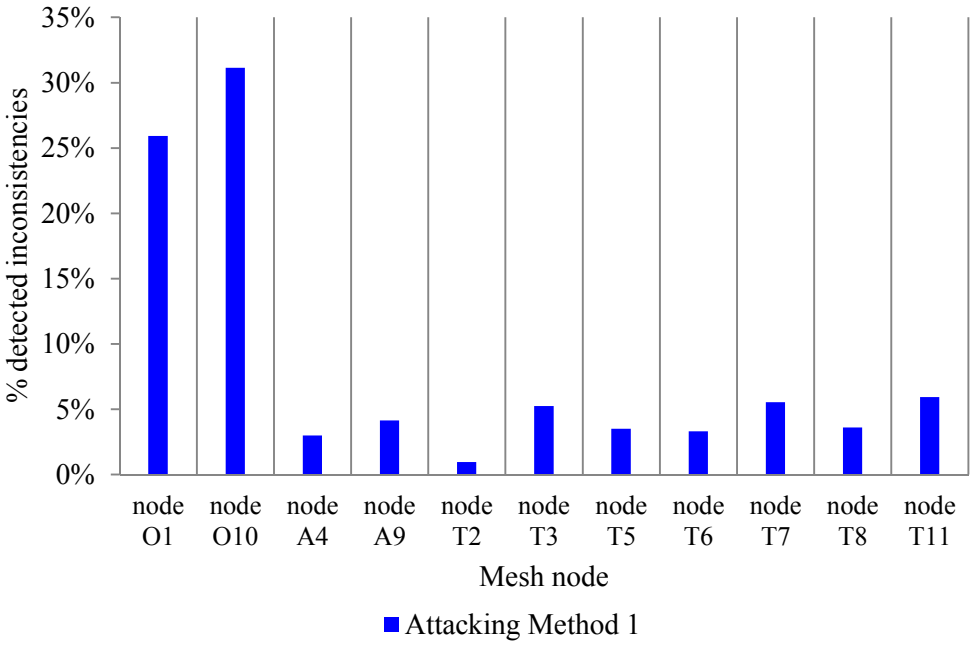


Figure 3.11: Detection results for Scenario 2.

Concerning the other nodes in Figure 3.11, we can notice that the rate of detected inconsistencies is below 6%. As explained on Section 3.6.1, these inconsistencies do not correspond to attack attempts, i.e., they are false positives that are caused by the lack of synchronization of routing information contained in the log files collected at each node.

The modified *batctl* tool was able to detect the majority of fake OGMs broadcasted by malicious node A_4 for node O_1 and the fake OGMs broadcasted by malicious node A_9 for node O_{10} in Scenario 2. In this scenario, the route manipulation attack poses a serious threat to the routes of target nodes since the attacker redirects these routes to the malicious node without great difficulty as showed in the results of Figure 3.7.

3.7 Conclusion

In this chapter, we have studied how an attacker efficiently manipulates the routes of BATMAN nodes. The attack consists of generating and broadcasting forged OGMs for a

legitimate mesh node in order to divert the routes of other mesh nodes to the malicious node. We show that this particular routing attack is very effective against BATMAN routing protocol. The detailed security analysis of attack feasibility establishes bounds on the real impact of this routing attack in a realistic mesh network scenario.

The experiment results demonstrate that routing operations of mesh network and routing tables of target nodes can be seriously impacted by the flooding of forged OGMs throughout the network. In addition, BATMAN protocol has proved to be vulnerable to route manipulation attack. However, this type of attack can be detected by tracing all the OGMs broadcasted by each node in the network making use of BATMAN log files collected for each node. To find the source of attack we rely on the mesh network topology. Our detection approach to defend against routing manipulation attacks combines monitoring-based detection and trace-based analysis techniques.

However, two attackers can collude to perform a coordinated routing manipulation attack. For instance, in Scenario 1 shown in Figure 3.4, if a second attacker is placed at node T_4 which coordinates with the first attacker A_2 to drop the fake OGMs broadcasted by attacker A_2 at node T_4, then the detection scheme, that basically traces malicious OGMs broadcasted to target nodes, would be not able to identify such type of attack using the Sequence Number tree structure. To detect such type of collusion attack, it is more appropriate to use a distributed intrusion detection mechanism which independently monitors the routing behavior of each node. The cooperative intrusion detection architecture introduced in Chapter 4 is capable of detecting such collusion attacks.

Chapter 4

Distributed and Cooperative Intrusion Detection

Many security schemes proposed for wired networks cannot be directly applied in the wireless environment. Since existing IDSs approaches are not suitable for WMNs, in this chapter, we propose an efficient and scalable IDS architecture to build distributed and collaborative intrusion detection for WMN architectures.

We focus on addressing a severe security threat against the correct operation of mesh network, known as routing misbehavior intrusion. The intrusion detection architecture is designed to effectively detect routing attacks in real-time. The IDS utilizes a practical and scalable architecture for collecting sufficient evidence in order to precisely diagnose routing attacks. In the intrusion detection architecture, cooperative detectors are placed at each node to promiscuously monitor all routing messages on the node interface, and exchange local routing events with one-hop neighboring nodes for collaborative analysis of the neighborhood routing behavior. Then, a tailored intrusion detection engine processes these generated routing events in conjunction with predefined Routing Constraints to concurrently calculate misbehaving metrics that indicate the presence of malicious routing behavior in the local vicinity. If suspected routing behavior is identified a consensus mechanism is initiated at the node, which analyzes detection results in the neighborhood and make a collective decision about the culpability of suspicious node.

The cooperative IDS proposed in this chapter has fewer assumptions and greatly improves the performance and scalability of realistic IDS executing in the WMN. We demonstrate both routing attacks and our distributed and cooperative intrusion detection mechanism employing BATMAN proactive routing.

The rest of the chapter is organized as follows. Sections 4.1 and 4.2 present the system model and attacker model respectively. Section 4.3 introduces our distributed intrusion

detection architecture. Section 4.4 proposes a threshold mechanism that improves the detection accuracy. Section 4.5 exemplifies the distributed intrusion detection scheme for detection of message fabrication attacks. In Section 4.6, we demonstrate the effectiveness of our intrusion detection approach through implementation. In Section 4.7, we analyze the performance of the intrusion detection solution in terms of resources consumption. In Section 4.8, we present additional considerations regarding the intrusion detection architecture, such as security issues. Finally, Section 4.9 concludes the chapter.

4.1 System Model

We adopt the same WMN model as described in Section 3.1. We assume that wireless channels are symmetric, and all mesh nodes have the same transmitting power and consequently the same transmission range. In particular, for performing distributed and cooperative intrusion detection, nodes taking part in WMN have powerful enough CPU and sufficient memory resources to run the IDS tool.

We assume that each IDS is trusted and is not corrupted, despite the node itself is compromised. Thus, the IDS do not provide fake or erroneous information, such as incorrect detection results, where attackers may collude to falsely accuse a legitimate node as malicious. We also assume that communication among IDSs is secure and reliable, so exchange of routing events and detection results cannot be compromised by outsider attackers. We assume the existence of end-to-end message authentication scheme, such as traditional digital signature or message authentication code (MAC), which allows each IDS to efficiently verify the integrity and authenticity of messages exchanged between IDSs. We assume that cryptographic protection prevents spoofing attacks and external attacks. As long as the node identity and its public key are cryptographically bound, the attacker cannot spoof the node's address unless the private key of the victim is compromised.

We assume that each IDS registers the MAC and IP address of the mesh node which it resides on and also the MAC and IP addresses from the neighboring nodes, and they are authentic. We do not consider attacks against the IDS itself, such as DoS attacks, or against the IDS communication mechanism, which is used to share detection results and traffic information, such message dropping attacks. We assume IDS instances always exchange data successfully, and are not compromised. Protecting the IDS is out of our work scope.

4.2 Attacker Model

We consider the same attacker model as described in Section 3.2. The attacker has full control of the authenticated node and performs arbitrary behavior to disrupt the communication of any node in the mesh network, such as injecting or discarding packets at the mesh node. As described in Section 3.2, we assume that the attacker do not attempt to violate the security properties of physical layer and MAC layer. In particular, the main objective of attacker is: (i) violate the integrity of routing tables and routing messages; and (ii) disrupt the routing and data forwarding operations of the network.

In this chapter, we apply our distributed intrusion detection solution for detecting message fabrication attacks, which is a common threat to WMNs. Message fabrication attacking method was detailed in Section 3.2.1. We define message fabrication attack as the process of creating and injecting legitimate routing packets into the network in order to violate the network routing services. The consequences of this attack are link spoofing (or route redirection), poisoning of routing tables, and routing loops.

The attacker can also launch specific attacks by applying message fabrication attacking method, such as routing manipulation attack which was studied in Chapter 3. In addition, this type of attack is a precondition for other more serious attacks, for instance, blackhole (or sinkhole attacks) and grayhole attacks, where data packets are indiscriminately discarded by the malicious node that previously redirected the network traffic to itself.

4.3 Distributed Intrusion Detection Architecture

An IDS instance resides on each mesh node and monitors all network traffic that passes through it. The IDS performs independent traffic analysis based on its own observation, and also collaborates with one-hop away mesh nodes. Since mesh nodes are the unique source of data in the network, all nodes have to contribute to the process of intrusion detection by: (i) carrying out local monitoring; (ii) analyzing the collected traffic; and (iii) providing valuable traffic information and detection results to other nodes when needed. The proposed intrusion detection architecture is illustrated in Figure 4.1.

Packets sent and received by a node are captured and filtered by a monitoring tool for further analysis. To that end, we employ Bro [10], a popular network IDS tool, which

monitors local traffic by sniffing the node's network interface. We opted for Bro since it provides a modular architecture that allows adding protocol analyzers. In addition, Bro provides a specialized language for writing policy scripts. Taking this into consideration, we implemented the Routing Protocol Analyzer (RPA) module at "detection level" of the IDS architecture, and policy scripts for intrusion detection in the Distributed Intrusion Detection Engine (DIDE) component, which is composed of different intrusion detection modules. We also added a specific packet filter into Bro main analyzer script in order to select all routing packets we are interested in when monitoring the traffic.

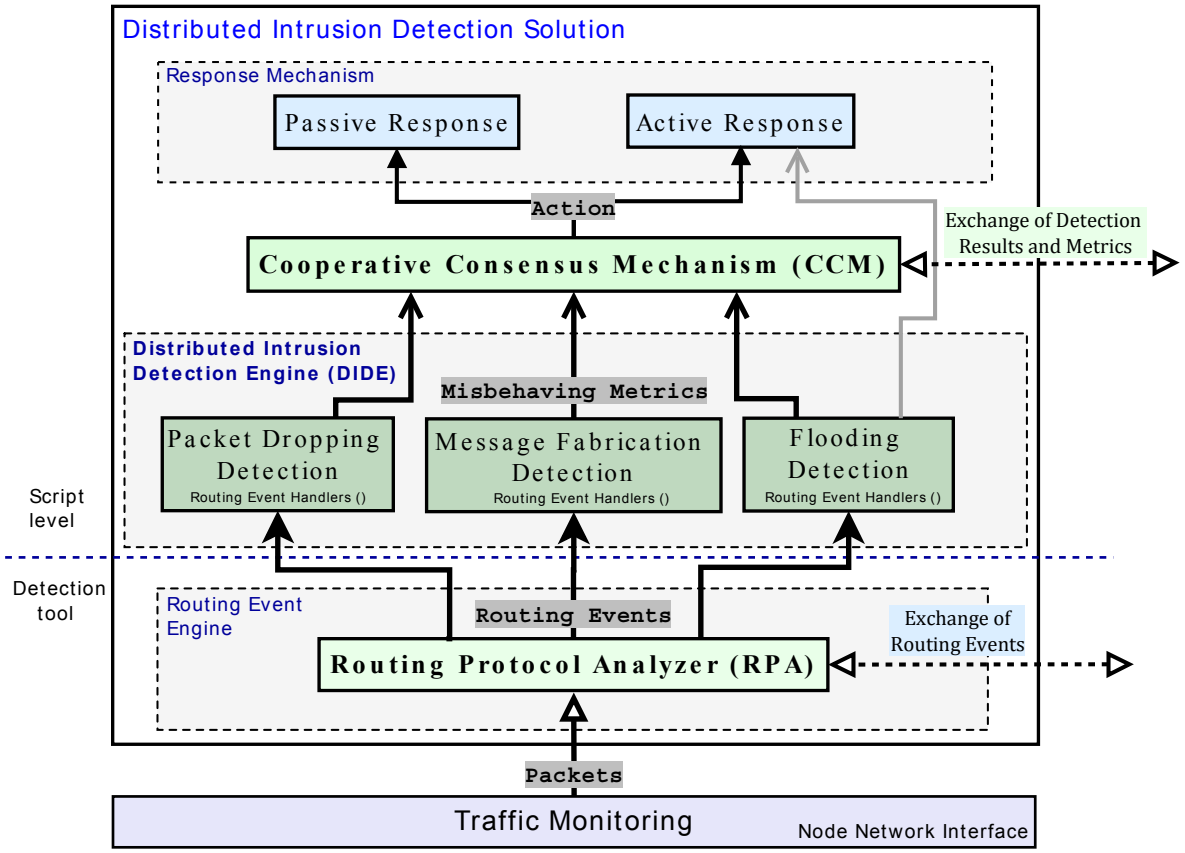


Figure 4.1: Distributed intrusion detection architecture.

4.3.1 Bro IDS

Bro is a NIDS that passively sniffs traffic on the network link and performs protocol analysis. Bro does packet capturing, message filtering, and content inspection, and invokes user-specified policy scripts, which contain event handlers that treat protocol events defined

by Bro. For high performance, Bro relies on the use of *libpcap* packet filters that captures only a subset of the traffic that transits the link it monitors, i.e., IP traffic.

Bro is divided into two layers: an "event engine" layer that transforms a stream of (filtered) packets into a stream of higher-level network events, and a "policy script interpreter" layer, which executes security policy scripts written in Bro specific language. Bro provides a distinct separation between the monitoring mechanism and policy script execution. The policy scripts specify event handlers, which are in charge of processing network events. A network event is a message containing arguments, which is generated by a general or specific protocol analyzer of "event engine" as a result of processing one or more protocol packets, mainly IP packets.

Bro provides a complete programming language. Bro policy scripts contain rules that describe what types of network activities are considered problematic. Policy scripts examine the network activity and can initiate specific actions based on that analysis. Although Bro language takes time and effort to learn, once mastered, the end-user is able to implement or change a variety of Bro scripts in order to detect and alert any sort of network activity. In addition, Bro language enables implementing elaborated policy scripts with timers, data structures, e.g., tables and sets, and connections.

Bro communication mechanism allows independent Bro instances to stay connected in order to exchange network events containing specific information and share common script variables between Bro peers, i.e., synchronize variables. Bro communication system uses typical TCP socket client-server model, which establishes a TCP connection between two hosts for sending and receiving packets. Bro uses a transparent communication model, where Bro instances do not differentiate between local and remote network events. In addition, Bro supports Secure Sockets Layer (SSL) cryptographic protocol for encryption and authentication of connections in order to secure the communication channel between two Bro peers and avoid external attacks, such as spoofing attacks and man in the middle attacks. However, the use of SSL for node communication incurs extra processing overheard to mesh nodes which has constrained hardware resources.

4.3.2 Routing Protocol Analyzer

In order to detect routing misbehavior, the IDS inspects every routing message transmitted

on the node interface. In the Routing Event Engine component, the RPA module examines all incoming and outgoing routing traffic on the node interface and generates respective Routing Events. The output of RPA module is a stream of Routing Events that describes the observed activities of the routing protocol in semantically rich, high-level terms. The Routing Events themselves do not represent security alerts, but rather provide valuable input for further processing by distributed intrusion detection algorithms.

A Routing Event is a message containing parameters as the result of analysis of one or more routing packets. The RPA module can generate different sorts of Routing Events according to the protocol behavior and the type of network activity it desires to signalize. The Routing Events defined for BATMAN protocol are specified in Section 4.5.1.

Since IDS instances collaborate with immediate neighbors to cooperate, the RPA module forwards all the node's Routing Events to its neighboring nodes, which in turn will also process these Routing Events which represent the routing information generated by the node. In order to exchange Routing Event messages between nodes, we employ Bro communication system, introduced in Section 4.3.1. We assume that Routing Event messages exchanged by the RPA modules of nodes are not violated by attackers, i.e., modified or dropped by malicious nodes. More concerns about the IDS security and vulnerabilities of the communication mechanism are discussed in Section 4.8.1.

The IDS approach only requires RPA modules to exchange Routing Event messages with neighboring RPA modules within one-hop maximum so that communication and computation overhead can be reduced for the nodes. Exchanging Routing Events instead of exchanging local audit data among neighboring nodes achieves cooperativeness with less communication overhead and processing load for the mesh nodes.

The constant exchange of Routing Events between nodes guarantees a reliable and robust transmission of routing information between IDSs. Because if messages are lost due to impairments on the wireless link, still the IDS is able to accurately analyze the routing behavior of neighboring nodes by making use of the Routing Events received so far, and calculate the respective Misbehaving Metrics with good precision.

As next step, Routing Events are processed by the distributed intrusion detection mechanism of IDS at "script level", i.e., the DIDE component. The architecture of DIDE component is introduced in next section.

4.3.3 Distributed Intrusion Detection Engine

The Distributed Intrusion Detection Engine (DIDE) component is implemented at "script level" of IDS architecture as policy scripts. DIDE component is composed of different intrusion detection modules, as shown in Figure 4.1. In DIDE component, each intrusion detection module treats the Routing Events generated by the node and also the Routing Events received from neighboring nodes for performing cooperative attack detection. Routing Events received from neighboring nodes are forwarded to individual intrusion detection modules of DIDE component by RPA module for subsequent treatment. Routing Event Handlers are defined at each intrusion detection module in order to process the respective Routing Events, i.e., each type of Routing Event is bound to a particular Routing Event Handler. The output of DIDE component is Misbehaving Metrics, which indicate the existence of malicious routing behavior in the own node or in the communication range of the neighborhood. However, each intrusion detection module calculates individual Misbehaving Metrics independently, as seen in Figure 4.1.

The attack detection algorithms, i.e., policy scripts, are integrated into DIDE component. An attack detection algorithm is essentially made up of Routing Event Handlers that specify what to do whenever a given Routing Event occurs. We make use of Bro's customized scripting language for implementing all the attack detection algorithms. Therefore, each intrusion detection module possess a set of attack detection algorithms that identify various types of attacks according to the attacking method of adversarial model, introduced in Section 3.2.1, which is used by the attacker to carry out the intrusions.

The DIDE component architecture is divided into three intrusion detection modules: (i) Packet Dropping Detection module, that is in charge of detecting malicious packet dropping behavior of misbehaving nodes; (ii) Message Fabrication Detection module, which detects forged routing packets in the routing flow; and (iii) Flooding Detection Module, that identifies fabricated messages flooded through the entire network.

For instance, the Packet Dropping Detection module, which is studied in Chapter 5, make it possible the detection of a wide range of packet dropping attack variations, such as selective/conditional packet dropping, probabilistic packet dropping, random packet dropping, and periodic packet dropping. The Packet Dropping Detection module processes local and remote Routing Events by making use of specific Routing Event Handlers, which

in turn implement algorithms for detection of packet dropping misbehavior. These algorithms compute the Misbehaving Metrics, i.e., Packet Dropping Metrics, which serve as input to the Cooperative Consensus Mechanism (CCM) module.

The Message Fabrication Detection module is better explained in Sections 4.5.2 and 4.5.3. In Section 4.5.2, we define Routing Constraints on the routing protocol behavior that are used for the message fabrication detection algorithms. In Section 4.5.3, we show how Misbehaving Metrics are calculated taking into account the defined Routing Constraints and Routing Events generated by RPA module. We make use of BATMAN routing protocol to exemplify the operation of Message Fabrication Detection module.

The Flooding Detection module implements intrusion detection algorithms for recognizing, for instance, an overflow of routing packets, which may contain legitimate content or not, for instance, a forged Source Address. The detection algorithms used in this module are close to the ones of Message Fabrication Detection module, i.e., the Routing Constraints employed by both modules are similar. However, if the Misbehaving Metrics calculated by this module exceeds a certain value in a short time period, an Active Response is promptly requested to the Response Mechanism component without previous analysis of the Misbehaving Metrics by CCM module, which can also trigger a response.

As discussed in Section 4.3.2, the active exchange of Routing Events between IDSs provides a fault tolerant intrusion detection solution that is capable of maintaining the detection accuracy at a good level still in the presence of communication failures, such as corruption and discarding of routing packets at the endpoints. Fault tolerance assumes that failures are random and are the result of faults introduced in the communication equipment, which will cause errors in the transmission of packets between hosts. However, failures can also be caused by external factors such as noisy lines and collisions on shared medium.

4.3.4 Cooperative Consensus Mechanism

We employ a collaborative decision-making [40] (also known as group decision-making) mechanism, where each node takes active part in giving the verdict of the intrusion detection process. According to the concept of synergy, decisions made collectively tend to be more effective than decisions made by a single individual. Collaborative decision-making systems are more reliable in case of failure of nodes or if particular nodes are

compromised. If all nodes contribute to the outcome, a few malicious or malfunctioning nodes cannot easily disrupt the decision-making process. However, if a single node is in charge of making the decision, it can subsequently trigger, for instance, active actions in response to a given attack that will affect the entire network operation and can make nodes vulnerable to internal attacks in case of wrong decision made.

We particularly employ a consensus decision-making [40] scheme, which is a group decision-making process. Consensus decision-making seeks the agreement of the majority of participants on a given course of action, but with the consent of the minority on this course of action since the decision made should meet the concerns of all group members as much as possible and it is considered the best possible decision for the group and all of its members. As *decision rule*, i.e., the level of agreement demanded to finalize a decision, we use "unanimous agreement minus one vote or two votes" approach. This *decision rule* is applied during the *consensus process*, which collects as much agreement as possible but allows the decision to be finalized without requiring unanimity, i.e., the consensus is reached without the consent of all members involved in the decision process. In this case, if some members do not completely agree or have a strong objection to the choice made, they will have to accept the decision and live with it. We do not consider modifying the decision of the majority to generate unanimous agreement or unanimous consent.

