



## CNAS REPORT

Printed January 2008

CNAS-2008-003  
Public

Supersedes CNAS-2007-003  
Dated Dec. 2007

---

# Visibility: A Novel Concept for Characterizing Provable Network Digital Evidences

Slim Rekhis  
Noureddine Boudriga

Prepared by  
CN&S Research Lab.

The Communication Network and Security (CN&S) research Laboratory,  
(Created in 1999, 02/UR/11-08) is located at the Communication School of Engineering  
(University of 7th of November at Carthage, Tunisia).

Approved for public release.

Copyright © 2007 by the Communication Networks and Security Research Lab. All rights reserved.

**NOTICE:** No part of this publication may be reproduced, stored in a retrieval system, or transmitted without written authorization from the CN&S research lab.

Available from

CN&S research lab.  
Engineering school of communications.  
Techno-parc El Ghazala, Route de Raoued.  
Ariana, 2083, Tunisia.

Telephone: (+216) 71857000 (ext. 2104)  
Facsimile: (+216) 71856829  
E-Mail: cnas@laposte.net

*Approved for public release*

Professor Nouredine Boudriga  
Head of CN&S research lab.

---

# Visibility: A Novel Concept for Characterizing Provable Network Digital Evidences

Slim Rekhis and Noureddine Boudriga

CN&S Research Lab., University of the 7th of November at Carthage, Tunisia.

<http://www.cnas.org.tn/home.htm>

[slim.rekhis@gmail.com](mailto:slim.rekhis@gmail.com), [nab@supcom.rnu.tn](mailto:nab@supcom.rnu.tn)

**Abstract**—Spoofing attacks represent one of the most remarkable attacks that challenged network investigators and can be associated to harmful attacks such as Distributed Denial of Service. Providing a formal method of investigation to be used with traceback techniques happened to be of utmost importance, allowing to prove absence of design weakness in the method, proving any result that an investigator would otherwise have to compute himself (e.g., attack occurrence, intruder source, path followed by the attack), and ease the admissibility of evidences. We provide in this work a new formal concept, entitled *Visibility*, and we develop its relation with network digital forensic investigation, particularly the investigation of source address spoofing attacks. To demonstrate the effectiveness of our visibility-based theory, we use it in conjunction with an efficient traceback technique to prove IP spoofing attack occurrences and identify their source.

**Keywords:** *visibility, formal proof, digital forensic investigation, source address spoofing, packet tracing.*

## I. INTRODUCTION

Due to the pervasiveness of information technology, systems and networks are becoming more and more available. This increases their exposure to vulnerability and attacks. Focusing merely on deploying defensive mechanisms is insufficient, as intruders are still gaining the upper hand. To counter this, recently research in security has attached an attention to the digital forensic investigation. Defined as “preservation, identification, extraction, documentation and interpretation of computer data” [1], the digital forensic investigation takes an interest to the analysis of data in networks, systems, disks, and memories for clues and evidences. It aims to reconstruct the security incident and give answers to what, who, when, where, how, and why incident-related questions.

One of the most remarkable attacks that threatened networks and can be associated to a large set of harmful attacks such as Distributed Denial of Service (DDoS), we can find the spoofing attacks. A spoofing attack is the process by which an entity (e.g., a program) forges data to impersonate another and gain an illegitimate access or privilege. The scope of such attack is very large to the extent that it affected a lot of protocols among them we distinguish the Internet Protocol, in which the attack takes as a name: IP spoofing attack [2].

In such context, several traceback approaches were proposed to identify the route of incoming traffic and trace intruders to their source. These techniques can be classified into link testing, deterministic, probabilistic, or selective packet marking, logging of packets information or packets digests, and ICMP

messaging [3], [4]. To the best of our knowledge, despite the work provided in [5], none contribution in this area has integrated the use of formal methods of specification and verification. In fact, using formal verification approaches during the design of a traceback technique happened to be of utmost importance, allowing to a) demonstrate absence of design weakness in the technique; b) formally prove any result that a security administrator would otherwise have to compute himself (e.g., attack occurrence, intruder source, path followed by the attack); and c) enhance the accuracy of digital attack analysis. Moreover, the concept of network digital investigation was practically absent when traceback methods were conceived. Thus, it turns out to be worthy to provide a formal technique of investigation that copes with the output of traceback techniques to provide accurate proof of digital evidences.

We take interest in this work into formalizing the proof in the context of digital network forensic investigation, and characterizing what is provable. We provide a new concept, entitled *Visibility*, whose essential idea is to prove a given property solely based on a partial observation of a system execution. We develop its relation with *Opacity* that was recently provided as a promising concept for the verification of security properties [6] and the digital forensic investigation [7]. After developing the visibility theory, we set up a relation between the concept of visibility and network digital forensic investigation, particularly the investigation of source address spoofing attacks. To demonstrate the effectiveness of our visibility-based theory, we use it in conjunction with an efficient traceback technique to prove IP spoofing attack occurrences and identify their source.

Our contribution is 3-fold. First, we complement the few existing contributions in formal digital investigation by a novel theory that is generic and promises to show its effectiveness in other fields of computer security. Second, we bring a set of visibility properties that allows an investigator to accurately define the scope of its proof and the details he wants to prove. Third, we develop an efficient method that is able to identify and trace spoofing attacks in packet switching protocols.

The remaining of this paper is organized as follows. Section II reviews the work on opacity. Section III introduces the concept of visibility, provides a set of relative properties, and states a set of results in terms of propositions. In section IV, a generic model of packet switching protocols is defined. Section V sets up a relation between the concept of visibility and digital network investigation, formalizes source address spoof-

ing attacks, considers the IP protocol as example, and provides a set of formal results in terms of visibility. To illustrate the proposal, a case study integrating a traceback technique, is provided in Section VI. Finally, Section VII concludes the work.

## II. OPACITY PROPERTY

We provide in this section the opacity background. We start giving a brief idea on the different recent contributions in this research area, then we describe the concept of opacity couched in a state-based logic, particularly the Temporal Logic of Security Actions [8].

### A. Related works

A few works were conducted along these years to properly define opacity, verify its features, and establish some decidability results. [9] defines opacity within the CSP notation in a way distinctive for the expression of anonymity. It decides such property in finite models using the Model Checking technique. [10] proposed a more generic framework for the specification of opacity properties without approaching the decidability issues. The framework was used in [11], where opacity properties were verified using an abstraction technique. Recently, the opacity concept was extended to systems in general rather than focusing on cryptographic protocols [12]. The work was discussed in environments such as Petri nets [13] and Labeled Transition Systems [6], where decidability of opacity verification is addressed. Later, the author of this paper provided, in [7], a novel approach that integrates opacity theory to digital forensic investigation. For this, he extended the opacity concept by adding multi-observability and new classes of properties to opacity. Then, he showed the appropriateness of using opacity in proving scenarios and evidences related to hacking activities.

