Formal Supervision of Mobile Ad Hoc Networks for Security Flaws Detection

Wissam Mallouli and Bachar Wehbi
Montimage EURL, 39 rue Bobillot, 75013 Paris Cedex, France
Ana Rosa Cavalli and Stéphane Maag
Institut TELECOM / TELECOM & Management SudParis,
CNRS / SAMOVAR, 9 rue Charles Fourier, 91011 Evry Cedex, France

ABSTRACT

Mobile ad hoc networks (MANETs) are self-configuring networks of mobile devices connected via wireless links. Every device in MANETs is also a router because it is required to forward traffic unrelated to its own use. These kinds of networks are exposed more than traditional networks to security threats due to their mobility and open architecture aspects. In addition, any attack can severely affect the network as all nodes participate in the routing task. For these mentioned reasons, it is important to check the validity of ad hoc protocols, and verify whether their running implementation is in accordance with their security goals in order to detect potential security flaws in the network.

In this chapter, we propose a formal methodology based on passive testing techniques to collect and analyze ad hoc network traffic trace and detect potential attacks (distributed or not). Observers running on a set of nodes collect local traces and send them later to a global observer. An adapted time synchronization mechanism allows the global observer to correlate the received traces obtaining thus a global trace. The latter is then analyzed to study the security aspects of the running routing protocol. This analysis is performed using dedicated algorithms that allow checking the collected trace against a set of functional and security properties specified in a well adapted formal language called Nomad.

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 (Perkins 2003), OLSR (Clausen 2003) or DYMO (Perkins 2006) etc.) and to guarantee the respect of predefined security properties.

Ad hoc networks are useful in a variety of applications, particularly in areas when timeliness is important or, where creating a network infrastructure is not cost effective, such as military or rescue operations. The sensitivity of these uses for ad hoc networks makes routing security an important consideration. To ensure that different security goals are effective and that a certain level of security is always maintained, the network behavior must be restrained by a security policy. A security policy is a set of rules that regulates the nature and the context of actions (emission and reception of protocol packets) that can be performed within a network, according to specific nodes. The main objective is to ensure that security policy is well defined and that is actually implemented and respected in the network.

To reach this aim, we usually carry out audits that focus on administrative procedures and networks configurations. Tests are then carried out to check if some known vulnerabilities would remain present. If several tools for some specific tests exist in classical networks (wired or not), there is no general solution analyzing the overall MANET conformance according to its security policy. Several reasons can explain these deficiencies. First, there is currently few research works on formal modeling of complete security policies for MANET networks. In addition, analytical work about security checking often focuses on the verification of punctual elements, such as cryptographic protocols or code analysis. Thus, responsibles for security and all the system administrators are missing a formal solution to ensure the coherence of a routing protocol implementation with respect to its security policy, even if this last has been fairly well defined.

Most current work only concentrates on defining meta-languages in order to clearly express security policies and provide unambiguous rules. Or-BAC (Abou El Kalam 2006), PDL (Lobo 1999), Nomad (Cuppens 2005) and Ponder (Damianou 2000) are typical examples of such generic policy description models. They suggest concepts to describe the security policy independently of the system/network functional behavior. Once the security policy is formally specified, it is essential to prove that the target network implements this policy. Indeed, if one cannot ensure this conformance, the global security cannot be guaranteed anymore.

To achieve this goal, we can rely on several strategies of formal testing to insure the network security requirements; testing can be either active or passive. Active testing permits to validate the network deployment by applying a set of test cases targeting security aspects and analyzing its reaction. It implies that we have a global control on the network architecture which is difficult to perform in a dynamic topology as in 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 passively testing, during the run time, the traffic flow in a deployed network. This testing consists in analyzing collected data according to some security requirements described in a formal language.

In this chapter, we use monitoring to collect distributed traces using local observers (called also probes) without interfering with the network under test. Two types 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 (Wehbi 2008) 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 security properties that we describe in a formal language well 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 work are:

- 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). In fact, if one of the different observers misses to detect the transmission of one message between two or many different nodes (due to interference reasons), the trace analysis can become strained. We claim that our methodologies can avoid this kind of failure and thus decrease the rate of false positives. These methodologies are indeed well defined according the dynamicity rate of the nodes in the network and their global topology.

- Definition of a method for correlating local traces to obtain a global network trace. This correlation relies on an adapted time synchronization mechanism for ad hoc networks that permits to synchronize all the local observers.

- 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 (Cuppens 2005) on the collected trace. This global analysis can detect without ambiguity the location of the malicious node (if it exists).

- 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 chapter is organized as follows. Section 2 provides the background of mobile ad hoc networks as well as the formal testing techniques. In section 3, we discuss the related work tackling with the security monitoring mainly in ad hoc networks. Section 4 presents the distributed collection of the ad hoc network traffic in a case of controlled network and an open area network. In section 5, we expose the approach to correlate the local traces in order to obtain the global network trace. Section 6 presents the methodology to analyze this global trace by comparing it to the security requirements described in Nomad formal language. In section 6 we apply our methodology on OLSR routing protocol. And finally we present future work and conclude the chapter.

2. BACKGROUND

2.1 Basic definition on routing and MANET

Mobile ad hoc networks originated from the DARPA Packet Radio Network (PRNet) (Jubin 1987) and SURAN project (Lauer 1995) are becoming increasingly popular and expected to be key components of the infrastructure for achieving future ubiquitous society. Several applications dedicated to citizens and commercial needs such as localized information services, military and emergency applications are developed.

From the unique and inherent characteristics of ad hoc networks, many requirements for routing protocol design are raised. There have been many proposals for routing protocols for ad hoc networks, and several protocols have emerged. They can be classified into three main categories: the *proactive*, *reactive* and *hybrid* protocols.

A proactive routing protocol, also called "table driven" routing protocol, allows continuously evaluating routes to all reachable nodes and attempting to maintain consistent, up-to-date routing information. Then, a path from a source node to a destination is always available.
The nodes using a proactive routing protocol need to maintain a consistent view of the network topology. When a network topology change occurs, respective updates must be propagated through the network to notify the change. Many proactive routing protocols have been inspired from algorithms used in wired networks. Such protocols are the Wireless Routing Protocol (WRP) (Murphy 1996) or the Optimized Link State Routing Protocol (OLSR) (Clausen 2003).

The reactive routing protocols are also called "on-demand" routing protocols meaning that the routing paths are searched only when demanded applying a route discovery procedure. It terminates either when a route has been found or no route available.

In a MANET, active routes may be broken due to node mobility. Then, route maintenance is an important operation of reactive routing protocols. Ad hoc On-demand Distance Vector routing (AODV) (Perkins 2003) and Dynamic MANET On-demand (DYMO) (Perkins 2006) are examples of reactive routing protocols.

Hybrid routing protocols are proposed to merge the advantages of both proactive and reactive routing protocols and overcome their drawbacks. Hybrid routing protocols often apply hierarchical network architectures. Different hierarchical levels are exploited. The Zone Routing Protocol (ZRP) (Haas 2002) and Hybrid Ad hoc Routing Protocol (HARP) (Nikaein 2001) are examples of such protocols.