In realistic WMNs, all nodes in the neighborhood should detect the same routing intrusions with equally accuracy, then reaching consensual decisions about those intrusion attempts with unanimous agreement. However, communication links of a few nodes may have poor quality due to packet collision and congested channels or because these nodes are very distant from the other nodes, then resulting in inconsistent intrusion detection results. Consequently, these nodes will oppose to the decision of the majority. Therefore, to achieve a consensus in such wireless environment with network fluctuations, we employ the *decision rule*, which does not demand unanimous consent from all of the nodes but the agreement from the majority. This consensus approach is considered fair for WMNs since it is highly improbable to reach unanimous consent in such unstable wireless scenario, and the final decision satisfy the expectation of all nodes, i.e., the legitimate nodes. Nonetheless, if the *consensus process* of various nodes in the neighborhood, i.e., the majority of nodes, is compromised, an attacker can induce the legitimate nodes to make

erroneous decisions, then violating the consensus approach. The security of the consensus scheme is discussed in Section 4.8.1.

The *consensus process* together with the *decision rule* is implemented in CCM module as the Consensus Algorithm. The CCM module is integrated to the IDS architecture at "script level" as seen in Figure 4.1. The Consensus Algorithm is implemented in CCM module in the form of policy scripts. The Consensus Algorithm is detailed as follow.

- The CCM module analyzes Misbehaving Metrics supplied by the different intrusion detection modules to eventually diagnose if a routing attack is occurring relying on a threshold scheme. For example, Packet Dropping Metrics provided by the Packet Dropping Detection module, which represent, for instance, the rate of routing packets received by a node that are intentionally discarded, are carefully analyzed by CCM module to identify possible packet dropping attack attempts.
- If strong proof of malicious routing behavior is detected by CMM module, then it consults neighboring CCM modules for sharing Detection Results, i.e., the Misbehaving Metrics related to the alleged routing intrusion.
- Based on the analysis of the Misbehaving Metrics from the own node and neighboring nodes using thresholds, each node makes its individual decision about the routing intrusion and designates or do not a suspected node as malicious.
- Thus, the CCM module shares the decisions among neighboring CCM modules, where each node designates a suspicious node or do not designated, if the node finds any evidence of attack occurrence when analyzing the Misbehaving Metrics.
- If all nodes agree on a common suspected node, then unanimous agreement is reached and the final decision is made on the inculpation of the malicious node.
- If any node disagrees with the decision of majority, then CCM module applies the *decision rule* to achieve consensus, i.e., finalize the decision about the suspicious node but without unanimous consent. In that case, there are two possibilities:
  (i) The majority of nodes make a joint decision on the same node, and then a consensus is reached on the guiltiness of the suspected node.
  (ii) The majority of nodes do not designate any suspicious node, and then a consensus is also reached, which means no routing attack is occurring in the

neighborhood and there is no malicious node.

In both cases (i) and (ii), it is likely that a few nodes are suffering from impairments on their wireless communication links or have reduced transmission power, which lead these nodes to make such divergent decisions in relation to the majority.

- If half of nodes decide upon the same sentence for a suspected node and the other half does not point out any suspicious node, there is no consensus among the participating nodes and no final decision is made. In this particular case, the Consensus Algorithm evokes a passive action from the Response Mechanism to request further investigation.

The Consensus Algorithm applies adjustable thresholds, which are based upon the link quality, in order to improve the analysis of Misbehaving Metrics and prevent nodes from making incorrect decisions on the detection of malicious routing traffic and malicious nodes. As explained earlier, inconsistent decisions are mainly caused by degradations in the node's communication channel, such as packet loss and packet corruption. Thus, incorrect decisions made by many nodes will surely corrupt the final decision. The threshold approach is further explained in Section 4.4. If the consensus reached by the nodes on the routing attack and malicious node is positive, i.e., the malicious node is properly identified, the CCM module sends a solicitation of passive and/or active reaction to the Response Mechanism, which in turn it will raise alerts to users and authorities and/or execute reactions at the network level, for instance, drop the malicious packets.

The CCM module allows IDS instances to effectively exchange necessary Intrusion Detection Results for accurate analysis of attack information. Communication between nodes follows a simple message-based protocol. For example, the consensus messages used to exchange Misbehaving Metrics and make collective decisions are transmitted by Bro communication system, presented in Section 4.3.1. Indeed, Detection Results and decisions are shared between nodes by using Bro variable synchronization mechanism, where variables of different policy scripts are synchronized across the connected nodes. Then, each modification a nodes does to a given variable, which is declared as synchronizable, it is propagated to the same variable in the scripts of neighbors, which likewise defined this variable as synchronizable. This synchronization scheme provides a global view of

detection results and individual decisions for all the nodes. Thus, nodes can share knowledge about local Misbehaving Metrics, which are calculated by individual intrusion detection modules, and decisions about suspicions nodes with all of the neighbors.

CCM module is exemplified in Sections 4.5.4 where the output of Message Fabrication Detection module, i.e., the Message Fabrication Metrics, is used as input for the Consensus Algorithm in order to reach a collective decision on the presumed misbehaving node. We consider some assumptions for secure message exchange. We assume that Intrusion Detection Results and decision messages exchanged by CCM modules during the *consensus process* are not compromised, i.e., modified or dropped by malicious nodes. The security of IDS communication mechanism is further discussed in Section 4.8.1.

The consensus approach is considered vulnerable to collusion attacks, where the majority of nodes are malicious in a neighborhood. So colluding malicious nodes can incriminate a legitimate node in the same neighborhood, via the consensus protocol, by violating the content of consensus messages, e.g., Misbehaving Metrics and decisions made, that are exchanged among nodes. Therefore, we assume that the majority of nodes in a neighborhood are not compromised. This assumption is mostly reasonable since in realist WMN scenarios the number of malicious nodes is generally reduced to a few nodes, which would not interfere in the consensus of the group. Nevertheless, the use of a trust scheme, which defines the level of trust for each node, can effectively defeat this type of attack.

4.3.5 Response Mechanism

If a routing intrusion is consensually detected by the nodes, a Passive Response and/or Active Response is requested by the CCM module according to the response policy defined for this attacking technique. The Response Mechanism specifies response policies taking into account the attacking methods of adversarial model, described in Section 3.2.1, and more specifically, the type of routing attack that is identified. For instance, selfishness misbehavior, which is an attacking technique based on packet dropping attacking method, would require an passive action since the damage to the routing service and data delivery service is tolerable and the network administrator has enough time to investigate the attack cautiously. However, for blackhole attacks, which is also a packet dropping attacking technique, an active action is demanded because the attack effect on the data delivery

service is devastating having serious consequences to the network users, therefore an mitigation action should be rapidly applied.

- Passive Response: In this procedure, the IDS firstly outputs all the intrusion information to log files, e.g., the Misbehaving Metrics and decisions made by the nodes, which are subsequently stored on the disk. Then, the IDS raises an alarm, according to the *notification policy*, to inform users or network administrators about the detected attack. For instance, emails are sent to a preconfigured list of email addresses. The passive Response Mechanism is implemented as policy scripts and it makes use of Bro notification system to send such alerts.
- Active Response: The IDS tries to mitigate the impact of the attack on the mesh network or try to deter the attack that is occurring in the compromised node. Active reactions are executed taking into account the attacking method and attacking technique utilized by the attacker. For example, if the adversary is employing message fabrication or message flooding attacking methods, the IDS tries to mitigate the attack by discarding all the phony routing messages generated by the malicious node. This attack response can be carried out in the malicious node, which is more efficient, or in the target nodes, which also works. In case of data packet dropping attacking technique, the IDS tries to avoid data packets to pass through the malicious node toward their destinations. This attack response can be achieved by dropping the routing packets broadcasted by the malicious node, so the target nodes will choose optional routes to send data packets toward the destination.

The Response Mechanism of the IDS implements a *notification policy* which checks the level of dangerousness of each detected routing intrusion, i.e., the priority of the malicious routing behavior in the notification list, and reports such intrusion to the entities that might be interested in it, such as end-users or any person responsible for the network maintenance. For example, if selfishness misbehavior is detected an alert is sent only to the network administrator, however if selective packet dropping is detected at the node both the node's user and the network administrator are notified since the user should be alerted that the administrator will soon apply an action on the node. We assume the IDS does not

execute active responses by itself, e.g., trying to mitigate or stop an attack in progress. We assume the network administrator is in a better position to formulate the most appropriate corrective action to minimize or eliminate the impact of the attack on the mesh network.

4.4 Threshold Approach

We developed a threshold-based approach to improve the investigation of the Misbehaving Metrics at CCM module so the IDS can achieve higher detection rate and lower false alarm rate, i.e., false positives, then enhancing the IDS detection accuracy. The threshold approach is applied to distinguish between false positives, which are mainly caused by degradations in the wireless link quality, and true positives, that are the malicious behavior produced by attackers, when analyzing the Misbehaving Metrics. For instance, the IDS cannot determine if routing packets are being dropped because of natural packet loss of link or malicious routing behavior performed by a misbehaving node.

Since we define a border separating detection rate from false positive rate, the false negatives are automatically eliminated and the range of false positives is delimited and can be appropriately measured. In our intrusion detection approach, false negatives could only happen in case of Routing Event messages are accidentally discarded by a faulty node and the corresponding routing messages are: (i) dropped by the malicious node, then the IDS at the node and its neighbors are unable to detect these routing intrusions; or (i) fabricated by the malicious node, then neighboring IDSs of the node are unable to detect these routing intrusions. The IDS does not fail to detect any type of malicious behavior based on the adversarial model defined in Section 3.2.1, because mostly Routing Events, which signalize such type of intrusions, will be properly processed by the distributed intrusion detection modules of DIDE component, and we assume nodes are not faulty. Moreover, since we assume that Routing Events messages exchanged between IDSs are not compromised by attackers, false negatives will not occur during the intrusion detection process.

In addition, the threshold mechanism is used by the Consensus Algorithm in order to help CCM module of individual nodes to make more precise decisions about routing intrusions and conviction of suspected nodes. As explained in Section 4.3.4, a few nodes can be misguided in making decisions on routing attacks and suspicious nodes due to degradations in their wireless transmissions, consequently a well-behaved node can be

unfairly accused if several nodes in the vicinity, which have impaired links, make such inconsistent decisions. Therefore, the threshold scheme assures the majority of nodes will make the correct decision about a detected intrusion and the corresponding dubious node.

The threshold mechanism takes into consideration the rate of packet loss of communication links between nodes to define threshold values. The threshold approach, which quantifies the false positive rate, is explained as follow.

1. For a given mesh network deployed in the open environment, we assume all nodes are well behaving and do not perform malicious routing behavior. Due to the intrinsic characteristics of the network, such as noisy areas, some communication links will undergo quality impairments, such random packet loss or packet corruption (if checksum is not used).
2. Based on that, some nodes, which are impacted by these transmission limitations, will be forced to calculate their respective Misbehaving Metrics since Routing Events are unintentionally dropped or corrupted during transmission. These calculated Misbehaving Metrics are in fact false positives since no attack is occurring in network.
3. Then, we calculate individual threshold values, considering a given period of time, for the respective Misbehaving Metrics computed by the nodes, e.g., the rate of routing packets received by the node from a neighbor that are discarded.
4. The threshold value, for a given Misbehaving Metric type, is the average number of violations of the related Routing Constraint within a period.
5. This procedure of threshold calculation is repeated when necessary, e.g., in case of changes in the network topology caused by mobility of nodes.

Defining a proper threshold is important since the primary measure of effective intrusion detection in the network is low false positives and no false negatives. Furthermore, the threshold mechanism makes possible the RPA module to tolerate a certain level of message loss, which is common in wireless networks, and it certainly affects the calculation of the Misbehaving Metrics.

The employment of thresholds along with the Consensus Algorithm is demonstrated in

Section 4.5.4. The procedure for calculating threshold values in the WMN scenario is presented in Section 4.6.2, where BATMAN is used as routing protocol.

4.5 Distributed Detection of Message Fabrication Attacks

We choose BATMAN as the routing protocol for the current investigation. In particular, we focus on specifying the routing behavior of the protocol by using Routing Events and Routing Constraints in order to calculate the respective Misbehaving Metrics. The attack detection algorithm consists of constraints on the specified routing behavior which is based upon the protocol routing messages described in the specification.

4.5.1 RPA: Routing Events

Routing packets transmitted on the node link are captured and analyzed by the RPA module. This component generates Routing Events based on the routing behavior of protocol, which are subsequently treated by Routing Event Handlers defined in the policy scripts of DIDE component.

We exemplify this mechanism in the context of BATMAN routing protocol, but the protocol analyzer can easily accommodate other routing protocols as well. BATMAN routing algorithm was detailed in Section 3.3.2. The RPA module handles BATMAN packets and generates Routing Events that represent the protocol semantics:

1) $OGM\_rcv(< params >)$: it means an OGM was received from one of the neighbors.
2) $OGM\_broad(< params >)$: it means the node broadcasted an OGM to its neighbors.
3) $OGM\_rebrd(< params >)$: it means the node rebroadcasted an OGM received from a neighboring node.

In addition, these Routing Events carry in parameters related to local context of node, such as the node's MAC address, and parameters extracted from the OGM packet such as Originator Address, Source Address, Previous Sender address, and Sequence Number

value. The validation of OGM parameters by the IDS prevents man in the middle attacks, where an attacker could sniffs and modifies the routing messages exchanged between two end-points. Then, the Routing Events are treated by the own node and sent to its neighboring nodes, which in turn will also treat the node's Routing Events.

The attack detection scripts of DIDE component, more specifically the Message Fabrication Detection module, implements Routing Event Handlers, which treat both local and remote Routing Events exchanged with the neighbors.

1. $EH\_OGM\_rcv()$: it extracts parameters from local and remote $OGM\_rcv(< params >)$ and stores the data in sets as follow.
   - Local $OGM\_rcv(< params >)$: $S\_rcv_{Orig}^{Src} = +Seqn$
     - $S\_rcv_{Orig}^{Src}$ is the set of Sequence Numbers $Seqn$ received by the node for the Originator Address $Orig$ from Source Address Src, which is a neighbor of the node.
   - Remote $OGM\_rcv(< params >)$: $NS\_rcv_{Orig}^{NSrc,PSrc} = +Seqn$
     - $NS\_rcv_{Orig}^{NSrc,PSrc}$ is the set of $Seqn$ received by the node's neighbor for $Orig$, where $NSrc$ is the neighbor address which forwarded the Routing Event to the node. And $PSrc$ is the Previous Sender address, which is a neighbor of $NSrc$, i.e., it is the neighbor of the neighbor of the node.
2. $EH\_OGM\_rebrd()$: it processes local and remote $OGM\_rebrd(< params >)$ and stores the data in sets as follow.
   - Local $OGM\_rebrd(< params >)$: $S\_rebrd_{Orig}^{Src} = +Seqn$
     - $S\_rebrd_{Orig}^{Src}$ is the set of $Seqn$ rebroadcasted by the node for $Orig$ that were previously received from neighbor Src.
   - Remote $OGM\_rebrd(< params >)$: $NS\_rebrd_{Orig}^{NSrc} = +Seqn$
     - $NS\_rebrd_{Orig}^{NSrc}$ is the set of $Seqn$ rebroadcasted by the node's neighbor $NSrc$ for $Orig$.

4.5.2 DIDE: Routing Constraints

We analyze BATMAN specification and defined attack detection constraints in order to prevent routing attacks which employ message fabrication attacking method. The Routing Constraints rely on the protocol behavior for detecting malicious routing behavior. They are implemented in the Routing Event Handlers defined in previous section for making use of their data sets.

1. $EH\_OGM\_rebrd(\,)$: Constraint $C_1 \leftarrow Seqn \in S\_rcv_{Orig}^{Src}$,
   - Check if the $Seqn$ rebroadcasted by the node for $Orig$ was received before from its neighbor Src.
2. $EH\_OGM\_rcv(\,)$: Constraint $C_2 \leftarrow Seqn \in NS\_rcv_{Orig}^{NSrc,PSrc}$,
   - Check if the $Seqn$ received by the node for $Orig$ was firstly received by its neighbor $NSrc$ which received it from its neighbor $PSrc$.

   $EH\_OGM\_rcv(\,)$: Constraint $C_3 \leftarrow Seqn \in NS\_rebrd_{Orig}^{NSrc}$,
   - Check if the $Seqn$ received by the node for $Orig$ was previously rebroadcasted by its neighboring node $NSrc$.

Additional Routing Constraints can be specified and added to Message Fabrication Detection module but respecting the routing protocol behavior and also taking into account the attacking techniques that IDS seeks to identify. In this chapter, we apply the Routing Constraints C_1, C_2, and C_3 for detecting message fabrication routing attacks.

4.5.3 DIDE: Misbehaving Metrics

In case of any Routing Constraint is violated, the message fabrication detection algorithm counts the number of inconsistencies F found for each Routing Constraint separately.

1. C_1: **if** $Seqn \notin S\_rcv_{Orig}^{Src}$ **then** F_{S_rcv}++ ,

   where F_{S_rcv} is the number of fake $Seqn$ rebroadcasted by the node for $Orig$, which are supposed to be firstly received from its neighbor Src but they didn't.
2. C_2: **if** $Seqn \notin NS\_rcv_{Orig}^{NSrc,PSrc}$ **then** F_{NS_rcv}++ ,

where F_{NS_rcv} is the number of fake $Seqn$ received by the node from its neighbor $NSrc$ for $Orig$, which are supposed to be received by $NSrc$ from its neighbor $PSrc$ but they didn't.

3. C_3: **if** $Seqn \notin NS\_rebrd_{Orig}^{NSrc}$ **then** F_{NS_rebrd}++ ,

   where F_{NS_rebrd} is the number of fake $Seqn$ received by the node for $Orig$, which are supposed to be rebroadcasted by its neighboring node $NSrc$ but they didn't.

Then, the message fabrication detection algorithm calculates the rates of routing misbehavior R considering the number of inconsistencies F detected, which are basically the Message Fabrication Metrics output by this module, as follow.

$$R_{S\_rcv} = \frac{F_{S\_rcv}}{|S\_rebrd_{Orig}^{Src}|} \quad (1)$$

where R_{S_rcv} is the rate of fake $Seqn$ rebroadcasted by the node for $Orig$.

$$R_{NS\_rcv} = \frac{F_{NS\_rcv}}{|S\_rcv_{Orig}^{Src}|} \quad (2)$$

where R_{NS_rcv} is the rate of fake $Seqn$ received by the node from its neighbor $NSrc$ for $Orig$ that were not received by $NSrc$ from $PSrc$.

$$R_{NS\_rebrd} = \frac{F_{NS\_rebrd}}{|S\_rcv_{Orig}^{Src}|} \quad (3)$$

where R_{SN_rebrd} is the rate of fake $Seqn$ received by the node for $Orig$ that were not rebroadcasted by its neighbor $NSrc$.

These Message Fabrication Metrics are provided to CCM module for further analysis. In addition, the Message Fabrication Detection module could provide additional Misbehaving Metrics if necessary since they are essentially calculated based on Routing Events processed by the Routing Event Handlers.

4.5.4 CCM: Consensus Algorithm

The nodes in the neighborhood collaborate with each other to make a decision about a suspicious node. This is performed by a consensus procedure, in which a node is considered malicious when the majority of its neighbors have reached a consensus as explained in Section 4.3.4. The reason for such procedure is that Misbehaving Metrics calculated by only one node may not be precise due to packet loss of wireless links caused by interferences and noisy channel. Then, a legitimate node transmitting on deficient links may be erroneously considered as a malicious node. Therefore, the consensus approach assures with a very high probability that malicious nodes will properly identified.

The Consensus Algorithm of CCM module is summarized in Algorithm 1. For a better understanding of the algorithm, we consider the rates of message fabrication $R_{S\_rcv}^{O_n}$, $R_{NS\_rcv}^{O_n}$, and $R_{NS\_rebrd}^{O_n}$ calculated by node O_n as the Misbehaving Metrics provided to CCM module for analysis. We assume the neighboring node N_1 of node O_n is advertising fake routing information in the neighborhood, i.e., broadcasting fabricated routing packets, so the Consensus Algorithm is initiated by node O_n based upon that premise. Since remote Routing Events exchanged by nodes can be lost, the calculated rates of routing misbehavior R are not accurate. Then, a threshold Th is necessary to distinguish between false positive rate and malicious routing behavior rate when analyzing the Misbehaving Metrics, as explained in Section 4.4. The thresholds $Th_{O_n} = \{Th_{O_n}^{S\_rcv}, Th_{O_n}^{NS\_rcv}, Th_{O_n}^{NS\_rebrd}\}$ are defined for node O_n, which are based on the packet loss rate of O_n's wireless link.

As the rate of routing misbehavior $R_{NS\_rcv}^{O_n} \geq Th_{O_n}^{NS\_rcv}$, node O_n suspects a routing attack is occurring in the vicinity, i.e., neighbor N_1 is fabricating packets, and then starts the consensus procedure by sending its Detection Results $DR_{O_n} = \{R_{S\_rcv}^{O_n}, R_{NS\_rcv}^{O_n}, R_{NS\_rebrd}^{O_n}\}$ and thresholds Th_{O_n} to the set of neighbors $P = \{N_1, \dots, N_k\}$ and receiving the respective Detection Results $DR_{N_1}, \dots, DR_{N_k}$ and thresholds $Th_{N_1}, \dots, Th_{N_k}$ from the neighboring nodes N_i, $1 \leq i \leq k$. Thus, node O_n analyses DR_{N_i} from the neighboring nodes N_i, particularly Detection Results DR_{N_1} from N_1 who is supposedly broadcasting fake routing packets, and decide to indicate neighbor N_1 as malicious node MN_{O_n}. Similarly, the neighbors N_i execute the same analysis procedure and indicate their malicious nodes MN_{N_i}

or not. So, node O_n exchange its individual decision MN_{O_n} with neighboring nodes N_i and reciprocally for the decision of the neighboring nodes MN_{N_i}.

