An Intrusion Detection Tool for AODV-based Ad hoc Wireless Networks

Giovanni Vigna Sumit Gwalani Kavitha Srinivasan
Elizabeth M. Belding-Royer Richard A. Kemmerer

Department of Computer Science
University of California, Santa Barbara
{vigna,sumitg,kavitha,ebelding,kemm}@cs.ucsb.edu

Abstract

Mobile ad hoc network routing protocols are highly susceptible to subversion. Previous research in securing these protocols has typically used techniques based on encryption and redundant transmission. These techniques prevent a range of attacks against routing protocols but are expensive to deploy on energy-constrained wireless devices. Experience in securing wired networks has demonstrated that, in addition to intrusion prevention techniques, it is useful to deploy intrusion detection techniques as a second line of defense. In this paper, we discuss some of the threats to wireless ad hoc networks, and, specifically, some attacks against the AODV routing protocol. We also present a tool aimed at real-time detection of these attacks. The tool monitors network packets to detect local and distributed attacks within its radio range. Experiments show that the tool provides effective intrusion detection functionality while using only a limited amount of resources.

1. Introduction

A mobile ad hoc network (MANET) is a collection of mobile nodes that are capable of communicating with each other, establishing and maintaining connections as needed. In ad hoc networks, there is no established infrastructure or centralized administration. The topology of an ad hoc network is defined by the geographical positions and the transmission ranges of the nodes. These networks do not have a clearly defined physical boundary, and, therefore, have no specific entry point. As a consequence, access control mechanisms, similar to firewalls in wired networks, are not feasible. In addition, the hop-by-hop routing used in ad hoc networks requires cooperation from the nodes in the network. Therefore, it is not possible to assume that the routing infrastructure can be trusted to any degree.

Wireless ad hoc networks are vulnerable to various attacks. These include passive eavesdropping, active interfering, impersonation, and denial-of-service. Intrusion prevention measures, such as strong authentication and redundant transmission, can be used to address some of these attacks. However, these techniques can address only a subset of the threats, and, moreover, are costly to implement.

The dynamic nature of ad hoc networks suggests that prevention techniques should be complemented by detection techniques that monitor the security status of the network and identify anomalous and/or malicious behavior. These techniques are usually less expensive to implement and can be easily deployed in existing ad hoc networks without requiring modifications to the nodes' configuration or the routing protocols being used.

Intrusion detection techniques have traditionally been classified into two paradigms, namely *anomaly detection* and *misuse detection* [5]. In anomaly detection techniques, historical data about a system's activity and specifications of the intended behavior of users and applications are used to build a profile of the "normal" operation of the system. The detection process then attempts to identify patterns of activity that deviate from the defined profile. Misuse detection techniques take a complementary approach. Misuse detection tools are equipped with a number of attack descriptions (or "signatures") that are matched against the stream of audit data to identify evidence of the occurrence of the modeled attacks. Misuse and anomaly detection both have advantages and disadvantages. Misuse detection can perform focused analysis of the audit data and usually produces very few false positives. However, it can detect only those attacks that have been modeled and possibly variations on those attacks. Anomaly detection has the advantage of being able to detect previously unknown attacks. This advantage is paid for in terms of the large number of false positives generated and the difficulty of training a system with respect to a highly dynamic environment.

Previous work in intrusion detection for ad hoc wireless networks focused on identifying anomalous behavior patterns [22]. The primary concern with anomaly detection approaches is that the large number of false positives and the

overhead involved in modeling the behavior of the system may be too expensive for mobile nodes.

Another limitation of previous intrusion detection approaches is that they have been validated using only simulation-based studies [3, 10, 15, 22]. Simulations are typically useful when larger topologies are used for evaluation. However, simulation-based evaluations need to be complemented by implementation results to completely understand the operational limits of the system and to evaluate the overhead introduced by the intrusion detection process.

This paper describes AODVSTAT, a tool aimed at network-based, real-time intrusion detection for wireless networks based on the Ad hoc On-Demand Distance Vector routing protocol (AODV) [12]. The intrusion detection tool is based on a stateful misuse detection technique [17] that supports effective intrusion detection while keeping the number of false positives low. The tool has been evaluated through both simulation-based and implementation-based testing.

The remainder of this paper is structured as follows: Section 2 gives an overview of the AODV routing protocol. Section 3 describes examples of possible attacks in an ad hoc wireless environment. Previous work on securing ad hoc networks is surveyed in Section 4. Section 5 proposes an intrusion detection architecture for wireless ad hoc networks. Section 6 presents the details of the attacks that were used to test the capabilities of AODVSTAT. Section 7 presents and discusses the results obtained in detecting attacks in wireless networks. Finally, Section 8 draws some conclusions and outlines future work.

2. AODV

The AODV routing protocol [12] is a reactive protocol designed for wireless ad hoc networks. When a source node needs a route to a destination, it initiates a route discovery process to locate the destination node. The source node S floods the network with a route request packet (RREQ), as shown in Figure 1(a), requesting a route to be set up to the destination D. On receiving an RREQ, intermediate nodes update their routing table with a reverse route to the source. All the receiving nodes that do not have a route to the destination broadcast the RREQ packet to their neighbors, with an incremented hopcount.

A route reply (RREP) is sent back to the source node when the RREQ query reaches either the destination itself or any other intermediate node that has a current route to the destination. As the RREP propagates to the source, the forward route to the destination is updated by the intermediate nodes receiving an RREP packet. In Figure 1(b), both the destination node D and intermediate node G have a route to the destination. Hence, they reply to the RREQ with an RREP packet.

AODV uses sequence numbers to determine the freshness of routing information and to guarantee loop-free routes. In case of multiple routes, a node selects the route with the highest sequence number. If multiple routes have the same sequence number, then the node picks the route with the shortest hopcount. Timers are used to keep the route entries fresh.

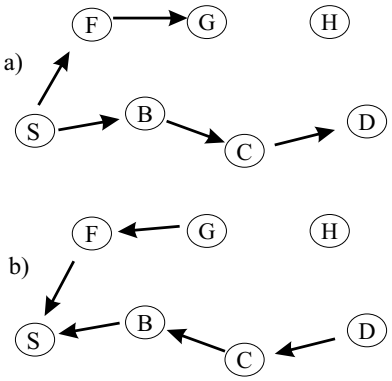


Figure 1. Route Discovery in AODV.

When a link break occurs, route error (RERR) packets are propagated along the reverse path to the source, invalidating all broken entries in the routing tables of the intermediate nodes. AODV also uses periodic HELLO messages to maintain updated information about the connectivity of neighboring nodes.

