Distributed Monitoring in Ad Hoc Networks: Conformance and Security Checking

Wissam Mallouli, Bachar Wehbi and Ana Cavalli

Institut Telecom/Telecom SudParis, CNRS/SAMOVAR
{wissam.mallouli,bachar.wehbi,ana.cavalli}@it-sudparis.eu

Abstract. Ad hoc networks are exposed more than traditional networks to security threats due to their mobility and open architecture aspects. In addition, any dysfunction due to badly configured nodes can severely affect the network as all nodes participate in the routing task. For these reasons, it is important to check the validity of ad hoc protocols, to verify whether the running implementation is conform to its specification and to detect security flows in the network. In this paper, we propose a formal methodology to collect and analyze the network traffic trace. Observers running on a set of nodes collect local traces and send them later to a global observer that correlates them into a global trace thanks to an adapted time synchronization mechanism running in the network. The global trace is then analyzed to study the conformance and the security of the running routing protocol. This analysis is performed using dedicated algorithms that check the collected trace against a set of functional and security properties specified in an adapted formal language.

Keywords: *Ad Hoc Networks, Monitoring, Trace Collection and Correlation, Conformance Testing, Security Analysis, Nomad Logic.*

1 Introduction

Mobile ad hoc networks (MANET) are infrastructureless networks composed of a set of wireless mobile nodes. Nodes send packets directly to destinations that are in their coverage zone. When destinations are farther than the coverage range intermediate nodes cooperate to establish the communication path. This open and cooperative network aspect and the limited resources of mobile nodes make it difficult to define an efficient testing methodology to validate the conformance of existing routing protocols (like AODV [11], OLSR [6] or DYMO [5] etc.) and to guarantee the respect of predefined security properties.

Formal testing allows to insure the respect of the functional behavior and the security requirements of a system; it can be either active or passive. Active testing permits to validate a system implementation by applying a set of test cases and analyzing the system reaction. It implies that we have a global control on the network architecture which is difficult to perform in a dynamic topology such as ad hoc networks. Besides, the active testing becomes difficult to perform when the network is built from components (nodes) that are running in their

real environment and cannot be interrupted or disturbed. In this situation, there is a particular interest in using monitoring techniques that consist in testing passively during the run time the traffic flow in a deployed network. This testing consists in analyzing collected data according to some functional and security requirements described in a formal language.

In this paper, we use monitoring to collect distributed traces using local observers (called also probes) without interfering with the network under test. Two type of networks are considered. The first consists of a controlled area where a set of dedicated probes is installed to monitor the network. While the second is an open area network where the nodes perform themselves the trace collection task. In both cases, the local traces are sent to a global observer which is responsible for the traces correlation and analysis tasks. The correlation is performed based on an accurate time synchronization protocol [14] designed for ad hoc networks. This protocol follows the *receiver to receiver* mechanism that eliminates the major sources of synchronization inaccuracy. Whereas, the analysis consists of checking whether the trace is conform to a set of functional and security properties that we describe in a formal language adapted to distributed communicating systems. This checking is performed using a set of appropriate algorithms that we developed for this end. Once a property violation is detected, we identify the irregular node(s) behind it. Our mechanism allows to spot distant attacks that can only be discovered by the analysis of the global trace. More precisely, the main contributions of this paper are:

1. Definition of a precise method to collect distributed traces to cover the whole network. The collection methodology differs depending on the network nature (controlled or open areas).
2. Definition of a method for correlating local traces to obtain a global network trace. This correlation rely on an adapted time synchronization mechanism for ad hoc networks that permits to synchronize all the local observers.
3. Analysis of this global trace using specific algorithms to study the conformance and the security requirements of the considered routing protocol. The proposed algorithms allow to check a set of functional and security properties specified in Nomad formal language [7] on the collected trace.
4. Demonstration of the reliability of our approach by applying it on different ad hoc network scenarios running OLSR routing protocol to detect recurrent failures and attacks.

