Witness-based Detection of Forwarding Misbehaviors in Wireless Networks

Sookhyun Yang*, Sudarshan Vasudevan*, and Jim Kurose*

*Dept. Computer Science, University of Massachusetts, Amherst, MA, 01003, Email: {shyang, svasu, kurose}@cs.umass.edu

Abstract—We consider the problem of identifying a node that incorrectly forwards packets in a static wireless ad hoc network. We propose a detection scheme that identifies a misbehaving node based on observations made by neighboring nodes (“witnesses”) near the forwarding node. Without longterm or cumulative observation, the proposed scheme makes an instantaneous decision about whether a node is correctly forwarding a packet. Through extensive analysis under various threat scenarios, we show that our scheme can unambiguously identify a misbehaving node when there is no collusion. We analyze our scheme’s detection accuracy accounting for the lossy nature of wireless links, finding that our scheme can achieve high detection accuracy while incurring low communication overhead.

1. INTRODUCTION

In a wireless ad hoc network, each node transmits data packets to a destination via intermediate nodes on a path. A number of secure ad hoc routing protocols have been proposed for guaranteeing secure end-end data transmission. In these solutions, signed routing messages prevent an unauthenticated node from joining a network. However, these schemes suffer from one important drawback: if a previously authenticated node becomes compromised, there are no safeguards to identify and thwart adversaries that misuse the compromised node. Consequently, a path verification mechanism (i.e., a mechanism to ensure that each node along an end-end path is correctly forwarding packets) is necessary to observe and identify such a misbehaving node. The detection of compromised nodes is particularly important in military MANETS [4, 7, 8] and other networks in which nodes can be compromised either physically or remotely.

Path-verification mechanisms can be classified into two categories: control-plane verification mechanisms, and data-plane verification mechanisms. Control-plane verification mechanisms detect routing disruption attacks that inject false routing control messages. Data-plane verification deals with data forwarding misbehavior attacks such as packet drops, reordering, and message corruption. In this paper, we focus on detecting forwarding misbehavior attacks and identifying the source of attacks in the data plane.

Prior work on detecting forwarding misbehavior can be divided into two broad categories, based on whether evidence of forwarding behavior is gathered passively or actively. In watchdog schemes employing passive measurements [1, 2, 5], each node typically monitors its neighbor node’s forwarding behavior by matching the neighbor node’s incoming and outgoing data packet pairs. To reduce a high rate of false positives, some of these schemes derive a mismatch rate of a series of incoming and outgoing data packet pairs. Other schemes also attempt to precisely decide the forwarding nodes’ states by sharing observations with multiple collaborating nodes. However, solutions relying on such accumulative schemes have an inevitable detection delay before detecting misbehavior. Schemes relying on observations from multiple collaborating nodes can be easily thwarted by sending out false information signed using the stolen key if a node becomes compromised. The second class of schemes (e.g., [3]) involve a node actively sending a probe packet to its observing node. The absence of a response from the observing node within a predefined time is used as evidence to incriminate an intermediate node. These schemes are also vulnerable to an attacker generating a valid probe response using a stolen key without correctly forwarding a data packet.

Our solution for detecting forwarding misbehavior is similar in a spirit to watchdog schemes. However, our solution differs from previous approaches in two important ways:

- Our solution detects forwarding misbehavior after one round of observation and is not based on long-term statistics of a node’s forwarding behavior.
- Previous solutions have primarily addressed packet drop attacks and the detection accuracy of these solutions degrades in the presence of multiple independent compromised nodes. Our work considers a wide variety of attacks (in addition to packet drop) and evaluates detection accuracy in the presence of these attacks.

In this paper, we present a witness-based detection scheme that utilizes witness nodes (i.e., nodes in the neighborhood of a forwarding node) that observe a forwarding node’s behavior, and help perform packet-by-packet and accurate detection of forwarding misbehavior. The main contributions of this paper can be summarized as follows:

- We propose a scheme that uses a tamper-proof evidence format to record the forwarding behavior of a node.
- We show that our scheme can unambiguously identify a compromised node, as long as there is at least one uncompromised observing node and compromised nodes do not collude.
- Using an analytical model, we show that our scheme can achieve very low false positive and false negative rates in a lossy wireless network. Furthermore, we show that our scheme requires relatively low communication overhead.

