Detecting Critical Nodes for MANET Intrusion Detection Systems

A. Karygiannis, E. Antonakakis, and A. Apostolopoulos
National Institute of Standards and Technology
{karygiannis, manos, aimilios}@nist.gov

Abstract

Ad hoc routing protocols have been designed to efficiently reroute traffic when confronted with network congestion, faulty nodes, and dynamically changing topologies. The common design goal of reactive, proactive, and hybrid ad hoc routing protocols is to faithfully route packets from a source node to a destination node while maintaining a satisfactory level of service in a resource-constrained environment. Detecting malicious nodes in an open ad hoc network in which participating nodes have no previous security associations presents a number of challenges not faced by traditional wired networks. Traffic monitoring in wired networks is usually performed at switches, routers and gateways, but an ad hoc network does not have these types of network elements where the Intrusion Detection System (IDS) can collect and analyze audit data for the entire network. A number of neighbor-monitoring, trust-building, and cluster-based voting schemes have been proposed in the research to enable the detection and reporting of malicious activity in ad hoc networks. The resources consumed by ad hoc network member nodes to monitor, detect, report, and diagnose malicious activity, however, may be greater than simply rerouting packets through a different available path. This paper presents a method for determining conditions under which critical nodes should be monitored, describes the details of a critical node test implementation, presents experimental results, and offers a new approach for conserving the limited resources of an ad hoc network IDS.

Keywords: mobile ad hoc network, MANET, intrusion detection, IDS, security, edge-cut, vertex-cut.

1. Introduction

Mobile ad hoc networks (MANETs) present a number of unique problems for Intrusion Detection Systems (IDS). Network traffic can be monitored on a wired network segment, but ad hoc nodes can only monitor network traffic within their observable radio transmission range. A wired network under a single administrative domain allows for discovery, repair, response, and forensics of suspicious nodes. A MANET is most likely not under a single administrative domain, making it difficult to perform any kind of centralized management or control. In an ad hoc network, malicious nodes may enter and leave the immediate radio transmission range at random intervals, may collude with other malicious nodes to disrupt network activity and avoid detection, or behave maliciously only intermittently, further complicating their detection. A node that sends out false routing information could be a compromised node, or merely a node that has a temporarily stale routing table due to volatile physical conditions. Packets may be dropped due to network congestion or because a malicious node is not faithfully executing a routing algorithm [1].

Researchers have proposed a number of collaborative IDS systems to address these challenges. In general, collaborative IDSs will perform best in a densely populated MANET with limited mobility, and will perform worse in a sparsely populated MANET with significant mobility. The effectiveness of a collaborative IDS also depends on the amount and trustworthiness of data that can be collected by each node. The longer the nodes are members of the MANET, the greater the availability of meaningful data for further analysis. In a MANET with a high degree of mobility, if the number of routing error messages caused by legitimate reasons far exceeds the number of routing error messages caused due to the presence of malicious nodes, the effectiveness or benefit of an IDS may be severely limited. The damage that could be caused by a malicious node in highly mobile environment would, however, also be minimal. Mobility introduces additional difficulty in setting up a system of cooperating nodes in an IDS.

Ad hoc networks comprised of stationary sensors and mobile collector nodes may be less ephemeral and less mobile, while ad hoc networks comprised of personal handheld devices may be characterized by sporadic participation of individual members. A node's movements cannot be restricted in order to let the IDS

cooperate or collect data and a node cannot be expected to monitor the same physical area for an extended period of time. A single node may be unable to obtain a large enough sample size of data to accurately diagnose other nodes.

MANETs with loose or no prior security associations are more difficult to diagnose than a MANETs comprised of nodes from the same organization with strong security associations and access to higher-level security services. Establishing trust in an open network in which higher-level security services are unavailable can be hampered by the short-lived presence of both collaborating and malicious nodes. In addition to having no previously established trust associations, nodes in an open ad hoc network have little incentive for reciprocity to faithfully execute a routing protocol or provide a minimum level of service. Closed ad hoc networks that support critical applications may not be able to tolerate the presence of malicious nodes; fortunately closed networks can more easily establish prior trust associations for a collaborative IDS. A closed ad hoc network is at a greater risk by allowing the extended presence of malicious nodes, but more likely to have preinstalled security mechanisms to detect these malicious nodes. Malicious nodes in sparsely populated networks can be more harmful than malicious nodes in a densely populated network since these nodes can effectively not only disrupt communication but also disconnect the network.