The remainder of this paper is organized as follows. In section 2, we discuss the related work tackling with monitoring in ad hoc networks. Section 3 presents the distributed collection of the ad hoc network traffic in a case of controlled network and an open area network. In section 4, we expose the approach to correlate the local traces in order to obtain the global network trace. Section 5 presents the methodology to analyze this global trace by comparing it to the functional and security requirements described in Nomad formal language. In section 6 we apply our methodology on OLSR routing protocol and the conclusion id given in section 7.

2 Related Work

Many papers [1,3,12,13] tried to tackle monitoring methodologies in ad hoc networks. In [13] the authors present DAMON, a distributed system for monitoring multi hop mobile networks. DAMON uses agents to collect the network traffic and sends collected measurements to data repositories. It was implemented in an AODV based ad hoc network. WiPal [1] is a merging tool dedicated to IEEE 802.11 traces manipulation which enables merging multiple wireless traces into a unique global one. Although DAMON and WiPal collect the network trace, they provide no process for its analysis.

The authors in [9] propose an intrusion detection scheme based on Extended Finite State Machines (EFSM) [8]. Indeed, they provide a formal model of the correct behavior of the routing protocol and detect specific deviations of the routing protocol implementation using a backward checking algorithm [2]. This work can only detect local attacks that violate the EFSM model of OLSR protocol (which is not the case of a big range of attacks).

The authors in [10] make use of a combination of deontic and temporal logic to specify the correct behavior of a node and to express complex security properties. They investigate different attacks targeting the link sensing mechanism of routing protocols and describe security policies to prevent them. Contrary to our methodology, this work considers only the local traffic trace of a given node. It does not allow to detect remote and distributed attacks. Moreover, it can only discover the existence of an incoherence in the collected traffic without determining the malicious node. In this paper we propose a different formal end-to-end methodology to collect and analyze global ad hoc network traffic.

3 Distributed Traffic Collection in Ad Hoc Networks

Network monitoring is an interesting approach that allows to collect the required information in order to analyze the behavior of the network. Monitoring in ad hoc networks can be *local* with respect to a node or *global* with respect to the network. In ad hoc networks, local monitoring is not sufficient to detect some types of errors and security anomalies [9, 12]. For this reason we adopt in this paper the global monitoring approach based on a distributed monitoring.

Controlled Area Network: In this type of network, nodes move inside a defined limited area. Therefore, it is possible to place a set of wireless observers responsible for capturing transited packets. These observers are placed to cover the whole network area. They collect the communication traces and send them to the global observer in the network. The choice of this node (global observer) can be based on administrative preferences. The broadcast nature of the wireless medium combined with the interferences problems represent a classical problem in the monitoring of ad hoc networks. That's why we chose to install the observers in such a way they cover each zone portion twice or more. The advantage of this method is the collection of real network traffic (attackers cannot alter the collected traces).

Open Area Network In the case of an open area network, the observers are the network nodes themselves. They perform a collaborative observation action. Each network node collects its local traffic trace and sends it to the global observer. We assume here that all the nodes have the collector program running on their systems. As the observers are the network nodes, it is possible for a node (attacker) to alter its collected trace. The traffic analyzer module on the global observer must take this property in consideration. This is the major difference with the limited area network where the collect is made by dedicated observers.

4 Traces Correlation Mechanism

The global observer receives the local traces collected by the local observers in order to analyze them. The first step toward performing this analysis is to correlate the traces and order them chronologically. We use a *receiver to receiver* network wide synchronization mechanism that we designed for wireless multi hop networks. Using this mechanism all the nodes in the network run with the same clock value allowing thus to perform the trace correlation. In the following we briefly describe the synchronization mechanism in section 4.1 and then describe the correlation procedure in section 4.2.

