EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Formal reconstruction of attack scenarios in mobile *ad hoc* and sensor networks

Slim Rekhis* and Noureddine Boudriga

Abstract

Several techniques of theoretical digital investigation are presented in the literature but most of them are unsuitable to cope with attacks in wireless networks, especially in Mobile *Ad hoc* and Sensor Networks (MASNets). In this article, we propose a formal approach for digital investigation of security attacks in wireless networks. We provide a model for describing attack scenarios in a wireless environment, and system and network evidence generated consequently. The use of formal approaches is motivated by the need to avoid *ad hoc* generation of results that impedes the accuracy of analysis and integrity of investigation. We develop an inference system that integrates the two types of evidence, handles incompleteness and duplication of information in them, and allows possible and provable actions and attack scenarios to be generated. To illustrate the proposal, we consider a case study dealing with the investigation of a remote buffer overflow attack.

Keywords: Digital investigation, Wireless networks, Formal proof, Attack scenarios reconstruction, Network of observation

Introduction

Faced with an increasing number of security incidents and their sophistication, and the inability of preventive security measures to deal with all latest forms of attacks, digital forensic investigation has emerged as a new research topic in information security. It is defined as the use of scientifically derived and proven methods towards the preservation, collection, validation, identification, analysis, interpretation, and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal or helping to anticipate unauthorized actions shown to be disruptive to planned operations [1]. One important element of digital forensic investigation is the examination of digital evidence (i.e., trails and clues left by attacker when they executed malicious actions) collected from the compromised systems to make inquiries about past events and answer "who, what, when, why, how, where" type questions. Several objectives can be fulfilled by a digital forensic investigation, including:

- reconstruction of the potentially occurred attack scenario;
- identification of the location(s) from which the attacker(s) has/have remotely executed the actions part of the scenario;
- understanding what occurred to prevent future similar incidents;
- argumentation of the results with non-refutable proofs.

As informal and unaided reasoning would make the analysis of traces and chains of events collected from evidence sketchy and prone to errors, the formalization of the digital forensic investigation of security incidents is of paramount importance. In fact, a formal description of the event reconstruction algorithm would make the potential scenarios it generates multiple and rigorous. It also helps to develop an independent verification of incident analysis, and prevents attackers from evading responsibility due to lack of rigorous and proven techniques that could convict them. Moreover, the attack scenarios generated using a formal and mathematical way can be used to feed data in attack libraries, helping administrators preventing further occurrence of such attacks. Formal methods can also be used to provide

* Correspondence: slim.rekhis@gmail.com
Communication Networks and Security Research Laboratory, University of Carthage, Tunisia

SpringerOpen

© 2011 Rekhis and Boudriga; licensee Springer. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/2.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

multiple ways to cope with incompleteness of the collected data.

During recent years, some research [2-8] has been proposed in the literature to form a digital investigation process based on formal methods, theories, and principles. The aim is to support the generation of irrefutable proofs regarding reconstructed attack scenarios, reducing the complexity of their generation, and automating the reasoning on incidents. A review of these approaches, which were designed without bearing in mind that the attacks can be conducted in a wireless network, will be provided in the next section. Due to the increasing use of wireless communication and network community interest in mobile computing, industry, and academia have granted a special attention to Mobile *Ad hoc* and Sensor Networks (MASNets). The inherent characteristics of these networks, including the broadcast and unreliable nature of links, and the absence of infrastructure, force them to exhibit new vulnerabilities to security attacks in addition to those that threaten wireline networks. These characteristics make it harder to use the evidence collection techniques and scenarios analysis methods proposed by the above-cited works, in order to address digital investigation in MASNets [9].

To the best of our knowledge, none of the existing research has considered the problem of formal investigation of digital security attacks in the context of wireless networks. In this article we provide a framework for formal digital investigation of security attacks when they are conducted in MASNets. The proposal deals with both evidence collection mechanisms in wireless multihop networks, and inference of provable attack scenarios starting from evidence collected at different locations in the network and the victim system. It is worth noting that a special case of the results have been addressed in [10], where a first version of an inference system was proposed to generate theorems regarding potential attack scenarios executed in an *ad hoc* network. The work in [10] was unable to cope with investigation in sensor networks as nodes may be scheduled to sleep and wake up to save energy, which affects the process of evidence collection and reassembly. In this work, we substantially reshaped the inference system, addressed energy management, and developed several missing properties and proofs. The model, that we propose to describe attack scenarios, is based on a formalism inspired from Investigation-based Temporal Logic of Actions [8]. The proposed model describes two types of evidence that can be generated, namely network and system evidence. The evidence in the network are generated by a set of nodes, called observers, that we distribute in the MASNet to monitor the traffic sent to/from nodes within their transmission range. The evidence in the system are generated by the set of installed security solutions. We propose an inference system that integrates the two types of evidence, handles incompleteness and duplication of information in them, and allows the generation of potential and provable actions and attack scenarios. We consider a case study dealing with the investigation of a remote buffer overflow attack on a vulnerable server, where the evidence are captured by observers which change their locations during the attack occurrence. While the proposal does not provide a solution to the conducted attack scenarios, their formal reconstruction from the collected evidence is a step toward a good protection. In fact, the generation of a provable scenario enables a good understanding of the weakness of the system that led the scenario to succeed, identification of steps that should be prevented by security solutions to avoid a further compromise of the system, and updating of the library of attacks to enhance the reliability of further investigations.

The article contributions are fourfold. First, we propose a method which helps engineers to conduct a digital investigation free of errors. Typically, these errors could happen due to the complexity of analysis and misunderstanding of the evidence content. Second, we provide a formal environment for the description and management of evidence, which allows enabling a digital investigation using a theorem proving based method. Third, the generation of evidence and the investigation process consider the use of system and network evidence while providing an efficient matching and correlation of them. It is worth mentioning that while the use of formal techniques could make the approach less usable than rival approaches, the techniques we propose are more useful. In fact they can be easily automated helping the development of automated incident analysis tools that generate results acceptable in a court of law, since all the results they deduce are provable. Fourth, the model we propose can cope with a large set of attack scenarios. It suffices to choose the suitable variables to model the attacker behavior and the manner by which the system is expected to react. Nonetheless, some extensions need to be considered to cope with distributed and cooperative forms of attack.

The article is organized as follows. The next section describes the set of requirements for digital investigation in MASNet and describes the characteristics of the considered MASNet. Section IV provides a model for describing wireless attack scenarios and characterizes evidence provided by security solutions and observer nodes. Section V proposes an inference system to prove attack scenarios in wireless networks. In Sect. VI, we describe a methodology for digital investigation which shows the use of the inference system. In Sect. VII a case study is proposed. The last section concludes the work.

Related Works

Stephenson [2] took interest in the root cause analysis of digital incidents and used Colored Petri Nets. Stallard and Levitt [3] used an expert system with a decision tree that exploits invariant relationships between existing data redundancies within the investigated system. Gladyshev [4,11] provided a Finite State Machine (FSM) approach for the construction of potential attack scenarios discarding scenarios that disagree with the available evidence. Carrier and Spafford [5] proposed a model that supports existing investigation frameworks. It uses a computation model based on a FSM and the history of a computer. A digital investigation is considered as the process that formulates and tests hypotheses about past events or states of digital data. Willanssen [12] takes interest in enhancing the evidentiary value of timestamp evidence. The aim is to alleviate problems related to the use of evidence whose timestamps were modified or refer to an erroneous clock (i.e., which was subject to manipulation or maladjustment). The proposed approach consists of formulating hypotheses about clock adjustment and verifying them by testing consistency with observed evidence. Later, in [6], the testing of hypotheses consistency is enhanced by constructing a model of actions affecting timestamps in the investigated system. An action may affect several timestamps by setting new values and removing the previous ones. In [7], a model checking-based approach for the analysis of log files is proposed. The aim is to search for a pattern of events expressed in formal language using the model checking technique. Using this approach logs are modeled as a tree whose edges represent extracted events in the form of algebraic terms. In [8], we provided a logic for digital investigation of security incidents and its high level specification language. The logic is used to prove the existence or non-existence of potential attack scenarios which, if executed on the investigated system, would produce different forms of specified evidence. In [13], we developed a theory of digital network investigation which enables characterisation of provable and unprovable properties starting from the description of security solutions and their generated evidence. A new concept, entitled *Visibility*, was developed for that purpose and its relation with *Opacity*, which was recently presented as a promising concept for the verification of security properties and the characterisation of unprovable incidents in digital investigation, was shown.

While the above cited approaches have proved to be able to support formal analysis of digital evidence, they are unsuitable for the investigation of attacks in wireless networks, especially, in MASNets. While the formalism they use to model attacks can support the description of a wide range of attacks scenarios, the techniques they provide to reconstruct scenarios of attacks, are not suitable to deal with evidence collected in wireless multi-hop system. In fact, the following assumptions they make are unable to cope with the characteristics of MASNets: First, the intermediate routers are assumed to be trusted and do not contribute to the security incident. In MASNets, any node in the network can participate in relaying the multi-hop traffic. These nodes which could be malicious, may generate serious forms of attacks, which need to be investigated. Second, the network topology is assumed to be static during the attack and the routing paths followed by the malicious traffic are supposed to be, in the great majority of cases, unchangeable during the attack scenario. In MASNet, the network security solutions (e.g., IDS) installed to monitor the attacker or the victim network, are unable to capture all the network traffic that convey the attack, especially if they move out of the transmission range of the nodes which participate in generating and forwarding the traffic from the attacker to the victim. Third, all nodes in the network are supposed always to be active and ready to generate evidence if a malicious activity is noticed. However, as in wireless sensor networks, energy is an important concern, so nodes may sleep when the communication channel is idle and wake up to receive messages. Therefore, providing a formal investigation scheme, which is suitable for the reconstruction of potential attack scenarios in the context of MASNet, is of major importance.

To the best of our knowledge, none of the existing research has considered the problem of formal investigation of digital security attacks in the context of wireless networks, with only a few pointing out the problem. Slay and Turnbul [14], for instance, discussed the forensic issues associated with the 802.11a/b/g wireless technology. They stressed the need for technical solutions to evidence collection that cope with the wireless environment. Some other works have concentrated on a specific issue which is the traceback of the intruders' source. Huang and Lee [15], for instance, proposed a Hotspot-based traceback approach to reconstruct the attack path in a MASNet and handle topology variation. They used Tagged Bloom Filters to store information on incoming packets when they cross the network routers. The technique is tolerant to adversaries, that try to mislead the investigation by injecting false information. It allows suspicious areas, called hotspots, where some adversaries may reside, to be detected. Kim and Helmy [16] used small worlds in MANET, and base the traceback scheme on traffic pattern and volume matching. Despite its significant results, the proposed scheme is not suitable for a precise tracking of the mobility of intermediate nodes

and attack path variation. In a previous work [17], we proposed a cooperative observation network for the investigation of attacks in mobile *ad hoc* networks. A set of randomly distributed nodes, in charge of collecting and forwarding evidence, are deployed to monitor node mobility, topology variation, and patterns of executed actions. While the article took interest in the assembly and analysis of evidence, and identification the reconstruction of the potential executed attack scenarios, the algorithms it proposes do not follow a formal technique that generates irrefutable results, do not allow the generation of scenarios along with guarantee of reliability and correctness, and do not integrate an efficient tool for a mechanical proof of properties. Describing the generation of scenarios in a formal manner so that the results will be more reliable and rigorous is of paramount importance. Using theorem proving techniques, for example, will allow inferring theorems describing the root cause of the incident and steps involved in the attacks.