The AODV protocol does not incorporate any specific security mechanism, such as strong authentication. Therefore, there is no obvious way to prevent mischievous behavior such as MAC spoofing, IP spoofing, dropping packets, or altering the contents of control packets. Protocols like SAR [19] and SAODV [21] protect AODV against a limited number of attacks but at the cost of performance in terms of overhead and latency.

3. Attacks in Wireless Ad hoc Networks

Authentication, non-repudiation, availability, integrity, confidentiality, and privacy are the goals for a secure ad hoc network [24]. In this section, we classify the attacks against ad hoc routing protocols, and in particular against AODV, according to these security goals. Figure 2 summarizes most of the attacks in a mobile ad hoc network running the AODV routing protocol. Note that other attacks exist that operate at different abstraction levels, e.g., at the physical level or at the application level. Hereafter, we focus exclusively on attacks that affect the routing of traffic in MANETs.

3.1. Authentication and Non-Repudiation Attacks

Authentication allows a node to verify the identity of a peer node with which it is communicating. Non-repudiation is the ability to prove that a sender sent a message. Most ad hoc routing protocols use either MAC or IP addresses to

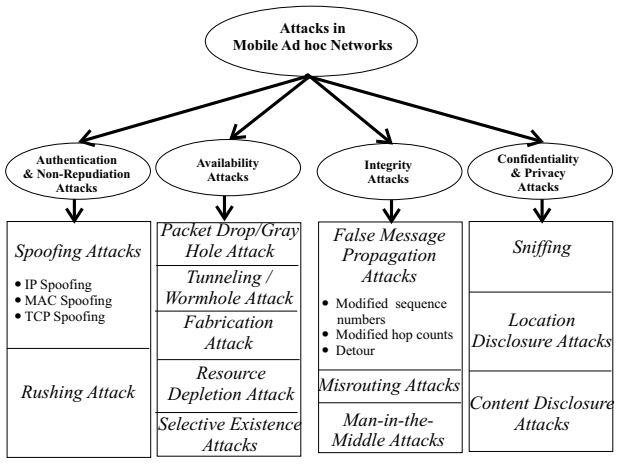


Figure 2. Attacks in AODV MANETs.

uniquely identify hosts in the network. Therefore, spoofing one of these two addresses is the simplest method to attack the security goals of authentication and non-repudiation.

3.2. Availability Attacks

Availability guarantees that network services (e.g., bandwidth and connectivity) are accessible to authorized entities in a timely manner. In the following, we present a variety of denial-of-service attacks, which are used to reduce or completely deny the availability of network services.

Dropping of Packets. These attacks are carried out by dropping either data packets or control packets. In the former case, a malicious node, after advertising a correct path to the destination, simply drops data packets to perform a denial-of-service attack. In the latter case, a malicious node drops control packets sent by other nodes, while at the same time initiating a route discovery whenever it needs to send data. This is also known as routing misbehavior [10].

Fabrication Attacks. A malicious node can launch a denial-of-service attack against a destination node by impersonating a node along a path to that destination and continually sending route error messages for the destination node. On receiving the route error messages, the nodes using that route will delete the route table entry for the destination node. In the absence of alternative routes, or by continually transmitting error messages, communication to the destination node can be successfully prevented.

Resource Depletion Attacks. Network bandwidth utilization is an important issue in a wireless ad hoc network. A single node or a group of nodes in collusion can perform a resource depletion attack. For example, two nodes can transfer large volumes of data between themselves, thereby clogging the network and depleting the available network bandwidth. A resource depletion attack can also be achieved by using control messages. For example, a malicious node can flood the network with route requests to random addresses. This would lead to an RREQ storm that depletes the available network bandwidth.

Selective Existence Attacks. In these attacks, a malicious node behaves selfishly, using the network for its own needs, without participating in the overall routing process. For example, a malicious node may not send any HELLO messages. As a result, nodes in the ad hoc network do not know about the existence of the malicious node. However, when the malicious node needs to send a data packet, it performs a route discovery and obtains a route to the destination. Then, the malicious node only forwards packets from the neighboring nodes that the malicious node has to depend on to send data. When the malicious node no longer needs to use the network, it switches back to "silent mode," and, as a consequence, the routing entries for the malicious node are invalidated in the routing tables of its neighbors.

3.3. Integrity Attacks

Integrity guarantees that a message is not altered on its path to the destination. In the following, a variety of integrity attacks are discussed.

False Message Propagation Attacks. One instance of this type of attack is the *redirection with modified sequence numbers*. In this attack, a malicious node advertises a route to a node with a destination sequence number greater than the authentic value. By doing this, it diverts the traffic towards the attacker because the nodes will select the RREP with the highest destination sequence number. This attack is specific to AODV. Modification of the hopcount also has a similar effect on the routes chosen.

Another example of this class of attacks is the *detour attack*, where the attacker adds a number of virtual nodes to a route during the route discovery process. As a consequence, the traffic is diverted to other routes that appear to be shorter, and the attacking node can save energy because it does not have to forward packets to that destination. This attack is specific to source routing protocols.

Misrouting Attacks. In this class of attacks, a malicious node attempts to send a data packet to the wrong destination. For example, this can be achieved by forwarding a data packet to the wrong next hop in the route to the destination or by modifying the final destination address of the data packet.

Man-in-the-Middle Attacks. A malicious node can combine the spoofing attack and the dropping of packets to perform a man-in-the-middle attack. The malicious node uses its place on the route as the first step in a man-in-the-middle attack by disallowing route requests to the destination. The attacker must be the only node within the range of the destination, or must be able to prevent any other route requests to the destination. The attacker sends a spoofed route reply to the source node and establishes a route with the source. The attacker then sends a new route request to the destination node,

establishes a route with the destination, and then drops the reply packet on its way to the source node. By doing this, the attacker controls the communication between the source and the destination.

3.4. Confidentiality and Privacy Attacks

Privacy guarantees non-disclosure of personal information stored at a node to any other node in the network. Confidentiality ensures that certain information is disclosed only to authorized entities.

Location Disclosure Attacks. The location disclosure attack reveals physical information about a particular node (e.g., the location of the commander of a troop). This attack is based on the principle that, for all multi-hop routing protocols, any two consecutive nodes in a route must be geographically close. This information can be gathered through promiscuous listening or by using tools such as *traceroute*. Using this principle in a recursive manner, location information about nodes in an ad hoc network can be disclosed.

Content Disclosure Attacks. The content disclosure attack enables a malicious attacker to learn the contents of messages being transmitted. Encryption protocols such as WEP, that are a part of the 802.11 standard, are used to prevent the disclosure of message contents to unauthorized nodes. In order to violate the privacy of the communication, an attacker needs to break the WEP encryption protocol [11] and any additional encryption protocols used.

