Attack Analysis and Detection for Ad Hoc Routing Protocols

Yi-an Huang and Wenke Lee

College of Computing
Georgia Institute of Technology
801 Atlantic Dr.
Atlanta, GA, USA 30332
{yian, wenke}@cc.gatech.edu

Abstract. Attack analysis is a challenging problem, especially in emerging environments where there are few known attack cases. One such new environment is the Mobile Ad hoc Network (MANET). In this paper, we present a systematic approach to analyze attacks. We introduce the concept of basic events. An attack can be decomposed into certain combinations of basic events. We then define a taxonomy of anomalous basic events by analyzing the basic security goals.

Attack analysis provides a basis for designing detection models. We use both specification-based and statistical-based approaches. First, normal basic events of the protocol can be modeled by an extended finite state automaton (EFSA) according to the protocol specifications. The EFSA can detect anomalous basic events that are direct violations of the specifications. Statistical learning algorithms, with statistical features, i.e., statistics on the states and transitions of the EFSA, can train an effective detection model to detect those anomalous basic events that are temporal and statistical in nature.

We use the AODV routing protocol as a case study to validate our research. Our experiments on the MobiEmu wireless emulation platform show that our specification-based and statistical-based models cover most of the anomalous basic events in our taxonomy.

Keywords: MANET, Attack Analysis, Intrusion Detection, Routing Security, AODV

1 Introduction

Network protocol design and implementation have become increasingly complex. Consequently, securing network protocols requires detailed analysis of normal protocol operations and vulnerabilities. The process is tedious and error-prone. Traditional attack analysis categorizes attacks based on knowledge of known incidents. Therefore, such analysis cannot be applied to new (unknown) attacks. The problem is even more serious in new environments where there are very few known attacks. Mobile ad hoc networking (MANET) is such an example. An ad hoc network consists of a group of autonomous mobile nodes with no infrastructure support. Recently, many MANET applications have emerged, such as battlefield operations, personal digital assistant (PDA) communication, among others. MANET and its applications are very different from traditional network and applications. They are also more vulnerable due to their unique characteristics, such as open physical medium, dynamic topology, de-centralized computing environment, and lack of a clear line of defense. Recent research efforts, such as [Zap01,HPJ02] attempt to apply cryptography techniques to secure MANET routing protocols. However, existing experience in wired security has already taught us the necessity of defense-in-depth because there are always human errors and design flaws that enable attackers to exploit software vulnerability. Therefore, it is also necessary to develop *detection* and *response* techniques for MANET.

Designing an effective intrusion detection system (IDS), as well as other security mechanisms, requires a deep understanding of threat models and adversaries' attack capabilities. We note that since MANET uses a TCP/IP stack, many well-known attacks can be applied to MANET but existing security measures in wired networks can address these attacks. On the other hand, some protocols, especially routing protocols, are MANET specific. Very few attack instances of these protocols have been well studied. It follows that traditional attack analysis cannot work effectively. In this paper, we propose a new attack analysis approach by decomposing a complicated attack into a number of basic components called *basic events*. Every basic event consists of casually related protocol behavior and uses resources solely within a single node. It is easier to study the protocol behavior more accurately from the point of view of a single node. Specifically, we study the basic routing behavior in MANET. We propose a taxonomy of anomalous basic events for MANET, which is based on potential targets that attackers can compromise and the security goals that attackers attempt to compromise for each target.

Based on the taxonomy, we build a prototype IDS for MANET routing protocols. We choose one of the most popular MANET routing protocols, AODV, as a case study. We develop specifications in the form of an extended finite state automaton (EFSA) from AODV IETF Draft [PBRD03] . We apply two detection approaches which use the EFSA in different ways. First, we can detect violations of the specification directly, which is often referred to as a **specification-based approach**. Second, we can also detect statistical anomalies by constructing statistical features from the specification and apply machine learning methods. This **statistical-based approach** is more suitable for attacks that are temporal and statistical in nature.

In short, our main contribution is the concept of basic events and its use in attack taxonomy analysis. We also show how to use protocol specifications to model normal basic events and derive features from the specification to design an intrusion detection system.

We use MobiEmu [ZL02] as our evaluation platform for related experiments. MobiEmu is an experimental testbed that emulates MANET in a wired network. It shows that our approach involves a much smaller set of features in order to capture the same set of attacks, compared with our previous work in developing

IDS for MANET [HFLY02] that attempted an exhaustive search of features without the help of taxonomy and protocol specification. As the feature set is smaller and derived directly from the protocol specification, it has the additional advantage that domain experts can review it. This further improves accuracy.

The rest of the paper is organized as follows. Section 2 discusses related concepts of basic events and presents a taxonomy of anomalous basic events in MANET. Section 3 presents an AODV EFSA specification. Section 4 describes the design of a MANET IDS, experiments and results. Finally, related work and conclusions are discussed in Sections 5 and 6.

2 Taxonomy of Anomalous Basic Events

2.1 Concepts

Anomalies or attacks can be categorized using different criteria. Since there is no well-established taxonomy yet in MANET, we describe a systematic approach to study MANET attacks based on the concept of **anomalous basic events**. We use MANET routing as the subject of our study.

A **routing process** in MANET involves causally related, cooperative operations from a number of nodes. For example, the *Route Discovery* process, frequently appeared in on-demand routing protocols [JMB01,PBRD03], consists of chained actions from the source node to the destination node (or an intermediate node who knows a route to the destination) and back to the source node. Such process can be decomposed into a series of basic routing events. A **basic routing event** is defined as an indivisible local segment of a routing process. More precisely, it is the smallest set of causally related routing operations on a single node. We will use the term **basic event** for short. Therefore, the *Route Discovery process* can be decomposed into the following basic events: 1) The source node delivers an initial *Route Request*; 2) Each node (except for the source node and the node that has a route to the destination) in the forward path receives a *Route Request* from the previous node and forwards it; 3) The replying node receives the *Route Request* and replies with a *Route Reply* message; 4) An intermediate node in the reverse path receives a *Route Reply* message and forwards it; 5) Finally, the source node receives the *Route Reply* message and establishes a route to the destination.

Note that a basic (routing) event may contain one or more operations, such as receiving a packet, modify a routing parameter, or delivering a packet. However, the integrity of routing logic requires every basic event be conducted in a transaction fashion. That is, it is considered successful (or normal) if and only if it performs all of its operations in the specified order. We assume that certain **system specification** exists which specifies normal protocol behavior. As we will show later in the paper, system specification can be represented in the form of an extended finite state machine; a (normal) basic event maps to a single transition in a given extended finite state machine. We further note that to define a basic event, operations are restricted to the scope of a single MANET node because only local data source can be fully trusted by the intrusion detection agent on the same node.