Algorithm 1: Consensus Mechanism

Input: Detection Results DR_{O_n}, thresholds Th_{O_n}
Output: malicious node MN_{O_n}, $consensus$

1. **if** $R_{NS\_rcv}^{O_n} \geq Th_{O_n}^{NS\_rcv}$
2. **if** $consensus \neq done$ **then**
3. $Sync(DR_{O_n}, Th_{O_n})$
4. **for all** $N_i \in P$ **do**
5. $Sync(DR_{N_i}, Th_{N_i})$
6. **if all** $R_{NS\_rcv}^{N_i} \geq Th_{N_i}^{NS\_rcv}$ **and** $R_{S\_rcv}^{N_1} \geq Th_{N_1}^{S\_rcv}$ **then**
7. $MN_{O_n} \leftarrow N_1$
8. $Sync(MN_{O_n}, MN_{N_i})$
9. **if all** $MN_{N_i} = N_1$ **then**
10. $consensus \leftarrow unanimous\_agreement;\ MN_{O_n} \leftarrow N_1$
11. Raise($alert$); Execute($active\_response$)
12. **else if any** $MN_{N_i} \neq N_1$ **then**
13. Execute($decision\_rule$)
14. **if majority** $MN_{N_i} = N_1$ **then**
15. $consensus \leftarrow done;\ MN_{O_n} \leftarrow N_1$
16. Raise($alert$); Execute($active\_response$)
17. **else if majority** $MN_{N_i} = null$ **then**
18. $consensus \leftarrow done;\ MN_{O_n} \leftarrow null$
19. **else**
20. $consensus \leftarrow failed;\ MN_{O_n} \leftarrow null$
21. Raise($alert$)
22. **end if**
23. **end if**
24. **else**
25. $MN_{O_n} \leftarrow null$
26. **end if**
27. **end for**
28. **end if**
29. **end if**

If all neighboring nodes N_i indicate the same node as malicious, i.e., $MN_{O_n} = MN_{N_i}$, the final decision is made with unanimity, and an active reaction is requested to Response Mechanism. If any neighbor N_i indicates a different malicious node or does not indicate: $MN_{N_i} \neq MN_{O_n}$, the *decision rule* is executed. In that case, there are three options: (i) the

majority of neighbors N_i indicate N_1 as malicious, then consensus is reached and an active reaction is similarly solicited from Response Mechanism; (ii) the majority of neighbors N_i does not indicate any malicious node, then consensus is also reached and no response is required from Response Mechanism; and (iii) Nearly half of neighbors N_i designate N_1 as malicious node and the other part of neighbors N_i indicates no malicious node, then it means the consensus procedure has failed, but an alarm is triggered for further investigation since the consensus procedure has been possibly violated by the attacker.

Optionally, CCM module broadcasts intrusion alert messages to all nodes in the network so each node is aware of the routing attack and the respective malicious node N_1. Once the malicious node was detected, the IDS tool can react to the routing attack by discarding the malicious packets broadcasted by N_1 or depriving node N_1 of network resources, i.e., isolating node N_1 from the rest of the network. However, if the consensus mechanism of majority of neighbors N_i is compromised, node O_n may not be able to identify any malicious behavior performed by the neighbors N_i, e.g., in case the neighbors N_i send incorrect detection results or individual decisions. Apparently the routing behavior of these malicious nodes is considered normal from the point of view of node O_n.

To share Detection Results DR, thresholds Th, and decisions MN, the Consensus Algorithm utilizes Bro's synchronization mechanism explained in Section 4.3.4, where each time a synchronizable variable is updated in the script: $Sync(< var >)$, its modification is propagated to all neighbors that use this same variable and the inverse is true. In that manner, detection information can be easily shared between neighboring nodes.

4.6 Performance Evaluation

In this section, we use a virtualized mesh network environment to evaluate the efficiency of the distributed intrusion detection approach and demonstrate the feasibility of our solution. We apply the same virtualized network architecture used in Chapter 3 for emulating the mesh network and detect routing attacks.

4.6.1 Experiment Platform

The configuration of the virtualized mesh network platform is quite similar to that one

described in Section 3.5.1. The mesh topology is emulated by VMs, which consists of QEMU [37] instances, with Linux OS preinstalled, running BATMAN routing protocol [14]. *Batman-adv* v.2011.1.0 [36] is compiled from source code and loaded into Linux OS of each QEMU instance as kernel module. A QEMU instance consists of standard IBM PC 32Bit hardware architecture with one CPU, that is default QEMU 32Bit CPU type, and 256 MB of memory and compatible with Linux OS. We assume each node in the network has the hardware configuration equivalent to the one of the QEMU VM, for instance, a common wireless mesh device, precisely a Wi-Fi router, or a mobile device with enough hardware resources. For convenience, QEMU emulates different types of CPU and hardware configurations, including resource-constrained hardware platforms, which is very useful for performance evaluation of security solutions in a controlled environment.

The virtual nodes are interconnected by a virtual switch [38], which implements bi-directional communication links between nodes and the corresponding degradations in the quality of links including packet loss rate. For the experiments, we assume a fixed topology, but node mobility could be emulated as well. The distributed intrusion detection approach is implemented in Bro v.1.5.3 [42] according to the intrusion detection architecture described in Section 4.3 and the module of message fabrication attack detection detailed in Section 4.5. Each node has two configured interfaces for Linux OS, the first one is used by BATMAN to send routing and data packets and the second interface is used by the IDS for exchanging intrusion detection related messages.

During the experiment execution, we make use of Bro logging mechanism to collect the Misbehaving Metrics, routing information, and other intrusion detection data at each mesh node. Those data are saved to log files, which are synchronized with a global clock, i.e., the time of the main machine that executes the VMs.

4.6.2 Defining the Threshold

Before we start the experiments for intrusion detection, we execute the distributed intrusion detection approach in a mesh topology in order to calculate the thresholds Th for the rates of routing misbehavior R, as introduced in Section 4.4. Figure 4.2 presents the emulated mesh network used in this scenario, which is comprised of four nodes: O_1, O_2, O_3, and O_4. The link between nodes O_1 and O_2 has a packet loss rate of 15%, which is high. In this

experiment step, all nodes are well behaving, i.e., no node disseminates malicious routing messages. Then, we collect the supposed Misbehaving Metrics data calculated by nodes O_1 and O_2 for defining the thresholds $Th_{O_1}$and Th_{O_2}. The Misbehaving Metrics are caused by the loss of Routing Event messages exchanged between the nodes.

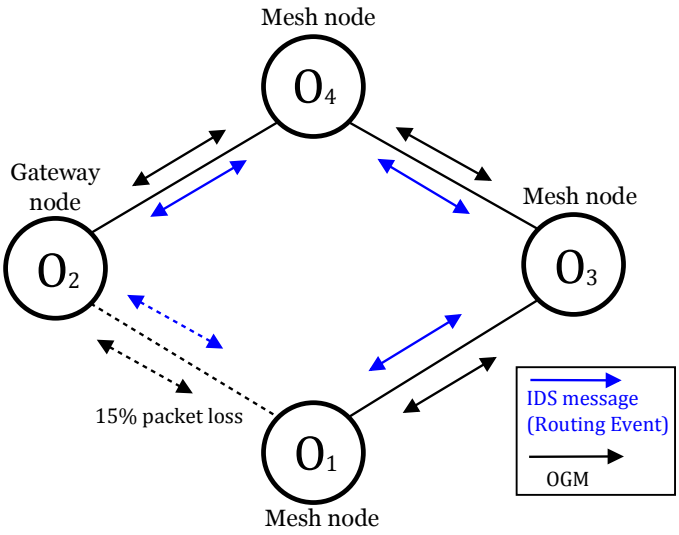


Figure 4.2: Mesh topology emulated for the experiment.

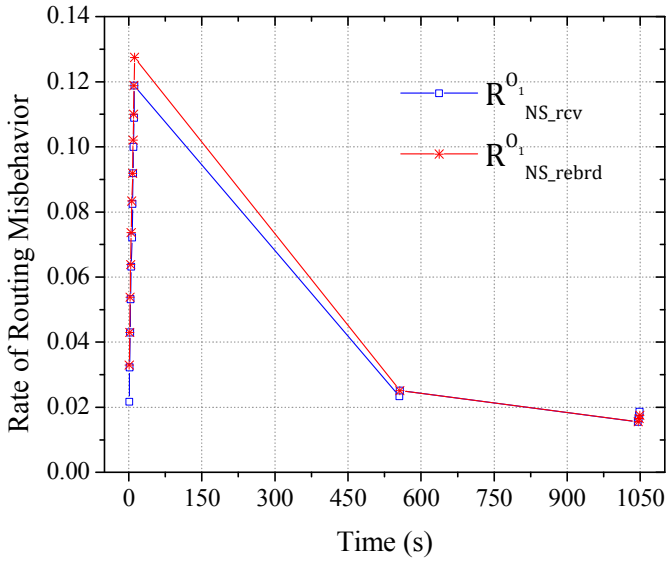


Figure 4.3: Misbehaving Metrics calculated by node O_1.

The duration of the experiment is 20 mins. Figure 4.3 shows the rates of routing misbehavior $R_{NS\_rcv}^{O_1}$, and $R_{NS\_rebrd}^{O_1}$ calculated by node O_1 during the experiment run,

which are actually false positives. We note that the maximum value of $R_{NS\_rcv}^{O_1}$ is close to 0.12 and for $R_{NS\_rebrd}^{O_1}$ is close to 0.13, and the minimum value of both rates $R_{NS\_rcv}^{O_1}$ and $R_{NS\_rebrd}^{O_1}$ is close to 0.02, therefore these two rates $R_{NS\_rcv}^{O_1}$ and $R_{NS\_rebrd}^{O_1}$ are affected in the same degree by the packet loss rate between nodes O_1 and O_2. The threshold for rate R_{NS_rcv} is defined as $Th^{NS\_rcv} = \frac{\sum F_{NS\_rcv}}{N}$, where N is total number of inconsistencies F_{NS_rcv} detected by the node. The threshold Th^{NS_rebrd} regarding the rate R_{NS_rebrd} is determined in the same way, i.e., by calculating the average F_{NS_rebrd}. Table 4.1 presents the thresholds calculated for nodes O_1, O_2, and O_3.

Table 4.1: Thresholds estimated for nodes O_1, O_2, and O_3.

Threshold	$\boldsymbol{O_1}$	$\boldsymbol{O_2}$	$\boldsymbol{O_3}$
Th^{S_rcv}	0.04	0.01	0.01
Th^{NS_rcv}	0.05	0.02	0.02
$\boldsymbol{Th^{NS\_rebrd}}$	0.06	0.03	0.03

Thus, applying the threshold formulas with node O_1's Misbehaving Metrics we find $Th_{O_1}^{NS\_rcv} = 0.05$ and $Th_{O_1}^{NS\_rebrd} = 0.06$. Since the loss of remote Routing Events does not have influence on $R_{S\_rcv}^{O_1}$, i.e., the packet loss rate does not impact $R_{S\_rcv}^{O_1}$, we fix $Th_{O_1}^{S\_rcv} = 0.04$ and $Th_{O_2}^{S\_rcv} = Th_{O_3}^{S\_rcv} = 0.01$. For node O_2, we found $Th_{O_2}^{NS\_rcv} = 0.02$ and $Th_{O_2}^{NS\_rebrd} = 0.03$ using the respective Misbehaving Metrics. The estimated thresholds can be applied to larger topologies since they rely exclusively on the pack loss rate between links. Therefore, even if the number of neighboring nodes increases and the amount of packets transmitted also increases but the packet loss rate undergoes small fluctuations, then the same calculated thresholds are applicable. Hence, the threshold-based approach is considered scalable with respect to the number of nodes in the vicinity.

4.6.3 Emulation of Message Fabrication Attack

In this scenario, node O_1 is compromised, as shown in Figure 4.4, and disseminates fake OGMs in the name of gateway node O_2, which in reality it did not receive, in order to

divert the route of target nodes O_3 and O_4 toward gateway node O_2 to the own malicious node O_1. As routing decisions are based on the statistical analysis of amount of OGMs received instead of the information contained in packets, the malicious node O_1 has to fabricate a number of fake OGMs that are numerous enough to surpass the legitimate OGMs of node O_2 in order to redirect the route of target nodes O_3 and O_4 to the malicious node. In other words, malicious node O_1 has to continuously win the neighbor ranking of target nodes O_3 and O_4 towards gateway node O_2, then overriding the legitimate OGMs broadcasted by gateway node O_2.

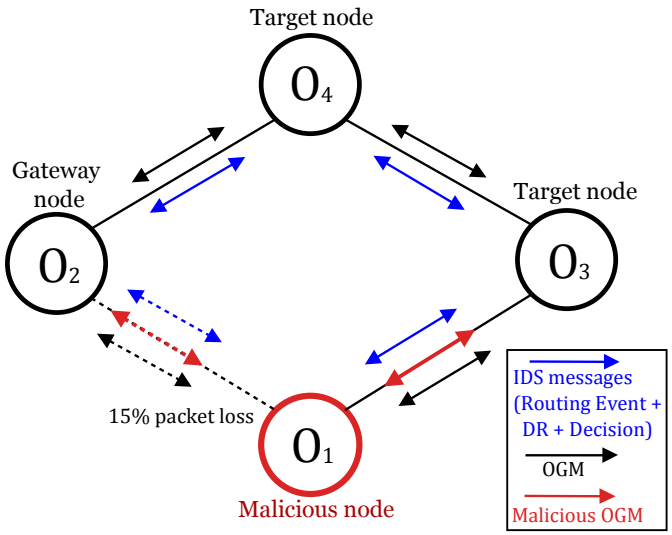


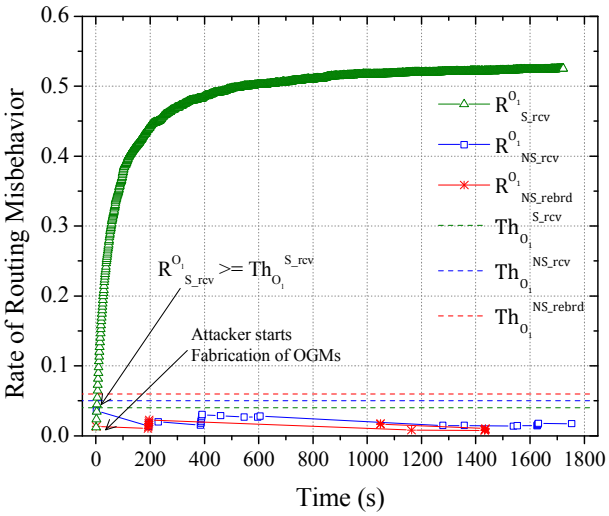
Figure 4.4: Routing redirection attack.

Since the link between nodes O_1 and O_2 has a packet loss rate (15%) higher than the one between nodes O_2 and O_4 (0%), target nodes O_3 and O_4 normally prefer a route toward gateway node O_2 via node O_4. For instance, in the routing table of target node O_3, the route entry of node O_2 has as best next-hop node O_4. However, as a consequence of the routing attack execution, target nodes O_3 and O_4 will choose malicious node O_1 as best next-hop to gateway node O_2 because malicious node O_1 has broadcasted the most number of OGMs for gateway node O_2. As a result, in the routing table of target nodes O_3 and O_4, the route entry of gateway node O_2 will be updated to node O_1 as next-hop.

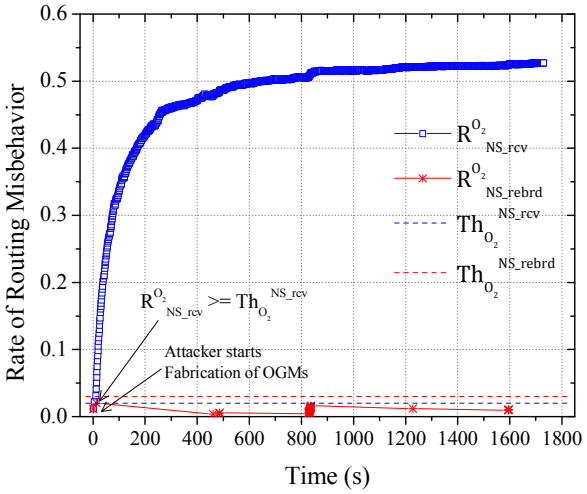
This routing manipulation attack, studied in Chapter 3, aims to disrupt the network routes by violating the integrity of routing tables of the target nodes. We implement this

routing attack by employing the packet generator packETH [39] as introduced in Section 3.5.2, which allow node O_1 fabricating and broadcasting fake OGMs with continuous valid $Seqn$ on the network interface of own node O_1.

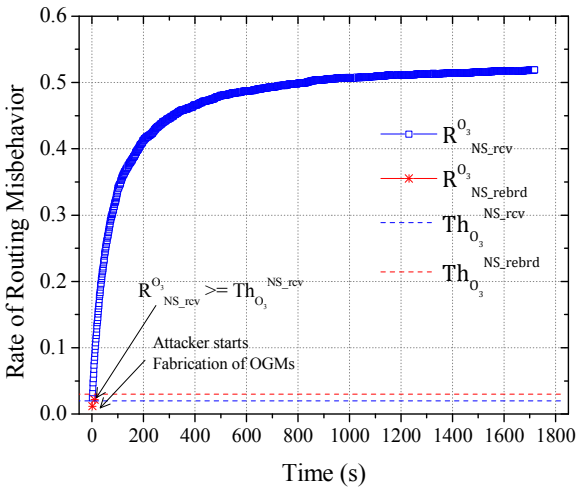
4.6.4 Results



(a) Rate of message fabrication calculated by node O_1.



(a) Rate of message fabrication calculated by node O_2.



(a) Rate of message fabrication calculated by node O_3.

Figure 4.5: Misbehaving Metrics calculated by nodes O_1, O_2, and O_3.

We carried out several experiments to evaluate the robustness of our distributed intrusion

detection approach. We take into consideration different broadcasting rates, i.e., Originator Intervals, for sending the fake OGMs. We used broadcasting rates of 500 ms, 1000 ms, 1500 ms, and 2000 ms. The duration of each experiment is 30 minutes.

Figure 4.5 shows the rates of routing misbehavior R^{O_n} calculated by nodes O_1, O_2, and O_3 while node O_1 broadcasts fake OGMs at the rate of 1000 ms. As node O_1 starts broadcasting fake OGMs at 3 secs, we can remark a large increase of rate $R_{S\_rcv}^{O_1}$ in Figure 4.5 (a), and consequently the increase of rates $R_{NS\_rcv}^{O_2}$ and $R_{NS\_rcv}^{O_3}$, as shown in Figures 4.5 (b) and (c), respectively. Since the number of detected inconsistencies $R_{S\_rcv}^{O_1}$ continues increasing during the attack execution, at 5 secs $R_{S\_rcv}^{O_1} \geq Th_{O_1}^{S\_rcv}$ as seen in Figure 4.5 (a). Moreover, at 7 secs $R_{NS\_rcv}^{O_2} \geq Th_{O_2}^{NS\_rcv}$, and at 8 secs $R_{NS\_rcv}^{O_3} \geq Th_{O_3}^{NS\_rcv}$, as seen in Figures 4.5 (b) and (c). For node O_3, we assume $Th_{O_3} = Th_{O_2}$.

At the moment $R_{S\_rcv}^{O_1}$ surpasses $Th_{O_1}^{S\_rcv}$, node O_1 perceives that possibly routing manipulation attack is happening in the neighborhood and trigger the consensus procedure, as explained in Section 4.5.4. After exchanging Detection Results DR_{O_1}, DR_{O_2}, and DR_{O_3} with neighbors O_2 and O_3, the Consensus Algorithm of node O_1 realizes the own node O_1 is generating phony OGMs for Originator O_2, and designates it as malicious node $MN_{O_1} = O_1$. Accordingly, nodes O_2 and O_3 conclude that OGMs received from node O_1 for Originator O_2 were not previously received from node O_2, i.e., node O_1 is fabricating fake OGMs for node O_2, and identically indicate it as malicious node $MN_{O_2} = MN_{O_3} = O_1$. Then, the three nodes share their individual decisions. Since all nodes agree on the same malicious node O_1, at this point, a consensus is reached and node O_1 is deemed malicious. A possible countermeasure for this type of routing redirection attack is the IDS of node O_1 start dropping the fake OGMs broadcasted by node O_1 itself for Originator O_2 that violate Routing Constraint C_1, or deny access to node O_1 into the access control list of the mesh routers. Alternatively, the IDS of nodes O_2 and O_3 can drop the fake OGMs received from node O_1 for Originator O_2 that violate Routing Constraint C_2.

In this attacking scenario, false positives are only caused by the dropping of specific Routing Events resulted from the packet loss of the communication link. In this case, if remote Routing Events generated by node O_1, which confirm the reception of routing

packets, are lost during transmission, the node itself has the proof of reception of the routing packets, i.e., it treats its own local Routing Events generated for these routing packets. Then, node O_1 does not produce false positives. However, the neighboring node O_2 detects false positives since it does not receive the remote Routing Events which prove node O_1 received the routing packets. Since own node O_1 is in charge of initiating the consensus mechanism, even if false positives detected by its neighbors are high, those false positives do not have great impact on the intrusion detection process.

The effective detection rate, i.e., true positive rate, is defined as: $Dr = \frac{F-FT}{TF}$, where F is the number of inconsistencies detected by the node during the attack, i.e., the malicious traffic, as explained in Section 4.5.3; FT is the number of inconsistencies, i.e., the false positives, detected during the phase of threshold definition in the scenario of Section 4.6.2; and TF is the total number of fake OGMs broadcasted by the attacker throughout the attack. Then, we calculate the effective detection rate Dr for nodes O_1, O_2, and O_3 taking into account the malicious scenario of Section 4.6.3, where node O_1 fabricates and broadcasts OGMs for gateway node O_2.

We obtained $Dr^{O_1} = 100\%$, which is normal since detection of routing misbehavior in node O_1 relies exclusively on local Routing Events which are not affected by packet loss. And for the neighbors we found $Dr^{O_2} = 86\%$, and $Dr^{O_3} = 100\%$. The true positive rate of node O_2, Dr^{O_2}, is lower compared to one of node O_1, Dr^{O_1}, because of the high packet loss rate (15%) between nodes O_1 and O_2 which seriously impacts the intrusion detection accuracy at node O_2. Nonetheless, node O_3 detects all the fake OGMs broadcasted by node O_1 since there is no packet loss in the link between nodes O_1 and O_3. The average Dr for the nodes is 95.3%. However, in realistic wireless environment, where the packet loss rate of links is not so elevated, it is highly likely that the effective detection rate, i.e., true positive rate, of the neighboring nodes would be around 96%.