Investigating Attacks in Wireless Networks

In this section, we identify the requirements to be fulfilled by a digital investigation scheme suitable to support attack scenarios reconstruction in wireless networks. After that, we describe the characteristics of an investigation-prone MASNet.

Requirements for an efficient digital forensic investigation in MASNets

Defining a framework for digital investigation in wireless networks, especially sensor and *ad hoc* networks, turns out to be more tricky and challenging than in wireline networks. To do so, a set of requirements should be fulfilled.

First, attacks are mobile, meaning that during an attack scenario, the attacker can change its identity, position, location, and point of access. Using a formal model of digital investigation in wireless networks should integrate such mobility-based information when modeling actions in the attack scenario. Keeping track, for every user, the history of values taken by these parameters is important to trace mobile attacks. Additionally, contrary to wireline networks where intermediate routers are in most cases supposed to be trusted, usually all nodes in the networks can participate in forwarding datagrams from the source to the destination nodes, giving rise to several types of network attacks. Therefore, digital evidence should be collected at distributed locations within the network.

Second, to efficiently collect the mobility-based information, a set of trusted nodes should be distributed over the network and used for that purpose. These nodes, which we call *observers*, should be equipped with a set of mechanisms and solutions useful to supervise, log, and track events related to node movement, topology variation, roaming and IP handoff, and cluster creation, splitting and merging. Especially, in wireless sensor networks, observer nodes should be equipped with additional computational, energy, and communication resources in comparison with regular nodes in the network, so that they can: (a) process and buffer the generated evidence when no route could be established to forward them to the node in charge of analyzing the collected evidence; (b) reduce the number of scheduled active-sleep cycles, especially for sensor networks; and (c) have a long-range wireless power transmission and reception system so that they can monitor data exchange within a wide area in the network. The security of observer nodes should be strengthened as they store and process sensitive information in the form of evidence.

Third, as observer nodes are distributed over the network and under mobility, an occurring event may be: (a) detected and reported by all observers in the network, (b) detected and reported by a subset of observer nodes, since some of them are out of the communication range of the attacker, the victim, and the intermediate nodes which route the attack traffic, or (c) totally unobserved as the attack propagation zone was not covered by any observer during the attack scenario occurrence. In fact, the observers positions may not be located within the attack zone, or the observers may exist within such a zone but are sleeping. To efficiently investigate an attack scenario, mechanisms for correlating, filtering, and aggregating the collected events should be developed. The aim of these mechanisms is to eliminate any redundant information that can be determined by different generated evidence, collect missing information in them, and complete it from other observations.

Fourth, typically the investigation of an attack requires a secure delivery of observations to a central investigation node. However, due to mobility effects, the establishment of a routing path between an observer and the central investigation node may not be guaranteed. Therefore, choosing any observer node in the network (based, for instance, on the availability rate of its computational resources, or the degree of its connectivity to other observer nodes that have observed the traffic related to the attack) to be in charge of collecting observations and investigating the attack, is of high interest. While the use of distributed approaches for the analysis of evidence could provide tolerance to reachability problems, the use of a centralized approach allows reducing the effect of false positives and negatives. In fact, the more evidence, fewer potential attack scenarios are generated during investigation; using a distributed approach will lead observer nodes to generate a wide set of false

positive scenarios. Additionally, using a centralized approach helps better detecting and eliminating false evidence, by performing an efficient correlation of all collected evidence, avoiding thus false negative scenarios.

Fifth, some malicious events, part of an attack scenario, may target the network layer and therefore do not generate evidence in the system. Conversely, some of the events that compromise the system, are invisible to the network security solutions. In fact, some local actions may be triggered by the execution of remotely actions on the target system. Or even some local actions may be executed by the target system as a response to a remote executed action. Providing suitable mechanisms to correlate all types of evidence (network, system, and storage), handle incompleteness in them, and characterize provable system properties is of utmost importance.

Sixth, in wireless sensor networks, nodes may go into sleep mode to save energy [18]. In this case, they do not participate in broadcasting the datagram they receive. Observer nodes should take into consideration this feature and avoid detecting sleeping nodes as malicious. In the case where observer nodes are sleeping they could not contribute in relaying the received traffic or generating alerts, nor they generate or collect evidence.

Finally, to prove attack scenarios starting from incomplete evidence, a formalism for hypothesis generation should be developed to provide tolerance to missing information. The latter allows the investigation of scenarios which include unknown techniques of attacks, or use incomplete evidence. Hypothetical actions could be generated based on knowledge of the system behavior in response to user actions.

Characteristics of the investigated MASNet

The mobile *ad hoc* or sensor network, which we consider in this work, is composed of two types of nodes which are randomly deployed over the network and under mobility, namely user nodes, and observer nodes. A user node can be a malicious or a legitimate node, and may also be the target of the attack scenarios. Typically, in wireless *ad hoc* networks, user devices can dynamically connect and disconnect to the network, making their number variable. Observer nodes form a network of observation and are responsible for:

- maintaining a library of known attacks and their patterns;
- generating, for every pair of communicating user nodes, digital evidence containing information on the remotely executed actions and values of some parameters extracted from the datagrams sent by the attacker;
- securely sending and forwarding evidence generated by other observers to the node in charge of investigation.

The node in charge of investigation can be any observer node which is chosen, based for instance on the distance separating observers to the attacker node, to:

- securely collect observations from the remaining observer nodes and the compromised node;
- correlate and merge collected evidence;
- reconstruct and identify possible attack scenarios satisfying the obtained evidence;
- generate hypotheses regarding the undetected actions.

Depending on the sensitivity of the traffic exchanged between nodes, the observer nodes can be special nodes in charge of observation or any user node endowed with extra investigation and evidence-collection based functions. We believe that, for efficiency of observation and investigation, the network of observers is appropriate. Knowing that if the nodes in the MASNet are sufficiently dense in a special area, the size of the observer network would be smaller than the number of nodes in the MASNet with a factor of $\frac{R}{r}$ where R and r are the communication radius of observer nodes and user nodes, respectively. An interesting value of $\frac{R}{r}$ would vary from 2 to 4, allowing the observer to cover at least two hops and reducing the portion of nodes to equip with extra resources to less than 2%.

Two security levels are assumed. The first level is related to mobile devices which can either be legitimate or malicious. The second level is related to observers and the central investigation node which manipulate very sensitive information (i.e., the digital evidence). The latter are designed to be highly secured, trusted, and able to communicate securely. To do so, a set of key credentials are securely distributed and stored in each node during the system initialization, and a set of cryptographic protocols are used. Properties such as authentication, secrecy, non-repudiation, and anti-replay are assumed to be guaranteed, preventing attackers from spoofing, altering, or replaying data exchanged between observers. These data include evidence and analysis output in addition to routing information. This assumption goes with the required characteristics of the observer nodes that we enunciated in the previous section.

All network links are supposed to be bidirectional allowing an observer node to continuously monitor the network while delivering its observations to the central investigation nodes. The probability of datagrams collisions is reduced to its lowest value. All observer nodes

are supposed to overhear traffic within their transmission range. Their interfaces operate in promiscuous mode to monitor traffic of neighboring nodes [19]. For every node in the network a list of neighbors is supposed to be available. A secure neighbor discovery protocol could be used for that purpose.

Modeling Wireless Attack Scenarios

We describe in this section a model for describing attack scenarios, digital evidence, and the security solutions that generate them. When an attack scenario is remotely executed, the impact at the network and the target system is different. At the network level, several datagrams are generated and forwarded to execute the remote actions of the scenario. The information visible by observer nodes, which are deployed in the network to monitor the exchange of these datagrams between intermediate nodes, is in the form of datagrams. These datagrams allow the executed actions to be determined, and do not provide a precise idea on how the system behaves when it executes it. At the end-system level (i.e., the target), actions are executed by the operating system, leading to modifications of the system components. The information visible by the security solutions at these systems is typically in the form of log and alert files, which only show the impact of the executed action and not the action itself. The evidence to collect on the target system will be modeled in the form of observations over executions (i.e., attack scenarios).

Modeling attack scenarios from the system viewpoint

We consider a system specification Spec that models the investigated system by a set of variables $\mathcal{V}$ and a library of elementary actions $\mathcal{A}$ containing suspicious and legitimate actions. A system state $s \in \mathcal{S}$ is a valuation of all variables in $\mathcal{V}$. It can be written as $s = (\nu_1[s],..., \nu_n[s])$, where $\forall i \in [1..n] : \nu_i \in \mathcal{V}$ and $\nu_i[s]$ is the value of variable ν_i in state s. A system action $A \in \mathcal{A}$, denotes the event to be executed on the specified system. It describes for every variable ν in $\mathcal{V}$ the relation between its value in the previous state, say s, and its value in the new state, say t. $A(s, t)$ = true, iff action A is enabled in state s and the execution of action A on state s would produce state t.

A wireless attack scenario, say ω, such that $\omega \in \Omega$ is generated by sequentially executing a series of actions in $\mathcal{A}$, starting from an initial state, say s_0, letting the system move to a state, say s_n, along by a series of intermediate states. Formally, we define a system execution ω in the following form $\omega = \langle s_0, A_1, s_1,..., s_{n-1}, A_n, s_n\rangle$, where:

- $\forall i \in [0..n] : (A_i \in \mathcal{A})$;
- $\forall A_i \in \mathcal{A},\ i \in [1..n] : \{A(s_{i-1},\ s_i) = \text{true}\}$.

An execution $\omega = \langle s_0, A_1, s_1,..., A_n, s_n\rangle$ can be written as $\omega = \omega_x|\omega_y$, where $\omega_x = \langle s_0, A_1, s_1,..., A_i, s_i\rangle$ and $\omega_y = \langle A_{i+1}, s_{i+1},..., A_n, s_n\rangle$ for $i \in [1, n-1]$. We denote by ω^{act} the series of actions obtained from ω after deleting all system states, and by ω^{st} the series of system states obtained from ω after deleting all executed actions.