On the other hand, an **anomalous basic event** is a basic event that does not follow the system specification. Obviously, it is useful to study anomalous basic events in order to capture the characteristics of basic attack components. Nevertheless, we note that it is possible that some attacks do not trigger any anomalous basic events. For example, an attack may involve elements from a different layer that the system specification does not describe, or it may involve knowledge beyond a single node. A Wormhole attack [HPJ01] is an example of the first case, where two wireless nodes can create a hidden tunnel through wires or wireless links with stronger transmission power. A network scan on known (vulnerable) ports is an example of the latter case because each single node observes only legitimate uses. To deal with these issues, we plan to work on a multiple layer and global intrusion detection system.

2.2 Taxonomy of Anomalous Basic Events in MANET Routing Protocols

We identify an anomalous basic event by two components, its **target** and **operation**. A protocol agent running on a single node has different elements to operate on, with different semantics. The routing behavior of MANET typically involves three elements or targets: *routing messages*, *data packets* and *routing table (or routing cache) entries*. Furthermore, we need to study what are the possible attack operations on these targets. Individual security requirements can be identified by examining the following well-known security goals: *Confidentiality*, *Integrity* and *Availability*. We summarize possible combinations of routing targets and operations in Table 1. In this table, we list three basic operations for *Integrity* compromise: add, delete and change. The exact meanings of these operations need to be interpreted properly in the context of individual targets.

Conceptually, we can characterize a normal basic event in a similar way, i.e., its target and its operation type. Nevertheless, many different normal operations can be applied and it is hard to find a universal taxonomy of normal operations for all system specification. Thus, a more logical way is to represent normal basic events with a different structure, such as the extended state machine approach we introduce in Section 3.

In MANET routing security, cryptography addresses many problems, especially those involving confidentiality and integrity issues on data packets. Intrusion detection techniques are more suitable for other security requirements. *Availability* issue, for example, is difficult for protection techniques because attack packets appear indistinguishable from normal user packets. Some *integrity* problems also require non-cryptographic solutions for efficiency reasons. For example, an attacker can compromise the routing table in a local node and change the cost of any specific route entry. It may change the sequence number or a hop count so that some specific route appears more attractive than other valid routes. Encrypting every access operation on routing entries could be too expensive. Intrusion detection solutions can better address these issues, based on existing experience in the wired networks. We identify a number of anomalous basic events that are more suitable for intrusion detection systems in bold face in Table 1.

There are two types of anomalous basic events marked by asterisks in the table, *Fabrication of Routing Messages* and *Modification of Routing Messages*. There are cryptographic solutions for these types of problems, but they are not very efficient and sometimes require an expensive key establishment phase. We want to study them in our IDS work because they are related to the routing logic and we can see later that some attacks in these categories can be detected easily.

Table 1. Taxonomy of Anomalous Basic Events

Compromises to Security Goals		Events by Targets		
		Routing Messages	Data Packets	Routing Table Entries
Confidentiality		Location Disclosure	Data Disclosure	N/A
Integrity	Add	**Fabrication***	Fabrication	**Add Route**
	Delete	**Interruption**	**Interruption**	**Delete Route**
	Change	**Modification***	Modification	**Change Route Cost**
		Rushing		
Availability		**Flooding**	**Flooding**	Routing Table Overflow

We examine a number of basic MANET routing attacks noted in the literature [HFLY02,NS03,TBK$^+$03]. By comparing them (shown in Table 2) with taxonomy in Table 1, we find they match very well with the definitions of anomalous basic events. We refer to each attack with a unique name and optionally a suffix letter. For example, “Route Flooding (S)” is a flooding attack of routing messages that uses a unique source address.

In addition, we consider a number of more complex attack scenarios that contains a sequence of anomalous basic events. We use some examples studied by Ning and Sun [NS03]. These attack scenarios are summarized in Table 3.

As a case study, we analyze AODV [PBRD03], a popular MANET routing protocol. We analyze its designed behavior using an extended finite state automaton approach. This is inspired by the work on TCP/IP protocols in [SGF$^+$02].

3 A Specification of the AODV Protocol

3.1 An Overview of Extended Finite State Automaton (EFSA)

Specification-based approach provides a model to analyze attacks based on protocol specifications. Similar to the work by Sekar et al. for TCP/IP protocols [SGF$^+$02], we also propose to model the AODV protocol with an EFSA approach.

An *extended finite state automaton (EFSA)* is similar to a finite-state machine except that transitions and states can carry a finite set of parameters.

Table 2. Basic MANET Attacks, where suffix letters stand for different attack variations. R, S, D stand for randomness, source only and destination only, respectively. Other letters include M (maximal value), F (failure), Y (reply), I (invalid) and N (new).

Attacks	Attack Description	Corresponding Anomalous Basic Events
Active Reply	A *Route Reply* is actively forged with no related incoming *Route Request* messages.	Fabrication of Routing Messages
False Reply	A *Route Reply* is forged for a *Route Request* message even though the node is not supposed to reply.	
Route Drop (R)	Drop routing packets. (R) denotes a random selection of source and destination addresses.	Interruption of Routing Messages
Route Drop (S)	A fixed percentage of routing packets with a specific source address are dropped. (S) stands for source address.	
Route Drop (D)	A fixed percentage of routing packets with a specific destination address are dropped. (D) stands for destination address.	
Modify Sequence (R)	Modify the destination's sequence number randomly. (R) stands for randomness.	Modification of Routing Messages
Modify Sequence (M)	Increase the destination's sequence number to the largest allowed number. (M) stands for the maximal value.	
Modify Hop	Change the hop count to a smaller value.	
Rushing (F)	Shorten the waiting time for *Route Replies* when a route is unavailable. (F) stands for failure.	Rushing of Routing Messages
Rushing (Y)	Shorten the waiting time to send a *Route Reply* after a *Route Request* is received. (Y) stands for reply.	
Route Flooding (R)	Flood with both source and destination addresses randomized.	Flooding of Routing Messages
Route Flooding (S)	Flood with the same source address and random destination addresses.	
Route Flooding (D)	Flood to a single destination with random source addresses.	
Data Drop (R | S | D)	Similar to Route Drop (R), Route Drop (S), or Route Drop (D), but using data packets.	Interruption of Data Packets
Data Flooding (R | S | D)	Similar to Route Flooding (R), Route Flooding (S), or Route Flooding (D), but using data packets.	Flooding of Data Packets
Add Route (I)	An invalid route entry is randomly selected and validated. (I) stands for invalid.	Add Route of Routing Table Entries
Add Route (N)	A route entry is added directly with random destination address. (N) stands for new.	
Delete Route	A random valid route is invalidated.	Delete Route of Routing Table Entries
Change Sequence (R | M)	Similar to Modify Sequence attacks but the sequence number is changed directly on the routing table.	Change Route Cost of Routing Table Entries
Change Hop	Similar to Modify Hop, but the hop count is changed directly on the routing table.	
Overflow Table	Add excessive routes to overflow the routing table.	Routing Table Overflow of Routing Table Entries

Conventionally, we call them transition *parameters* and state *variables*. We can derive EFSA from documentation, implementations, RFCs or other materials.