4.1 Synchronization Mechanism Overview

The objective of the time synchronization mechanism is to support each network node with the required timing information in order to build an adjustment function that transforms its local clock value to that of the reference node existing in the network. Using the adjustment functions they calculated, nodes, all over the network, run with similar clock values achieving therefore network wide synchronization. The mechanism is based on *receiver to receiver* synchronization which by definition eliminates the major sources of synchronization inaccuracy (send time and access time). The mechanism consists of two complementary parts; the sender nodes selection and the synchronization process. First, a hierarchy of sender nodes is constructed in order to guide the synchronization process in a multi hop environment. Sender nodes are responsible for transmitting reference messages. A reference message does not contain an explicit timestamp; instead, receivers use its arrival time to compare their clocks. Using information exchanged trough reference messages, each node constructs a table that contains for each received reference message the mapping between its local reception time and that of the reference node (or an already synchronized node). Then the node performs least squares linear regression to estimate the best fit line relating the node's clock to the reference node's clock. The estimated best fit line is an adjustment function that transforms the client's local clock value to that of the reference node. This adjustment function is given by equation 1 below:

$$T_{synch} = (1 + \widetilde{F}) \times T_{local} + \widetilde{Off} \tag{1}$$

Where $\widetilde{F}$ and $\widetilde{Off}$ are the estimated frequency error and offset parameters respectively. The synchronization process uses time information exchanged through reference messages to achieve first an initial estimate of the node's adjustment function. Then, by observing the offset estimate variation on longer time period, it improves the frequency error estimation and therefore the time synchronization accuracy. Details about the synchronization mechanism can be found in [14].

4.2 Global Trace Construction

Using the synchronization mechanism, network nodes run in phase with the reference clock value. This network virtual clock will assist the global observer in correlating the different local traces received from the set of observers. In [14] we showed that in a multi hop network the precision P of the synchronization mechanism is in the order of few micro seconds (maximum of $5\mu sec$ for nodes at 5 hops away of the time reference) which is by far less than the time difference between two message transmissions (a minimum of $100\mu sec$) and the time difference between the transmission time of a message and its reception time at a neighbor node (higher than $20\mu sec$). According to this accurate precision the following properties are always satisfied:

- If two nodes, $N1$ and $N2$, in the same broadcast region, send two different messages $M1$ then $M2$ at local times $t1$ and $t2$; then $t1$ ¡ $t2$.
- If a node sends a message at local time $t1$, a receiver receives the message at local time $t2$ where $t1$ ¡ $t2$.
- If two messages $M1$ and $M2$ are collected at local times $t1$ and $t2$ where $|t1 - t2| < P$ then either M1 and M2 are the same message or $M1$ and $M2$ are independent (i.e. they are transmitted in two different broadcast zones).

5 Monitoring Methodology

5.1 Functional and Security Properties Formal Specification

We specify a set of properties that the network nodes have to respect using Nomad formal language which allows to express privileges on non atomic actions. It combines deontic and temporal logics and can describe conditional privileges and obligations with deadlines. It can also formally analyze how privileges on non atomic actions can be decomposed into more basic privileges on elementary actions. More details about Nomad syntax and semantics are presented in [7].

Definition 1. *Atomic action*
We define an atomic action as the emission or the reception of a message between two nodes using the following syntax:

$$Node_1 \ ?or!\ Msg(Par_1,Par_2,...,Par_n)\ Node_2$$

where Node_1 and Node_2 represent the source or the destination of the message. '?' and '!' define a reception and an emission of a message by Node_1. $\text{Msg}(\text{Par}_1,\text{Par}_2,...,\text{Par}_n)$ represents the message exchanged between Node_1 and Node_2 with its parameters. Node_1, Node_2, Msg, and Par_i can be replaced by the symbol $*$ to represent any node, any message or any parameter.

Definition 2. *Non-atomic action*
If α and β are actions, then $(\alpha;\beta)$, which means "α is followed immediately by β" and $(\alpha;*;\beta)$, which means "α is followed by β" are non-atomic actions.

Definition 3. *Formulae*
If α is an action then start(α) (action α is being started) and done(α) (action α is done) are formulae.

