**MiMl: Mitigating Packet Misrouting in Locally-Monitored Multi-hop Wireless Ad Hoc Networks**

Issa Khalil

College of Information Technology, United Arab Emirates University, UAE; Email: ikhalil@uaeu.ac.ae

Abstract—local monitoring has been demonstrated as a powerful technique for mitigating security attacks in multi-hop ad-hoc networks. In local monitoring, nodes overhear partial neighborhood communication to detect misbehavior such as packet drop or delay. However, local monitoring as presented in the literature is vulnerable to an attack called misrouting attack. Packet misrouting disrupts the packet from reaching the destination by malicious behavior at an intermediate node. However, the malicious node gives the impression to its neighbors that it performed the legitimate forwarding action. Moreover, a legitimate node comes under suspicion. I provide a protocol called MiMi based on local monitoring to remedy the attack. It requires the neighbors to maintain additional information about the routing path. I show through performance analysis that the basic local monitoring fails to mitigate the misrouting attack while MiMi successfully does.

**Keywords:** Packet dropping, multi-hop wireless networks, local monitoring, misrouting, transmission power control.

I. INTRODUCTION

It has been shown in literature that wireless ad-hoc networks are vulnerable to a wide range of security attacks against both control (e.g., routing establishment) and data traffic. Control traffic attacks include wormhole [4], rushing [3], and Sybil [9]. The most notable data traffic attacks are blackhole, selective forwarding, delaying of packets, and packet misrouting in which respectively a malicious node drops data (entirely or selectively) passing through it, delays its forwarding, or relaying the packet to the wrong next-hop. These attacks could result in a significant loss of data or degradation of network functionality, say through disrupting network connectivity.

Cryptographic mechanisms alone cannot prevent these attacks since many of them, such as the wormhole and the rushing attacks, can be launched without needing access to cryptographic keys or violating any cryptographic check. To mitigate such attacks, many researchers have used the concept of cooperative Local Monitoring (basic LM) within a node’s neighborhood (e.g., [1], [4]-[8]). In local monitoring, nodes oversee part of the traffic going in and out of their neighbors. Different types of checks are done locally on the observed traffic to make a determination of malicious behavior. For systems where arriving at a common view is important, the detecting node initiates a distributed protocol to disseminate the alarm. Many protocols have been built on top of local monitoring for intrusion detection (e.g., [2]), building trust and reputation among nodes (e.g., [1]), protecting against control and data traffic attacks (e.g., [4]-[6], [8]) and in building secure routing protocols (e.g., [6], [7]). Specifically, in [4] and [6] the authors have presented a technique for the detection of control and data attacks in sensor networks using local monitoring. These attacks are detected by a group of nodes, called guard nodes that perform local monitoring. The guard nodes are normal nodes in the network and perform their basic functionality in addition to monitoring. Under local monitoring, a guard node verifies for a fraction of the packets if it is being forwarded within the requisite delay bound, without modification and without fabrication. However, basic LM fails to countermeasure the packet misrouting attack. For example, packet misrouting attack would be very damaging for protocols that rely on overhearing for building reputation scores among neighbors [18]. The misrouting cannot be detected while the adversary nodes achieve high reputation scores.

In this paper, I present a mitigation mechanism to countermeasure the misrouting attack using local monitoring. Through the misrouting attack, the attacker achieves the objective of disrupting the packet from reaching the destination by maliciously forwarding it to the wrong next-hop. The malicious node gives the impression to its neighbors capable of overseeing the packet that it has performed the required action (i.e., relaying the packet to the correct next-hop en route to the destination). This attack is applicable to unacknowledged packets. Due to the resource constraints of bandwidth and energy, much traffic in multi-hop ad hoc wireless networks (e.g., sensor networks) is unacknowledged. This is particularly true for the more common data traffic or broadcast control traffic than for rare unicast control traffic.

Consider a scenario in which a node called $S$ is forwarding a packet to a compromised node called $M$. Node $M$ is supposed to relay the packet to the next-hop node $D$. In packet misrouting, $M$ relays the packet to the wrong next-hop neighbor, i.e., a neighbor node other than $D$. The result is that the packet does not reach its intended next-hop ($D$) while $M$ appears to the guards as doing its job correctly. Therefore, the adversary not only can successfully perform the attack, but also it hides the presence of its malicious activities. Additionally, a legitimate node ($D$) is accused of packet dropping.