However, the research efforts dedicated to these protocols are most of the time focused on simulation. Only few efficient, scalable and reliable implementations have been provided.

The main reasons are to insure that all the functionalities presented in the IETF standards have been implemented as well the scalability of the implementation and the security of the algorithms. Besides, there are issues that make the design of mobile ad hoc network routing protocols a hard work. First, in MANET, node mobility causes frequent topology changes and network partitions. Secondly, because of the variable and unpredictable capacity of wireless links, packet losses may happen frequently. Furthermore, wireless medium and its broadcast aspect may leads to the hidden terminal and exposed terminal problems. Additionally, mobile nodes have power constraints, computing and bandwidth resources and require effective routing schemes.

As a consequence, conformance testing becomes a crucial phase of the ad hoc routing protocol design and development. Functionality and security failures caused by poor testing may have a catastrophic effect on the reliability of mobile wireless networks. For example, we may easily lose data, which may be used by other terminals outside the confidential network. An intruder may announce fake links and have access to sensitive data or deprive another node of it (Deny of Service attack) etc. In the past, different conformance testing methods have been developed for distributed communicating systems. Nevertheless, to adapt them to the validation of such routing protocols is not an easy task. Table 1 provides some attacks in mobile ad hoc networks and their known countermeasures.

Table 1: Some known attacks in mobile Ad hoc networks and their countermeasures.

Name	**Objective**	**Some Known Countermeasures**
Flooding Attack	Exhaust network resources, overall bandwidth, and individual nodes resources of computational and battery power. For Instance, in AODV attacking node sends out a large number of RREQs for a route to a non-existent node.	1. Calculate rate of neighbors RREQs, block if they exceed threshold. 2. Can not stop flooding below threshold and could block valid node if A is spoofing real nodes. 3. Use statistical analysis to detect varying rates of flooding.
Blackhole Attack	Attacking node *A* returns fake routing information, causing the source node to choice a route through *A*, the attacker can then misuse or drop messages as it sees fit.	1. Introduce route confirmation requests CREQ and route confirmations reply CREP. 2. Intermediate nodes return RREPs and send CREQs to the next-hop node in the route to the destination node D, that node can send a CREP to the source node if it has a route to D. 3. Can not defend against collusion between attacking nodes that returns false CREPs that validate the false RREPs. 4. Proposal for a statistical analysis that compares

		destination sequence numbers to compare RREPs.
Link Withholding Attack	Attacker does not advertise a link to a specific node or group of nodes (OLSR).	1. Nodes listen for the TC message from the MPR node they selected, if they do not hear one that MPR node is rated suspicious and additional MPR nodes are selected.
Link Spoofing Attack	A node A advertises links to non-neighbors, by faking links to the two-hop neighbors of a source node S, A can become one of its MPR nodes, and then manipulate traffic.	1. Equip nodes with GPS and calculate whether two nodes could really have a link. 2. Another solution is to include the 2-hop neighbors in the Hello message, this gives every node a 3-hop topology of the network, less expensive then special hardware, but is defeated by spoofing outside of 3-hops
Replay Attack	Attacker records another nodes control messages and resends them later. Can be used to spoof another node or just disrupt routing.	1. Add time stamp and asymmetric key to messages. 2. Reject old messages as suspicious.
Wormhole Attack	Two colluding attackers have a high speed link between them. Any RREQs that passes through the colluding nodes A_1 and A_2 will appear to cross the shortest path because of the high-speed link. This will cause a source node S to send all messages to D through the compromised links A_1 and A_2.	1. Packet leashes, temporal and geographical. 2. These prevent a packet from moving too far too fast.
Colluding Misrelay Attacks	Two colluding attackers modify or drop packets.	1. An acknowledgment system could detect this but will increase overhead. 2. Another solution is to increase transmission power twice to detect the colluding attackers. However even if we increase the transmission power K times, K+1 attackers can drop packets.

Indeed, ad hoc networks suffer from weaknesses and they are more vulnerable that wired networks. Because of their characteristics they have a very dynamic topology and they rely on an open medium and cannot rely on a fixed infrastructure to monitor their activities, to distribute keys or to manage security policies. As a result they are vulnerable to many kind of attacks. The majority of MANET attacks are based on design or implementation errors. Experience shows that protocols are often subject to implementation flaws that can be exploited by an attacker to compromise the network. For instance, in the proactive routing protocol, OLSR, a node can send a wrong HELLO message claiming symmetrical links to non-neighbor nodes or to nonexistent nodes. As a consequence, the node could be selected as a relay node and, thus, act as router for the traffic between other nodes. It can then discards all the traffic or just disrupt some messages to disturb the routing operation. In the case of the OLSR protocol, this is possible if the implementation does not strictly control the steps necessary to establish a symmetric link between two nodes, meaning actually the implementation is not in accordance to its RFC.

Some works have been carried out in order to design and test them. Most of these works rely on protocol descriptions using simulators in order to have an idea of the implemented protocol behaviour. However, the test coverage is rather low and the verdicts are only restricted to the simulated context. Furthermore, some works (Bhargavan 2002) have reported specific problems, such as the AODV implementation in NS-2 that was false regarding some properties like the initial value for the hops number in a RREP. All of this illustrates the importance of testing methods to check routing protocols.

2.1 Testing Techniques

Due to the flexible nature of wireless services and the high size of their user community, errors not detected through the incremental non-formal design process (standard requirements), could have severe impacts notably regarding the security aspects especially if they appear once deployed. In addition, although formal testing techniques have been used to validate wired systems for many years now (Holzmann 1991), wireless mobile ad hoc networks raise many novel constraints that open current new issues in the design and formal testing of their routing protocols (infrastructure-less, broadcast, dynamicity, etc.).

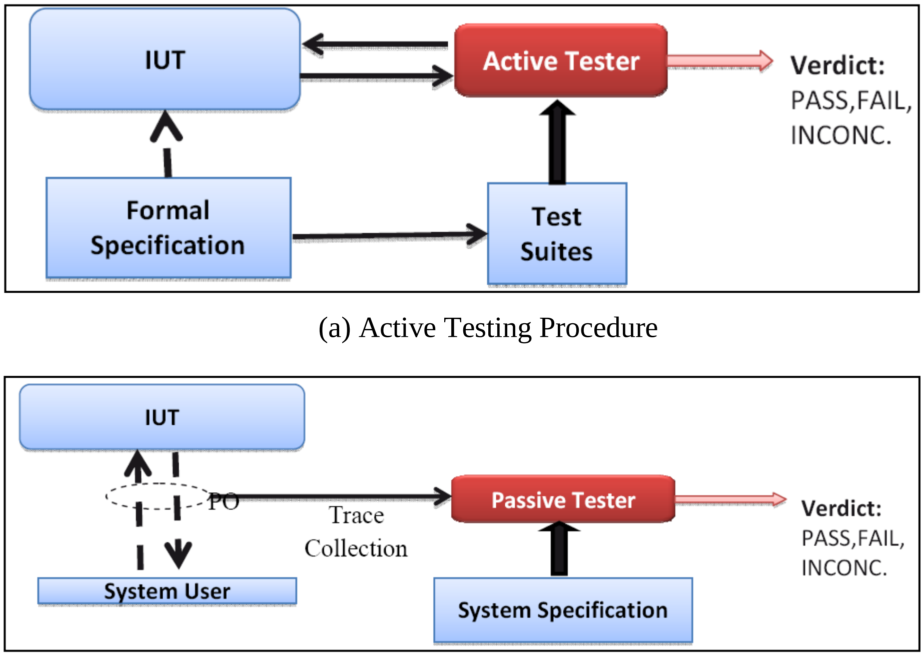
Nevertheless, some formal methods and tools may be successfully adapted to be used in the wireless context. The main objective of formal methods is to provide a non ambiguous model of the protocol from which properties (such as security aspects) may be extracted in order to be matched to the ones contained in a real implementation of the protocol specification.