The level of effort required of resource constrained devices to monitor, detect, and diagnose malicious activity in a dynamic ad hoc network may be too costly when compared to the cost of simply rerouting packets through an alternative path. For example, in a densely populated network where several alternative paths are typically available, selecting an alternative route may be a more judicious use of limited resources. Alternatively, in some situations as we describe in this paper, the ad hoc network should expend additional resources to monitor critical nodes. In this paper we provide the motivation and implementation details for detecting critical nodes in an ad hoc network.

The paper is organized as follows: Section 1 provides an introduction to ad hoc IDS and the problem this paper addresses, Section 2 provides a brief overview of previous research, Section 3 describes our approach to detecting critical nodes, Section 4 provides implementation details and illustrative examples, Section 5 summarizes our experimental results, Section 6 outlines future research topics, Section 7 concludes the paper, and Section 8 includes relevant references.

2. Related Work

A number of IDS techniques have been proposed in the research literature. Moreover, a number of trust-building and cluster-based voting schemes have been proposed to enable the sharing and vetting of messages, and data, generated and gathered by IDS systems. Zhang and Lee describe a distributed and collaborative anomaly detection-based IDS for ad hoc networks [2, 3]. Tseng et al. describe an approach that involves the use of finite state machines for specifying correct AODV routing behavior and distributed network monitors for detecting run-time violation of the specifications [4]. Pirzada and McDonald present a method for building confidence measures of route trustworthiness without a central trust authority. The authors also present a concise summary of previous work in the area of establishing trust in ad hoc networks [5]. Theodorakopoulos and Baras present a method for establishing trust metrics and evaluating trust [6]. Michiardi and Molva assign a value to the "reputation" of a node and use this information to identify misbehaving nodes and cooperate only with nodes with trusted reputations [7]. Albers and Camp couple a trust-based mechanism with a mobile agent based intrusion detection system, but do not discuss the security implications or overhead needed to secure the network and individual nodes from the mobile agents themselves [8]. Sun, Wu and Pooch introduce a geographic zone-based intrusion detection framework that uses location-aware zone gateway nodes to collect and aggregate alerts from intrazone nodes. Gateway nodes in neighboring zones can then further collaborate to perform intrusion detection tasks in a wider area and to attempt to reduce false positive alarms [9].

3. Detecting Critical Nodes

The approach described in this paper is built around the notion of a critical node in an ad hoc network. Our definition of a critical node is a node whose failure or malicious behavior disconnects or significantly degrades the performance of the network. Once identified, a critical node can be the focus of more resource intensive monitoring or other diagnostic measures. If a node is not considered critical, this metric can be used to help decide if the application or the risk environment warrant the expenditure of the additional resources required to monitor, diagnose, and alert other nodes about the problem. In order to detect a critical node we look towards a graph theoretic approach to detect a vertex-cut and an edge-cut. A vertex-cut is a set of vertices whose removal produces a subgraph with more components than the original

graph. A cut-vertex, or articulation point, is a vertex-cut consisting of a single vertex. An edge-cut is a set of edges whose removal produces a subgraph with more components than the original graph. A cut-edge, or bridge, is an edge-cut consisting of a single edge. Although the cut-vertex or cut-edge of a graph G can be determined by applying a straight forward algorithm [14], finding a cut-vertex in the graphical representation of an ad hoc network is not as straight-forward, since the nodes cannot be assumed to be stationary. A network discovery algorithm can give an approximation of the network topology, but the value of such an approximation in performing any kind of network diagnosis or intrusion detection depends on the degree of mobility of the nodes.

Determining the global network topology in a mobile ad hoc network given the time delays of the diagnostic packets and the mobility of the nodes makes this task futile, but determining an approximation of this topology, or subset of this topology, within a certain time frame may be useful. An approximation of the network topology can still provide useful information about network density, network mobility, critical paths, and critical nodes. Even with the uncertainty associated with correctly reconstructing the network topology for a given time period, this additional information can help reduce the resources consumed to monitor all nodes in the absence of this information.