4. Related Work

Most of the previous work in securing ad hoc networks has focused on strengthening the ad hoc routing protocols using techniques such as hash functions [7, 21], encryption [6, 9, 13, 19, 24], and redundant transmission [24]. Hu and Perrig [20] survey the weaknesses and strengths of various secure routing protocols. However, these may not be suitable for a power-critical device. In addition, the lack of a centralized key distribution authority or a shared public key infrastructure makes it difficult to design secure protocols. Finally, these protocols address only a subset of the possible attacks. For instance, [24] does not protect against packet dropping attacks, while [13] is susceptible to packet dropping, denial-of-service, and replay attacks. Therefore, these prevention techniques should be complemented by intrusion detection techniques.

Zhang and Lee [22] present an intrusion detection technique for wireless ad hoc networks that uses cooperative statistical anomaly detection techniques. Each intrusion detection agent runs independently and detects intrusions from local traces. Only one-hop information is maintained at each node for each route. If local evidence is inconclusive, then neighboring IDS agents cooperate to perform global intrusion detection. This approach leverages information about the physical location of the nodes. Therefore, each node needs to have an IDS running and a built-in GPS device, which requires extra processing power and sensing capabilities at each of the MANET nodes. Unfortunately, in most ad hoc networks, some of the nodes are low-powered devices that are not capable of handling the additional overhead introduced by the detection process. Furthermore, the use of only anomaly detection techniques can result in too many false positives, which will further increase the overhead. The approach to intrusion detection presented in this paper utilizes misuse detection techniques to reduce the number of false positives and it does not require all the nodes in the network to have sensing capabilities.

Venkatraman [16] extends the Zhang and Lee model by modifying the protocol so that two-hop information is maintained at each node for each route. The detection scheme uses threshold levels to identify falsified route requests and replies, packet dropping, and route hijacking attacks. However, this scheme requires a modified protocol in addition to requiring an intrusion detection system on each node.

Albers et al. [1] suggest an architecture for local detection based on mobile agents that has several similarities to the one proposed in [22]. In the proposed architecture, the intrusion detection agent running at each node depends on the SNMP agent for audit data. Unfortunately, no information has been provided regarding the techniques that the authors use for intrusion detection or the overhead involved.

Stamouli proposes an architecture for Real-Time Intrusion Detection for Ad hoc Networks (RIDAN) [15]. The detection process relies on a state-based misuse detection system. In this case, every node needs to run the IDS agent. There is no mention of a distributed architecture to detect attacks that require more than one-hop information.

An additional limitation of most of the previous studies is that they were restricted to simulations to analyze the effectiveness of the intrusion detection process. They did not implement an ad hoc network IDS, nor did they quantify the performance of their IDS in terms of the percentage of attacks detected and the performance overhead.

5. An Intrusion Detection System for Ad hoc Wireless Networks

We designed and implemented an intrusion detection tool, called AODVSTAT, to detect attacks against the AODV routing protocol in wireless ad hoc networks. AODVSTAT is based on the State Transition Analysis Technique (STAT), which was initially developed to model host-based and network-based intrusions in a wired environment [8]. AODVSTAT sensors are deployed on a subset of the nodes of an AODV-based ad hoc network.

In STAT, computer penetrations are described as sequences of actions that an attacker performs to compromise the security of a computer system. States represent a snapshot of a system's volatile, semi-permanent, and permanent memory. A description of an attack has a *safe* starting state, zero or more intermediate states, and at least

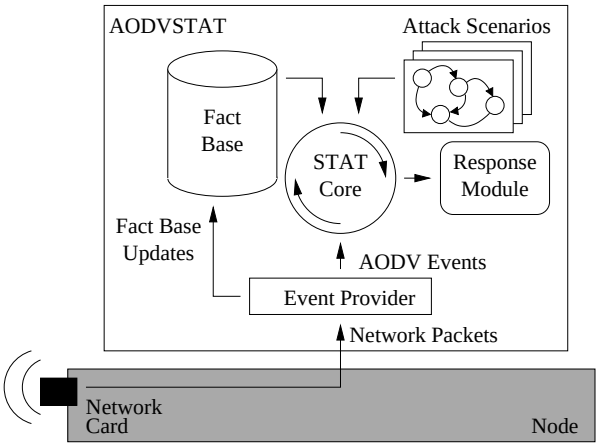


Figure 3. AODVSTAT Sensor Architecture.

one *compromised* ending state. States are characterized by means of assertions, which are functions with zero or more arguments returning boolean values. Typically, these assertions describe some aspects of the security state of the system, such as file ownership, user identification, or network traffic characteristics. Transitions between states are annotated with signature actions that describe the actions that, if omitted from the execution of an attack scenario, would prevent the attack from completing successfully. Signature actions are expressed by leveraging an event model. In AODVSTAT, events are either data packets or AODV control messages exchanged over a wireless network.

An AODVSTAT sensor has two modes of operation. In stand-alone mode, a sensor detects attacks within its immediate neighborhood only. In distributed mode, sensors periodically exchange UPDATE messages containing details of the neighboring nodes of each sensor. More precisely, UPDATE messages contain the list of known MAC/IP pairs, the number of hops to known nodes in the network, and information regarding detected local attacks. This information is then used to detect attacks in a distributed fashion.

5.1. AODVSTAT Sensor Architecture

An AODVSTAT sensor performs stateful analysis of the packet streams to detect signs of intrusion. The architecture of an AODVSTAT sensor is shown in Figure 3. The AODVSTAT sensor monitors the surrounding network using a packet sniffer. The packets retrieved from the network are then matched against a number of state-transition attack scenarios, each describing a particular type of attack. When a match is found, a response is initiated, usually in the form of an intrusion detection alert.

The detection process relies on an internal fact base, which contains updated information about the neighbor nodes. The fact base is updated by analyzing the observed data packets and AODV control messages. More precisely, data packets are used to determine how much traffic has been generated, received, and forwarded by each node, while AODV control packets are used to extract the AODV sequence numbers of the active nodes, the IEEE 802.11 header details (such as the frame control field and the IEEE sequence number), and the MAC/IP address pairs of the nodes in the sensor's range. When a sensor operates in distributed mode, the fact base also contains information received from other sensors by means of UPDATE messages.

6. Attack Modeling Details

A sensor running AODVSTAT gathers information about the network by monitoring packets within its range and by collecting UPDATE messages from other sensors. In the following, we present scenarios for both the one-hop attacks and the distributed attack that were used in the AODVSTAT experiments. State transition diagrams are used to graphically represent the various detection signatures.