The false positive rate Fr considered for the nodes is the same as the ones defined in Section 4.6.2: $Fr^{O_1} = 4\%$, and $Fr^{O_{2,3}} = 2\%$ for nodes O_2, and O_3. The false positive rate of node O_1 is not correct since this node is not affected by packet loss. Nonetheless, the false positive rate of nodes O_2 and O_3 are more coherent but not exact for real WMN where the packet loss rate of links is equitable among the neighbors experiencing congestion.

The false negative rate is defined as: $Nr = 1 - \frac{F}{TF}$. For that, we employ the same malicious scenario of Section 4.6.3, where node O_1 broadcasts phony OGMs in the network, but there is no packet loss rate in the link between nodes O_1 and O_2 since we seek to estimate the malicious behavior that is not detected by IDS at neighboring nodes O_2 and O_3. Then, we obtain $Nr^{O_{1,2,3}} = 0\%$ for the three nodes O_1, O_2, and O_3. As explained in Section 4.4, since we assume that nodes do not present faulty behavior, then all packets fabricated and disseminated by malicious node O_1 are precisely detected by node O_1 itself and its neighboring nodes O_2 and O_3 since the packet loss of the links is 0%. Therefore, the distributed intrusion detection mechanism does not yield false negatives.

4.7 Performance Analysis

In this section, we calculate performance metrics that consist of the memory, CPU, and communication overhead. These performance metrics are important for the proper evaluation of the proposed intrusion detection solution showing the efficiency, effectiveness, and robustness of the approach. Consumption of hardware and communication resources by the IDS at the node is studied in this section. In addition, we analyze how our intrusion detection solution copes with the limited bandwidth in WMNs, and the bandwidth overhead of the approach is evaluated.

4.7.1 CPU and Memory Consumption

The computation overhead of the IDS instance consists of: (i) capturing of routing packets from network interface by using *libpcap* on Linux OS; (ii) processing of routing packets; (iii) generation of Routing Events; and (iv) execution of attack detection scripts. In addition, the IDS tool, which is connected to the neighbors by TCP sockets, sends Routing Events to neighboring IDSs through IP packets. Moreover, the IDS instance saves all the intrusion detection related information to log files in the VM. All these intrusion detection operations are added to the computation overhead of the intrusion detection solution.

We measure the consumption of IDS resources, i.e., CPU and memory usage, at node O_1 during the intrusion detection operations for the two scenarios described in Sections 4.6.2 and 4.6.3 : (i) $Scen1$: no malicious node is present; (ii) $Scen2$: the attacker performs

message fabrication attack. For that purpose, we apply *sysstat* performance measurement tool [43] to collect the performance log data in the QEMU VM, which has the hardware configuration described in Section 4.6.1. *Sysstat* contains utilities to monitor system performance and usage activity, such as CPU and input/output statistics for devices, in order to fix performance issues and hardware bottlenecks. Figures 4.6 and 4.7 show the computation overhead resulting from the execution of IDS tool at node O_1.

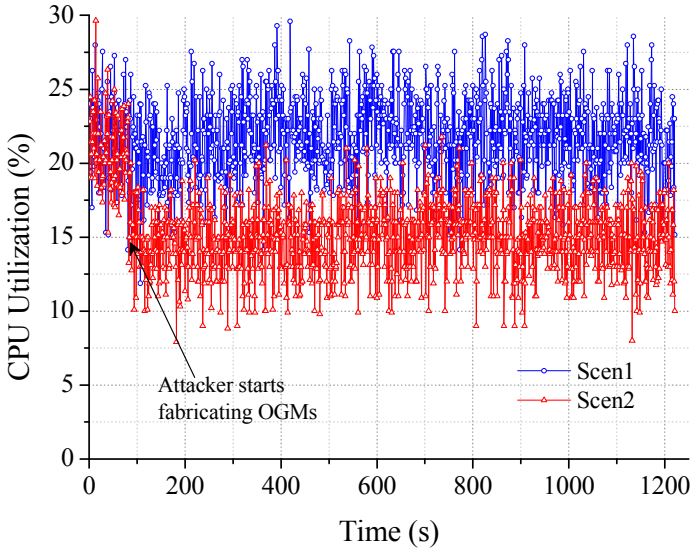


Figure 4.6: CPU usage for the IDS at node O_1.

Table 4.2: CPU statistics for the IDS at node O_1.

Scenario		**Average CPU usage (%)**	**Maximum CPU usage (%)**	**Minimum CPU usage (%)**
$Scen1$		**21.6**	29.6	11.9
Scen2	Before Attack	20.6	29.6	13.3
	After Attack	14.8	21.8	7.9
	All	**15.2**	29.6	7.9

In Figure 4.6, the CPU utilization for the IDS in the first scenario $Scen1$, i.e., the percentage of CPU time spent by the IDS while executing at the kernel, is fair considering all the operations executed by the IDS during the intrusion detection process. In this scenario, the average CPU usage is 21.6% while the highest utilization is 29.6% and the lowest consumption is 11.9%, as shown in Table 4.2. However, for $Scen2$, we observe a

reduction of CPU usage to 14% at the moment the attacker starts broadcasting fake OGMs in the network interface of node O_1, i.e., at 88 secs, as shown in Figure 4.6. In $Scen2$, before the attack is initiated, the average CPU usage is 20.6%, having a peak of 29.6% and the lowest usage is 13.3%, which is very close to the CPU consumption for the IDS in $Scen1$, as presented in Table 4.2. Nevertheless, after the attack is triggered the mean CPU usage is decreased to 14.8% having a minimal utilization of 7.9% and maximal usage of 21.8%. In addition, for the entire experiment duration in $Scen2$, the average CPU utilization is reduced to 15.2%, a difference of 6.4% compared to the average CPU usage for the IDS in $Scen1$, as showed in Table 4.2.

In scenario $Scen2$, the CPU should be more occupied by IDS task since the intrusion detection operations are more intensive, such as the number of routing packets treated per second, Routing Events generated and exchanged with IDSs, and the intense logging of intrusion detection information, because of the attack execution at node O_1. Nonetheless, we perceive an opposite behavior from the IDS, i.e., its average CPU usage decreases when the traffic load is increased. That unusual behavior is attributed to a "software bug" in Bro communication framework [44], where the Bro process keeps an initial CPU usage of about 30%, that is high, regardless of whether there is network traffic to analyze, but if the traffic load increases, the CPU consumption is progressively reduced until it is stabilized.

The computation overhead added by IDS is low since the average CPU utilization during the attack detection process is 14.8%, and according to Bro communication framework, the CPU consumption of IDS tends to reduce and stabilize when the network traffic load is increased, i.e., number of packets processed per second. Therefore, if more nodes are added in the neighborhood, the IDS is capable of efficiently handle all the traffic generated by the neighboring nodes maintaining the accuracy of intrusion detection. Thus, the IDS solution is scalable with respect to the computation overhead.

In Figure 4.7, we note that the memory consumption in both scenarios $Scen1$ and $Scen2$ gradually increases over the experiment duration. In Table 4.3, the average memory consumed in the two scenarios is 4.97% of 256MB of main memory that is equivalent to 12393.25 Kbytes (12.10 Mbytes), which is indeed the physical memory used by IDS task. In $Scen2$, the upper limit of memory utilization is 5.34%, that is during the attack execution, and the lower limit of memory usage is 4.15% which is before the attack

execution, as presented in Table 4.3. Then, in $Scen2$, we notice an increase of 1.19%, i.e., 2.89 Mbytes, during the intrusion detection process, which is acceptable taking into consideration all the attack detection operations performed by the IDS. Therefore, the detection operations performed by the IDS incur a small memory overhead regarding the size of main physical memory of node. Since common wireless routers are equipped with 32MB of RAM memory, then the memory overhead added by the IDS is low.

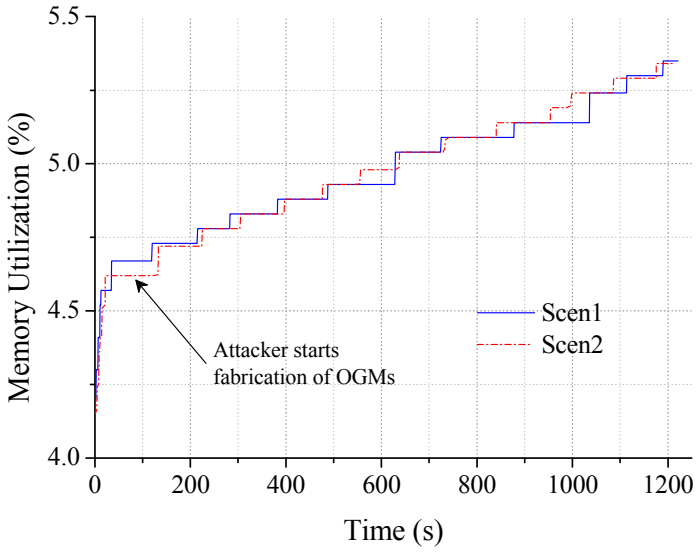


Figure 4.7: Memory usage for the IDS at node O_1.

Table 4.3: Memory statistics for IDS at node O_1.

Scenario		Average memory usage (%)	Maximum memory usage (%)	Minimum memory usage (%)
$Scen1$		**4.97**	5.35	4.2
Scen2	Before Attack	4.56	4.62	4.15
	After Attack	5	5.34	4.62
	All	**4.97**	5.34	4.15

In Figure 4.7, we can see that memory overhead grows progressively whether or not malicious traffic is broadcasted on the node interface, then, in theory, if traffic load increases, i.e., the number of packets processed per second, then the memory utilization of the IDS does not have a significant increasing. Considering this, we can deduce that in case of extra nodes are introduced to the vicinity then the memory overhead exhibits a trend of

linear increase as the number of packets transmitted by the neighboring nodes also increases, and the memory overhead remains at a reduced level. Therefore, the intrusion detection mechanism of IDS is scalable in relation to memory overhead.

We conclude that both CPU and memory consumption for the IDS does not achieve high levels even so the number of packets per second is increased on the node interface and consequently the intrusion detection activities, therefore the intrusion detection mechanism is scalable in respect to computational overhead. The CPU and memory overhead are maintained at a stable low level of around 15.2% (CPU time) and 4.97% (12.10 Mbytes) respectively during the distributed attack detection procedures. According to Bro's user manual, if CPU usage is less than 50% and memory usage is less than 70% of physical memory, then the IDS is able to perform accurate analysis, i.e., the IDS will not drop packets due to lack of computational resources. Anyhow, our distributed intrusion detection solution implemented in Bro imposes a small processing cost to the hardware, and yet incurs little memory overhead to the system.

4.7.2 Communication Overhead

Since BATMAN protocol essentially uses a flooding mechanism to transmit OGMs, the RPA module of the IDS has to analyze each OGM packet received or broadcasted and generate a respective Routing Event message to the neighbors. For that reason, Routing Event messages are the main source of message overhead during the intrusion detection.

We calculate the message overhead using the experiment scenario of Section 4.6.2 that is used to compute the thresholds for the nodes. The message overhead formula for Routing Events RE in terms of average message size (in bytes) is $MO_S = \frac{\sum |RE| / N}{\sum |OGM| / n}$, which is the ratio of message overhead for the average RE message size in relation to OGM packet size for the entire experiment duration, i.e., 20 mins, where $N = \sum RE$ and $n = \sum OGM$. The message overhead formula for RE in terms of message frequency is $MO_F = \frac{\sum RE}{\sum OGM}$, which is the ratio of message overhead for the number of RE messages exchanged by the node compared to the number of OGM packets transmitted by the node during the experiment.

The RE message overhead MO_S calculated for node O_1 is $MO_S^{O_1} = 8.5\%$, and for node

O_3 is $MO_S^{O_3} = 7.2\%$. We see that $MO_S^{O_1} > MO_S^{O_3}$ since the average RE message size: $\frac{\sum|RE|}{N}$ is smaller for node O_3 as there is no packet loss in its links, then a higher number of RE messages are processed by node O_3. We perceive that for both nodes the RE message overhead MO_S, in terms of average message size, is fair considering that RE messages are transmitted using TCP data protocol, which adds considerable packet overhead compared to BATMAN Ethernet frames on Layer 2. The average size of a RE TCP packet is 271 bytes while the size of OGM packet is only 44 bytes, then RE message overhead is considered reasonable. The reason for the message overhead MO_S behavior is that a unique TCP packet transports various RE messages as the effect of analysis of several OGMs.

The RE message overhead MO_F estimated for node O_1 is $MO_F^{O_1} = 5.9\%$, and for node O_3 we found $MO_F^{O_3} = 6.7\%$. For this message overhead, we note a contrasting behavior $MO_F^{O_1} < MO_F^{O_3}$ compared to MO_S, because the number of RE messages is higher for node O_3, as said before, attributable to the absence of packet loss in the links of node O_3. The RE message overhead MO_F, with respect to the number of packets, is low considering that ACK packets are also taken into consideration for the overhead calculation. The reason for that is the same: various RE messages are aggregated into a single TCP packet then reducing the number of TCP packets exchanged between nodes.

However, if we calculated the ratio of message overhead for the total RE message size: $MO_T = \frac{\sum|RE|}{\sum|OGM|}$, we find $MO_T^{O_1} = 49.9\%$ for node O_1, and $MO_T^{O_3} = 48.1\%$ for node O_3, which is considerably high compared to OGM packet overhead, since a priori, as explained before, RE messages are essentially exchanged using TPC packets which incurs significant message overhead in respect to OGM packets exchanged by the node.

The average transmission rate of RE messages for node O_1 is 0.176 Mbit/sec and the average transmission rate of OGMs is 0.0035 Mbit/sec, which is 2% of RE transmission rate. Then, the average total bandwidth overhead is 0.179 Mbit/sec, which is appropriate for a WMN where common mesh routers are equipped with wireless cards having the potential to provide a transfer speed of up to 108 Mbps. Thus, the network communication system could afford this kind of communication overhead (in megabit per second) imposed by the IDS communication mechanism along with BATMAN routing protocol. Hence, the IDS approach is scalable in the matter of communication overhead.

We also calculated the RE message overhead MO_S and MO_F for the malicious scenario of Section 4.6.3, where node O_1 broadcasts fake OGMs through the network. For node O_1 we found $MO_S^{O_1} = 7.6\%$ and $MO_F^{O_1} = 5.6\%$, and for node O_3 we obtained $MO_S^{O_3} = 6.3\%$ and $MO_F^{O_3} = 6.7\%$. We can remark that message overhead MO_S calculated for both nodes O_1 and O_3 presents a low reduction while message overhead MO_F exhibits a small increase for the two nodes O_1 and O_3 in relation to the first scenario of threshold definition where there is no attacker. The reason is that excessive number of OGMs is broadcasted by node O_1 because of message fabrication attack, causing an increase in the total number of OGMs processed by the nodes, and consequently the number of RE messages exchanged by the nodes. Nonetheless, the communication overhead added by the IDS does not present a significant increase during the attack detection process then the message overhead is kept nearly constant. In addition, we calculated the message overhead rates concerning the total RE message size for nodes O_1 and O_3: $MO_T^{O_1} = 42.4\%$ and $MO_T^{O_3} = 42.3\%$, where we note a decrease compared to the first scenario. This is because, in the attacking scenario, node O_1 diffuses additional fake OGM packets, which significantly introduce more OGM overhead to the node communication link.

4.8 Discussion

4.8.1 Security

Each IDS should not generally assume that other IDS instances are well behaving unless there is some strong supportive evidence. Assuming that IDS at the node is not compromised, it is a strong assumption. Security requirements of the IDS architecture should be satisfied since attackers may also try to disrupt the IDS components and its communication system. Thus, detection of compromised IDSs is an important security issue. Optionally, the IDS should be resistant to some kind of intrusions, i.e., it has to tolerate minor intrusions on a certain level maintaining the good detection accuracy while detecting more severe attacks, as exemplified in Section 5.6.1.

The functionality of cooperation between nodes creates new security risks. The mesh network model, described in Section 4.1, assumes that communication between IDSs are

secure, however communication between nodes is susceptible to main in the middle attacks. A malicious node may modify, for instance, Detection Results and decision messages originating from specific nodes in order to hinder or mislead the consensus scheme in the node so this node will falsely accuse well-behaved nodes. The solution is to encrypt and authenticate the communication channel of nodes and enforce signature verification by applying cryptographic solutions. To that end, we can employ Bro secure transmission mechanism, introduced in Section 4.3.1, which basically uses SSL protocol for key exchange, confidentiality, and message integrity. Unless the malicious node possesses the private keys of both end nodes, the attacker is not able to launch main in the middle attacks.

Nevertheless, a malicious node can drop routing information messages, i.e., Routing Event messages, diffused by the IDS at the node, to mislead neighboring nodes to decide upon the conviction of a node. One possible countermeasure to prevent this type of attack is the deployment of a second lightweight IDS at the node to exclusively monitors the IDS messages concerning traffic information and detection results that are exchanged between IDSs. Then, each node overhears packets sent and received in its radio range promiscuously for subsequent analysis and crosschecking of collected data. We choose to monitor the neighborhood by overhearing packet transmissions in order to not incur extra message overhead to the network communication system which is already saturated by the intensive exchange of messages among IDSs.

The premise of trustiness and good reputation for the node IDS is a strong assumption. Malicious nodes, owning the private key of the node, may transmit false intrusion detection information upon request, known as blackmail attack [41], and collude with other malicious nodes to mislead the consensus mechanism of target nodes to incorrectly decide on the culpability of legitimate nodes. A compromised node, which is supposed to collaborate for cooperative intrusion detection, might transmit altered routing information, e.g., Routing Events, or false detection results in order to obstruct the intrusion detection process in the neighborhood, cover malicious activities of other nodes, or wrongly incriminate benign nodes as malicious. A potential solution to avoid blackmail attacks is the utilization of a cooperative trust management system at each node for estimating trust values for nodes in the neighborhood based on the intrusion detection information and consensus outcome provided by these nodes at regular intervals. Then, the trust mechanism avoids requesting

cooperation, e.g., during consensus process, and analyzing routing information from nodes that have not previously established trusted relationships. However, this approach can have an impact on the accuracy of IDS consensus scheme, for instance, in a network with limited trusted relationships among nodes the lack of enough trustful nodes can lead the IDS to make inconsistent decisions about suspected nodes.

4.8.2 Complexity and Limitations

As discussed in Section 4.7.2, the communication overhead introduced by the IDS message exchange compared to the OGM overhead is acceptable. However, for restricted bandwidth networks, communication between IDS instances should be minimized due to the constrained communication capacity of the wireless links. Therefore, low bandwidth requirement should be met for the IDS communication mechanism in certain wireless environments. Moreover, a large amount of traffic load in dense networks could not be efficiently analyzed by a resource constrained mesh device then causing packets to be discarded. A remediation solution for this type of wireless scenario is to keep of list of trusted nodes and watches the traffic of suspected nodes instead of monitoring all the nodes in the neighborhood. A second alternative is to monitor a random choice of neighboring nodes at regular intervals. Alternatively, the IDS could monitor routing packets randomly chosen, or monitor routing packets periodically, in order to reduce the amount of messages, i.e., Routing Events, exchanged between IDS instances. The use of UDP protocol or other optimized protocol in place of TPC protocol for the IDS communication mechanism would likewise reduce the message overhead imposed by the IDS message exchange.

For the IDS to be practical and effective, it has to be scalable. Our collaborative IDS architecture is considered scalable since it performs well in sparsely populated topologies as well as in densely populated topologies with low computation overhead. The only requirement to perform the distributed attack detection is running an IDS tool at each node for exchanging Routing Event messages with the neighboring IDSs. The IDS is capable of collecting and analyzing routing traffic in neighborhoods with a large number of neighbors and maintain the detection efficiency at a good level. However, the communication overhead also increases accordingly, i.e., additional message overhead is added as number of nodes increases and consequently more packets and Routing Events are exchanged,

which can be a problem for bandwidth constrained wireless networks, but it is scale for networks with sufficient bandwidth resource. The IDS scalability concerning CPU overhead, memory overhead, and communication overhead is discussed in Section 4.7. The scalability of proposed threshold mechanism is analyzed in Section 4.6.2.

The consensus scheme, proposed in Section 4.3.4, is considered scalable concerning the neighborhood size, i.e., the number of participating nodes, since it works suitably accurate for a reduced group of nodes and also works efficiently when applied to larger groups of nodes keeping the same detection accuracy. Moreover, our consensus approach does not have special requirements, for instance, the need for multiple neighboring nodes to collaborate in order to make a decision about a suspicious node as the consensus approach proposed by SCAN [22], presented in Section 2.3. Our consensus mechanism always applies the same *decision rule* does not matter the group size in order to make a final decision about the suspicious node. Therefore, the consensus approach does not fail when the number of nodes in the group increases, thus it do scale.

Our threshold approach pre-calculates thresholds for a given network topology considering a given time period. However, if the traffic load of the network changes unexpectedly, i.e., several nodes join the mesh network at the same time and the network topology is redefined, then the calculated threshold could no more possibly reflect the actual network performance in terms of link quality, i.e., packet loss rate due to channel congestion and collision. Then, it is necessary to recalculate the threshold values for all the nodes, particularly to the ones directly impacted by the topology modification. To avoid that situation, a possible solution is to apply a dynamic threshold approach, where threshold values are dynamically calculated for all the nodes at regular intervals, or calculated separately for neighborhoods where new nodes are added to it.

If a node moves out of the communication range of the node to another node, the IDS approach has to take this roaming behavior of the node into account since node mobility has a considerable impact on the detection accuracy and rate of false positives calculated by the node. Timeout values can be added to the intrusion detection module, which are triggered before initiating the consensus mechanism, to avoid making incorrect decisions about a roaming node. In addition, this approach avoids unnecessary exchange of messages between IDSs and falsely accusing a legitimate node of being malicious. The IDS can make

use of the mobility management scheme provided by the routing protocol [36] in order to be aware of current state of each node in the network, i.e., which are the current neighbors of each node, and if the particular node is roaming or is no more part of the network.

4.9 Conclusion

In this chapter, we presented a distributed intrusion detection methodology for WMNs which is integrated to a popular IDS tool used in production environment. The IDS is responsible for monitoring the traffic at the node and exchanging Routing Events with neighboring nodes in order to identify possible routing misbehavior in the vicinity. We employ a monitoring based detection approach which precisely detects routing attacks that violate the routing functionality of nodes, for instance, in case the node is diffusing incorrect routing packets through the network. The IDS monitors the routing activity of the neighborhood over a period of time and set thresholds to distinguish malicious routing behavior from communication failures, i.e., false positives. Then, we propose a consensus scheme which permits a group of nodes to be flexible enough to make a decision when it is convenient, e.g., the majority agrees upon a verdict of guilty, while still adopting the main concepts of consensus decision-making process.

