Received 27 August 2015
Revised 11 November 2015
Accepted 11 November 2015

An efficient intrusion detection and prevention framework for *ad hoc* networks

Abdelaziz Amara Korba and Mehdi Nafaa
Department of Computer Science, Networks and Systems Laboratory (LRS), Badji Mokhtar-Annaba University, Annaba, Algeria, and
Salim Ghanemi
Department of Computer Science, Embedded Systems Laboratory (LASE), Badji Mokhtar-Annaba University, Annaba, Algeria

Abstract
Purpose – Wireless multi-hop *ad hoc* networks are becoming very attractive and widely deployed in many kinds of communication and networking applications. However, distributed and collaborative routing in such networks makes them vulnerable to various security attacks. This paper aims to design and implement a new efficient intrusion detection and prevention framework, called EIDPF, a host-based framework suitable for mobile *ad hoc* network's characteristics such as high node's mobility, resource-constraints and rapid topology change. EIDPF aims to protect an AODV-based network against routing attacks that could target such network.
Design/methodology/approach – This detection and prevention framework is composed of three complementary modules: a specification-based intrusion detection system to detect attacks violating the protocol specification, a load balancer to prevent fast-forwarding attacks such as wormhole and rushing and adaptive response mechanism to isolate malicious node from the network.
Findings – A key advantage of the proposed framework is its capacity to efficiently avoid fast-forwarding attacks and its real-time detection of both known and unknown attacks violating specification. The simulation results show that EIDPF exhibits a high detection rate, low false positive rate and no extra communication overhead compared to other protection mechanisms.
Originality/value – It is a new intrusion detection and prevention framework to protect *ad hoc* network against routing attacks. A key strength of the proposed framework is its ability to guarantee a real-time detection of known and unknown attacks that violate the protocol specification, and avoiding wormhole and rushing attacks by providing a load balancing route discovery.

Keywords Computer security, Communications technology, Computer networks, Computer privacy, Communications industry

Paper type Research paper

1. Introduction
Wireless multi-hop *ad hoc* networks include a wide range of networking paradigms such as mobile *ad hoc* networks (MANETs), vehicular *ad hoc* networks and opportunistic and sensor networks. In multi-hop *ad hoc* networks, each node acts as a router to forward packets between nodes which are not within communication range of each other. Multi-hop *ad hoc* networks are useful in many critical applications such as emergency rescues, pollution monitoring and military applications. Unfortunately, most of the *ad hoc* routing protocols have no security considerations, and they operate on the

assumption that all nodes are friendly and participate correctly in relaying routing and data packets. Multi-hop *ad hoc* networks are often secured by using either cryptography or intrusion detection systems (IDSs). Cryptographic approaches can protect *ad hoc* networks against external attackers using node authentication and data encryption. However, these techniques cannot prevent insider attacks, consumes considerable resources and have their associated problems such as key issuing and management (Mulert *et al.*, 2012).

Intrusion detection and prevention mechanisms can detect malicious activities performed by external or internal attackers, by monitoring and analyzing network activities. The existing IDS architectures for *ad hoc* networks can be classified into stand-alone, cooperative and hierarchical. In stand-alone architecture, every node in the network performs detection based on its local data using an IDS agent installed on it. In cooperative architecture, each node has an IDS agent that communicates and collaborates with other nodes' agents, forming a global intrusion detection to resolve inconclusive detections. Hierarchical IDS is a sort of cooperative architecture suited to multi-layered networks. In this architecture, the network is divided into clusters, where some nodes are selected as cluster heads to undertake more responsibility than other cluster members. Each cluster member performs local detection, while cluster heads perform global detection (Sen, 2010).

In this paper, we propose a taxonomy of routing attacks and conduct attack analysis using a semi-formal method attack tree that decomposes attacks into a set of basic attacks. Based on the analysis, we propose an intrusion detection and prevention framework composed of an IDS agent, load balancer and intrusion response mechanism. We take one of the most popular routing protocols, AODV, as a case study. We develop specification model in the form of a set of possible interactions that can be performed by a node from AODV RFC (Perkins *et al.*, 2003). Based on the specification model, we build a specification-based IDS agent to detect specification violation attacks. We propose a load balancing AODV route discovery to avoid and prevent attacks that do not violate the specification such as wormhole and rushing attacks. An adaptive response mechanism is invoked each time an intrusion is detected. Simulation results show that our intrusion detection and prevention framework outperforms other schemes proposed in the literature in terms of number of detected attacks, detection rate and false positive rate.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents a taxonomy and analysis of typical MANET routing attacks. In Section 4, we introduce our proposed framework. Section 5 describes the details of our proposed framework, efficient intrusion detection and prevention framework (EIDPF). In Section 6, we provide ns-2 simulation results and performance analysis. Finally, we conclude the paper in Section 7.

2. Related work

According to Nadeem and Howarth (2013b)), the security mechanisms proposed in the literature to protect the network against routing attacks can be divided into two categories. The first category is point detection algorithms which can detect a single type of routing attacks, such as Su (2010, 2011), Gonzalez-Duque *et al.* (2008), Medadian *et al.* (2009) and Zhang *et al.* (2009). The second is IDS which can detect a range of attack types, such as Tseng *et al.* (2003), Panos *et al.* (2010, 2014), Shakshuki *et al.* (2013), Nadeem and Howarth (2013a, 2014) and Alattar *et al.* (2012). As the proposed mechanism in this paper is an intrusion detection framework, only researches belonging to the second category are introduced. Before we discuss the proposed systems in the literature, we start by reviewing the three main intrusion detection techniques.

The first technique is anomaly based intrusion detection which defines the normal behavior of the system using classification and statistical methods. It detects any anomalous observed activity that deviates significantly from the normal behavior as intrusion. The advantage of this technique is its ability to detect unknown attacks; however, it generates high false positives rate and needs high computational cost. Signature-based detection compares current system activities with signatures or patterns of known attacks. It is reliable and has low false positive rate; however, it cannot detect new attacks. Specification-based detection specifies the normal behavior using a set of rules and constraints and detects intrusions as runtime violations of the specification. It generates low false positive rate and can detect unknown attacks; however, defining specification is a time-consuming process.

Tseng *et al.* (2003) proposed the first specification-based IDS in MANETs. They use distributed network monitors (NM) which are assumed to cover all nodes to detect run-time violation of the specifications. NMs use finite state machines (FSM) to define the correct AODV routing behavior. Each route request (RREQ) and route reply (RREP) message in the range of the NM is monitored in a request-reply flow. NMs need to exchange information about previous messages or particular nodes, especially in networks with high mobility and substantially dynamic topologies. Although this approach can detect both known and unknown attacks against routing protocols, exchanging information between monitoring nodes impose a significant overhead in high mobility conditions. Furthermore, the authors have assumed that monitors can cover all network nodes and have all nodes' IP and MAC addresses, which is not realistic. In addition, monitor nodes need to stay permanently in a promiscuous mode, which can consume significant energy.

Panos *et al.* (2010) proposed an IDS integrating a random walk-based IDS architecture and a multi-layer specification-based detection engine to monitor the transport, network and data link layers of the protocol stack. A set of self-contained random walk detectors (RWDs) randomly move around the network from node to node, to monitor node's behavior, and detect possible attacks that take place in the visited node. RWDs can detect specification violation attacks at multiple layers. However, the migration process of RWD induces significant data transmission and extra communication overhead which increases by incrementing the number of RWD. The proposed protocol specifications are incomplete. Furthermore, the authors do not consider drastic consequences that may result from letting nodes without a protection for a time while RWDs are visiting other nodes.

Panos *et al.* (2014) proposed a specification-based IDS called SIDE to monitor the behavior of hosting node. SIDE monitors protocol operations in real time, through the use of a FSM which defines the legitimate functionality of the AODV protocol. To protect the IDS from attacks carried out by malicious host nodes, the authors proposed a remote attestation procedure which checks the integrity of running SIDE instances in the network. Furthermore, SIDE runs on a trusted computing platform which provides hardware-based root of trust and cryptographic acceleration to provide resilient IDS. SIDE can detect specification violation attacks in real time and with high detection accuracy. However, it relies on hardware support and uses cryptographic and authentication functions which are very expensive in terms of resource usage.

Shakshuki *et al.* (2013) proposed a new IDS called enhanced adaptive acknowledgment (EAACK). It is specially designed for MANETs to tackle weakness of watchdog scheme. EAACK is an acknowledgement-based scheme requiring end-end acknowledgement for every packet sending and uses digital signature to guarantee the validity and authenticity of the acknowledgment packets. EAACK demonstrates high detection rates against a particular class of attacks. However, it generates a significant overhead (acknowledgement packets), and it is not able to detect unknown attacks, or even most well-known attacks such as flood and black hole attacks. Furthermore, although the authors have raised the question associated to the extra cost induced by digital signature in terms of resources usage, they do not propose any solution to minimize it.

Nadeem and Howarth (2014) proposed a cluster-based intrusion detection and adaptive response (IDAR) mechanism, which is an extension of their previous proposal generalized intrusion detection and prevention mechanism (Nadeem and Howarth, 2013a). IDAR combines signature-based and anomaly based techniques. In the first phase, a cluster head gathers audit data from network nodes and then uses collected data to build training profiles. Finally, the testing module is launched periodically to detect possible intrusion and identify attacks and intruders. The IDS takes action once the attack is occurred, and it is not able to prevent its occurrence. Continuous data gathering, repeated training, attack inference and knowledge base management are time-, bandwidth- and resource-consuming tasks. A trade-off should be made between workload, classification accuracy and energy consumption. Furthermore, constructing and adding a rule for the new attacks is prone to generate false attack signatures.

Alattar *et al.* (2012) proposed IDAR, a signature-based distributed intrusion detection based on OLSR (Clausen and Jacquet, 2003). IDAR extracts evidence from OLSR-collected logs, and according to the activity's suspicion level, it initiates an in-depth cooperative investigation to confirm intrusion. To identify patterns of attacks, IDAR compares logs with a set of predefined signatures, where a signature is defined as a partially ordered sequence of events that characterizes a malicious activity. Although IDAR demonstrates high detection and low false positives rate, it can only detect specific attacks present in its signature database. In addition, IDAR is vulnerable to malicious node that may transmit deceptive opinion during the investigation process (blackmail attack). Furthermore, tasks such as collecting and analyzing logs consume significant resources (memory and bandwidth).

Different techniques have been used to implement anomaly based IDS such as those proposed in Mitrokotsa and Dimitrakakis (2013), most of them are based on statistical approaches and artificial intelligence methods. Jabbehdari *et al.* (2012) proposed an IDS based on neural networks to detect DoS attacks in MANETs. Barani *et al.* (2012) proposed an anomaly based IDS named BeeID which can detect a wide range of attacks using a hybrid approach based on the artificial bee colony (ABC) and negative selection algorithms.