The critical node test detects nodes whose failure or malicious behavior disconnects or significantly degrades the performance of the network (i.e. introduces unacceptably long alternative paths). In an effort to further reduce the number of tests performed, a lightweight trigger mechanism monitors network traffic and initiates a critical node test when it suspects such a condition might exist. The trigger mechanism is designed to allow false positives that the critical node test will later screen out. The only false-negatives that can occur are when there is no traffic to analyze on a cut-edge, but this condition is most likely short-lived and of no consequence. The trigger mechanism monitors the number of connections served by the test node as well as the number of packets traversing the test node. Note that the trigger itself can also serve as a lightweight alternative to the critical node test. The node performing the test is referred to as the testing node, and the node being tested is referred to as the node under test.

The critical node test implementation makes extensive use of the ip, route, and ping utilities. The ip utility is a TCP/IP interface configuration and routing utility that configures the network interfaces. The route utility manipulates the kernel's IP routing tables. Its primary purpose is to set up static routes to specific hosts or networks via an interface after it has been configured with the ifconfig program. When used together, ip route provides the necessary tools for manipulating any of the routing tables – such as displaying routes, routing cache, adding routes, deleting routes, altering existing routes, getting route information, and clearing routing tables.

Three steps are required to detect whether a testing node shares a critical link with its neighbor. The first step is to temporarily modify the testing node's routing table to allow only one communication link to be operational at a time, while blocking communication through all others. The enabled communication link will be between the testing node and a node other than the node under test. Each communication link will be tested sequentially in this manner to determine if an alternative path to the link under test exists. If an alternative path exists, then the link is not critical because its removal will not disconnect the network. In order to temporarily change the routing tables, we route all the outgoing network traffic through the link shared with a neighbor node other than the node under test, and execute the following commands:

#ip route change <network_area>/24 via <neighbor_node>

The second step is for the host to attempt to discover an alternative path by using ping to the node under test without using the suspected cut-edge between the testing node and the node under test. To discover an alternative path to the node under test, the testing node executes the following command:

#ping –c 5 –s 10 <node_under_test> -A -R

Where –c is the number of pings that the host executes, -s is the number of data bytes to be sent, -A is the audit flag, and the –R flag returns the route, if exists, to the <node_under_test> node. Once the results of the ping are returned, the network routing table is restored during the third and final step to its initial configuration as follows:

#ip route del <network_area >/24 via <neighbor_node> dev wlan0
#ip route add <network_area>/24 dev wlan0

Once a critical link is detected, the host node may choose expend additional resources to initiate an IDS module that is more resource intensive, such as a traffic monitoring watchdog module or collaborative IDS [10, 11, 12]. If there is no critical link then the host can use the lighter weight modules to continue to monitor network traffic. More experimental data is needed to find the right balance between more and less resource intensive IDS techniques. The difficulty in

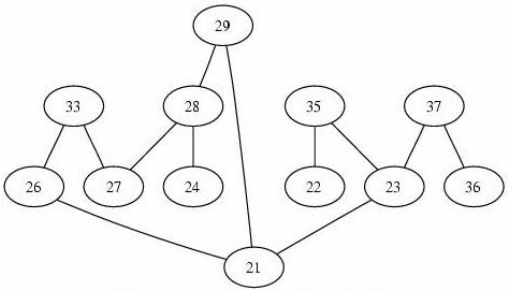
characterizing typical behavior in an ad hoc network is further complicated by the lack of publicly available MANET traffic traces.

Altering the routing tables to perform the critical node test should not disrupt normal traffic, perturb any exiting IDS, or alert neighboring nodes that they are being tested. The current implementation uses ip rules that cause some level of traffic disruption during the critical node test. Currently, all the packets traversing the testing node through the shared communication link with the node under test and the communication link being used as an alternate path experiences some packet loss as shown in Figure 5. AODV RREQ and RREP packets will be affected since they are encapsulated in UDP packets, but there are no RERR messages created by the host node during the critical test and the AODV HELLO messages are completely unaffected. This is very important, because after the end of the critical test all the previously established routes are restored, and the routing table is restored to its original state. The duration of the critical node test depends on the network density and topology. Critical node conditions, however, are likely to occur when a node has a relatively small degree and therefore fewer tests are required. We are currently investigating ways to eliminate this traffic disruption completely for topologies in which there is no critical node.