Formal models can be defined using high level formalisms (for example SDL, PROMELA, stochastic Petri Nets, etc.) or more low level formalisms that are often the underlying semantic models of the high level ones. According to the type of properties to be checked, the appropriate formalism and its associated underlying model have to be properly chosen, especially because they share shortcomings. Besides, some formalisms are more convenient to address certain properties such as the broadcasting feature with the process algebra, the table-driven feature with SDL or PROMELA, the security aspects with Nomad.

Therefore, by designing formal description techniques and applying formal testing technique, we are able to guarantee that the protocol is free of errors, scalable, secure, etc. and that the implementation satisfies the requirements. This procedure consists in testing the conformity of the implementation according to the formal model. The goal of formal testing is to generate test sequences from the correct and verified models in order to be checked on the real implementation.

Two main mechanisms, having their own advantages and drawbacks, may be utilized for such purpose: *passive* and *active* testing. These techniques are illustrated in the Figure 1.

Active testing is based on the execution of specific test sequences to the implementation under test. These test sequences are obtained from the formal model according to coverage criteria. These criteria can be applied on the specification, e.g. coverage of all logical conditions, coverage of all paths. This allows us to set if we have covered the specification as well as the code in testing. The tests may be generated automatically or semi-automatically from test criteria, hypothesis, and test goals. When generating the tests, we are faced to the feasibility problem, the problem of deciding the feasibility of a path is undecidable. The format of these sequences which is commonly used by the testing community is TTCN3 (873-1 2007), from which their execution are performed through points of control (execution interfaces). These points of control are installed in the context of a testing architecture, which means the way to put the testers (e.g. upper and lower testers to test a specific stack layer, the different interfaces, and the oracle in order to provide a verdict on the executed tests to the tested implementation). We can mention two families of testing: the *static* and *dynamic active testing*. The first one is based on static analysis of the source code, i.e. the implementation. The code is inspected regarding the elaborated checklist or by analyzing the control and data flow graph. Using this kind of test, we do not have to exercise the system under test with real data. On contrary, dynamic testing implies that the system under test is executed under different configurations, i.e. with different input data tests. The tests sequences to be exercised on the implementation are derived from the model described by a FDT. Afterwards, the inputs of the test sequence are given to the implementation and the output results are compared to those expected by the specification.



(a) Active Testing Procedure

(b) Passive testing procedure

Figure 1 – Passive and Active Testing

Passive testing consists in observing the input and output events of an implementation under test in run-time. The term “passive” means that the tests do not disturb the natural run-time of a protocol or service. This concept is sometimes also refereed to as monitoring in the literature. The record of the event observation is called an event trace. This event trace will be compared to the formal specification as a test sequence. The passive testing techniques are applied especially because the active ones require important testing architectures, whose the testers need to control the system at some specific points. This is sometimes not feasible or even undesired. Nevertheless, while test sequences in active testing may give concrete verdicts (excepted for ”*inconclusive*” ones), an event trace that satisfies the model does not mean that the whole implementation satisfies the specification. On the other hand, if a trace does not satisfy, then neither does the implementation.

Figure 2 shows a summary of the formal testing techniques.

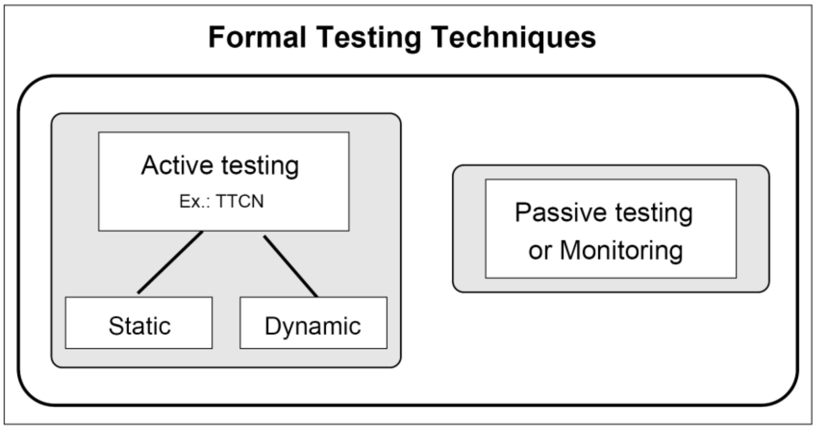


Figure 2 - Formal Testing Techniques

Although passive and active testing have their own advantages and drawbacks especially when used to wireless self-organizing network protocols, the results that may be obtained depends on the system under test and essentially on the testing goal, the testing type. The testing type considers the whole testing process of the protocol, which consists in different steps: unit, conformance, interoperability, integration testing, and so on. Most of these testing types are normalized. For instance the active conformance testing is standardized by the ITU-T in (9646-1 1994) in which common testing architectures, interfaces or points of control and observation are mentioned and specified. Nevertheless, these standards are mainly designed for wired systems and most of the time (if not always), the new inherent constraints of MANET (such as the lack of infrastructure or non-centralized systems) are omitted from these documents. This is mainly the reason why passive testing approaches are preferred when testing security aspects in such wireless mobile environments. Some works have been performed for testing the conformance or interoperability of MANET routing protocol applying active approaches. But from our knowledge, none of these techniques have been processed for distributed security properties. We thus present in the following the related works on the area but mainly focusing on the passive testing techniques.

3. RELATED WORK

The security checking is usually performed using intrusion detection systems (IDS) that employ either misuse detection or anomaly detection (Chen 2006). An Intrusion detection system (IDS) is software and/or hardware designed to detect unwanted attempts at accessing, manipulating, and/or disabling of computer systems, mainly through a network, such as the Internet. These attempts may take the form of attacks, as examples, by crackers, malware and/or disgruntled employees. It is used to detect several types of malicious behaviors that can compromise the security and trust of a computer system. This includes network attacks against vulnerable services, data driven attacks on applications, host based attacks such as privilege escalation, unauthorized logins and access to sensitive files, and malware (viruses, trojan horses, and worms).

An IDS can be composed of several components: *Sensors* which generate security events, a *Console* to monitor events and alerts and control the sensors, and a *Central Engine* that records events logged by the sensors in a database and uses a system of rules to generate alerts from security events received. There are several ways to categorize an IDS depending on the type and location of the sensors and the methodology used by the engine to generate alerts. In many simple IDS implementations all three components are combined in a single device or appliance.