### B. Observations over states and executions

Generally speaking, opacity of a given property means that a third party which only has access to some part (called the observable part) of the system executions, cannot deduce the truth of that property. To formally describe opacity, we consider the Temporal Logic of Security Action, which was introduced in [8] as an extension to TLA [14] to provide reasoning on systems with uncertainty by adding forward hypotheses to fulfill potential lack of details. S-TLA is a state-based logic that allows the description of states and state transitions. Every state is a valuation of all system variables and a state transition is described by an action that is a relation between an old state, say  $s$ , and a new state, say  $t$ . Written as a conjunct of a hypothesis and an event, an action is true or false for a pair of states  $\langle s, t \rangle$ .

Given a specification  $Spec$  that may generate a set  $S$  of reachable states, where every state represents a valuation of all system variables. We denote by  $\Omega = \cup\{\omega_i\}$  the set of all acceptable executions with respect to specification  $Spec$ . Every execution  $\omega$  represents a series of system states in the form of:  $\omega_i = \langle s_{i0}, s_{i1}, \dots, s_{in} \rangle$ , where  $s_{i(j+1)}$  is derived from  $s_{ij}$  after an action is executed. In the remaining part of this paper, we denote by  $\hat{\omega}$ , the execution obtained from  $\omega$  after collapsing the set of subsequent states that are equal (that may appear due to

stuttering [14]). Moreover, we denote by  $\hat{\omega}^{init}$ ,  $\hat{\omega}^{fin}$ ,  $\hat{\omega}^i$  the initial, final, and  $i^{th}$  state in execution  $\hat{\omega}$ , respectively.

We consider an observation function  $Obs$  that allows an observer, who can have a complete knowledge of the system specified by  $Spec$ , to see limited information of the system states and executions. Given a system modeled by a set of variables  $Var = \{v_1, \dots, v_n\}$ , where every state represents a valuation of all variables  $v \in Var$ . We define the observable part of a state  $s$ , in either a static or dynamic manner. In the static manner, observation is in the form of  $Obs(s)$  where  $Obs(s) = [l^s(v_1)l^s(v_2)\dots l^s(v_n)]$  and where  $l^s(v_i)$  represents the label of variable  $v_i$  in state  $s$ , which can take one of the following three forms depending on  $v_i$ 's visibility:

- *A variable value:* A variable is visible and its value is interpretable by the observer. Its label is equal to the variable value in that state  $s$ . Precisely, we have  $l^s(v) = s(v)$ .
- *A fictive value:* A variable is visible but not interpretable by the observer, meaning that its variation does not bring any supplementary information to an observer. Examples include visualization of encrypted values or compressed data. A static label is then assigned to the variable through all the system behavior:  $\forall s \in S : l^s(v) = x$  such that  $x \in NoVal$  where  $NoVal$  is a set of fictive values.
- *An empty value:* The variable is completely invisible, such that none information regarding its value could be determined. An empty value is then affected to the variable through all the system behavior:  $\forall s \in S : l^s(v) = \emptyset$ .

In the dynamic manner, an observation is conditioned by the value of an additional state predicate. It is in the form of  $Obs(s, \phi)$  denoting how  $s$  is observed when  $\phi$  is true. We define the observation of an execution  $\omega = \langle s_0, \dots, s_n \rangle$  as the sequence ( $m \leq n$ ):  $Obs(\omega) = \langle Obs(s_0), \dots, Obs(s_m) \rangle$  where any maximal sub-sequence  $Obs(s_i), \dots, Obs(s_j)$  such that  $i < j$  and  $Obs(s_i) = \dots = Obs(s_j)$  will be collapsed into a single state observation (say,  $Obs(s_i)$ ), as this represents what will be observable to a third party.

### C. Opacity definition

A predicate-based property  $\psi$ , defined over  $S$ , is said to be *opaque* if whenever it is true at some specific state(s) of an execution  $\omega$ , a third party cannot establish it solely based on its observation  $obs(\omega)$  and the set of deductions that he can make. Four variants of opacity were proposed in [13]: the *Initial-opacity*, *Final-opacity*, *Always-opacity*, and *Total-opacity*. We present them in S-TLA, respectively as follows:

- A property  $\psi$  is *initial-opaque* if, for every execution  $\omega = \langle s_0, s_1, \dots, s_n \rangle$  such that  $\hat{\omega}^{init} \models \psi$ , there exists an execution  $\omega' = \langle s'_0, s'_1, \dots, s'_n \rangle$  such that  $\hat{\omega}'^{init} \not\models \psi$  and  $obs(\omega') = obs(\hat{\omega})$ .
- A property  $\psi$  is *final-opaque* if, for every execution  $\omega = \langle s_0, s_1, \dots, s_n \rangle$  such that:  $\hat{\omega}^{fin} \models \psi$ , there exists an execution  $\omega' = \langle s'_0, s'_1, \dots, s'_n \rangle$  such that:  $\hat{\omega}'^{fin} \not\models \psi$  and  $obs(\omega') = obs(\hat{\omega})$ .
- A property  $\psi$  is *always-opaque* if, for every execution  $\omega = \langle s_0, s_1, \dots, s_n \rangle$  and some  $i$ , such that:  $\hat{\omega}^i \models \psi$ , there exists an execution  $\omega' = \langle s'_0, s'_1, \dots, s'_n \rangle$  such that  $\hat{\omega}'^i \not\models \psi$  and  $obs(\omega') = obs(\hat{\omega})$ .

- A property  $\psi$  is *total-opaque* if, for every execution  $\omega = \langle s_0, s_1, \dots, s_n \rangle$  an some  $i$  such that  $\hat{\omega}^i \not\models \psi$ , there exists an execution  $\omega' = \langle s'_0, s'_1, \dots, s'_n \rangle$  such that  $\forall x \in \hat{\omega}' : x \not\models \psi$  and  $obs(\hat{\omega}') = obs(\hat{\omega})$ .

Typically, the concept of opacity, as it is formally described above, involves one observation function by which an observer can monitor a given execution. We proposed in [7] a new class of opacity entitled *Obs-free opacity* to apply to the context of digital investigation. Such class takes into account availability of multiple observations functions. This allowed us to build a theory able to handle cooperative digital forensic investigation. Formally, a property  $\phi$ , defined over  $S$ , is *Obs-free opaque*, if for every available observation function  $Obs_i$  the following assertion is satisfied: if  $\phi$  is true at some state(s) in an execution  $\omega$ , a third party cannot establish it solely based on observation  $obs_i(\omega)$  and the set of deductions that he can make.