6.1. Spoofing

Many routing protocols use MAC and IP addresses as unique identifiers. Unfortunately, MAC addresses can be spoofed in most of the 802.11 cards with the help of either the *ifconfig* tool or by using the ioctl function with the `SIOCSIFHWADDR` flag. Wright [18] discusses some techniques to detect MAC spoofing in wireless LANs. In particular, Wright describes a method for using the values of the sequence numbers in the IEEE 802.11 frame header to detect MAC spoofing. These sequence numbers usually increase linearly as traffic is produced. By identifying anomalous changes in the generation of sequence numbers for a specific MAC address, it is possible to detect MAC spoofing. This technique is difficult to evade because IEEE 802.11 sequence numbers can be modified only by using the firmware of the wireless cards and cannot be tampered with in most implementations [2].

AODVSTAT extends the sequence control technique to detect MAC spoofing and uses stored mappings of MAC/IP pairs to detect IP spoofing attacks. A simplified version of the corresponding scenario is shown in Figure 4. When a packet arrives, the MAC/IP information is stored for the first time. If another packet with the same IP but different MAC, or vice versa, arrives, the packet is detected as spoofed.

6.2. Dropping of Packets

A misbehaving node can drop packets to conserve its energy. Packet dropping is detected by checking whether a sensor's neighbor forwards packets towards the final destination. Therefore, a neighbor table is maintained to determine the nodes that are neighbors of the IDS. Scenarios to detect dropping of different packet types are as follows:

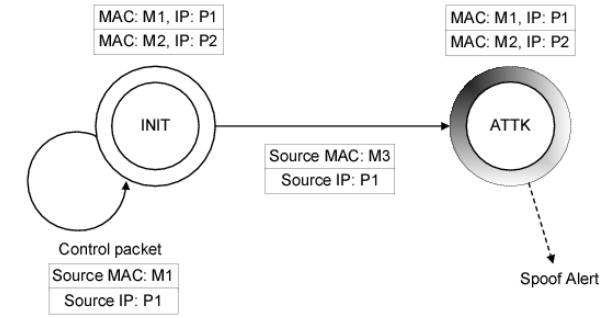


Figure 4. Spoofing.

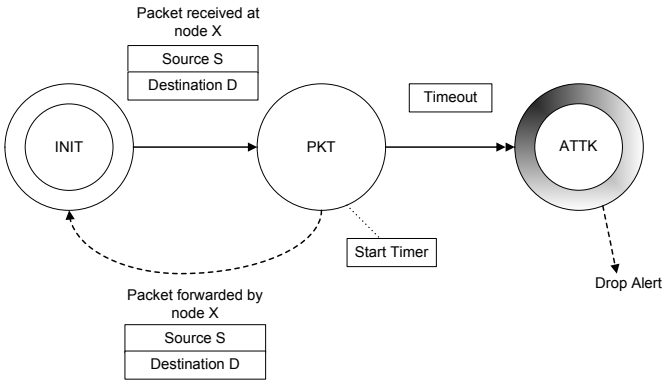


Figure 5. Dropping of RREP, RREQ, and Data Packets.

Dropping of AODV RREQ Packets. RREQ packets are broadcast packets. As described in Section 2, an intermediate node either replies with an RREP packet if it has a route to the destination, or it rebroadcasts the RREQ packet. It drops the RREQ packet only if it has previously received that packet. The IDS detects this attack by maintaining a list of all of its neighbors. This list is refreshed dynamically to allow for the continually changing topology. The IDS detects a dropped packet if any of its neighbors fails to reply with an RREQ or RREP packet. If the IDS itself replies to an RREQ packet with an RREP packet before its neighbors, then the IDS considers the packet to be forwarded. In this case, the neighbors will not be expected to forward the RREQ packet.

Dropping of AODV RREP Packets. To detect dropped RREP packets, the IDS checks whether its neighbors forward the control packets correctly. For every RREP packet sent to a sensor's neighbor and observed by the IDS, the IDS checks whether its neighbor forwards the packet towards the source node, if the neighbor is not the intended final destination of the RREP packet. If the destination MAC address of the RREP packet does not belong to a neighbor node, the IDS ignores the RREP packet.

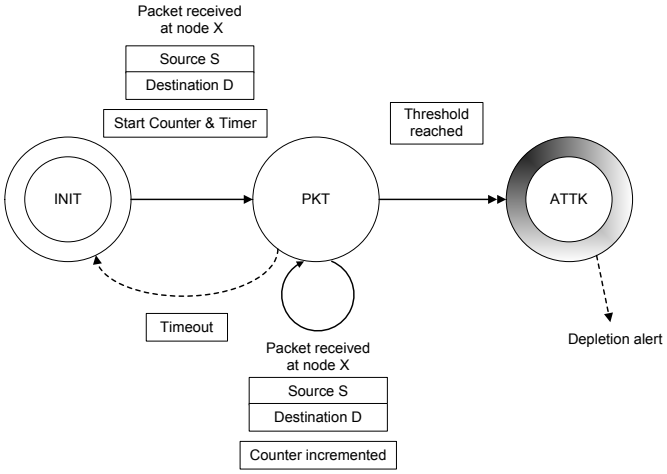


Figure 6. Resource Depletion.

Dropping of Data Packets. This is similar to the above scenario. Here, the IDS checks whether its neighbors forward the data packet towards the final destination node. An attack is detected if, after a certain time interval, the observed packet is not forwarded by the neighbor.

The three scenarios described above have a similar structure, shown in Figure 5. The sensor starts from the initial state (INIT). On sniffing a unicast packet sent to one of its neighbor that is not the final destination, the sensor moves to an intermediate state (PKT) and starts a timer. If the neighbor forwards the packet before the timer expires, then the sensor returns to the initial state. If the timer expires, then the sensor moves to the final ATTK state and raises a packet drop alert. To reduce the number of false positives and to ignore alerts due to collisions, an alert is raised only if the number of dropped packets reaches a certain threshold.

6.3. Resource Depletion Attack

A malicious node can deplete network resources by transmitting a large number of data or control packets. The IDS detects this scenario by maintaining a count of the number of packets that it receives from each node within a specific time window. If this count crosses a threshold, then an alert is signaled. The threshold for control packets is obtained as a function of the number of neighbors that a node has, the rate of HELLO messages, and the rate of retransmission of other control packets.