Actions parts of ω^{act} are locally or remotely executed on the target system. Typically, local execution is done when a local action on the target system is triggered by the remote execution of a script. An action could also be executed locally as an automated response of the target system (or the deployed security solutions) to the execution of some malicious action. We denote by $\omega^{act|rem}$ the series of remote actions obtained from ω^{act} after deleting local actions, and by $\omega^{act|loc}$ the series of local actions obtained from ω^{act} after deleting remote actions.

Modeling security solutions and system evidence

We consider an observation function obs( ) over states, and attack scenarios. It allows the characterization of security solutions used to monitor the investigated system. The output of obs( ) represents the evidence generated by the related security solution. Such evidence will only show incomplete information regarding the executed actions and the description of the system states generated consequently.

We define the observable part of a state s, as obs(s) = $[l(\nu_1[s]), l(\nu_2[s]),..., l(\nu_n[s])]$ where l( ) represents a labeling function, that is used to assign to $\nu_i[s]$, a value equal to one of the following three, depending on the ability of the security solution to monitor the system variables

- $\nu_i[s]$: The variable ν_i is visible and its value can be captured by the observer. The variable value is thus kept unchanged.
- *A fictive value ε such that $\varepsilon \notin$ Val* (Val represents the set of values which could be taken by variables with regard to the system specification). The variable is visible by the observer but the variation of its value does not bring it any supplementary information (e.g., the observer is monitoring a variable value which is encrypted). The variable value is transformed to a fictive value ε.
- *An empty value, denoted by $\varnothing$*: The variable is invisible, such that none information regarding its value could be determined by the observer.

Note that $l(\nu_i[s])$ can be defined in a conditional form letting it depend on the value of an additional predicate (e.g., the value of variable ν cannot be visible is some state s, unless another variable, say ν', takes a special value in that state).

Given an attack scenario $\omega = \langle s_0, A_1, s_1,..., s_{n-1}, A_n, s_n\rangle$, we define the observable part of ω, by obs(ω). obs(ω) is

computed in two stages. First, by letting $\overline{\text{obs}(\omega^{st})}$ be the sequence obtained from $\omega^{st} = \langle \text{obs}(s_0),..., \text{obs}(s_n)\rangle$ after replacing each state s_i by obs(s_i). obs(ω) is obtained from $\overline{\text{obs}(\omega^{st})}$ by replacing any maximal sub-sequence $\langle \text{obs}(s_i),..., \text{obs}(s_j)\rangle$ such that obs(s_i) = ... = obs(s_j) by a single state observation, namely obs(s_i). The evidence to be collected by a security solution when an attack scenario, say ω, is executed, will be equal to obs(ω), which is computed with respect to the labeling function that characterizes that solution. Note that, an observation over an execution becomes an evidence when it is generated by a trusted observer, communicated and exchanged securely over the networked systems, and retrieved using the legal procedures that are admissible in a court of law.

The intermediate steps followed to compute obs(ω) are based on that fact that:

- the great majority of installed security solutions are able to monitor the system behavior resulting from the execution of an action and not the executed action itself;
- if successive states have the same observation, an observer of the execution is not able to distinguish whether the system has progressed from a state to another or not.

Definition 1. (the $\sqsubseteq$ relation)

Given two evidence, say O and O', where $O = \langle o_1,..., o_m\rangle$, $O' = \langle o'_1, \ldots, o'_n\rangle$, and $m < n$. We have:

$$O \sqsubseteq O' \Leftrightarrow \exists x = m \text{ such that : } o_1 = o'_1, \ \ldots, \ o_m = o'_x$$

Informally, the relation $O \sqsubseteq O^{'}$ means that the evidence O is included in the evidence $O^{'}$ and appears in it starting from the beginning.

Definition 2. (The *idx*( ) function)

Given an attack scenario $\omega = \langle s_0,..., s_n\rangle$, a security solution defined by the observation function obs( ), and an evidence $O = \langle o_1,..., o_m\rangle$ generated by that solution such that obs(ω) = O. We have

$$\forall s \in \omega : \{(idx(s, O) = i) \Leftrightarrow obs(s) = 0_i\}$$

Informally, function *idx* (s, O) takes as input a state and an evidence and returns the index of the observation of that state in O.

Definition 3. (The *satisfied* relation)

Given a security solution which is defined by the observation function obs( ), and an evidence e generated by that solution when an attack scenario, say ω, was conducted on the system (i.e., obs(ω) = e). A scenario, say ω', is *satisfied* by the evidence e if and only if: obs (ω') $\sqsubseteq e$.

Example 1. We consider a system modeled by two variables, namely ν_1 and ν_2. Variable ν_1 represents the state of a service, say Srv. It can take value 0 or 1 to mean that the service is down or up, respectively. Variable ν_2 represents the size (in bytes) of the buffer from which the service Srv reads the user commands. It can take any integer value between 0 and 2, where 2 is the buffer size limit. We consider a library of elementary actions composed of two actions, namely A_1 and A_2. Action A_1 consists of stopping the service. It sets the value of variable ν_1 to 0. Action A_2 consists of typing a specific user command whose size is equal to 1 byte. It is only enabled if the value of variable ν_2 is less than or equal to 2. If the value of ν_2 is strictly less than 2, only the value of variable ν_2 in the new state is set to 1 greater that its value in its old state. If the value of variable ν_2 is equal to 2, its value is kept unchanged while the value of variable ν_1 becomes equal to 0 (the buffer is overloaded.

Consequently ν_2 remains equal to 2 while the service becomes unexpectedly down). A state s, which is a valuation of the two variables ν_1 and ν_2, is represented as ($\nu_1[s]$, $\nu_2[s]$). The initial system state, say s_0, which is equal to (1, 0) denotes that the service is running and the buffer is empty. We consider two scenarios. The first, say ω_1, which represents administratively shutting down the service, consists in executing action A_1 only. The second, say ω_2, which represents a buffer overflow attack against the running service, consists in executing action A_2 twice. We have:

- $\omega_1 = \langle (1, 0), A_1, (0, 0)\rangle$
- $\omega_2 = \langle (1, 0), A_2, (1, 1), A_2, (0, 2)\rangle$

We consider two security solutions deployed on the considered system. The first allows monitoring of variable ν_1 only and is described by the observation function obs$_1$( ), while the second allows monitoring of variable ν_2 only and is described by the observation function obs$_2$( ). The two observation functions obs$_1$( ) and obs$_2$( ) are characterized by labeling functions, say l_1( ) and l_2( ), respectively. We have:

- $\forall s$: $\{(l_1(\nu_1[s]) = \nu_1[s]) \wedge (l_1(\nu_2[s]) = \varnothing)\}$.
- $\forall$: $\{(l_2(\nu_1[s]) = \varnothing) \wedge (l_2(\nu_2[s]) = \nu_2[s])$.

The digital evidence generated by the first security solution if ω_1 are ω_2 are executed, are equal, respectively, to:

- obs$_1(\omega_1) = \langle \text{obs}_1(1, 0), \text{obs}_1(0, 0)\rangle = \langle (1,\varnothing), (0, \varnothing)\rangle$
- obs$_1(\omega_2) = \langle \text{obs}_1(1, 0), \text{obs}_1(1, 1), \text{obs}_1(0, 2)\rangle = \langle (1, \varnothing), (0, \varnothing)\rangle$

The digital evidence generated by the second security solution if ω_1 and ω_2 are executed, are equal, respectively, to:

- $obs_2(\omega_1) = \langle obs_2(1, 0), obs_2(0, 0)\rangle = \langle(\varnothing, 0)\rangle$
- $obs_2(\omega_2) = \langle obs_2(1, 0), obs_2(1, 1), obs_2(0, 2)\rangle = \langle(\varnothing, 0), (\varnothing, 1), (\varnothing, 2)\rangle$

According to the obtained observations, the first security solution, which is modeled by the observation function $obs_1(\ )$, would not differentiate between the two executed scenarios. In other words, an investigator, which tries to reconstruct the potentially occurred scenarios based on the evidence generated by $obs_1(\ )$, should consider that the two scenarios ω_1 and ω_2 are potential. This is not the case for the evidence generated by the observation function $obs_2(\ )$, where each one of the two scenarios produces a different observation.

Modeling attack scenarios from the network viewpoint

From the network viewpoint, an attack scenario ω creates a series of network datagrams, say π, sent from the attacker host to the victim host over the MASNet, in order to remotely execute actions in $\omega^{act|rem}$. Formally, $\pi = \langle p_0, p_1,..., p_n\rangle$ where every $p \in \pi$ represents a network datagram and is a valuation of six variables, namely, ip_s, ip_d, rp, ttl, loc, and A. The first five variables represent the source IP address related the attacker node, the destination IP address related to the victim node, the routing path which is composed of the ordered set of identities related to nodes used to forward the packet, the initial Time To Live value of the generated packet, and the location of the node when it sends the datagram, respectively. The last variable A represents a global action as two-tuple information, say (act, dgt). The first information, which is act, stands for the action remotely executed by the attacker on the target system. The second information, which is dgt, represents the digest of the packet sent to remotely execute action act. The digest is computed over the immutable fields of the IP header and portion of the payload [20], respectively. We denote by A.act and A.dgt the value of the executed action and the packet digest related to the global action A, respectively. Among the fields in the packet header and portion of the payload, over which the digest is computed is the IP identification field. The latter is expected to change from one generated packet to another. Therefore, it enables distinguishing between the two situations:

- the attacker executes the same action twice, leading to the generation of two packets containing the same action but a different digest;
- the attacker generates the action only one time, but the packet generated to remotely execute it was observed by different observers and therefore two pieces of evidence are obtained, which are related to a single executed action.

Even if the attacker could try to mislead investigation, by executing the action twice while setting the packet fields to be similar in the two generated datagrams (the aim is to lead the central investigation node to discard one copy), this malicious behavior could be detected. In fact, when an observer detects that a node is forwarding the same copy of the packet twice, it generates an alert to inform the central investigation node, and creates a separate evidence for the second copy of the packet so that the two executed actions will be part of two different global actions.

In *ad hoc* networks the identity of the attacker may change when it changes its point of attachment. In this work, we suppose that every pattern (created by remotely executed actions) in the network datagram is associated with a unique action in the library of elementary system actions. Due to the dynamic aspect of the network topology the set of datagrams, which are sent by the attacker to remotely execute actions, may follow different routing paths.