Some properties on actions and formulae:

- If A and B are formulae then $(A \wedge B)$ and $(A \vee B)$ are formulae.
- If A is a formula then $\neg A$,$\oplus A$ ("Next in the trace," A is true), $\ominus A$ ("previously in the trace, A is true") are formulae.
- If A is a formula then $O^{\leq d}A$ ("d units of time ago, A was true if $d < 0$, or in the next d units of time, A will be true if $d > 0$") is a formula.
- $(A|C)$ is a formula: 'In the context C the formula A is true'.

Definition 4. *Deontic modalities*
If A is a formula then modality $\mathcal{O}$ ("A" is mandatory), $\mathcal{F}$ ("A" is forbidden) and $\mathcal{P}$ ("A" is permitted) are formulae.

5.2 Trace Analysis Approach

To run the distributed monitoring process, the global observer needs two different input files: the traces files collected by the local observers and the properties file where are specified expected functional and security properties.

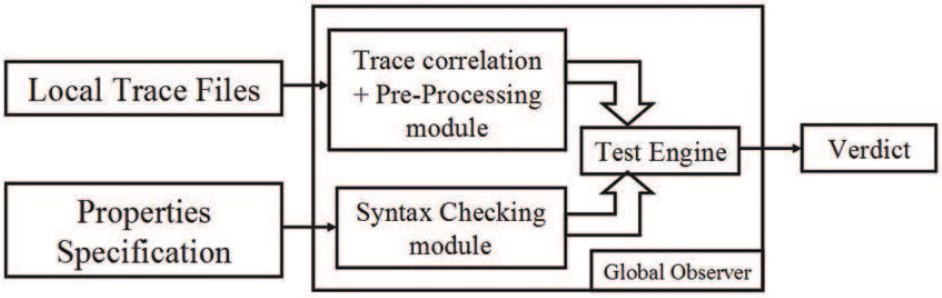

Fig. 1. Monitoring Architecture.

First, the global observer verifies through a syntax checking module that the desired behavior is well specified according to the Nomad format. This avoids syntax-related bugs in the test engine module.

Second, the collected traces files have to be analyzed using a pre-processing module that performs the following tasks: (i) filtering the traces files keeping only the relevant information for the protocol(s) under test. The basic idea is to keep in the traces only the messages and parameters corresponding to the specified properties to check. (ii) correlation of the traces files and the construction of a unique global trace file. (iii) parsing the global trace and creating a trace table which constitutes the target of the 'Test Engine' module queries. Each line of the trace table corresponds to an emission or a reception of a message in the network.

Finally, the trace analysis is performed using three algorithms according to the property type: permission, prohibition or obligation. These three algorithms are based on the same concept: each line in the trace table can correspond to (i.e. can be an instantiation of) one or many atomic actions described in one or many properties.

5.3 Properties Checking Algorithms

In this section we describe the general idea of the properties checking algorithms and provide in particular the overview of the algorithm verifying the prohibition properties on a network traffic trace.

Prohibitions Handler: The algorithm that allows checking prohibition properties begins first by parsing the trace table (build from the trace file) line by line to check if any context of any prohibition property is verified. For each line L, it verifies if L is an instantiation of an action A described in the context of the prohibition property Pr. If it is the case, it checks if the the chronological order of the actions described in this context is verified (using the procedure *Check_Context*), then it can deduce if the whole context is verified or not. If the context is verified, the algorithm has to ensure that the action described in the first part of the prohibition rule (the prohibited action) is not present in the trace. If it finds such action (using *Check_Prohibited_Activity* procedure), the verdict is FAIL. Otherwise, it concludes that the current rule is verified, the verdict in this case is: PASS. If the trace length is not long enough to ensure the verification, the output verdict is INCONCLUSIVE. The algorithm 1 presents the pseudo-code of the procedure used to check the prohibition properties on a trace and deduce the appropriate verdict. For each property Pr , we define 'Pr.action' as the prohibited action of the property and 'Pr.context' as the context of the property. 'Pr.action' (respectively 'Pr.context') is composed of one or many chronologically ordered actions 'Pr.act.action$_i$' (respectively 'Pr.context.action$_j$') where i (respectively j) is the number of atomic actions in the prohibited action (respectively context).

