Intrusion Detection System to Detect Wormhole using Fault Localization Techniques [1]

Maitreya Natu
Dept. of Computer and Information Science
University of Delaware
Newark, DE, USA, 19716
Email: natu@cis.udel.edu

Adarshpal S. Sethi
Dept. of Computer and Information Science
University of Delaware
Newark, DE, USA, 19716
Email: sethi@cis.udel.edu

***Abstract*— In this paper, we present a strategy to detect an intrusion using fault localization tools. We propose an intrusion detection system to detect a self-contained in-band wormhole attack using a combination of active probing and passive monitoring tools. We exploit anomaly in the end-to-end delay and per-hop delay patterns to identify the nodes involved in a wormhole attack. We present an architecture and an algorithm for wormhole detection and support the proposed approach with simulation results.**

***Index Terms*— Intrusion detection, Wormhole, Probing, Fault localization.**

I. Introduction

Intrusion detection is beginning to assume enormous importance in today's computing environment. A large array of intrusions occur via the network by exploiting the network vulnerabilities and spreading in a stealthy manner. In the past, various network-based approaches have been proposed for building intrusion detection systems [6] [5]. In such network-based intrusion detection systems, the sensors are located at choke points in the network or at the network borders. The sensors capture and analyze the network traffic to detect intrusions. Analysis is done by identifying malicious traffic content, malicious traffic pattern etc. These systems perform analysis at various levels of granularity like networks, protocols, applications, hosts etc.

Many intrusions hold a close resemblance to faults in their manifestation [11]. For instance, a symptom such as increased end-to-end delay could be caused by a failure like loss of some link connectivity, poor performing routers etc. The same symptom can also be observed due to a denial of service attack or malicious routing. Similarly, increased loss rate on a path could be the result of either node failure or an intrusion. The similarity in the manifestation of faults and intrusions provides motivation to integrate the intrusion detection and fault localization mechanisms. Moreover, an integrated approach will minimize the network overhead and redundancy involved in performing the two tasks independently.

[1]Prepared through collaborative participation in the Communications and Networks Consortiumsponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

In this paper, we propose to use fault localization tools to perform intrusion detection. We had made a case for integrated intrusion detection and fault localization in [11]; Here we focus on the specific intrusion detection task, which is identifying in-band wormhole attacks in mobile ad-hoc networks [1], [3]. An in-band wormhole is an attack in which colluding nodes create an illusion that two remote regions of a MANET are directly connected through nodes that appear to be neighbors, but are actually distant from each other [7]. The illusory shortcut is created by connecting the purported neighbors using a covert tunnel through other unsuspecting nodes. The wormhole undermines shortest path routing calculations, allowing the attacking nodes to attract traffic from other parts of the network such that it is routed through them and can be subsequently controlled, e.g., to delay, damage, discard, or misroute packets. In this paper, we propose to perform detection of a wormhole using probing, passive monitoring, and event correlation.

II. In-band wormhole attack

Wormhole refers to an attack on MANET routing protocols in which colluding nodes create an illusion that two remote regions of a MANET are directly connected through nodes that appear to be neighbors but are actually distant from one another. This shortcut is created by connecting the purported neighbors through a covert communication channel. A wormhole thus allows an attacker to create two attacker-controlled choke points to which traffic is attracted and which can be utilized by the attacker to degrade or analyze traffic at a desired time. The covert communication channel used by the attackers could be a separate communication mechanism not generally used by the network, forming an out-of-band wormhole attack. On the other hand, an in-band wormhole attack uses the primary link layer to develop the covert tunnel.

In this paper we focus on the self-contained in-band wormhole attack on MANETs. MANET routing protocols such as OLSR have been shown to be vulnerable to wormhole attacks [6]. [1] explains the wormhole attack in detail and also presents variations of the attack. Figure 1 shows a scenario of a wormhole attack where three attacker nodes, nodes 2, 5, and 11, develop an in-band wormhole tunnel to attract network traffic. The two end nodes, nodes 2 and 11, create an illusion

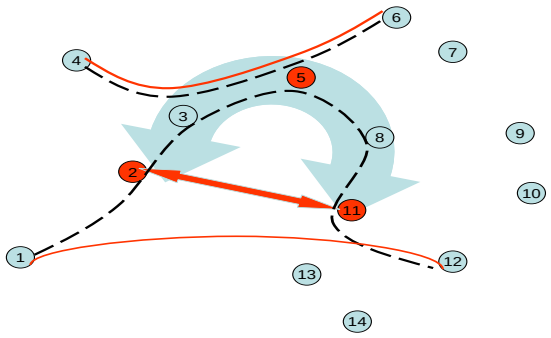

Fig. 1. Network topology with wormhole created by nodes 2, 5, and 11. Figure shows actual paths with dotted lines and advertised paths with solid lines. Actual path between nodes 1 and 12 is different from advertised path. Actual path and the advertised path between nodes 4 and 6 stays the same but overlaps with the wormhole tunnel.