* The authors wish to acknowledge the support of the Army Research Laboratory through the Army University Research Initiative (AURI) program under contract W911NF-08-1-0171.
while achieving high accuracy.

The rest of this paper is organized as follows. In section II, we list our assumptions, define forwarding misbehavior and describe two detection schemes. In section III, we describe the witness-based detection scheme in more detail. Section IV shows how our scheme detects attacks in various threat scenarios. In section V, we evaluate our scheme’s detection accuracy and communication overhead in the presence of lossy links. In the last section, we conclude our paper.

II. DETECTING FORWARDING MISBEHAVIOR

In this section, we list our assumptions, and define various forwarding misbehavior attacks. Then, we briefly describe two detection schemes: data-path-based detection and witness-based detection.

A. Assumptions

1) We consider a static wireless ad hoc network that is composed of authenticated nodes using public-key authentication.
2) Each node in a network has a public/private key pair.
3) We consider the case that a previously authenticated node is compromised and its private key is stolen.
4) We consider a decentralized approach (i.e., each node on a source-destination path monitors the forwarding behavior of its downstream node).
5) We assume that each node on a source-destination path knows the next hop from its downstream node (i.e., the node two hops away along the path to the destination). Such information is readily available in the case of link-state routing algorithms and distance vector routing algorithms using source routing (e.g., DSR).
6) We assume that the data path is fixed during a detection procedure.

B. Forwarding misbehavior attacks

The compromised node then launches one or more of the following forwarding misbehavior attacks:

- **Drop (Blackhole or Grayhole)** is an attack in which a compromised node does not forward a data packet. This drop attack includes complete, partial, or selective dropping of packets.
- **Fake forwarding** is an attack in which a compromised node forwards a data packet to a nonexistent node.
- **Route deviation** is an attack in which a compromised node forwards a data packet to an incorrect next-hop neighbor.
- **Power control** is an attack in which a compromised node forwards a data packet with insufficient transmission power, causing the data packet to be unreachable to its next-hop neighbor on the data path.
- **Reorder (Jellyfish)** is an attack in which a compromised node forwards a data packet out-of-order.
- **Message corruption** is an attack in which a compromised node corrupts a message field in a data packet.

C. Two forwarding misbehavior detection schemes

We first describe reliable hop-by-hop data forwarding, where each node on a data path exchanges Data and ACK packets with its next-hop neighbor as part of the normal forwarding of packets. Without loss of generality, we consider data path $S \rightarrow A \rightarrow B \rightarrow C \rightarrow D$, where $S$ is a source, $D$ the ultimate destination, and $A$, $B$, $C$ are intermediate nodes. From now on, we focus on node $B$’s forwarding behavior verification by its upstream node $A$. We call node $A$ a judge and node $B$ a defendant. Judge $A$ is a “trusted” node. As evidence of defendant $B$’s forwarding, we use the Data packet from node $B$ to node $C$ and the ACK packet from node $C$ to node $B$. A witness is a node who may overhear the Data or ACK packet, and is located within node $B$ or node $C$’s transmission radius. Let $W$ be the set of witness nodes excluding nodes $B$ and $C$. Note that nodes $B$, $C$ or witness nodes can be compromised. Our scheme must be able to operate in the face of any such compromises, as discussed below.

Now we introduce two forwarding misbehavior detection schemes, which differ in the role of witness nodes.

- **Data-path-based detection** was suggested in [3]. Without the intervention of witnesses, data-path-based detection only relies on nodes on the data path. As evidence, the ACK packet from downstream neighbor $C$ reaches judge node $A$ through defendant node $B$. As we will see, this ACK will need to be signed by node $C$. Node $A$ makes a decision on the state of defendant node $B$ using this ACK evidence.

- **Witness-based detection** is our newly proposed scheme. Witnesses operating in promiscuous mode, can overhear a Data packet as well as an ACK packet as evidence, and transmit this evidence through diverse paths to judge node $A$. For decision making, node $A$ utilizes Data and ACK evidence received from both defendant node $B$ and a witness node in set $W$. If there are no witnesses in range of nodes $B$ and $C$, then this approach is the same as the data path approach.