3. Analysis and classification of routing attacks

To design an efficient security mechanism, a deep knowledge of attack patterns is required. To facilitate the analysis, we classify routing attacks into two categories: basic and compound attacks.

3.1 Basic attacks

A basic attack can be defined as an indivisible succession of routing operations that does not follow the routing protocol specification. However, there are some attacks in the literature that do not violate the routing protocol specification, but violate other layer's protocol specification, such as wormhole and rushing attacks. Therefore, we classify basic attacks into two categories.

3.1.1 Specification violation attacks

- *Drop*: The malicious node illegitimately drops the received routing or data message.
- *Modification*: Modifying non-mutable fields, or incorrectly modifying mutable fields of the routing message.
- *Fabrication*: Generating and sending forged routing message with erroneous routing information.
- *Replay*: Non authorized resend of routing messages.

Impersonation can be seen as the modification of the source node identity (IP header), or fabrication of new identity. In some cases, malicious node may use multiple forged IP addresses, which is known in the literature as Sybil attack (Nadeem and Howarth, 2013b).

3.1.2 Fast forwarding attacks

- *Wormhole*: This is usually carried out by two colluding malicious nodes situated at different locations within the network; when one malicious node receives routing messages, it forwards them only to the second one through a high-quality out-of-band link called tunnel, to invade discovered routes and attract traffic. This attack is performed in the network layer, but the specification violation is carried out within lower layers (physical and MAC layer) (Mulert *et al.*, 2012).
- *Rushing*: This attack exploits the property of on demand protocols which requires nodes to forward only the first RREQ to minimize the routing overhead. The rushed RREQ reaches the other nodes before the legitimate one, leading to discarding legitimate RREQs when they arrive later. To disseminate the RREQ faster than others, the malicious node either ignores exponential Backoff and interframe spacing specified by MAC layer protocol (IEEE 802.11) or performs a high power transmission (physical layer) of the routing message (Mulert *et al.*, 2012).

3.2 Compound attacks

Combination of diverse basic attacks, or repetition of one basic attack, to perform more sophisticated attack which makes more powerful and persistent impacts in the network. In this paper, we use the semi-formal method attack tree first proposed in Schneier (1999) and then applied for MANET routing in Ebinger and Bucher (2006) to analysis and classify compound routing attacks. The attack is graphically represented by a tree (Figure 1), the root represents the primary goal of the attack. The branches represent the sub-goals of the attack or sub attacks which constitute the preconditions to accomplish the primary goal. Sub-goals are connected by logical "OR" or "AND", leafs represent basic attacks (indivisible attacks), as shown in Figure 1. We describe below well-known attacks in the literature.

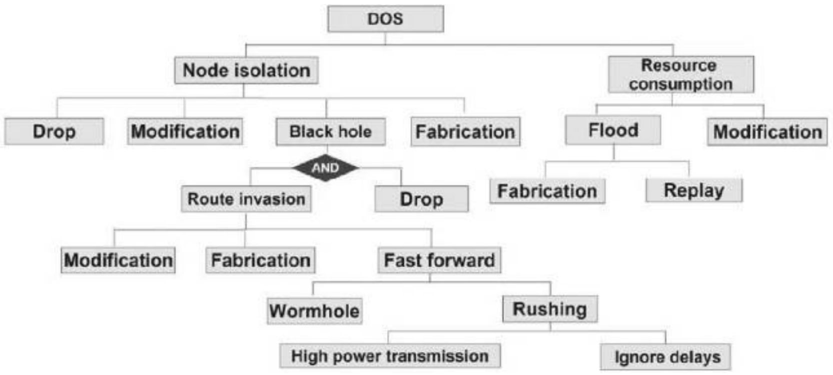


Figure 1.
DOS attack tree

3.2.1 Route invasion. Consists on disseminating attractive routing information by malicious node to be part of discovered routes. To achieve this goal, malicious node illegitimately modifies the received routing message or fabricates new ones. Another efficient technique to invade route is to fast forward the received messages to first reach the destination.

3.2.2 Node isolation. To deny a node(s) from using network services, malicious node may modify routing messages originated from or destined to that victim node with erroneous routing information. The malicious node can disseminate erroneous routes by fabricating fake routing message with incorrect routing information. If malicious node is the gateway of the victim node to the network, it can simply drops all packets received from or destined to the victim node.

3.2.3 Resources consumption (or sleep deprivation). To consume network bandwidth and nodes' resources, malicious node can either perform a flood attack, which is a repetitive replaying and fabrication of routing and data packets, to consume network resources, or modify particular fields in routing message to create loops, or just to make packets endlessly circulate in the network (Nadeem and Howarth, 2013b).

3.2.4 Black and gray hole. In this attack, malicious node disseminates attractive routing information to invade routes [route invasion (RI)], once in the route it drops all the received data packets or particular routing messages. In a gray hole attack, a sophisticated version of black hole, malicious node performs selective dropping to defeat security mechanisms (Nadeem and Howarth, 2013b).

3.2.5 Denial of service and distributed denial of service. Wood and Stankovic in Cayirci and Rong (2008) define DoS as any event that diminishes a network's capacity to perform its expected function correctly or in a timely manner. DoS targets the availability of routing services provided by the network. To achieve DoS malicious node either isolates the victim node(s) from the network to deprive it of the services, or consume its energy to disable it, in a way that become unable to use network services. The effect of this attack can be more important and cause more damage, when multiple attackers collude together to launch DoS in a coordinated way, known as Distributed Denial of Service (DDoS) attack (Mulert *et al.*, 2012). Figure 1 shows attack tree of Denial of service attack, as well as all its sub attacks such as node isolation (NI), sleep deprivation, black hole [. . .].

4. Proposed protection mechanism

Designing protection mechanisms for MANETs is a challenging task, considering limitations of the existing IDSs (i.e. analyzed in Section 2). We think limitations of IDSs architectures such as audit data collection and extra overhead communication can be resolved by host-based standalone architecture. We believe more effort is needed on developing intrusion detection mechanisms which can guarantee real-time detection of various attacks, particularly for reactive routing protocols due to their frequent usage. Considering the limitations of misuse and anomaly detection techniques, we believe that a specification-based detection technique provides a trade-off solution that balances the strengths and weaknesses of both techniques.

4.1 Architecture

In this paper, we propose a standalone host-based intrusion detection and prevention framework implemented at each node. The protection framework is composed of three components:

1. intrusion detection and prevention module;
2. load balancer; and
3. intrusion response module as shown in Figure 3.

The detection and prevention module consists on specification-based IDS agent to monitor the interactions of the hosting node with the other nodes and to detect specification violation attacks. The statistical-based load balancer monitors the route selection process and avoids fast-forwarding attacks. The response mechanism is trigged by the IDS agent each time an intrusion is detected. Its function is to take the suited defensive action against the attacker.

4.2 Specification-based intrusion detection system agent

The specification-based IDS is composed of two components: the specification model and monitoring mechanism. The specification model defines the valid behavior of routing protocol. Specification-based IDS monitors the interactions with other nodes according to the specification model using local routing information that we call as local knowledge saved mostly within the routing table. Any interaction inconsistent with the protocol specification is detected as a run-time violation of the specification.

4.2.1 Specification model. The correct operation of routing protocol is well defined in protocol specification documents such as RFCs. However, extracting complete and correct specification model is a time-consuming and challenging task. The extraction process is facilitated by taking advantage of the commonality of *ad hoc* routing protocols. We can distinguish two types of routing protocol operations. Internal operations affect only the local knowledge of the node, such as adding or updating a route within the routing table. External operations or interactions which can be defined as any action visible by other nodes in the vicinity and can affect their local knowledge or incite them to launch other interaction(s). We model routing protocol specification as a set of all possible interactions which can take place between two nodes in the network (Figure 2).

In our model, we only consider interactions because internal operations are not visible and therefore cannot be monitored. Each time a node receives a routing message, IDS agent checks the coherence of the ongoing interaction to the protocol specification.

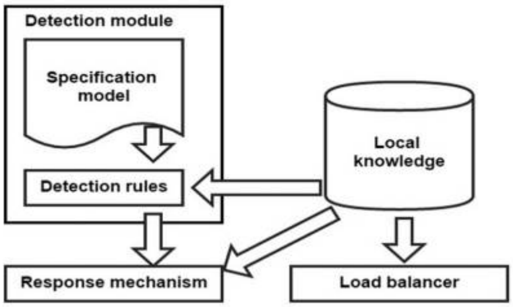


Figure 2.
EIDPF architecture

The protocol specification can be defined as a tuple $\boldsymbol{SM}(\boldsymbol{I},\boldsymbol{R},\boldsymbol{M})$: $\forall \boldsymbol{i} \in \boldsymbol{I}$, $\exists\, \boldsymbol{r} \in \boldsymbol{R}$, $\exists\, \boldsymbol{m} \in \boldsymbol{M}$, where **SM** stands for specification model, $\boldsymbol{I}$ represents the set of interactions between two nodes in the network and $\boldsymbol{R}$ represents the set of requirements a sender must satisfy to launch the corresponding interaction. $\boldsymbol{M}$ represents the message format which defines restrictions on critical message fields. If we take as an example RREQ-sending interaction, the sender must respect a set of requirements such as not exceeding the maximum number of route discovery attempts and waiting for a timeout between consecutive attempts. Also, the RREQ message must carry out a new RREQ ID and sequence number.

The generated model which is a set of interactions is easily transformable to detection rules to efficiently detect protocol misuses as shown by our experiments in Section 5.1. To detect specification violations, the IDS agent tries to detect inconsistencies between the information in its interaction history and the messages it receives using the detection rules.

4.3 Load balancer

To avoid fast forwarding attacks such as rushing and wormhole, we add a load balancer module to each node. The principle of load balancer is to enable neighboring nodes of a fast-forwarding node to detect that malicious node has a high capacity of competition in route discovery. The goal of the load balancer is to select routes that do not pass through loaded neighbor nodes, which have a route-discovering rate higher than the threshold. Therefore, traffic concentration points will be eliminated from the network, and fast-forwarding nodes are gradually identified and isolated. Even though a legitimate node may be placed at a key location of connectivity in a network, and thus be isolated because of its high route-discovering rate. It would not stay on the same location for long, as the network topology is dynamic. Furthermore, the proposed load balancer allows the recovery of isolated node that behaves normally.

4.4 Response mechanism

Each time an intrusion (specification violation) is detected, the response mechanism immediately punishes malicious node by completely isolating it from the network. The malicious node is simply treated as non-existent. To minimize the negative impact or the adverse effect of isolation on the network operations, we use an isolation scheme with different isolation periods that consider repeated intrusions so as to isolate recidivist nodes for a longer period.