Figure 6 shows the state transition diagram for a resource depletion attack. On sniffing the first packet between the source and the destination, the sensor makes a transition from the initial state (INIT) to an intermediate state (PKT). In this state, both a counter and a timer are initialized. The

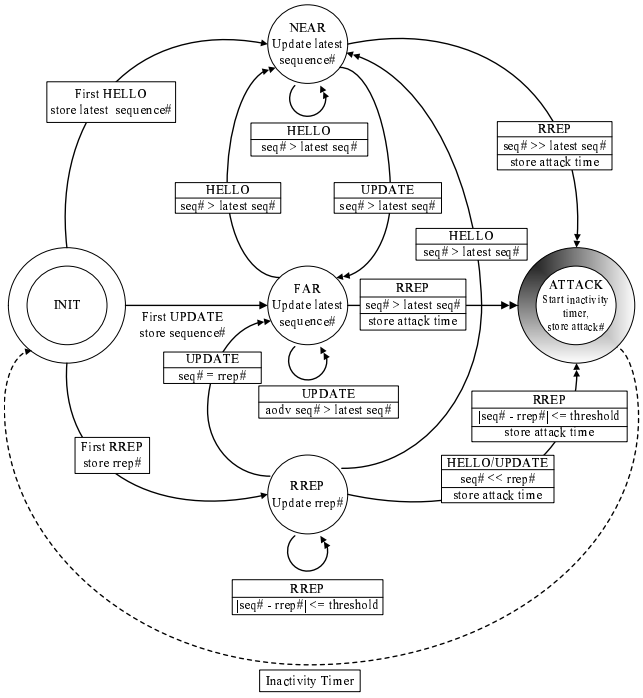


Figure 7. False Propagation of Sequence Numbers.

counter is incremented for every packet the sensor observes between the same source and destination. If the count crosses a threshold within a specific time interval, the sensor moves to ATTK state and raises an alert. The counter is reset if the sensor observes an idle period for a substantial amount of time. A separate counter for the total number of packets sent by each source to multiple destinations is maintained to detect variations of the depletion attack.

6.4. False Propagation of Sequence Numbers

A malicious node can send out RREP packets with a very high sequence number for some destination, in order to increase the chances of being included in the route. To detect this kind of attack, it is necessary to rely on a distributed set of sensors that communicate neighbor information to every other sensor node. Each sensor node detects an attack on receipt of an abnormally high sequence number contained in the RREP for a particular destination.

The state transition diagram in Figure 7 represents the various states associated with the evolution of the attack. The NEAR state denotes that the sensor is within range of the concerned node. The FAR state indicates that the node is not within monitoring range. In this case, the sensor needs to update the sequence number based on the information obtained from other sensors. Sequence number information for a particular node is updated only through HELLO messages from a particular node or through UPDATE messages that are used for sensor communication. The RREP state is reached when the sensor is unable to receive HELLO or UPDATE messages, but receives the RREP containing the destination sequence number for a particular node. In this case, the latest known sequence number, called the *rrepnum*, is updated instead of the latest known AODV sequence number. From the INIT state, depending on the type of message received, the corresponding transition to the next state takes place.

A transition from the NEAR state to the FAR state is made when an UPDATE message from another sensor is received with a greater sequence number for the node. Consequently, a transition from the FAR state to the NEAR state is made when a HELLO message is received. Similarly, a transition from the RREP state to NEAR or FAR state occurs on the receipt of the appropriate message. If an RREP packet with a sequence number higher than the known value by a certain threshold is received, an attack is detected. A transition is made back to the initial state after a certain period of inactivity.

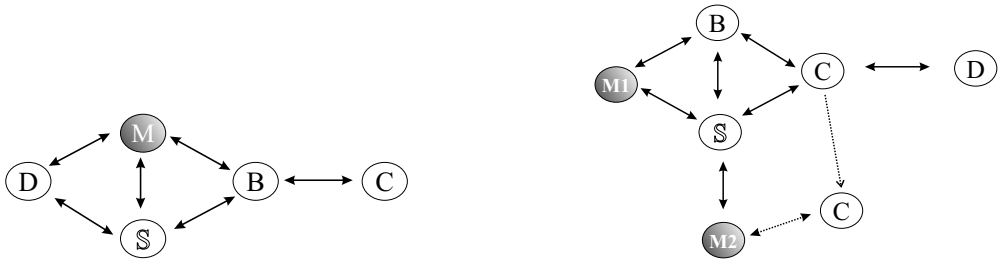
Other attacks discussed in Section 3 can be detected by a combination of the scenarios presented so far or by writing additional scenarios. For instance, the man-in-the-middle attack can be detected by a combination of the spoofing and the packet drop scenarios. Scenarios to detect attacks against integrity, described in Section 3.3, require additional information from the peer sensors in the network.

7. Experiments

The effectiveness and the performance of the AODVSTAT tool were evaluated using both implementation-based and simulation-based testing. More precisely, for smaller topologies and one-hop attacks, we used implementation-based testing, while for larger networks and multi-hop attacks we used a simulation tool. This was done because larger topologies were required to test the effectiveness of distributed detection. This section presents the experimental setup and the results for the implementation-based testing, the emulator that was used for testing the distributed scenario, and the experimental setup and results obtained from the simulation-based testing.

7.1. Implementation-based Testing

A wireless intrusion detection testbed comprised of six ad hoc nodes, including four Dell Latitude C600 laptops and two Sony Vaio laptops, was set up for all the tests. The Dell laptops have Mobile Pentium III-750 MHz processors and 128MB of RAM, and the Sony laptops have Mobile Pentium IV-1.6 GHz processors and 512MB RAM. All the laptops had the Linux operating system and ran the UCSB AODV implementation [4]. For wireless connectivity, Lucent Orinoco and NETGEAR IEEE 802.11b wireless cards were used. The laptops were physically co-located and the connectivity was controlled using *iptables* for MAC



(a) Static Configuration.

(b) Controlled Movement Configuration.

Figure 8. Experimental Network Topologies.

layer filtering. This provided a controllable environment in which AODVSTAT could be tested.

One of the nodes in the ad hoc network acted as an AODVSTAT intrusion detection sensor. During all the experiments, the sensor nodes participated in the AODV routing protocol. For each experiment, data traffic consisted of 512-byte UDP packets. The transmission rate for data packets was set at ten packets per second.

