Chapter 10

A PROBABILISTIC NETWORK FORENSIC MODEL FOR EVIDENCE ANALYSIS

Changwei Liu, Anoop Singhal and Duminda Wijesekera

Abstract Modern-day attackers use sophisticated multi-stage and/or multi-host attack techniques and anti-forensic tools to cover their attack traces. Due to the limitations of current intrusion detection systems and forensic analysis tools, evidence often has false positive errors or is incomplete. Additionally, because of the large number of security events, discovering an attack pattern is much like finding a needle in a haystack. Consequently, reconstructing attack scenarios and holding attackers accountable for their activities are major challenges.

This chapter describes a probabilistic model that applies Bayesian networks to construct evidence graphs. The model helps address the problems posed by false positive errors, analyze the reasons for missing evidence and compute the posterior probabilities and false positive rates of attack scenarios constructed using the available evidence. A companion software tool for network forensic analysis was used in conjunction with the probabilistic model. The tool, which is written in Prolog, leverages vulnerability databases and an anti-forensic database similar to the NIST National Vulnerability Database (NVD). The experimental results demonstrate that the model is useful for constructing the most-likely attack scenarios and for managing errors encountered in network forensic analysis.

Keywords: Network forensics, logical evidence graphs, Bayesian networks

1. Introduction

Digital forensic investigators use evidence and contextual facts to formulate attack hypotheses and assess the probability that the facts support or refute the hypotheses [5]. However, due to the limitations of forensic tools and expert knowledge, formulating a hypothesis about

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics XII, IFIP AICT 484, pp. 189–210, 2016.
DOI: 10.1007/978-3-319-46279-0_10

a multi-step, multi-host attack launched on an enterprise network and using quantitative measures to support the hypothesis are major challenges. This chapter describes a model that helps automate the process of constructing and analyzing quantitatively-supportable attack scenarios based on the available evidence. The applicability and utility of the model are demonstrated using a network attack case study.

The proposed method uses a Bayesian network to estimate the likelihood and false positive rates of potential attack scenarios that fit the discovered evidence. Although several researchers have used Bayesian networks for digital evidence modeling [3, 5, 12, 13], their approaches construct Bayesian networks in an *ad hoc* manner. This chapter shows how the proposed method can help automate the process of organizing evidence in a graphical structure (called a logical evidence graph) and apply Bayesian analysis to the entire graph. The method provides attack scenarios with acceptable false positive error rates and dynamically updates the joint posterior probabilities and false positive error rates of attack paths when new items of evidence for the attack paths are presented.

2. Background and Related Work

Bayesian networks have been used to express the credibility and relative weights of digital and non-digital evidence [2, 3, 5, 12, 13]. Several researchers have used Bayesian networks to model dependencies between hypotheses and crime scene evidence, and have employed these models to update the belief probabilities of newly-discovered evidence given the previous evidence [2–4, 12–14].

Digital forensic researchers have used Bayesian networks to reason about evidence and quantify the reliability and traceability of the corresponding hypotheses [5]. However, these Bayesian networks were custom-built without using a uniform model. In contrast, the proposed model is generic and helps address the problems posed by false positive errors, analyze the reasons for missing evidence and compute the posterior probabilities and false positive rates of attack scenarios constructed using the available evidence.

Meanwhile, few, if any, tools directly support the automated construction of Bayesian networks based on the available evidence and estimate belief probabilities and potential error rates. A software tool for network forensic analysis was developed for use with the proposed probabilistic model. The tool, which is written in Prolog, leverages the MulVAL reasoning system [1, 10] and employs system vulnerability databases and an anti-forensic database similar to the NIST National Vulnerability

Database (NVD). The experimental results demonstrate that the tool facilitates the construction of most-likely attack scenarios and the management of errors encountered in network forensic analysis.

3. Logical Evidence Graphs

This section defines logical evidence graphs and shows how rules are designed to correlate attack scenarios with the available evidence. Because logical reasoning is used to link observed attack events and the collected evidence, the evidence graphs are referred to as logical evidence graphs.

Definition 1 (Logical Evidence Graph (LEG)): A logical evidence graph $LEG = (N_f, N_r, N_c, E, L, G)$ is a six-tuple where N_f, N_r and N_c are three disjoint sets of nodes in the graph (called fact, rule and consequence fact nodes, respectively), $E \subseteq ((N_f \cup N_c) \times N_r) \cup (N_r \times N_c)$ is the evidence, L is a mapping from nodes to labels and $G \subseteq N_c$ is a set of observed attack events.