The use of opacity in digital investigation, has pointed out the need to introduce a new opacity property, called *I-obs-free Opacity* [7]. This property allows to capture digital forensic evidences that should never switch to false whenever they are true. Formally, property  $\phi$  is *I-obs-free opaque* if for every available observation function  $Obs_i$  ( $i \in [1..m]$ ) and for every execution  $\omega = \langle s_0, \dots, s_n \rangle$  for which there exists  $j \in [0..n]$  such that  $\forall x \in \{s_0, \dots, s_{j-1}\} : x \not\models \phi$  and  $\forall x \in \{s_j, \dots, s_n\} : x \models \phi$ , then there exists an execution  $\omega' = \langle s'_0, \dots, s'_n \rangle$  such that  $\forall y \in \omega' : y \not\models \phi$  and  $Obs_i(\hat{\omega}') = Obs_i(\hat{\omega}) \forall i \in [1..m]$ .

### III. VISIBILITY PROPERTY

While opacity is an adequate concept for ensuring that a property  $\phi$  is always indistinguishable, its negation cannot be used to state that  $\phi$  is always distinguishable. To notice this, it suffices to analyze the following example relating to a system that may generate four possible executions, denoted by  $\omega_1, \omega_2, \omega'_1$ , and  $\omega'_2$ .  $\omega_1$  and  $\omega_2$  represent executions for which  $\phi$  is true in the initial state while  $\omega'_1$  and  $\omega'_2$  represent executions for which  $\phi$  is false in the initial state. Suppose that  $obs(\omega_1)$  is equal to  $obs(\omega'_1)$  while  $obs(\omega_2)$  is different from  $obs(\omega'_1)$  and  $obs(\omega'_2)$ . Then, it can be easily noticed that  $\phi$  is not opaque with respect to the definition (It can be realized that  $\phi$  is true when observing execution  $\omega_2$ , as there do not exist any other execution which is observed similarly and has  $\phi$  false in the initial state). However, as execution  $\omega_1$  and  $\omega'_1$  have the same observation, a third party cannot distinguish whether  $\phi$  is true or false when observing these two executions. Consequently, the non opacity of  $\phi$  does not guarantee that  $\phi$  is always distinguishable.

We propose in the following a concept that is, somehow, the opposite to *Opacity*. The concept is called *Visibility*. We are here interested in proving that a property  $\phi$  is distinguishable whenever it is true at some specific state(s) of an execution. We divide the visibility properties into three different classes as follows. For all the definitions, we denote by  $\Omega$  the set of possible executions that a given system may follow. Note that likely to opacity, visibility is based on the fact that an observer has a complete knowledge of the system, and therefore of all the possible executions. In the following definitions, we denote by  $S_\omega$  the set of states that are part of execution  $\omega$ .

*Definition 1: (Provable property)* Let an execution  $\omega$ , a predicate-based property  $\phi$  defined over system states  $S$ , and a state  $s$  in  $S_\omega$ .  $\phi$  is called *provable* in a state  $s$  w.r.t observation  $obs$ , if  $\phi(s)$  can be computed based on  $obs(\omega)$ . Moreover  $\omega$  is called *provable* if it can be completely computed based on  $obs(\omega)$ .

*Definition 2: (Simple, Strong, and Absolute Visibility):* Let  $\phi$  be a predicate-based property, defined over  $S$ :

- **Simple Visibility:** property  $\phi$  is said to be simply visible w.r.t observation  $obs$  if, for every execution  $\omega$  such that  $\phi(\hat{\omega}^j) = true$  for some  $j$ , there do not exist an execution  $\omega'$  such that  $\phi(\hat{\omega}'^j) = false$  and  $obs(\hat{\omega}') = obs(\hat{\omega})$ .
- **Strong Visibility:** property  $\phi$  is said to be strongly visible w.r.t observation  $obs$  if, for every execution  $\omega$  such that  $\phi(\hat{\omega}^j) = true$  for some  $j$ , there do not exist an execution  $\omega' \neq \omega$  such that that  $obs(\hat{\omega}') = obs(\hat{\omega})$ .
- **Absolute Visibility:** property  $\phi$  is said to be Absolutely Visible w.r.t observation  $obs$  if both  $\phi$  and  $\bar{\phi}$  are strongly visible.

For everyone of the three classes of opacity described above, three properties can be distinguished: *Initial-Visibility*, *Final-Visibility*, and *Always-Visibility*, depending on the value of variable  $j$  which can take value *init*, *fin*, or whatever, respectively.

*Example 1:* We provide in Figure 1 four different executions. Each execution is a sequence of two states, and every state, is a valuation of three variables  $x, y$ , and  $z$ . Let  $Obs_1, Obs_2$ , and  $Obs_3$  be three different static observations defined as follows:

- $Obs_1(s) \triangleq l^s(x) = s(x) \wedge l^s(y) = \emptyset \wedge l^s(z) = \emptyset$ : only value of variable  $x$  is visible.
- $Obs_2(s) \triangleq l^s(x) = \emptyset \wedge l^s(y) = s(y) \wedge l^s(z) = \emptyset$ : only value of variable  $y$  is visible.
- $Obs_3(s) \triangleq l^s(x) = \emptyset \wedge l^s(y) = \emptyset \wedge l^s(z) = s(z)$ : only value of variable  $z$  is visible.

In this example, we are interested in the *initial-visibility* of a property  $\phi$  that is defined as:  $\phi \triangleq (v_1 + v_2) \leq 10$ .  $\omega_1$  and  $\omega_3$  represent executions, for which property  $\phi$  is true in their initial states, while  $\omega_2$  and  $\omega_4$  represent executions for which property  $\phi$  is false in their initial states. With respect to the above definitions, observations  $obs_1$  makes  $\phi$  simply visible, as both  $obs(\omega_1)$  and  $obs(\omega_3)$  have an observation that is different from the remaining executions in which  $\phi$  is false (particularly  $\omega_1$  and  $\omega_2$ ). However, while a third party could deduce that  $\phi$  is true in the initial state of the execution it is observing, he remains unable to determine which one of executions it is currently observing.

Despite determining whether  $\phi$  is true or false, strong and absolute visibility allows two additional statements. Strong visibility allows (particularly for the case of executions in which  $\phi$  is true) to distinguish the exact execution starting from its observation. On the other side, absolute visibility allows to distinguish any execution in each  $\phi$  is either true or false, starting from its observation. To illustrate this, we focus on observation  $obs_2$  which makes  $\phi$  strongly visible. In fact, observation  $\langle 1, 2 \rangle$ , for instance, allows a third party not only to deduce that  $\phi$  is true, but also to notice that this observation is relating to execution  $\omega_1$ . In the other side,  $obs_3$  makes  $\phi$  absolutely visible. As a consequence, all the observations are different, and

therefore. any execution in which  $\phi$  is either true or false can be determined.

*Proposition 1:* For  $x \in \{initial, final, always\}$ , a predicate-based property  $\phi$  is simply  $x$ -visible implies  $\neg\phi$  is simply-visible.