We apply the cooperative approach for detecting message fabrication attacks, which pose as a generic and potential threat to any node in the network. This attacking technique allows malicious nodes to gain considerable advantage in attracting traffic in the network to itself. Thereafter, the malicious node can perform either passive attacks such as eavesdropping and traffic analysis, or more severe active attacks such as dropping or selectively dropping data packets in order to deny data delivery services to target nodes.

The intrusion detection approach provides rapid and efficient malicious traffic detection for collaborative recognition of routing attacks in the neighborhood. It should be noted that the approach is adaptable to most routing protocols and can be applied to detect a wide range of attacks. The distributed intrusion detection solution has been validated by implementation in a virtualized mesh network environment. Experimental results show that the intrusion detection scheme has very low false positive rate, no false negatives, and high detection rate. In addition, the attack detection algorithms take into account the limited hardware resources of the node such as CPU and memory, then appropriately fitting in

mesh devices owning restricted computational resources. Despite the approach presents a good efficiency, the message overhead introduced to the communication system is not the minimum possible message overhead for the communication system.

Therefore, we have implemented a scalable, efficient, and effective intrusion detection architecture since it overcomes the main challenges of intrusion detection in wireless scenarios, such as a decentralized network, lack of central point of traffic convergence, and the limitations of wireless communication, i.e., intrinsic packet loss in the links. All the intrusion detection modules of the approach including RPA, DIDE, and CCM are successfully integrated together and are able to precisely identify routing misbehavior targeting BATMAN routing protocol with practical assumptions, and employing low computation overhead, i.e., CPU and memory overhead.

Chapter 5

Distributed Detection of Packet Dropping Attacks

The IDS needs to be practical and have a scalable architecture in order to collect and process sufficient traffic information to effectively detect malicious traffic. Since the routing service of WMNs requires strong cooperation among all nodes, then detecting selfish and misbehaving nodes and enforcing cooperativeness are important in this collaborative wireless environment.

In Chapter 4, we have defined a complete distributed intrusion detection architecture, which is exemplified with detection of message fabrication attacks, for instance, a malicious node disseminating fake routing information through the network. In this chapter, we apply this intrusion detection architecture for detection of packet dropping attacks.

The approach employs a collaborative intrusion detection mechanism implemented into the IDS tool at each node, which allows neighboring nodes to cooperatively diagnose malicious packet dropping behavior, and achieve such consensus on the routing misbehavior. Consequently, the packet dropping detection mechanism is able to discover the misbehavior of any node, for example, a node selectively discarding routing packets from specific neighboring nodes.

The rest of the chapter is organized as follows. In Sections 5.1 and 5.2, we describe the system model and adversarial model respectively. In Section 5.3, we present the distributed intrusion detection architecture for detection of packet dropping misbehavior. Section 5.4 proposes the consensus algorithm for malicious packet dropping scenarios. In Section 5.5, we demonstrate the packet dropping detection approach efficiency by implementation. In Section 5.6, we analyze the computational and communication overhead added by the intrusion detection approach. In Section 5.7, we discuss security problems and limitations of our packet dropping detection method. Finally, Section 5.8 concludes the chapter.

5.1 System Model

We consider the same system model for the WMN as introduced in Sections 3.1 and 4.1 since we use the same intrusion detection architecture at the nodes as employed in Chapter 4. We assume all nodes participating in the WMN have suitable hardware resources, such as CPU and physical memory resources, in order to execute the IDS tool for appropriate distributed and cooperative detection of packet dropping intrusions. In addition, avoid the node to discard packets under heavy traffic load because the system it is lacking computational resources to process all incoming packets.

We also take into consideration the same assumptions for the IDS security and trustiness as described in Section 4.1. We assume the communication mechanism of the IDSs is not compromised, then routing information and intrusion detection results shared between nodes are not violated by adversarial nodes, neither the messages exchanged by the consensus mechanism are exploited by attackers.

5.2 Attacker Model

We adopt the attacker model as described in Sections 3.2 and 4.2. We do not consider attacks targeting the physical layer and MAC layer. Defending against such attacks is complementary to this work. In this chapter, we focus on packet dropping attacks, which is a generic threat to any node of the mesh network. Since packet dropping attacks target the routing service and packet forwarding operations of the network, we briefly summarize this attacking method as follows.

Packet dropping attacking method was introduced in Section 3.2.1. We define packet dropping attack as intentional drop of legitimate routing packets or data packets by the misbehaving node, which is supposed to relay such packets, in order to provoke denial of service for target nodes, i.e., the malicious node impedes the affected nodes of having a correct participation in the routing or data delivery services of the network. The attack consequences are disconnection of nodes from the network, interruption of valid routes, and selfishness of nodes. By using packet dropping attacking method, the attacker is able to execute an ample range of packet dropping misbehavior, such as selective packet dropping, probabilistic packet dropping, arbitrary packet dropping, and intermittent packet dropping.

Because packet dropping attacks may present such different variants, detecting precisely this kind of routing misbehavior is a difficult task for IDSs.

5.3 Distributed Detection of Malicious Packet Dropping

The first step of the intrusion detection process starts at the traffic monitoring engine, which captures and analyses packets of interest, i.e., routing packets, being transmitted on the node interface. Subsequently, the collected routing packets are processed by the routing protocol analyzer, which generates routing events that reflect the routing behavior of protocol in operation. Next, these routing events are handled by cooperative intrusion detection algorithms to collaboratively produce intrusion detection metrics in the neighborhood. These intrusion detection metrics are then aggregated and correlated between neighboring nodes in order to reach a global view of the actual state of network activities and localize the source of intrusion in the vicinity if it exists.

In Section 4.3, we have introduced our complete distributed intrusion detection architecture for detection of routing attacks according to the common attacking methods specified in Section 3.2.1. In this section, we show how this architecture can be specifically applied to detect packet dropping intrusions. In special, we provide details on the module of Packet Dropping Detection which outputs Packet Dropping Metrics that signalize the presence of malicious message dropping in the network traffic.

5.3.1 Routing Protocol Analyzer

In order to detect any disruption in the network routing functionality, the IDS tool should firstly examine every routing message traversing the node interface. To that end, we implemented a Routing Protocol Analyzer (RPA) module by using Bro monitoring architecture as explained in Section 4.3.2.

Basically, the RPA module inspects each routing packet captured and generates corresponding Routing Events in accordance with the routing protocol behavior. These Routing Events, which are defined based on BATMAN protocol behavior for detection of packet dropping misbehavior, are described in Section 5.3.3. A Routing Event consists of a message containing arguments extracted from the routing packet and it designates an action

taken by the protocol. The Routing Events are then sent to the Packet Dropping Detection module at script level so they can be properly processed by the node and its neighbors.

5.3.2 Distributed Intrusion Detection Engine

In this chapter, we study in detail the Packet Dropping Detection module of Distributed Intrusion Detection Engine (DIDE) component, which was introduced in Section 4.3.3. The Packet Dropping Detection module receives as input local and remote Routing Events forwarded from the RPA module of node and from neighboring RPA modules for performing cooperative packet dropping attack detection, and it yields Packet Dropping Metrics, which designate whether or not malicious routing behavior is present in the neighborhood. The algorithms for detection of packet dropping intrusion are implemented in this module by making use of particular Routing Event Handlers which treat respective Routing Events. If DIDE component perceives any potentially risky routing behavior within Packet Dropping Metrics which reveals any packet dropping misbehavior that has been monitored, it promptly marks the situation and calls the Cooperative Consensus Mechanism (CCM) module.

The Packet Dropping Detection module is further explained in Sections 5.3.4 and 5.3.5. In Section 5.3.4, Routing Constraints are defined on the routing protocol behavior, and in Section 5.3.5, Packet Dropping Metrics are estimated taking into consideration the Routing Events produced by RPA module and the Routing Constraints defined in Section 5.3.4. BATMAN routing protocol is applied to illustrate the functions of the Message Packet Dropping Detection module.

5.3.3 RPA: Routing Events

The RPA module parses every message broadcasted by the routing protocol based on the protocol behavior and produces Routing Events, which are then treated by corresponding Routing Event Handlers. We implemented this parsing mechanism for BATMAN proactive routing protocol. Nevertheless, the design of RPA module is flexible and can be easily adapted to fit other traditional routing protocols, such as OLSR and AODV.

The RPA module processes BATMAN OGM packets and generates the following Routing Events, which represent BATMAN protocol behavior.

1) $OGM\_rcv(< params >)$: an OGM is received from a neighboring node Nb.
2) $OGM\_broad(< params >)$: the node broadcasts an OGM to its neighbors Nb.
3) $OGM\_rebrd(< params >)$: the node rebroadcasts an OGM received from a neighboring node Nb.

Routing Events own two type of parameters: (i) parameters extracted from the node's local context such as the MAC address of node; and (ii) parameters extracted from the OGM such as Originator Address $Orig$, Source Address Src, Sequence Number value $Seqn$, and TTL. The following Routing Event Handlers are defined to process local and remote Routing Events, i.e., the ones generate by the node and those received from the node's neighbors Nb. Routing Event Handlers store routing information about the current activity of the protocol and incorporate it into the analyses of the future activity of protocol.

1. $EH\_OGM\_rcv(\,)$: it processes the parameters of local and remote Routing Events $OGM\_rcv(< params >)$ and stores the extracted metric data in the following specifics sets.
   (a) $Rcv_{Orig}^{Nb} = +Seqn$: set of $Seqn$ (OGMs) received by the node from Nb for $Orig$.
   (b) $Nb\_Rcv_{Orig}^{Src} = +Seqn$: set of $Seqn$ received by the node's Nb from Src for $Orig$.
2. $EH\_OGM\_broad(\,)$: it processes local and remote Routing Events $OGM\_broad(< params >)$ and stores the extracted data in the following sets.
   (a) $Broad = +Seqn$: set of $Seqn$ broadcasted by the node.
   (b) $Nb\_Broad^{Nb} = +Seqn$: set of $Seqn$ broadcasted by the node's Nb.
3. $EH\_OGM\_rebrd(\,)$: it processes local and remote Routing Events $OGM\_rebrd(< params >)$ and stores the extracted data in the following sets.
   (a) $Rebrd_{Orig}^{Nb} = +Seqn$: set of $Seqn$ rebroadcasted by the node (received from Nb) for $Orig$.
   (b) $Nb\_Rebrd_{Orig}^{Src} = +Seqn$: set of $Seqn$ rebroadcasted by the node's Nb for $Orig$ that were received from Src.

5.3.4 DIDE: Routing Constraints

To detect malicious dropping behavior, we define Routing Constraints $C1$, $C2$, $C3$, and $C4$ based upon the routing protocol behavior. These Routing Constraints are implemented by the Routing Event Handlers defined in previous section, and they employ the respective data sets of the Routing Event Handlers. If a Routing Constraint is violated, we count the number of malicious packet dropping PD detected for this Routing Constraint violated.

1. $EH\_OGM\_rcv(\,)$: it implements: (i) Routing Constraint $C1$ which checks the routing behavior of node's Nb by using remote Routing Events; and (ii) Routing Constraint $C2$ that checks the routing behavior of node taking into account local Routing Events.
   (a) $C1 \leftarrow$ **if** $Seqn \notin Nb\_Rebrd_{Orig}^{Src}$ **then** PD_{Nb_Rebrd}++,
      - check if the $Seqn$ received by neighbor Nb from Src for $Orig$ is rebroadcasted by Nb, where PD_{Nb_Rebrd} is the number of $Seqn$ dropped by Nb for $Orig$ that are supposed to be rebroadcasted by Nb.

   (b) $C2 \leftarrow$ **if** $Seqn \notin Rebrd_{Orig}^{Nb}$ **then** PD_{Rebrd}++,
      - check if the $Seqn$ received by the node from Nb for $Orig$ is rebroadcasted by the own node, where PD_{Rebrd} is the number of $Seqn$ dropped by the node for $Orig$ that are supposed to be rebroadcasted by the node.
2. $EH\_OGM\_broad(\,)$: it implements: (i) Routing Constraints $C3$ for checking the routing behavior of the node and its Nb simultaneously taking into account local Routing Events; and (ii) Routing Constraints $C4$ to concurrently check the routing behavior of the node and its Nb by using remote Routing Events.
   (a) $C3 \leftarrow$ **if** $Seqn \notin Rcv_{Orig}^{Nb}$ **then** PD_{Rcv} ++,
      - check if the $Seqn$ broadcasted by neighbor Nb has been received by the node, where PD_{Rcv} is the number of $Seqn$ dropped by the node that are originated by Nb.

   (b) $C4 \leftarrow$ **if** $Seqn \notin Nb\_Rcv_{Orig}^{Src}$ **then** PD_{Nb_Rcv}++,
      - check if the $Seqn$ broadcasted by the node is received by its Nb, where

PD_{Nb_Rcv} is the number of $Seqn$ presumably dropped by Nb.

The intrusion detection module allows integrating extra Routing Constraints in order to detect specific packet dropping attacks, for example, a selfish node that refuses forwarding packets from particular neighbors in order to save hardware resources.

5.3.5 DIDE: Misbehaving Metrics

The next step of the packet dropping detection algorithm is calculate the rate of packet drop routing misbehavior R for the respective Routing Constraints $C1$, $C2$, $C3$, and $C4$ by using the number of malicious packet dropping PD found for the node and its Nb. The set of R compose the Packet Dropping Metrics, i.e., the Misbehaving Metrics.

- For $C1$ at $EH\_OGM\_rcv(\,)$:

$$R_{Nb\_Rebrd} = \frac{PD_{Nb\_Rebrd}}{|Nb\_Rcv|} \quad (1)$$

where R_{Nb_Rebrd} is the rate of $Seqn$ dropped by Nb for $Orig$ that are not forwarded by Nb.

- For $C2$ at $EH\_OGM\_rcv(\,)$:

$$R_{Rebrd} = \frac{PD_{Rebrd}}{|Rcv|} \quad (2)$$

where R_{Rebrd} is the rate of $Seqn$ dropped by the node for $Orig$ that are not forwarded by the node.

- For $C3$ at $EH\_OGM\_broad(\,)$:

$$R_{Rcv} = \frac{PD_{Rcv}}{|Nb\_Broad|} \quad (3)$$

where R_{Rcv} is the rate of $Seqn$ broadcasted by Nb that are dropped by the node before being processed by the node itself.

- For $C4$ at $EH\_OGM\_broad(\,)$:

$$R_{Nb\_Rcv} = \frac{PD_{Nb\_Rcv}}{|Broad|} \quad (4)$$

where R_{Nb_Rcv} is the rate of $Seqn$ broadcasted by the node that are dropped by Nb before being processed by Nb.

The Packet Dropping Metrics R are constantly monitored by the CCM module for distributed packet dropping detection purposes. These Misbehaving Metrics are sufficient for detecting most of packet dropping attacks threating the nodes of the mesh network.

5.4 Cooperative Consensus Mechanism

The CCM module often analyzes the Packet Dropping Metrics R provided by Packet Dropping Detection module to eventually identify packet dropping intrusions. Before making a final decision, the CCM module consults the neighboring nodes and exchanges Detection Results, i.e., Packet Dropping Metrics, related to the claimed packet dropping attack. Neighboring nodes collaborate directly with the node leading the consensus procedure, so this node can make the right decision about the suspicious activity. Based on that, a consensus is reached among nodes in the neighborhood on the routing intrusion and the malicious node. The motivation is that Packet Dropping Metrics provided by a single node may contain inaccurate detection results due to poor quality transmission of Routing Events in the node link, resulted from interference, channel congestion, or node mobility. Therefore, a legitimate node can be falsely accused by its neighbors just because the node is transmitting packets in a link with poor quality or mostly saturated.

The *consensus process* described in Section 4.3.4, which make decisions by consensus, is concretized as the Consensus Algorithm, which is implemented into CCM module. In the decision process, each node provides inputs to collaboratively generating a verdict. The *consensus process* seeks to generate full agreement from all participating nodes by combining the outcomes from different nodes to form a consensual judgment about the suspected node. If a few nodes do not consent to the verdict, the *consensus process* applies the final *decision rule* to evaluate if the current level of agreement is sufficient to finalize the decision. Nonetheless, if a few nodes are unsatisfied with the decision, they will have to tolerate it since it is believed to be the best possible decision for the group of nodes.

Algorithm 1: Consensus Mechanism

Input: $R^{O_n}_{Nb\_Rebrd}, R^{O_n}_{Rebrd}, Th^{O_n}_{Nb\_Rebrd}, Th^{O_n}_{Rebrd}$
Output: $MN^{O_n}, consensus$

1. **if** $R^{O_n}_{Rebrd} \geq Th^{O_n}_{Rebrd}$ **then**
2. **if** $consensus = false$ **then**
3. $Sync(DR^{O_n}, Th^{O_n})$
4. **for all** Nb_i **do;** where $Nb_i, \dots, Nb_k$: $1 \leq i \leq k$
5. $Sync(DR^{Nb_i}, Th^{Nb_i})$
6. **end for**
7. **if all** $R^{Nb_i}_{Nb\_Rebrd} \geq Th^{Nb_i}_{Nb\_Rebrd}$ **then;** where $Nb_i, \dots, Nb_k$: $1 \leq i \leq k$
8. $MN^{O_n} \leftarrow O_n$
9. $Sync(MN^{O_n})$
10. **for all** Nb_i **do;** where $Nb_i, \dots, Nb_k$: $1 \leq i \leq k$
11. $Sync(MN^{Nb_i})$
12. **end for**
13. **if all** $MN^{Nb_i} = O_n$ **then;** where $Nb_i, \dots, Nb_k$: $1 \leq i \leq k$
14. $consensus \leftarrow unanimous\_agreement$
15. $MN^{O_n} \leftarrow O_n$
16. Raise $(alert)$ **and** Execute $(reaction)$
17. **else if any** $MN^{Nb_i} \neq O_n$ **then**
18. Execute$(decision_rule)$
19. **if majority** $MN^{Nb_i} = O_n$ **then**
20. $consensus \leftarrow true$
21. $MN^{O_n} \leftarrow O_n$
22. Raise$(alert)$ **and** Execute$(reaction)$
23. **else if majority** $MN^{Nb_i} = none$ **then**
24. $consensus \leftarrow true$
25. $MN^{O_n} \leftarrow none$
26. **else**
27. $consensus \leftarrow false$
28. $MN^{O_n} \leftarrow none$
29. Raise$(alert)$
30. **end if**
31. **end if**
32. **else**
33. $MN^{O_n} \leftarrow none$
34. **end if**
35. **else**
36. Raise$(alert)$
37. **end if**
38. **end if**

Since WMNs presents non-negligible natural packet loss, thresholds should be defined. With the use of thresholds, the detection rate is improved and false negatives are excluded

since there is a clear division between malicious traffic and false alarms. Our threshold-based mechanism results in lower false positives rate and provides better efficiency in terms of intrusion detection rate. The threshold mechanism employed by the Consensus Algorithm was explained in Section 4.4.

The Consensus Algorithm for detecting packet dropping attack is summarized Algorithm 1. For the sake of a better comprehension, the algorithm is illustrated using the rates of packet dropping: $DR^{O_n} = \{R^{O_n}_{Nb\_Rebrd}, R^{O_n}_{Rebrd}\}$, which are the Packet Dropping Metrics calculated by node O_n and supplied to CCM module. In the packet dropping scenario, node O_n drops the OGMs originated and broadcasted by Nb_1 which are supposed to be rebroadcasted by node O_n.

We define the threshold Th to differentiate the false alarm rate and the packet dropping detection rate when analyzing the Packet Dropping Metrics of the nodes since the rate of packet dropping R_{Nb_Rebrd}, which is basically calculated using remote Routing Events, is not precise because of the packet loss between the neighboring nodes. For instance, the threshold Th^{O_n}, which is defined for node O_n, is calculated taken into account the packet loss rate of the link of node O_n. Then, the threshold Th avoids the node making inaccurate decisions because of poor quality links resulted from interference and congestion.

We make use of Bro's synchronization mechanism at script level to exchange Detection Results DR^{O_n} and DR^{Nb} and individual decisions about the presumed malicious node MN^{O_n} and MN^{Nb} among the neighboring nodes. For instance, the function $Sync(< var >)$ synchronizes variable var, then every modification node O_n applies to variable var the change is propagated to all neighboring nodes Nb, and the opposite is also true.

The Consensus Algorithm of node O_n, i.e., Algorithm 1, follows the steps.

1. If the rate of packet dropping $R^{O_n}_{Rebrd} \geq Th^{O_n}_{Rebrd}$, node O_n diagnoses that own node O_n is possibly dropping packets and trigger the consensus mechanism.
2. Then, node O_n consults its neighbors by exchanging Detection Results DR^{O_n} with all Nb_i and also receiving Detection Results DR^{Nb_i} from all Nb_i, where $Nb_i, \ldots, Nb_k: 1 \leq i \leq k$
3. If node O_n checks all Nb_i have detected similar packet dropping misbehavior, node

O_n points out the malicious node $MN^{O_n} \leftarrow O_n$, otherwise it does not point any malicious node. In addition, all Nb_i execute the same analysis procedure and equally designate their malicious node MN^{Nb_i} or not.

4. Then, node O_n share its individual decision about the suspected node MN^{O_n} with all Nb_i, and collect the decisions MN^{Nb_i} from all Nb_i.
5. If all the nodes determine the same node as malicious $MN^{O_n} = MN^{Nb_i}$, such consensual decision is made with unanimous agreement. Then, the alarm is triggered, and a response action is required from the Response Mechanism since a reaction should follow the attack detection.
6. If at least one neighbor Nb_i does not find the suspected node O_n is malicious then the *decision rule* is applied. In a first case, if most of neighbors Nb_i indicate the same node O_n as malicious, by having the decision of majority, the consensus procedure is finished. Then, the alert is triggered to inform the nodes about the imminent intrusion, and an active reaction is executed accordingly.
7. The second possibility is the majority of nodes Nb_i does not accuse node O_n as being malicious, so a consensual decision is made on the innocence of suspicious node O_n, and no further action is required.
8. As third option, half of nodes points node O_n as being malicious and the other half decide on the guiltlessness of node O_n, then no final decision is made since there is no consensus among the nodes, and an alert is raised for performing more accurate analysis of network current situation.