Furthermore, we distinguish two types of transitions: input and output transitions. Input transitions include packet-receiving events and output transitions

Table 3. More Complex MANET Attacks

Attacks	Attack Description	Corresponding Anomalous Basic Events
Route Invasion	Inject a node in an active route.	Fabrication of Routing Messages (two RREQs)
Route Loop	Create a route loop.	Fabrication of Routing Messages (two RREPs)
Partition	Separate a network into two partitions.	Fabrication of Routing Messages (RREP) Interruption of Data Packets

include packet-delivery events. If there are no packet communication events involved in a transition (which can take place with a timeout, for example), it is also treated as an input transition.

According to the original definition in [SGF$^+$02], input and output transitions are separate transitions because only one event can be specified in a transition. Here we relax the definition of a transition by allowing a transition to have both a packet-receiving event and a packet-delivery event (either of them can still be optional). The relaxed definition of a transition δ is: $\delta = \{$S_old $\rightarrow$ S_new, input_cond $\rightarrow$ output_action$\}$, where the old and new states are specified in S_old and S_new. The new definition assumes the following semantics. The output action, if defined, must be performed immediately after the input condition is met, and before the new state is reached. Unless the output action has accomplished, no other transitions are allowed.

An **input condition** (input_cond) can specify timeouts or predicates and at most one packet-receiving event. It uses a C-like expression syntax where operators like &&, || etc., can be used. State variables (of the original state) and transition parameters can be accessed in input conditions. To distinguish, state variables start with lower case letters and transition parameters start with capitalized letters. **Packet-receiving events**, predicates and **timeouts** can be used as Boolean functions in input conditions. A packet-receiving event or a predicate has its own parameters, which must be matched with provided values, unless the value is a dash (-), which specifies that the corresponding parameter can match any value. An **output action** (output_action) can specify state variable modifications, tasks and at most one packet-delivery event. Predicates and tasks refer to functionalities that we plan to implement later. An output action is a list of operations, which can be **packet-delivery event**s, **state variable assignments**, or tasks. Either input_cond or output_action can be optional but at least one must be present.

In addition, a number of **auxiliary functions** can be used in either input conditions or output actions. They are actually evaluated by IDS. We use auxiliary functions simply to improve readability.

Protocol state machines are in general non-deterministic, as one incoming packet can lead to multiple states. We solve non-determinism by introducing a set of finite state automata, which start from the same state, but fork into different paths when a state can have multiple transitions based on an incoming event. For instance, in TCP, every extended finite state automaton corresponds to the state of a unique connection. In AODV, operations on a particular route entry to a single destination can be defined in an extended finite state automaton. For example, an incoming *Route Reply* message can add new routes to both the destination node and the previous hop node, thus the EFSAs for both nodes need to process this message. In addition, the *Route Reply* message may also be forwarded to the originator, which is conducted by a third EFSA corresponding to the originator.

Clearly, the number of state machines can increase up to the number of possible nodes in the system if their lifetime is unbounded. Thus, we should remove unnecessary state machines to reduce memory usage. In AODV, a route entry is removed after it has been invalidated for a certain period. In other words, we can identify a final state from which no further progress could be made. Therefore, state machines reaching the final state can be deleted from the state machine repository safely.

We construct an AODV EFSA by following the AODV Internet draft version 13 [PBRD03]. AODV uses hop-by-hop routing similar to distant vector based protocols such as RIP [Mal94]. Nevertheless, there are no periodical route advertisements. Instead, a route is created only if it is demanded by data traffic with no available routes [PBRD03].

3.2 The AODV EFSA Specification

Our AODV EFSA is based on the AODV state machine from Bhargavan et al.'s work [BGK$^+$02]. It is shown in Figures 1 and 2.

Each EFSA contains two sub graphs. The second sub graph (Figure 2) is only in use within a certain period after a node has rebooted. After all other nodes have updated their routing entries accordingly, normal routing operations resume and the other graph (the normal sub graph, Figure 1) is used. The two sub graphs are shown separately for a better layout.

Note that we only capture major AODV functionalities in the EFSA. Some specified protocol behavior relies on information from other layers, which we cannot model for now.

The routing behavior in AODV is defined for every single route entry or destination. In other words, there is a unique EFSA for each destination host. We use the abbreviation *ob* to specify the destination, which stands for an **ob**served node. We define EFSA(*ob*) as the corresponding EFSA of *ob*. In addition, there is a special EFSA, EFSA(**cur**), where **cur** is a global variable that defines the node's IP address. We create this special EFSA specifically to reply *Route Requests* for the current node. Thus, for each node, we have a total of $n + 1$ EFSAs where n is the number of entries in the node's routing table. That is, n instances of EFSA(*ob*), one for each destination, and one instance of EFSA(**cur**).

Timeouts, predicates, packet-receiving events, packet-delivery events, tasks and auxiliary functions are further explained below. Note that a predicate or a packet-receiving event ends with '?', while a packet-delivery event ends with '!'.