Modeling wireless network evidence

Let ω be an executed attack scenario, and π be the series of datagrams sent by the attacker to remotely execute actions in ω^{rem}. Since observer nodes are mobile, they may go out of the transmission range of the attacker, the victim, or the intermediate nodes which participated in routing the traffic. Moreover, in the context of sensor networks, nodes are scheduled to sleep and wake-up to save energy without compromising the system functionality. Consequently, an observer node will only be able to:

- detect from π a sub-series containing only datagrams that went across its coverage. In fact, some datagrams in π may be invisible by the observer due to its position (i.e., the position of the observer node does not allow it to receive the forwarded datagram), or it status (i.e., the observer is sleeping when the datagram is forwarded);
- store from that sub-series the observable part, which will be provided as network evidence. The observer is assumed to specify its location in the network when it captured the packet.

The network observation of the series of datagrams π, which is sent by the attacker to remotely execute actions

in ω^{rem}, is computed based on the observation of candidate datagrams. It is obtained in two stages. First, by transforming π to $\bar{\pi}_j$ after deleting datagrams which were not transmitted within the coverage of the observer j. Second, by replacing every packet p in $\bar{\pi}_j$ by $\text{obs}_j(p)$.

Let π be the series of datagrams sent to remotely execute actions within some attack scenario, where $\bar{\pi}_j = \langle p_0, .., p_m \rangle$ is the series of datagarms in π which were captured by some observer j. We have:

$$\text{obs}_j(\pi) = \text{obs}_j(\bar{\pi}_j) = \langle \text{obs}(p_0), \ \ldots, \text{obs}(p_m) \rangle \tag{1}$$

$$\forall p \in \bar{\pi}_j : \{\text{obs}\ (p) = [l(ip_s[p]),\ l(ip_s[p]),\ l(rp[p]),\ l(\text{TTL}\ [p]),\ l(loc[p]), l(A[p])]\} \tag{2}$$

The computed labels comply with the following rules:

- $l(ip_s[p])$ and $l(ip_d[p])$ are equal to $ip_s[p]$ and $ip_d[p]$, respectively, since the IP source and destination address of the attacker are always interpretable. In fact, to be efficiently routed by an intermediate node, every packet should have these two addresses in a clear format.
- $l(rp[p])$ is obtained from $rp[p]$ after deleting the identities of intermediate nodes which cannot be determined. Typically, only the identities of intermediate nodes which are in the coverage of the observer node could be determined as the observer is monitoring the forwarding of datagrams. Nevertheless, if the packets are source routed, the observer could determine the full identities of nodes in rp.
- $l(\text{TTL}[p])$ is equal to to the value returned by TTL $[p]$. In fact, the TTL value can always be read from the packet header. However, since this value decreases when the packet is routed from one node to another, the value to be included in the evidence will be the one observed in the packet when it appears in the first time in the coverage of the observer.
- $l(\text{loc}[p])$ strongly depends on the techniques and model chosen to represent the location (i.e., GPS, Bluetooth, RFID). It is equal to $\text{loc}[p]$ if the attacker is in the coverage of the observer node and the latter has the possibility to determine its exact position. It is equal to $\varnothing$ if the attacker is out of the observer coverage.
- $l(A[p])$ is equal to $(A.\text{act}[p], A.\text{dgt}(p))$ if the pattern of the executed action in datagram is readable and can be determined. If the traffic is encrypted, or the pattern of the action is unknown, $l(A[p])$ is equal to $\varnothing$.

Other information of interest can be added to the observation generated by network observers such as the observer's position in the network, or its list of neighbors. All of this information would be useful during the correlation of the collected evidence.

In Wireless Sensor Networks, when the observer is going to sleep during the observation of the packets related to the attack scenarios, it inserts the symbol ε in the network evidence to denote that some packets may not have been observed due to weak-up/sleep cycles.

Given a packet p, we denote by p_A the tuple of information composed of the packet digest and the remotely executed action. Formally $p_A = (\text{act}[p], \text{dgt}[p])$ where p_A is called a global action. We denote by $p_A.\text{act}$ and $p_A.\text{dgt}$ the action and the packet digest, respectively.

Definition 4. (last index function, *lidx*( ))

Given the network evidence $\Pi = \langle A_1, \ldots, A_m \rangle$ in the form of a series of global actions and an attack scenario $\alpha = \langle s_0, a_1, s_{;1}, \ldots, a_n, s_n \rangle$. We have:

$$lidx\ (\alpha,\ \Pi) = i \Leftrightarrow (\exists x \in [1..n] : \{a_x = A_i.\text{act}\}) \wedge (\forall \gamma \in]x..n] : \{\nexists A \in \Pi \text{ such that } a_\gamma = A.\text{act}\}) \tag{3}$$

Informally, the definition states that function *lidx*( ) takes as input an attack scenario and a network evidence as a series of global actions. It returns the index (in the network evidence) of the last action in the attack scenario which is mentioned by the global action in the network evidence. With respect to example 1. For the network evidence $\Psi = \langle A_1 A_3 A_2 A_3 \rangle$, we have $lidx(\omega_2, \Psi) = 3$

Conducting Proofs in the Wireless Context

We propose a deduction system which is described using a set of inference rules. For the sake of space, we settle for only describing those that have to be inevitably used to generate proofs. An investigator is assumed to have a complete knowledge of the specification of the investigated system (i.e., description of all possible initial system states, system variables, and a library of elementary actions). Let ω be the attack scenario executed to compromise the system, π be the series of datagrams sent by the attacker to remotely execute actions in ω^{rem}, SO be the set of observer nodes deployed on the system (i.e., system security solutions), NO be the set of observer nodes deployed on the network (i.e., network security solution), $\mathcal{O}$ be the set describing the observation functions of the system observers and the evidence they collected, and $\mathcal{E}$ be the set describing the observation functions of the network observers and the evidence they collected. We denote by $\text{obs}_i(\ )$ the observation function which characterizes the ith security solution (i.e., the ith observer), and O_i be the evidence generated by that solution. We have:

$$\begin{cases} \mathcal{O} = \cup_{i \in SO}\{ (O_i, \text{obs}_i())\} \\ \mathcal{E} = \cup_{j \in NO}\{O_j, \text{obs}_j()\} \\ \forall i \in I : \{\text{obs}_i(\omega) = O_i\} \\ \forall j \in J : \{\text{obs}_j(\pi) = O_j\} \end{cases}$$

In the sequel, we denote by Π the aggregated network evidence, as a sequence of global remote actions. It is computed using network evidence collected from the observer nodes in the network. The sets *InSt* and $\mathcal{A}$ will describe all the possible initial system states, and the library of actions, respectively.

Rules for aggregating the network evidence

Rule 5 appends to the aggregated evidence under construction Π, which is already empty, the sequence of global actions extracted from a network evidence, say E. The evidence E represents the longest one, in terms of observed packets, in the set of available network evidence in Π. The operator $\lceil\ \rceil$ extracts from the sequence of packets observations, in a network evidence, the sequence of global actions. Function Len( ) computes the length of a network observation in terms of packets observations.

$$\frac{\Pi = \emptyset,\ \exists E \in \mathcal{E} \text{ such that } \{\forall (E' \in \mathcal{E}) \wedge (E' \neq E) : \{\text{Len}(E') \leq \text{Len}(E)\}\}}{\Pi = \Pi \cup \lceil E \rceil} \quad (5)$$

In the sequel, rules 6 and 7 aim to detect the missing global actions in the aggregated network evidence Π and try to retrieve them from the other available network observations. Obviously, as outlined previously, network observers may not capture the same packets and every collected obs $(\bar{\pi})$, related to the same sent series of datagrams π, will be different from one observer to another.

Rule 6 locates a pair of consecutive global actions, say A_i and A_{i+1}, in the aggregated network evidence Π, which exist in another network evidence $E \in \mathcal{E}$ but are separated by a sub-sequence of global actions. Typically, this sub-sequence did not exist in Π due to a potential variation of the network topology during the observation of the attack scenario. This variation could be detected by comparing the TTL or routing path value in the two observed packets containing A_i and A_{i+1}. The rule inserts between A_i and A_{i+1} the series of global actions retrieved from the missing sub-sequence (in Π) of packet observations.

This insertion is performed when the observer, which generated the network evidence E, detected a modification in the TTL or routing path through the packet observations of the missing sequence.

$$\frac{\begin{array}{c}\Pi = \langle A_1, \ldots, A_i, A_{i+1}, \ldots, A_n\rangle,\ E \in \mathcal{E},\ \langle e_x, \ldots, e_y\rangle \in E, \\ ((\lceil e_x \rceil = A_i) \wedge (\lceil e_y \rceil = A_{i+1}) \wedge (y > x+1)), \\ (e_{x+1}.\text{ttl} \neq e_x.\text{ttl}) \vee (e_{x+1}.rp \neq e_x.rp))\end{array}}{\Pi = \langle A_1, \ldots, A_i\rangle \lceil \langle e_{x+1}, \ldots, e_{y-1}\rangle \rceil \langle A_{i+1}, \ldots, A_n\rangle} \quad (6)$$

Rule 7 locates two non-consecutive global actions, say A_i and A_j , in the aggregated network evidence Π, which are separated differently by a different sequence of actions in some available network evidence, say E, containing the two global actions A_i and A_j . Let the two sub-sequences of global actions, separating A_i and A_j in Π and E, be denoted by S and S', respectively.

The aggregated network evidence under construction is updated by transforming the sub-sequence S into a new sub-sequence composed of actions from S and S'. Function Cmb takes as input two sub-sequences of global actions (in this rule S and S' are chosen as input) and transforms them into a sub-sequence, say S'', composed of actions from S randomly inserted between actions from S'. The order of appearance of actions in S and S' is maintained in S''. This rule allows capture of the situation, where the two mobile observers which observe packets in Π and E, move at the same time instants, so that each datagram sent by the attacker is captured by only one of them.

$$\frac{\begin{array}{c}\Pi = \langle A_1, \ldots, A_i, \ldots, A_j, \ldots, A_n\rangle, \\ \exists E \in \mathcal{E} \text{ such that : } (\langle e_x, \ldots, e_y\rangle \in E) \\ \wedge (e_x.\text{Act} = A_i) \wedge (e_y.\text{Act} = A_j) : \{\lceil \langle e_{x+1}, \ldots, e_{y-1}\rangle \rceil \cap \langle A_{i+1}, \ldots, A_{j-1}\rangle = \emptyset\}\end{array}}{\Pi = \langle A_1, \ldots, A_i\rangle \text{Cmb}\ (\langle A_{i+1}, \ldots, A_{j-1}\rangle, \lceil \langle e_{x+1}, \ldots, e_{y-1}\rangle \rceil) \langle A_j, \ldots, A_n\rangle} \quad (7)$$