I provide here a protocol called MiMi (Mitigating Packet Misrouting in Locally-Monitored Multi-hop Wireless Ad Hoc Networks) that is constructed on local monitoring and that can mitigate the misrouting attack. The MiMi mitigation technique takes two forms—having guard nodes maintain additional next-hop information gathered during route establishment, and adding some checking responsibility to each neighbor. On the other hand, I show that basic LM [4]-[8] is unable to detect any instance of the packet misrouting attack. To the best of my knowledge, I am the first to propose a protocol suited to resource constrained wireless networks that can countermeasure the packet misrouting attack using local monitoring.
I build a simulation model for the misrouting attack using ns-2 and perform a comparative evaluation of local monitoring with and without MiMi. The simulation results show that MiMi can deliver 60% of packets to the destination under 20% nodes compromised, while basic LM fails to deliver almost any packet. The likelihood of framing of legitimate nodes is also three-folds under basic LM for the same network (results are not presented due to space limitations). The performance advantages come at the expense of a slightly higher false isolation (due to natural collisions on the wireless channel) and end-to-end delay for packets in MiMi.

I summarize my contributions in this paper as follows:

1. I present the packet misrouting attack and detail how it can be launched in locally-monitored networks without the malicious node being detected.
2. I provide mitigation remedy for the packet misrouting attack with minimal addition to the resource consumption and responsibility of a node over baseline local monitoring.
3. I show through simulations the security advantage of MiMi over basic LM.

The rest of the paper is organized as follows. Section II presents the related work. Section III presents the foundations and background knowledge. Section IV describes the packet misrouting attack. Section V presents the mitigation technique. Sections VI presents the performance analysis of MiMi. Finally, Section VII concludes the paper.

II. RELATED WORK

In the last few years, researchers have been actively exploring many mechanisms to ensure the security of control and data traffic in wireless networks. The path diversity techniques increase route robustness by first discovering multi-path routes [7, 10] and then using these paths to provide redundancy in the data transmission between a source and a destination. The data is encoded and divided into multiple shares sent to the destination via different routes. The method is effective in well-connected networks, but does not provide enough path diversity in sparse networks. Moreover, many of these schemes are expensive for resource-constrained wireless networks due to the data redundancy. Additionally, these protocols could be vulnerable to route discovery attacks, such as the Sybil attack, that prevent the discovery of non-adversarial paths.

A technique proposed to detect malicious behavior involving selective dropping of data, relies on explicit acknowledgement for received data [10]. However, the diagnosis process used can only work with a source routing protocol, such as DSR, where the source knows all the intermediate nodes to the destination. Moreover, the diagnosis packets have to be encrypted with the shared key between the source and the intermediate nodes. Furthermore, the isolation is done per link rather than per node, i.e., with every malicious node blacklisted another honest node is blacklisted. Finally, the blacklisting of malicious nodes is done at the source of packet and not locally at the neighbors of the malicious node. Therefore, even if the malicious node had been blacklisted by some nodes in the network, it continues to be active and cause harm to traffic from other sources. In contrast, in MiMi, the node is detected and diagnosed locally by its neighbors, which is speedier and the malicious node can be isolated locally as in [4].

A widely used technique for mitigating control and data attacks in multi-hop wireless networks is cooperative local monitoring by overhearing traffic in the vicinity. The idea of overhearing traffic in the vicinity (e.g. [1]-[2]) has been used to build trust relationships among nodes in networks (e.g. [1]), detect and mitigate certain kinds of attacks (e.g. [2], [4]-[6]), or discover routes with certain desirable properties, such as being node disjoint (e.g. [10]). Our previous protocol called LiteWorp [4] used local monitoring to address a specific control attack, called the wormhole attack in static sensor networks. The follow-on work in [6] generalized the detection mechanism to detect a wider class of control attacks against static sensor networks. The MobiWorp protocol [5] uses local monitoring to mitigate the wormhole attack in hybrid (static & mobile) sensor networks. However, local monitoring, as used by us and other researchers, fails to mitigate the packet misrouting attack presented here.