that they are neighbors, thus attracting traffic from other parts of the network. The traffic is then forwarded through a tunnel with the help of the third attacker node, node 5. The illusion is created by making the two tunnel end nodes falsely advertise a 1-hop symmetric link between them without exchanging HELLO messages. The false link information is propagated to other nodes across the network via a broadcast of TC messages. Traffic is attracted towards the end-points due to the advertised shorter path and is forwarded by the attackers through the tunnel.

In the past, work has been done in detecting out-of-band wormhole attacks. Hu et. al. [6] described the out-of-band wormhole concept and presented several countermeasures to detect remote forwarding of packets. Hafslund et. al. [4] and Hong et. al. [5] defined security extensions to OLSR to prevent generation of false OLSR messages or replay of legitimate OLSR messages. Lazos et. al. [8] proposed a geography-based countermeasure to defend the wormhole attack. Some of these methods propose to defend by temporally or geograhically limiting the spread of HELLO messages. However, a self-contained in-band wormhole attack does not require exchange of HELLO messages and can defeat such defenses. Also some approaches rely on using source authentication using signing keys. Such defenses can be defeated if a node is compromised and the attacker has access to secured information.

Baras et. al. [1] proposed a mechanism to detect a self-contained in-band wormhole by observing the anomaly between the cumulative path loss and delay and the perceived path length. Cardenas et. al. [3] proposed an approach based on the space-time framework to detect the changed hop count distribution caused by a wormhole. These approaches detect the wormhole endpoints. They cannot identify the intermediate tunnel nodes that knowingly or unknowingly participate in forming a wormhole tunnel. Kruus et. al. [7] propose several countermeasures for wormhole detection and prevention. These countermeasures include 1) OLSR protocol extensions that incorporate link quality in routing decisions, 2) higher protocol layer measurements of packet loss and round-trip delays over 1-hop and 3-hop routing paths, and 3) monitoring for the presence of asymmetric links and other potential indicators.

We propose to use the fast real-time fault localization algorithms together with active and passive techniques for network monitoring to detect a wormhole attack. We exploit the anomaly between the end-to-end delays and the sum of hop queuing delays. We use probing to compute end-to-end delays and passive monitoring to obtain the hop queuing delays. We present the anomalies that can be observed at various places in the wormhole affected network and report the observed anomalies as symptoms for the symptom-fault correlation. We then use event correlation algorithms to localize the wormhole end-points and the intermediate tunnel nodes.

III. Fault Localization Architecture

Fault localization refers to the task of determining the location of a failure in the network. This task becomes challenging because of a variety of reasons. A single fault can cause numerous alarms in a variety of domains. This situation is compounded when there are multiple concurrent failures or when the failures are transient. Moreover, the presence of incomplete and inaccurate information about the underlying network further aggravates the problem.

In the past, various approaches have been proposed to perform fault localization. The proposed approaches vary in various aspects like the network models, involved complexities etc. The approaches proposed can be broadly classified into active and passive approaches. Passive approaches do not insert additional traffic in the network but instead perform analysis based on the anomalies observed at the passive network monitors. Active approaches on the other hand send out probe traffic over the network to collect end-to-end statistics and deduce the network health. Passive approaches hold the advantage of not adding additional traffic, while active approaches provide an additional power of probing any area of the network as required. Active approaches also provide the capability of measuring various end-to-end monitoring statistics. We propose to use a combination of active and passive monitoring approaches for detecting a wormhole in a network. We present details of the proposed approach in Section IV.

An important task in fault localization is to correlate the various symptoms reported either by active or passive monitors. Recently a promising new algorithm has been developed called Incremental Hypothesis Updating (IHU) [10]. The algorithm uses a probabilistic dependency model that represents the causal relationship between the faults and the symptoms. When a symptom is received, a set of hypotheses is constructed using the dependency model. The hypotheses set is incrementally updated with each received symptom. The IHU algorithm has been shown to be fast, scalable, and accurate, with the potential to be deployable in real-time. The traditional fault localization techniques do not take the dynamics of a MANET into consideration. Mobility may cause a change in dependencies which if not taken into account can lead to incorrect diagnosis. In [9], we have extended the

IHU algorithm to adapt it to the changing dependencies in a dynamic environment of a MANET. We use a dynamic dependency model to track the changing dependencies and use a temporal correlation algorithm to perform localization in the presence of such a dynamic dependency model.