Rule 8 allows update of the aggregated network evidence after determining whether the observer slept and woke up between the observation of two packets. If it is the case, it tries to locate the sub-series of packets observations in other collected network evidence, from which global actions can be extracted and inserted immediately after the action observed before the observer slept, and immediately before the action observed when the observer woke up,

$$\frac{\begin{array}{c}\Pi = \langle A_1, \ldots, A_i, \varepsilon, A_{i+1}, \ldots, A_n\rangle, \\ \exists E \in \mathcal{E},\ \langle e_x, \ldots, e_y\rangle \in E \text{ such that : } \{(([e_x] = A_i) \\ \wedge (\lceil e_y \rceil = A_{i+1}) \wedge (y > x+1))\end{array}}{\Pi = \langle A_1, \ldots, A_i\rangle \lceil \langle e_{x+1}, \ldots, e_{y-1}\rangle \rceil \langle A_{i+1}, \ldots, A_n\rangle} \quad (8)$$

Rule 9 tests whether all the global actions, which were extracted from the collected network evidence, were included in the aggregated network evidence under construction. $\overline{\Pi}$ stands for the aggregated network evidence containing all actions provided by the evidence in Π.

$$\frac{\forall E \in \mathcal{E} \ :\ e \in E \Rightarrow e.Act \in \Pi}{\overline{\Pi} = \Pi} \quad (9)$$

Rules for ensuring that an attack scenario is satisfied by system evidence

Rule 10 states that an attack scenario, which is composed of a single state (i.e., the initial system state), is

coherent with a set $\mathcal{O}$ describing the observation functions of the system observers and the evidence they collected, if: (a) it is satisfied by all available system evidence; and (b) the observation of that state represents the first element in the available observations.

$$\frac{s \in InSt,\ \forall (O, \text{obs}()) \in \mathcal{O} : \{(\text{obs}(s) \sqsubseteq O) \wedge (idx(s,\ O) = 1)\}}{\langle s \rangle \text{is coherent with } \mathcal{O}} \tag{10}$$

Rule 11 states that an attack scenario α, which can be written as the concatenation of two fragments is coherent with a set $\mathcal{O}$ describing the observation functions of the system observers and the evidence they collected if: (a) the first fragment is coherent with $\mathcal{O}$; and (b) for every system observer and the evidence e that it generates, the second fragment should be satisfied by the remaining part of the evidence obtained after eliminating the content which covers the first fragment.

$$\frac{\begin{array}{c}\alpha' = \langle s_0,\ a_1,\ s_1,\ \ldots,\ a_i,\ s_i \rangle,\ \alpha = \alpha' \langle a_j,\ s_j \rangle, \\ \alpha' \text{is coherent with } \mathcal{O},\ \forall\ (O, \text{obs}()) \in \mathcal{O} \wedge (O = \langle o_1,\ \ldots,\ o_n \rangle) \\ \wedge (idx(s_i,\ O) = x) : \{\text{obs}(s_i, a_j,\ s_j) \sqsubseteq \langle o_x,\ \ldots,\ o_n \rangle\}\end{array}}{\alpha \text{ is coherent with } \mathcal{O}} \tag{11}$$

Rules for generating possible attack scenarios based on network and system evidence

Rule 12 states that an attack scenario composed of a single state, which is coherent with regard to a set $\mathcal{O}$ describing the observation functions of the system observers and the evidence they collected, is possible.

$$\frac{\begin{array}{c}\alpha = \langle s \rangle, \\ \alpha \text{ is coherent with } \mathcal{O}\end{array}}{\alpha \text{ is possible w.r.t. } \mathcal{O}} \tag{12}$$

Given an attack scenario α which is composed of two fragments where the first fragment is also a scenario composed of several states and actions, and the second fragment contains only an action and the state it generates if it is executed from the last state in the first fragment, Rule 13 proves that such an attack scenario is possible if: (a) the first fragment is possible; (b) the index (in the aggregated network evidence) of the action provided by the second fragment, is one higher than the index (in the aggregated network evidence) of the last remote action provided by the first fragment; and (c) the attack scenario is coherent with the set $\mathcal{O}$ describing the observation functions of the system observers and the evidence they collected

$$\frac{\begin{array}{c}\alpha' = \langle s_0,\ a_1,\ s_1,\ \ldots,\ a_{n-1},\ s_{n-1} \rangle,\ \alpha = \alpha' \langle a_n,\ s_n \rangle, \\ \alpha' \text{ is possible w.r.t. } \mathcal{O},\ lidx(\alpha',\ \overline{\Pi}) = x, \\ A \in \overline{\Pi} \text{ such that } : \{idx(A,\ \overline{\Pi}) = x + 1)\}, \\ a_n = A.\text{act}, a_n(s_{n-1},\ s_n) = \text{true}, \\ \alpha \text{ is coherent with } \mathcal{O}\end{array}}{\alpha \text{ is possible w.r.t. } \mathcal{O}} \tag{13}$$

Given an attack scenario α which is composed of two fragments where the first fragment is also a scenario composed of several states and actions, and the second fragment contains only an action and the state it generates if it is executed from the last state in the first fragment, Rule 14 proves that such an attack scenario is possible if: (a) the first fragment is possible; (b) the action in the second fragments does not exist in the aggregated network evidence (i.e., the action represents a locally executed action on the remote system, or a remotely executed action which was not captured by any deployed observer due to mobility effects) or does not correspond to the one that exists just after the last observed remote action of the first fragment; and (c) the attack scenario is coherent with a set $\mathcal{O}$ describing the observation functions of the system observers and the evidence they collected. The rule can be used to append a hypothetical action (action a_n in the rule) to the scenario under construction to alleviate any incompleteness of information in the aggregated network evidence.

$$\frac{\begin{array}{c}\alpha' = \langle s_0,\ a_1,\ s_1,\ \ldots,\ a_{n-1},\ s_{n-1} \rangle,\ \alpha = \alpha' \langle a_n,\ s_n \rangle \\ \alpha' \text{ is possible w.r.t. } \mathcal{O},\ a_n \in \mathcal{A}, \\ (\nexists\ A \in \overline{\Pi} \text{ such that } : \{(idx(A,\ \overline{\Pi}) = lidx(\alpha',\ \overline{\Pi}) + 1) \wedge (A.\text{act} = a_n)\}, \\ a_n(s_{n-1},\ s_n) = \text{true},\ \alpha \text{ is coherent with } \mathcal{O}\end{array}}{\alpha \text{ is possible w.r.t. } \mathcal{O}} \tag{14}$$

Rules for generating provable actions and scenarios

Rule 15 states that the last action executed in an attack scenario, is provable if both the attack scenario fragment, say α, after which it is executed and the attack scenario fragment obtained after its execution are possible. The executed action should exist in the aggregated network evidence and correspond to the one that immediately succeeds the last remote action executed in the first fragment.

$$\frac{\begin{array}{c}\alpha' = \langle s_0,\ a_1,\ s_1,\ \ldots,\ a_{n-1},\ s_{n-1} \rangle,\ \alpha = \alpha' \langle a_n,\ s_n \rangle,\ A \in \overline{\Pi}, A.\text{act} = a_n \\ idx(A,\ \overline{\Pi}) = lidx(\alpha',\ \overline{\Pi}) + 1, \\ \alpha \text{ is possible w.r.t. } \mathcal{O}\end{array}}{\alpha_n \text{ is provable}} \tag{15}$$

Rule 16 states that an hypothetical action, say a_n, which is executed from state s_{n-1}, is provable if: (a) action a_n could not be a remote action which is observed by network observers and located just after the last observed remote action in the scenario fragment preceding its execution; (b) its execution generates an attack scenario which is possible; and (c) by replacing that action by any other action from the library of attacks, the generated attack scenario would not be coherent with all available system evidence.

$$\frac{\begin{array}{c}\alpha' = \langle s_0,\ a_1,\ s_1,\ \ldots,\ a_{n-1},\ s_{n-1}\rangle,\ \alpha = \alpha'\langle a_n,\ s_n\rangle,\\ \alpha' \text{ is possible w.r.t. } \mathcal{O},\\ \nexists A \in \Pi : \{(A.\text{act} = a_n) \wedge (idx(A,\ \overline{\Pi}) = lidx(\alpha, \overline{\Pi}) + 1)\},\\ \forall \langle a',\ s'\rangle \text{such that } (a' \neq a) \wedge (a'(s_{n-1},\ s') = \text{true}) :\\ \{\neg(\alpha\langle a',\ s'\rangle) \text{is coherent with } \mathcal{O})\}\end{array}}{\alpha_n \text{ is provable}} \quad (16)$$

Rule 17 states that an attack scenario, say α, is provable if: (a) all the actions it contains are provable; (b) all the remote executed actions, which were observed by the network observers and included in the aggregated network evidence, are in the generated scenario α; and (c) the order of appearance of remote actions in the aggregated network evidence is maintained in the scenario α.

$$\frac{\begin{array}{c}\alpha \text{ is possible w.r.t. } \mathcal{O},\ \forall a \in \alpha : \{a \text{ is provable}\},\\ \forall \langle A,\ A'\rangle \in \overline{\Pi} : \{\exists \langle a_i,\ s_i,\ \ldots,\ a_j,\ s_j\rangle \in \alpha \text{ such that} (j > i)\\ \wedge (A.\text{act} = a_i) \wedge (A'.\text{act} = a_j)\}\end{array}}{\alpha \text{ is provable}} \quad (17)$$

Example 2. Given an attack scenario $\alpha = \langle s_0, a_1, s_1\rangle$, an aggregated network evidence $\overline{\Pi} = \langle A_1\rangle$ which was generated starting from the evidence collected from the different network observers, a set $\mathcal{O} = \{(o, obs())\}$ describing the observation function obs( ) of a system security solution and the evidence $O = \langle o_1, o_2\rangle$ it generated when the attack scenario α was executed, and a set of initial system states denoted by St. The aim is to prove that the scenario α is possible.

By hypothesizing that $s_0 \in$ St and obs(s_0) = o_1, Rule 10 can be used to prove that the scenario $\langle s_0\rangle$ is coherent with $\mathcal{O}$. This demonstration is followed by the use of Rule 12 to prove also that $\langle s_0\rangle$ is possible.

Since α can be written as $\langle s_0\rangle\ \langle a_1, s_1\rangle$, and by hypothesizing that $\langle s_0\rangle$ is coherent with $\mathcal{O}$ and obs(a_1, s_1) = o_2, we can use rule 11 to prove that α is coherent with $\mathcal{O}$. It suffices to consider that $idx(s_0, O) = 1$ and obs(s_0, a_1, s_1) $\sqsubseteq \langle o_1, o_2\rangle$.

Since $\langle s_0\rangle$ is possible with regard to O , $lidx(\langle s_0\rangle) = 0$, $A \in \Pi$, $lidx(A, \overline{\Pi}) = 1$, $A.\text{act} = a_1$, $a_1(s_0, s_1)$ = true, and α is coherent with $\mathcal{O}$, Rule 13 can be used to prove that α is possible with regard to $\mathcal{O}$.