The total number of neighboring nodes has little effect on the consensus accuracy. That means if the number of nodes participating in the consensus process increases or decreases, a consensus is reached in the same manner with equal precision regardless of the number of nodes concurrently analyzing Packet Dropping Metrics for detection of malicious packet dropping. In addition, if the amount of decisions made by these nodes about the suspicious node increases or decreases, it does not have impact on the consensus accuracy since the *decision rule* is always applied in case of unanimous agreement is not achieved in the group. Thus, the consensus approach is considered scalable concerning its accuracy.

5.5 Performance Evaluation

In this section, we employ the virtualized mesh network environment to demonstrate the efficiency of the packet dropping detection approach. We have applied this virtualized mesh network architecture in Sections 3.5 and 4.6 to emulate and analyze the performance of routing manipulation attacks in mesh networks.

5.5.1 Experiment Platform

In the mesh network, each node is emulated by a VM that is composed of QEMU emulator [37] running Linux OS and BATMAN routing protocol [14] as Linux kernel module, i.e., each mesh node is represented by a corresponding virtual node. The QEMU VM hardware configuration consists of standard PC 32Bit machine type having one default QEMU 32Bit CPU type with 256 MB of main memory. We consider a common wireless mesh device owns similar hardware configuration as the QEMU VM, then having sufficient hardware resources to execute the distributed intrusion detection solution.

The VMs are connected with each other by a virtual switch [38] which emulates bi-directional communication link between neighboring nodes and also link limitations, such as packet loss rate. In the experiments, we employ a static mesh topology, but node mobility could be similarly emulated. The distributed packet dropping detection approach described in Section 5.3 is implemented and integrated into Bro tool v.1.5.3 [42]. Packet Dropping Metrics and intrusion detection information is collected as log files at each node. The mechanism of data collecting at each node is synchronized with a global clock, i.e., the clock of the main machine that supervises, controls, and connects all the VMs.

5.5.2 Defining the Threshold

For defining the thresholds Th for the rates of packet drop routing misbehavior R, we firstly execute the distributed packet dropping detection solution at each node, then we carry out the packet dropping attacks. Figure 5.1 illustrates the mesh network topology we employ for this first trial, which is composed of four nodes O_1, O_2, O_3, and O_4. The link between nodes O_4 and O_2, and nodes O_4 and O_3 have packet loss rate of 10%.

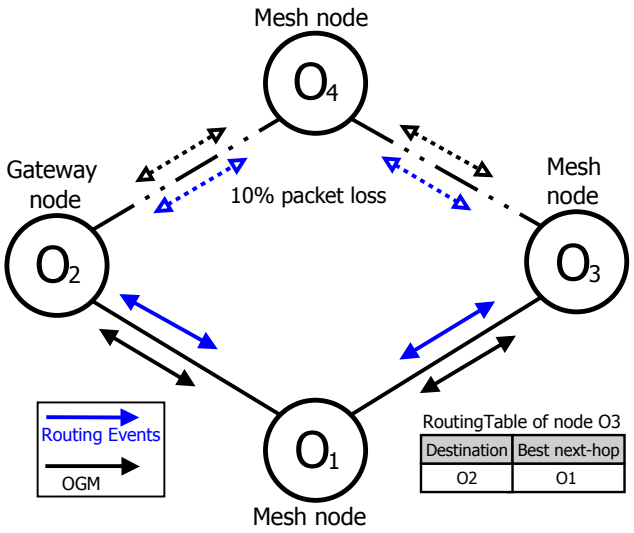


Figure 5.1: Mesh topology emulated for threshold definition.

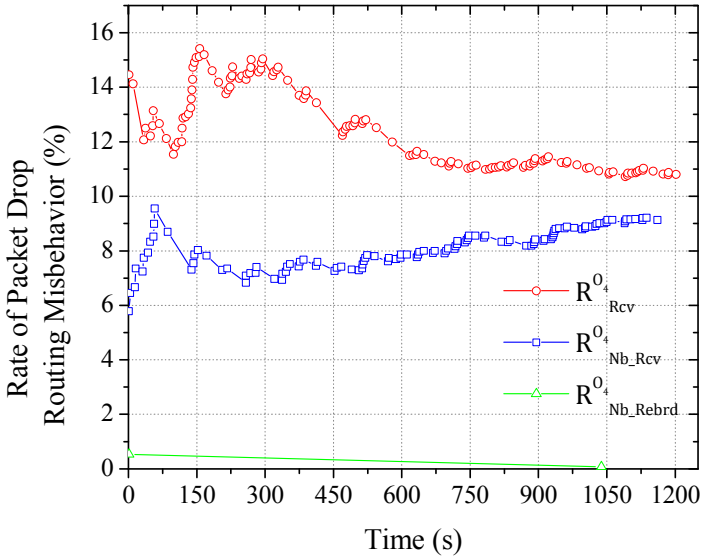


Figure 5.2: Packet Dropping Metrics calculated by node O_4.

In this scenario, all nodes are legitimate and do not present malicious behavior such as packet dropping behavior. Therefore, eventual Packet Dropping Metrics R collected at each node are consequence of Routing Event messages lost during transmission, which is caused by the packet loss rate of the link. The experiment duration is 20 mins. Figure 5.2 shows the rates of packet drop routing misbehavior $R^{O_4}_{Nb\_Rebrd}$, $R^{O_4}_{Rcv}$, and $R^{O_4}_{Nb\_Rcv}$ calculated by node O_4 considering $Nb = O_3$. Those rates are in reality false positives. We notice that

while rate $R_{Rcv}^{O_4}$ has a maximum value of 15.4% and $R_{Nb\_Rcv}^{O_4}$ has a maximum value of 9.5%, the maximum value of rate $R_{Nb\_Rebrd}^{O_4}$ is only 0.5%, which means $R_{Nb\_Rebrd}^{O_4}$ is less affected by the packet loss rate between nodes O_4 and O_3.

The threshold Th_{Rcv} for the rate of packet dropping R_{Rcv} is calculated as:

$$Th_{Rcv} = \frac{\sum R_{Rcv}}{N} \tag{5}$$

where N is the total number of malicious packet dropping PD_{Rcv} detected by the node.

The thresholds Th_{Nb_Rcv} and Th_{Nb_Rebrd} concerning the rates of packet dropping R_{Nb_Rcv} and R_{Nb_Rebrd} are defined in similar way, i.e., by calculating the ratio of Packet Dropping Metrics R_{Nb_Rcv} and R_{Nb_Rebrd} in terms of the number of malicious packet dropping PD_{Nb_Rcv} and PD_{Nb_Rebrd} detected by the node. Seeing that R_{Rebrd} is explicitly determined by making use of local Routing Events and external factors, such as packet loss, has no effect on R_{Rebrd}, we define $Th_{Rebrd} = 5\%$. Table 5.1 displays the thresholds calculated for nodes O_2, O_3, and O_4 by using the respective Packet Dropping Metrics R^{O_2}, R^{O_3}, and R^{O_4}. For nodes O_2 and O_3, we consider $Nb = O_4$ when calculating the thresholds.

Table 5.1: Thresholds calculated for nodes O_2, O_3, and O_4.

Threshold	$\boldsymbol{O_2}$	$\boldsymbol{O_3}$	$\boldsymbol{O_4}$
Th_{Rcv} (%)	11.3	7.9	12.4
Th_{Nb_Rcv} (%)	9.1	8.6	8.1
$\boldsymbol{Th_{Nb\_Rebrd}}$ (%)	0.49	0.48	0.31

In case the mesh topology configuration changes, i.e., the number of nodes in the neighborhood is altered and consequently the amount of routing packets broadcasted, the threshold values calculated for the nodes will have small variations since they depend uniquely on the link's packet loss rate. Therefore, if the link quality is not modified in terms of packet loss, the thresholds do not need to be updated. Therefore, the threshold mechanism achieves scalability with respect to the number of neighboring nodes.

5.5.3 Emulation of Packet Dropping Attack

In this scenario, the malicious node O_1 intentionally drops packets in two different scenarios, as illustrated in Figure 5.3.

1) $Scen\ 1$: malicious node O_1 drops the OGMs received from (broadcasted by) gateway node O_2 which are supposed to be rebroadcasted by node O_1.
2) $Scen\ 2$: malicious node O_1 drops the OGMs rebroadcasted by own node O_1 that are received from node O_2.

Since routing decisions are made relying on statistical analysis of the amount of OGMs received rather than information contained in the routing packets, after the packet dropping attack has initiated, node O_3 will choose node O_4 as best next-hop to gateway node O_2 because node O_4 has rebroadcasted the most number of OGMs for node O_2. However, as shown in Figure 5.1, nodes O_3 would normally choose node O_1 as best next-hop toward gateway node O_2 since the link between nodes O_3 and O_1 has lower packet loss rate (0%) compared to the link between nodes O_3 and O_4 (10%).

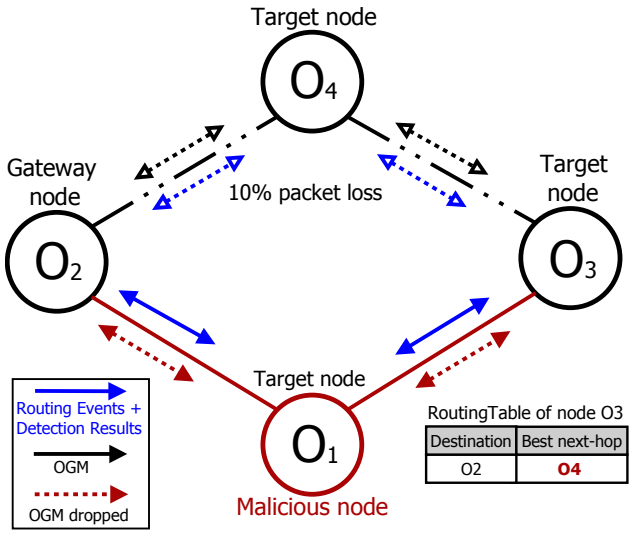


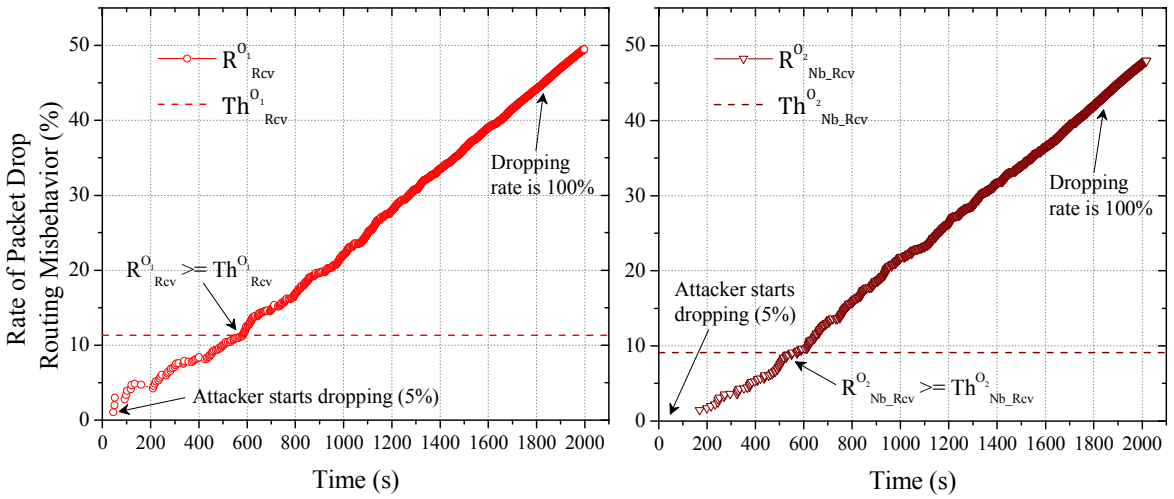
Figure 5.3: Malicious node executing packet dropping attack.

If node O_4 is compromised and since it is the only relay point to gateway node O_2, node O_4 can carry out more serious packet dropping attacks without being noticed by the

target nodes, such as black hole or gray hole attacks. In this type of misbehavior, the malicious node attracts all the data traffic from the target nodes by redirecting the routes of target nodes to the own malicious node. Then, the malicious node drops randomly or selectively a portion of the data traffic in case of gray hole attack, or the malicious node drops all of the data traffic that was attracted to itself in case of black hole attack, instead of forwarding such data traffic as expected by the target nodes.

The packet dropping attack aims to disrupt the integrity of data delivery service and routing service by particularly violating the integrity of routing table of nodes. The packet dropping attack is implemented into *wirefilter* tool [38]. We added a new feature to *wirefilter* tool which allows applying different packet loss rates to the OGMs traversing the virtual link between two nodes according to the Source Address and the Originator Address provided by the user, then OGMs are discarded before reaching the destination node. In addition, we employ *netem* tool [45] in order to apply different packet loss rates to OGMs broadcasted or rebroadcasted by the node in accordance with the Originator Address provided by the user.

5.5.4 Results



(a) Packet Dropping Metric calculated by node O_1.

(a) Packet Dropping Metric calculated by node O_2.

Figure 5.4: Packet Dropping Metrics calculated for $Scen\ 1$.

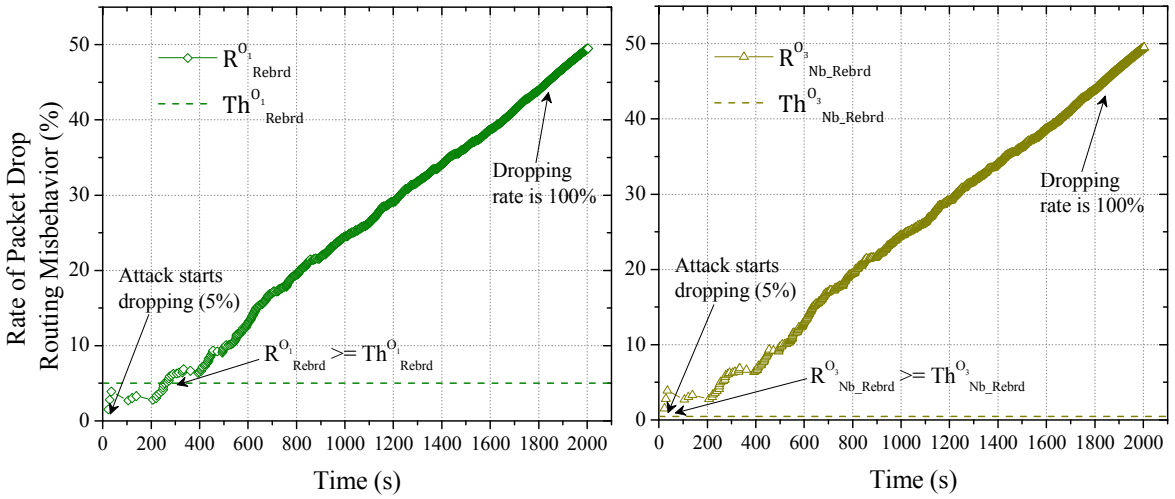
We performed several experiments to evaluate the efficacy of the packet dropping detection approach. The duration of each experiment is around 35 minutes. Figure 5.4 shows the rates of packet drop routing misbehavior $R_{Rcv}^{O_1}$ and $R_{Nb\_Rcv}^{O_2}$ calculated by nodes O_1 and O_2 respectively for packet dropping scenario $Scen\ 1$, where node O_1 drops the OGMs received from node O_2. We assume $Th_{Rcv}^{O_1} = Th_{Rcv}^{O_2}$ concerning node O_1.

The malicious node O_1 starts dropping OGMs at 16 secs applying a dropping rate of 5%. After 180 secs, the malicious node applies a dropping rate of 10 %, then at 376s a dropping rate of 20% is applied, at 556s a dropping rate of 30% is applied, and so on, until the dropping rate reaches 100%. Therefore, the malicious packet dropping rate applied by the attacker is 3.33% per minute. In Figure 5.4 (a), we note that the rate of packet dropping $R_{Rcv}^{O_1}$ follows closely the increase of malicious packet dropping rate applied by the attacker, i.e., the packet dropping detection algorithm of node O_1 precisely detects that own node O_1 is discarding the OGMs received from node O_2. Since the rate of packet dropping $R_{Rcv}^{O_1}$ keeps increasing during the packet dropping attack, at 583 secs $R_{Rcv}^{O_1} \geq Th_{Rcv}^{O_1}$, i.e., the rate of packet drop routing misbehavior exceeds the threshold. In Figure 5.4 (b), we remark that the rate of packet dropping $R_{Nb\_Rcv}^{O_2}$ also correctly detects the malicious packet dropping rate applied by the attacker, i.e., node O_2 perceives that its broadcasted OGMs do not reach node O_1. As the rate of packet dropping $R_{Nb\_Rcv}^{O_2}$ continues increasing, similarly at 573 secs $R_{Nb\_Rcv}^{O_2} \geq Th_{Nb\_Rcv}^{O_2}$, as seen in Figure 5.4 (b).

Since $R_{Nb\_Rcv}^{O_2} \geq Th_{Nb\_Rcv}^{O_2}$ at earlier time than $R_{Rcv}^{O_1} \geq Th_{Rcv}^{O_1}$, node O_2 first notices the malicious packet dropping behavior occurring in the neighborhood and starts the consensus mechanism as described in Section 5.4. Then, nodes O_2 and O_1 share their corresponding Detection Results DR^{O_2} and DR^{O_1} for further analysis. When the common analysis is finished, node O_2 detects node O_1 is intentionally discarding the OGMs broadcasted by node O_2 and designates node O_1 as malicious node: $MN^{O_2} \leftarrow O_1$. From another standpoint, nodes O_1 determines that the OGMs received from node O_2 are being dropped by own node O_1 and identically indicates node O_1 as malicious: $MN^{O_1} \leftarrow O_1$. Since both decisions are equal: $MN^{O_1} = MN^{O_2}$, a final decision is made on the guiltiness of node O_1.

Figure 5.5 illustrates the rates of packet drop routing misbehavior $R_{Rebrd}^{O_1}$and $R_{Nb\_Rebrd}^{O_3}$

calculated by nodes O_1 and O_3 respectively for malicious scenario $Scen\ 2$, where node O_1 drops the OGMs rebroadcasted by own node O_1 which are received from node O_2. For the sake of simplicity, we do not show the rate of packet dropping $R^{O_2}_{Nb\_Rebrd}$ since it is quite similar to the rate $R^{O_3}_{Nb\_Rebrd}$ of node O_3. As stated in Section 5.5.2, we assume $Th^{O_1}_{Rebrd} = 5\%$. The malicious node O_1 starts dropping OGMs at 19 secs. A dropping rate of 5 % is applied after 180 secs, at 379s a dropping rate of 10% is applied, and so on, until the dropping rate achieves 100%. As expected, the rate of packet dropping $R^{O_1}_{Rebrd}$ follows very close the increase of malicious packet dropping rate presenting a linear trend, as shown in Figure 5.5 (a). Therefore, the packet dropping detection algorithm of node O_1 accurately identifies that node O_1 is not rebroadcasting OGMs received from node O_2 as it should proceed. In Figure 5.5 (b), we can see that the rate of packet dropping $R^{O_3}_{Nb\_Rebrd}$ likewise follows the increase of malicious packet dropping rate applied by the attacker, which means nodes O_3 efficiently detects node O_1 is refusing to forward OGMs received from node O_2.



(a) Packet Dropping Metric calculated by node O_1.

(a) Packet Dropping Metric calculated by node O_3.

Figure 5.5: Packet Dropping Metrics calculated for $Scen\ 2$.

At 254 secs, $R^{O_1}_{Rebrd} \geq Th^{O_1}_{Rebrd}$, as shown in Figure 5.5 (a). In addition, at 1 sec, $R^{O_3}_{Nb\_Rebrd} \geq Th^{O_3}_{Nb\_Rebrd}$, as seen in Figure 5.5 (b). Since the threshold Th_{Nb_Rebrd} of nodes O_1, O_2, and O_3 is very low as shown in Table 5.1, we assume the consensus

procedure is triggered when $R_{Rebrd}^{O_1} \geq Th_{Rebrd}^{O_1}$, i.e., at 254 secs. In that way, we avoid starting the consensus process earlier and erroneously accuse a legitimate node by prematurely alarming the presumed packet dropping attack at 1 sec.

As $R_{Rebrd}^{O_1} \geq Th_{Rebrd}^{O_1}$, node O_1 diagnoses a packet dropping attack is happening and trigger the consensus mechanism as described in Section 5.4. Thus, node O_1 sends its Detection Results DR^{O_1} to nodes O_2 and O_3 and receives the corresponding Detection Results DR^{O_2} and DR^{O_3} for checking purpose. When the mutual analysis is finished, node O_1 detects that own node O_1 is deliberately dropping OGMs rebroadcasted by node O_1 received from node O_2, and points node O_1 as the malicious node: $MN^{O_1} \leftarrow O_1$. The neighboring nodes O_2 and O_3 conclude that node O_1 is not forwarding the OGMs received from node O_2 and similarly designate node O_1 as malicious node: $\{MN^{O_2}, MN^{O_3}\} \leftarrow O_1$. Therefore, since $MN^{O_1} = MN^{O_2} = MN^{O_3}$, such consensus is reached with unanimity. A proper active response to this type of packet dropping attack is temporarily blocking the network activity of rebroadcasting routing messages on node O_1. In that way, the correct routes of nodes O_3 and O_4 would restored toward gateway node O_2.

In attacking scenario $Scen\,2$, false positives are only caused by the dropping of particular Routing Events caused by inherent packet loss of communication links. If remote Routing Events generated by node O_1, which confirm the transmission of routing packets, are lost during transmission, own node O_1 has the evidence of transmission of the routing packets, i.e., it processes its local Routing Events generated for the routing packets, then it does not produce false positives. However, the neighboring nodes O_2 and O_3 would detect false positives because they miss to receive the remote Routing Events that prove that node O_1 transmitted the routing packets. Since we assume node O_1 itself triggers the consensus mechanism in case of routing attack evidence, the false positives detected by its neighbors do not influence the intrusion detection process.