The purpose of the trigger mechanism is to minimize the invocation of the critical node test. The trigger mechanism is a light weight monitor that calls the critical node test only when it suspects the neighboring nodes might be critical. The trigger mechanism runs on a testing node and records the Ethernet and IP headers of each incoming and outgoing packet that is routed through the testing node. The testing node does not store any packets it sends or receives; instead it tabulates statistics on the Ethernet and IP packet headers. The testing node tabulates information such as the ID of each neighboring node, its IP address, MAC address, and the time that the packet was forwarded. In addition, the testing node counts the number peer-to-peer pair connections that traverse the testing node. The term connection refers to a pair of nodes that have a peer-to-peer TCP, UDP, or ICMP connection. This connection is not the same as a TCP session. These peer-to-peer connections are associated with the neighbor's MAC Ethernet source address, if the packet is incoming, or with the neighbor's MAC Ethernet destination address, if the packet is outgoing. The trigger mechanism can distinguish between the two cases because if the testing node's MAC address is in the Ethernet destination field, that means the destination is the testing node, therefore it is an incoming packet. If the testing node's MAC address is located in the Ethernet destination field this means that the host is either generating this packet or the host is forwarding this packet. The trigger mechanism creates two tables: incoming packets and outgoing packets. From these tables the trigger mechanism tries to determine if several nodes rely on a communication link incident to the testing node or if the incident communication link is responsible for a significant amount of traffic. If either of these of these conditions occurs, the trigger mechanism can invoke the critical node test. The trigger mechanism is configurable and requires no changes to the routing table. The approach presented in this paper allows the node to make an informed decision on how it will expend its resources. Neither the trigger mechanism nor the critical node tests need to be executed by all the nodes in the network. Moreover, they do not require the collaboration of other nodes in the network beyond the faithful execution of the TCP/IP protocols. The following section will provide detailed examples of how the trigger mechanism and the critical node test work.

4. Implementation Details and Examples

The implementation described in this paper was tested using the mLab testbed that allows users to create arbitrary network topologies. By changing the logical topology of the network, mLab users can conduct tests in an ad hoc network without having to physically move the nodes. mLab controls the test scenarios through a wired interface, while the ad hoc nodes communicate through a wireless interface [15].

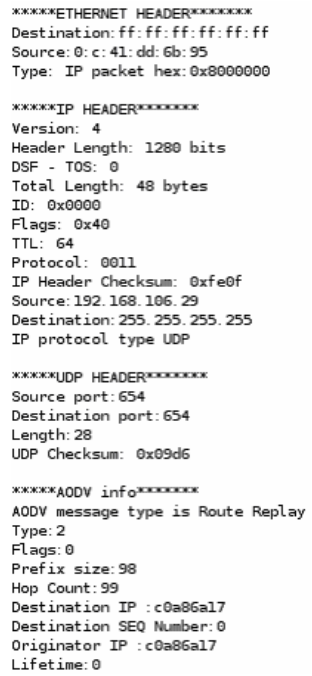


"Figure 1. A sample topology generated by mLab used to demonstrate the trigger mechanism and the critical node test."

The topology shown in Figure 1 is used to show how the trigger mechanism collects information and determines if a node and its incident communication links warrant the invocation of the critical node test.

In order to illustrate the detection of critical nodes, we first generate some test traffic in the network. TCP socket servers are initiated at nodes 33 and 22 to generate TCP traffic. Two TCP socket clients are

initiated at nodes 36 and 28. These clients send simple socket messages every 2 to 3 seconds to the servers. Node 37 initiates a ping of node 33 and similarly node 35 initiates a ping of node 24 in order to create ICMP packet traffic within the network. Finally two Secure Shell (SSH) sessions are initiated between node 27 and node 37, and node 36 and node 27. The trigger mechanism begins by separating the incoming and outgoing activity into two categories: broadcast and non-broadcast packets.



"Figure 2. An AODV broadcast packet that was sent by node 29 and captured by node 'testing node 21'."

Figure 2 shows an AODV broadcast packet that has been sent by node 29 and captured by node testing node 21. This broadcast packet illustrates how testing node 21 can associate node 29's source IP address of 192.168.106.29 with its MAC address 0:c:41:dd:6b:95. This is enabled by the AODV Hello messages with an Ethernet destination address ff:ff:ff:ff:ff:ff that are broadcast every 0.5 or 1 second. Similarly, each node can build a table mapping its neighbors' MAC addresses with their corresponding IP addresses.