5. Efficient intrusion detection and prevention framework

5.1 Specification-based intrusion detection system

Notation and abbreviation: The notation in Table I will be used in the rest of the paper

5.1.1 Specification of the AODV protocol. Within an *ad hoc* network routed by AODV protocol, we can identify four broad possible interactions ***I*** = {***RREQ sending***, ***RREP sending***, ***RERR sending***, ***hello sending***}. Each interaction corresponds to a routing message, we do not consider Route Reply Acknowledgment (RREP-ACK) message because it does not transport any routing information that can update the receiving node's local knowledge or incite it to launch other interaction.

5.1.1.1 Interaction 1: route request sending. A node sends an RREQ when it determines that it needs a route to a destination and does not have one available (RREQ generation), or to forward a received RREQ:

(1) *Requirements*: Whenever a node generates RREQ, it must respect the following set of requirements (R)
   - *The requested destination must be unknown*: It may also be expired or marked as invalid within the routing table
   - *The number of RREQ attempts* must not exceed the maximum number of attempts (RREQ_RETRIES).
   - *Dissemination range* AODV uses expanding ring search technique to avoid wide dissemination of RREQ through the network. Initially, the originator node uses a TTL_START as TTL_value in the IP header and sets the timeout for receiving a RREP to RING_TRAVERSAL_TIME milliseconds, which is calculated based on TTL_value. If the corresponding RREP is not received within that timeout, a new RREQ with a TTL_incremented by TTL_INCREMENT is broadcasted by the originator. The same process continues till the TTL_value reaches TTL_THRESHOLD, beyond which a NET_DIAMETER is used for each further attempt. Therefore, we can represent the expanding ring search process by the equation (1) (**n** represents the number of attempts):

$$\forall(\boldsymbol{n} \in \boldsymbol{N}) \wedge (\boldsymbol{n} \geq 1) \wedge \boldsymbol{TTL\_value} < \textit{Threshold}, \quad \boldsymbol{TTl\_value} = \boldsymbol{TTL\_start} + (\boldsymbol{n} - \mathbf{1})\boldsymbol{TTL\_increment} \tag{1}$$

Table I. Notation and abbreviation

Notation	Definition
RTE	Route table entry
BID	Broadcast ID
orig	Originator address
dst	Destination address
dst_seq	Destination sequence number
ori_seq	Originator sequence number
ip_src	IP source address on the IP header
ip_dst	IP destination address on the IP header
TTL	Time to live
rcv_	Received
New	New value of the following field
TTL_initial	A set of TTL values calculated by originator node

- *Interval between RREQ attempts*: After sending an RREQ, the node must wait for an RREP a timeout. If an RREP is not received within that timeout, the originator node may send another RREQ, up to a maximum of RREQ_RETRIES [timeout is calculated as described in eqation (2)]. When a TTL is set to Network_diameter, AODV uses a binary exponential Backoff to diminish congestion in network. After the second attempt with TTL equal to Network_diameter, the waiting time is multiplied by two, and so on for each further attempt m. Therefore, when Backoff is used, we can calculate timeout by equation (3):

$$\begin{gathered} \boldsymbol{TTL} < \mathit{Threshold}, \\ \boldsymbol{Timeout} = 2 \times \boldsymbol{node\_traversal\_time} \times (\boldsymbol{TTL\_value} + \boldsymbol{timeout\_buffer}) \end{gathered} \tag{2}$$

$$\begin{gathered} \boldsymbol{TTL} \geq \boldsymbol{Threshold}, \forall (\boldsymbol{m} \in N) \wedge (\boldsymbol{m} \geq 1) \\ \boldsymbol{Timeout_m} = 2^m \boldsymbol{node\_traversal\_time} \times (\boldsymbol{Network\_diameter} + \boldsymbol{timeout\_buffer}) \end{gathered} \tag{3}$$

- *RREQ forwarding*: Whenever a node originates, a new RREQ, the hop count field is set to zero and TTL in the IP header is set to TTL_value. Each time an RREQ is forwarded through the network, the TTL_value in the outgoing IP header is decreased by one, whereas the hop count value in the RREQ message is incremented by one, to account for the new hop through the intermediate node. Therefore, the sum of hop count and TTL values of the received RREQ will be always equal to the initial TTL_value set by the originator node. Therefore, by knowing the number of RREQ attempts **n**, receiving node can determine exactly the initial value of TTL owing to equation (4), and therefore it can detect incorrect values of hop count field. Even if the number of RREQ attempts n is unknown, receiving node can detect incoherencies resulting from hop count manipulation by checking equations (5) or (6):

$$\begin{gathered} ((\mathit{RREQ.HC} + \mathit{RREQ.TTL} < \mathit{TTL\_threshold})\ \mathit{is\ True}) \wedge (\exists n \in \mathit{reference\_knowledge}) \\ \Rightarrow ((\boldsymbol{RREQ.HC} + \boldsymbol{RREQ.TTL}) - \boldsymbol{TTL\_start})/\boldsymbol{TTL\_increment}) + 1 = \boldsymbol{n} \end{gathered} \tag{4}$$

$$\begin{gathered} ((\mathit{RREQ.HC} + \mathit{RREQ.TTL}) - \mathit{TTL\_start})/\mathit{TTL\_increment}) + 1 \in N \\ \Leftrightarrow ((\boldsymbol{RREQ.HC} + \boldsymbol{RREQ.TTL}) - \boldsymbol{TTL\_start})\ \%\ \boldsymbol{TTL\_increment} = \boldsymbol{0} \end{gathered} \tag{5}$$

$$\begin{gathered} (\mathit{RREQ.HC} + \mathit{RREQ.TTL} \geq \mathit{TTL\_threshold})\ \mathit{is\ true} \\ \Rightarrow \boldsymbol{RREQ.HC} + \boldsymbol{RREQ.TTL} = \boldsymbol{Network\_diameter} \end{gathered} \tag{6}$$

(2) *RREQ format*: There are two possible set of format restrictions There are two possible set of format restrictions, the first one (RREQ_f1) corresponds to an RREQ received from originator node, and the second (RREQ_f2) corresponds to forwarded RREQ. Expressions (1) and (2) summaries interaction requirements and format restrictions of RREQ generation and RREQ forward.

$$\boxed{\begin{array}{l} \textbf{\textit{if}}\ \Big((\exists\ \mathit{data\ packets\ to\ send}) \wedge \Big((RTE_{RREQ.dst} \notin RT) \vee \Big((RTE_{RREQ.dst} \in RT) \wedge (RTE_{RREQ.dst}.\mathit{stat} = \\ \quad \mathit{down}) \Big) \Big) \wedge (RREQ\_atemp < RREQ\_retries) \wedge (RREQs\_interval \geq Timeout) \Big) \quad \textbf{\textit{then}} \\ \quad \boldsymbol{RREQ\_f1\,\{HC = 0, NewBID, New\ orig\_seq, TTL \in TTL\_initial, ip\_src = orig\}} \end{array}}$$

Expression 1 RREQ generation

$$\boxed{\begin{array}{l} \textbf{\textit{if}}\ (((\mathit{rcv\ New\ RREQ}) \wedge ((RTE_{RREQ.dst} \notin RT) \vee ((RTE_{RREQ.dst} \in RT) \wedge (RTE_{RREQ.dst}.dst\_seq < \\ RREQ.dst\_seq)) \vee ((RTE_{RREQ.dst} \in RT) \wedge (RTE_{RREQ.dst}.dst\_seq \geq \\ RREQ.dst\_seq)) \wedge (RREQ.D\_flag = 1))) \quad \textbf{\textit{then}} \\ \quad \boldsymbol{RREQ\_f2\,\{HC = rcv\_HC + 1, BID = rcv\_BID, dst = rcv\_dst, dst\_seq \geq rcv\_dst\_seq, orig\_seq} \\ \qquad\qquad \boldsymbol{= rcv\_orig\_seq, TTL = rcv\_TTL - 1, ip\_src \neq orig, ip\_src \neq dst\,\}} \end{array}}$$

Expression 2 RREQ forward

5.1.1.2 Interaction 2: route reply sending. A node sends a RREP if it is itself the destination, or it has an active route to the destination, the destination sequence number in the node's existing route table entry for the destination is valid and greater than or equal to the Destination Sequence Number in the RREQ, and the "destination only" ('D') flag is not set (Perkins *et al.*, 2003). Also a node sends a RREP to forward a received one from destination to originator. We describe RREP format restrictions (RREP_f1, RREP_f2, and RREP_f3) and their corresponding set of requirements in expressions (3), (4) and (5).

$$\boxed{\begin{array}{l} \textbf{\textit{if}}\ \big(((\mathit{rcv\ new\ RREQ}) \wedge (RREQ.orig \in RT) \wedge (my\_addr = RREQ.dst)) \big) \quad \textbf{\textit{then}} \\ \qquad \boldsymbol{RREP\_f1\,\{HC = 0, dst = my\_addr, dst\_seq \geq rcv\_RREQ.dst\_seq, orig} \\ \qquad\qquad \boldsymbol{= RREQ.orig, lifetime = My\_Route\_Timeout, ip\_dst} \\ \qquad\qquad \boldsymbol{= RTE_{orig}.NH, ip\_src = dst, TTL = NET\_DIAMETER\}} \end{array}}$$

Expression 3 RREP generation by destination node

$$\boxed{\begin{array}{l} \mathit{if}\ ((\mathit{rcv\ new\ RREQ}) \wedge (\mathit{rcv\ RREP}) \wedge (my\_addr \neq RREP.orig) \wedge (RREP.orig \\ \qquad \in RT) \wedge [\mathit{if}\ (\exists\, RTE_{RREP.dst}) \wedge (RREP.dst\_seq \\ \qquad \geq RTE_{RREP.dst}.dst\_seq)])\ \mathit{then} \\ \quad \boldsymbol{RREP\_f3\,\{HC = rcv\_HC + 1, dst = rcv\_dst, dst\_seq = rcv\_dst\_seq, orig} \\ \qquad\qquad \boldsymbol{= rcv\_orig, lifetime = rcv\_lifetime, ip\_src \neq orig, ip\_dst} \\ \qquad\qquad \boldsymbol{= RTE_{orig}.NH\}} \end{array}}$$

Expression 4 RREP generation by intermediate node

$$\textbf{\textit{if}}\ \big((rcv\ new\ RREQ) \wedge (RREQ.orig \in RT) \wedge ((RREQ.dst \in RT) \wedge (RTE_{dst}.dst\_seq \geq RREQ.dst\_seq)) \wedge (rcv\_RREQ.D\_flag \neq 1)\big)\ \textbf{\textit{then}}\ \boldsymbol{RREP\_f2}\ \{\boldsymbol{HC} > 1, dst = rcv\_RREQ.dst, dst\_seq \geq rcv\_RREQ.dst\_seq, orig = rcv\_RREQ.orig, ip\_src \neq orig, ip\_src \neq dst, ip\_dst = RT\boldsymbol{E_{orig}}.\boldsymbol{NH}, \boldsymbol{TTL} < NET\_DIAMETER\}$$

Conditions between [ ] are optional expression 5 RREP forward

5.1.1.3 Interaction 3: RERR sending. A node sends a RERR in two cases:

(1) if it detects a link break for the next hop of an active route in its routing table while transmitting data (and route repair, if attempted, was unsuccessful); or
(2) it gets a data packet destined to a node for which it does not have an active route.