III. FOUNDATIONS

A. Attack Model and System Assumptions

Attack model: An attacker can control an internal node, which, since it possesses the keys, can be authenticated by other nodes in the network. An insider node may be created, for example, by compromising a legitimate node.

System assumptions: MiMi assumes that all the legitimate communication links are bi-directional. MiMi assumes that secure neighbor discovery has been performed and that every node knows both first and second hop information. This can be achieved through our protocol described in [17] as well as by approaches developed by other researchers [3]. Note that while this knowledge is enormously useful, this by itself cannot mitigate many attack types. For example, as has been pointed out in [4], further work is needed to detect the wormhole attack. Intuitively this information subsets the nodes from which a given node will accept packets but does not eliminate the possibility of malicious nodes within that subset. Local monitoring assumes that the network has sufficient redundancy, such that each node has more than an application defined threshold number of legitimate nodes as guards. MiMi assumes a key management protocol, e.g., [11], exists such that any two nodes can communicate securely.

B. Background: Local Monitoring

Local monitoring is a collaborative detection strategy where a node monitors the control traffic going in and out of its neighbors. This strategy was introduced in [4] for static sensor networks and here I give the background needed to understand the concepts presented in this paper.

For a node, say a, to be able to watch a node, say N2, a must be a neighbor of both N2 and the previous hop from N2, say N1. Then I call a a guard node for N2 over the link N1→N2. I use the notation R(N) to denote the set of all nodes that are within the radio range of node N and G(N1, N2) to denote the set of all guard nodes for N2 over a link N1→N2. Formally, G(N1, N2) = N1 ∩ R(N2) − N1.amiento, where N2 ∈ R(N1). For example, in Figure 1, G(XA) = MNX.

![Diagram](attachment:image1.png)

Figure 1: X, M, and N are guards of A over X→A
Information from each packet sent from X to A is saved in a watch buffer at each guard. The guards expect that A will forward the packet toward the ultimate destination, unless A is itself the destination. Each entry in the watch buffer is time stamped with a time threshold, τ, by which A must forward the packet. Each packet forwarded by A with X as a previous hop is checked for the corresponding information in the watch buffer. The check can be to verify if the packet is fabricated or duplicated (no corresponding entry in the buffer), corrupted (no matching hash of the payload), dropped or delayed (entry is not matched within τ).

A malicious counter (MalC(i,j)) is maintained at each guard node, i, for a node, j, at the receiving end of each link that i is monitoring over a sliding window of length T win. MalC(i,j) is incremented for any malicious activity of j detected by i. The increment to MalC depends on the nature of the malicious activity, being higher for more severe infractions. When the growth in the counter value maintained by a guard node i for node j (MalC(i,j)) crosses a threshold rate (MalC(i)) over T win, node i revokes j from its neighbor list (called direct isolation since it will henceforth not perform any communication with node j), and sends to each neighbor of j, an authenticated alert message indicating j is a suspected malicious node. When a neighbor N j gets the alert, it verifies the authenticity of the alert message. When N j gets enough alert messages about j, it marks the status of j as revoked (called indirect isolation). The notion of enough number of alerts is quantified by the detection confidence index γ. Each node maintains a memory of nodes that it has revoked through a local blacklist so that a malicious node cannot come back to its neighborhood and claim to be blameless. This constitutes local isolation of a malicious node by its current neighbors.

IV. PACKET MISROUTING ATTACK DESCRIPTION

In packet misrouting, a malicious node relays the packet to the wrong next-hop, which results in a packet drop. Note that, in basic LM [4], a node that receives a packet to relay without being in the route to the destination either drops the packet or sends a one-hop broadcast that it has no route to the destination. The authors in [4] argue that the latter case would be more expensive and dangerous since it gives malicious nodes valid excuses to drop packets. Therefore, they go with the first choice, even though it may result in some false accusations. In this attack, a malicious intermediate node achieves the same objective as if it were dropping a packet. However, none of the guard nodes using basic LM become any wiser due to the action. In addition, some legitimate node is accused of packet dropping.