Figure 3 shows a sample incoming non-broadcast packet sent by node 23 and captured by testing node 21. Testing node 21 has a MAC address of 0:c:41:dd:69:4e and IP address 192.168.106.21. Packets without an Ethernet destination address of ff:ff:ff:ff:ff:ff are considered to be non-broadcast. Having already created a table linking each of node 21's neighbors' MAC address to their IP address, testing node 21 knows that MAC address 0:60:b3:6a:5:1a belongs to node 23 (IP address 192.168.106.33). This packet shows that the communication link between node 21 and node 23 serves the connection between node 36 (IP address 192.168.106.36) and node 33 (IP address 192.168.106.33).

Packets that belong to the same peer-to-peer connection have the same IP source and IP destination address pair in the IP header fields of every non-broadcast packet. The trigger mechanism stores this information in two separate tables. One table summarizes all the incoming packets (in this case the testing node's MAC is located in the Ethernet destination field). The other table stores the outgoing or forwarded packets (in this case the testing node's MAC address is located in the Ethernet source field). If the host's MAC address is not in the Ethernet destination field, then the packet is forwarded and the source-destination pair count is incremented.

"Figure 3. A sample packet sent by node 23 and received by node 21."

For example, the first row of table 1 shows packets recorded by node 21 related to the peer-to-peer connection between IP source address 192.168.106.36 and IP destination address 192.168.106.33. This connection relies on the communication link between node 21 and node 23 serves. Note that 152 packets have been logged with node 21 as the Ethernet Destination address and node 23 as Ethernet Source address. Using the same topology as shown in Figure 1, the following example shows how testing node 21 initiates the trigger mechanism.

"Table 1. Incoming packets captured by node 21 (IP address 192.168.106.21). [Z=192.168.106]"

Ethernet Source (Neighbor's IP)	IP - Remote source	IP – Remote Destination	Number of packets recorded
Z.23	*Z*.36	*Z*.33	152
	Z.23	*Z*.21	31
	Z.22	*Z*.28	157
	Z.35	*Z*.24	104
	Z.37	*Z*.33	102
	Z.37	*Z*.27	18
	Z.36	*Z*.27	21
	Total number of packets through node 23		585
Z.26	*Z*.33	*Z*.36	121
	Z.26	*Z*.21	33
	Z.33	*Z*.37	102
	Z.26	*Z*.37	1
	Total number of packets through node 26		257
Z.29	*Z*.29	*Z*.21	4
	Z.28	*Z*.22	264
	Z.24	*Z*.35	104
	Z.27	*Z*.37	21
	Total number of packets through node 29		393

Table 2 shows the outgoing packets captured by node 21 after 70 seconds have elapsed under the traffic conditions described above. The number of sessions that the testing node serves can underscore the importance of its links with each of its neighbors. Table 2 shows that the link that testing node 21 (192.168.106.21) shares with neighboring node 23 (192.168.106.23) serves 8 different incoming connections. If this shared link fails, the remote IP sources that appear in the second column of the table must discover new routes to the corresponding nodes in the third column.

The testing node can evaluate the network traffic by the total number of incoming and outgoing connections. Table 2 shows the outgoing packets captured by node 21 (IP address 192.168.106.21). Note that 8 connections are served between node 21 and node 23, 3 between node 21 and node 26, and 4 between node 21 and node 29 during the test sample period. Since the link between node 21 and 23 serves 8 of 15 outgoing connections it can be considered for additional testing. The trigger mechanism also notes that the communication link between node 21 and neighbor node 23 (192.168.106.23) is responsible for forwarding 585 out of 1235 packets. Depending on the sample duration and the configurable settings, the trigger mechanism can request a critical node test to determine if the communication link between node 21 and 23 is a cut-edge.