Other rules of interest can be defined. For instance, if the observers append to the generated evidence their lists of neighbors, a rule can be added to aggregate network evidence by exploiting the difference between the list of neighbors recorded with the observation of two consecutive packets in the same attack scenario. The rules that we describe in this section allow the generation of two types of theorems, built from observations and true system and network evidence, that characterize:

- aggregated (merged, optimal, and maximal) network evidence using rules 5, 6, 7, 8, and 9;
- coherent, possible, and provable attack scenarios based on both system evidence and an aggregated network evidence, using rules 10, 11, 12, 13, 14, 15, 16, and 17. The more network evidence there is, the simpler is the proof and the fewer is the number of possible candidate attacks.

Methodology for Digital Investigation in Wireless *ad hoc* Networks

We propose in this section a methodology for formal digital investigation of security attacks in the context of mobile *ad hoc* and sensor networks, which is composed of four main steps. In the first step, the node in charge of investigation starts by securely collecting sufficient evidence from observer nodes and the compromised system. The collected network evidence should be filtered to discard those which are not related to the attack under investigation (e.g., exploit address of the victim and time period of evidence generation).

The second step consists in aggregating the set of collected network evidence to generate a merged, optimal, and maximal network evidence. To do that, rules 5, 6, 7, and 8 are applied until an aggregated network evidence is obtained that includes all packets observations contained in the collected network evidence. After that, the aggregated network evidence is transformed to a maximal network evidence using rule 9. When applying rule 7, several possible sequences of events could be obtained. A set of heuristics, based on information appended by observers when they generated observations, should be used to help retain plausible sequences. Example of heuristics include: (a) choose the sequence which integrates the longest series of actions that appear in some collected network evidence as being sent using the same routing path; (b) choose the sequence which integrates the longest series of actions that appear in some collected network evidence as observed with the same included TTL value; (c) Choose the scenarios in which the location of the reported events (when they appear in collected network evidence) remains, at the maximal possible, unchanged in the sequence of actions; and (d) try to insert the new global actions, using the order in which they appear in some network evidence, in the position which contains the notation ε (i.e., the observer was in sleeping mode).

The third step consists in looking for possible attack scenarios using rules 10, 11, 12, 13, and 14. From the obtained possible scenarios, rules 15, 16, and 17 will be used to look for provable actions and scenarios. If no provable attack scenario is found, digital investigators

have to select from the set of potential evidence a plausible attack scenario. We describe hereinafter three additional techniques that can be explored for this purpose. The first technique consists of extracting from every possible attack scenario a set of information (e.g., scanned services and vulnerabilities, type of executed commands) that profiles the attacker.

The plausible attack scenario is the one that is associated to the most malicious profile. The second technique consists of extracting from every possible attack scenario information showing for every system resource which is affected by the attack, the estimated degree of damage. The plausible attack scenario is the one that exhibits the highest degree of system damage.

The third technique consists in labeling every action in the possible attack scenario by a value that estimates the probability of its occurrence from the state in which it is enabled. Obviously, actions, which are extracted from the aggregated network evidence, will get a probability equal to 1. Several techniques [21,22] for estimating the probability of occurrence of the whole attack scenario, starting from the probabilities of elementary actions in the graphs of attacks, would be used. To enhance the accuracy of the obtained values, additional parameters such as frequency of the attack and associated risk could also be used.

As the generation of an attack scenario is supported by a set of network and system evidence whose length is finite, and since by definition an attack scenario is composed of a finite sequence of actions, the generated attack scenario would be finite and the deduction system is expected to terminate. Note that, loops that could appear in the scenarios under construction should be eliminated. Rules for aggregating network evidence are mainly composed of rules for simplification and rules for merging evidence. The convergence of the deduction system would only depend on the completeness of the library of actions.

However, as it could be argued that most of the attack scenarios are reusing the same elementary actions, the library of actions could be supposed to be complete to some extent. The deduction system is consistent since inference rules do not reveal any inconsistency. In fact, the aggregation rules are all used to simplify or merge actions in evidence or concatenate actions and states in the attack scenario under construction.

Case Study

In this section we describe a case study related to the investigation of a buffer overflow attack [23] on a remote service, showing the use of the inference system to generate possible scenarios and provable actions. The described attack was chosen for the following reasons. First, it includes one of the most damaging actions, which is buffer overflow. Second, it generates network and system evidence, showing the benefit of using two types of observers. Third, to process evidence and generate scenarios of attacks, the example will require the use of almost all rules described in the inference system. Fourth, the scenarios to be generated further to the collection of evidence left by the attacker will include provable and possible forms of actions.

The investigated system is modeled using three variables, namely Pr, SrvSt, and SzBuf. The first variable, which is Pr, describes the privilege granted to the remote user. Three possible values could be affected to such a variable: 0, 1, and 2, which stand for, a disconnected user, a user logged in with an unprivileged access, and a user logged in with a privileged access, respectively. Variable SrvSt describes the status of the remote service and takes two possible values 1 or 0 to indicate whether the service is enabled or down, respectively. Variable SzBuf represents the size of the buffer used to read the commands sent by the remote user to use the service. The maximal size of this buffer is equal to one character.

During the occurrence of the attack scenario, both user nodes and observer nodes change their locations and some of them move in and out of the coverage of each other. Some of the observer nodes, for example, go out of the coverage of the user nodes (which participated in routing the attack traffic) during some period of the attack scenario.

The network topology is shown in Figure 1. The small arrows drawn beside user nodes in the graph represent their mobility direction during the attack scenario occurrence. Nodes n_1 and n_5 represent the attacker and victim hosts, respectively, while nodes n_2, n_3, and n_4 are the intermediate nodes used to route the traffic between the attacker and the victim. Three observer nodes, o_1, o_2, and o_3, are considered in this topology. During the occurrence of the attack scenario, these observers change their locations many times. Their trajectory and positions are shown in the same figure.

The attack scenario can be modeled, from the network point of view, as a series of five packets $\pi = p_v, p_w, p_x, p_y, p_z$). Note that the identity of the attacker remained unchanged during the occurrence of the attack. Precisely, we have $\pi = \langle(ip_s, ip_d, [n_1n_2n_3n_4n_5], 64, \mathrm{loc}_1, (a_1, d_1)), (ip_s, ip_d, [n_1n_2n_3n_4n_5], 64, \mathrm{loc}_1, (a_2, d_2)), (ip_s, ip_d, [n_1n_2n_3n_4n_5], 64, \mathrm{loc}_1, (a_3, d_3)), (ip_s, ip_d, [n_1n_2n_4n_5], 64, \mathrm{loc}_1, (a_4, d_4)), (ip_s, ip_d, [n_1n_2n_4n_5], 64, \mathrm{loc}_1, (a_5, d_5))]\rangle$. Note that to cope with attacks in which the attacker changes its identity, new rules should be appended to the inference system to analyze evidence and detect identities related to the same nodes, by correlating actions in the set of evidence and detect causalities between them. For example, two node identities can be

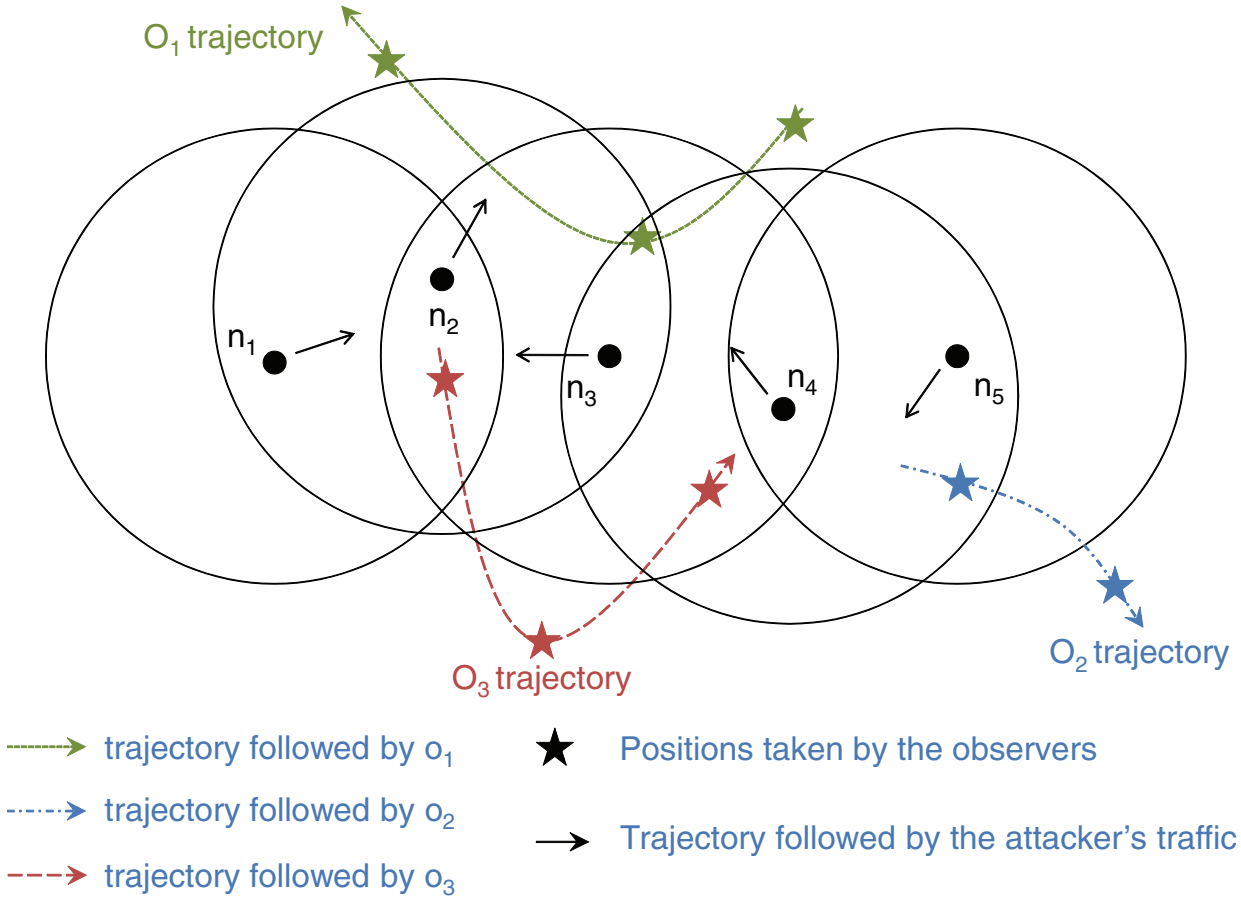

Figure 1 Network topology variation.

considered as related to the same node, if the last action executed by the first enables the first action executed by the second.