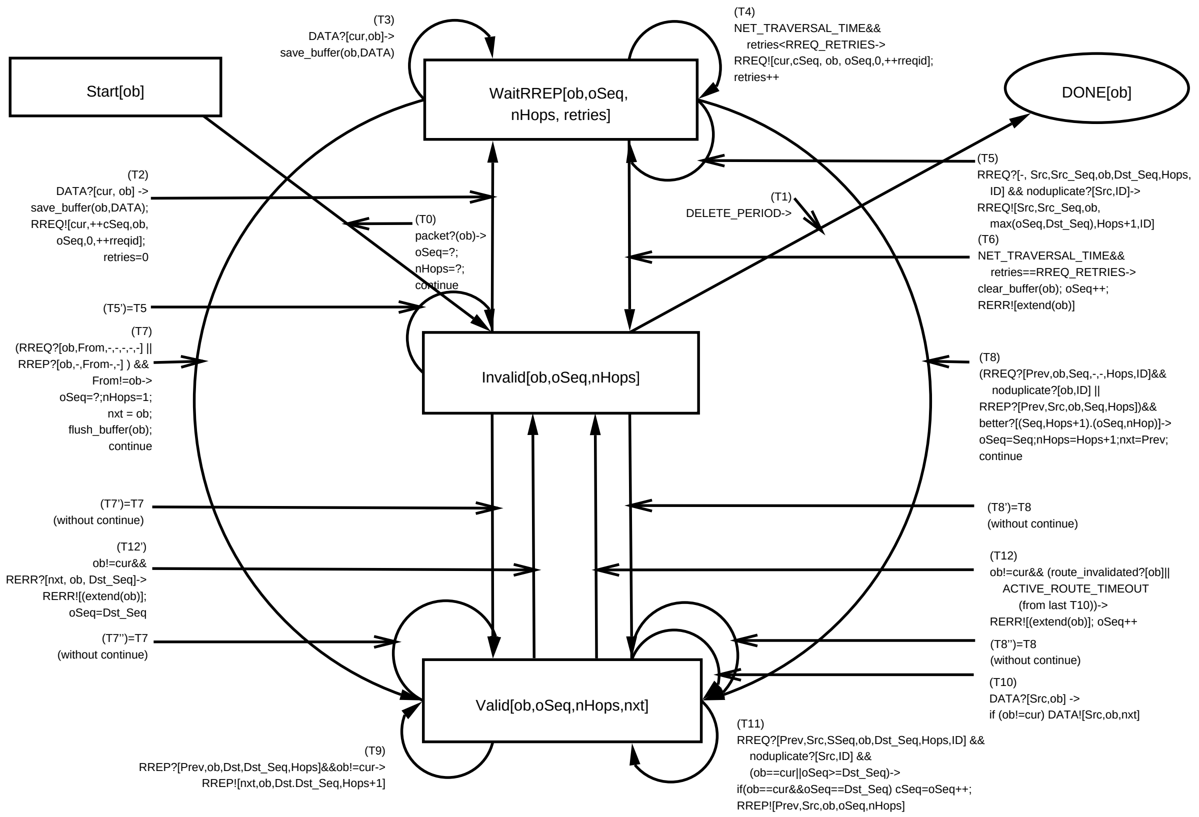

Fig. 1. AODV Extended Finite State machine (*ob*): In Normal Use

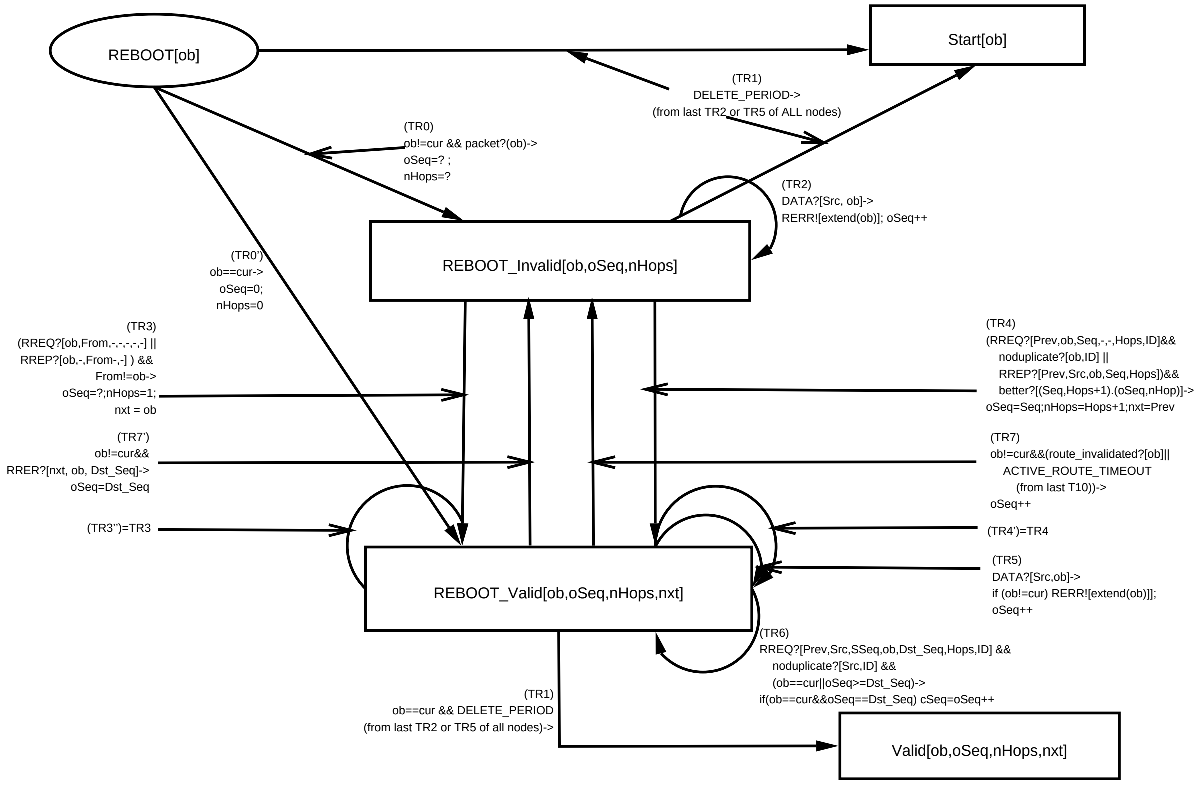
REBOOT[ob]
Start[ob]
(TR1)
DELETE_PERIOD->
(from last TR2 or TR5 of ALL nodes)
(TR0)
ob!=cur && packet?(ob)->
oSeq=? ;
nHops=?
(TR2)
DATA?[Src, ob]->
RERR![extend(ob)]; oSeq++
REBOOT_Invalid[ob,oSeq,nHops]
(TR0')
ob==cur->
oSeq=0;
nHops=0
(TR3)
(RREQ?[ob,From,-,-,-,-,-] ||
RREP?[ob,-,From-,-]) &&
From!=ob->
oSeq=?;nHops=1;
nxt = ob
(TR4)
(RREQ?[Prev,ob,Seq,-,-,Hops,ID]&&
noduplicate?[ob,ID] ||
RREP?[Prev,Src,ob,Seq,Hops])&&
better?[(Seq,Hops+1).(oSeq,nHop)]->
oSeq=Seq;nHops=Hops+1;nxt=Prev
(TR7')
ob!=cur&&
RRER?[nxt, ob, Dst_Seq]->
oSeq=Dst_Seq
(TR7)
ob!=cur&&(route_invalidated?[ob]||
ACTIVE_ROUTE_TIMEOUT
(from last T10))->
oSeq++
(TR3'')=TR3
(TR4')=TR4
(TR5)
DATA?[Src,ob]->
if (ob!=cur) RERR![extend(ob)]];
oSeq++
REBOOT_Valid[ob,oSeq,nHops,nxt]
(TR6)
RREQ?[Prev,Src,SSeq,ob,Dst_Seq,Hops,ID] &&
noduplicate?[Src,ID] &&
(ob==cur||oSeq>=Dst_Seq)->
if(ob==cur&&oSeq==Dst_Seq) cSeq=oSeq++
(TR1)
ob==cur && DELETE_PERIOD
(from last TR2 or TR5 of all nodes)->
Valid[ob,oSeq,nHops,nxt]