"Table 2. Outgoing packets captured by node 21 (IP address 192.168.106.21). [Z=192.168.106]"

Ethernet Destination (Neighbor's IP)	IP - Remote source	IP – Remote Destination	Number of packets recorded
Z.23	*Z*.21	*Z*.23	36
	Z.33	*Z*.36	121
	Z.28	*Z*.22	264
	Z.24	*Z*.35	104
	Z.21	*Z*.37	1
	Z.26	*Z*.37	1
	Z.33	*Z*.37	101
	Z.27	*Z*.37	1
	Total number of packets through node 23		629
Z.26	*Z*.36	*Z*.33	152
	Z.21	*Z*.26	28
	Z.37	*Z*.33	102
	Total number of packets through node 26		282
Z.29	*Z*.21	*Z*.29	2
	Z.22	*Z*.28	157
	Z.35	*Z*.24	104
	Z.37	*Z*.27	18
	Total number of packets through node 29		281

Table 3 shows a summary of the total incoming and outgoing connection network traffic for testing node 21. The shared communication link between node 21 and node 23 serves most of the current forwarding activity. Therefore, either the communication link between node 21 and node 23 is supporting the exchange of large files or is a concentration point for the networks traffic. This link serves 9 different nodes of the ad hoc network (192.168.106.36, 192.168.106.33, 192.168.106.23, 192.168.106.22, 192.168.106.28, 192.168.106.35, 192.168.106.24, 192.168.106.37 and 192.168.106.27) and 15 total different connections as shown in Table 3. This information is enough for the trigger mechanism to consider the link between testing node 21 and node 23 (192.168.106.23) as a potential critical link and to initiate the critical node test.

"Table 3. Total incoming and outgoing connections between nodes 21 and 23, 26, and 29. [Z=192.168.106]"

IP Address of Node 21's Neighbors	Incoming and outgoing packets per neighbor	Total Incoming and Outgoing Connections	Number of remote nodes served
Z.23	*1214*	*15*	*9*
Z.26	*539*	*7*	*5*
Z.29	*674*	*8*	*7*

The critical node test checks whether the testing node shares a critical link with node 23 by blocking all the neighbor edges except one and try to ping the specific node. In this example the only communication link that the routing table allows is through node 29.

```
************************************************************

Now we are going to check if a path from the host to  the
neighbor with ip '192.168.106.23' exists [excluding the obvious one hop away]

************************************************************

**********Examing the link HOST - '192.168.106.23' using the link
 HOST - '192.168.106.29' as the only active one .******************
                                                 .
ip route change '192.168.106'/24 via '192.168.106.29'

ping -c 5 -s 10 '192.168.106.23' -A -R

************************************************************

Restore to normal routing...
ip route del '192.168.106'/24 via '192.168.106.29' dev wlan0
ip route add '192.168.106'/24 dev wlan0

************************************************************

The results of the ping ...
PING 192.168.106.23 (192.168.106.23) 10(78) bytes of data.
From 192.168.106.21 icmp_seq=1 Destination Host Unreachable
From 192.168.106.21 icmp_seq=2 Destination Host Unreachable
From 192.168.106.21 icmp_seq=3 Destination Host Unreachable
From 192.168.106.21 icmp_seq=4 Destination Host Unreachable
From 192.168.106.21 icmp_seq=5 Destination Host Unreachable

--- 192.168.106.23 ping statistics ---
5 packets transmitted, 0 received, +5 errors, 100% packet loss, time 4046ms
, pipe 4, ipg/ewma 1011.501/0.000 ms

5 packets transmitted, 0 received, +5 errors, 100% packet loss, time 4046ms
************************************************************
The HOST has a critical (or semi critical) link with node 192.168.106.23
************************************************************
```

"Figure 4. The results of 'critical node test' executed by testing node 21 (HOST) through node 29 to node 23."

Testing node 21 temporarily changes the outgoing routing table. The routing table temporarily blocks all outgoing routing except a single link with one of its neighbors. In this example all traffic is forwarded through the shared communication link between 21 and 29. Testing node 21 then attempts a ping to the node 23 through node 29, without relying on shared communication link between node 21 and node 23. If the ping successfully finds a new route to the node under test without using the shared communication link between node 21 and node 23, then testing node 21 concludes that node 21 and node 23 do not share a critical link. The routing tables are then restored. If nodes 21 and 23 shared a critical link, or cut-edge, more resource intensive monitoring could be requested or alternative measures suitable to the risk environment could be taken.