IV. Proposed approach for wormhole detection

In this section we propose our approach to detect a wormhole intrusion using probing and event correlation. Certain paths get affected by the presence of a wormhole attack. The advertised properties for such paths are different from the actual properties. We aim to exploit these properties to detect a wormhole attack.

We focus on two types of anomalies that are caused by a wormhole.

Incompatible hop queuing delays and end-to-end delay: The paths that are attracted by a wormhole have different advertised and actual routes. The advertised routes in this case are much shorter than the actual routes which go through the wormhole tunnel. For instance, consider the path between nodes 1 and 12 in Figure 1. The advertised route for this path goes through nodes 2 and 11, while the actual route taken by packets between nodes 1 and 12 goes through nodes 2, 3, 5, 8, and 11. A large part of the end-to-end delay for a path consists of queuing delays at each hop. Thus, with the available advertised information, the end-to-end delay for such a path will not be explained by the sum of queuing delays of the hops present on its advertised path. We exploit this observation to detect an anomaly.

Increased end-to-end delay: Another anomaly can be observed on the nodes that form the wormhole tunnel. The traffic received by these nodes is not explained by the overall end-to-end traffic. Given the advertised routes and the amount of end-to-end traffic, the amount of traffic received by these nodes should be significantly less than what the nodes actually receive. The additional unexplained traffic is due to the wormhole tunnel traffic. Thus due to the wormhole, the queuing delay of tunnel nodes would increase. This in turn would increase the end-to-end delay of the routes that do not get attracted by the wormhole but pass through some of the tunnel nodes. For instance, in Figure 1, the path between nodes 4 and 6 does not get attracted by the wormhole but actually goes through nodes 3 and 5 that are part of the wormhole tunnel. Nodes 3 and 5 would have increased queuing delay due to the wormhole traffic, leading to an increased end-to-end delay on the path between nodes 4 and 6. Thus, unlike the previous anomaly, paths belonging to this anomaly show a consistent end-to-end delay and hop queuing delay sum. However, they show an abrupt increase in the end-to-end delay and the hop queuing delay values that are not explained by the traffic supposedly flowing through these nodes.

We use the above explained anomalies to propose an algorithm for detecting endpoint and tunnel nodes of a wormhole. We send out probes and compute an average end-to-end queuing delay for various paths. As the probes incur additional network traffic, the end-to-end paths to be probed should be carefully selected such that the desired network coverage is obtained while causing minimum management traffic. Various approaches can be used to select end-to-end paths to probe. Some work has been done in the past on a systematic selection of probes to cover the area of interest in the network [2]. We propose to send out probes from each node to its three-hop neighbors. Such a probe set can allow us to analyze various paths having different path properties and cover the anomalous paths explained above in this section while maintaining probes within a limited hop distance from a node. As a part of ongoing research, we are looking at other approaches to select probes to meet the requirements of traffic optimization and network coverage. These probes will provide the end-to-end delays for various paths in the network. Each node computes an average of the end-to-end delay and standard deviation for its $3-$hop paths over a sliding window of time. Nodes report the average end-to-end path delay values to the central manager. Nodes also report an alarm for a specific end-to-end path on observing an unexpected increase in the end-to-end delay, for instance, when the increase in the end-to-end delay is greater than three times the standard deviation. Note that after the wormhole attack, many new three hop paths emerge. These paths could form the most relevant symptoms for diagnosis of the wormhole. However these paths have no past history to compare the end-to-end delays with. Hence nodes also maintain a generic 3-hop average path end-to-end delay. This value provides a benchmark for comparing the newly formed 3-hop paths to detect an abnormal increase in the end-to-end delay.

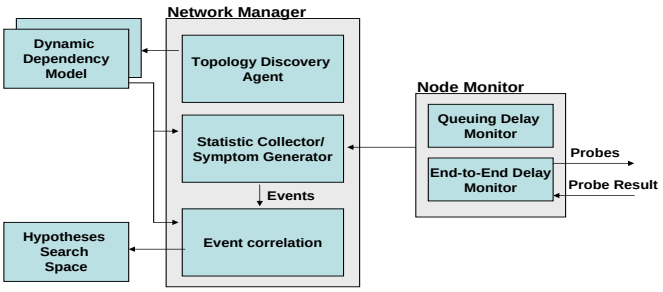

Fig. 2. System architecture representing various modules that cooperate to perform intrusion detection

We also need to monitor the hop queuing delays to check for anomalies explained above. We deploy a passive monitoring tool at each node to maintain an average queuing delay. The queuing delay is measured as the amount of time elapsed from the arrival of a packet till its forwarding to the appropriate next hop. Each node maintains an average queuing delay over a sliding window of time.