*Proof:* We rephrase the simple visibility definition as follows:  $\forall\omega' \neq \omega$  such that  $\phi(\hat{\omega}^j) \neq \phi(\hat{\omega}'^j)$  for some  $j$  ( $j$  can take value: *init*, *fin*, or whatever), we have:  $obs(\hat{\omega}) \neq obs(\hat{\omega}')$ .

As  $\phi(x) \neq \phi(x') \iff \neg\phi(x) \neq \neg\phi(x')$ , by replacing  $\phi(\hat{\omega}^j) \neq \phi(\hat{\omega}'^j)$  by  $\neg\phi(\hat{\omega}^j) \neq \neg\phi(\hat{\omega}'^j)$  in the above expression, we can state that  $\neg\phi$  is simply visible. ■

*Proposition 2:* For  $x \in \{initial, final, always\}$ , a predicate-based property  $\phi$ , which is absolutely  $x$ -visible, is strongly  $x$ -visible. A predicate-based property  $\phi$ , which is strongly  $x$ -visible, is simply  $x$ -visible. A predicate-based property  $\phi$ , which is simply always-visible is provable in any reachable state  $s$ .

*Proof:* The first and second part of the proposition follow from the definition. The third part is demonstrated as follows:

Given some execution  $\omega$  and some state  $s \in \omega$ . It follows from Definition 1 that in the case where  $\phi(s) = true$ , and  $\phi$  is simply always visible, that  $\phi$  is provable to be equal true in  $s$  starting from  $obs(\omega)$ . Using proposition 2, which states that  $\neg\phi$  is also simply always visible, it follows that in the case where  $\phi(s) = false$ ,  $\phi$  is provable to be equal false in  $s$  starting from  $obs(\omega)$ . As  $\phi$  is either true or false in any reachable state  $s$ , it is thus provable there. ■

*Proposition 3:* Let  $\omega$  be an execution,  $S$  be the set of reachable states, and  $\phi$  and  $\psi$  be two predicate-based properties defined over  $S$ .

- Property  $\psi$  is provable in any state  $s \in S_\omega$  if  $\exists s \in S_\omega$  such that  $\phi(s) = true$  and  $\phi$  is strongly visible.

- Property  $\psi$  is provable in any state  $s \in S$  if  $\exists s \in S$  such that  $\phi(s) = true$  and  $\phi$  is absolutely visible.

*Proof:* We remind that an observer has a complete knowledge of the system or of the protocol specification. In the case where  $\phi$  is strongly visible and holds in a state  $s$  that belongs to an execution  $\omega$ ,  $s$  can be proved and  $obs(\omega)$  is different from any other observation of  $\omega'$  (such that  $\omega' \neq \omega$ ). In this case,  $\omega$  can be inferred from  $obs(\omega)$  and thus be proved (all its states computed). Consequently, any property  $\psi$  that holds in any state  $s \in S_\omega$  can be proved.

In the case where  $\phi$  is strongly visible, we have  $\forall\omega \neq \omega'$ :  $obs(\omega) \neq obs(\omega')$ . Consequently, every execution  $\omega$  can be inferred, and therefore every property  $\psi$  can be computed in every state  $s \in S$ . ■

*Proposition 4:* The opacity of a predicate-based property  $\phi$  implies non visibility of  $\phi$ .

*Proof:* Follows from the definition, and therefore it is easily deducible that visibility of a predicate property  $\phi$  implies non opacity of  $\phi$ . ■

#### IV. PACKET SWITCHING PROTOCOLS MODEL

Packet switching protocols are intended to provide a global addressing mechanism to deliver datagrams across a packet-switching network. Roughly speaking, one can say that every packet is composed by a header and a payload. The first contains source and destination addresses. The second contains

data from high-level protocols. To communicate with host  $y$ , a host  $x$  needs to send a packet containing the address of host  $y$  as a destination address. In the case, where host  $x$  is located in the same physical sub-network with host  $y$ , the packet is immediately delivered after encapsulating it in a link-level datagram. In the other case, where the two hosts belong to physically different networks, delivery should be performed indirectly using one or more gateways, called routers. In this case, hosts  $x$  sends the packet to the directly connected gateway, which decides whether it can directly deliver the packet or it has to forward it to another directly connected gateway. The process is called routing and uses routing tables to take such decision after reading the destination address from the packet.

We model a network as a graph  $(V, E)$  where  $V$  is a set of vertexes representing communicating nodes, and  $E = \{e_1, \dots, e_l\}$  is a set of edges representing communication links. We partition the set of vertexes  $V$  into two subsets: the set of hosts  $H = \{h_1, \dots, h_m\}$  and the set of routers  $R = \{r_1, \dots, r_n\}$ . Every edge  $e_i$  in  $E$  takes the form of  $(x_i, x_j)$  such that:  $\{x_i, x_j\} \subset V$ . We define a route  $rte \in RTE$  between two hosts  $h_i$  and  $h_j$  as a sequence in the form of:  $\langle r_1, \dots, r_n \rangle$  such that:

- $\forall\{i, j\} \subset [1..n]$  with  $i \neq j$ , we have  $r_i \in R$  and  $r_i \neq r_j$ .
- $\exists\langle e_0, \dots, e_n \rangle$ : 
$$\begin{cases} e_0 = (h_i, r_1) \\ e_n = (r_n, h_j) \\ e_x = (r_x, r_{x+1}) \text{ for } x \in [1, n-1] \end{cases}$$

Figure 2 shows a network with six hosts, five routers and twelve communication links:

$$\begin{aligned} H &= \{h_a, h_b, h_c, h_d, h_e, h_f\} \\ R &= \{r_a, r_b, r_c, r_d, r_e\} \\ E &= \{e_o, e_p, e_q, e_r, e_s, e_t, e_u, e_v, e_w, e_x, e_y, e_z\} \end{aligned}$$

A message  $m (\in M)$ , sent between two hosts, is received at the target as tuple in the form of  $(h_s, h_d, p, rt)$  where  $h_s$  and  $h_d$  stand for source and destination addresses, respectively;  $p \in P$  represents the datagram payload (from some  $p \in P$ ); and  $rt$  stands for the route from  $h_s$  to  $h_d$  which is used to forward traffic. We introduce functions *Src*, *Dst*, *Rte*, and *Head* to extract from message  $m$  the source address, destination address, route followed by the traffic, and the head of the route, respectively. Their definition is given as follows:

- *Src* :  $M \rightarrow H$ , defined by  $Src(h_s, h_d, p, \langle r_1, \dots, r_n \rangle) = h_s$
- *Dst* :  $M \rightarrow H$ , defined by  $Dst(h_s, h_d, p, \langle r_1, \dots, r_n \rangle) = h_d$
- *Rte* :  $M \rightarrow RTE$ , defined by  $Rte(h_s, h_d, p, \langle r_1, \dots, r_n \rangle) = \langle r_1, \dots, r_n \rangle$
- *Head* :  $RTE \rightarrow R$ , defined by  $Head(\langle r_1, \dots, r_n \rangle) = r_1$

#### V. VISIBILITY-BASED INVESTIGATION

The visibility framework provided as the first part of our contribution represents a generic formal verification technique that is independent of any system or resource under investigation. While the technique is intended to investigate a great set of security attacks including DDoS and anti-forensic attacks, we concentrate hereinafter on the case of source address spoofing attacks.