5. Experimental Results

A series of experiments were conducted using the mLab testbed to examine the effectiveness of the critical node test, as the mLab test bed allows one to replay the same topology changes and traffic scenarios, in order to analyze the effects of each parameter individually. The goal of these experiments was to measure the computational and the communication load of the nodes under various traffic and mobility conditions. In addition, a comparison was made between operating a watchdog IDS continuously versus running the trigger/critical test mechanism to determine if and when the watchdog IDS should be activated. Unfortunately, since there is little to no experience with commercial MANET applications, any speculation on what constitutes high or low mobility is based mostly on intuition.

The emulated ad hoc network was comprised of 10 ARM and 2 ix86 architecture nodes. A heuristic algorithm was used to generate a fully connected graph every few minutes (depending on the mobility scenario). The results were collected after running almost one thousand consecutive topologies, with topology changes taking place every 5 to 10 minutes, depending on the mobility scenario. Another parameter in our experiments was the network traffic, which consisted of TCP, UDP/AODV and ICMP packets. The experiments were conducted under low (4 sockets) and higher (10 sockets) traffic conditions. Using the Linux tools under /proc, we measured CPU usage (total and per process), memory footprints, hard disk utilization and wireless network traffic.

We also measured packet loss for each peer-to-peer connection, based on ICMP packets. A summary of the results and our conclusions is presented here and is displayed in figure 5. Detailed test conditions, adjacency matrices used to represent the logical topology, open source code, and test results can be found on our project web site.

CPU: The watchdog IDS utilized, in most cases, an average of 60-70% of the CPU, while the trigger/critical mechanism utilized an average of about 0.3% and under no conditions did it exceed 1%.

Memory: The initial memory footprint of the watchdog was about 450KB, while the trigger/critical mechanism occupied about 125KB. Of course, since no other applications were running, the processes gradually dominated all of the available memory, in all cases.

Hard disk: The rate at which data was written to and read from the hard disk was, on average, about double in the machines that the watchdog was running, than in the machines that ran the trigger / critical node test mechanism.

Wireless network: The experiments indicate that packet loss is caused mainly because of sudden topology changes and increased traffic, ranging from

1-2% loss for low mobility and traffic to 10-15% for high mobility and traffic. When the critical test mechanism test ran, it caused additional packet loss, as expected, but the overall packet loss only increased by an extra 2-4%, depending on the case.

	X mobility, Y+Z traffic		
	Watchdog	mCritical	simple traffic
Process CPU	79%	0.27%	N/A
Initial memory footprint	456 KB	121 KB	N/A
Packet loss	7.80%	3.80%	1.40%

	X' mobility, Y+Z traffic		
	Watchdog	mCritical	simple traffic
Process CPU	75%	0.24%	N/A
Initial memory footprint	443 KB	127 KB	N/A
Packet loss	4.60%	7.40%	4%

	X mobility, Y'+Z traffic		
	Watchdog	mCritical	simple traffic
Process CPU	34%	0,29%	N/A
Initial memory footprint	446 KB	122 KB	N/A
Packet loss	1.40%	5.00%	1%

Figure 5. A brief table of the experimental results. X stands for 7 topology changes in one hour, X' for 14 topology changes in one hour, Y for traffic created from 2 TCP and 2 UDP sockets, Y' for traffic created from 5 TCP and 5 UDP sockets and Z for 5 peer-to-peer ICMP (ping) connections every 3 seconds.

Of course, one should always bear in mind, that these experiments were conducted in emulation, not in simulation, which implies that, these figures also depend on several factors that also cause packet loss, such as the wireless card temperature during the experiment, environmental interference, potential routing algorithm implementation problems, driver stability issues, etc.

In conclusion, our experiments confirm that the trigger/critical node test mechanism is a lightweight solution that can be used in order to determine the proper conditions to activate a more demanding IDS [2, 3, 4]. When compared to running a watchdog IDS on all nodes under all conditions, the proposed mechanism offers a significant improvement in the efficient use of limited resources, and packet loss introduced by the critical node test is negligible compared to the packet loss resulting from the network mobility.