Fig. 2. AODV Extended Finite State machine (*ob*): After Reboot

- Timeouts:
  1. **DELETE_PERIOD**: specify how long an invalidated route should remain in memory.
  2. **ACTIVE_ROUTE_TIMEOUT**: specify how long before a valid route should be invalidated due to inactivity.
  3. **NET_TRAVERSAL_TIME**: specify the maximal round trip time after a RREQ has been sent and before the corresponding RREP is received.
- Predicates (the expected behavior):
  1. **noduplicate?[Src, ID]**: return true if RREQ from *Src* with RREQ ID is not seen before. The pair is then cached and can be used for comparison in later calls.
  2. **route_invalidated?[Dst]**: return true if a route to *Dst* has been invalidated due to link loss or incoming RERR, etc.
- Packet-receiving events:
  1. **DATA?[Src, Dst]**: return true if there is an incoming data packet that was originated from *Src*, and is destined to *Dst*.
  2. **RREQ?[Prev, Src, Src_Seq, Dst, Dst_Seq, Hops, ID]**: return true if a RREQ message has been received and it contains the following fields. The originator is *Src* with sequence number *Src_Seq* and a unique RREQ *ID*. The destination is *Dst* with sequence number *Dst_Seq*. The number of hops from *Src* is *Hops*. Finally, the *Prev* field specifies the address of the previous hop. Although not shown in the outgoing RREQ! event, this field can be found in the incoming packet's IP header.
  3. **RREP?[Prev, Src, Dst, Dst_Seq, Hops]**: return true if there is an incoming RREP and named fields match the specified parameters (similar to RREQ?, except that *Hops* here represents the hops to *Dst*).
  4. **RERR?[Src, Dst, Dst_Seq]**: return true if an incoming RERR message was sent by *Src*, and includes *Dst* in its unreachable destination list, with sequence number *Dst_Seq*.
- Packet-delivery events:
  1. **DATA![Src, Dst, Next]**: forward a data packet that was originated from *Src* and is destined to *Dst*, to the next hop *Next*.
  2. **RREQ![Src, Src_Seq, Dst, Dst_Seq, Hops, ID]**: broadcast RREQ with supplied fields.
  3. **RREP![Next, Src, Dst, Dst_Seq, Hops]**: deliver RREP. We explicitly specify *Next* here since RREP, different from RREQ, is not broadcast.
  4. **RERR![Dsts]**: deliver RERR with the list of unreachable destinations in *Dsts*. Corresponding sequence numbers of these destinations are also included in *Dsts*.
- Tasks (the expected behavior):
  1. **save_buffer(Dst, DATA)**: buffer the data with destination *Dst*.
  2. **flush_buffer(Dst, Next)**: deliver all packets in the data buffer with destination *Dst* through *Next*, and removes them from the data buffer.

   3. **clear_buffer(Dst)**: remove all data with destination *Dst* from the data buffer.
- Auxiliary functions:
   1. **packet?(Dst)**: return true if there is an incoming packet destined to *Dst*. It is a shorthand of DATA?[-,Dst] || RREQ?[-,-,-,Dst,-,-,-] || RREP?[-,-,Dst,-,-] || RERR?[Dsts] && Dst $\in$ Dsts.
   2. **better?([seq1, hop1],[seq2, hop2])**: return true if (seq1 $>$ seq2 || seq1==seq2 && hop1 $<$ hop2 || seq2 is unknown).
   3. **extend(Dst)**: return a list of unreachable destinations (with their sequence numbers) due to a broken link to *Dst*. Routes to these destinations include *Dst* as their next hop. Obviously, $Dst \in extend(Dst)$.
   4. **continue**: do not stop in the new state after a transition. Instead, attempt to make another state transition from the new state.

4 Design of an Intrusion Detection System for AODV

Before we analyze design issues of an Intrusion Detection System (IDS) for AODV, we make the following assumptions: 1) IDS should have access to internal routing elements, such as routing table entries. Currently, we modify the AODV implementation in our testbed to store routing table entries in a shared memory block, so that other processes can access them. In the future, hardware assistance may be necessary to achieve this; 2) IDS should also have the capability of intercepting incoming and outgoing packets, including data and routing messages.

Statistical-based detection technique, equipped with machine learning tools, can be used to detect abnormal patterns. It has the potential advantage of detecting unknown attacks. But it usually comes with a high false alarm rate. Its detection performance heavily depends on selected features.

In contrast, specification-based techniques use specifications to model legitimate system behavior and do not produce false alarms. However, developing specification is time consuming. Furthermore, many complex attacks do not violate the specification directly and cannot be detected using this approach.

Our detection approach combines the advantages of both techniques. Consequently, we separate anomalous basic events into two sets, events that directly violate the semantics of EFSAs, and events that require statistical measures.

4.1 Detection of Specification Violations

Some anomalous basic events can be directly translated into violations of EFSAs. We identify three types of violations: **Invalid State Violation**, **Incorrect Transition Violation** and **Unexpected Action Violation**.

Invalid State Violation involves a state that does not appear to be valid in the specification. In our specification, an invalid state means the combination of state variables in the current state is invalid according to the specification. For example, a state with a negative hop count is considered an invalid state. In our implementation, we keep a copy of state variables every five seconds. Thus, we can track invalid changes in state variables.

Incorrect Transition Violations occur if invalid transitions are detected. We verify the proper transition by comparing possible input conditions on all transitions from the current state. If a state change occurs while no input conditions can be met, this type of violation is detected. In addition, there are self-looping transitions that do not change the current state. For these transitions, we examine output actions. If some of these output actions (which include packet delivery events and state variable modifications) are detected while corresponding input conditions do not match, we also identify this type of violation. Our implementation monitors incoming and outgoing traffic to determine if input conditions and output actions are properly handled.