A node forwards a received RERR, if the sender is its next hop toward the unreachable destination. We describe RERR format restrictions and their corresponding set of requirements by expressions (6) and (7):

$$if\ (((link\ break) \wedge (no\ local\ repairir)) \vee ((rcv\ Data) \wedge (\nexists\ RTE_{dst})))\ \boldsymbol{then}\ \boldsymbol{RERR\_f1}\ \{\boldsymbol{dst\_count} \geq \boldsymbol{1}, \boldsymbol{unr\_dst} \notin \boldsymbol{RT} \vee (\boldsymbol{unr\_dst} \in \boldsymbol{RT} \wedge \boldsymbol{unr\_dst}.\boldsymbol{stat} = \boldsymbol{down}), \boldsymbol{TTL} = \boldsymbol{1}\}$$

Expression 6 RERR generation

$$\textbf{\textit{if}}\ ((rcv\ RERR) \wedge ((\exists\ unr\_dst \in RT) \wedge (RERR.ip\_src = RTE_{unr\_dst}.NH)))\ \textbf{\textit{then}}\ \boldsymbol{RERR\_f2}\ \{\boldsymbol{dst\_count} \geq \boldsymbol{1}, \boldsymbol{unr\_dst} \in \boldsymbol{rcv\_RERR}.\boldsymbol{unr\_dsts}, \boldsymbol{TTL} = \boldsymbol{1}\}$$

Expression 7 RERR forward

5.1.1.4 Interaction 4: Hello sending. A node broadcasts a hello message to its neighbors only if it is part of an active route, and it has not sent a broadcast (e.g. a Hello, RREQ) or an appropriate layer 2 message (Perkins *et al.*, 2003) within the last hello interval (expression 8):

$$\textbf{\textit{if}}\ ((Time\_interval \geq Hello\_interval) \wedge ((no\ REEQ\ sent) \wedge (no\ Hello\ sent))\ \textbf{\textit{then}}\ \boldsymbol{Hello}\ \{\boldsymbol{dst} = \boldsymbol{ip\_src}, \boldsymbol{HC} = \boldsymbol{0}, \boldsymbol{lifetime} = \boldsymbol{ALLOWED\_HELLO\_LOSS} * \boldsymbol{HELLO\_INTERVAL}, \boldsymbol{TTL} = \boldsymbol{1}\}$$

Expression 8 hello sending

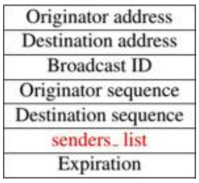
5.1.2 Monitoring and detection process. From the specification description (expressions 1 to 8), we generate for each interaction a set of detection rules that a node must execute each time it receives a routing message.

5.1.2.1 Assumptions. In EIDPF, an intermediate node is prohibited to reply to the RREQ with an RREP, and only the destination node can send RREPs, to avoid malicious intermediate node pretending holding valid routes, and facilitate RREP monitoring. We also assume that node's sequence number value kept by the node itself is always the highest one and must not be updated by received RREQs.

5.1.2.2 Local knowledge. Based on specification description, each node needs to record from ongoing interactions some additional routing information within its history and routing tables, to use them for future interactions monitoring. As shown in Figure 3, the history table has the following additional fields: the senders list, which contains the IP address of nodes from which the RREQ has been received. Destination address, destination sequence number and originator sequence number are needed for comparison when the same RREQ (Orig, BID) is received. Destination sequence is used as local knowledge only if the RREQ has been received from the originator, to avoid taking as reference manipulated values.

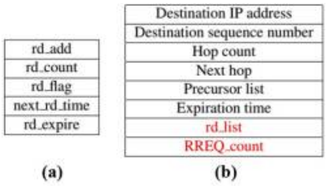
As shown in Figure 4(b), the entry of routing table has two additional fields: requested destinations list "*rd_list*" to maintain a list of destination addresses requested by the route entry destination address and "*RREQ_count*" to accumulate the count of new RREQs received from a neighbor node (more details about RREQ_count in Section 5.2). As shown in Figure 4(a), the requested destination contains the IP address of the requested destination in *rd_add* field. The field *rd_count* records the number of RREQ originated by the route entry destination address toward the IP address in *rd_add* field. When the *rd_Flag* is set, the *rd_count* represents the number of received RREQ with TTL equal to network diameter; otherwise, it represents the number of RREQ with TTL lesser than Network diameter. The expected arriving time of the next RREQ is saved in *next_rd_time*. Expiration time of requested destination is recorded in *rd_expire*, and it is calculated by taking the maximum between network traversal time and *next_rd_time*.

5.1.2.3 Monitoring and detection rules. Whenever a node receives a RREQ, it checks whether the hop count field is coherent with the TTL value in the IP header based on RREQ-forwarding requirement (*Rule 1*) and then verifies whether the received format satisfied the format restrictions (*Rule 2*). If the sender is the originator node, receiving node checks if the number of RREQ for the requested destination has not reach the



Originator address
Destination address
Broadcast ID
Originator sequence
Destination sequence
senders_ list
Expiration

Figure 3.
Fields of history table



rd_add
rd_count
rd_flag
next_rd_time
rd_expire

(a)

Destination IP address
Destination sequence number
Hop count
Next hop
Precursor list
Expiration time
rd_list
RREQ_count

(b)

Notes: (a) Requested destination structure; (b) routing table

Figure 4.
Fields of routing table and request destination structure

maximum RREQ retries and verifies whether the sender has respected the timeout between successive RREQs toward that destination (*Rule 3*). For more accurate monitoring, this control is extended even for intermediate node because RREQ forward depends on RREQ generation; thus, both processes must respect RREQ rate limit. *Rule 4* allows receiving node to detect manipulated values of broadcast ID, sequence number and destination sequence number. If the RREQ has been already received, based on information within its local knowledge and *Rule 5* receiving node can detect modified values of destination address, originator and destination sequence number fields, and also detect replayed RREQ.

The same as in RREQ monitoring, receiving node first checks RREP hop count field using (*Rule 6*), then RREP format restrictions using (*Rule 7*). As intermediate node cannot receive a RREP without prior reception of its corresponding RREQ, receiving node checks whether it exists a valid reverse route within its routing table, and whether the destination address is present within requested destination list of the reverse route (*Rule 8*). If receiving node is the originator, the destination address must be the same as destination address in the history table, otherwise the sender has forged the RREP (*Rule 8*). Also, a node cannot forward a RREP without prior broadcast of its corresponding RREQ, unless it is the destination. Therefore, receiving node must check whether the sender exists in the senders_list of the corresponding RREQ within its history table, otherwise the sender has forged the RREP (*Rule 8*). Even if the sender has actually broadcasted the RREQ, but the RREQ has not been received for an unknown reason (e.g. unreliable link), the EIDPF guarantees in this way the selection of reliable links and avoid unidirectional links.

As there are several mutable fields in the RERR message and their values depend on the sender's local knowledge, there are only two format restrictions to be checked in the monitoring process (*Rule 9*). Receiving node checks whether at least one of the unreachable destinations exists in its routing table, and whether the sender IP address is the next hop toward that destination (*Rule 10*). However, it is not possible for receiving node to check whether a link break has really happened, or if actually the sender does not have an active route to a particular destination, or even to check whether the sender has really received an RERR.

Receiving node checks Hello format restrictions based on (*Rule 11*), then checks whether no RREQ (broadcast) has been sent by the sender within the last HELLO_INTERVAL, and whether the sender has respected the minimum required time interval between successive hello messages (*Rule 12*).

Rule 1: $\{((HC + TTL) - TTL\_start) \% TTL\_increment \neq 0\} \vee \{(((HC + TTL) - TTL\_start) / TTL\_increment < 0))\} \vee \{(\exists\, rd \in RTEorig \cdot rd\_list) \wedge (((HC + TTL) - TTL\_start) / (TTL\_increment)) + 1 = rd.rd\_count)\} \Rightarrow$ "*HOP COUNT VIOLATION*"

Rule 2: $\{(ip\_src = orig)\,\&\,(format \neq RREQ\_f1)\} \vee \{(ip\_src \neq orig)\,\&\,(format \neq RREQ\_f2)\} \Rightarrow$ "*RREQ FORMAT VIOLATION*"

Rule 3: $\{if\,(dst \in RTE_{orig}.rd\_list)\} \wedge \left\{\begin{matrix}(rd.rd\_count > RREQ\_retries) \vee \\ (timeout < rd.next\_rd\_time)\end{matrix}\right\} \Rightarrow$ "*TIMING & RATE LIMIT VIOLATION*"

Rule 4: $\{(my\_ip = orig) \wedge ((BID < RREQ.BID) \vee (SN < RREQ.orig\_seq))\} \vee \{(my\_ip = dst) \wedge (SN < RREQ.dst\_seq)\} \Rightarrow$ "*BID & SEQUENCE NUMBER VIOLATION*"

Rule 5: $\{(orig, BID) \in HT\} \wedge \{(RREQ.dst \neq HTEorig.dst) \vee (dst\_seq \neq HTEorig.dst\_seq) \vee (orig\_seq \neq HTEorig.orig\_seq) \vee (ip\_src \in HTEorig.senders\_list)\} \Rightarrow$ "*RREQ FORWARD VIOLATION*"

Rule 6: $\{(HC + TTL) \neq Net\_diameter\} \Rightarrow$ "*HOP COUNT VIOLATION*"

Rule 7: $\{(ip\_src = dst) \& (format \neq RREP\_f1)\} \vee \{(ip\_src \neq dst) \& (format \neq RREP\_f3)\} \Rightarrow$ "*RREP FORMAT VIOLATION*"

Rule 8: $\{(My\_ip = orig) \wedge (HTEorig.dst \neq dst)\} \vee \{(My\_ip \neq orig) \wedge ((RREQ.orig \notin RT) \vee (dst \notin RTEorig.rd\_list))\} \vee \{(\mathrm{ip\_src} \neq \mathrm{dst}) \wedge (\mathrm{ip\_src} \notin \mathrm{HTE(orig, dst).senders\_list})\} \Rightarrow$ "*RREP FABRICATION*"

Rule 9: $\{(dstCount < 1) \vee (TTL \neq 1)\} \Rightarrow$ "*RERR FORMAT VIOLATION*"

Rule 10: $\{(unr\_dst \in RT) \wedge (ip\_src \neq RTE_{unr\_dst}.NH)\} \vee \Rightarrow$ "*RERR FORGE*"

Rule 11: $(TTL \neq 1) \vee (HC \neq 0) \vee (dst \neq ip\_src) \vee (lifetime \neq ALLOWED\_HELLO\_LOSS \times HELLO\_INTERVAL)\} \Rightarrow$ "*HELLO FORMAT VIOLATION*"

Rule 12: $\{(Time\_interval < Hello\_interval) \vee (rcv\ REEQ\ from\ ip\_src)\} \Rightarrow$ "*TIMINGVIOLATION*"

5.2 Load balancer

To identify fast-forwarding nodes, each node must keep a global counter "RREQ_total" to accumulate the count of received RREQ (duplicate RREQ are not counted). And a counter for each neighbor "RREQ_count" to accumulate the count of new RREQs first received from a neighbor node. We add one additional field "RREQ_count" to the entry of routing table as shown in Figure 4(b). The two counters are used to calculate neighboring node's forwarding rate FR. The forwarding rate FR is defined as (RREQ_count)/(RREQ_total + 1), which represents the probability of a node who forwards a RREQ before others to be finally part of the discovered route. A higher forwarding rate means a higher possibility that a node is a rushing or wormhole node. We modify the route discovery process to consider load balance in route selection.