	$\omega_1$	$\omega_2$	$\omega_3$	$\omega_4$
	$\phi(s_0) = true$	$\phi(s_0) = false$	$\phi(s_0) = true$	$\phi(s_0) = false$
	(2, 8, 6)	(5, 9, 5)	(2, 7, 4)	(5, 9, 3)
	(1, 1, 2)	(2, 3, 3)	(1, 5, 4)	(2, 3, 5)
$Obs_{s_1}$	(2)	(5)	(2)	(5)
	(1)	(2)	(1)	(2)
$Obs_{s_2}$	(8)	(9)	(7)	(9)
	(1)	(3)	(5)	(3)
$Obs_{s_3}$	(6)	(5)	(4)	(3)
	(2)	(3)	(4)	(5)

Fig. 1. Difference between visibility classes

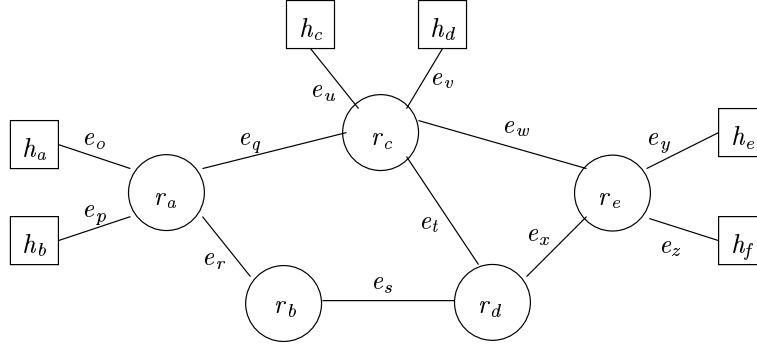


Fig. 2. Network graph

### A. Source address spoofing attacks

Source address spoofing refers to the process of forging the source address of a datagram to make appear as it comes from a higher security entity (e.g., host, network). Even if these attacks can be defeated using two way authentication protocol, some network nodes do not support such protocol as they are unable to use cryptography. We concentrate our attention in this section on formally proving external source address spoofing attacks, meaning that we suppose that an intruder tries to impersonate another machine that does not belong the sub-network under which it is connected.

Given a host  $h$ , we introduce function  $Gtw$ , defined by  $Gtw : H \rightarrow R$  to return the network gateway (router), say  $r$ , that is used by  $h$  to communicate with external hosts. This means that there exists an edge  $e_x \in E$  in the graph which is equal to  $(h, r)$ . In Figure 2, for instance,  $Gtw(h_a) = r_a$  and  $Gtw(h_d) = r_c$ . Moreover, given a message  $m = (h_s, h_d, p, rt)$  such that:  $rt = \langle r_1, \dots, r_n \rangle$ ,  $\{h_s, h_d\} \subset H$ , and  $p \in P$ , we introduce predicate  $Spoof(m)$  defined as follows:

$$Spoof(m) \triangleq Head(Rte(m)) \neq Gtw(Src(m))$$

To capture source address spoofing attacks, we determine the value of such predicate, which is equal to true if  $r_1$  rep-

resents the gateway that is used by  $h_s$  to communicate with host  $h_d$ . We say that a message  $m$  is well-formed if and only if:  $Spoof(m) = False$ . Conversely, a messages  $m$  is under source address spoofing attack if  $Spoof(m) = True$ . Formally, such an attack consists of modifying a well-formed message  $m = (h_x, h_y, p, rte)$  by a message  $m' = (h'_x, h_y, p, rte)$  such that  $h'_x \neq h_x$  and  $Head(rte) \neq Gtw(h'_x)$ . Obviously, the forged message is expected to follow the same physical route as the original message, that is why  $Rte(m)$  is equal to  $Rte(m')$ .

### B. Characterizing provable evidences

Since intruders can hide their identity using source address spoofing, conducting a network investigation analysis in order to trace the anonymous attack flow or reveal the identity of the intruder is amongst the challenging tasks. In this context, several solutions have been developed as an attempt to locate hosts in a specific network independently of its source address (included the datagram that it sends). They are called tracing systems. However, providing a proof of the correctness of these solutions has still to be developed. Using the visibility concept, we build a formal theory for providing proofs in network digital investigation of source address spoofing attacks.

We denote by session  $s$  a sequence of messages in the form of:  $\langle m_1, \dots, m_n \rangle$  representing a conversation between two

hosts, defined by their addresses  $Src(m_i)$  and  $Dst(m_i)$  for some  $i \in [1..m]$ , respectively. A distinctive feature of a session  $s$  is highlighted by the following point: for two consecutive messages  $(m_i, m_{i+1}) \subset s$ , we have:  $src(m_i) = dst(m_{i+1})$  and  $dst(m_{i+1}) = dst(m_i)$ .

In addition, for every message  $m_{i+2} \in S$  ( $m_{i+2}$  represent messages sent by the same source), the head  $Head(Rte(m_{i+2}))$  of the route is the same. However, the route taken by different messages can be different.

Typically, a network forensic investigator performs data collection and analysis at the network level. The digital evidence that he is able to access would represent the exchanged traffic between victim and intruder. Formally, the evidences have the form of a session  $s = \langle m_1, \dots, m_n \rangle$  where each message  $m_i$  is a valuation of variables  $(h_s, h_d, p, rt)$ . Unfortunately, from such evidences, the investigator has only access to  $obs(s)$ . The definition of  $obs$  function, which characterizes what is observable to the investigator, will vary from a protocol to another.

Starting from the set of previously defined propositions and the above described model, we establish that: Given a packet switching protocol  $P$ , four situations can occur when performing digital  $P$ -investigation, depending on the visibility of predicate  $Spoof$ :

- Occurrence of source address spoofing is not provable if Predicate  $Spoof$  is not *simply initial-visible*.
- Occurrence of source address spoofing is provable if Predicate  $Spoof$  is *simply always-visible*.
- Both occurrence of source address spoofing and real intruder source address are provable for every attack if predicate  $Spoof$  is *strongly always-visible*.
- Both occurrence of source address spoofing and real source address are provable for both legitimate and malicious traffic if predicate  $Spoof$  is *absolutely always-visible*.