Unexpected Action Violation corresponds to the situation when the input condition during a transition matches and the new state is as expected, but the output action is not correctly or fully performed.

We show that the specification-based approach can detect the following anomalous basic events:

Interruption of Data Packets: We monitor the transition T10, where data is forwarded when a valid route is available. An attacker interrupts data packets by receiving but not forwarding data. It is observed as a type of *Unexpected Action Violation* in the transition.

Interruption of Routing Messages: An attacker may choose to interrupt certain types of routing messages by conducting the corresponding transition but not actually sending the routing packets. For more details, *Route Request* messages are delivered in transition T4, T5, or T5'; *Route Reply* messages are delivered in T9 or T11; *Route Error* messages are delivered in TR2, TR5, T6, T12 or T12'. They can always be identified as *Unexpected Action Violations* in the corresponding transition.

Add Route of Routing Table Entries: We monitor state change to the state when a route to *ob* becomes available (state **Valid**) from other states. If it does not go through legitimate transitions (which include T7, T8, T7' and T8'), it implies that a new route is created bypassing the normal route creation path. It is an *Incorrect Transition Violation* in these transitions.

Delete Route of Routing Table Entries: Similarly, we monitor state change in a reverse direction, i.e., from a valid state (state **Valid**) to a state when a route becomes unavailable (state **Invalid**). If it does not go through legitimate transitions (T12 and T12'), it is detected as an *Incorrect Transition Violation* of these transitions.

Change Route Cost of Routing Table Entries: We can identify changes in sequence numbers or hop counts to the routing table using the memorized state variable copy, when a valid route is available (state **Valid**). They are examples of *Invalid State Violations.*

Fabrication of Routing Messages: Currently, our approach can identify a special type of *Fabrication of Routing Messages*, namely, *Route Reply Fabrication.* We examine the transitions that deliver *Route Reply* messages (transitions

T9 and T11). If the output actions are found but the input conditions do not match, we will identify an *Incorrect Transition Violation* in these transitions, which is an indication that outgoing routing messages are in fact fabricated.

To summarize, we define a violation detection matrix. It maps violation information (the violated transition(s) or state and the violation type) to an anomalous basic event. The matrix is shown in Table 4. It can be used to detect attacks that directly violate the AODV specification where we can identify the corresponding types of anomalous basic events. Detection results are summarized in Section 4.3.

Table 4. Violation Detection Matrix in AODV

State or Transition(s)	Invalid State Violation	Incorrect Transition Violation	Unexpected Action Violation
TR2, TR5, T6			Interruption of Route Errors
T4, T5, T5'			Interruption of Route Requests
T7, T8, T7', T8'		Add Route	
T9, T11		Fabrication of Route Replies	Interruption of Route Replies
T10			Interruption of Data Packets
T12, T12'		Delete Route	Interruption of Route Errors
Valid	Change Route Cost		

4.2 Detection of Statistical Deviations

For anomalous events that are temporal and statistical in nature, statistical features can be constructed and applied to build a machine learning model that distinguishes normal and anomalous events. RIPPER [Coh95], a well-known rule based classifier, is used in our experiments.

We first determine a set of statistical features based on activities from anomalous basic events that cannot be effectively detected using the specification-based approach. Features are computed periodically based on the specified statistics from **all** running EFSAs, and stored in audit logs for further inspection. To build a detection model, we use a number of off-line audit logs (known as training data) which contain attacks matching these anomalous basic events. Furthermore, each record is pre-labeled with the type of the corresponding anomalous basic event (or normal if the record is not associated with any attacks) because we know which attacks are used. They are processed by RIPPER and a detection model is generated. The model is a set of detection rules. The model is then used to detect attacks in the test data.

Using the taxonomy of anomalous basic events in Table 1, we identify the following anomalous basic events that remain to be addressed, because they cannot be detected in the specification-based approach. For each type of anomalous basic event, we discuss what features are needed to capture its behavior. All features are defined within a sampling window. We use a sampling window of five seconds in all cases. In addition, features are normalized in a scale of 0 to 50.

Flooding of Data Packets: In order to capture this anomalous event, we need to capture the volume of incoming data packets. In AODV, data packets can be accepted under three different situations: when a valid route is available (which is transition T10), when a route is unavailable and no route request has been sent yet (transition T2) or when a route is unavailable and a route request has been sent to solicit a route for the destination (transition T3). Accordingly, we should monitor frequencies of all these data packet receiving transitions. We define three statistical features, *Data1*, *Data2*, and *Data3*, for each transition (T10, T2 and T3) respectively.

Flooding of Routing Messages: Similarly, we need to monitor the frequencies of transitions where routing messages are received. However, a larger set of transitions need to be observed because we need to take into account of every type of routing messages (which include 15 transitions, T5, T5', T7, T8, T7', T8', T7", T8", T9, T11, TR3, TR4, TR3', TR4', and TR6). In order not to introduce too many features, we use an aggregated feature *Routing* which denotes the frequency of all these transitions. Note that it is not the same as monitoring the rate of incoming routing messages. An incoming routing message may not be processed by any EFSA in a node. We need only to consider messages that are being processed.

Modification of Routing Messages: Currently, we consider only modifications to the sequence number field. We define *Seq* as the highest destination sequence number in routing messages during transitions where they are received (see above for the transitions involved in routing messages).

Rushing of Routing Messages: We monitor two features where some typical routing process may be rushed. *Rushing1* is the frequency of the transition where a route discovery process fails because the number of *Route Requests* sent has exceeded a threshold (RREQ_RETRIES) or certain timeout has elapsed (NET_TRAVERSAL_TIME in transition T6). *Rushing2* is the frequency of the transition where a *Route Request* message was received and it is replied by delivering a *Route Reply* message (transition T11).

4.3 Experiments and Results

Environment: We use MobiEmu [ZL02] as the evaluation platform. MobiEmu is an experimental testbed that emulates MANET environment with a local wired network. Mobile topology is emulated through Linux's packet filtering mechanism. Different from many simulation tools, MobiEmu provides a scalable application-level emulation platform, which is critical for us to evaluate the intrusion detection framework efficiently on a reasonably large network. We use the AODV-UIUC implementation [KZG03], which is designed specifically to work with the MobiEmu platform.