During the occurrence of the attack scenarios nodes change their positions. In particular, node $_3$ goes out of the coverage of nodes n_2 and n_4 after sending the second packet. Two routes were successively used to forward traffic from node n_1 to node n_5, which are $n_1n_2n_3n_4n_5$ and $n_1n_2n_4n_5$. The observer node o_1 took three positions. In the first and third positions (denoted by $1oc_1^{o_1}$ and $1oc_3^{o_1}$, respectively) none of the nodes n_1, n_2, n_3, n_4, and n_5 were in its coverage. In the second position (denoted by $1oc_2^{o_1}$) that it takes, it was able to listen to datagrams exchanged between n_2 and n_4 though n_3. The second observer node o_2 took two positions. In the first position, say $1oc_1^{o_2}$, it was able to listen to datagrams sent between nodes n_4 and n_5. When it moved to the second position, say $1oc_2^{o_2}$, it became unable to listen to any datagram sent between nodes n_1 to n_5. The observer node o_3 took three positions. In the first position, say $1oc_1^{o_3}$, it was able to listen to datagrams exchanged between nodes n_1, n_2, and n_3. The second position, say $1oc_2^{o_3}$, that it took does not allow it to listen to any datagrams sent from n_1 to n_5. In the third position, say $1oc_3^{o_3}$, it was able to listen to datagrams exchanged between n_2 and n_4 through n_3 (when node n_3 was in the coverage of node n_2) and also directly between n_2 and n_4 (when node n_3 moved out of the coverage of node n_2).

Due to the mobility of observers, from π, the observer nodes o_1 was able to capture the second and third packets which were sent to remotely execute the actions a_2 and a_3. The observer node o_2 captured the first, second, and third packets containing actions a_1, a_2, and a_3. The third observer, say o_3, captured packets containing actions a_1, a_3, and a_5. Immediately after the reception of the datagram used to execute action a_3, the observer node o_3 changed its internal state to sleeping. Later,

before capturing the datagram containing action a_5, it woke up. Therefore, we have:

$$\begin{aligned}\text{obs}_{o_1}(\pi) = \langle &(ip_s,\ ip_d,\ [n_2n_3n_4],\ 63, 1\text{oc}_2^{o_1},\ (a_2,\ d_2)),\\ &(ip_s,\ ip_d,\ [n_2n_3n_4],\ 63, 1\text{oc}_2^{o_1},\ (a_3,\ d_3))\rangle\end{aligned} \quad (18)$$

$$\begin{aligned}\text{obs}_{o_2}(\pi) = \langle &(ip_s,\ ip_d,\ [n_4n_5],\ 63, 1\text{oc}_1^{o_2},\ (a_1,\ d_1)),\\ &(ip_s,\ ip_d,\ [n_4n_5],\ 63, 1\text{oc}_1^{o_2},\ (a_2,\ d_2)),\\ &(ip_s,\ ip_d,\ [n_4n_5],\ 63, 1\text{oc}_1^{o_2},\ (a_3,\ d_3))\rangle\end{aligned} \quad (19)$$

$$\begin{aligned}\text{obs}_{o_3}(\pi) = \langle &(ip_s,\ ip_d,\ [n_1n_2n_3],\ 63, 1\text{oc}_1^{o_3},\ (a_1,\ d_1)),\\ &(ip_s,\ ip_d,\ [n_3n_4],\ 62, 1\text{oc}_2^{o_3},\ (a_3,\ d_3))\rangle, \varepsilon,\\ &(ip_s,\ ip_d,\ [n_2n_4],\ 63, 1\text{oc}_2^{o_3},\ (a_5,\ d_5))\rangle\end{aligned} \quad (20)$$

When actions and a_2 and a_3 were executed, the observer node o_1 was in the second position and had in its coverage node n_3. When node n_2 forwarded the packet (used to remotely execute action a_2 or a_3) to node n_3, the observer node was able to detect that datagram when the related signal propagated to the area defined by its reception coverage. It also determined that n_3 was the next hop and kept a copy of that datagram in its buffer. The same explanation can be used to demonstrate the observation of the second packet in π, which indicates the remote execution of action a_4.

After collecting observations from the observer nodes o_1, o_2, and o_3, the node in charge of investigation generates a set Π equal to $\{\langle A_2, A_3\rangle, \langle A_1, A_2, A_3\rangle, \langle A_1, A_3, \varepsilon, A_5\rangle\}$ where $A_i = (a_i, d_i)$. Rule 5 is used to set the aggregated network evidence $\overline{\Pi}$ be equal to $\langle A_1, A_3, \varepsilon, A_5\rangle$. Rule 6 uses the sequence of global actions $\langle A_1, A_2, A_3\rangle$ to set $\overline{\Pi}$ be equal to $\langle A_1, A_2, A_3, \varepsilon, A_5\rangle$.

Note that, if action A_4 had been captured by the first observer, and the following observation had been provided

$$\begin{aligned}\text{obs}_{o_1}(\pi) = \langle &(ip_s,\ ip_d,\ [n_2n_3n_4],\ 63, 1\text{oc}_2^{o_1},\ (a_2,\ d_2)),\\ &(ip_s,\ ip_d,\ [n_2n_3n_4],\ 63, 1\text{oc}_2^{o_1},\ (a_3,\ d_3))\\ &(ip_s,\ ip_d,\ [n_2n_3n_4], 63, 1\text{oc}_2^{o_1},\ (a_4,\ d_4)\rangle\end{aligned} \quad (21)$$

then using rule 7, and the sequence of global action $\langle A_3, A_4\rangle$, two sequences, namely $\langle A_1, A_2, A_3, A_4, A_5\rangle$, and $\langle A_1, A_2, A_3, A_5, A_4\rangle$ would have been possible to obtained. By applying the second heuristic (see Sect. VI), the favorite sequence would have been $\langle A_1, A_2, A_3, A_4, A_5\rangle$ since the third observer was sleeping between the observation of the two global actions A_3 and A_5, and the TTL value of the observed packets, which would have been provided by $\text{obs}_{o_1}(\pi)$ and contained actions A_3 and A_4, had been equal.

At the system layer, two evidence were retrieved. The first is provided by a service monitoring application which is only able to monitor and interpret variable SrvSt (i.e., Variable SrvSt is visible by the related observer while variables Pr and SzBuf are invisible). Its content is described as $O_1 = \langle\varnothing$ "on" $\varnothing, \varnothing$ "off '' $\varnothing\rangle$. The second evidence is provided by the system log daemon which only allows monitoring of the user privilege on the system. The related observation function is able to only observe variable Pr (i.e., variable Pr is visible while variables SrvSt and SzBuf are invisible). The recovered evidence is equal to $O_2 = \langle 0\varnothing\varnothing, 1\varnothing\varnothing, 2\varnothing\varnothing, 1\varnothing\varnothing, 0\varnothing\varnothing\rangle$.

The initial system state, described as follows, states that no privilege is provided to the users (i.e., Pr = 0), the service is running (i.e., SrvSt = "on"), and the buffer is empty (i.e., SzBuf = 0). Using the inference system and the above described network and system observations (used as axioms in the deduction system), two possible attack scenarios are generated. They are described by Figure 2 and can be summarized as follows. In the first scenario, a user connects to the system with a simple user privilege and execute some command, whose size is equal to one character, in order to use that service. After that, it exploits a buffer overflow vulnerability related to that version by executing a command which overflows the buffer attached to that service. It induces the service to read a number of characters higher than one (the maximal size of the buffer). Consequently only one character will be stored in the service buffer, while the remaining character will overwrite adjacent buffers and result in erratic program behavior. This induces the service to a denial of service state, and raises the user privilege (the value of variable Pr becomes equal to 2). Later the attacker exits the root's session, then, it disconnects from the system. The second scenario is similar to the first, except that the attacker restarts the service (it used the root privilege that it got immediately after executing the buffer overflow attack) before it exits the root's session, and then logs out.

The generated attack scenario, and the description of states obtained further to the execution of actions, is shown by Figure 2. For the sake of simplicity, we do not describe the content of the library of attacks, nor do we give the formal description of actions part of the attack scenarios. To understand how the inference system works, the description of states obtained further to the application of these actions (see Figure 2) suffices.

Considering the availability the aggregated network evidence $\overline{\Pi}$ and the set $\mathcal{O}$ describing the two collected system evidence and the observation functions of the security solutions which generate them, rule 10 is used to make $\langle s_0\rangle$ coherent with $\mathcal{O}$. After that, rule 12 makes $\langle s_0\rangle$ a possible scenario. Later tuples of actions and states $\langle a_1, s_1\rangle$, $\langle a_2, s_2\rangle$, $\langle a_3, s_3\rangle$, $\langle a_4, s_4\rangle$, and $\langle a_5, s_5\rangle$ are appended one by one to the scenario under construction based on the use of rule 11. This makes the attack