Tests were carried out using three experimental setups. Each setup examines the ability of the AODVSTAT sensor to detect one-hop attacks under varying settings. The three setups are as follows:

- **Static Configuration:** The testbed for the first experiment is comprised of five nodes as shown in Figure 8(a). Node M is the malicious node, and node S is the node running AODVSTAT. The arrows indicate the pairs of nodes that can communicate directly. Since the node S is in range of the malicious node M, local one-hop attacks should be detected by the IDS.
- **Controlled Movement:** The testbed for the second experiment is comprised of six nodes. Node S is the host running AODVSTAT. The initial network configuration is as shown in Figure 8(b). In this configuration, nodes M1 and M2 are malicious. After a certain time period, node C moves, and, as a consequence, it can only communicate with node M2. This shows the effect of movement on the ability of the IDS to detect attacks.
- **Random Movement Configuration:** The final experiment also consists of six nodes. Similarly to the previous case, nodes M1 and M2 act maliciously and node S runs AODVSTAT. The nodes are placed in random positions. Movement is simulated by randomly changing the configuration of the ad hoc network at the half-way point of the test. This experiment helps to determine the ability of the tool to detect attacks in dynamically changing environments.

The following attacks were run on the three experimental setups.

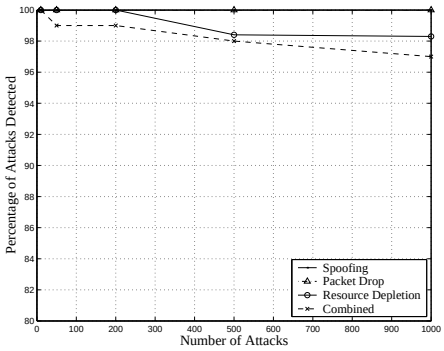
- **Spoofing:** For the first experimental setup, node M spoofs node D. In the second setup, node M1 initially spoofs node C. Then, M2 impersonates node B after node C moves out of the range of S. In the third setup, M1 and M2 impersonate other nodes in the ad hoc network.
- **Dropping Packets:** In this attack, node M drops packets for a short interval in the first setup. M1 and M2 drop packets in the last two experimental setups.
- **Resource Depletion Attacks:** In the first setup, node M sends an excessive number of unwanted data and control packets to saturate the ad hoc network. In the other two setups, similar activities are carried out by nodes M1 and M2.

For each of the three experimental setups, individual attacks, as well as a combination of the attacks, were executed. The number of attacks in each run was varied from 10 to 1000. For each run, we measured the percentage of attacks that were correctly detected, the percentage of false positives generated, and the overhead in terms of both memory and CPU. The results for each of the experimental setups were obtained by averaging the values obtained across five runs.

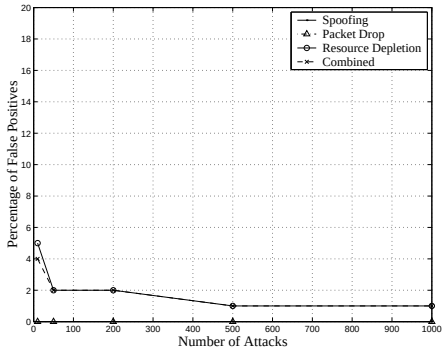
Figure 9 shows the effect that varying the number of one-hop attacks has on the percentage of attacks detected and the percentage of false alerts raised for the static configuration. In all cases, more than 97% of the attacks were detected by the IDS with fewer than 5% false positives. Though not shown, we found that there is a trade-off between the attacks detected and the false positives that depends on the threshold value used in the resource depletion scenario.

The percentage of attacks detected and the number of false positives for the controlled movement configuration are shown in Figures 10(a) and 10(b), respectively. The results are similar to the static configuration. There is an increase of approximately 5% in the number of false positives due to the node mobility. This is caused by the latency of the IDS in updating its list of neighbors when one of the neighbors of the IDS moves away.

The results for the random configuration for one-hop attacks are shown in Figure 11. The percentage of attacks de-

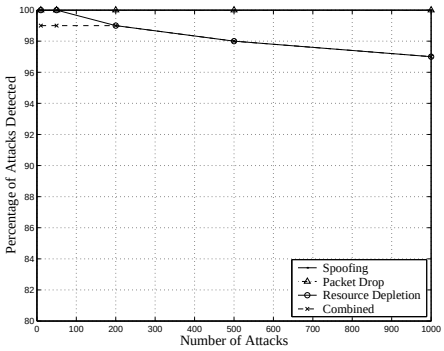


(a) Percentage of Attacks Detected.

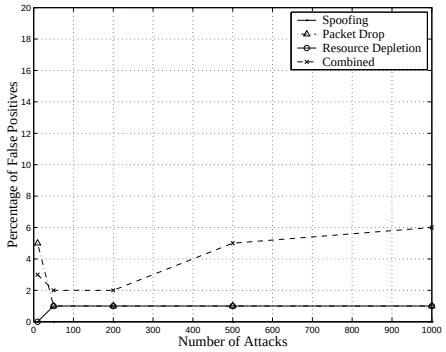


(b) Percentage of False Positives.

Figure 9. Static Configuration Results.



(a) Percentage of Attacks Detected.



(b) Percentage of False Positives.

Figure 10. Controlled Movement Configuration Results.

tected indicates that even with random movement the IDS is capable of detecting most of the attacks with a small number of false positives (only 5% of the total alerts raised were false positives). Also, the IDS was able to detect 96% of the local one-hop attacks.

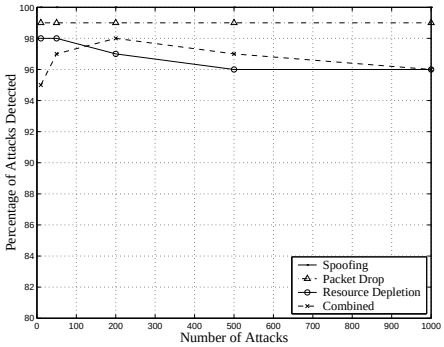
For all three configurations, the false positives were primarily due to threshold levels set for the resource depletion and packet dropping attacks. Some packet drop false alarms occurred because a node took a considerably long time to forward the packet if it was busy or being attacked by a resource depletion attack. The threshold value for resource depletion can be increased to reduce the depletion false alarms. Packet collisions caused by the high load in the network also gave rise to additional false positives. Though not shown, AODVSTAT detected 98% of the MAC spoofing attacks with less than 1% false positives.

Figure 12(a) shows the extra memory overhead as a function of the number of one-hop attacks. The memory required by the node with and without the sensor is plotted. It can be seen that the additional memory overhead due to the AODVSTAT sensor is small. The percentage of average CPU load in the presence and absence of the AODVSTAT sensor is shown in Figure 12(b). The increase in CPU overhead required by a node running the AODVSTAT sensor is less than 3%.