The schemes operate by having each node on a data path verify the forwarding behavior of its next hop neighbor, cumulatively resulting in the verification of every node in the path from $S$ to $D$.

III. WITNESS-BASED DETECTION

Let us focus our attention on the data path $A-B-C$. Our witness-based detection scheme consists of three sequential steps: 1) Evidence generation, 2) Evidence dissemination, and 3) Judge’s decision.

A. Evidence generation

The evidence-generation step adds two additional fields into a traditional link-level Data packet: an evidence field and a judge address field. The evidence field in a Data packet will ensure that the evidence of node $B$’s forwarding behavior can be created only if node $B$ forwards that Data packet. The judge address field is the address to which evidence (discussed
Evidence dissemination

Signed message checksum: The checksum is a one-way hash (e.g., MD5) of the message and the message’s next-hop recipient address (e.g., addr(C)). \( r_B \) denotes the checksum for a message sent by node B. This checksum field is signed by the Data packet forwarder. \( K_B(r_B) \) denotes the signed checksum subfield in a packet sent by node B, where \( K_B \) is node B’s private key.

Timestamp: This subfield denotes the time at which the message containing this evidence field was generated and sent, where we assume a global clock in a network, or Lamport-like time ordering. This subfield is included only if the network needs to detect a “reorder” attack.

Node B sends the following Data packet (where M is a message) to node C, which can be overheard by node W:

\[
B \rightarrow \{C, W\} : K_B(r_B), addr(A), M
\]

The checksum and timestamp subfields, unique for each message, are used to prevent replay attacks.

Having described the Data packet format, we next describe the two types of evidence (Data-based and ACK-based). As we will see, Data-based evidence is used to detect every forwarding misbehavior described in Section II.B except the power-control attack, which is detected using ACK-based evidence.

Data-based evidence generation: A witness node w in set \( W \) that successfully overhears a Data packet constructs Data-based evidence in the following manner:

1) extracts \( K_B(r_B) \) from node B’s Data packet.
2) computes a message checksum \( (r_w) \) with node B’s Data packet, as described earlier.
3) records the timestamp \( (t_w) \) when it successfully overhears the Data packet.
4) concatenates and signs these three pieces of information using its private key: \( K_w(K_B(r_B), r_w, t_w) \).

ACK-based evidence generation: Using a similar procedure as that employed in Data-based evidence generation, node C generates ACK-based evidence, \( K_C(K_B(r_B), r_C, t_C) \).

Note that the size of the sent evidence is relatively small, since it consists of a signed message hash, not the message itself. In order to reduce complexity of per-packet signature, witness nodes and node C can generate aggregated evidence hashing multiple messages.

B. Evidence dissemination

In this step, witness nodes and node C that generate evidence transmit Data-based and ACK-based evidence to judge node A using new control packets, ED and ED_ACK.

ED packet: Node C and each of the witness nodes use an ED packet to transmit evidence. The ED packet has an evidence field and an evidence type field to distinguish the type of evidence in an evidence field.

ED_ACK packet: Judge node A acknowledges the ED packet using an ED_ACK packet that also consists of an evidence type and an evidence field. The ED_ACK packet traverses the data path (A-B-C) to reach all nodes participating in evidence dissemination. Using the evidence type field, node A marks successfully received evidence, which allows ED_ACK receivers to retransmit evidence that node A is missing. The evidence field contains the message checksum signed by node A and prevents malicious ED_ACK packets sent by a node other than node A.

In a witness node’s evidence dissemination, communication overhead during the evidence dissemination step is directly proportional to witness node density. In order to reduce communication overhead, witness nodes perform randomized feedback suppression. Witness nodes stay idle for a random time up to a maximum backoff duration. Witness nodes are suppressed if they overhear an ED_ACK packet that contains more than or equal to the amount of evidence that they have to transmit. Highest priority is given to a witness node having both Data-based and ACK-based evidence by setting the shortest maximum backoff duration.