V. MITIGATING PACKET MISROUTING

To detect packet misrouting, the local monitoring mechanism [4] has to incorporate additional functionality and information. The basic idea is to extend the knowledge at each guard to include the identity of the next-hop of the packet being relayed. This additional knowledge can be collected during route establishment. Many multi-hop wireless routing protocols provide this knowledge without any modification while some changes are necessary in others. The first class includes both reactive routing protocols such as Dynamic Source Routing (DSR) and its variants [12] and proactive routing protocols such as TinyOS beacon routing [14] and Destination Sequenced Distance Vector routing (DSDV [15]). In all source routing protocols, the packet header carries the identity of all the nodes in the route from the source to the destination. Therefore, no additional traffic is required to be generated for the guard nodes to be able to detect this kind of attack. Moreover, no additional information is required to be maintained at the guards since each packet carries the required information in its header. In TinyOS beacon routing, the base station periodically broadcasts a beacon to establish a breadth first search tree rooted at the base station. Each node within the transmission range of the base station overhears the beacon, sets its parent to be the base station, sets the hop count to the base station to be one, and rebroadcasts the beacon. Each beacon carries the identity of the broadcasting node, the identity of its parent, and the hop count to the base station. Each guard overhearing the beacon broadcasting saves parent node identity for each neighbor. Later, when a node, say B, is sent a packet to relay, the guard of B can detect any misrouting by B since it knows the next-hop en route to the base station.

The second class of routing protocols requires modification to the protocol to build the next-hop information at the guards. Examples of these protocols are the reactive routing protocols that use control packet flooding of route requests (REQ) and route replies (REP) to establish the route between the source and the destination (e.g., LSR [6] and AODV [13]). In these protocols, when a source node desires to send a message to some destination node and does not already have a valid route to that destination, it initiates a route discovery process to locate the other node. It broadcasts a route request (REQ) packet to its neighbors, which then forward the request to their neighbors, and so on, until either the destination or an intermediate node with a “fresh enough” route to the destination is located. During the process of forwarding the REQ, intermediate nodes record in their route tables the address of the neighbor from which the first copy of the broadcast packet is received, thereby establishing a reverse path. Once the REQ reaches the destination, the destination node responds by unicasting a route reply (REP) packet back to the neighbor from which it first received the REQ. As the REP traverses along the reverse path, nodes along this path set up forward route entries in their route tables which point to the node from which the REP came.

Next, I show the required changes to the basic version of AODV to enable the guards to build the necessary knowledge for detecting the misrouting attack. The idea behind the solution is that during route establishment, when the relation about which node to forward a
packet between a given source-destination pair is determined, this information is broadcast by a neighbor to the guards which will be responsible for monitoring the node. To collect the next-hop identity information, the forwarder of the \( RREQ \) attaches the previous two hops to the \( RREQ \) packet header. Let the previous hop of \( M \) be \( A \) for a route from source \( S \) to destination \( D \), and the next hop from \( M \) be \( B \) (Figure 2). When \( M \) broadcasts the \( RREQ \) received from \( A \), it includes the identity of \( A \) and its own identity \( (M) \) in the \( RREQ \) header \(<S, D, RREQ_id, A, M>\). When \( B \) and the other neighbors of \( M \) get the \( RREQ \) from \( M \), they keep in a Verification Table (VT) \(<S, D, RREQ_id, A, M, \cdot \cdot \cdot \>\) (last field is currently blank). When \( B \) broadcasts the \( RREQ \), the common neighbors of \( M \) and \( B \) update their VT to include \( B, <S, D, RREQ_id, A, M, B>\). When \( B \) receives a \( RREP \) to be relayed to \( M \), it includes in that \( RREP \) the identity of the node that \( M \) needs to relay the \( RREP \) to, which is \( A \) in this example. Therefore, all the guards of \( M \) now know that \( M \) not only needs to forward the \( RREP \) but also that it should forward it to \( A \) and not any other neighbor.

Therefore, two tasks have been added to the functionality of the guards in monitoring the \( RREP \) packets. First, the guard \( G \) of a node \( N \) verifies that \( N \) forwards the \( RREP \) to the correct next-hop. In the example above, \( G_2 \) verifies that \( M \) forwards the \( RREP \) to \( A \). Second, \( G \) verifies that \( N \) has updated the forwarded \( RREP \) header correctly. In the example shown above, \( G_2 \) verifies that when the input packet to \( M \) from \( B \) is \(<RREP, S, D, RREQ_id, C, B, M>\), then the output packet from \( M \) should be \(<RREP, S, D, RREQ_id, B, M, A>\). Thus \( M \) and its guards over the link \( B \rightarrow M \) know that the next-hop is \( A \) from the information built in the VT table during the \( RREQ \) flooding.