Every rule node has one or more fact nodes or consequence fact nodes from prior attack steps as its parents and a consequence fact node as its only child. Node labels consist of instantiations of rules or sets of predicates specified as follows:

1. A node in N_f is an instantiation of predicates that codify system states, including access privileges, network topology and known vulnerabilities associated with host computers. The following predicates are used:

   - hasAccount(_principal, _host, _account), canAccessFile(_host, _user, _access, _path) and other predicates model access privileges.
   - attackerLocated(_host) and hacl(_src, _dst, _prot, _port) model network topology, including the attacker's location and network reachability information.
   - vulExists(_host, _vulID, _program) and vulProperty(_vulID, _range, _consequence) model node vulnerabilities.

2. A node in N_r describes a single rule of the form $p \leftarrow p_1 \wedge p_2 \cdots \wedge p_n$. The rule head p is an instantiation of a predicate from N_c, which is the child node of N_r in the logical evidence graph. The rule body comprises p_i $(i = 1..n)$, which are predicate instantiations of N_f from the current attack step and N_c from one or more prior attack steps that comprise the parent nodes of N_r.

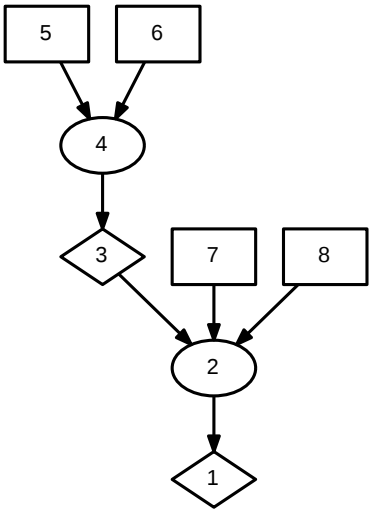

Figure 1. Example logical evidence graph.

3. A node in N_c represents the predicate that codifies the post-attack state as the consequence of an attack step. The two predicates execCode(_host, _user) and netAccess(_machine, _protocol, _port) are used to model the attacker's capability after an attack step. Valid instantiations of these predicates after an attack update valid instantiations of the predicates listed in (1).

Figure 1 shows an example logical evidence graph; Table 1 describes the nodes in Figure 1. In Figure 1, fact, rule and consequence fact nodes are represented as boxes, ellipses and diamonds, respectively. Facts (Nodes 5, 6, 7 and 8) include network topology (Nodes 5 and 6), computer configuration (Node 7) and software vulnerabilities obtained by analyzing evidence captured by forensic tools (Node 8). Rule nodes (Nodes 2 and 4) represent rules that change the attack status using attack steps. These rules, which are based on expert knowledge, are used to link chains of evidence as consequences of attack steps. Linking a chain of evidence using a rule creates an investigator's hypothesis of an attack step given the evidence. Consequence fact nodes (Nodes 1 and 3) codify the attack status obtained from event logs and other forensic tools that record the postconditions of attack steps.

Lines 9 through 17 in Figure 2 describe Rules 1 and 2 in Table 1. The rules use the Prolog notation ":-" to separate the head (consequence fact) and the body (facts). Lines 1 through 8 in Figure 2 list the fact and consequence fact predicates of the two rules.

Table 1. Descriptions of the nodes in Figure 1.

Node	Notation	Resource
1	execCode(workStation1, user)	Evidence obtained from event log
2	THROUGH 3 (remote exploit of a server program)	Rule 1 (Hypothesis 1)
3	netAccess(workStation1, tcp, 4040)	Evidence obtained from event log
4	THROUGH 8 (direct network access)	Rule 2 (Hypothesis 2)
5	hacl(internet, workStation1, tcp, 4040)	Network setup
6	attackerLocated(internet)	Evidence obtained from log
7	networkServiceInfo(workStation1, httpd, tcp, 4040, user)	Computer setup
8	vulExists(workStation1, 'CVE-2009-1918', httpd, remoteExploit, privEscalation)	Exploited vulnerability alert from intrusion detection system

Rule 1 in Lines 9 through 12 represents an attack step that states: if (i) the attacker is located in a "Zone" such as the "internet" (Line 10: attackerLocated(Zone)); and (ii) a host computer "H" can be accessed from the "Zone" using "Protocol" at "Port" (Line 11: hacl(Zone, H, Protocol, Port)); then (iii) host "H" can be accessed from the "Zone" using "Protocol" at "Port" (Line 9: netAccess(H, Protocol, Port)) via (iv) "direct network access" (Line 12: rule description).

Rule 2 in Lines 13 through 17 states: if (i) a host has a software vulnerability that can be remotely exploited (Line 14: vulExists(H, _, Software, remoteExploit, privEscalation)); and (ii) the host can be reached using "Protocol" at "Port" with privilege "Perm" (Line 15: networkServiceInfo(H, Software, Protocol, Port, Perm)); and (iii) the attacker can access host using "Protocol" at "Port" (Line 16: netAccess(H, Protocol, Port)); then (iv) the attacker can remotely exploit host "H" and obtain privilege "Perm" (Line 13: execCode(H, Perm)) via (v) "remote exploit of a server program" (Line 17: rule description).