5.2.1 Procedure for receiving a route request. The procedure for a node receiving an RREQ processing with load balance re as follows:

```
Algorithm 1: RREQ processing
  begin
    if (orig, BID) is new then
      RREQ_total ++
      $RTE_{ip\_src}.RREQ\_count++$
      if $FR(ip\_src) < Threshold$ then
        Add RREQ to HT
        Create or update $RTE_{orig}$
        if $my\_ip \neq RREQ.dst$ then
          RREQ.HC ++
          Broadcast RREQ
        else
          Send RREP
      end
    else
        Remove RREQ from HT // to treat the next RREQ
        Drop RREQ
      end
    else
      Drop RREQ
    end
  end
```

Receiving node first increments its RREQ_total value and the RREQ_count of the sender. Then it calculates the forwarding rate FR value of the sender. If the FR value of sender is lesser than the threshold, it adds the RREQ to the history table and creates or updates the reverse route toward the originator. The receiving node sends an RREP if it is the destination, otherwise it forwards the RREQ. In the case, FR value is equal or greater than the threshold. The receiving node would drop the RREQ and would not install a reverse route until it receives the RREQ from a sender whose FR value does not exceeds the threshold. We can take advantage of the attacker's fast-forwarding behavior to reach destination faster. In this case, the receiving node forwards the RREQ even if the sender's FR value exceeds threshold, but it does not install a reverse route until it receives the same RREQ from a neighbor node with a FR value lesser than the threshold.

5.2.2 Wormhole and rushing prevention. As the fast-forwarding value of each node is zero at the initial phase, route attraction property cannot be noticed, and thus fast-forwarding nodes are unavoidable for a period. However, when a rushing or a wormhole node is constantly selected in the route discoveries, its neighbors would accumulate its FR values. When the FR values reach the threshold, the RREQ messages received from the malicious node will be discarded.

5.2.3 Recovery from isolation. If one node, such as X, its forwarding rate value reaches the threshold value of one of its neighbor node, say Y, on the routes. Therefore, node Y would reject RREQs from node X. After a certain number of route discoveries, the forwarding rate value of X saved in node Y would gradually become smaller because the increase of the RREQ_total value. As the forwarding rate value is defined as (RREQ_count)/(RREQ_total + 1), the increase of denominator makes the whole forwarding rate value become smaller. Besides the proposed route discovery assure load balance, it avoids network disconnection resulted by a long-term isolation of a normal node, which is likely placed in a key position of connectivity within the network, and attracts too many routes. Furthermore, the topology is dynamic, and therefore, the current key node would not stay on the same position. Extended simulation results by ns-2 (The network simulator, 2016) are presented in Section 6.

5.3 Response mechanism

To avoid isolation of honest nodes that may forward packet of malicious nodes due to insufficient local knowledge caused by high mobility or a new coming node. Violations committed by malicious originator nodes are punished for a long period (long isolation), while those committed by malicious intermediate node are punished for short period (short isolation). We use a binary exponential backoff to consider repeated intrusions so as to isolate recidivist node for a longer period. The first time a node commits a specification violation it will be isolated for a short_isolation or long_isolation interval depending on if it is originator or intermediate node. The second time the isolation period is 2* previous isolation period, for each additional detection, the isolation period is multiplied by 2. During the isolation period, routing packets received from malicious intermediate node will be directly discarded, and thus no further punishment can be assigned. However, packet received from malicious originator node are treated, and if a violation is detected, the isolation period will be extended. For our experimentation (Section 6), we set a short_isolation equal to NET_TRAVERSAL_TIME, and long_isolation equal to 3 * NET_TRAVERSAL_TIME.

5.3.1 Recovery from isolation. Using different isolation periods avoids network disconnection resulted by a long-term isolation of honest node, which has forwarded packets of malicious node because of the insufficient local knowledge. During isolation, the honest node continues receiving routing packets from malicious node and thus enrich its local knowledge and may detect further specification violations, and avoid forwarding further packets from malicious nodes.

6. Evaluation and simulation results

6.1 AODV basic attacks implementation

Ning and Sun (2005) presented a systematic analysis of attacks against AODV, and they used the definition of atomic misuses, which is similar to our definition of specification violation attacks. They identified the following set of attack goals: [route disruption (RD); RI; NI; resources consumption (RC)]. They showed how the following set of atomic misuses – drop (DR); modify and forward (MF); forge reply (FR); active forge (AF) – are conducted to achieve the previous goals. However, they limited the definition of compound misuses which is similar to our definition of compound attacks to the repetition of the same type of atomic misuse and do not consider combination of different misuses. In this paper, we follow the same naming scheme used in (Ning and Sun, 2005) which combines routing message type, atomic misuse action and attack goal, in the form of *MessageType_Action_Goal*. For instance, RREQ_MF_RI represents a malicious node attempting to invade a route (RI) by modifying (MF) an RREQ message. The reader is referred to Ning and Sun (2005) for further details about classification of such misuses as well as detailed scenarios of such attacks, and to http://discovery.csc.ncsu.edu/software/MisuseAODV/1 to download the implementation of these attacks. We test and evaluate our framework in two steps. We first test EIDPF on basic attacks (misuses) described in Ning and Sun (2005) based on their goals. Then, we consider compound attacks.

6.2 Simulation setup

We conducted experiments using the ns-2 simulator (The network simulator, 2016), version 2.35 set up on a core i3 Ubuntu 13.10 with 4 GB memory. In all our experiments, we used continuous bit rate, we considered for each simulation scenario 50 mobile nodes. The field configuration is 1,000 × 1,000 m. Every simulation runs for 300 s during which an originating node sends five data packets per simulated seconds. After arriving at a location, a node stays there for 2.0 s before moving to the next location. There are at most 50 connections during each simulation run. The nodes mobility rate is 0, 5, 10, 15, and 20 m/s; Table II summarizes the used parameters. For each mobility rate, ten simulation runs were conducted to get an average value. To measure and validate the effectiveness of our proposal, we implemented a new extension of AODV integrating the proposed framework using ns-2 (The network simulator, 2016), with the Rice Monarch extension for the AODV protocol. To evaluate our framework and quantify its performance, we used the following metrics:

- percentage of data packets transmitted through the malicious node;
- routing overhead (i.e. the total number of routing packets);
- the detection rate (i.e. ratio between the number of correct detected intrusions and the total number of intrusions); and
- the packet delivery ratio (PDR) (i.e. the percentage of transmitted packets that reach their destination).

6.3 Results analysis

6.3.1 Route invasion (using specification violation attacks). To invade route malicious node within victim node's transmission range can increase originator RREQ (RREQ_MF_RI). It can also fabricate RREQ, as though it has been originated and received from the victim node (RREQ_AF_RI). After receiving a RREQ, malicious node may forge a RREP with increased hop count and destination sequence number to suppress the legitimate RREP message (RREP_FR_RI). It can also forge an RREP without receiving RREQ (active forge) to invade an existing route (RREP_AF_RI), more details about scenarios in Ning and Sun (2005). EIDPF detects these attacks owing to detection *Rules 1*, *4*, *7* and *8* (Table III) and

Simulation parameters	Value
Number of nodes	50
Simulation area	1,000 × 1,000 m
Mobility model	Random waypoint Model (RWP)
Simulation time	300 s
Simulation traffic	CBR (constant bit rate)
Traffic volume	5 packets per second
MAC protocol	IEEE 802.11
Node's mobility	Ranged from 0 to 20 m/s

Table II. Simulation parameters

Goals	Specification violations	Detection rules
Route invasion	RREQ_MF_RI	BID and SN violation (*Rule 4*)
	RREQ_AF_RI	BID and SN violation (*Rule 4*)
	RREP_FR_RI	Hop count violation (*Rule 6*)
		RREP format violation (*Rule 7*)
		RREP fabrication (*Rule 8*)
	RREP_AF_RI	RREP fabrication (*Rule 8*)
	RREQ replay	RREQ forward violation (*Rule 5*)
	RREP replay	RREP fabrication (*Rule 8*)
Resources consumption	RREQ_MF_RC	Timing and rate limit violation (*Rule 3*)
	RREQ_AF_RC	Timing and rate limit violation (*Rule 3*)
	RREP_AF_RC	RREP fabrication (*Rule 8*)
	RREQ replay	RREQ forward violation (*Rule 5*)
Node isolation	RREQ_MF_NI	BID and SN violation (*Rule 4*)
	RREQ_AF_NI	Hop count violation (*Rule 1*)
	RREP_FR_NI	RREP format violation (*Rule7*)
	RREP_AF_NI	RREP fabrication (*Rule 8*)
Route disruption	RREQ_MF_RD	BID and SN violation (*Rule 4*)
	RREQ_AF_RD	Hop count violation (*Rule 1*)
		Timing and rate limit violation (*Rule 3*)
	RREP_MF_RD	RREP fabrication (*Rule 8*)
		RREP format violation (*Rule 7*)
	RREP_FR_RD	Hop count violation (*Rule 6*)
		RREP format violation (*Rule 7*)
	RREP_AF_RD	RREP fabrication (*Rule 8*)
	RERR_MF_RD	RERR forge (*Rule 10*)

Table III. Specification violation detection rules

prevent malicious node from invading routes. Figure 5 shows a small number of data packets transmitted through the malicious node which is similar to the normal case (no attacks).