Using the additional information mentioned above, MiMi detects misrouting attacks as follows. In the example above, assume that \( S \) is sending a data packet to \( D \) through a route that includes \(<Y,A,M,B,C>\). The malicious node \( M \) cannot misroute the data packet received from \( A \) to a node other than the next-hop, \( B \), since each guard of \( M \) over the link \( A \rightarrow M \) has an entry in its VT which indicates \( B \) is the correct next-hop. This results in an additional checking activity for the guard node involved in local monitoring—verifying the data packet is forwarded to the correct next-hop, as indicated by the entry in the guard node’s VT. Moreover, \( M \) cannot frame another neighbor, say \( X \), by misrouting the packet to \( X \). The guards of \( X \) over \( M \rightarrow X \) do not have an entry like \(<S, D, RREQ_id, Y, A, M, X>\) and therefore, they would not increment the MalC of \( X \) when it drops the packet.

VI. PERFORMANCE ANALYSIS

I use the ns-2 simulation environment [16] to simulate a data exchange protocol, individually with basic LM and with MiMi. I distribute the nodes randomly over a square field with a fixed average node density. I use a generic on-demand shortest path routing protocol that floods route requests and unicasts route replies in the reverse direction. When a malicious node gets a data packet, it relays the packet to a wrong next-hop with a probability of \( f_{\text{data}} \). The simulation also accounts for losses due to natural collisions.

**Input parameters:** Each node generates data using an exponential random variable with inter-arrival rate \( \phi \). The destination is chosen at random and used for a random time following an exponential distribution with rate \( \xi \). I use \( N_M \) for the number of malicious nodes and \( N \) for the total number of nodes. The input parameters with the experimental values are given in Table 1, I use the same settings as in [4] so that the results are comparable.

**Output parameters:** The output parameters include (i) the fraction of data packets received (delivery ratio) calculated as the total number of packets successfully received by final destinations over the total number of packets sent, (ii) the false isolation ratio, which is defined as the fraction of good nodes that have been isolated due to natural causes (collisions and losses on the wireless channel) over the total number of good nodes, (iii) the malicious node isolation ratio (true isolation), which is defined as the number of malicious nodes isolated to the total number of malicious nodes, (iv) the average end-to-end delay of data packets, which is the time a packet takes after leaving the source until it reaches its final destination. The output parameters that I present here are measured at the end of the simulation time (1500 s). The output parameters are obtained by averaging over 30 runs.

**Table 1: Input parameters for MiMi simulation**

<table>
<thead>
<tr>
<th>Param.</th>
<th>Value</th>
<th>Param.</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx Range ((r))</td>
<td>30 m</td>
<td>(\xi, \phi)</td>
<td>0.02, 0.2</td>
</tr>
<tr>
<td>MalC increment</td>
<td>15</td>
<td>(f_{\text{data}})</td>
<td>0.6</td>
</tr>
<tr>
<td>Timeout, (T_{\text{con}})</td>
<td>50 s, 200</td>
<td>(N_M)</td>
<td>0-20</td>
</tr>
<tr>
<td>MalC, (\gamma)</td>
<td>150, 3</td>
<td>(\tau)</td>
<td>0.5 sec</td>
</tr>
<tr>
<td># nodes ((N))</td>
<td>100</td>
<td>BW</td>
<td>40 kbps</td>
</tr>
</tbody>
</table>

Figure 3 (a) shows the variations in delivery ratio as the number of malicious nodes varies. The figure shows that the delivery ratio decreases as \(N_M\) increases. This is due to the packets dropped before the malicious nodes are detected and isolated. As \(N_M\) increases, this initial drop increases and thus the delivery ratio decreases. Moreover, as \(N_M\) increases, the true isolation decreases (Figure 3 (b)). Therefore, the malicious nodes that could not be detected continue to drop packets and this decreases the delivery ratio. The delivery ratio in basic LM is much less than in MiMi and the difference increases as the number of malicious nodes increases. This is due to two main reasons. The first is that basic LM fails to detect any of the malicious nodes and thus they continue to drop packets constantly. The second is that some of the good nodes in basic LM get framed by the adversary and thus become isolated and reduce the overall throughput.