The effective detection rate, i.e., true positive rate, is defined as: $Dr = \frac{PD-PT}{TP}$, where PD is the number of malicious packet dropping detected by the node during the packet dropping attack, as explained in Section 5.3.4; PT is the number of malicious packet dropping, i.e., the false positives, found during the period of threshold definition in Section 5.5.2; and TP is the total number of packets dropped by the attacker. Thus, we calculate the

effective detection rate Dr for nodes O_1, O_2, and O_3 using the malicious scenario $Scen\ 2$, where node O_1 drops its own OGMs rebroadcasted that are received from node O_2.

For node O_1, we found $Dr^{O_1} = 100\%$, which is considered normal since detection of malicious packet dropping at node O_1 only depends on local Routing Events that are not impacted by the packet loss of link. For nodes O_2 and O_3, we obtained $Dr^{O_2} = 100\%$ and $Dr^{O_3} = 99.9\%$, which is likewise a typical behavior since there is no packet loss rate in the link between nodes O_1 and O_2 and in the link of nodes O_1 and O_3, therefore the detection accuracy of packet dropping attack is not the impacted by external parameters at nodes O_2 and O_3 concerning malicious node O_1. Hence, nodes O_2 and O_3 detect all the OGMs dropped by node O_1. However, in real WMNs, where there is possibly small packet loss rate in links, the effective detection rate of the nodes would be reduced around to 96%.

We assume the false positive rate Fr of nodes O_1, O_2, and O_3 is equivalent to the ones defined in Section 5.5.2 during threshold definition: $Fr^{O_1} = 5\%$, $Fr^{O_2} = 0.49\%$, and $Fr^{O_3} = 0.48\%$. In fact, we fix the false positive rate Fr^{O_1} of node O_1 since packet loss has no influence on the packet dropping detection at node O_1, then this false positive rate is symbolic. The false positive rate of nodes O_2 and O_3 do not represent proper rates for this attacking scenario since they are calculated considering node O_4 as the unique neighbor of nodes O_2 and O_3 in which there is 10% of packet loss in their links, however in the link between node O_1 and nodes O_2 and O_3 there is no packet loss.

The false negative rate is defined as: $Nr = 1 - \frac{PD}{TP}$. We use malicious scenario $Scen\ 2$ for estimating the malicious packet dropping behavior that is not be detected by IDS at nodes O_1, O_2, and O_3. Hence, we obtain $Nr^{O_{1,2}} = 0\%$ for nodes O_1 and O_2, and $Nr^{O_3} = 0.1\%$ for node O_3. As expected, the three nodes accurately detect all the OGMs purposely dropped by malicious node O_1, but node O_3 miss to detect only one OGM dropped by node O_1. That happens probably due to a synchronization issue in the packet dropping detection algorithm at node O_3 when the experiment starts. Therefore, we conclude that the packet dropping attack detection mechanism does not produce false negatives.

5.6 Performance Analysis

In this section, we estimate the amount of system resources consumed by the IDS solution

which runs at each mesh node, such as memory, CPU, and network resources utilization. Since mesh nodes as wireless routers are expected to be limited in computation power, we evaluate the computation and communication cost added by the distributed packet dropping detection mechanism.

5.6.1 CPU and Memory Consumption

The computation cost of the IDS tool includes: (i) routing packets are collected from the node network interface; (ii) the collected routing packets are handled by RPA module, which generates respective Routing Events; (iii) the packet dropping detection algorithm makes use of the Routing Events to produce Packet Dropping Metrics; (iv) Routing Events are sent to neighboring nodes through TCP sockets; and (v) packet dropping detection information is recorded to log files. All those intrusion detection operations incur computation overhead to the mesh node.

We determine the consumption of CPU and memory resources for the IDS at node O_1 in the course of the intrusion detection operations for two scenarios: (i) $S1$: this is the scenario of threshold definition in Sections 5.5.2; (ii) $S2$: this is malicious scenario $Scen\ 2$ of Section 5.5.3, where malicious node O_1 drops the OGMs rebroadcasted by own node O_1. We employ *sysstat* performance measurement tool [43] to collect the required performance data at each node, as described in Section 4.7.1. Figures 5.6 and 5.7 present the computational overhead for the IDS solution at node O_1 for both scenarios $S1$ and $S2$.

In Figure 5.6, we can see the average percentage of CPU time used by the IDS task for the two scenarios $S1$ and $S2$ is acceptable taking this into account all the intrusion detection operations carried out by the IDS during the process of packet dropping attack detection. In scenario $S1$, the average CPU utilization is 24.9% while maximal CPU usage is 34.4% and minimal CPU consumption is 16.2%, as presented in Table 5.2. Concerning scenario $S2$, before the attack is triggered, the average CPU consumption is 23.1%, having maximum usage of 28.4% and the minimum utilization is 16.3%, which is not so different from the CPU usage of scenario $S1$, as shown in Table 5.2. After the attack is initiated, the average CPU utilization is 23.3% having lowest usage of 13.3% and highest utilization of 33.0%, which is also much the same as the CPU utilization in scenario $S1$. For the whole experiment duration of scenario $S2$, the average CPU usage is 23.3%, which is similar to

the average CPU consumption for the IDS in scenario $S1$.

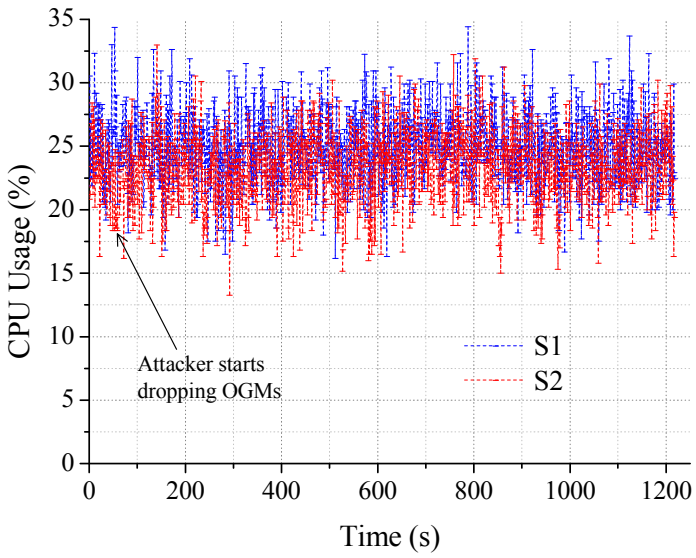


Figure 5.6: CPU utilization for the IDS at node O_1.

Table 5.2: CPU consumption statistics for the IDS at node O_1.

Scenario		**Average CPU usage (%)**	**Maximum CPU usage (%)**	**Minimum CPU usage (%)**
$S1$		**24.9**	34.4	16.2
$S2$	Before Attack	23.1	28.4	16.3
	After Attack	23.3	33.0	13.3
	All	**23.3**	33.0	13.3

We observe only a very small reduction of the average CPU usage of 1.6% for the IDS in scenario $S2$ compared to scenario $S1$. This is because the number of routing packets treated per second by the IDS, i.e., the traffic load, does not change from scenario $S1$ to scenario $S2$ and consequently the intrusion detection operations are practically the same with or without attack. Since the attacker drops OGMs that are processed earlier by the IDS, the IDS workload is not affected by the malicious behavior of attacker in node O_1. This CPU usage behavior observed for the IDS in scenarios $S1$ and $S2$, that is based on the network traffic load, is consistent with the abnormal CPU usage behavior remarked in Section 4.7.1, where the average CPU utilization for the IDS decreases when the traffic load, i.e., number of packets per second, is increased on the node.

The total computation overhead remains quite stable during the intrusion detection activities at scenarios $S1$ and $S2$. Furthermore, the CPU overhead imposed by IDS is not high since the average CPU consumption throughout the packet dropping detection operation is 23.3%. Thus, the execution of the packet dropping detection algorithm at the node adds a very low overhead. In case of additional nodes are introduced to the neighborhood, the IDS tool efficiently processes the extra traffic generated by these neighboring nodes and maintain the intrusion detection accuracy at a high level. Therefore, the IDS is scalable in respect to the computational overhead.

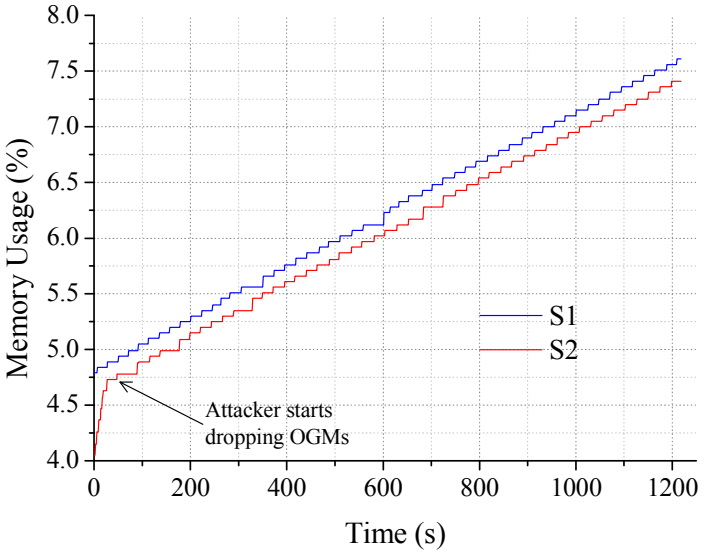


Figure 5.7: Memory utilization for the IDS at nodeO_1.

Table 5.3: Memory consumption statistics for IDS at node O_1.

Scenario		**Average memory usage (%)**	**Maximum memory usage (%)**	**Minimum memory usage (%)**
$S1$		**6.22**	7.61	4.79
$S2$	Before Attack	4.57	4.78	4.05
	After Attack	6.12	7.41	4.62
	All	**6.05**	7.41	4.78

In Figure 5.7, we can notice the rate of memory utilization for the IDS in both scenarios $S1$ and $S2$ slightly increases with respect to time, i.e., the memory usage increases linearly over the time. In addition, the rate of memory usage of scenario $S1$ is

marginally higher than that in scenario $S2$. The average memory consumed by the IDS in scenario $S1$ is 6.22% of 256MB of physical memory, which is equal to 15508.88 Kbytes, i.e., 15.14 Mbytes consumed by IDS from the main memory, as seen in Table 5.3. For scenario $S2$, the peak of memory usage is 7.41% when the attack is performed, and the lowest level of memory utilization is 4.05% that is before the attack is triggered, as shown in Table 5.3. Therefore, there is an increase of 3.36%, i.e., 8.18 Mbytes, during the packet dropping detection process in scenario $S2$, which is tolerable considering the total intrusion detection operations executed by the IDS. Therefore, the packet dropping detection solution incurs low memory overhead concerning the amount of main physical memory, for instance, mesh devices equipped with at least 64MB of RAM memory can bear this memory overhead introduced by the IDS.

In Figure 5.7, we observe the rate of memory utilization increases linearly over the time despite malicious packet dropping behavior is processed by IDS or not. Thus, in case of the traffic load, i.e., the number of packets per second processed by the IDS, increases, then the memory consumption would present a linear increase as well. Hence, if it happens that more nodes are added to the neighborhood, the memory overhead would present a linear increase as the number of packets broadcasted by these nodes also increases, and the memory overhead would remain at a low level. Therefore, the IDS approach is scalable with respect to the memory overhead.

The IDS solution has a similar level of computation overhead across different scenarios. CPU and memory usage for the IDS does not reach elevated levels still the traffic load is increased in the neighborhood and consequently the intrusion detection operations executed by the IDS. The CPU and memory consumption exhibited an average rate of 23.3% (CPU time) and 6.05% (14.72 Mbytes) respectively during packet dropping misbehavior detection process. Therefore, the distributed packet dropping detection mechanism is scalable with regard to the computational overhead.

5.6.2 Communication Overhead

The communication overhead of the IDS solution is basically due to Routing Event RE messages exchanged between nodes by using TCP packets. The average size of a RE TCP packet is 271 bytes whereas the size of an OGM packet is 44 bytes, but a TPC packet

contains many RE messages in consequence of the analysis of various OGMs. The major bandwidth occurs when RE messages are transmitted to neighboring nodes.

We estimate the message overhead added by the IDS using the scenario of threshold definition of Section 5.5.2. The formula of message overhead for RE in the matter of average message size is the same as the one used in Section 4.7.2: $MO_S = \frac{\frac{\sum|RE|}{N}}{\frac{\sum|OGM|}{n}}$, which is basically the ratio of average RE message size with reference to OGM packet size considering the whole experiment duration, where $N = \sum RE$ and $n = \sum OGM$. The message overhead formula in terms of message frequency is: $MO_F = \frac{N}{n}$, which is the ratio of total number of RE messages in relation to the number of OGM packets transmitted and received by the node during the entire experiment duration.

We calculated the RE message overhead for node O_1: $MO_S^{O_1} = 7.2\%$ and $MO_F^{O_1} = 6.7\%$, and for node O_3: $MO_S^{O_3} = 8\%$ and $MO_F^{O_3} = 6.2\%$. We remark that $MO_S^{O_1} < MO_S^{O_3}$, as a result of the higher average RE message size: $\frac{\sum|RE|}{N}$ for node O_3 since less RE messages are processed by node O_3 caused by the packet loss rate in the link of node O_3. However, concerning MO_F we see an opposite behavior $MO_F^{O_1} > MO_F^{O_3}$ since there is no packet loss in the link of node O_1, then the number of RE messages exchanged by node O_1 is higher than node O_3. The RE message overhead MO_S for nodes O_1 and O_3 is considered low if we take into account that RE messages are transmitted using TCP packets which incur significant message overhead in relation to BATMAN packets. This is because usually one TCP packet comprises several RE messages. In addition, the RE message overhead MO_F is also small. The reason for that behavior is the same as explained earlier, i.e., various RE messages are transported by a unique TCP packet then decreasing the total number of TCP packets exchanged between two endpoints.

Nevertheless, regarding the ratio of total RE message size: $MO_T = \frac{\sum|RE|}{\sum|OGM|}$, we obtain $MO_T^{O_1} = 48.5\%$ for node O_1, and $MO_T^{O_3} = 49.8\%$ for node O_3. These RE message overhead MO_T values are considered high and reflect the extra communication overhead added by the constant RE messages exchange of the IDS. That means, a priori, that one RE message is twice the size of the OGM packet.

The effective transmission rate of RE messages for node O_1 is 0.191 Mbit/sec and the

effective transmission rate of OGMs is 0.0039 Mbit/sec. Then, the total bandwidth overhead added by the IDS messages together with BATMAN protocol is 0.195 Mbit/sec, which is convenient for a typical WMN comprised of mesh routers equipped with wireless network cards capable of transfer rates of up to 108 Mbps. Therefore, the WMN communication channels could tolerate this communication overhead introduced by the IDS messages in combination with BATMAN protocol. Hence, the IDS packet dropping detection approach achieves scalability with respect to communication overhead.

In addition, we estimate the RE message overhead for the malicious scenario $Scen\ 2$ of Section 5.5.2, where node O_1 drops the OGMs rebroadcasted by own node O_1 that are received from node O_2. We obtained $MO_S^{O_1} = 6.3\%$ and $MO_F^{O_1} = 7.1\%$ for node O_1, and $MO_S^{O_3} = 7\%$ and $MO_F^{O_3} = 6.6\%$ for node O_3. We note the overhead ratio MO_S estimated for both nodes O_1 and O_3 in this scenario demonstrates a small decrease and the overhead ratio MO_F presents a low increase compared to MO_S and MO_F calculated in the threshold scenario. Therefore, as expected, the IDS does not introduce extra communication overhead during the packet drop attack detection process. Moreover, we estimate the RE message overhead regarding the total RE message size MO_T for nodes O_1 and O_3: $MO_T^{O_1} = 44.5\%$ and $MO_T^{O_3} = 46.3\%$. We observe a small reduction in these rates in relation to ones of the threshold scenario. This is because malicious node O_1 drops OGM packets broadcasted by node O_1 resulting in less RE messages generated by the neighboring nodes, then decreasing the message overhead in terms of average RE message size: $\frac{\sum|RE|}{N}$ and increasing the message overhead in respect to average OGM packet size $\frac{\sum OGM}{n}$. The main reason for that behavior is the reduction of RE messages frequency which is responsible for producing the majority of message overhead.

The possible drawback of the distributed packet drop detection scheme is the overhead that the exchange of RE messages imposes to each node communication link since a node is required to transmit a RE message to its neighbors for each routing packet received or broadcasted by the node. RE messages are essentially exchanged using TPC packets which incurs significant message overhead in respect to OGM packets. Thus, the message overhead added to the communication system is fairly high and in consequence a low-overhead optimized solution is necessary. To overcome that overhead issue, the IDS may

consider exchanging RE messages periodically by employing datagrams (UDP) to transmit such RE messages in order to optimize the communication overhead. Since the size of a typical RE UDP packet would be around 160 bytes, an average reduction of 41% is expected compared to the RE TCP packet size.

5.7 Discussion

5.7.1 Security

The security issues of the IDS solution concerning the assumed trust of IDS behavior and its communication mechanism, that is used to exchange routing information and detection results between cooperating nodes, are discussed in Section 4.8.1. The IDS message exchange mechanism is mainly threaten by main in the middle attacks, malicious message dropping attacks, and blackmail attacks. The solutions to mitigate these potential attacks and vulnerabilities of the intrusion detection architecture are equally discussed in Section 4.8.1. We summarize the attacks targeting the packet drop detection mechanism as follow.

- Main in the middle attack: the malicious node modifies Detection Results messages, i.e., Packet Dropping Metrics, or decision messages from the majority in order to misguide the node in charge of the consensus procedure to incorrectly incriminate legitimate nodes. The security solution for this attack is encrypting and authenticating the messages exchanged among nodes in the neighborhood.
- Malicious message dropping: in this attack, the malicious node intentionally discards, for instance, Routing Event messages disseminated by the node IDS above the established threshold, which should detect this rate of packet drop routing misbehavior, in order to deceive the neighboring nodes to decide on the node's guiltiness. One manner of avoiding this type of attack is the use of a complementary independent IDS at the node which uniquely captures and analyzes Routing Event, Detection Results, and decision messages exchanged between the nodes.
- Blackmail attack: in this attack, the malicious node has the private key of the node and deliberately sends fake intrusion detection information, such as Packet

Dropping Metrics and decisions, to other nodes during the consensus process. Additionally, it may conspire with other malicious nodes to delude the consensus mechanism in the target node to spuriously decide upon the guilt of honest nodes. One way of preventing blackmail attacks is the employment of a distributed and cooperative trust based intrusion detection scheme which calculates the level of trust for each node in the neighborhood. Then the IDS would only accept intrusion detection information, e.g., Packet Dropping Metrics and individual decisions, from trustworthy neighbors, i.e., neighboring nodes that have achieved a certain level of trust in the network.

5.7.2 Complexity and Limitations

The IDS, particularly the consensus mechanism, fails to detect minor packet dropping attacks for two cases: (i) the malicious node slightly drops packets below the threshold that was established to identify this type of dropping attack for saving hardware and communication resources; or (ii) the malicious node drops a few packets intermittently which averages a packet dropping rate over a period of time that is below the threshold. As a result, the packet dropping detection mechanism allows certain types of malicious packet dropping behavior to pass unnoticed. However, the damage caused to the network services by such intermittent packet dropping behavior would be acceptable since it does not considerably affect the network operations.

Nevertheless, we have to define an appropriate threshold for the packet dropping detection algorithm considering two situations: (i) signalize packet dropping intrusions in case the packet dropping misbehavior becomes intolerable, i.e., the routing service or data delivery service is considerably impacted by the node malicious behavior; or (ii) put up with actual packet dropping misbehavior while it does not disrupt the mesh network communication, i.e., violate the network routes and the integrity of data delivery for nodes.

In this chapter, we do not address the problem of data packet dropping misbehavior, where the malicious node conditionally discards data packets in place of forwarding such data traffic, such as black hole and gray hole attacks. However, BATMAN protocol is likewise conceived for data packet forwarding service, although, in this work, we focus exclusively on the routing functionality of BATMAN protocol. To that end, *Batman-adv*

creates a specific virtual interface in Linux OS, which have an IP address. Then, each data packet that enters this particular virtual interface is examined by *batman-adv* kernel module regarding its destination MAC address, and it is forwarded with the help of BATMAN routing tables toward the destination node, which in turn will process this data packet using its respective BATMAN virtual interface. In that way, data packets are transmitted between source and destination nodes in the network using BATMAN protocol. Therefore, if a malicious node selectively drops data packets at a node, our packet dropping detection approach is capable of precisely detecting this malicious data packet dropping behavior by using the same packet dropping detection algorithms. However, small modifications are required to RPA module and Packet Dropping Detection module since these modules have to deal with BATMAN data packets instead of BATMAN routing packets, which have some differences concerning the packet header added by BATMAN protocol.

The distributed and cooperative packet dropping detection mechanism is said to scale since it is efficient and practical when applied to a large number of participating nodes, i.e., for ampler neighborhoods. The IDS tool can be easily installed at every node of the network for exchanging Routing Events and Packet Dropping Metrics messages with neighboring nodes in order to perform collaborative packet dropping attack detection. The approach scalability regarding the consumption of system and communication resources, such as CPU, memory and network bandwidth overhead is analyzed in Section 5.6. The scalability of the consensus mechanism is discussed in Section 5.4, and the threshold scheme scalability is presented in Section 5.5.2.

Our cooperative IDS solution performs better in mesh networks with limited mobility since the intrusion detection approach does not handle node mobility. For example, if a node moves from a mesh node to another node that is more distant, then the IDS consensus scheme of the first node would judge this node as malicious since it is does not have the exact knowledge of the current position of the mobile node in the network. As a consequence, the Response Mechanism would unfairly apply active responses, such as blocking the network access to this mobile node, which does reflect the actual state of the mesh network. To remediate that limitation of the intrusion detection approach, the IDS consensus approach could make use of the client announcement mechanism of BATMAN routing protocol [36], where the actual closest neighbor of the mobile node informs the

entire network that the mobile node is now to be found at a new location. Thus, the IDS has just to retrieve such location information concerning the mobile node from the node which it is installed before making a final decision about this node.

5.8 Conclusion

In this chapter, we have presented a distributed and cooperative approach for detecting packet dropping attacks in WMNs. The IDS tool is placed at each node for passively monitoring the routing traffic on node network interface, i.e., the routing behavior of node, and exchanging Routing Events with the neighboring nodes for monitoring the neighborhood routing behavior in order to detect possible packet dropping routing misbehavior in a collaborative manner. Then, the workload of intrusion detection is fairly distributed among all nodes in the neighborhood. The distributed detection scheme makes it possible to detect a wide range of packet dropping attacks, such as selective packet dropping, probabilistic packet dropping, and periodic packet dropping.