As an example, SNORT[1] (Roesch 1999) is a free and open source Network Intrusion prevention system (NIPS) and network intrusion detection system (NIDS) capable of performing packet logging and real-time traffic analysis on IP networks. SNORT performs protocol analysis, content searching/matching, and is commonly used to actively block or passively detect a variety of attacks and probes, such as buffer overflows, stealth port scans, Web application attacks, SMB probes, and OS fingerprinting attempts, amongst other features. The software is mostly used for intrusion prevention purposes, by dropping attacks as they are taking place. SNORT can be combined with other software such as SnortSnarf, sguil, OSSIM, and the Basic Analysis and Security Engine (BASE) to provide a visual representation of intrusion data.

Many researches (Arnedo 2003, Shu 2006, Botha 2001, Ali 2001, Hoh 2006) are conducted to improve attacks detection for security purposes in various fields. These works are based on different models. Some of them describe the desired behavior of the system. In this case, the collected traffic that violates this behavior is considered as a malicious traffic due to a potential attack. In others methodologies, we only specify the known attacks to avoid any fault positive. A fault positive arises when an alarm is activated when no attack is performed. Based on this last methodology, non-known attacks are not detected.

Due to the mobility and open architecture aspects of MANET, classical IDS cannot be applied directly and have to be adapted to the specificities of ad hoc networks. Many papers (Ploskonka 2006, Badonnel 2005, Ramachandran 2004, Claveirole 2008) tried to tackle monitoring methodologies in ad hoc networks. In (Ramachandran 2004) 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 (Claveirole 2008) 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 (Orset 2005) propose an intrusion detection scheme based on Extended Finite State Machines (EFSM). 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 (Alcalde 2004). 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). Moreover, the detection of an eventual violation is not immediate; using the backward checking algorithm, there is a considerable delay before an alarm generation in the case of an error detection.

The authors in (Orset 2006) 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 incoherence in the collected traffic without determining the malicious node.

In this chapter, we propose a different formal end-to-end methodology to collect and analyze global ad hoc network traffic. The next section presents the first step of our methodology: the traffic collection.

[1] http://www.snort.org/

4. DISTRIBUTED TRAFFIC COLLECTION IN AD HOC NETWORKS

Unlike traditional networks, in ad hoc networks each node participates in the routing function. An irregular node could introduce serious disruptions in the network by advertising false routing information or simply not participating in the packet forwarding. Monitoring such networks is nowadays indispensable to detect security attacks. In order to analyze the behavior of the network and detect its potential security flaws, we need to collect its exchanged messages traffic based on several monitoring techniques. Monitoring in ad hoc networks can be **local** with respect to a node or **global** with respect to the network as explained in the following:

- In local monitoring (also called self-monitoring (Ploskonka 2006)), nodes capture the packet exchange in their neighborhood. In wireless networks, a node can sense all the packets sent by its neighbors.
- Global monitoring consists of collecting local traffic traces of a set of nodes in order to get a complete packet exchange trace of the network. Any packet injected into the network should be recorded in the global trace.

In ad hoc networks, local monitoring can be considered as a first step to detect some security attacks but is not sufficient to detect other types of errors or distant security attacks (Ploskonka 2006, Orset 2005). Indeed, some attacks (like Link Spoofing Attack) in MANET networks, can be in accordance with the standard protocol message exchange and only a complete picture of the network can allow their detection. For this reason we adopt a global monitoring approach described in the cases of a controlled area network and in an open area network.

4.1 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 external observers based-method is the collection of real network traffic (attackers cannot alter the collected traces). Figure 3 illustrates a controlled area network with a set of dedicated observers.

The main benefit of this kind of monitoring architecture is that the network nodes and especially malicious nodes do not participate in the trace collection task. That means that they cannot alter the collected traces by adding/deleting sent/received messages or modifying existing ones. The trace collection is performed by external observers in a secure manner. Indeed, communication between local observers and the global observer for the transmission of local traces is encrypted to ensure the integrity of transmitted data.

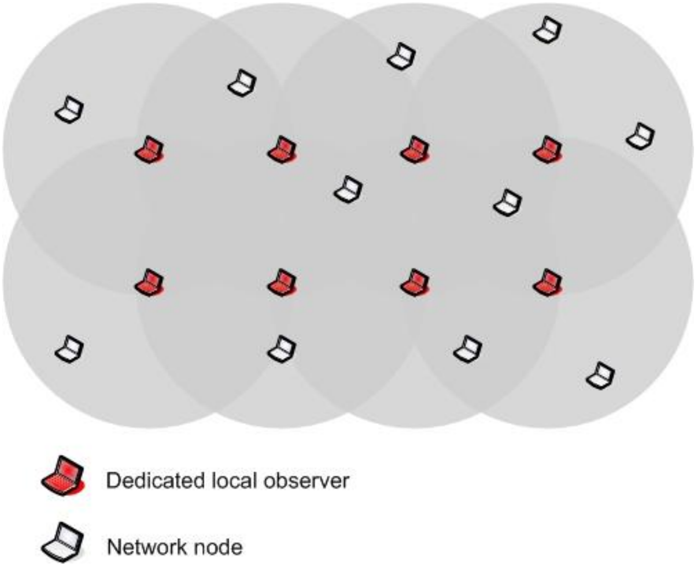


Figure 3: Network with dedicated observers.

4.2 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.

Another problem that can be considered in this kind of architecture is the malicious node can also prevent the transmission of the local traces to the global observer (based on Blackhole Attack for instance) or can alter transmitted traces by other nodes when they participate in the relaying procedure. In this case, the collection of the network traffic is performed by a confidence set of nodes that trust each other. The construction of this set can be done according to administrative issues (nodes that know each other) or to an adapted mechanism that evaluates the implication of each node in the network. The confidence group is not necessarily a dominating set. However, it constitutes one or many connected groups. Each connected group will analyze the traces in its neighborhood by collecting local traces and sending it to on elected global observers that correlates the traces to analyze them and report eventual running attacks. If the topology of node changes some separated confidence groups can merge and other one can be split into two or more and new global observers are elected.

5. TRACE 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 5.1 and then describe the correlation procedure in section 5.2.

5.1 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 through 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 below:

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

Where $\tilde{F}$ and $\widetilde{Off}$ are the estimated frequency error and offset parameters respectively. The synchronization process uses time information exchanged through reference messages to achieve in a short time period 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 (Wehbi 2008).

5.2 Global Trace Construction

Using the synchronization mechanism, network nodes run with relatively the same clock value. This network virtual clock will assist the global observer in correlating the different local traces received from the set of observers. In (Wehbi 2008), 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, N_1 and N_2, in the same broadcast region, send two different messages M_1 then M_2 at local times t_1 and t_2 ; then $t_1 < t_2$.

- If a node sends a message at local time t_1, a receiver receives the message at local time t_2 where $t_1 < t_2$.

- If two messages M_1 and M_2 are collected at local times t_1 and t_2 where $| t_1 - t_2 | < P$ then either M_1 and M_2 are the same message or M_1 and M_2 are independent (i.e. they are transmitted in two different broadcast zones).