C. Judge’s decision

Before describing judge node A’s decision making algorithm, we first define variables used in the algorithm and introduce the notion of evidence consistency. Using the variables in the Table I, when node A receives evidence \( e_n \) associated with Data packet \( n \), node A evaluates \( S(e_n) \) using equation (1). Equation (1) first compares the equality of message checksum in the first and second terms. The remaining terms check the correctness of message order. If \( S(e_n) \) evaluates to true, we say that the evidence \( e_n \) is consistent. Otherwise, we call \( e_n \) inconsistent.

\[
S(e_n) \leftarrow (r_n = e_n, r_B) \land (t_n = t_n, t_B) \land (t_n < e_n, t_x) \\
\land (e_n, t_x < e_n, t_x), \text{ where } e_n, t_x = \max_{X \in [X, 1 \leq n \leq n-1]} e_i, t_x
\]

Due to space constraints, we only explains a decision making algorithm without collusion among compromised nodes, which determines whether node B is compromised based on evidence consistency. The collusion case is discussed in [9] (Collusion is an attack in which compromised node x and compromised node B generate fake consistent evidence together by including a false message checksum or timestamp to conceal node B’s forwarding misbehavior). Note that a Sybil attack is equivalent to the case of multiple compromised nodes (i.e., the Sybil node is a single node with multiple identities). Our algorithm can detect the existence (but not
the identities of colluding nodes as long as there is at least one uncompromised node.

The algorithm makes a precise decision regarding the compromised node(s) using two cheat-proof lemmas below with the following assumptions (The proof of lemmas is found in [9]):

1) Generated evidence successfully reaches node A.
2) At least one uncompromised node \( S \in W \cup \{C\} \) is present.

Lemma 1. Absence of evidence implies that defendant node B is compromised.

Lemma 2. Suppose that there is no collusion. Consistent evidence exists iff defendant node B is not compromised.

Algorithm 1 Identify defendant node’s state (w/o collusion)

1: \( S_n \leftarrow \) false
2: while \( T > 0 \) do
3: \( S_n \leftarrow S_n \lor S(e_n) \)
4: if \( (S_n = \text{true}) \land (addr_n = e_n.addr(x)) \) then
5: \( \text{return} \) uncompromised
6: end if
7: end while
8: if \( S_n = \text{true} \) then
9: \( \text{return} \) suspicious
10: end if
11: \( \text{return} \) compromised

The algorithm works as follows. The algorithm initially sets \( S_n \) false, where \( S_n \leftarrow S_n \lor S(e_n) \). Whenever evidence \( e_n \) is received, the algorithm updates \( S_n \). After a predefined time, if \( S_n \) evaluates to true (in other words, there exists at least one consistent piece of evidence) and if the evidence is ACK-based evidence, Algorithm 1 decides that the defendant node is not compromised, in accordance with Lemma 2. However, if \( S_n \) is true but all evidence is Data-based evidence, node A classifies node B’s state as suspicious, since the absence of ACK-based evidence from node C leaves open the possibility of a power-control attack by node B, where node B transmits with just enough power to be overhead by the witness nodes, but not high enough to reach node C. Last, if \( S_n \) is not true, node B is deemed compromised, based on Lemmas 1 and 2.

IV. DETECTION PROPERTIES

We now show how the algorithm identifies a compromised node launching various forwarding attacks described in Section II. We also show that the algorithm is invulnerable to two new attacks defined below, which attempt to circumvent the witness-based detection scheme:

- **Bypassing**: Compromised node B that launches forwarding misbehaviors attempts to circumvent the witness-based detection scheme by including a false message checksum or judge’s address in a Data packet.
- **Badmouthing**: Compromised node \( x \in X \) generates evidence that falsely accuses uncompromised node B.

A. Forwarding misbehavior

We describe how our witness-based detection scheme identifies misbehaving node B, when the remaining nodes behave correctly and do not launch the two attacks described above.