In case of suspected dropping behavior is detected, the IDS evokes the consensus mechanism which shares detection results, i.e., packet dropping metrics, and individual verdicts with the neighbors in order to achieve a consensus on the presumed malicious dropping behavior. In the consensus decision-making approach, the final decision is collaboratively built with full participation of nodes. Thus, packet dropping intrusions are consensually identified in real-time with collective decision. In addition, a threshold scheme, which takes into consideration unintentional packet loss of wireless links in realistic scenario, is employed to distinguish malicious behavior from normal behavior when analyzing the collected routing traffic. If all nodes or the majority judge the suspicious node is malicious, the IDS performs a passive response by reporting the dropping intrusion and logging the incident for further investigation.

The approach is evaluated and validated by implementation using customized VMs which constitute the virtualized mesh network environment. Different packet dropping attacks are emulated in this virtual testing platform. The approach proved to be accurate, reliable, and scalable for the task of detecting packet dropping misbehavior in different scenarios. Through both analysis and experimental evaluations, we show that our packet dropping detection mechanism is effective and incurs low computational overhead, i.e.,

average CPU and memory consumption, and moderate bandwidth overhead. The proposed packet dropping detection scheme is distributed and cooperative in nature.

Chapter 6

Conclusion and Perspectives

6.1 Conclusion

WMNs are cooperative in nature since mesh nodes forward packets on the behalf of other nodes to reach a destination that is out of the node communication range. This emerging multi-hop routing architecture assures reliable data delivery since there is often more than one route from a source node to a destination node in the network. However, WMNs are extremely vulnerable to a wide range of attacks that threaten their operation and provided services. WMNs do not have a clear line of defense. Cryptographic solutions, such as authentication services and access controls schemes, can increasingly improve the security of WMNs. Nevertheless, these preventive mechanisms alone cannot deter all types of attacks, such as internal attackers. Even with end-to-end cryptographic protection, a malicious node can drop packets going through the node and execute DoS attacks.

The assumption of every node will broadcast correct messages and that every node is cooperating to forward correct messages makes WMNs susceptible to attacks. An adversarial node can easily exploit these assumptions to violate the cooperation of nodes, e.g., spreading false routing information. Therefore, a security mechanism is required to track down misbehaving nodes that have valid keys and access rights to the network. IDSs may act as a second line of defense and provide efficient defensive mechanisms to WMNs. An effective way to identify when an attack occurs in a WMN is the deployment of an IDS. Hence, a distributed IDS is of fundamental importance to protect routing services in the WMN. Until now, several intrusion detection schemes for WMNs have been proposed in literature, however their validation has been limited to simulations. In this thesis, our main contribution is the implementation of a practical and scalable distributed and cooperative intrusion detection architecture for WMNs capable of detecting a broad range of attacks.

In this work, we proposed an efficient distributed intrusion detection approach suitable for WMNs. The intrusion detection process involves the tasks of data collection and traffic

analysis in conformance with the protocol behavior. The proposed intrusion detection mechanism relies on non-intrusive traffic monitoring at each node for generating Routing Events. IDSs individually installed at each node monitor independently and simultaneously their local neighborhoods over time and establish thresholds to distinguish malicious behavior from failures of packet transmission, such as packet loss and packet corruption. Our approach uses collaborative intrusion detection techniques to detect more accurately a wider set of attacks. Since detecting routing attacks involves intense collaboration among IDS instances, we defined a consensus mechanism that merges observations from multiple nodes in order to make a proper decision that is based on different points of view of traffic behavior, and finally formulates a consensual verdict about a suspicious node.

We proposed a Routing Protocol Analyzer (RPA) which inspects each routing packet passing through the node interface and produces respective Routing Events according to the protocol behavior. Then, we implemented a communication mechanism for the IDS that allows neighboring nodes to actively exchange local generated Routing Events for distributed analysis of the neighborhood routing behavior. This regular exchange of Routing Events between nodes assures a reliable distribution of routing information among nodes. In addition, our IDS approach is capable of tolerating loss of messages between two communicating IDSs while maintaining a good intrusion detection accuracy.

We then proposed a Distributed Intrusion Detection Engine (DIDE) divided into intrusion detection modules that are in charge of detecting general attacking methods that we studied in this work. Each intrusion detection module processes the Routing Events, generated by the node and by the neighbors, relying on Routing Constraints to calculate Misbehaving Metrics, which designate the existence of malicious traffic in the vicinity. We defined Routing Constraints which allows detecting a considerable number of intrusions to WMNs, which are based on generic routing attacking methods of WMNs. We identified the most common attacking methods utilized by insider attackers through systematic analysis of attacking techniques of WMNs. This allows the IDS to not only examine the routing packet content, but also understand the context of message diffusion of the routing protocol, then greatly reducing the number of false positives.

We proposed a threshold approach that results in significant improvement in both the intrusion detection rate and false positive rate compared to traditional intrusion detection

approaches. The threshold takes into account the link quality in terms of packet loss rate for defining an appropriate threshold level according to the traffic conditions on the node.

We also designed a Cooperative Consensus Mechanism (CCM) that is responsible for analyzing in detail Misbehaving Metrics provided by DIDE component but also taking into consideration the Misbehaving Metrics calculated by the neighbors and respective thresholds. As a result, CCM module outputs a verdict on the presumed malicious behavior. But before that, it collects individual decisions from neighboring nodes, which are explicitly made based on their current vision of the network traffic behavior, in order to form a final collective decision by applying consensus decision-making process. We also implemented a decision rule to resolve impasses during the consensus process, which basically considers the decision of the majority since it copes with the concerns of all group members and it is considered the best possible decision for the group.

We then proposed a Response Mechanism in which the IDS initiates a passive response procedure in case intrusions are detected with consensual decision. The response policy script raises alerts and properly notifies the security administrator of the network. The IDS does not apply active responses since it leaves this task of mitigating or blocking the attack to the network administrator. We implemented a logging scheme in the attack detection script which generates output files recording the malicious activity observed on the network traffic. In addition, we implemented response policy scripts that execute programs according to the response policy, which in turn send e-mail messages automatically to end-users and network administrators.

The primary way of measuring the IDS efficiency is the number of low false positives and false negatives. Our intrusion detection solution diagnoses routing attacks in WMNs effectively and efficiently in real-time with practical assumptions and conditions, then providing good performance. We demonstrated through implementation that our intrusion detection method achieves high detection rate with reduced false positive rate and no false negatives for different types of routing attacks. In particular, we implemented malicious scenarios for emulating message fabrication attacks and packet dropping attacks. Our experimental results show that the intrusion detection approach is able to precisely identify attacker nodes in neighborhoods. The misbehaving attacks are implemented in a virtual mesh network platform that we developed specifically for testing and evaluating the

performance of security solutions for WMNs. Moreover, we analyzed the performance and discussed the security properties of BATMAN proactive routing protocol, which was employed to exemplify the real use of our distributed intrusion detection architecture. Furthermore, we presented a detailed analysis on the impact of routing attacks on the routing service of mesh nodes running BATMAN routing protocol.

Since every security solution comes with a cost, for achieving high detection accuracy for most of the attacking methods of WMNs, our intrusion detection solution introduces a reasonable message overhead to the communication system since frequent exchange of Routing Events is necessary to detect such attacks in a distributed manner. However, our intrusion detection approach incurs small computational overhead, i.e., CPU and memory overhead, during the intrusion detection process. In that way, an appropriate balance between security and overhead should be found for the WMN, i.e., how much of bandwidth overhead the deployed mesh network is willing to sacrifice for executing an efficient intrusion detection solution in terms of intrusion detection rate and diversity of attacks. We analyzed the security and overhead of the IDS approach and proved that in most of the cases it is able to identify and track the attacker node on every occurrence of routing misbehavior. Our intrusion detection solution showed to be scalable.

In particular, we have identified the importance of deploying a distributed and cooperative IDS for WMNs. However, significant progress may not be expected until WMNs are widely deployed in the field and are actively utilized by community wireless networks. At the present time, attack and intrusion detection in this heterogeneous environment are more speculative and theoretical. Nonetheless, further real experience with WMNs will undoubtedly accelerate research progress on this area.

6.2 Future work

There are a number of open problems and future research directions for this work. Throughout the thesis development, we have discussed the limitations of our distributed intrusion detection solution which could surely evolve to new approaches. The following are some suggestions for the development of future work.

Implementation of an Active Response Mechanism. IDS will not be very useful

without an effective active response scheme which executes punctual reactions, i.e., corrective actions. Active responses attempt to extenuate the effect of intrusions on the network and interrupt the attack in progress. Active response mechanism is divided into two groups: (i) collective responses: those that try to protect the target nodes, and (ii) individual responses: those that try to control over the source of the attack, i.e., the attacker node. The first one tries to recuperate the damaged system by killing processes, terminating network connections, or filtering the traffic. The second one tries to isolate the attacker and prevent future attacking attempts by denying network resources to the malicious node. We envisage to develop a cooperative intrusion response mechanism that will precisely recover the disrupted routes by repairing compromised routing messages based on specific detection constraints defined according to the routing protocol behavior. In a second approach, the IDS will interrupt the malicious traffic at the compromised node. The intrusion response mechanism will also consider dynamic thresholds to not cause damage to innocent nodes that could be mistakenly identified as malicious nodes.

Implementation of a Reputation Management Mechanism. In WMNs, nodes need to trust on each other since a node relies on intermediate nodes to reach other nodes in the network. Consequently, our IDS approach assumes that each participating node provides accurate routing information and cooperatively shares intrusion detection results and individual decisions with no malicious intention as specified by the intrusion detection architecture, i.e., each IDS is trusted. By exploiting this assumed trust, a malicious node can adulterate routing information messages, and detection results and decisions exchanged between IDSs to violate the cooperative intrusion detection mechanism. Therefore, a reputation management mechanism is strictly necessary for WMNs. We will implement a collaborative reputation scheme at each node which estimates trust values for neighboring nodes in order to deduct the node's reputation that represents the node behavior concerning its intrusion detection activities. Trustiness will be calculated by monitoring and analyzing the intrusion detection information and consensus outcomes provided by the node at regular periods, and by crosschecking and correlating these data with respective data received from neighboring reputation mechanisms at the neighbors. Then, each IDS keeps a list of trusty nodes to guarantee that only nodes having a certain level of trust, i.e., trustworthy nodes,

are authorized to exchange intrusion detection results and routing information with the IDS. This trust mechanism minimizes the risk of transmission of false intrusion detection information and inconsistent consensus messages.

Reduction of Communication Overhead. Usually, wireless networks are more constrained bandwidth than wired networks. IDS instances have to communicate with other IDS instances to exchange protocol events and intrusion detection information, or alerts and consensus information. Since this intense message exchange between IDSs could cause congestion in the communication channels and limit the network throughput, the IDS instances are required to reduce their data transfers. In addition, in wireless environments with restricted bandwidth, the IDS operations can become ineffective since many messages are lost. For instance, an IDS may not be able to detect an attack in real-time or at reasonable time due to communication delays between neighbors. Furthermore, IDS instances may get disconnected from the network due to the wireless link saturation. Therefore, minimizing the message overhead of the IDS communication system and its impact on the entire mesh network traffic is of extreme importance. We envisage reducing the communication overhead of our intrusion detection approach without affecting the accuracy of the distributed intrusion detection scheme and the cooperative mechanism, i.e., routing attacks should be detected with equal precision respecting the principles of distributiveness and cooperativeness introduced by the approach.

Introduction of Mobility Models. Although nodes in the WMN are quite static with restricted mobility, the mesh network architecture still supports mobiles nodes, such as smartphones, which connect and disconnect to the network in an arbitrary manner. Therefore, the IDS architecture has to take into account this premise when monitoring the communication links in order to ensure the efficiency of intrusion detection and avoid significant false positive rate, which can lead to inconsistent intrusion detections. We will integrate mobility models based on arbitrary movement of nodes, such as Random Waypoint model where mobile nodes move randomly and freely without restrictions, into our virtualized mesh network platform. Mobility models will be implemented by executing connection management scripts which will connect and disconnect the virtual links of the

nodes, i.e., the VMs, to simulated the node movement. For instance, when a node moves out the network transmission range and after a period of time it is attached to different nodes, then the script will emulate this movement behavior by terminating the actual virtual switch connections of the node, triggering a timeout, and then reconnecting the node virtual links to the new neighboring nodes.

Apply the Approach to WSNs and VANETs. Wireless Sensor Networks (WSNs) and Vehicular Ad-Hoc Networks (VANETs) are two other relevant research areas related to wireless networks that have recently attracted much attention. WSNs, VANETs, and WMNs have multiple similarities, however WSNs and VANETs present additional restrictions concerning the deployed environment for implementing security solutions. For instance, VANETs considers moving cars as nodes in a network in order to configure the mobile wireless network, then introducing the issue of node mobility. VANETs also considers the nodes, i.e., cars, are placed within a certain distance range in the deployed area and have the tendency to move in an organized fashion, then having specific mobility models and topology characteristics. In WSNs, sensor nodes have constrained hardware resources, such as memory, CPU, and communication bandwidth, and very restricted power, such as small batteries, which demands security solutions highly efficient in terms of bandwidth and computational overhead. In addition, WSNs introduce mobility of nodes, network topology changes, and the issue of network size, i.e., the network consists of several hundreds or even thousands nodes. Therefore, we will extend our distributed intrusion detection architecture to address intrusions in WSNs and VANETs. However, we should firstly integrate particular mobility models concerning these environments into our intrusion detection mechanism and correspondingly reduce the communication and computational overhead added by our IDS solution in order to cope with the resource limitations imposed by these network architectures.

References

[1] W. Zhang, Z. Wang, S. K. Das, and M. Hassan, "Security issues in wireless mesh networks," In Book Wireless Mesh Networks: Architectures and protocols. New York: Springer, 2008.

[2] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," Computer Networks, Vol. 47, pp. 445- 487, January 2005.

[3] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. BeldingRoyer, "A Secure routing protocol for ad hoc networks," In Proc of the 10th IEEE International Conference on Network Protocols (ICNP), November 2002.

[4] Y-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: a secure on-demand routing protocol for ad hoc networks," Wireless Networks, Vol. 11, No. 1-2, pp. 21-38, January 2005.

[5] M. G. Zapata, "Secure ad hoc on-demand distance vector routing," ACM Mobile Computing and Communications Review (MC2R), Vol. 6, No. 3, pp. 106-107, July 2002.

[6] IEEE P802.11 Task Group S, "Status of Project IEEE 802.11s" [Online], available at http://grouper.ieee.org/groups/802/11/Reports/tgs_update.htm.

[7] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich, "Better approach to mobile ad-hoc networking (B.A.T.M.A.N.)", April 2008, IETF Internet-Draft (expired October 2008) [Online], available at http://tools.ietf.org/html/draft-wunderlich-openmesh-manet-routing-00.

[8] OSSEC, "Open Source Host-based Intrusion Detection System" [Online], available at http:// http://www.ossec.net.

[9] A. Baker, J. Esler, and R. Alder, "Snort IDS and IPS Toolkit," Syngress, 2007.

[10] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," Computer Networks, vol.31, no. 23–24, pp. 2435-2463, 1999.

[11] Suricata, "Open Source Network-based Intrusion Detection System" [Online], available at http://www.openinfosecfoundation.org/.

[12] H. S. Javitz, and A. Valdes, “The SRI IDES Statistical Anomaly Detector,” In Proc of the IEEE Symposium on Research in Security and Privacy, 1991.

[13] P. Uppuluri, and R. Sekar, “Experiences with Specification-based Intrusion Detection,” In Proc of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID '00), pp. 172-189, 2001.

[14] C-Y Tseng, P. Balasubramayan, C. Ko, R. Limprasittiporn, J. Rowe, and K. Levitt, “A Specification-Based Intrusion Detection System for AODV,” In Proc of the 1st ACM workshop on Security of ad hoc and sensor networks (SASN '03), 2003.

[15] Yi-an Huang, and W. Lee, “Attack analysis and detection for ad hoc routing protocols,” In Proc of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID'04), 2004.

[16] J.-M. Orset, B. Alcalde, and A. Cavalli, “An EFSM-based intrusion detection system for ad hoc networks,” In Proc of the Third international conference on Automated Technology for Verification and Analysis (ATVA'05), pp. 400-413, 2005.

[17] K.A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R.A. Olsson, “Detecting disruptive routers: a distributed network monitoring approach,” IEEE Network, vol. 12, issue 5, pp. 50-60, Sept./Oct. 1998.

[18] Y. Zhang, W. Lee, and Y. Huang, “Intrusion detection techniques for mobile wireless networks,” Wireless Networks, Vol. 9, No. 5, pp. 545-556, September 2003.

[19] S. Marti, T.J. Giuli, K. Lai, and M. Baker, “Mitigating routing misbehavior in mobile ad hoc networks,” In Proc of 6th Annual International Conference on Mobile Computing and Networking (MobiCom '00), pp. 255-265, Boston, 2000.

[20] O. Kachirski, and R. Guha, “Effective Intrusion Detection Using Multiple Sensors in Wireless Ad Hoc Networks,” In Proc of the 36th Annual Hawaii International Conference on System Sciences (HICSS’03), p. 57.1, January 2003.

[21] D. Sterne, P. Balasubramanyam, D. Carman, B. Wilson, R. Talpade, C. Ko, R. Balupari, C-Y. Tseng, T. Bowen, K. Levitt, and J. Rowe, “A General Cooperative Intrusion Detection Architecture for MANETs,” In Proc of Third IEEE International Workshop on Information Assurance (IWIA '05), pp. 57–70, March 2005.

[22] H. Yang, J. Shu, X. Meng, and S. Lu, “SCAN: self-organized network-layer security

in mobile ad hoc networks," IEEE Journal on Selected Areas in Communications, vol. 24, no. 2, pp. 261–273, February 2006.

[23] S.A. Razak, S.M. Furnell, N.L. Clarke, and P.J. Brooke, "Friend-assisted intrusion detection and response mechanisms for mobile ad hoc networks," Ad Hoc Networks, vol. 6, issue 7, pp. 1151-1167, September 2008.

[24] N. Komninos, and C. Douligeris, "LIDF: layered intrusion detection framework for ad-hoc networks," Ad Hoc Networks, Volume 7, Issue 1, pp. 171-182, January 2009.

[25] Z. Zhang, P. Ho, and F. Naıt-Abdesselam, "RADAR: A reputation-driven anomaly detection system for wireless mesh networks," Wireless Networks, Vol. 16, Issue 8, pp. 2221-2236, November 2010.

[26] N. Saxena, M. Denko, and D. Banerji, "A hierarchical architecture for detecting selfish behaviour in community wireless mesh networks," Computer Communications, Vol. 34, Issue 4, pp. 548-555, April 2011.

[27] C. Popper, M. Strasser, and S. Capkun, "Jamming-resistant broadcast communication without shared keys," In Proc of the 18th conference on USENIX security symposium (SSYM'09), pp. 231-248, August 2009.

[28] A. Liu, P. Ning, H. Dai, Y. Liu, and C. Wang, "Defending DSSS-based broadcast communication against insider jammers via delayed seed-disclosure," In Proc of the 26th Annual Computer Security Applications Conference (ACSAC '10), pp. 367–376, December 2010.

[29] IEEE 802.11i-2004, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Security Enhancements" [Online], available at http://standards.ieee.org/getieee802/download/802.11i-2004.pdf.

[30] I. Khalil, S. Bagchi, and N.B. Shroff, "LiteWorp: Detection and isolation of the wormhole attack in static multihop wireless networks," Computer Networks, Vol. 51, Issue 13, pp. 3750-3772, September 2007.

[31] Y. Wang, Z. Zhang, and J. Wu, "A Distributed Approach for Hidden Wormhole Detection with Neighborhood Information," In Proc of the IEEE Fifth International Conference on Networking, Architecture, and Storage (NAS '10) , pp. 63-72, 2010.

[32] J. Friginal, D. Andrés, J. Ruiz, and P. Gil, "Towards benchmarking routing protocols in wireless mesh networks," Ad Hoc Networks, Vol. 9, Issue 8, pp. 1374-1388,

November 2011.

[33] M. Abolhasan, B. Hagelstein, J. C.-P. Wang, “Real-world performance of current proactive multi-hop mesh protocols,” In Proc of 15th Asia-Pacific conference on Communications (APCC'09), pp. 42-45, 2009.

[34] BATMAN ELP protocol, “BATMAN Echo Location Protocol (ELP)” [Online], available at http://www.open-mesh.org/projects/batman-adv/wiki/ELP.

[35] D. Johnson, N. Ntlatlapa, and C. Aichele, “A simple pragmatic approach to mesh routing using BATMAN”, In 2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries, pp. 10, October 2008.

[36] Open-Mesh.net, “B.A.T.M.A.N. (better approach to mobile ad-hoc networking)” [Online], available at http://www.open-mesh.net/.

[37] QEMU, “Machine emulator and virtualizer” [Online], available at http://wiki.qemu.org.

[38] VDE switch, “Virtual Distributed Ethernet switch” [Online], available at http://wiki.virtualsquare.org/wiki/index.php/VDE.

[39] PackETH, “Ethernet packet generator” [Online], available at http://packeth.sourceforge.net/sourceforge/Home.html.

[40] Tim Hartnett, “Consensus-Oriented Decision-Making: The CODM Model for Facilitating Groups to Widespread Agreement”, New Society, 2011.

[41] M. O. Pervaiz, M. Cardei, and J. Wu, “Routing Security in Ad Hoc Wireless Networks,” Network Security, pp. 117-142, Springer 2010.

[42] Bro, “Network Security Monitor” [Online], available at http://bro-ids.org/.

[43] Sysstat, “Performance monitoring tools for Linux” [Online], available at http://sebastien.godard.pagesperso-orange.fr/.

[44] Bro communication issue, “High CPU usage of Bro” [Online], available at http://mailman.icsi.berkeley.edu/pipermail/bro/2012-February/005338.html.

[45] Netem, “Network Emulation” [Online], available at http://www.linuxfoundation.org/collaborate/workgroups/networking/netem/