Experiment Parameters: The following parameters are used throughout our experiments. Mobility scenarios are generated using a random way-point model with 50 nodes moving in an area of 1000m by 1000m. The pause time between movements is 10s and the maximum movement speed is 20.0m/s. Randomized TCP and UDP/CBR (Constant Bit Rate) traffic are used but the maximum number of connections is set to 20; and the average traffic rate is 4 packets per second. These parameters define a typical MANET scenario with modest traffic load and mobility. They are similar to the parameters used in other MANET experiments, such as [PRDM01,MBJJ99,MGLB00]. Nevertheless, we have not systematically explored all possible scenarios (for instance, with high mobility or under high traffic load). We plan to address this issue in our future work.

We test our framework with multiple independent runs. A normal run contains only normal background traffic. An attack run, in addition, contains multiple attack instances which are randomly generated from attacks specified in Tables 2 and 3 or a subset according to certain criteria.

We use ten attack runs and two normal runs as the test data, each of which runs 100,000 seconds (or 20,000 records since we use a sampling window of five seconds). In each attack run, different types of attacks are generated randomly with equal probability. Attack instances are also generated with random time lengths, but we guarantee that 80% of total records are normal. It is a relatively practical setting considering that normal events should be the majority in a real network environment. We use normal data in normal runs and attack runs to evaluate false alarm rates.

Detection of Specification Violations: The following attacks are detected in the test data as direct violations of the EFSA, which verifies our previous analysis that these attacks match anomalous basic events that can be directly detected by verifying the specification. For complex attacks, a different network size may be used if appropriate. Note that detection rates are 100% and false alarm rates are 0% for attacks when the specification-based approach is used.

Data Drop (R | S | D): detected as *Interruption of Data Packets.*

Route Drop (R | S | D): detected as *Interruption of Routing Messages.*

Add Route (I | N): detected as *Add Route of Routing Table Entries.*

Delete Route: detected as *Delete Route of Routing Table Entries.*

Change Sequence (R | M); Change Hop: detected as *Change Route Cost of Routing Table Entries.*

Active Reply; False Reply: detected as *Route Reply Fabrication.*

Route Invasion; Route Loop: They are detected since they use fabricated routing messages similar to what the **Active Reply** attack does. In particular, *Route Invasion* uses *Route Request* messages, and *Route Loop* uses *Route Reply* messages. With the same set of transitions in **Route Drop**, we can detect them as *Incorrect Transition Violations* in *Route Request* or *Route Reply* delivery transitions.

Partition: This attack can be detected since it uses a fabricated routing message (*Route Reply*) and interrupts data packets. Therefore, monitoring the transitions related to *Route Reply* (as in **Route Drop**), and the transition related to data packet forwarding (T10, as described in **Data Drop**), we can detect this attack with the following violations identified: *Incorrect Transition Violation* in *Route Reply* delivery transitions and *Unexpected Action Violation* in the data forwarding transition.

Detection of Statistical Deviations: Some attacks are temporal and statistical in nature and should be detected using the statistical approach. The following are four representative examples of such attacks: **Data Flooding (S | D | R)**; **Route Flooding (S | D | R)**; **Modify sequence (R | M)**; **Rushing (F | Y)**.

Four attacks data sets, each of which contains an attack run of 25,000 seconds (or 5,000 records), are used to train the detection model. Each data set contains attacks that match to one type of anomalous basic event. Attack instances are generated in such a way that the number of abnormal records accounts for roughly 50% of total records, instead of 80% in the case of test data. It helps improve detection accuracy by using approximately the same amount of normal and abnormal data. We train separately with each training data set. The same test data set is used to evaluate the learned model.

Table 5. Detection and False Alarm Rates of the Statistical-based Approach

(a) Attack Detection Rates

Attack	Detection rate
Data Flooding (S)	93±3%
Data Flooding (D)	91±4%
Data Flooding (R)	92±4%
Route Flooding (S)	89±3%
Route Flooding (D)	91±2%
Route Flooding (R)	89±3%
Modify sequence (R)	59±19%
Modify sequence (M)	100±0%
Rushing (F)	91±3%
Rushing (Y)	85±4%

(b) Detection and False Alarm Rates of Anomalous Basic Events

Anomalous Basic Event	Detection Rate	False Alarm Rate
Flooding of Data Packets	92±3%	5±1%
Flooding of Routing Messages	91±3%	9±4%
Modification of Routing Messages	79±10%	32±8%
Rushing of Routing Messages	88±4%	14±2%

The detailed detection results are shown in Table 5. We show the detection rates of tested attacks (in Table 5(a)). We consider a successful detection of an attack record if and only if the corresponding anomalous basic event is correctly identified. We also show the detection and false alarm rates (in Table 5(b)) directly against anomalous basic events. We analyze these results for each type of anomalous basic event below.

Flooding of Data Packets and Routing Messages: We implement flooding as traffic over 20 packets per second. For flooding of data packets, 92% can be detected. They are detected by observing abnormally high volume on at least one of related statistics, *Data1*, *Data2*, or *Data3*. Similar results are also observed for flooding of routing messages.

Modification of Routing Messages: The corresponding detection result is not very satisfactory. It shows high variations in both the detection and false alarm rates. In fact, the corresponding detection rule assumes that this anomalous basic event can be predicted when at least some incoming packet has a sequence number larger than certain threshold. It is not a rule that can be generally applied. Randomly generated sequence numbers may only be partially detected as attacks. We further discuss problem in the end of this section. In contrast, for a special type of sequence modification (*Modify Sequence (M)*), the detection rate is perfect. Because we know that it is very rare for the largest sequence number to appear in the sequence number field of routing messages.

Rushing of Routing Messages: Detection performance varies significantly on different rushing attacks, namely, *Rushing (F)* and *Rushing (Y)*. In *Rushing (F)*, the attacker tries to shorten the waiting time for a *Route Reply* message even if a route is not available yet. Because more requests to the same destination may follow if route discovery was prematurely interrupted, the attack results in abnormally high frequency where the route discovery process is terminated (*Rushing1*). In *Rushing (Y)*, the attacker expedites *Route Reply* delivery when a *Route Request* message has been received. It can be captured because the corresponding transition (T11) now occurs more frequently than a computed threshold (*Rushing2*). Nevertheless, we also observe significant false alarms in detecting these attacks. It results from irregularity of route topology change due to MANET's dynamic nature. Some normal nodes may temporarily suffer a high route request volume that exceeds these thresholds.

Discussion: Comparing with the taxonomy of anomalous basic events in Table 1, we realize that a few of them cannot be detected effectively yet. First, we cannot detect **Route Message Modification with incoming packets** in which the modification patterns are not known in advance. We identify the problem as it requires knowledge beyond a local node. However, these attacks can usually be detected using other security mechanisms or by other nodes. If the message comes from external sources, it may be successfully prevented by a cryptographic authentication scheme. Otherwise (i.e., it was delivered by the routing agent from another legitimate node), the IDS agent running on that node may have detected the attack. In addition, **Rushing** attacks cannot be detected very effectively, especially when features beyond the routing protocol, such as delays in the MAC layer, are involved. Our system can be improved if we were able to extend our detection architecture across multiple network layers. It is part of our future work.