Algorithm 1 Prohibition Properties Handler

Require: $PPS[Pr]$: Prohibition Properties Set + $Tr[l]$: the trace table.

```
1: for each property Pr of PPS do
2:    Context(Pr) = 'not verified'
3: end for
4: for each line l of Tr do
5:    for each property Pr of PPS do
6:       if (Context(Pr)='verified') then
7:          verdict[Pr] := INCONCLUSIVE
8:          if (Prohibition deadline Reached) then
9:             verdict[Pr] := PASS
10:            Context(Pr)='not verified'
11:         else
12:            if (l=instantiation(Pr.act.action_i)) then
13:               verdict [Pr] := Check_Prohibited_Action (Pr.action)
14:               if (verdict [Pr] := 'FAIL') then
15:                  Memorize error and position in the trace
16:                  Context(Pr)='not verified'
17:               else
18:                  Memorize verified parts of the prohibited activity /* (in this case
                     verdict [Pr] := 'INCONCLUSIVE') */
19:               end if
20:            end if
21:         end if
22:      end if
23:      if (l=instantiation(Pr.context.action_i)) then
24:         Context(Pr) = Check_Context(Pr.context)
25:         if (Context(Pr) = 'verified') then
26:            Calculate prohibition deadline
27:         else
28:            if (Context(Pr) = 'not yet verified') then
29:               Memorize verified parts of the context
                  /* (Context (Pr) = 'not yet verified' if some actions of the context
                  are verified and are in the right chronological order. But the whole
                  context is not yet verified. We have to check next messages in the
                  trace, to deduce if the tested system is in the right context or not.) */
30:            else
31:               Erase memorized parts of the context if exist
                  /* (This is case when the context is no more verified) */
32:            end if
33:         end if
34:      end if
35:   end for
36: end for
```

Permissions Handler The permission to perform an action in a particular context does not mean that action must be systematically executed when this context is verified. In the case of checking permission properties, we first look in the traces file (the trace table) if the permitted activity exists; then, we ensure that the context was true to conclude that the property is respected (verdict PASS), otherwise the verdict is FAIL. If the trace is not long enough to check the context, the verdict is INCONCLUSIVE.

Obligations Handler For obligation properties the approach is very similar to that used for testing prohibition properties. We start first by checking whether the context of the property is verified. Then, we check if the action specified in the first part of the property (mandatory action) is present in trace. If it is the case, the verdict is PASS otherwise it is FAIL. If the trace is not long enough, the verdict is INCONCLUSIVE.

5.4 Irregular Node Determination

Once a property violation is detected, the monitor has to analyze the source of the violation in order to deduce the irregular node. The methodology of this determination is the following:

- Identification of the corresponding trace section: a violation is in general due to some messages in the global trace that does not respect a given property.
- Identification of the nodes implicated in a detected violation: in the case of a message reception related violation, the node claiming the reception, the assumed sender and its neighbors are implicated. In the case of an emission related violation, the assumed sender node and its neighbors are implicated.
- Identification of the implicated trace part: going backward in the trace from the position of the message causing the property violation to extract the messages related to the nodes implicated in the violation. The number of extracted messages depends on the studied protocol. In wireless networks, messages can be lost because of the interference and collisions problem. For this reason, ad hoc protocols like OLSR and AODV wait a certain number of periods before announcing a link break. In our study, we go backward in the trace for a certain period that guarantees the protocol convergence. For example, OLSR waits 3 periods of 2 seconds each before announcing a link break with a neighbor from which he has not received Hello messages. To guarantee that OLSR has converged (i.e. the link break is advertised) we go backward one more period; this means we extract the messages exchanged in the last 8 seconds.
- Construction of coherent nodes sets: the extracted trace part is analyzed to detect coherent and non coherent nodes within those implicated in the violation. We compare each pair of implicated nodes to detect if they are coherent or not. The set with the highest number of nodes is considered as the regular set whereas the remaining set (or sets) contain the irregular nodes. We assume that the number of irregular nodes in the network is lower than the number of regular nodes in all the broadcast regions.