A. Architecture

Figure 2 presents the architecture for the proposed approach. As explained above, each node computes the average queuing delay over a sliding window of time. Each node sends out probes to its $3-$hop neighbors and maintains an average end-to-end delay and standard deviation for these $3-$hop paths. Each node periodically reports the end-to-end delay statistics

and queuing delay to the manager and reports an alarm for a particular probe path on observing an unexpected increase in the end-to-end delay.

The manager node performs a periodic topology discovery using the topology discovery agents and maintains a dynamic dependency model. The dynamic dependency model contains the dependencies between nodes and end-to-end paths at different times. Dependencies are based on the advertised paths computed from the OLSR routing table available at the manager node. On receiving the end-to-end delay value for a source-destination pair, the manager performs an analysis to observe the compatibility between the end-to-end delay value and the sum of the hop queuing delays of the nodes on the advertised path between that source-destination pair. Based on this analysis, the manager identifies the end-to-end paths that can be used as symptoms to detect the wormhole tunnel end-points and the intermediate nodes. If an end-to-end path is reported to show a significant increase in the end-to-end delay but shows consistency with the sum of queuing delays on the nodes on its advertised end-to-end path, the path then is used as a symptom for identifying the intermediate nodes on a wormhole tunnel. An end-to-end path is reported as a symptom for identifying wormhole tunnel endpoints, if the end-to-end delay value is not consistent with the sum of hop queuing delays on the nodes on its advertised path. We then perform two separate diagnosis one for end-point diagnosis and another for intermediate nodes diagnosis. We use the IHU and the temporal correlation algorithm presented in [10] and [9] respectively to correlate the symptoms and identify the wormhole endpoints and the intermediate nodes. We present the operations performed at each node and the manager in Algorithm NodeMonitor and Algorithm NetworkManager respectively.

The event correlation algorithm reports a relatively small set of nodes as likely causes of failure. The set of nodes reported by the algorithm includes the attacker nodes. However, the set might contain some more nodes other than the actual attacker nodes. Further analysis might be needed on the inferred set of suspected nodes to identify the exact attackers. Further analysis to refine the results can be done using various properties that a wormhole affected network might show. We list some such observations here:

Traffic anomaly: Traffic received at the tunnel nodes will not be explained by the end-to-end traffic being sent through these tunnel nodes. If an analysis is done of the end-to-end traffic and the traffic received per node, the traffic routed by the wormhole tunnel would show up in the per-node traffic of the tunnel nodes; however, no end-to-end traffic would explain this additional traffic. Attack nodes can defeat this observation by incorrectly reporting end-to-end traffic between the attacker nodes to explain the additional traffic received by the tunnel nodes. However this incorrectly reported end-to-end traffic would have to be large enough to represent the sum of all the traffic routed through the tunnel. This high end-to-end traffic between the attack nodes, coupled with other observations can itself act as a symptom for further analysis.

NodeMonitor Monitoring at Node n {The algorithm is event based, where the algorithm waits for the events and responds as programmed}

input : 3−hop neighbor nodes
output: symptom triggers, path average end-to-end delays
Define $T_h(n)$: Average Hop queuing delay at node n;
Define $t^{'}_{avg}(p)$: Current average delay for path p;
Define $t^{'}_{avg}(3hop)$: Current average delay for 3-hop path;
Define $d(p)$: Standard deviation of delay on path p;
Define $d(3hop)$: Standard deviation of delay on 3-hop paths;
Define $t_{avg}(p)$: Average delay on path p;
Define $t_{avg}(3hop)$: Average delay on 3−hop paths;
Define $pingtimer$: Timer to send pings;
Define $updatetimer$: Timer to update the manager with statistics;
Initialize $t_{avg}(3hop)$, d(3hop), $t_{avg}(p)$, d(p) from the sample values collected during the learning period;
if *a packet received for forwarding* **then**
    Update $T_h(n)$ with queuing delay for each forwarded packet;
end
if $pingtimer$ *expires* **then**
    foreach *3−hop path p* **do**
        Send ping and compute $t_e(p)$;
    end
    foreach *3−hop path p* **do**
        if *($t_e(p) > (t_{avg}(p)+3*d(p))$ or ($t_e(p) > (t_{avg}(3hop)+3*d(3hop))$)* **then**
            Report p to the manager node;
        end
        Update $t^{'}_{avg}(p)$ and $t^{'}_{avg}(3hop)$ with $t_e(p)$;
    end
end
if *update timer expires* **then**
    Update d(p), $t_{avg}(p)$ with the values $t^{'}_{avg}(p)$;
    Update d(3hop), $t_{avg}(3hop)$ with the values $t^{'}_{avg}(3hop)$;
    Report $T_h(n)$ and $t^{'}_{avg}(p)$ for all 3−hop paths p to the manager node;
end

Increased 3−hop neighbors: The creation of a wormhole would increase the 3−hop neighbors of all nodes around the tunnel endpoints. For instance, node 1 in Figure 1 would find nodes 8, 12, and 13 as its 3−hop neighbors after creation of the wormhole.