### C. Example: IP Spoofing attacks

We consider in this sub-section the particular case of IP protocol showing how IP spoofing attacks can mislead investigation. Such protocol fits acutely with the packet switching protocol described in section IV. A message  $m$  represents now an IP packet  $pck = (ip_s, ip_d, p, rt)$  where  $ip_s$  and  $ip_d$  stand for IP source and destination addresses, respectively,  $p$  is the packet payload that represents data from upper layer protocols (e.g., TCP, UDP, ICMP, ARP), and  $rt$  stands for the set of IP addresses of routers crossed by the message. Given a state  $st$ , we define the observation function  $obs$  as follows:

$$obs(st) \triangleq \begin{aligned} \wedge l^{st}(ip_{st}) &= st(ip_s) \wedge l^{st}(ip_d) = st(ip_d) \\ \wedge l^{st}(p) &= "a" \wedge l^{st}(rte) = \emptyset \end{aligned}$$

- Variables  $ip_s$  and  $ip_d$  are observable and their values is interpretable. In fact, a network, investigator which analyzes message  $m$  at the network level, is able to interpret IP source and destination address as they occur on the packet.
- Variable  $p$  is observable and its value is non interpretable. In fact, an investigator is able to see the packet payload, but since his observation is performed at the network level,

the investigator does not have the appropriate applications to interpret the payload content (the content may be encrypted, compressed, or encoded appropriately). As a result, the variation of variable  $p$  does not bring any supplementary information to the investigator, and therefore all the values of  $p$  in the different messages are observable under the same label, say  $lb$ .

- Variable  $rte$  is not observable. In fact, since an investigator does not have a complete control under the whole network routers, it is unable to determine from which router the packet has flowed.

*Vulnerability to IP spoofing*:: Starting from Figure 2, we concentrate particularly on hosts  $h_a$  and  $h_f$  that are identified by their IP addresses  $ip_a$  and  $ip_f$ , respectively.  $r_a$  represents the gateway used by  $h_a$  to communicate with  $h_f$  and whose IP address is equal to  $ip_{r_a}$ . We consider two sessions  $s_1$  and  $s_2$  taking place between  $h_a$  and  $h_f$  where each one of them is composed of one packet.  $s_1$ , represents a legitimate session and is equal to  $s_1 = \langle pck_{11} \rangle = \langle (ip_{r_a}, ip_{r_f}, p_x, \langle ip_{r_a}, ip_{r_c}, ip_{r_e} \rangle) \rangle$ . It contains one packet sent from  $ip_{r_a}$  to  $ip_{r_f}$  crossing routers  $r_a, r_c$ , then  $r_e$ , which are identified by their IP addresses  $ip_{r_a}, ip_{r_c}$ , and  $ip_{r_e}$ , respectively.  $s_2$ , holds an IP spoofing attack and is equal to  $s_2 = \langle pck_{21} \rangle = \langle (ip_{r_a}, ip_{r_f}, p_y, \langle ip_{r_c}, ip_{r_e} \rangle) \rangle$ . It contains one packet that appears to be sent from  $ip_a$  to  $ip_f$ . In reality, the packet is sent by  $h_c$ , which impersonates  $h_a$ ; that is why it crossed routers  $r_c$  then  $r_e$  (as identified by their IP addresses  $ip_{r_c}$  and  $ip_{r_e}$ , respectively). Obviously, it can be noticed that predicate  $Spoof$  is false, as  $ip_{r_c}$  does not represent the gateway that is used by  $ip_a$ . Observing the two sessions, an investigator has only access to  $obs(s_1) = obs(s_2) = (ip_a, ip_f, lb, \emptyset)$ . The  $Spoof$  property is not Initial-Simply visible. The occurrence of spoofing attacks in IP protocol cannot be proved.

## VI. CASE STUDY

In this section, we consider a method called ASPM technique as a traceback solution, then we use our visibility-based technique to formally prove that ASPM enables IP spoofing detection and identification of real intruder sources.

ASPM stands for Adaptive and Selective Packet Marking. It was presented in [3] as an IP traceback approach which adapts its behavior according to the characteristics of the processed traffic. Besides that, the selectiveness of the technique allows it instead of marking every traffic unit (i.e., a packet, a datagram or a cell) to exploits a set of properties for every supported protocol, so that: a) only traffic units that matches these properties are marked; and b) all the remaining elements are identified without any mark by exploiting their dependence to the marked elements. The marking is only enabled at the closest interface to the source of the packet on the edge ingress router, reducing processing and bandwidth overload. In the case where a traffic unit arrives to the edge ingress router while being marked (an intruder who wishes to subvert investigation may have inserted an erroneous mark). The latter overwrites the mark and inserts its own one, providing immunity against mark spoofing attacks. In the case of TCP protocol, the ASPM technique marks only segments that contain the TCP SYN flag and the mark is inserted in the data field which is usually not intended to contain

any data. We are aware that such modification may generate alarms at some Intrusion Detection Systems, but it counts for nothing as an investigator have to handle the TCP SYN segments separately to extract and read the mark.

#### A. Modeling TCP protocol with ASPM implementation

TCP protocol guarantees reliable and ordered delivery of sender to receiver data using the IP protocol. It uses the port number to distinguish data for multiple, concurrent applications (e.g. Web server and e-mail server) running on the same host. A virtual circuit is created between the sender and the receiver and is identified by a tuple of four information: source and destination IP addresses, and source and destination TCP ports. Likely to IP protocol, a TCP segment is composed of a header and a payload. The header contains source and destination ports, a sequence number that identifies the position of the first byte (relative to data) in the segment, an acknowledgment number of the first byte (relative to data) that is expected to be sent from the corresponding TCP entity, and a set of flags that are used to mainly manage connections.

We model a TCP segment as a tuple in the form of:  $seg = (p_s, p_d, flg, seq, ack, p_{tcp})$  where  $p_s$  and  $p_d \in \mathbb{N}$  stand for source and destination TCP ports,  $flg$  stands for a subset of a set of TCP flags  $FLAGS = \{SYN, ACK, FIN, ST\}$ ,  $seq$  and  $ack \in \mathbb{N}$  stand for sequence and acknowledgment numbers, and finally  $p_{tcp} \in P_{TCP}$  stands for the TCP payload. Tackling back the IP model, a packet will now look in the form of:  $pck = (ip_{h_s}, ip_{h_d}, (p_s, p_d, flg, seq, ack, p_{tcp}), rte)$  where the IP payload field  $p$  is replaced by the TCP segment  $seg$ .