6. MONITORING METHODOLOGY

The monitoring methodology we rely on is composed of three different steps:

- *The definition of the monitoring architecture:* in this chapter we rely on a distributed architecture to collect execution traces in an ad hoc network. Local traces files collected by local observers are sent to the global observer. These files contain the different messages exchanged in the network along with their emission/reception timestamps. The global observer merges them into a global trace according to the ordering mechanism described in section 5.2.

- *The network security formal description:* this description consists in a set of properties, specified in Nomad formal language that the network under test has to respect. These properties can express both functional or security requirements including temporal aspects.

- *Security analysis*: The global observer (monitor) analyzes the global trace and verifies it against the specified properties to deduce a global verdict. The verdict is *PASS* if the system respects its security requirements and *FAIL* if it does not. The *INCONLUSIVE* verdict is deduced if the monitor can not extract the necessary information from the collected traces (in the case of a short trace for example).

6.1 Security Properties Formal Specification

To specify the set of security properties that the ad hoc network traffic has to respect, we rely on Nomad formal language. The choice of this language was mainly motivated by the characteristics of Nomad that provides a way to describe permissions, prohibitions and obligations related to non-atomic actions within different contexts. Nomad allows to express privileges on non atomic actions. It combines deontic and temporal logics and can describe conditional[2] 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.

The main advantages of Nomad model are to provide means to specify:
- Privileges (that is permission, prohibition or obligation) associated with non atomic actions.
- Conditional privileges, which are privileges only triggered when specific conditions are satisfied.
- Privileges that must be fulfilled before some specific deadlines.

Nomad formal model is designed to regulate the nature and the context of actions that can be performed within different systems or organizations. Intended to be generic, Nomad do not formally define the atomic action it rely on, so that it can be applied to multiple case studies with different characteristics. In this chapter, we instantiate Nomad to fulfill passive testing aspects and we define then an atomic action as the following definition.

Definition 1: An atomic action
An atomic action in MANET represents the occurrence of an event within the studied network. It corresponds to the emission or the reception of a message between two nodes. It can be detected by the external probes to be stored in a log file. It has the following syntax:

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

or

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

[2] Most privileges do not apply unconditionally. Thus, we need to model privileges that are only active in specific contexts.

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 $Node_1$. $Msg(Par_1, Par_2,...,Par_n)$ 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. $(i \in \{1,\dots,n\})$

Definition 2: An atomic action
If α is an atomic action, then $\overline{\alpha}$ which means that "α did not occur", is an atomic action.

Definition 3: Non-atomic action
If α and β are actions, then (α;β), which means "α is followed immediately by β" and (α;;β), which means "α is followed by β" are non-atomic actions.*

Definition 4: Formulae
If α is an action then start(α) (action α is being started) and done(α) (action α is done) are formulae.

Definition 5:

- *If A is a formula then $\neg A$ is a formulae*
- *If A and B are formulae then $(A \wedge B)\$$ and $(A \vee B)$ are formula.*
- *If A is a formula then $\oplus^n A$ ("In the n next messages in the trace, A will be true) and $\Theta^n A$ ("in the n previous messages in the trace, A was true") are formula. Where $n \in \mathrm{N}^* \cup \infty$*
  - *$\oplus^1 A$ also denoted $\oplus A$ means that in the next message in the trace, A will be true.*
  - *$\Theta^1 A$ also denoted ΘA means that in the previous message in the trace, A was true.*
  - *$\oplus^\infty A$ means that next in the trace, A will be true.*
  - *$\Theta^\infty A$ means that previously in the trace, A was true.*
- *If A is a formula and d<0 then $O^d A$ ("d units of time ago, A was true") and $O^{<d} A$ (A was true within d units of time) are formula.*
- *If A is a formula and d>0 then $O^d A$ ("after d units of time, A will be true") and $O^{<d} A$ (A will be true within d units of time) are formula.*
- *If A and C are formula, then (A|C) is a formula: in the context C the formula A is true.*

Definition 6: 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.

To illustrate the Nomad formalism, we provide in the following two basic properties for an ad hoc network running OLSR routing protocol extracted from its RFC (Clausen 2003):

Example 1:

$$\mathcal{O}\ (\ start(Node_1\ !\ Hello(Node_2{:}Asym)\ *\ )\ |\ O^{<-2sec}\ done\ (Node_1\ ?\ Hello()\ Node_2\ ))$$

This obligation property $\mathcal{O}$ means that $Node_1$ has to declare an asymmetric link with $Node_2$ if he receives before (within 2 seconds) an empty Hello message from it.

Example 2:

$$\mathcal{F}(start\ (Node_1\ ?\ Hello(Node_1{:}Sym)\ Node_2)\ |\ O^{<-2sec}\ \neg\ done(Node_1\ !\ Hello(Node_2\ {:}Asym)\ *\ )\ \vee$$
$$\neg\ done(Node_1\ !\ Hello(Node_2\ {:}Sym)\ *\ ))$$

This prohibition property $\mathcal{F}$ means that it is not allowed that a $Node_1$ receives a Hello message claiming a symmetric link with $Node_2$ if $Node_1$ did not send before a Hello message claiming a symmetric or asymmetric link.

6.2 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 the security properties that the network has to respect.

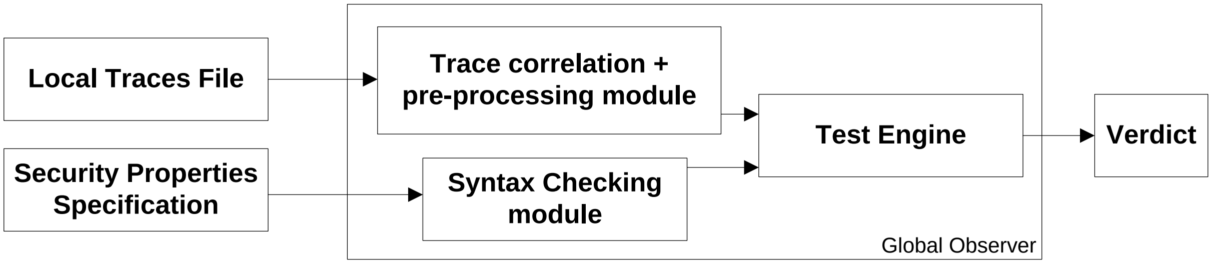

Local Traces File
Security Properties Specification
Trace correlation + pre-processing module
Syntax Checking module
Test Engine
Verdict
Global Observer

Figure 4: 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:

- 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.
- Correlation of the traces files and the construction of a unique global trace file.
- 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.

Definition 7 (L=Instantiation(Act))
A line L in the trace table T is an instantiation of an atomic action Act described in the security property Pr if the sender component S, the receiver component R and all the message parameters Par_i *mentioned in the action Act are the same existing in the table line L.*

For example, the first line of the table 2 is an instantiation of each of the following actions:

- $Node_1$! Hello ($Node_2$: Sym) Broadcast
- * ! Hello ($Node_2$: Sym) Broadcast
- $Node_1$! Hello (*) Broadcast

Table 2: An Example of a Line in a Formatted Table

	Timestamp	Sender	Destination	Msg_Type	Msg_Content
Line 1	T_1	$Node_1$	Broadcast	Hello	$Node_2$:Sym

6.3 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.