Decreased path lengths: The creation of wormhole would decrease the length of certain paths. For instance, before the launch of a wormhole, path from node 1 to 14 goes through nodes 2, 3, 5, 8, 11, and 13. However after the wormhole creation, advertised hops for the same path are reduced to nodes 2, 11, and 13.

Increased loss rate: Nodes will observe an increased loss rate on the wormhole affected paths due to an increased number of hops and an increased amount of traffic on the wormhole tunnel nodes.

V. Simulation results

In this section, we present simulation results to show the correctness of the observations discussed in this paper. We simulated a wormhole attack and implemented the proposed algorithm to detect the wormhole nodes. The algorithm successfully detects the wormhole attackers. In this section, we show how the anomalies discussed in this paper are manifested in a wormhole attack scenario. These anomalies provide the symptoms for the correlation algorithm.

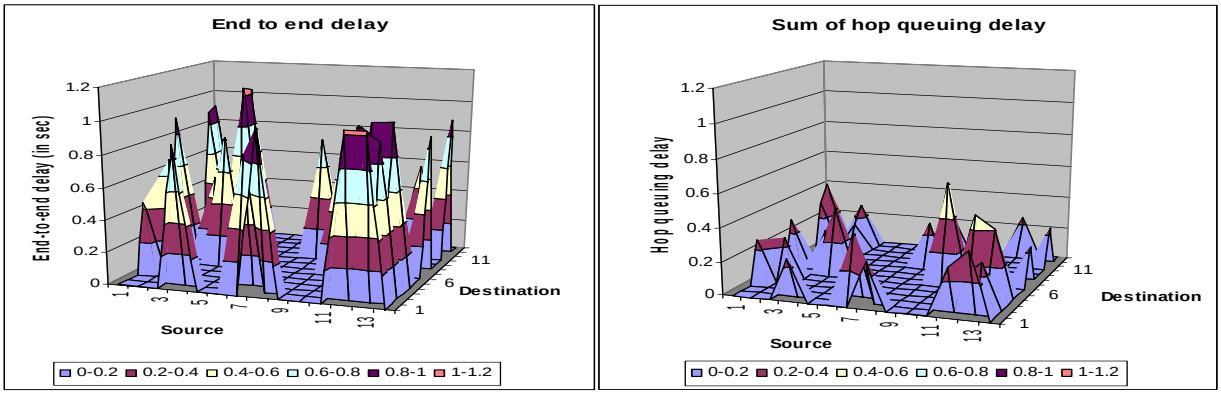

Fig. 3. The end-to-end delay and sum of queuing delay of the hops on various end-to-end paths