6. Conclusions

Members of an open ad hoc network are confronted with a dependence on other nodes, for which there are no previous security associations, to faithfully route packets to and from their source and destination, and to collaborate in the detection and notification of the presence of malicious or faulty nodes. Numerous ad hoc IDS methods for detecting malicious nodes have been simulated, fewer have been implemented, and given the scarcity of empirical data and the limited deployment of ad hoc networks in hostile environments, there is not much real-world experience in detecting malicious activity in ad hoc networks. If the total cost of running an ad hoc IDS exceeds the cost of rerouting traffic when confronted with network congestion, faulty nodes, or dynamically changing topologies, there will be little incentive for nodes to collaborate in open ad hoc networks. This paper presents a method for reducing the instances in which nodes must employ resource-consuming collaborative IDS techniques in open ad hoc networks in which the associations are ephemeral and there is no easily enforceable commitment for reciprocity.

The examples provided in this paper illustrate how an ad hoc communication link and node can be tested to determine if the expenditure of additional resources to monitor the behavior of a neighboring node to detect malicious activity is warranted. This test is completed in a relatively short time window (usually a few seconds), without collaborating with any other nodes in the network. As a result, the local resources are used far more efficiently, but, when one considers the total network resources consumed, the approach provides more dramatic savings. The same techniques described in this paper may be used to detect critical links and to provide guidance for how the location of the nodes in an ad hoc network might be better physically arranged in order to provide more fault tolerance and better quality of service.

7. References

[1] A. Patwardhan, J. Parker, A. Joshi, A. Karygiannis and M. Iorga. Secure Routing and Intrusion Detection in Ad Hoc Networks. Third IEEE International Conference on Pervasive Computing and Communications 2005.

[2] Y. Zhang and W. Lee. Intrusion detection in wireless ad hoc networks. In Proceedings of the 6th annual international conference on Mobile computing and networking, pp. 275–283. ACM Press, 2000.

[3] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. ACM/Kluwer Mobile Networks and Applications (MONET), 2002.

[4] C.-Y. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe, and K. Levitt. A specification-based intrusion detection system for AODV. In Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks, pp. 125–134. ACM Press, 2003.

[5] Pirzada, Asad Amir, and McDonald, Chris. Establishing trust in pure ad-hoc networks. Proceedings of the 27th conference on Australasian Computer Science - Volume 26, pp 47-54, 2004

[6] Theodorakopoulos, George and Baras, John. Trust evaluation in ad-hoc networks. Proceedings of the 2004 ACM workshop on Wireless security, pp. 1-10, 2004.

[7] Michiardi, P. and Molva, R., "Core: A Collaborative Reputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks", Communication and Multimedia Security 2002 Conference.

[8] Albers, Patrick and Camp, Olivier. Security in Ad hoc Networks: a general Intrusion detection architecture enhancing trust based approaches. Proceedings of the First International Workshop on Wireless Information Systems 2002.

[9] Sun, Bo, Wu, Kui and Pooch, Udo. Alert aggregation in mobile ad hoc networks. Proceedings of the 2003 ACM workshop on Wireless security, pp.69 – 78, 2003.

[10] Puttini, R; Percher, JM; Me, L, Camp, O; de Sousa, R. A Modular Architecture for a Distributed IDS for Mobile Ad Hoc Networks. Lecture Notes on Computer Science vol. 2669, Springer-Verlag, pp. 91-113, 2003.

[11] Ngai, Edith C. H., and Lyu, M. R.. Trust- and Clustering-Based Authentication Services in Mobile Ad Hoc Networks. 24th International Conference on Distributed Computing Systems Workshops, vol. 04, pp. 582-587, 2004.

[12] Parker, J., Undercoffer, J. L., Pinkston, J., and Joshi, A. On Intrusion Detection in Mobile Ad Hoc Networks. In 23rd IEEE International Performance Computing and Communications Conference – Workshop on Information Assurance. IEEE, April 2004.

[13] S. Marti, T.J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (Mobicom '00), pp. 275-283, 2000.

[14] Graphs: Theory and Algorithms, K. Thulasiraman, M.N.S. Swamy

[15] Karygiannis, A. and Antonakakis, E. mLab: A Mobile Ad Hoc Network Test Bed. 1st Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing in conjunction with the IEEE International Conference in Pervasive Services 2005, July 14, 2005.

[16] Andresen R.: Monitoring Linux with native tools. 30th Annual International Conference of the Computer Measurement Group, Inc. December 5-10, 2004.