6 Case Study : OLSR

We tested our methodology on OLSR ad hoc routing protocol in an open area network. We started first by extracting from the RFC some OLSR properties that we described in Nomad formal language. Then we changed in NS2 the behavior of OLSR in order to model typical attacks against OLSR like Hello message poisoning, link spoofing and black hole attack. We added in NS2 a special module that allows each node to collect its local network trace. This module gives the attacker the possibility to alter its local trace. A standalone module is also developed to correlate the collected local traces and analyze the obtained global trace using the algorithms presented in the previous sections.

We run a simulation with 100 mobile nodes located in a topology of 1500x1500 for 1200 seconds. Among these nodes, 5 are attackers and 2 of them can alter their local trace to simulate collaborative attack. In total 20 different attacks were launched. The simulation provided us the local traces that the standalone module correlated and analyzed. The global trace was around 5 million of lines. The analysis of the global trace gave 21 fail and 2 inconclusive verdicts. The inconclusive verdicts are due to incomplete execution trace due to multiple link breaks. The 21 fail verdicts correspond to the attacks and one false negative due to nodes mobility. In the next subsections, we emphasize on 2 of these attacks:

6.1 Hello messages poisoning

One of the first properties to check is the correct logical order of HELLO messages exchange. That is a node cannot announce a symmetrical link to any neighbor without having previously received a HELLO message claiming an asymmetric link from that node. The connectivity establishment process must respect the following properties:

- $Pr1 : \mathcal{F}\ (start\ (n\ ?\ Hello(n : Asym)I)$ — $O^{\leq 2sec} \neg\ done(n\ !\ Hello()*))$
- $Pr2 : \mathcal{F}\ (start\ (n?Hello(n : Sym)I)$ —
  $O^{\leq 2sec}\ (\neg\ done(n!Hello(I : Asym)*) \wedge \neg\ done(n!Hello(I : Sym)*)))$
- $Pr3 : \mathcal{F}\ (start\ (n?Hello(n : MPR)I)$ —$O^{\leq 2sec} \neg\ done(n!Hello(I : Sym)*))$

In figure 2, node I sends a Hello message claiming a symmetrical link to node A after receiving an empty Hello from it. In addition to this protocol violation I may insert a fake entry in its trace claiming the reception of an asymmetrical Hello message from A. In both cases, our methodology detected the violation:

1. I has not changed its local trace: In this case I violates the property $Pr2$. We can conclude that I is the malicious node.
2. I changed its local trace by claiming the reception of an asymmetrical Hello message from A. In this case the trace violates the property $Pr4$ which indicates that a message must have been emitted in order for a node to receive it.

- $Pr4 : \mathcal{O}\ (\ominus\ done\ (Node_1\ !\ M(p)\ Node_2)$ — $start\ (Node_2\ ?\ M(p)Node_1))$

6.2 Link Spoofing with Distant Node

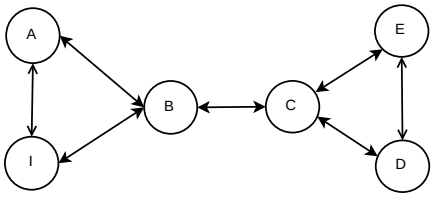

Fig. 2. A distant Link Spoofing Attack on OLSR.

In figure 2, we illustrate an example of a link spoofing attack on OLSR. The intruder I can insert Hello messages claiming a non existing symmetrical link to C. Consequently, the intruder might be selected as a MPR by A and the traffic from A to C will be disrupted to the intruder. If we analyze the global traffic in this part of the network, we notice one of these two cases :

1. Node I has not changed its local trace: Node I can not claim a symmetrical link to C according to the protocol specification violating thus property $Pr2$. We can conclude that I is the malicious node.
2. Node I has changed its local trace to claim the reception of a Hello message M from C specifying a symmetric link. Here, the trace violates the property $Pr5$ which indicates that if a node C receives a message from node N, all the symmetric neighbors of N ($V_S(N)$) must have received the same message. Therefore, we are in a message reception related violation; node I claiming the reception, the assumed sender C and its neighbors B, D and E are implicated. We split these nodes into two sets $\{I\}$ and $\{B, D, E\}$, the first claims the reception of the Hello message from node C where this message does not appear in the traces of the nodes in the second set. We can conclude that I is the irregular node. We note again that we are assuming that the number of irregular nodes is lower than that of regular ones in any neighborhood.