6.3.2 Resources consumption (or sleep deprivation). Whenever malicious node receives an RREQ, it increases its BID to make it appear fresher and then broadcasts it (RREQ_MF_RC) to consume nodes' energy in rebroadcasting process. In the case of RREQ_AF_RC, malicious node forges RREQ message, increase the hop count and proceeds in the same way as described above. Another way for malicious node to consume nodes' resources is to create a routing loop between them by sending forged RREP (RREP_AF_RC) just as described in Ning and Sun (2005). EIDPF detects RREQ_MF_RC and RREQ_AF_RC as timing and rate limit violation owing to *Rule 3*. As shown in Figure 6(a), EIDPF maintains routing overhead almost similar as in normal situation, which is not the case against RREQ_AF_RC because EIDPF detects the attack after the second forged RREQ. Despite that, EIDPF minimizes the impact of the attack and reduces the network routing overhead by ten times as shown in Figure 6(b). RREP_AF_RC is detected as RREP fabrication owing to *Rule 8*.

6.3.3 Node isolation. Malicious node may isolate a node from receiving data packets from other nodes for a short period, if it is the only neighbor of the victim node. It

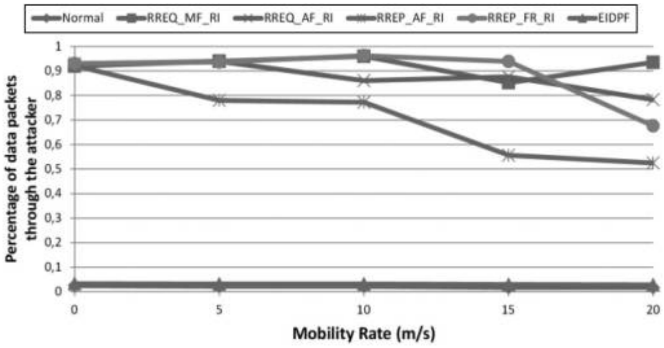


Figure 5.
Route invasion detection

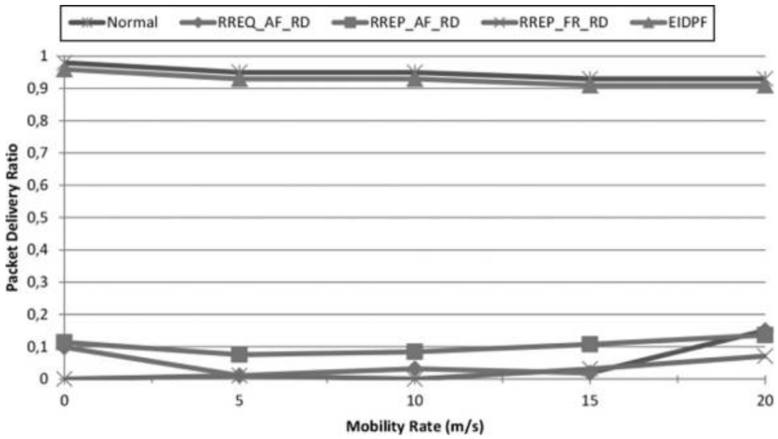


Figure 6.
Resources consumption detection

achieves this attack by replacing destination address with a non-existent address, and increasing BID and originator sequence number in received RREQs (RREQ_MF_NI), or in forged RREQ (RREQ_AF_NI). EIDPF detects RREQ_MF_NI as BID and SN violation (*Rule 4*) and detects RREQ_AF_NI as hop count violation owing to *Rule 1*. Malicious node can also partially isolate victim node by replying with forged RREP (with increased destination sequence number) each time it receives a RREQ originated by victim node (RREP_FR_NI). EIDPF detects RREP_FR_NI as RREP format violation using *Rule 7*. Another way to isolate victim node is to become its next hop toward all other network nodes. In this case, malicious node sends to victim node multiple forged RREP with different destination addresses and increased destination sequence numbers (RREP_AF_NI). EIDPF detects RREP_AF_NI as RREP fabrication (*Rule 8*). As shown in Figure 7, EIDPF presents a packet delivery rate similar to PDR in normal situation where no attacks occur.

6.3.4 Route disruption. Malicious node may prevent routes from being established by manipulating routing information carried by received RREQ or RREP. It can for instance decrease originator sequence number and BID of the received RREQ (RREQ_MF_RD) to be discarded, or decrease destination sequence number, or setting lifetime field to zero in the received RREP (RREP_MF_RD) to invalidate possible updates. Malicious node can also break down an existing route by forging RREQ (RREQ_AF_RD) or RREP (RREP_AF_RD) with erroneous routing information; more details about scenarios can be found in Ning and Sun (2005). EIDPF detects disruption attacks owing to detection rules mentioned in Table III, as shown in Figure 8, the PDR is similar to PDR in normal situation. Table III summaries how EIDPF detects specification violation basic attacks, using detection rules.

6.3.5 Wormhole and rushing. Wormhole was implemented by creating a tunnel between two colluding nodes. The first colluding node encapsulates the received RREQ into WRREQ, which is a format recognized only by wormhole nodes, and sends it to the second colluding node without incrementing the hop count. In the case of rushing attack, the rushing node ignores backoff and interframe spacing time imposed by the 802.11 standard to propagate its RREQ faster and to get a time advantage over normal RREQs. As shown in Figure 9, we have used EIDPF with different threshold values to investigate if the forwarding rate threshold would affect the avoidance of attack. The three thresholds values signify a neighbor node can be consecutively selected in routes no more than one (0.51), two (0.67) or three (0.76) times, and then, its neighbor node would temporarily reject it. As shown in Figure 9, EIDPF reduces the percentage of data packets transmitted through malicious node by approximately 60 per cent when threshold value is set to 0.51, and 50 per cent when threshold value is set to 0.67. To select the optimal threshold, we also compute the packet delivery rate for each threshold, based on results shown in Figure 10, threshold value 0.67 provides the best trade-off between the two metrics, by offering low RI rate and high PDR.

6.3.6 Black and gray hole. In this attack, malicious node first invades routes using basic attacks (RREQ_MF_RI, RREQ_AF_RI, RREP_FR_RI and RREP_AF_RI) and then drops all (or selectively in the case of gray hole) received data packets. EIDPF detects RI using specification detection rules as described previously (Table III) and prevents the attack on the RI stage, and therefore avoid the packet dropping. As shown in Figure 11, EIDPF assure a PDR similar to the normal case.