5 Related Work

Many cryptographic schemes have been proposed to secure ad hoc routing protocols. Zapata [Zap01] proposed a secure AODV protocol using asymmetric cryptography. Hu et al. [HPJ02] proposed an alternative authentication scheme based on symmetric keys to secure the DSR protocol [JMB01], because public key computation appears too expensive for MANET nodes with limited power and computation capabilities. As we have demonstrated, protection approaches are suitable for a certain class of security problems. Intrusion detection approaches may be more suitable to address other problems.

Vigna and Kemmerer [VK98] proposed a misuse intrusion detection system, NetSTAT, which extends the original state transition analysis technique (STAT) [IKP95]. It models an attack as a sequence of states and transitions in a finite state machine. Whereas in our work, finite state machines are modeled for normal events. Specification-based intrusion detection was proposed by Ko et al. [KRL97] and Sekar et al. [SGF$^+$02]. Specification-based approaches reduce false alarms by using manually developed specifications. Nevertheless, many attacks do not directly violate specifications and thus, specification-based approaches cannot detect them effectively. In our work, we apply both specification-based and statistical-based approaches to provide better detection accuracy and performance.

Bhargavan et al. [BGK$^+$02] analyzed simulations of AODV protocols. Their work included a prototype AODV state machine. Our AODV EFSA is based on their work but has been heavily extended. Ning and Sun [NS03] also studied the AODV protocol and used the definition of atomic misuses, which is similar to our definition of basic events. However, our definition is more general because we have a systematic study of taxonomy of anomalous basic events in MANET routing protocols.

Recently, Tseng et al. [TBK$^+$03] proposed a different specification-based detection approach. They assume the availability of a cooperative network monitor architecture, which can verify routing request-reply flows and identify many attacks. Nevertheless, there are security issues as well in the network monitor architecture which were not clearly addressed.

6 Conclusion and Future Work

We proposed a new systematic approach to categorize attacks. Our approach decomposes an attack into a number of basic events. We showed its use in attack taxonomy analysis. In addition, protocol specifications can be used to model normal protocol behavior and can be used by intrusion detection systems. By applying both specification-based and statistical-based detection approaches, we have the advantages of both. Specification-based approach has no false alarm, statistical-based approach can detect attacks that are statistical or temporal in nature.

We proposed a taxonomy of anomalous basic events in MANET routing protocols and presented a case study of the AODV protocol. We constructed an AODV extended finite state automaton specification. By examining direct violations of the specification, and by constructing statistical features from the specification and applying machine learning tools, we showed that most anomalous basic events were detected in our experiments.

Future Work: We plan to enhance our framework by automatically extracting useful features for detection of unknown attacks. We also plan to design an intrusion detection system across multiple network layers to detect more sophisticated attacks.

7 Acknowledgment

This work is supported in part by NSF grants CCR-0133629 and CCR-0311024 and Army Research Office contract DAAD19-01-1-0610. The contents of this work are solely the responsibility of the authors and do not necessarily represent the official views of NSF and the U.S. Army.

References

[BGK$^+$02] K. Bhargavan, C. A. Gunter, M. Kim, I. Lee, D. Obradovic, O. Sokolsky, and M. Viswanathan. Verisim: Formal analysis of network simulations. *IEEE Transactions on Software Engineering*, 2002.

[Coh95] W. W. Cohen. Fast effective rule induction. In *Proceedings of the International Conference on Machine Learning*, pages 115–123, 1995.

[HFLY02] Y. Huang, W. Fan, W. Lee, and P. S. Yu. Cross-feature analysis for detecting ad-hoc routing anomalies. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, May 2002.

[HPJ01] Y.-C. Hu, A. Perrig, and D. B. Johnson. Wormhole detection in wireless ad hoc networks. Technical Report TR01-384, Department of Computer Science, Rice University, December 2001.

[HPJ02] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom'02)*, September 2002.

[IKP95] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. *Software Engineering*, 21(3):181–199, 1995.

[JMB01] D. B. Johnson, D. A. Maltz, and J. Broch. DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In C. E. Perkins, editor, *Ad Hoc Networking*, chapter 5, pages 139–172. Addison-Wesley, 2001.

[KRL97] C. Ko, M. Ruschitzka, and K. N. Levitt. Execution monitoring of security-critical programs in distributed systems: A specification-based approach. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 134–144, 1997.

[KZG03] V. Kawadia, Y. Zhang, and B. Gupta. System services for ad-hoc routing: Architecture, implementation and experiences. In *First International Conference on Mobile Systems, Applications, and Services (MobiSys'03)*, San Francisco, CA, May 2003.

[Mal94] G. Malkin. RIP version 2 - carrying additional information. RFC 1723, Internet Engineering Task Force, November 1994.

[MBJJ99] D. A. Maltz, J. Broch, J. G. Jetcheva, and D. B. Johnson. The effects of on-demand behavior in routing protocols for multi-hop wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, August 1999.

[MGLB00] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[NS03] P. Ning and K. Sun. How to misuse AODV: A case study of insider attacks against mobile ad-hoc routing protocols. In *Proceedings of the 4th Annual IEEE Information Assurance Workshop*, pages 60–67, June 2003.

[PBRD03] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (AODV) routing. Internet draft draft-ietf-manet-aodv-13.txt, Internet Engineering Task Force, February 2003. expired 2003.

[PRDM01] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE Personal Communications Magazine special issue on Ad hoc Networking*, pages 16–28, February 2001.

[SGF$^+$02] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification-based anomaly detection: A new approach for detecting network intrusions. In *Proceedings of the ACM Computer and Communication Security Conference (CCS'02)*, 2002.

[TBK$^+$03] C.-Y. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe, and K. N. Levitt. A specification-based intrusion detection system for AODV. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03)*, George W. Johnson Center at George Mason University, Fairfax, VA, October 2003.

[VK98] G. Vigna and R. A. Kemmerer. NetSTAT: A network-based intrusion detection approach. In *Proceedings of the 14th Annual Computer Security Applications Conference*, 1998.

[Zap01] M. G. Zapata. Secure ad hoc on-demand distance vector (SAODV) routing. Internet draft draft-guerrero-manet-saodv-00.txt, Internet Engineering Task Force, August 2001. expired 2002.

[ZL02] 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.