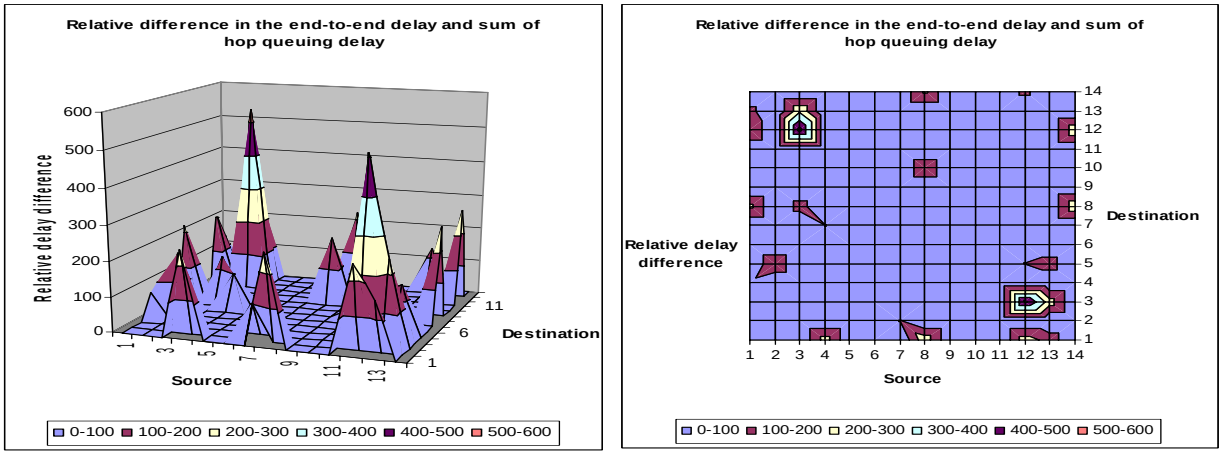

Fig. 4. Two different views of the relative difference in the end-to-end delay and sum of the queuing delay of the hops on various end-to-end paths. The relative difference is computed in this graph as ((end-to-end delay - sum of hop queuing delay)/end-to-end delay)*100.

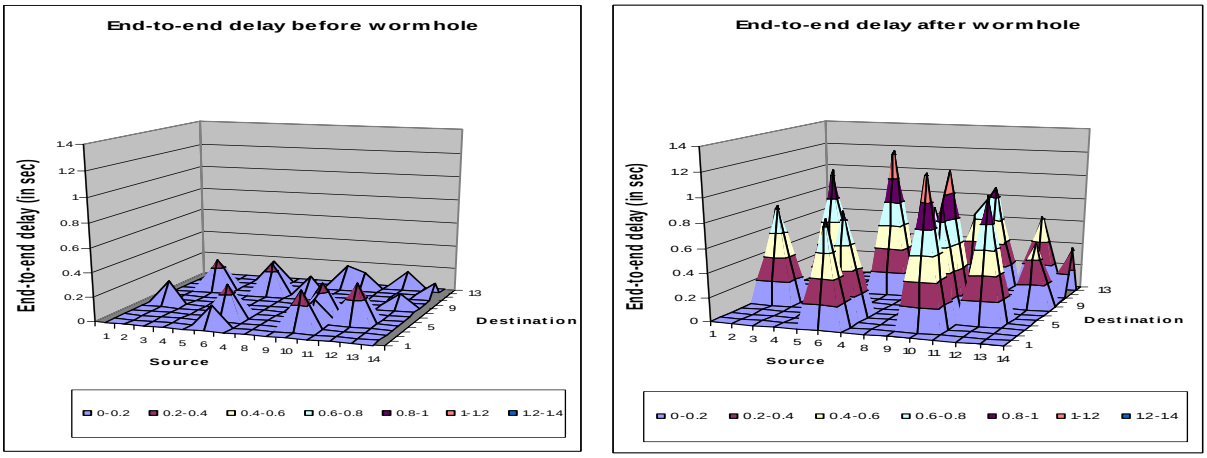

Fig. 5. End-to-end delay on various paths before and after the launch of wormhole attack.

NetworkManager Analysis at the manager node {The algorithm is event based, where the algorithm waits for the events and responds as programmed}

input : Nodes
output: Intrusion location
Define $T_h(n)$: Queuing delay for node n;
Define S_e: Symptoms for endpoint detection;
Define S_t: Symptoms for tunnel nodes detection;
Define Threshold : Minimum unexplained per-hop delay to trigger a symptom;
Define MinSymptomSize : Minimum number of symptoms to trigger event correlation;
Define Report { $T_h(n)$, $\forall_{p \in 3-hoppaths} t^{'}_{avg}(p)$ } : Report received from a node n;
if *a report r received from node n* **then**
    Update $T_h(n)$;
    foreach *path p* $\in$ *r* **do**
        UnexplainedPerHopDelay $\leftarrow$ ($t^{'}_{avg}(p)$ - ($\sum_{m \in PathNodes(p)} T_h(n)$))/|PathNodes(p)|;
        if *UnexplainedPerHopDelay* $>$ *Threshold* **then**
            Add p to S_e;
        end
    end
end
if *a symptom trigger for path p is received from a node* **then**
    Add p to S_t;
end
if $|S_e| >$ *MinSymptomSize* **then**
    Run event correlation algorithm on S_e to detect the wormhole tunnel end points;
end
if $|S_t| >$ *MinSymptomSize* **then**
    Run event correlation algorithm on S_t to detect the wormhole tunnel nodes;
end

A. Simulation Model

We performed the simulation using Qualnet. We implemented the topology presented in Figure 1 and launched a wormhole as shown in the same figure. We used OLSR routing protocol for this setup. The presented simulations do not consider node mobility at this time; however defending wormhole attacks in the presence of node mobility is part of our future research plans. Each node periodically discovers its 3−hop neighbors from OLSR tables and sends a train of pings to these 3−hop nodes to collect end−to−end delay values. Each node also monitors the average queuing delay in forwarding packets. As explained in Section IV, each node reports the average queuing delay and average end-to-end delay for each of its 3−hop paths to the central manager. Each node also reports an abnormal increase in the end-to-end delay on a path as a symptom for wormhole tunnel node detection.