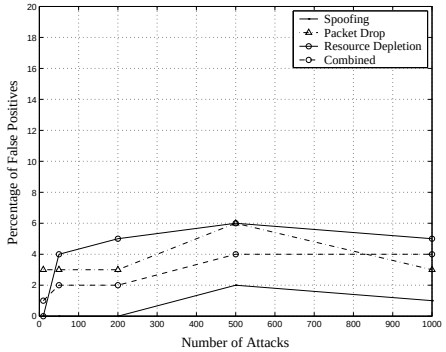
7.2. An IEEE 802.11 Emulator for AODVSTAT

The AODVSTAT tool has been implemented to handle live IEEE 802.11 network traffic. Unfortunately, performance evaluation of the tool implementation is limited to small topologies with constraints on mobility speeds. Therefore, simulation is required to evaluate the system for a broader range of test cases. One advantage of using a simulation-based approach is that larger topologies can be used with randomly-generated mobility patterns.

Since AODVSTAT uses audit data streams obtained from sniffing the wireless interface, simulated events cannot be directly used as the input stream for the sensor, because these events are not in the IEEE 802.11 standard format. An em-

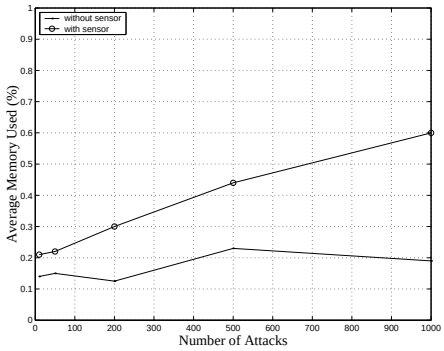


(a) Percentage of Attacks Detected.

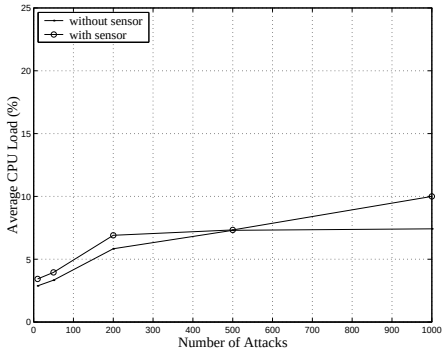


(b) Percentage of False Positives.

Figure 11. Random Configuration Results.



(a) Percentage of Additional Memory Utilization.



(b) Percentage of Average CPU Load.

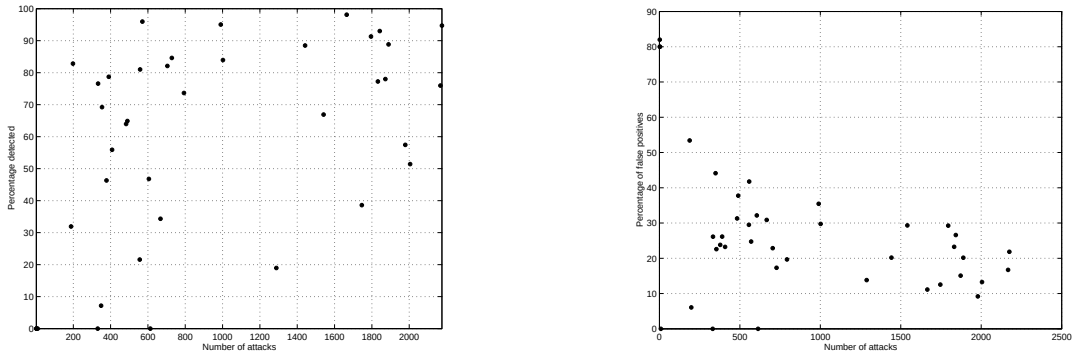
Figure 12. Overhead Introduced by AODVSTAT.

ulator is therefore required to create IEEE 802.11 standard packets, based on the header information available in the simulated events.

We used Network Simulator-2 (NS-2) for modeling various node movement scenarios. A specific set of nodes were designated as IDS agents. All packets sent and received by each sensor node were obtained in the form of traces. These traces were then used to generate actual 802.11 traffic, and the packet stream thus created was provided as input to each AODVSTAT sensor node. It is important to note that the delays between packets were maintained while injecting the packet stream into the network. Thus, the setup ensures that the wireless sensor receives packets at the same rate as during the simulation.

In order to overcome resource constraints involved in creating large topologies, the wired interface was used to deliver network traffic to the sensors. By doing so, the need for a large ad hoc network setup was eliminated. This was achieved by encapsulating the stream of IEEE 802.11 packets within Ethernet 802.1 packets. Each AODVSTAT sensor processes the packets by first extracting the original packet from the encapsulated packet and then adding the packet to the sensor's set of events observed. Prior work [23] on using wired environments for wireless emulation discusses the advantages of performing such integrated testing. MobiEmu [23] is an emulator for wireless networks that allows the use of the local area network for testing. Its primary limitation is that it does not emulate the 802.11 physical and MAC layers, and, therefore, it can be used only for testing and not for performance evaluation. To overcome this problem, a customized version of the 802.11 emulator was developed using libnet [14].

Communication between sensors is simulated by having the nodes send periodic UPDATE messages, which contain the latest known sequence number and the IP/MAC address pair of all the neighbors, to every other sensor. A simulator trace for these UPDATE messages is translated into the format used by AODVSTAT and sent on the network.



(a) Percentage of Attacks Detected.

(b) Percentage of False Positives.

Figure 13. Distributed detection for 50 nodes at speed 5m/s

7.3. Simulation-based Testing

The simulation tool described in the previous section was used to evaluate AODVSTAT in larger topologies. The simulated network consisted of 10, 20 and 50 nodes moving at varying speeds of 0m/s, 2m/s, 5m/s and 10m/s. Around 40% of the nodes chosen at random were equipped with an AODVSTAT sensor. For each experiment, data traffic consisted of 512-byte UDP packets transmitted at a rate of ten packets per second. The duration of each simulation was 500 seconds. The experimental results were averaged over ten different seed values.

False sequence number propagation attacks were performed. The attacks were generated by modifying the AODV and MAC layer implementation in NS-2 for the attacking nodes. The number of attacks was variable for each run and depended on the number of RREP messages generated during the simulation. For each setup, the hit ratio and false positives were measured.

The results indicate, that on average, a greater number of attacks were generated when the size of the network increased. This factor influenced the attack detection rate for different network sizes. The percentage of attacks detected is fairly scattered for less than 200 attacks, but increases as the number of attacks increases. This trend can be observed in Figure13(a) for a network size of 50 nodes. The experimental results for a network consisting of 10 and 20 nodes do not have this characteristic because the number of attacks produced is less. For distributed attack detection, each sensor gathers information about the network from other sensors. Therefore, when a malicious node persistently attacks, the sensors have the necessary information to detect the abnormal increase in sequence numbers. Thus, there is sufficient time to obtain recent snapshots of the network state. This explains the increasing trend for an increasing number of attacks. From the experiments, it was also determined that mobility does not seem to influence the performance of the sensors. Here again, this observation can be attributed to the exchange of information among sensors regarding nodes in the network. Thus, irrespective of the speed, the detection capabilities are similar, although an improvement is observed in the case of Figure 13(a).