Roughly speaking, a TCP session is a composed from three parts: The three-way TCP handshake, the data transmission phase, and the connection closing phase. For the sake of readability, we only consider the most interesting phase of a TCP session: the three-way TCP handshake. First, from the security point of view, such mechanism was considered as one of the famous factors that contributed to the proliferation of DDoS attacks. Second, the ASPM technique that we are taking interest is based on the exploitation of the first segment in the TCP handshake. A TCP session between host  $h_s$  and  $h_d$  (identified by their IP addresses  $ip_{h_s}$  and  $ip_{h_d}$ , respectively) will look as  $s = \langle pck_1, pck_2, pck_3 \rangle$  where each packet is described as follows:

- $pck_1 = (ip_{h_s}, ip_{h_d}, (p_s, p_d, \{SYN\}, ISN_s, 0, Head(rte)), rte)$ : The client initiates the connection with the server by sending a TCP segment with a SYN flag set, a sequence number equal to  $ISN_s$ , and an acknowledgment number equal to 0. As we are using the ASPM marking scheme, the TCP payload, which usually does not contain any data in the case where SYN flag is set, will encompass the IP address of the first router crossed by the packet. Such IP address is equal to  $Head(rte)$ .
- $pck_1 = (ip_{h_d}, ip_{h_s}, (p_d, p_s, \{SYN, ACK\}, ISN_d, ISN_s + 1, \emptyset), rte)$ : The server replies with a TCP segment containing its initial sequence number  $ISN_d$ , while setting the acknowledgment number equal to  $ISN_s + 1$ . Both flags  $SYN$  and  $ACK$  are set in the segment. As for the TCP payload, it does not contain any data.

- $pck_3 = (ip_{h_s}, ip_{h_d}, (p_s, p_d, \{ACK\}, ISN_s + 1, ISN_d + 1, \emptyset), rte)$ : The client responds by a TCP segment containing the  $ACK$  flag, a sequence number equal to  $ISN_s + 1$  and an acknowledge number equal to  $ISN_d + 1$ . As for the TCP payload, it does not contain any data.

Note that all the constraints that are imposed for the IP model still hold (e.g.,  $src(pck_1) = dst(pck_2)$ ).

#### B. Investigation on IP spoofing attacks using ASPM technique

Given a TCP segment in the form of  $(ip_{h_s}, ip_{h_d}, (p_s, p_d, flg, seq, ack, p_{tcp}), rte)$ , we define function  $Flags$  and  $Pay$  to extract from an IP packet  $pck$  the set of flags and the payload of a TCP segment, respectively. Let  $\psi$  be a state predicate that is equal to true if the IP packet holds the first TCP segment in the handshake. Its definition is given by  $\psi(pck) \triangleq Flags(pck) = \{SYN\}$ . We define function  $obs$  in a dynamic form depending on predicate  $\psi$ , to characterize what is visible within an IP packet for a network investigator, as follows:

$$\begin{aligned} obs(pck, \psi) &\triangleq \wedge l^{pck}(ip_{h_s}) = pck(ip_{h_s}) \\ &\wedge l^{pck}(ip_{h_d}) = pck(ip_{h_d}) \wedge l^{pck}(rte) = \emptyset \\ &\wedge l^{pck}(p_s) = pck(p_s) \wedge l^{pck}(p_d) = pck(p_d) \\ &\wedge l^{pck}(flg) = pck(flg) \wedge l^{pck}(seq) = pck(seq) \\ &\wedge l^{pck}(ack) = pck(ack) \wedge l^{pck}(p_{tcp}) = Head(rte) \\ &\wedge l^{pck}(p_{tcp}) = p_{tcp} \\ obs(pck, \neg\psi) &\triangleq \wedge l^{pck}(ip_{h_s}) = pck(ip_{h_s}) \\ &\wedge l^{pck}(ip_{h_d}) = pck(ip_{h_d}) \wedge l^{pck}(rte) = \emptyset \\ &\wedge l^{pck}(p_s) = pck(p_s) \wedge l^{pck}(p_d) = pck(p_d) \\ &\wedge l^{pck}(flg) = pck(flg) \wedge l^{pck}(seq) = pck(seq) \\ &\wedge l^{pck}(ack) = pck(ack) \\ &\wedge l^{pck}(p_{tcp}) = pck(Head(rte)) \wedge l^{pck}(p_{tcp}) = pck(lb) \end{aligned}$$

Fields  $ip_{h_s}$ ,  $ip_{h_d}$ ,  $p_s$ ,  $p_d$ ,  $flg$ ,  $seq$ ,  $ack$  are observables and their values is interpretable. The  $rte$  field is invisible due to the same reason discussed with the IP protocol. As for the packet payload  $p_{tcp}$ , it is observed differently from a segment to another. In fact, if the SYN flag is activated (i.e.,  $\psi$  is true), the payload will contain the IP address of the first router crossed by the IP traffic, and hence its value is observable and interpretable. In the opposite case (i.e.,  $\psi$  is false), the payload may or not contain data, that is why it is observable but not interpretable by the investigator. Its value is equal to a fictive label, say  $lb$  for every payload.

*Proving IP spoofing attacks occurrence:* Let us consider the example provided in Section V-C and let us take again the legitimate and malicious sessions between host  $h_a$  and host  $h_f$ . Applying the ASPM technique, the legitimate session will be under form  $s_1 = \langle pck_{11}, pck_{12}, pck_{13} \rangle$ , where:

- $pck_{11} = (ip_{h_a}, ip_{h_f}, (p_s, p_d, \{SYN\}, ISN_s, 0, ip_{r_a}), \langle ip_{r_a}, ip_{r_c}, ip_{r_e} \rangle)$
- $pck_{12} = (ip_{h_f}, ip_{h_a}, (p_d, p_s, \{SYN, ACK\}, ISN_d, 0, \emptyset), \langle ip_{r_a}, ip_{r_c}, ip_{r_e} \rangle)$
- $pck_{13} = (ip_{h_a}, ip_{h_f}, (p_s, p_d, \{ACK\}, ISN_s, 0, \emptyset), \langle ip_{r_a}, ip_{r_c}, ip_{r_e} \rangle)$

If this legitimate session would have been established maliciously to hold an IP spoofing attack, we would have obtained a session  $s_2$  in the form of  $s_2 = \langle pck_{21}, pck_{22}, pck_{23} \rangle$  where  $Spoof(pck_{21})$  is true  $\forall pck_{2i} \in s_2$  and  $Pay(pck_{21}) \neq$

## VII. CONCLUSION

$Pay(pck_{11}) \forall pck_{21} \in s_2$ . As  $Pay(pck_{21})$  is visible and interpretable,  $obs(s_1)$  will be different from  $obs(s_2)$  for every malicious session  $s_2$ . Consider, for instance, the case of an IP spoofing attack, which was initiated by  $h_c$  to impersonate  $h_a$ . We have:

- $pck_{21} = (ip_{h_a}, ip_{h_f}, (p_s, p_d, \{SYN\}, ISN_s, 0, ip_{r_c}), \langle ip_{r_c}, ip_{r_e} \rangle)$
- $pck_{22} = (ip_{h_f}, ip_{h_a}, (p_d, p_s, \{SYN, ACK\}, ISN_d, 0, \emptyset), \langle ip_{r_c}, ip_{r_e} \rangle)$
- $pck_{23} = (ip_{h_a}, ip_{h_f}, (p_s, p_d, \{ACK\}, ISN_s, 0, \emptyset), \langle ip_{r_c}, ip_{r_e} \rangle)$