6.3.1 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 chronological order of the actions described in this context is verified (using the *Check_Context* procedure), 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: *FPS[Pr]: Prohibition Properties Set + Tr[l]: the trace table.*

```
1 for each property Pr of  FPS do
2 Context(Pr) := 'not verified'
3 end for
4 for each line l of Tr do
      for each property Pr of FPS do
          if (Context(Pr) = 'verified') then
              verdict[Pr] := INCONCLUSIVE
              if (Prohibition deadline Reached) then
                  verdict[Pr] := PASS
                  Context(Pr) := 'not verified'
```

```
            else
                if ($l=instantiation(r.act.action_i)$) then
                    verdict [r] := Check_Prohibited_Action(r.action)
                    /* only FAIL and INCONCLUSIVE verdicts are generated */
                    if (verdict [r] = 'FAIL') then
                        Memorize error and position in the trace
                        Context(Pr) :='not verified'
                    else
                        Memorize verified parts of the prohibited activity
                    endif
                endif
            endif
        endif
        if ($l=instantiation(r.context.action_i)$) then
            Context(Pr) = Check_Context(r.context)
            if (Context(Pr) = 'verified') then
                Calculate prohibition deadline
            else
                if (Context(Pr) = 'not yet verified') then
                    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.) */
                else
                    Erase memorized parts of the context if exist
                    /* (This is case when the context is no more verified) */
                endif
            endif
        endif
    end for
end for
```

6.3.2 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*. The approach and algorithm is very similar to that used for testing prohibition properties.

6.3.3 Obligations Handler

For obligation properties the approach is also 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*.

6.4 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.

7. INDUSTRIAL EXPERIMENTATIONS AND RESULTS

The monitoring methodology has been conceived and implemented to check whether we can detect attacks in ad hoc networks. It was especially applied to a MANET network running OLSR routing protocol. The analyses tool has been implemented (in JAVA language) as a standalone module. This latter was also applied to other industrial application. Section 7.1 presents the application of the methodology to OLSR case study and section 7.2 describes this same methodology to SAP R/3.

7.1 OLSR Case study

7.1.1 OLSR Overview

The OLSR, Optimized Link State Routing protocol is a proactive link state routing protocol, employing periodic message exchange to update topological information in each node in the network. OLSR uses two kinds of control packets: hello packet and TC packet (Topology Control). Each node uses Hello packet to update the neighbor table and compute its multipoint relays (MPR). The idea of multipoint relays is to minimize the overhead of flooding message in the network. TC packet broadcasted by a MPR node to the entire network contains the list of its neighbors. These control packets are used to build the routing table and to detect link disconnection. A node detects link disconnection to a neighbor node if not receiving a hello packet from a neighbor node within lifetime of the link. If link layer's information regarding connectivity to neighbor nodes is available, it is used together with the information from hello packet to maintain the neighbor table and the routing table.

In link layer notification method, when a node does not receive CTS after RTS transmission or does not receive ACK after data packet transmission, the link is identified as disconnected and link disconnection will be known to network layer. Then disconnected link will be discarded from neighbor table and routing table. When the packet transmission fails due to link disconnection, this packet is deleted from the MAC queue. However, other packets, which will use the same link, may exist in the MAC queue. If these packets are not removed from the MAC queue, unnecessary transmission will be repeated in the MAC layer, leading congestion. Incoming packet is dropped if a node has no route to the destination. To solve these problems, the mechanism to follow up detection of link disconnection is needed.

We tested our methodology on OLSR ad hoc routing protocol in an open area network and more precisely the Hello messages mechanism. 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:

7.1.2 Hello messages poisoning

The easiest way to launch an attack on OLSR is to announce fake links by generating wrong HELLO messages.
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:

- Pr_1: $\mathcal{F}(start\ (n\ ?\ Hello(n : Asym)\ I) \mid O^{<-2sec}\ \neg done(\ n\ !\ Hello()\ *\ ))$

- Pr_2: $\mathcal{F}(start\ (n\ ?\ Hello(n{:}Sym)\ I) \mid O^{<-2sec}\ (\neg done(n\ !\ Hello(I{:}Asym)\ *\ ) \vee$
  $\neg done(n\ !\ Hello(I{:}Sym)\ *\ )))$

- Pr_3: $\mathcal{F}(start\ (n\ ?\ Hello(n : MPR)\ I\ ) \mid O^{<-2sec}\ \neg done(n\ !\ Hello(I{:}Sym)\ *\ ))$

In figure 5, 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:

- *I* has not changed its local trace: In this case *I* violates the property Pr_2. We can conclude that *I* is the malicious node.
- *I* changed its local trace by claiming the reception of an asymmetrical Hello message from *A*. In this case the trace violates the property Pr_4 which indicates that a message must have been emitted in order for a node to receive it.

$$Pr4\text{: } \mathcal{O}\ (\Theta^{\infty}\ done\ (Node_1\ !\ M(p)\ *\ )\ |\ start(\ Node_2\ ?\ M(p)\ Node_1))$$

7.1.3 Link Spoofing with Distant Node

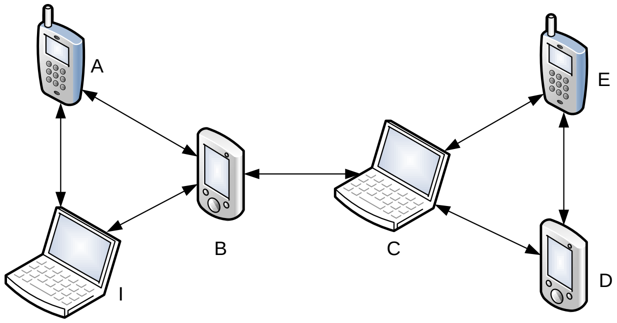


Figure 5: A distant Link Spoofing Attack on OLSR.

In figure 5, 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:

- 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 Pr_2. We can conclude that *I* is the malicious node.
- 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 Pr_5 which indicates that if a node *C* receives a message from node *N*, all the symmetric neighbors $V_S(N)$ of *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 that we are assuming that the number of irregular nodes is lower than that of regular ones in any neighborhood.

$$Pr_5\text{: } \forall\ \ B \in V_S(N),\ \mathcal{O}\ (done\ (B\ ?\ M(p)\ N)\ |\ done(C\ ?\ M(p)\ N))$$

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.

7.1.4 Blackhole Attack

The blackhole attack has two properties. First, the node exploits the mobile ad hoc routing protocol to advertise itself as having a valid route to a destination node, even though the route is spurious, with the intention of intercepting packets. Second, the attacker consumes the intercepted packets without any forwarding. There is a more subtle form of these attacks when an attacker selectively forwards packets. An attacker suppresses or modifies packets originating from some nodes, while leaving the data from the other nodes unaffected, which limits the suspicion of its wrongdoing.

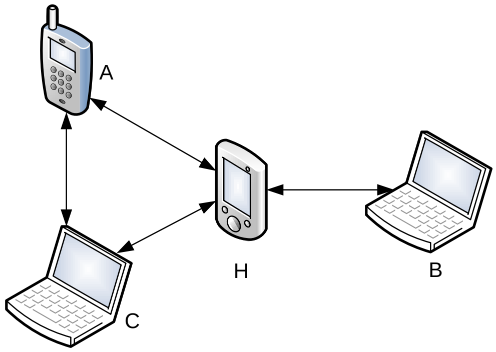


Figure 6: A Blackhole Attack on OLSR.