False positives are produced from two sources - packet drops due to mobility that were falsely detected as drop attacks and the modification of RREP messages, in order to generate false sequence number attacks. Packet drops are detected by checking if the packets received by neighboring nodes are forwarded towards the destination. Since the sequence numbers of RREP packets are modified, even when the packet is forwarded, a false drop attack is detected because the packet is not the same. Mobility related packet drops cause false positives because the packet was not forwarded by a neighbor that incidentally had moved out of range.

A common trend observed is a steady decline in the percentage of false positives produced with an increase in the number of attacks as shown in Figure13(b). For smaller networks, since fewer than 500 attacks were produced, the false positive rate is very high compared to a 50 node network. This is because the false positive rate is a ratio of the number of false attacks detected to the number of attacks produced. When the number of attacks produced drastically increases for a significantly smaller increase in the false positives produced due to packet drops (as in the case for 50 nodes) the ratio decreases. For static topologies, false positives occur due to the modification of sequence numbers in RREP packets. Thus, an increase in the percentage of attacks detected for no mobility scenarios produces a corresponding increase in the percentage of false positives detected. However, the overall decline in the trend is still maintained. Finally, it was observed that mobility increases the average percentage of false positives owing to the higher number of packet drops that are falsely detected.

8. Conclusions

This paper surveys the various security issues and attacks possible in a wireless ad hoc network and motivates the need for an intrusion detection system. AODVSTAT is an intrusion detection tool that performs real-time detection of attacks in mobile ad hoc networks running the AODV routing protocol. Experimental results validate the ability of AODVSTAT to successfully detect both one-hop and distributed attacks against the AODV routing protocol, with a low number of false positives. Also, the tool imposes a small overhead on the network nodes, which is an important factor for resource-constrained equipment.

Acknowledgments

This research was supported by the Army Research Laboratory and the Army Research Office, under agreement DAAD19-01-1-0484, and by a AFOSR MURI grant.

References

[1] P. Albers, O. Camp, J.-M. Percher, B. Jouga, L. Me, and R. Puttini. Security in Ad Hoc Networks: a General Intrusion Detection Architecture Enhancing Trust Based Approaches. In *The 1^{st} International Workshop on Wireless Information Systems, Proceedings of the 4^{th} International Conference on Enterprise Information Systems*, pages 1–12, Ciudad Real, Spain, April 2002.

[2] J. Bellardo and J. Savage. 802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions. In *Proceedings of the USENIX Security Symposium*, pages 15–28, Washington DC, August 2003.

[3] S. Buchegger and J.-Y. L. Boudec. Performance Analysis of the CONFIDANT Protocol. In *Proceedings of the 3^{rd} ACM Symposium on Mobile Ad Hoc Networking and Computing*, pages 226–236, Lausanne, Switzerland, June 2002.

[4] I. D. Chakeres. AODV-UCSB Implementation from University of California Santa Barabara. http://moment.cs.ucsb.edu/AODV/aodv.html.

[5] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822, 1999.

[6] Y.-C. Hu, D. B. Johnson, and A. Perrig. Ariadne: A Secure On-Demand Routing Protocol for Ad hoc Networks. In *Proceedings of the 8^{th} ACM International Conference on Mobile Computing and Networking (Mobicom)*, pages 12–23, Atlanta, GA, September 2002.

[7] Y.-C. Hu, D. B. Johnson, and A. Perrig. Sead: Secure efficient distance vector routing in mobile wireless ad hoc networks. In *Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*, pages 3–13, Callicoon, NY, June 2002.

[8] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State Transition Analysis: A Rule-Based Intrusion Detection Approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, 1995.

[9] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks. In *Proceedings of the 9^{th} IEEE International Conference on Network Protocols*, pages 251–260, Riverside, CA, November 2001.

[10] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad hoc Networks. In *Proceedings of the 6^{th} International Conference on Mobile Computing and Networking*, pages 255–265, Boston, MA, August 2000.

[11] A. Mishra and W. Arbaugh. An Initial Security Analysis of the IEEE 802.1X Protocol. Technical Report, Department of Computer Science, University of Maryland, February 2002.

[12] C. E. Perkins and E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *Proceedings of the 2^{nd} IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.

[13] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. Belding-Royer. A Secure Protocol for Ad Hoc Networks. In *Proceedings of the 10^{th} IEEE International Conference on Network Protocols*, Paris, France, November 2002.

[14] M. Schiffman. libnet. `http://packetfactory.net/libnet/`, 2002.

[15] I. Stamouli. Real-time intrusion detection for ad hoc networks. Master's thesis, University of Dublin, September 2003.

[16] L. Venkatraman. Securing Routing Protocols for Ad Hoc Networks. Master's thesis, University of Cincinnati, November 2000.

[17] G. Vigna, F. Valeur, and R. Kemmerer. Designing and Implementing a Family of Intrusion Detection Systems. In *Proceedings of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2003)*, pages 88–97, Helsinki, Finland, September 2003.

[18] J. Wright. Detecting Wireless LAN MAC Address Spoofing. White Paper, January 2003.

[19] S. Yi, P. Naldurg, and R. Kravets. Security Aware Ad Hoc Routing for Wireless Networks. In *Proceedings of the 2^{nd} ACM Symposium on Mobile Ad Hoc Networking and Computing, Poster Session*, pages 299–302, Long Beach, California, October 2001.

[20] H. Yih-Chun and A.Perrig. A survey of secure wireless ad hoc routing. *IEEE Security & Privacy Magazine*, 2(3):28–39, May/June 2004.

[21] M. G. Zapata and N. Asokan. Securing Ad-Hoc Routing Protocols. In *Proceedings of the 2002 ACM Workshop on Wireless Security*, pages 1–10, Atlanta, GA, September 2002.

[22] Y. Zhang and W. Lee. Intrusion Detection in Wireless Ad-hoc Networks. In *Proceedings of the 6^{th} International Conference on Mobile Computing and Networking*, pages 275–283, Boston, MA, August 2000.

[23] Y. Zhang and W. Li. An integrated environment for testing mobile ad-hoc networks. In *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '02)*, Lausanne, Switzerland, June 2002.

[24] L. Zhou and Z. J. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine*, 13(6):24–30, 1999.