Obviously, it can be noticed that the payload of the first packet contains  $ip_{r_c}$ , which does not represent the gateway used by  $ip_{h_a}$ .  $obs(s_1)$  is different from  $obs(s_2)$  as from a session  $s_1$ , an investigator has only access to  $obs(s_1)$  equal to:  $obs(s_1) = \langle (ip_{h_a}, ip_{h_f}, (p_s, p_d, \{SYN\}, ISN_s, 0, ip_{r_a}), \emptyset), (ip_{h_f}, ip_{h_a}, (p_d, p_s, \{SYN, ACK\}, ISN_d, 0, \emptyset), \emptyset), (ip_{h_a}, ip_{h_f}, (p_s, p_d, \{ACK\}, ISN_s, 0, \emptyset), \emptyset) \rangle$ , while session  $s_2$ , is observed as  $obs(s_2) = \langle (ip_{h_a}, ip_{h_f}, (p_s, p_d, \{SYN\}, ISN_s, 0, ip_{r_c}), \emptyset), (ip_{h_f}, ip_{h_a}, (p_d, p_s, \{SYN, ACK\}, ISN_d, 0, \emptyset), \emptyset), (ip_{h_a}, ip_{h_f}, (p_s, p_d, \{ACK\}, ISN_s, 0, \emptyset), \emptyset) \rangle$ . As a consequence, predicate *Spoof* is simply always-visible and the occurrence of source address spoofing is thus provable.

*Proving source of IP spoofing attacks:* Unfortunately, the use of TCP protocol does not inhibit IP spoofing attacks, but rather it complicates its success. Let  $I$  be an intruder,  $V$  be a victim, and  $F$  be an entity that the victim believes it communicates with, the attack is performed as follows<sup>1</sup>: a)  $I$  initiates a connection with  $V$  by sending a SYN TCP segment using the  $F$ 's IP address; b)  $V$  replies by sending SYN|ACK TCP segment to  $F$ . However such segment will not reach  $I$ , which is not located in the same network with  $F$ , and  $I$  will not be able to pick up  $V$ 's sequence number; c) Assuming that  $F$  is down (e.g., flooded in a previous attack, or simply turned off) so it does not mess up the attack, the intruder predicts the correct sequence number (exploiting some Operating Systems-based TCP implementation vulnerabilities), sends a correct TCP ACK segment and starts interacting with the victim.

To prove the source of a received packet  $pck_n$  that is part of a TCP session ( $n \geq 2$ ), we consider the predicate property  $SrcProof(pck_n) \triangleq Head(Rte(pck_n)) = Pay(pck_1)$  which is defined to be true if the source of the packet, particularly its gateway, is identified. By identifying the source we mean that the source is proved to be beyond the router whose address appears in the payload of the first packet  $pck_1$  in the same session.

To show that source of the IP spoofing attack can be proved for any TCP segment using the ASPM technique, it suffices to demonstrate that predicate *SrcProof* is initially always-visible. According to our description of TCP protocol (with ASPM implementation) and the investigator observation function defined in the beginning of this sub-section, it can be easily noticed that such property is always true. This makes *SrcProof* property simply always-visible. The identification of source of IP spoofing attacks is thus proved.

We took interest in this paper to formal digital forensic investigation of network-based attacks, particularly the source address spoofing. We proposed to this need a new visibility-based theory for proving proof of attack occurrence and source identification. Three different visibility properties were provided and their use is demonstrated through different propositions. To exemplify the proposal, we considered the particular case of IP spoofing attacks while using the ASPM as a marking technique. Further works, will consider enhancing visibility concept to prove anti-forensic attacks, as well as enhancing the packet switching model to consider traceback of intruder's source in adhoc networks.

## REFERENCES

- [1] W. G. K. II and J. G. Heiser, *Computer Forensics : Incident Response Essentials*. Addison-Wesley Professional, 2001.
- [2] V. Santiraveewan and Y. Permpoontanalarp, "A graph-based methodology for analyzing ip spoofing attack," in *Proceedings of the 18th International Conference on Advanced Information Networking and Applications (AINA 2004)*, (Fukuoka, Japon), 29 - 31 March 2004.
- [3] Y. Djemaiel, S. Rekhis, and N. Boudriga, "Adaptive and Selective Packet Marking in Communication Networks," in *Proceedings of the 9th WSEAS International Conference on Communications*, (Athens, Greece), July 2005.
- [4] F. Zarai, S. Rekhis, N. Boudriga, and K. Zidane, "Sdppm: An IP Traceback Scheme for MANET," in *Proceedings of the 12th IEEE International Conference on Electronics, Circuits and Systems (ICECS'05)*, (Gammarth, Tunisia), December 2005.
- [5] D. Dean, M. Franklin, and A. Stubblefield, "An algebraic approach to IP traceback," *Information and System Security*, vol. 5, no. 2, pp. 119-137, 2002.
- [6] J. W. Bryans, M. Koutny, L. Mazare, and P. Y. A. Ryan, "Opacity generalised to transition systems," Tech. Rep. 868, University of Newcastle upon Tyne, School of Computing Science, Nov 2004.
- [7] S. Rekhis and N. Boudriga, "Opacity: A Theoretical Technique for Digital Investigation," in *Proceedings of 2nd IEEE International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA'06)*, (Damascus, Syria), 24 - 28 April 2006.
- [8] S. Rekhis and N. Boudriga, "A Temporal Logic-based Model for Forensic Investigation in Networked System Security," in *The Third International Workshop*, pp. 325-338, Springer Verlag, LNCS 3685, September 2005.
- [9] S. Schneider and A. Sidiropoulos, "CSP and Anonymity," in *ESORICS '96: Proceedings of the 4th European Symposium on Research in Computer Security*, (London, UK), pp. 198-218, Springer-Verlag, 1996.
- [10] D. Hughes and V. Shmatikov, "Information Hiding, Anonymity and Privacy: a Modular Approach," *Journal of Computer Security*, vol. 12, no. 1, pp. 3-36, 2004.
- [11] A. Boisseau, *Abstractions pour la vérification de propriétés de sécurité de protocoles cryptographiques*. PhD thesis, Laboratoire Spécification et Vérification, ENS Cachan, France, September 2003.
- [12] L. Mazaré, "Using Unification For Opacity Properties," in *Proceedings of the Workshop on Issues in the Theory of Security (WITS'04)*, (Barcelona, Spain), 2004.
- [13] P. Y. R. Jeremy W. Bryans, Maciej Koutny, "Modelling Dynamic Opacity using Petri Nets with Silent Actions," in *Proceedings of the Workshop on Formal Aspects in Security and Trust (FAST)*, (Toulouse, France), Kluwer Academic Press, August 2004.
- [14] L. Lamport, *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.

<sup>1</sup>IP-Spoofing Demystified <http://bau2.uibk.ac.at/matic/ipspoof.htm>