A central manager performs a periodic topology discovery using OLSR tables, and builds a probabilistic dependency model representing the path−node dependencies. The manager uses this information to find anomalies between the path delay and the sum of hop queuing delays to identify symptoms for wormhole end-point detection. The manager collects the symptoms reported by the nodes. These symptoms report end-to-end paths that show an abrupt increase in the end-to-end delay. The manager then performs event correlation on these symptoms to identify possible wormhole attacker nodes.

B. Simulation observations

We observe that certain paths show a significant anomaly between the end-to-end path delay and the sum of hop queuing delays when a wormhole attack is launched. These are the paths that get affected by the wormhole attack. Their actual hop count is significantly larger than the hop count perceived by the source nodes. Figure 3 shows the difference in the end-to-end delay and the sum of hop queuing delays on such 3−hop paths. Figure 4 presents the relative difference in the end-to-end delays and the sum of hop queuing delays for these end-to-end paths. As discussed in Section IV, relative difference in the value of end-to-end delays and the sum of hop queuing delays forms a basis to trigger symptoms for event correlation to detect the wormhole end-points. The event correlation process then localizes a set of nodes as the possible end-points of the wormhole tunnel.

Figure 5 shows how the end-to-end delay values change for cerain end-to-end paths after the wormhole attack is launched. Figure shows end-to-end delay values for some end-to-end paths before and after the wormhole. We can see that the delay values significantly increase after the wormhole launch. As explained in Section IV, these end-to-end paths are used as symptoms to detect the intermediate tunnel nodes.

We used Incremental Hypothesis Updating (IHU) [10] algorithm to correlate the symptoms to detect the end-points and the tunnel nodes. The event correlation algorithm correlated these symptoms and was able to successfully detect the wormhole. Figure 6 presents the event-correlation of the reported symptoms to infer the wormhole end-points and tunnel nodes. Figure shows the symptoms collected for the end-point node detection. These symptoms consist of end-to-end paths that show an inconsistency in the end-to-end delay and the sum of hop queuing delay on that path. Figure shows that the end-to-end delay and the sum of hop queuing delays for these reported symptoms differ significantly. IHU algorithm correlates theses symptoms to infer the root cause. As shown in the figure, the algorithm declares a set of nodes as possible cause of failure. The algorithm is able to successfully identify the actual wormhole end-point nodes (Node 2 and Node 11) in the reported set of nodes.

Figure 6 also shows the analysis done for detection of the wormhole tunnel nodes. Figure shows the symptoms collected to detect tunnel nodes. These symptoms consist of end-to-end paths that do not get attracted by the wormhole, but their paths overlap with the wormhole tunnel. These paths show a significant increase in the end-to-end delay after the launch of wormhole due to increased queuing delay at the tunnel nodes. Figure shows the hypotheses computed by the event correlation algorithm to infer the possible tunnel nodes. The algorithm successfully reports the actual tunnel nodes (Node 3, Node 5, and Node 8) in its hypotheses.

The hypotheses reported by the algorithms are ranked based on the belief values. These belief values are computed by the algorithm for each hypothesis set and represent the algorithms

End-point detection (actual end-point nodes: node-v2, node-v11)

End-point detection symptoms						Algorithm hypotheses *(Suspected end-point nodes)*
Symptom	End-to-end delay	Sum of hop queuing delay	Symptom	End-to-end delay	Sum of hop queuing delay	node-V11 node-V1 | Rank 0 node-V11 node-V2 | Rank 1 node-V11 node-V4 | Rank 2 *{ node-V11, node-V1, node-V2, node-V4}*
path-V2->V8	0.608997	0.046722	path-V12->V3	1.041048	0.118377	
path-V3->V11	0.613258	0.046707	path-V1->V12	1.126465	0.198812	
path-V8->V2	0.408844	0.046722	path-V3->V12	1.158543	0.118377	
path-V11->V3	0.673510	0.046707	path-V1->V8	1.163029	0.142576	
path-V11->V7	0.728440	0.282520	path-V3->V13	1.318174	0.150773	
path-V14->V8	0.706576	0.147122	path-V13->V3	0.965026	0.150773	
path-V4->V1	0.776509	0.302232	path-V1->V13	1.022991	0.231208	
			path-V13->V9	0.640050	0.199095	

Tunnel nodes detection (actual tunnel nodes: node-V3, node-V5, node-v8)