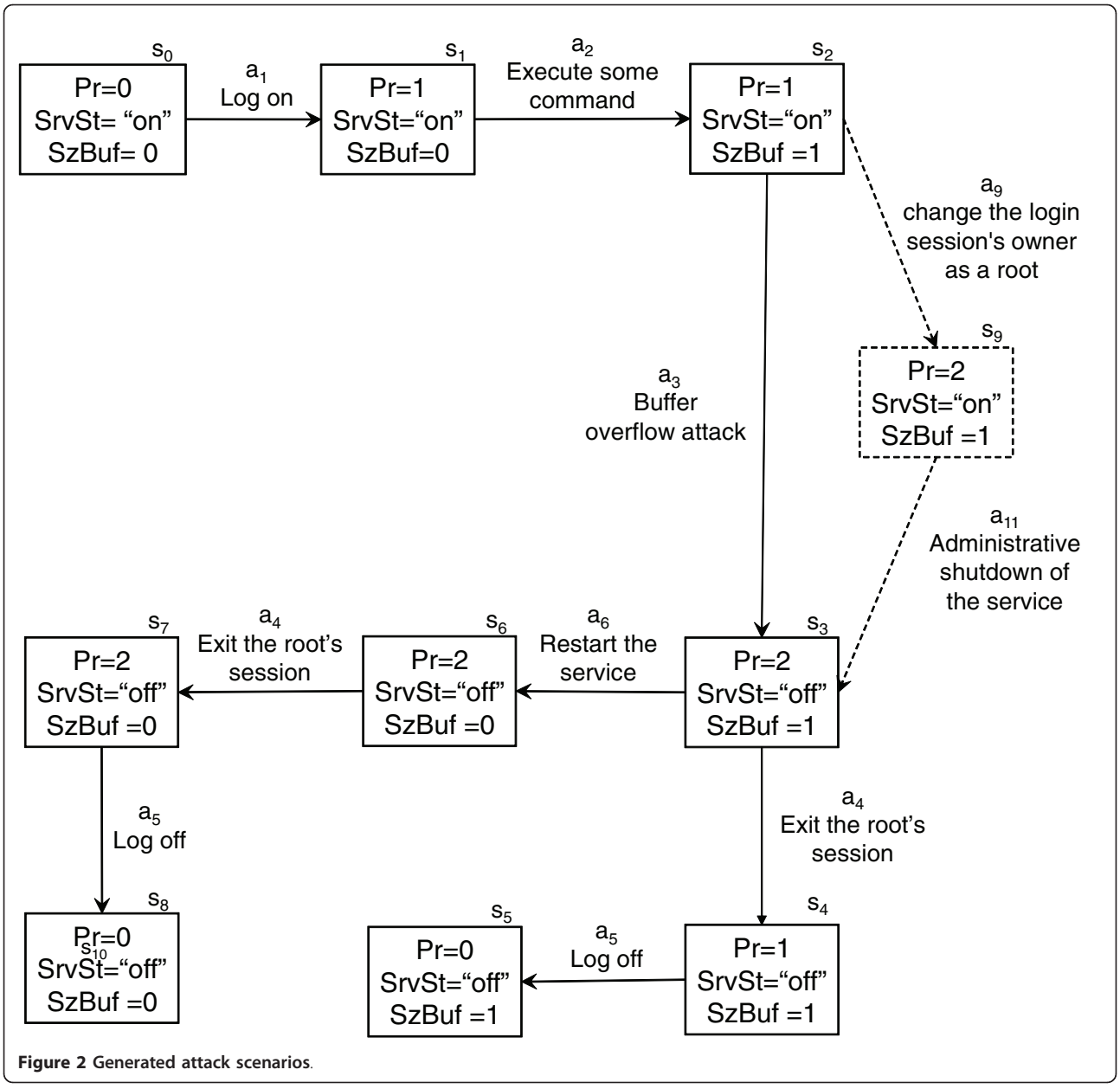

Figure 2 Generated attack scenarios.

scenario $\langle s_0, a_1, s_1, a_2, s_2, a_3, s_3, a_4, s_4, a_5, s_5\rangle$ be coherent with $\mathcal{O}$. Using rule 13, and starting from the scenario $\langle s_0\rangle$ the same tuples of actions and states can be appended one after the other to prove that the scenario $\langle s_0, a_1, s_1, a_2, s_2, a_3, s_3, a_4, s_4, a_5, s_5\rangle$ is a possible one.

Starting from state s_3, another alternative is possible. In fact action a_6, which consists in cleaning up the service buffer, is executed to generate $\langle a_6, s_6\rangle$. Starting from that state, actions a_4 and a_5 are executed similarly to the previous scenario to generate states s_7 and s_8, respectively. Using rule 14, the scenario $\langle s_0, a_1, s_1, a_2, s_2, a_3, s_3, a_6, s_6\rangle$ is generated as a possible scenario starting from $\langle s_0, a_1, s_1, a_2, s_2, a_3, s_3\rangle$ where action a_6 is retrieved from the library of actions. Later, when applied twice, rule 13 can be used to generate the possible scenario $\langle s_0, a_1, s_1, a_2, s_2, a_3, s_3, a_6, s_6, a_4, s_7, a_5, s_8\rangle$.

From the generated possible attack scenarios, actions a_1, a_2, a_3, and a_5 are provable using rule 15. However, action a_5 which, does not belong to $\overline{\Pi}$, cannot be proved using rule 16 since when it is replaced by a_6 (starting from state s_3), it leads to the creation of a coherent attack scenario. From the two generated possible attack scenarios, the first scenario $\langle s_0, a_1, s_1, a_2, s_2, a_3, s_3, a_4, s_4, a_5, s_5\rangle$ is provable. From the second scenario, the two fragments $\langle s_0, a_1, s_1, a_2, s_2, a_3, s_3\rangle$ and a_4, s_7, a_5, $s_8\rangle$ are provable.

Note that, if a_1 and a_2 were executed by the administrator, the latter would have executed action a_9 to change the login session's owner as a root. After that, it shuts down the service (action a_{11}) and executes actions a_4 and a_5. While the two scenarios $\langle s_0, a_1, s_1, a_2, s_2, a_9, s_9, a_{11}, s_3, a_4, s_4, a_5, s_5\rangle$ and $\langle s_0, a_1, s_1, a_2, s_2, a_9, s_9, a_{11}, s_3, a_6, s_6, a_4, s_7, a_5, s_8\rangle$ could be generated using rules 10 and 11 as two coherent scenarios, they could not be shown to be possible or provable. In fact, rule 13, which is used to generate a possible scenario containing an action captured by network observers, could not be executed, as it requires that action a_3 is integrated to the scenario under construction before action a_4 could be appended.

Conclusion

A formal technique and methodology for digital investigation in wireless *ad hoc* and sensor networks was provided in this work. We considered an *ad hoc* network, which is composed of two types of nodes, namely mobile nodes and observer nodes. We provided an inference system to aggregate network evidence collected from the deployed observers, and generate possible and provable attack scenarios using network and system evidence. To exemplify the proposal, we proposed a case study dealing with the investigation of a remote buffer overflow attack which induced a denial of service. Future work will address techniques for cooperative analysis and reconstruction of the attack scenarios, the support of investigation of cooperative attacks, and the study of problems associated to scalability of the proposed approach.

Abbreviations

FSM: Finite State Machine; MASNets: Mobile *Ad hoc* and Sensor Networks.

Competing interests

The authors declare that they have no competing interests.

Received: 5 November 2010 Accepted: 13 July 2011
Published: 13 July 2011

References

1. Nicole Lang Beebe , Jan Guynes Clark , A hierarchical, objectives-based framework for the digital investigations process. Digital Investigation. **2**, 147–165 (2007)
2. Peter Stephenson, Modeling of post-incident root cause analysis. International Journal of Digital Evidence. **2**(2), 1–16 (2003)
3. Tye Stallard, Karl Levitt, Automated analysis for digital forensic science: Semantic integrity checking, in *Proceedings of the 19th Annual Computer Security Applications Conference*, (Las Vegas, Nevada, USA, December 2003)
4. Pavel Gladyshev, Finite State Machine Analysis of a Blackmail Investigation. International Journal of Digital Evidence. **4**(1), 130–149 (2005)
5. D Carrier Brian, H Spafford Eugene, Categories of digital investigation analysis techniques based on the computer history model. Digital Investigation Journal. **3**(S), 121–130 (August 2006)
6. Svein Yngvar Willassen, Timestamp evidence correlation by model based clock hypothesis testing, in *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia* (2008)
7. Ali Reza Arasteha, Mourad Debbabi, Assaad Sakhaa, Mohamed Saleh, Analyzing multiple logs for forensic evidence. Digital Investigation. **4**(1), 82–91 (September 2007)
8. Slim Rekhis, Noureddine Boudriga, Logic-based approach for digital forensic investigation in communication networks. Computers & Security (2011, in press)
9. Shuo Ding, A survey on integrating manets with the internet: Challenges and designs. Computer Communications, **31**(14), 3537–3551 (September 2008). doi:10.1016/j.comcom.2008.04.014
10. Slim Rekhis, Noureddine Boudriga, A formal rule-based scheme for digital investigation in wireless ad-hoc networks, in *Proceedings of Fourth International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering*, (Oakland, California, USA, 21 May 2009), pp. 62–72
11. Pavel Gladyshev, Ahmed Patel, Formalising event time bounding in digital investigations. International Journal of Digital Evidence, **4**(2) (2005)
12. Svein Willassen, Hypothesis-based investigation of digital timestamps, in *Proceedings of Fourth Annual IFIP WG 11.9 International Conference on Digital Forensics*, (Kyoto, Japan, 27–30 January 2008)
13. Slim Rekhis, Noureddine Boudriga, Visibility: a novel concept for characterising provable network digital evidences. International journal of security and networks, **4**(4), 234–245 (September 2009). doi:10.1504/IJSN.2009.028670
14. Jill Slay, Benjamin Turnbull, The need for a technical approach to digital forensic evidence collection for wireless technologies, in *Proceedings of the 2006 IEEE Workshop on Information Assurance*, (West Point, NY, June 21-23 2006), pp. 124–132
15. Yi Huangan , Wenke Lee, Hotspotbased traceback for mobile ad hoc networks, in *Proceedings of The ACM Workshop on Wireless Security (WiSe 2005)*, (Cologne, Germany, September 2005)
16. Yongjin Kim, Ahmed Helmy, SWAT: Small world-based attacker traceback in ad-hoc networks, in *The Second Annual International Conference on Mobile and Ubiquitous Systems*, (San Diego, USA, July 2005)
17. Slim Rekhis, Noureddine Boudriga, Investigating attack scenarios in multihop wireless systems. Journal of networks, **4**(7), 539–551 (September 2009)
18. Somnath Ghosh, Prakash Veeraraghavan, Samar Singh, Lei Zhang, Performance of a wireless sensor network mac protocol with a global sleep schedule. International Journal of Multimedia and Ubiquitous Engineering, **4**(2), 99–114 (April 2009)
19. Lei Huang, Lixiang Liu, Extended watchdog mechanism for wireless sensor networks. Journal of Information and Computing Science, **3**(1), 39–48 (2008)
20. C Snoeren Alex, Craig Partridge, Luis A Sanchez, E Jones Christine, Fabrice Tchakountio, T Kent Stephen, W Timothy Strayer, Hash-based IP traceback. ACM SIGCOMM Computer Communication Review, **31**(4), 3–14 (2001). doi:10.1145/964723.383060
21. Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund M Clarke, Jeannette M Wing, Ranking attack graphs, in *Proceedings of 9th International Symposium On Recent Advances In Intrusion Detection*, (Hamburg, Germany, 20–22 September 2006), pp. 127–144
22. Jide B Odubiyi, Casey W O'Brien, Information security attack tree modeling, in *proceedings of Seventh Workshop on Education in Computer Security (WECS7)*, (Monterey, California, 04–06 January 2006), pp. 29–37
23. Eugen Leontie, Gedare Bloom, Olga Gelbart, Bhagirath Narahari, Rahul Simha, A compiler-hardware technique for protecting against buffer overflow attacks. Journal of Information Assurance and Security, **5**, 1–8 (2010)

doi:10.1186/1687-1499-2011-39
Cite this article as: Rekhis and Boudriga: **Formal reconstruction of attack scenarios in mobile *ad hoc* and sensor networks.** *EURASIP Journal on Wireless Communications and Networking* 2011 **2011**:39.