Let us take the example of the network presented in Figure 6 where node *H* does not forward messages from *A* to *B*. Two cases are possible:

- Node *H* does not declare the reception of data message M_D from node *A*. Node *C*, the neighbor of *A* receive M_D. If this observation is frequent (more than twice), we can conclude that *H* is a malicious node.
- Node *H* claims that he forwarded the data message M_D received from node *A*. All its neighbors did not receive this message. Then, we can deduce that *H* is a malicious node

In both cases, we can deduce that *H* is a malicious node.

7.2 The SAP Application

Our methodology has been applied on the SAP R/3 technology product of SAP[3] group. It is a package that incorporates various functions grouped in distributed modules that can interact with each other through a centralized information system based on client/server architecture.

SAP R/3 is a real time based system. Thus, every consumption (purchase, sale, etc.) or movement (in stocks for example) has to be immediately valued by updating all system modules involved in this activity. Let us study for example the operation of *a shipment confirmation*. This operation engenders an automatic billing operation, an operation for recording movements in stocks and bills, and possibly other updates at certain accounting services of the company. All these operations have to be done in real time, hence the conception of SAP R/3 as an integrated management software able of updating data of different modules instantaneously.

[3] http://www.sap.com

7.2.1 SAP R/3 security specification

In SAP R/3, any transaction that could be performed by users is identified by a unique code named: transaction code. For example the function used to change the *principal supplier* field of the system information table has the transaction code *FK02*, and the code *FB60* is used to describe the transaction, that manages *customer invoices* etc. SAP R/3 administrators have defined a set of rules including permissions, prohibitions and obligations that regulates performing each transaction. However, certain combinations of these transactions codes can lead to situations of conflict and/or incoherence. To avoid such situations, SAP R/3 system uses a generator profile to define generic roles that can be assigned later to users.

In SAP R/3, many security rules are defined to guarantee the system integrity and to control the access to critical data. We identified more than 120 different rules relatives to 10 clients and 26 possible operations. In the following three examples of these rules are provided:

- Rule 1: $\mathcal{P}$ *(start (John ! Msg(Tr.Code = FK01) R/3))*: this rule means that *John* is permitted to execute the transaction *FK01*.

- Rule 2: $\mathcal{F}$*(start(User ! Msg(*) R/3) |* $\ominus^{\infty}$ *done(User ? Msg(ID = AUM) R/3)* □ $\neg$ *done (User ? Msg(ID = UNLOCK) R/3))*: This rule expresses that any user whose access has been locked by the system is prohibited to perform transactions.

- Rule 3: $\mathcal{O}$ *(Start(User ? Msg(ID = AUM) R/3) |* $\ominus^{\infty}$ *done((User ! Msg(ID = AU2) R/3); (User ! Msg(ID = AU2) R/3); (User ! Msg(ID = AU2) R/3)))*: This obligation rule expresses that the R/3 system has to lock any user who has made three unsuccessful connection attempts.

7.2.2 Trace collection: Audit System

SAP R/3 has an audit system that permits to store all the events and transactions that occur during a period of time and to give the system administrators an overview about the users' activities within the system. It provides a set of files that contain detailed information about every activity undertaken by system users. These files represent the execution traces of SAP R/3 and constitute an important source of data for all kinds of testing or checking system security to detect any intrusion attempts, fraud, or any other malicious activity. In our case, we will use these files to test the respect of SAP R/3 to its security policy specified by administrators.

In our case study and for confidentiality reasons, we collected the execution traces of a running SAP R/3 system managing a fake database (with fake users' names and fake transactions). All the transactions are performed randomly and many times by all the users. The obtained log file has a size of 800 MByte and contains the details of 2.5 millions transactions (lines). Each transaction has a pre-formatted structure which can be divided into eight fields.

In Table 3, we present three formatted examples of events that we can find in a file log of SAP R/3. In the first one (*E1*), the user *Bob* connects from the workstation 1 which is located in Room *S826*. In second event (*E2*), the same user tried unsuccessfully to execute the transaction *F110* which deals with a payment operation. In (*E3*), the user *John* has made three attempts to connect without success. Hence this account has been locked by the system.

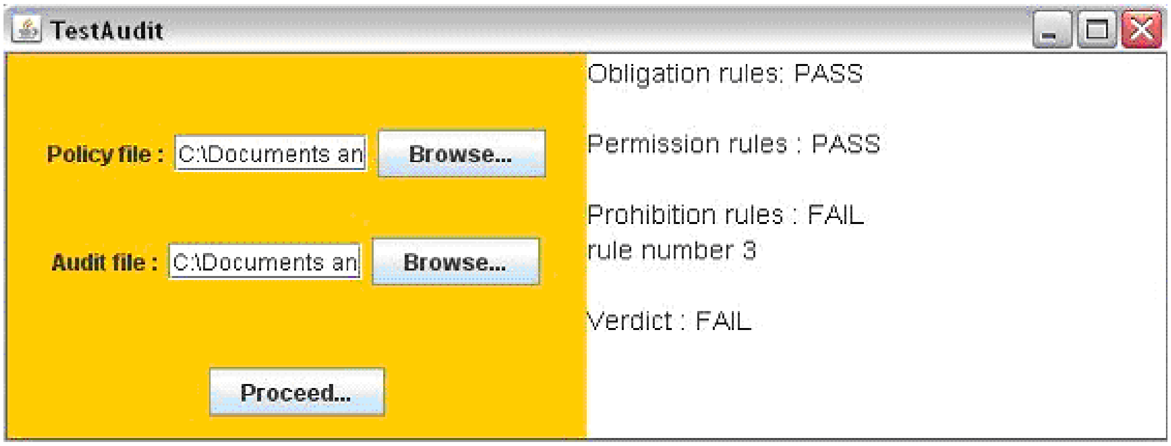
Table 3: Example of formatted contents in a file audit

	Date	Time	Client	User	Code	Terminal	ID	Message Text
E1	01.04.2009	08:55:04	600	Bob		S826-01	AU2	Login Failed
E1	01.04.2007	08:56:30	600	Bob		S826-01	AU1	Login Successful
E2	01.04.2007	13:43:11	600	Bob	F110	S826-01	AU4	Transaction F110 Failed
E3	08.04.2007	08:55:04	654	John		S826-01	AU2	Login Failed
E3	08.04.2007	08:55:06	654	John		S826-01	AU2	Login Failed
E3	08.04.2007	08:55:08	654	John		S826-01	AU2	Login Failed
E3	08.04.2007	08:55:09	654	John			AUM	User John Locked in Client 654

7.2.3 Passive testing results

We first performed our test to check the 120 rules on the SAP R/3 collected traces. The result of this first test was a *PASS* verdict which means that according to information contained in the audit file, all security rules were respected (we checked manually that each rule was verified at least once). To obtain this final verdict, the tester performed a simple Boolean operation, which combines the three partial verdicts already established by the three different sub-modules dedicated for permissions, prohibitions and obligations. Thus, if one of these verdicts is *INCONCLUSIVE* or *FAIL* the final verdict will be too. Otherwise *PASS* is the verdict to be deduced.