Figure 3 (b) shows the variations in true isolation as \(N_M\) varies. Most significantly, we observe that basic LM has a very poor performance while MiMi achieves above 80% isolation of malicious nodes with up to 12% compromised nodes. The figure shows that the true isolation decreases as we increase \(N_M\). This is because the number of available guards and comparators in the network decreases as more and more nodes get compromised. Furthermore, as \(N_M\) increases, local isolation becomes less effective since the number of legitimate neighbors decreases and if this goes below \(\gamma\), then local isolation has to wait for direct isolation individually by each legitimate neighbor. Moreover, as \(N_M\) increases the data traffic in the network decreases (in the simulation malicious nodes do not send data) which results in a decrease in the number of packets that a single malicious node may drop. This in turn results in decreasing the likelihood that the malicious node is detected and isolated.
Figure 4 (a) shows the variations in false isolation as $N_M$ varies. The figure shows that the false isolation initially increases as we increase $N_M$ and starts to decrease beyond a point. This is because not all guard nodes come to the decision to isolate a malicious node at the same time. Therefore, a given guard node may suspect another guard node when the latter isolates a malicious node but the former still has not. The occurrence of this situation increases with $N_M$ and hence the false isolation increases with $N_M$. For example, a guard node $G_I$ detects a malicious node $M$ earlier than the other guard nodes for the link to $M$. $G_I$ subsequently drops all the traffic forwarded to $M$ and is therefore suspected by other guard nodes of $M$. This problem can be solved by having an authenticated one-hop broadcast whenever a guard node performs a local detection. An opposing pull comes from the fact that the number of good nodes decreases as we increase $N_M$. This in turn results in a decrease of the indirect false isolation since a node may not have more than $\gamma$ good nodes to agree on falsely isolating a neighbor. Moreover, as $N_M$ increases, the data traffic decreases since malicious nodes are not generating data. This in turn decreases the chance for collisions and consequently decreases false isolation. Beyond a point ($N_M=6$), the latter factors dominate the first factor and there is a decrease in false isolation ratio. The false isolation in MIMI is slightly higher than in basic LM due to more aggressive detection with an increased level of monitoring (all neighbors, not just the guard nodes and also additional checking by each monitoring node).

Figure 4 (b) shows the variations in end-to-end delay as $N_M$ is varied. The figure shows that the end-to-end delay initially increases as $N_M$ increases and then starts to decrease. As $N_M$ increases, the route reestablishment frequency increases. This is due to the fact that a route remains active for a time $T_{out} \text{route}$ and this timer is reset with every packet forwarded using that route. Consequently cutting the flow of packets (by maliciously dropping the packet) causes the route entries to stale. Therefore, additional traffic is generated to reestablish the route which increases the end-to-end delay. The opposing pull comes from the fact that as $N_M$ increases the traffic decreases. This reduces the contention in the network which in turn decreases the end-to-end delay. As $N_M$ increases beyond a point, the latter factor dominates and the overall result is a decrease in the end-to-end delay. The end-to-end delay in MIMI is slightly higher than in basic LM. This is due to the modification of the routing protocol in MIMI (to collect next-hop information) which makes the route establishment time slightly higher.

Figure 4: Effect of $N_M$ on: (a) False isolation; (b) End-to-end delay

VII. CONCLUSION

I have introduced a new class of attacks called *stealthy packet dropping* through packet misrouting, which disrupts a packet from reaching the destination by malicious behavior at an intermediate node. However, the malicious behavior cannot be detected by any behavior-based detection scheme presented to date. Specifically, I showed that local monitoring based detection which relies on overseeing behavior of a neighboring node cannot detect these attacks. I then presented a protocol called MiMI based on local monitoring to remedy the attack. The solution takes two forms - having nodes maintain additional routing path information, and adding some checking responsibility to each neighbor. I showed through performance analysis that basic LM fails to mitigate the packet misrouting attack while MiMI successfully mitigates it.

In future work, I am considering detection techniques for multi-channel wireless networks. The listening activity for detecting malicious behavior is more complicated due to the presence of multiple channels. I also plan to analyze the impact of the detection technique on the network throughput under different adversary models.

VIII. REFERENCES