- **Drop** attack results from node B not forwarding a Data packet and causes the absence of both Data-based evidence and ACK-based evidence at node A. In this case, \( S_n \) evaluates to false because node A has no evidence.
- **Fake forwarding** and **Route deviation** attacks are detected, based on inconsistent Data-based evidence, which contains different message checksums. Let us consider the scenario in which node B maliciously forwards its Data packet to node E (where node C is on the correct data path). When node E is non-existent, we call the attack fake forwarding. Otherwise, the attack is called a route deviation attack. For every received Data-based evidence \( e_n = K_w(K_B(r_B), r_w, t_w) \), it is easy to see that \( S(e_n) \) evaluates to false, because \( r_n \neq e_n.r_B \) and \( r_n \neq e_n.r_w \), where \( r_n = H[M|addr(C)] \) and \( e_n.r_B = e_n.r_w = H[M|addr(E)] \). A route deviation attack is also detectable using node E’s inconsistent ACK-based evidence as well. That is, fake forwarding and route deviation attacks can be distinguished, based on whether or not node A receives ACK-based evidence.
- **Power control** attack allows for node A to receive consistent Data-based evidence. However, it also results in the absence of ACK-based evidence. After expiry of a timeout period, node A determines node B to be suspicious, since A cannot distinguish between the case that the ACK-based evidence is lost due to link losses and the case that B launches a power control attack. Power control attacks are feasible only when node C is farther from node B than node A and each of the witness nodes (assuming homogeneous wireless signal propagation). Note that this attack requires precise distance calculations by node B to its neighboring nodes in order for it to adjust its transmission power appropriately.
- **Reorder** attacks are detected based on the timestamps. Suppose that compromised B transmits Data packet \( n \) before Data packet \( n-1 \), where the correct sequence of two Data packets is \( n \) after \( n-1 \). Each witness node overhears Data packet \( n \) before \( n-1 \). Since \( e_{n-1}.t_X > e_n.t_X \) for all evidence, \( S_n \) evaluates to false. Note that the normal case when uncompromised node B transmits packet \( n \) before packet \( n-1 \) as retransmission of packet \( n-1 \) is distinguishable using evidence generated from the first transmission of packet \( n-1 \).
- **Message corruption** attack is detected when node A receives evidence with different checksums. Let \( M' \) be the message transmitted by node B in its Data packet, which is different from the original message M received from node A. For every received evidence, the message checksums are not equal, in that \( r_n \neq e_n.r_B \) and \( r_n \neq e_n.r_X \), where \( r_n = H[M|addr(C)] \) and \( e_n.r_B = e_n.r_X = H[M'|addr(C)] \). Thus, \( S_n \) evaluates
C. Badmouthing

Based on Lemma 2.

B. Bypassing

In addition to launching forwarding misbehavior attacks, compromised node \( B \) can potentially disrupt our witness-based detection scheme by launching a bypassing attack. We explain why our witness-based detection scheme cannot be disrupted by such an attack. First, a false judge’s address results in node \( A \) receiving no evidence because evidence-generating nodes cannot transmit evidence to judge \( A \). This attack causes node \( A \) to decide that node \( B \) is compromised based on Lemma 1. Second, the compromised node \( B \) may attempt to manipulate the evidence field in a Data packet so as to hide its forwarding misbehaviors. For instance, suppose that node \( B \) launches a message corruption attack manipulating the evidence field as follows (where \( r_n = H[M[addr(C)]] = e_n/r_B \)):

\[
B \rightarrow \{C,W\} : K_B(r_B, addr(A)), M'
\]

This attack enables node \( B \) to bypass the first equality check of message checksum in equation (1). However, the evidence is still inconsistent, because \( r_n \neq e_n/r_x \) if node \( x \) is not compromised. Node \( A \) decides that node \( B \) is compromised based on Lemma 2.

C. Badmouthing

Thus far, we discussed attacks launched by node \( B \). Let us now consider the case when node \( C \) or a witness node is compromised but node \( B \) is uncompromised. In particular, we consider the scenario where node \( C \) or a witness node includes false evidence attributes (e.g., a random message checksum or a false timestamp) or transmits no evidence despite overhearing node \( B \)’s correct forwarding. This may cause node \( A \) to decide that the uncompromised node \( B \) is compromised, which produces a false positive. We refer to this attack as badmouthing.

In data-path-based detection, node \( A \) cannot distinguish a bypassing attack by compromised node \( B \) from a badmouthing attack by compromised node \( C \). However, as long as at least one uncompromised witness node exists, this node can produce consistent evidence of node \( B \)’s correct forwarding and allow node \( A \) to distinguish that node \( C \) is compromised.