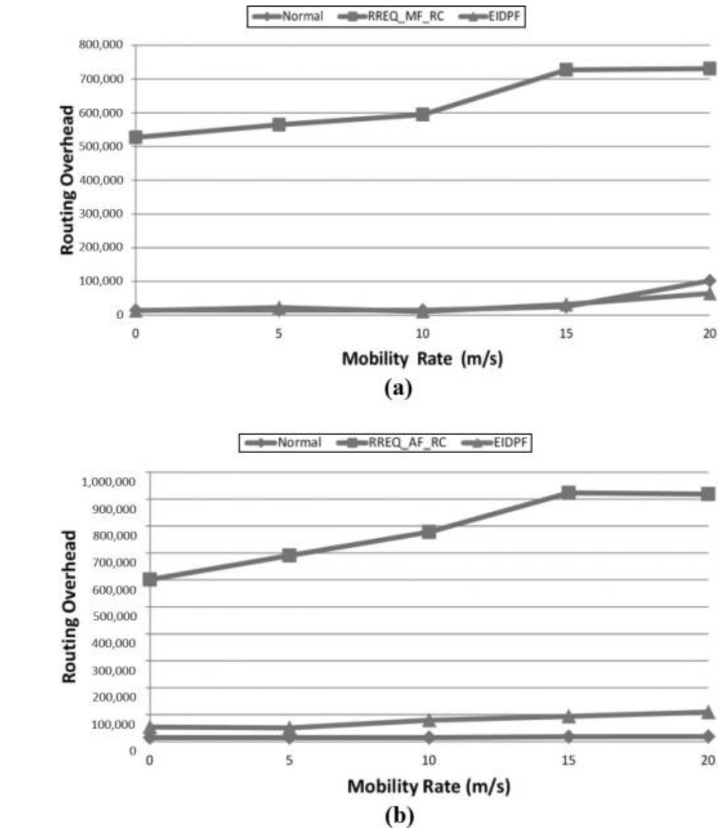


Figure 7.
Node isolation detection

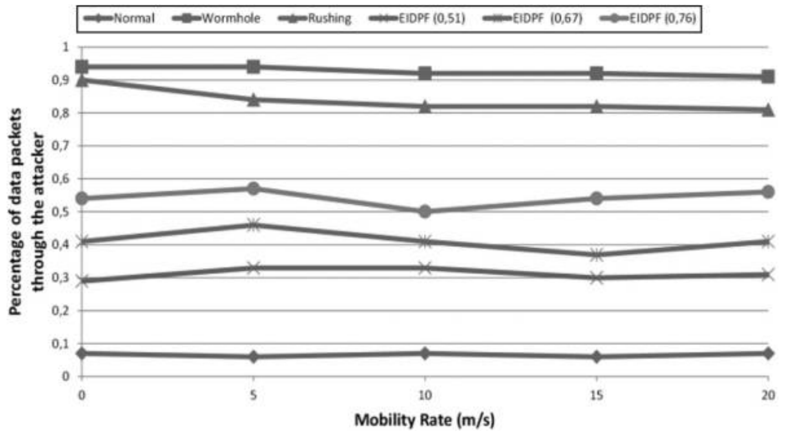


Figure 8.
Route disruption detection

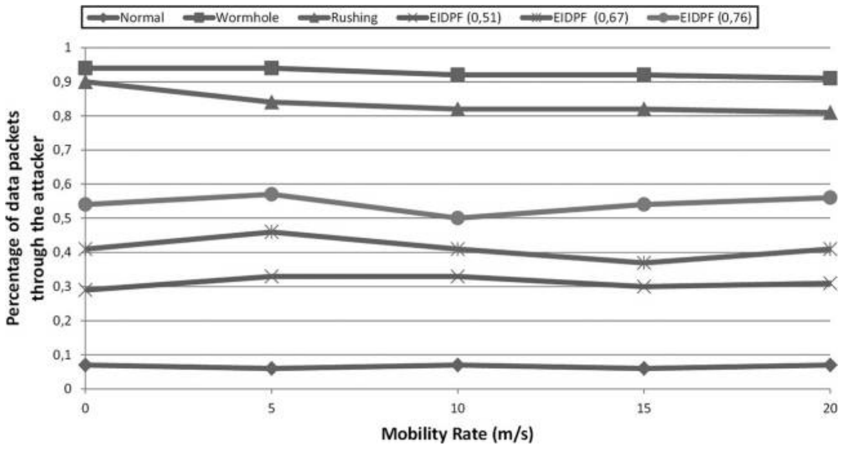


Figure 9. Wormhole and rushing detection

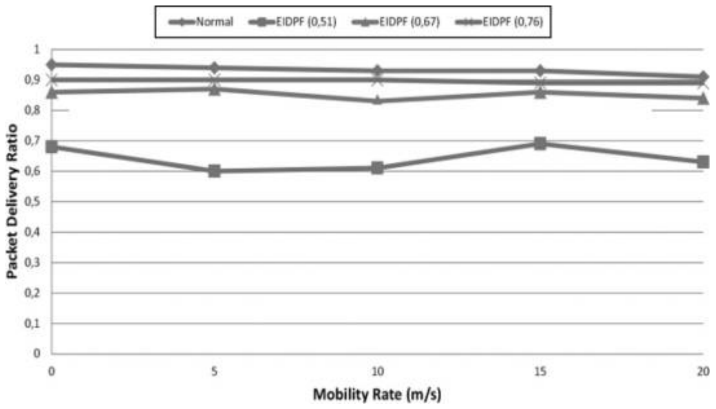


Figure 10. EIDPF PDR under different thresholds

6.3.7 Distributed denial of service (multiple attackers). We evaluate EIDPF against multiple attackers executing basic and compound attacks described earlier, and under mobility rate between 0 and 20 m/s. We do not consider RI attacks because of their own they do not target the availability of routing services (DoS definition in Section 3). As shown in Figure 12, the PDR is similar to the normal one, except in the case of RC attacks, where PDR drops slightly and gradually when the number of malicious nodes increases. We explain this packet loss by the direct correlation between numbers of malicious nodes and generated overhead. The more the overhead is important, the more queues are full and network is congested, leading to dropping data packets.

6.3.8 False positive discussion. We observe some cases of false positives, i.e. nodes that are detected as malicious. False positives rate is defined as the ratio between the number of legitimate nodes detected as malicious and the total number of legitimate nodes. As shown in Figure 13, the simulations results showed a negligible (1 per cent) false positive rate in the case of sleep deprivation attack (RC), the other attacks shows a null false positive rate. The false positive cases concern basic attacks violating timing and rate limit rule in high mobility rate, where nodes quit and join

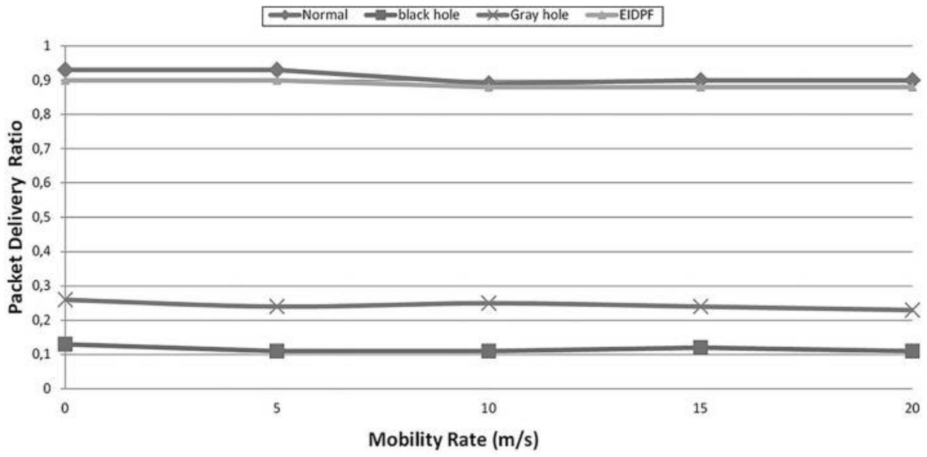


Figure 11.
Black and gray hole attacks

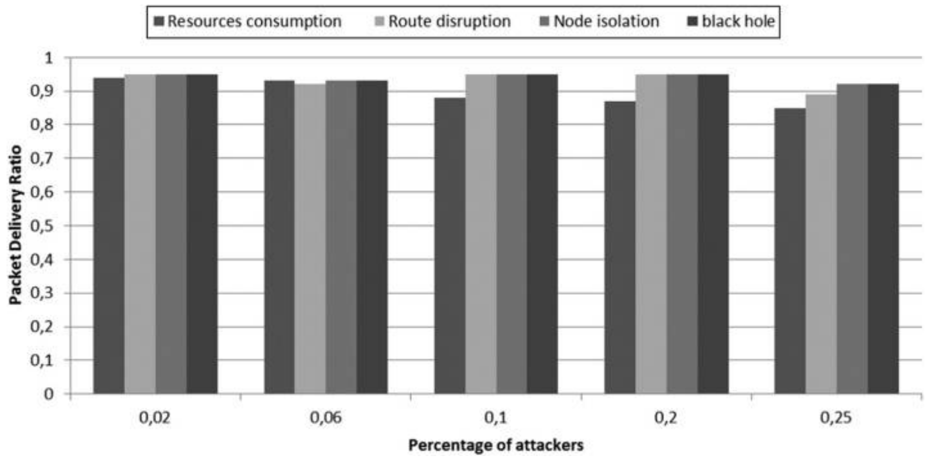


Figure 12.
EIDPF PDR against DDOS attack

a neighborhood frequently. A node that joined a neighborhood recently can forward a packet sent by malicious node, and thus it would be detected as malicious because it does not have enough knowledge about the sender and the previous interactions.

6.4 Comparison between efficient intrusion detection and prevention framework and other researches

As our proposition is an intrusion detection framework, we compare EIDPF only to research belonging to that category. Table IV provides comparison between EIDPF and other related works. EIDPF presents several improvements in comparison with existing specification-based IDS for AODV. Particularly, it is able to detect all possible attacks against the routing protocol because:

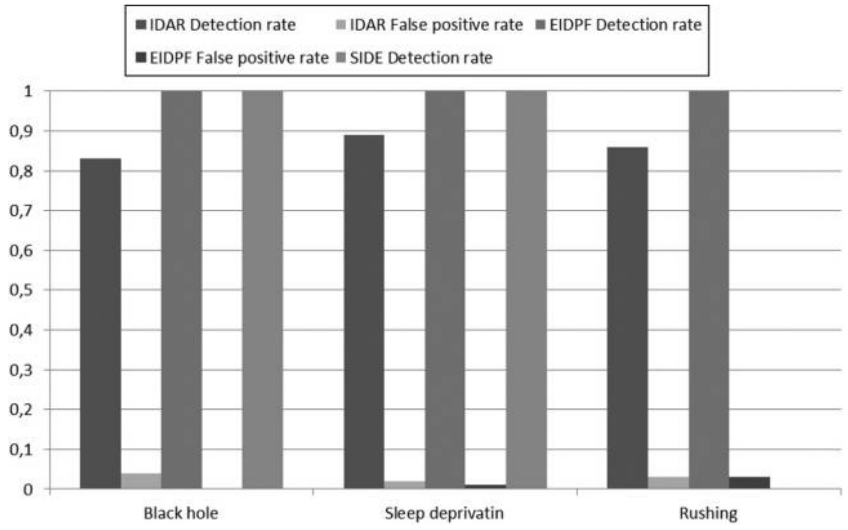


Figure 13. Comparison between EIDPF, SIDE and IDAR

- it is not based on partial protocol specification such in Tseng *et al.* (2003), Hassan *et al.* (2006), Grönkvist *et al.* (2007) and Panos *et al.* (2010) but considers the complete operation of AODV; and
- it monitors the reception of all routing messages.

Specification-based IDS proposed in Panos *et al.* (2010, 2014) and Huang and Lee (2004) can only monitor hosting nodes, either by using extra hardware support such as shared memory block like in Huang and Lee (2004) or protected zone (TrustZone SoC) such in Panos *et al.* (2014), or by using a number of walking agents such in Panos *et al.* (2010) which provide an intermittent protection of visited nodes.

Furthermore, EIDPF does not use promiscuous monitoring like in Tseng *et al.* (2003) which is error prone and resources consuming. Protection mechanisms proposed in Alattar *et al.* (2012) and Shakshuki *et al.* (2013) can only detect few particular attacks, whereas EIDPF is able to detect a number of specification violation attacks such as Tseng *et al.* (2003), Panos *et al.* (2010), Panos *et al.* (2014), and advanced attacks like Nadeem and Howarth (2014) and Barani and Abadi (2012). EIDPF neither generates extra overhead such as Tseng *et al.* (2003), Panos *et al.* (2010), Barani and Abadi (2012), Shakshuki *et al.* (2013) and Nadeem and Howarth (2014) nor high rate of false positives like in Jabbehdari *et al.* (2012), Barani and Abadi (2012) and Nadeem and Howarth (2014). Unlike the majority of propositions in the literature which do not consider intrusion response such as Alattar *et al.* (2012), Jabbehdari *et al.* (2012) and Panos *et al.* (2014) or just provide a passive response by raising alarms such in Tseng *et al.* (2003), Barani and Abadi (2012) and Shakshuki *et al.* (2013), EIDPF as well as Panos *et al.* (2010) and Nadeem and Howarth (2014) provides active and adaptive response by isolating malicious node.