– $Pr5 : \forall\ B \in V_S(N),\ \mathcal{O}\ (done\ (B?M(p)N) \text{ — } done\ (C?M(p)N))$

We highlight here that this property expresses a distributed network behavior that allows to detect distant attacks. This detection can only be made through checking the global trace.

7 Conclusions and Future Work

This paper proposes a distributed monitoring approach to detect functional and security flows in ad hoc networks. It considers two types of networks : an open area network and a controlled area network. Dedicated observers collect the local

network traffic in a controlled area network whereas this collection is performed by the nodes themselves in an open area network. In both cases, the local traces are sent to a global observer. This latter is responsible for the local traces correlation and their analysis. The correlation is performed based on an accurate synchronization mechanism designed for ad hoc networks.

Our analysis rely on two main features : (1) functional and security properties specified using an instantiation of Nomad model, and (2) a correlated trace of the network traffic. Based on dedicated algorithms, we prove that our methodology allows to detect a large range of flows and errors.

As future work, we are investigating several approaches to improve the passive testing algorithms in order to perform online monitoring, possibly by including vulnerability cause graphs [4] of the implementation under test. We are also studying the different reactions that the network has to perform following a property violation detection.

References

1. http://wipal.lip6.fr/index.html.
2. B. Alcalde, A. R. Cavalli, D. Chen, D. Khuu, and D. Lee. Network protocol system passive testing for fault management: A backward checking approach. In *FORTE*, pages 150–166, 2004.
3. R. Badonnel, R. State, and O. Festor. Monitoring end-to-end connectivity in mobile ad-hoc networks. In *ICN (2)*, pages 83–90, 2005.
4. D. Byers, S. Ardi, N. Shahmehri, and C. Duma. Modeling software vulnerabilities with vulnerability cause graphs. In *ICSM*, pages 411–422, 2006.
5. I. Chakers and C. Perkins. Dynamic manet on-demand (dymo) routing. *IETF Internet-Draft draft-ietf-manet-dymo-06 (work in progress)*, 'Oct. 2006.
6. T. Clausen, P. J. (editors), C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L.Viennot. Optimized link state routing protocol (OLSR). RFC 3626, October 2003. Network Working Group.
7. F. Cuppens, N. Cuppens-Boulahia, and T. Sans. Nomad: A security model with non atomic actions and deadlines. In *CSFW*, pages 186–196, 2005.
8. D. Lee and M. Yannakakis. Principles and methods of testing finite state machines - A survey. In *Proceedings of the IEEE*, volume 84, pages 1090–1126, 1996.
9. J.-M. Orset, B. Alcalde, and A. R. Cavalli. An EFSM-based intrusion detection system for ad hoc networks. In *ATVA*, pages 400–413, 2005.
10. J.-M. Orset and A. R. Cavalli. A security model for olsr manet protocol. In *MDM*, page 122, 2006.
11. C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561, July 2003.
12. J. A. Ploskonka and A. R. Hurson. Self-monitoring security in ad hoc routing. In *ADHOC-NOW*, pages 238–251, 2006.
13. K. Ramachandran, E. M. Belding-Royer, and K. C. Almeroth. DAMON: A Distributed Architecture for Monitoring Multi-hop Mobile Networks. In *Proceedings of the 1st IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*, October 2004.
14. B. Wehbi, A. Laouiti, and A. Cavalli. Accurate and efficient time synchronization mechanism for wireless multi hop networks. Technical report, TELECOM SudParis, 2008.