Identifying a compromised node using consistent evidence also implies that the detection accuracy of the witness-based detection scheme is unaffected by multiple compromised nodes that badmouthing node \( B \), as long as there exists at least one uncompromised node \( x \). Our scheme results in a false positive only if every node in the neighborhood of node \( B \) badmouths node \( B \).

V. Performance Evaluation

In this section, we compare the detection accuracy of the data-path-based and the witness-based detection schemes in the presence of lossy links without collusion among misbehaving nodes. We also quantify the communication overhead of uncompromised nodes in the witness-based detection scheme and study the efficacy of the feedback suppression mechanism. The analysis of collusion is found in [9].

A. Metrics of detection accuracy

The proposed model analyzes the detection accuracy as a function of the following parameters:

- \( p_{loss} \) is the probability that a node fails to receive a packet from its one-hop neighbor or overhear a neighboring node’s transmission. For simplicity, we assume that both data transmission and overhearing experience equivalent interference.
- For simplicity, we restrict our attention to a witness’s Data-based evidence and a witness directly reaching node \( A \). \( \Lambda \) is the expected number of witness nodes located in the intersection area between node \( A \)’s and node \( B \)’s transmission ranges, where the number of witness nodes follows a 2D-Poisson distribution with the density parameter \( \lambda \).
- \( p_c \) is the probability that a node is compromised. A compromised node launches various attacks described in section IV except drop or evidence drop. Thereby, the event of whether a node is compromised and the event of packet loss are mutually exclusive.
- \( N \) is the maximum number of Data or ED packet transmissions.

We use false positive probability (FPP) and false negative probability (FNP) as detection-accuracy metrics. We only observe false positives, since fake consistent evidence that may generate false negatives can be only created through collusion, as explained earlier (i.e., FNP is equal to 0 in the non-collusion case). We break down false positives into two disjoint events as follows:

1. Node \( A \) receives no evidence, given that node \( B \) is not compromised.
2. Node \( A \) only receives inconsistent evidence, given that node \( B \) is not compromised.

In [9], we provide analytic expressions that quantify each of the FPPs and FNPs for both data-path-based and witness-based detection. Here, we present results.

B. Numerical evaluation of detection accuracy

We now observe how the FPP and FNP vary with the expected number of witness nodes and packet loss probability. Note that the case of data-path-based detection corresponds to the case of \( \Lambda = 0 \). In our results below, we assume that a packet can be re-transmitted up to three times \( (N = 3) \); once this maximum number of retransmission is reached, the packet is considered lost (dropped) by the sender and is not received at the receiver.

Fig. 1 plots the FPP as a function of the expected number of witness nodes \( (\Lambda) \) for \( p_{loss} = 0.1, 0.5 \), and \( p_c = 0.1 \) (Fig. 1a) and \( p_c = 0.5 \) (Fig. 1b). As expected, the FPP decreases (improving detection accuracy) as \( \Lambda \) linearly increases, since an increased number of witness nodes improves success probability of overhearing and reliability of a path to node \( A \), thus provides increased evidence to the judge node. The FPP also decreases as \( p_{loss} \) decreases, for the same reason. Fig. 2 breaks down the causes of the false positives for the case of...
In this paper, we presented witness-based detection, a new scheme that verifies correct forwarding along data paths in wireless networks. Unlike existing schemes, the witness-based detection scheme detects misbehaving nodes without incurring significant delays, even in the presence of multiple colluding nodes. Under the assumption of reliable communication between nodes, we formally showed that the witness-based detection scheme does not produce false positives or false negatives, as long as there is at least one uncompromised witness node. Using an analytical model, we showed that witness-based detection can support low false positive and false negative rates even in the presence of lossy wireless links, without incurring a significant communication overhead.

VI. CONCLUSION

ACKNOWLEDGMENT

This material is based upon work under a grant from the National Science Foundation, CNS-051998, and subcontract #069153 issued by BAE Systems National Security Solutions, Inc. and supported by the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego under Contract No. N66001-08-C-2013.

REFERENCES