To quantify the comparison, we select among the above works SIDE (Panos *et al.*, 2014) and IDAR (Nadeem and Howarth, 2014) because the other propositions either had no experimental results provided such as Tseng *et al.* (2003) and Panos *et al.* (2010), or focus on a particular type of attacks such as Shakshuki *et al.* (2013), or use different

Table IV.
Comparison between EIDPF and related works

IDS	Architecture	Detection technique	Considered attacks	Routing protocol	Response mechanism	Contribution	Limits
Tseng *et al.* (2003)	Selective distributed and cooperative	Specification-based	Drop, modification, fabrication	AODV	Alarms	First specification-based IDS	Overhead, resources consuming
Panos *et al.* (2010)	Mobile agents	Specification-based	Routing table poisoning, black hole, DoS	AODV	Alarms, remove malicious node	multi-layer, specification-based detection	Interrupted protection, no analytic, or simulation validation
Panos *et al.* (2014)	Stand-alone	Specification-based	Modification, fabrication, sleep deprivation, black and grey hole	AODV	Not considered	Complete formal specification, Real-time detection	Extra hardware support, resources consuming
Shakshuki *et al.* (2013)	Stand-alone	Acknowledgment-based	Drop, false accusation, limited transmission power	DSR	Alarms	Resolve the weakness of Watchdog	Extra overhead, resources consuming
Nadeem *et al.* (2014)	Hierarchical, clustered	Hybrid (anomaly and signature)	Sleep deprivation, black and grey hole, rushing	AODV	Isolate attacker nodes	Generalized IDS with high detection rate and attacker isolation	Overhead, resources consuming, false accusation
Alattar *et al.* (2012)	Distributed and cooperative	Signature-based	Drop, Modification, fabrication	OLSR	Not considered	Adaptive cooperative investigation	Vulnerable to false accusation, resources consuming
Jabbehdari *et al.* (2012)	Not specified	Anomaly based	DoS attacks	DSR	Not considered	Uses neural network for MANET ABID, identify attacker	Manual definition of data training classes, continuous updates
Barani *et al.* (2012)	Stand-alone	Anomaly based	Sleep deprivation, black hole, rushing, wormhole	AODV	Alarms	Rapid adaptation to topology changes	Significant overhead
EIDPF	Stand-alone	Specification-based	All attacks except dropping	AODV	Isolate attacker nodes	Real time detection, load balancing, high detection rate and attacker isolation	Manual extraction of specification model

simulation parameters and evaluation metrics such as Barani and Abadi (2012), Jabbehdari *et al.* (2012) and Nadeem and Howarth (2013a). In addition, both SIDE and IDAR are based on AODV routing protocol. To make a fair comparison, we consider only attacks detected by the three mechanisms (sleep deprivation, black hole and rushing) and take the same simulation parameters (number of nodes: 50; mobility rates [0..20], number of malicious nodes between 0 and 10). Panos *et al.* (2014) claim that SIDE can detect wormhole and rushing attacks. However, in contrary to what was stated by the authors, wormhole tunnel cannot be created just by modifying the hop count field, and rushing attack cannot be achieved by omitting the Backoff mechanism used by originator node (rushing is performed by intermediate node).

The experimental data in Figure 13 show that EIDPF outperforms IDAR in terms of detection and false positive rates, except for rushing attack where it displays almost the same false positive rate as IDAR. Both EIDPF and SIDE perform detection of specification violation attack in real time with the same detection rate; however, EIDPF does not require special hardware to operate, such as trusted computing platform (TrustZone SOC), while SIDE does require special hardware. Unlike SIDE, our framework EIDPF does not induce additional computational costs and memory consumption (due to remote attestation procedures). EIDPF does not induce extra control packet overhead due to periodic packet gathering and accusation packet sending like IDAR, this can be attributed to the fact that EIDPF performs monitoring relying exclusively on local information. In addition, EIDPF does not rely on a single node (manager node) like IDAR, which constitutes a single point of failure. Both IDAR and EIDPF provide an adaptive response against malicious node. IDAR isolates malicious node completely or get around it based on the attack damage. EIDPF isolates the malicious node completely but for different durations based on the attack recurrency.

7. Conclusions

This paper proposed an intrusion detection and prevention framework called EIDPF, which is based on AODV, to defend against routing attacks in MANETs. EIDPF architecture includes three complementary modules:

(1) a specification-based IDS to detect attacks violating the protocol specification;

(2) a load balancer to prevent fast-forwarding attacks; and

(3) adaptive response mechanism to isolate malicious node from the network.

A key strength of EIDPF is its ability to guarantee a real-time detection of known and unknown attacks, and its capacity to avoid wormhole and rushing attacks by providing a load balancing route discovery. Simulation results showed that our proposed mechanism presents high detection rate and low false positives ratio, under different mobility rates, and against multiple attackers (DDOS). Moreover, EIDPF does not induce extra communication overhead compared to other protection mechanisms.

References

Alattar, M., Sailhan, F. and Bourgeois, J. (2012), "Log-based intrusion detection for MANET", *Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, *Limassol*, pp. 697-702.

Barani, F. and Abadi, M.I. (2012), "BeeID: intrusion detection inAODV-based MANETs using artificial bee colony and negative selection algorithms", *The ISC International Journal of Information Security*, Vol. 4 No. 1, pp. 25-39.

Cayirci, E. and Rong, C. (2008), *Security in Wireless Ad Hoc and Sensor Networks*, John Wiley & Sons (Ed.), The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ.

Clausen, T. and Jacquet, P. (2003), *Optimized Link stat E Routing Protocol (OLSR*), IETF RFC, p. 3626.

Ebinger, P. and Bucher, T. (2006), "Modelling and analysis of attacks on the MANET routing in AODV", in Springer-Verlag (Ed.), *5th international conference on Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW'06*, Springer-Verlag, Ottawa, Heidelberg, pp. 294-307.

Gonzalez-Duque, O.F., Ansa, G., Howarth, M.P. and Pavlou, G. (2008), "Detection and accusation of packet forwarding misbehaviour in mobile ad hoc networks", *Journal of Internet Engineering*, Vol. 2 No. 8, pp. 181-192.

Grönkvist, J., Hansson, A. and Sköld, M. (2007), "Evaluation of a specification-based intrusion detection system for AODV", *The Sixth Annual Mediterranean Ad Hoc Networking WorkShop*, Corfu, pp. 121-128.

Hassan, H.M., Mahmoud, M. and El-Kassas, S. (2006), "Securing the AODV protocol using specification-based intrusion detection", in ACM (Ed.), *The 2nd ACM International Workshop on Quality of Service & Security for Wireless and Mobile Networks*, ACM, Torremolinos, NY, pp. 33-36.

Huang, Y. and Lee, W. (2004), "Attack analysis and detection for ad hoc routing protocols", in Erland, J., Valdes, A. and Magnus, A. (Eds), *Recent Advances in Intrusion Detection*, Springer-Verlag, Berlin, Heidelberg, pp. 125-145.

Jabbehdari, S., Talari, S.H. and Modiri, N. (2012), "A neural network scheme for anomaly based intrusion detection systems in mobile ad hoc networks", *Journal of Computing*, Vol. 4 No. 2, pp. 61-66.

Medadian, M., Yektaie, M.H. and Rehmani, A.M. (2009), "Combat with black hole attack in AODV routing protocol in MANETs", in IEEE (Ed.), *First Asian Himalayas International Conference on Internet, Internet, AH-ICI 2009, Kathmandu*, pp. 1-5.

Mitrokotsa, A. and Dimitrakakis, C. (2013), "Intrusion detection in MANET using classification algorithms: the effects of cost and model selection", *Ad Hoc Networks*, Vol. 11 No. 1, pp. 226-237.

Mulert, J.V., Welch, I. and Seah, W.K.G. (2012), "Security threats and solutions in MANETs: a case study using AODV and SAODV", *Journal of Network and Computer Applications*, Vol. 35 No. 4, pp. 1249-1259.

Nadeem, A. and Howarth, M.P. (2013a), "Protection of MANETs from a range of attacks using an intrusion detection and prevention system", *Telecommunication Systems*, Vol. 52 No. 4, pp. 2047-2058.

Nadeem, A. and Howarth, M.P. (2013b), "A survey of MANET intrusion detection & prevention approaches for network layer attacks", *IEEE Communications Surveys & Tutorials*, Vol. 15 No. 4, pp. 2027-2045.

Nadeem, A. and Howarth, M.P. (2014), "An intrusion detection & adaptive response mechanism for MANETs", *Ad Hoc Networks*, Vol. 13 Part B, No. 0, pp. 368-380.

Ning, P. and Sun, K. (2005), "How to misuse AODV: a case study of insider attacks against mobile ad-hoc routing protocols", *Ad Hoc Networks*, Vol. 6 No. 3, pp. 795-819.

Panos, C., Xenakis, C. and Stavrakakis, I. (2010), "A novel intrusion detection system for MANETs", in SECRYPT (Ed.), *International Conference on Security and Cryptography (SECRYPT)*, IEEE, *Athens*, pp. 1-10.

Panos, C., Xenakis, C., Kotzias, P. and Stavrakakis, I. (2014), "A specification-based intrusion detection engine for infrastructure-less networks", *Computer Communications*, Vol. 54, pp. 67-83.

Perkins, C.E., Royer, E.M. and Das, S. (2003), "Ad hoc On-demand Distance Vector (AODV)", IETF RFC 3561.

Schneier, B. (1999), "Attack trees - modeling security threats, *Dr", Dobb's Journal*, Vol. 24 No. 12, pp. 21-29.

Sen, S. (2010), "Evolutionary computation techniques for intrusion detection in mobile ad hoc networks", Doctoral dissertation, University of York.

Shakshuki, E.M., Kang, N., Sheltami, T.R. (2013), "EAACK – A secure intrusion-detection system for MANETs", *Industrial Electronics, IEEE Transactions*, Vol. 60 No. 3, pp. 1089-1098.

Su, M.Y. (2010), "WARP A wormhole-avoidance routing protocol by anomaly detection in mobile ad hoc networks", *Computers & Security*, Vol. 29 No. 2, pp. 208-224.

Su, M.Y. (2011), "Prevention of selective black hole attacks on mobile ad hoc networks through intrusion detection systems", *Computer Communications*, Vol. 34 No. 1, pp. 107-117.

The network simulator (2016), ns2., available at: www.isi.edu/nsnam/ns/

Tseng, C.Y., Balasubramanyam, P., Ko, C., Limprasittiporn, R., Rowe, J. and Levitt, K. (2003), "A specification based intrusion detection for AODV", in ACM (Ed.), *ACM Workshop on Security of Ad Hoc and Sensor Networks*, ACM, VA, NY, pp. 125-134.

Zhang, X.Y., Sekiya, Y. and Wakahara, Y. (2009), "Proposal of a method to detect black hole attack in MANETs", *Proceeding IEEE International Symposium on Autonomous Decentralized System ISADS*, IEEE, pp. 1-6.

About the authors

Abdelaziz Amara Korba is a PhD candidate in Department of Computer Science at Badji Mokhtar University, Algeria. His research focuses on wireless network, especially on the security issues in *ad hoc* networks, wireless sensor networks and vehicular *ad hoc* networks. He received his master's degree in Computer Science from Badji Mokhtar University. He is currently member of Networks and Systems Laboratory. Abdelaziz Amara Korba is the corresponding author and can be contacted at: amarakorba.abdelaziz@gmail.com

Dr Mehdi Nafaa is a Doctor in computer science. He received his Engineer status in Computer Badji Mokhtar University in 2003, his master's degree in Computer Science, Poitiers France in 2005, and his PhD in Computer Science University Evry FRANCE. He currently teaches in the Department of Computer Science University Badji Mokhtar, Annaba, Algeria, and is a Head of research laboratory in LRS (Laboratory Network and System).

Pr Salim Ghanemi is a Professor in Computer Science. He received his engineer status in Computer from the University of Constantine, Algeria, in 1981, his Master in Computer Science from the University of Birmingham, United Kingdom in 1982, and his PhD in Computer Science from the University of Loughborough, United Kingdom in 1987. He is currently the Head of Embedded Systems Laboratory (LASE).