Tunnel node detection symptoms				Algorithm hypotheses *(Suspected tunnel nodes)*
path-V8->V14	path-V5->V12	path-V7->V11	path-V12->V14	node-V13 node-V5 | Rank 0 node-V11 node-V5 | Rank 1 node-V8 node-V3 node-V12 | Rank 2 *{node-V13, node-V11, node-V5, node-V8 node-V3 node-V12}*
path-V9->V7	path-V3->V9	path-V4->V8	path-V8->V4	
path-V4->V6	path-V2->V7	path-V1->V5	path-V7->V10	
path-V9->V6	path-V9->V3	path-V10->V13	path-V5->V13	
path-V5->V10	path-V13->V10	path-V13->V5	path-V9->V13	
path-V12->V5	path-V7->V9	path-V11->V6	path-V14->V12	
path-V10->V7	path-V2->V6	path-V7->V4		

Fig. 6. The symptoms collected and hypotheses computed by the localization algorithm to detect the wormhole tunnel nodes and end points.

confidence in the hypothesis. The hypothesis with rank 0 has the highest confidence associated with it. In Figure 6 we present the top 3 hypotheses computed by the algorithm. The algorithm successfully reports the end-point and tunnel nodes in the hypotheses sets. Algorithm reports some additional nodes in the inferred hypotheses. This happens due to incorrect reporting of some symptoms. Thus additional analysis can be done on the reported set of nodes to identify the wormhole end-points and the tunnel nodes.

VI. Conclusion

We presented an approach to use fault localization techniques to perform intrusion detection. We proposed an intrusion detection system to detect an in-band self-contained wormhole attack using probing, passive monitoring, and event correlation. We proposed an architecture and an algorithm for wormhole detection using the anomalies in the path end-to-end delay and sum of queuing delays at the hops on the advertised path. We simulated the proposed approach using Qualnet and presented the simulation observations and localization results.

As part of future research, we aim to find other ways to send probes to monitor the network in a more efficient and effective manner. We aim to perform experiments with a variety of wormhole scenarios to see the effectiveness of probing and fault localization algorithms in different cases. We also aim to develop algorithms to detect intermediate tunnel nodes of a wormhole while considering legitimate traffic variations. We aim to distinguish the cases when legitimate traffic variations develop similar patterns as manifested by a wormhole.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or the U.S. Government.

References

[1] J. S. Baras, A. A. Cardenas, and V. Ramezani. On-line detection of distributed attacks from space-time network flow patterns. In *ASC'04, the 24th Army Science Conference, Orlando, FL*, Nov. 2004.

[2] M. Brodie, I. Rish, and S. Ma. Optimizing probe selection for fault localization. In *Distributed Systems Operations Management*, pages 1147–1157, 2001.

[3] A. A. Cardenas, J. S. Baras, and V. Ramezani. Distributed change detection for worms, DDoS and other network attacks. In *ASC'03, the 23rd Army Science Conference*, 2003.

[4] A. Hafslund, A. Tonnesen, R. Bjorgum Rotvik, J. Anderson, and O. Kure. Secure extensions to the OLSR protocol. In *OLSR Interop Workshop, San Diego*, Aug. 2004.

[5] F. Hong, L. Hong, and C. Fu. Secure OLSR. In *19th International Conference on Advanced Information Networking and Applications (AINA'05)*, volume 1, pages 713–718, 2005.

[6] Y. Hu, A. Perrig, and D. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *IEEE Infocomm*, 2003.

[7] P. Kruus, D. Sterne, R. Gopaul, M. Heyman, B. Rivera, P. Budulas, B. Luu, T. Johnson, and N. Ivanic. In-band wormholes and countermeasures in OLSR networks. In *SecureComm2006, Baltimore, MD*, Aug. 2006.

[8] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L. W. Chang. Preventing wormhole attacks on wireless ad hoc networks: A graph theoretic approach. In *IEEE Wireless Communications and Networking Conference (WCNC)*, 2005.

[9] M. Natu and A. S. Sethi. Adaptive fault localization in mobile ad-hoc battlefield networks. In *MILCOM'05, Atlantic City, NJ*, 2005.

[10] M. Steinder and A. S. Sethi. Probabilistic fault diagnosis in communication systems through incremental hypothesis updating. *Computer Networks*, 45(4):537–562, July 2004.

[11] D. Sterne, S. Tsang, M. Natu, D. Balenson, P. Mouchtaris, and A. S. Sethi. Integrating intrusion detection and fault localization in MANETs. In *Milcom-2006, IEEE Military Communications Conference, Washington, DC*, Oct. 2006.