To demonstrate the reliability of our tester, we manually edited the file containing traces of SAP R/3 system. We have, for instance, added a transaction that violates the rule number 3. The verdict given by the tester shown in Figure 7 is FAIL: the violation of the rule has been detected which indicates the correctness of the tester.



Policy file : C:\Documents an

Audit file : C:\Documents an

Obligation rules: PASS

Permission rules : PASS

Prohibition rules : FAIL
rule number 3

Verdict : FAIL

Figure 7: Passive tester interface in the case of a fail verdict

8. FUTURE RESEARCH DIRECTIONS

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 of the implementation under test. We are also studying the different reactions that the network has to perform following any property violation detection. This will enable the detection on real-time system the crashes and the security rules violations and most importantly to be able to stop this kind of malicious behaviors without any delay. We are also planning to adapt the proposed methodology for bigger networks to study its scalability.

9. CONCLUSION

This chapter 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 relies on two main features: (1) 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 detecting a large range of flows and attacks.

The main advantage of the implemented security testing tool is that can be easily adapted to check classical protocols and services. Indeed, we demonstrate its effectiveness, through an industrial service proposed by SAP group namely: SAP R/3. It is important to notice that the performances of our tester are very suitable compared to the complexity of the described rules and the length of considered traces. We also showed that our approach permitted us to specify and verify temporal security constraints that are considered as an important issue in security testing.

REFERENCES

873-1 (2007). *Methods for Testing and Specification*, The testing and test control notation version 3, Part 1: TTCN-3 core language, v3.2.1. Tech. rep., ETSI.

9646-1 (1994). *Information technology* - *Open Systems Interconnection* - *conformance testing methodology and framework* - part 1: General concepts. Tech. rep., ISO.

A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin (2003). Organization Based Access Control. *In the 4th IEEE International Workshop on Policies for Distributed Systems and Networks (Policy'03)*. Lake Come, Italy.

H. Arafat Ali (2001). A New Model for Monitoring Intrusion Based on Petri Nets. *Information Management - Computer Security Journal, 9(4), 175-182.*

Baptiste Alcalde, Ana R. Cavalli, Dongluo Chen, Davy Khuu and David Lee (2004). Network Protocol System Passive Testing for Fault Management: A Backward Checking Approach. *In FORTE*, pages(150-166).

José Antonio Arnedo, Ana R. Cavalli and Manuel Nuñez (2003). Fast Testing of Critical Properties through Passive Testing. *In TestCom*, pages (295-310).

Remi Badonnel, Radu State and Olivier Festor (2005). Monitoring End-to-End Connectivity in Mobile Ad-Hoc Networks. *In ICN*, pages 83-90.
K. Bhargavan, C. Gunter, I. Lee, O. Sokolsky, M. Kim, D. Obradovic and M. Viswanathan (2002). Verisim: Formal analysis of network simulations. *In the IEEE Transaction Software. Engineering Journal*, 28(2):129–145.

Martin Botha and Rossouw Von Solms (2001). The Utilization of Trend Analysis in The Effective Monitoring of Information Security. Part 1: The Concept. *Information Management - Computer Security Journal, 9(5), 237-242.*

David Byers, Shanai Ardi, Nahid Shahmehri and Claudiu Duma (2006), Modeling Software Vulnerabilities With Vulnerability Cause Graphs. *In ICSM*, pages (411-422).

You Chen, Yang Li, Xueqi Cheng and Li Guo (2006). Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System. *In Inscrypt*. Pages (153-167).

T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L.Viennot (2003). *Optimized Link State Routing Protocol (OLSR)*. IETF RFC 3626.

T. Claveirole, M. Dias de Amorim (2008). WiPal and WScout, Two Hands-on Tools for Wireless Packet Traces Manipulation and Visualization, *In Mobicom Workshop on Wireless Network Testbeds*.

Frédéric Cuppens, Nora Cuppens-Boulahia and Thierry Sans (2005), Nomad: A Security Model with Non Atomic Actions and Deadlines. *In CSFW*. Pages (186-196).

Nicodemos Damianou, Naranker Dulay, Emil Lupu and Morris Sloman (2000). Ponder: An Object-oriented Language for Specifying Security and Management Policies. *In the 10th Workshop for PhD Students in Object-Oriented Systems*. Sophia Antipolis, France.

Baik Hoh, Marco Gruteser, Hui Xiong and Ansaf Alrabady (2006). Enhancing Security and Privacy in Traffic-Monitoring Systems. *IEEE Pervasive Computing Journal, 5(4), 38-46.*

G. J. Holzmann (1991). *Design and Validation of Computer Networks*. Prentice-Hall International Editions, AT&T Bell Laboratories.

John Jubin and Janet D. Tornow (1987), *The DARPA packet radio network protocols*. Proceedings of the IEEE, 75(1): 21-32.

G. S. Lauer, Packet-radio routing (1995). *In Routing in Communications Networks*, edited by Martha E. SteenStrup, Prentice-Hall, Englewood Cliffs, New Jersey.

Jorge Lobo, Randeep Bhatia and Shamim A. Naqvi (1999). A Policy Description Language. *In AAAI/IAAI*. pages (291-298).

D. Maltz, J. Broch, and D. Jonhson (1999), Experiences designing and building a multi-hop wireless ad hoc network testbed, *Carnegie Mellon University, Tech. Rep*.

S. Murthy and J.J. Garcia-Luna-Aceves (1996), *An Efficient Routing Protocol for Wireless Networks*, ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks, pp. 183-97.

Navid Nikaein, Christian Bonnet and Neda Nikaein (2001*), HARP - Hybrid Ad Hoc Routing Protocol*, in proceeding of IST 2001: International Symposium on Telecommunications.

Jean-Marie Orset, Baptiste Alcalde and Ana R. Cavalli (2005). An EFSM-Based Intrusion Detection System for Ad Hoc Networks. In ATVA, pages (400-413).

Jean-Marie Orset and Ana R. Cavalli (2006). A Security Model for OLSR MANET Protocol. *In FMUIT*, pages (122-126).

C. Perkins, E. Belding-Royer, and S. Das (2003). *Ad hoc On-Demand Distance Vector (AODV) Routing*. IETF RFC 3561.

C. Perkins and I. Chakeres (2006). *Dynamic MANET On-demand (DYMO) Routing*. IETF draft.

J. A. Ploskonka and Ali R. Hurson (2006). Self-Monitoring Security in Ad Hoc Routing. *In ADHOC-NOW*, pages (238-251).

Krishna Ramachandran, Elizabeth M. Belding-Royer and Kevin C. Almeroth (2004). 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.* Santa Clara, CA.

Martin Roesch (1999). SNORT: Lightweight Intrusion Detection for Networks. *In LISA*, pages (229-238).

Guoqiang Shu and David Lee (2006). Message Confidentiality Testing of Security Protocols - Passive Monitoring and Active Checking. In TestCom, pages (357-372).

Bachar Wehbi, Anis Laouiti, Ana R. Cavalli (2008). Efficient Time Synchronization Mechanism for Wireless Multi Hop Networks. *In the 19th annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*. Cannes, France.