4. Computing Probabilities

Bayesian networks can be represented as directed acyclic graphs whose nodes represent random variables (events or evidence in this work) and arcs model direct dependencies between random variables [11]. Every

```
Rule Head – Post-attack status as derived fact obtained via evidence analysis
1. Consequence: execCode(_host, _user).
2. Consequence: netAccess(_machine, _protocol, _port).
Rule Body – Access privilege
3. Fact: hacl(_src, _dst, _prot, _port).
Rule Body – Software vulnerability obtained from a forensic tool
4. Fact: vulExists(_host, _vulID, _program).
5. Fact: vulProperty(_vulID, _range, _consequence).
Rule Body – Network topology
6. Fact: hacl(_src, _dst, _prot, _port).
7. Fact: attackerLocated(_host).
Rule Body – Computer configuration
8. Fact: hasAccount(_principal, _host, _account).

Rule 1:
9. (netAccess(H, Protocol, Port) :-
10. attackerLocated(Zone),
11. hacl(Zone, H, Protocol, Port)),
12. rule_desc('direct network access', 1.0).

Rule 2:
13. (execCode(H, Perm) :-
14. vulExists(H, _, Software, remoteExploit, privEscalation),
15. networkServiceInfo(H, Software, Protocol, Port, Perm),
16. netAccess(H, Protocol, Port)),
17. rule_desc('remote exploit of a server program', 1.0).
```

Figure 2. Example rules expressing attack techniques.

node has a table that provides the conditional probability of the node's variable given the combination of the states of its parent variables.

Definition 2 (Bayesian Network (BN)): Let X_1, X_2, $\cdots$, X_n be n random variables connected in a directed acyclic graph. Then, the joint probability distribution of $X_1, X_2, ..., X_n$ can be computed using the Bayesian formula:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} [P(X_i)|parent(P(X_i))] \tag{1}$$

A Bayesian network helps model and visualize dependencies between a hypothesis and evidence, and calculate the revised probability when new evidence is presented [9]. Figure 3 presents a causal view of hypothesis H and evidence E. Bayes' theorem can be used to update an investigator's belief about hypothesis H when evidence E is observed:

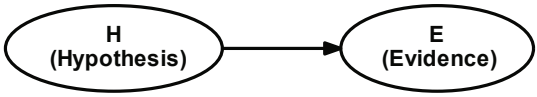

Figure 3. Causal view of evidence.

$$P(H|E) = \frac{P(H) \cdot P(E|H)}{P(E)} = \frac{P(H) \cdot P(E|H)}{P(E|H) \cdot P(H) + P(E|\neg H) \cdot P(\neg H)} \quad (2)$$

where $P(H|E)$ is the posterior probability of an investigator's belief in hypothesis H given evidence E. $P(E|H)$, which is based on expert knowledge, is a likelihood function that assesses the probability of evidence assuming the truth of H. $P(H)$ is the prior probability of H when the evidence has not been discovered and $P(E) = P(E|H) \cdot P(H) + P(E|\neg H) \cdot P(\neg H)$ is the probability of the evidence regardless of expert knowledge about H and is referred to as a normalizing constant [5, 9].

4.1 Computing $P(H|E)$

A logical evidence graph involves the serial application of attack steps that are mapped to a Bayesian network as follows:

- N_c as the child of the corresponding N_r shows that an attack step has occurred.
- N_r is the hypothesis of the attack step and is denoted by H.
- N_f from the current attack step and $N_c\prime$ from the previous attack step as the parents of N_r correspond to the attack evidence, showing the exploited vulnerability and the privilege the attacker used to launch the attack step.
- N_c propagates the dependency between the current attack step and the next attack step. N_c is also the precondition of the next attack step.

Computing $P(H|E)$ for a Consequence Fact Node. Equation (2) can be used to compute $P(H|E)$ for a consequence fact node of a single attack step when the previous attack step has not been considered. Because the rule node N_r provides the hypothesis H and both the fact node N_f and the consequence fact node from a previous attack step $N_c\prime$ provide evidence E, the application of Bayes' theorem yields:

$$P(H|E) = P(N_r|E) = \frac{P(N_r) \cdot P(E|N_r)}{P(E)} \tag{3}$$

The fact nodes from the current attack step and the consequence fact node from a previous attack step are independent of each other. They constitute the body of a rule, deriving the consequence fact node for the current attack step as the head of the rule. Consequently, their logical conjunction provides the conditions that are used to arrive at the rule conclusion. Accordingly, if a rule node has k parents $N_{p1}, N_{p2}, \ldots, N_{pk}$ that are independent, then $P(E) = P(N_{p1}, N_{p2}, \ldots, N_{pk}) = P(N_{p1} \cap N_{p2} \cap \cdots \cap N_{pk}) = P(N_{p1}) \cdot P(N_{p2}) \cdots P(N_{pk})$ (note that $\cap$ denotes the AND operator). Due to the independence, given rule N_r, $P(E|N_r) = P(N_{p1}, N_{p2}, \ldots, N_{pk}|N_r) = P(N_{p1}|N_r) \cdot P(N_{p2}|N_r) \cdots P(N_{pk}|N_r)$. Hence, by applying Equation (3), where H is N_r and E is $N_{p1} \cap N_{p2} \cap \cdots \cap N_{pk}$, $P(H|E)$ for a consequence fact node is computed as:

$$\begin{aligned} P(H|E) &= P(N_r|N_{p1}, N_{p2}, \cdots, N_{pk}) \\ &= \frac{P(N_r) \cdot P(N_{p1}|N_r).P(N_{p2}|N_r) \cdots P(N_{pk}|N_r)}{P(N_{p1}) \cdot P(N_{p2}) \cdots P(N_{pk})} \end{aligned} \tag{4}$$

However, because $P(E|N_r)$ represents the subjective judgment of a forensic investigator, it would be difficult for human experts to assign $P(N_{p1}|N_r)$, $P(N_{p2}|N_r)$, $\cdots$, $P(N_{pk}|N_r)$ separately. Therefore, the forensic investigator has the discretion to use Equation (3) to compute $P(E|N_r)$ directly.

Computing $P(H|E)$ for the Entire Graph. Next, it is necessary to compute $P(H|E)$ for the entire logical evidence graph comprising the attack paths. Any chosen attack path in a logical evidence graph is a serial application of attack steps. An attack step only depends on its direct parent attack steps and is independent of all the ancestor attack steps in the attack path. Upon applying Definition 2, the following equation is obtained:

$$\begin{aligned} P(H|E) &= P(H_1, H_2 \cdots H_n|E_1, E_2, E_3 \cdots E_n) \\ &= P(S_1)P(S_2|S_1) \cdots P(S_n|S_{n-1}) \end{aligned} \tag{5}$$

where S_i $(i = 1..n)$ denotes the i^{th} attack step in an attack path.

Let $N_{i,f}, N_{i,r}$ and $N_{i,c}$ be the fact, rule and consequence fact nodes, respectively, at the i^{th} attack step. Then, Equation (5) may be written as:

$$\begin{aligned} P(H|E) &= P(S_1) \cdots P(S_i|S_{i-1}) \cdots P(S_n|S_{n-1}) \\ &= P(N_{1,r}|N_{1,f}) \cdots P(N_{i,r}|N_{i-1,c}, N_{i,p}) \cdots P(N_{n,r}|N_{n-1,c}, N_{n,p}) \\ &= \frac{P(N_{1,r})P(N_{1,f}|N_{1,r})}{P(N_{1,f})} \cdots \frac{P(N_{n,r})P(N_{n-1,c}, N_{n,f}|N_{n,r})}{P(N_{n-1,c}, N_{n,f})} \end{aligned} \tag{6}$$

where $P(S_1)P(S_2|S_1) \cdots P(S_i|S_{i-1})$ is the joint posterior probability of the previous i attack steps (i.e., $1..i$) given all the evidence from the attack steps (e.g., evidence for attack step 1 is $N_{1,f}$; the evidence for attack step i includes $N_{i-1,c}$ and $N_{i,f}$ where $i = 2..n$.

$P(S_1)P(S_2|S_1) \cdots P(S_i|S_{i-1})$ is propagated to the $i + 1^{th}$ attack step by the consequence fact node $N_{i,c}$, which is also the precondition of the $i + 1^{th}$ attack step. Algorithm 1 formalizes the computation of $P(H|E)$ for the entire logical evidence graph.

Because a logical evidence graph may have several attack paths, to compute the posterior probability of each attack path, all the nodes are marked as WHITE (Lines 2 through 4 in Algorithm 1) and all the fact nodes are pushed from the first attack step of all attack paths to an empty queue (Lines 1 and 5). If the queue is not empty (Line 7), a fact node is taken out of the queue (Line 8) and a check is made to see if its child that is a rule node is WHITE (Lines 9 and 10). If the rule node is WHITE, a new attack path is created (Line 11), upon which Equation (6) is used recursively to compute the joint posterior probability of the entire attack path (Lines 16 through 30) and the node is marked as BLACK (Line 13) after the computation of the function PATH($N_{1,r}$) in Line 12 is complete. The above process is repeated until the queue holding the fact nodes from the first attack steps of all the attack paths is empty.

4.2 Computing the False Positive Rate

False positive and false negative errors exist in logical evidence graphs. A false negative arises when the investigator believes that the event was not caused by an attack, but was the result of an attack. A false positive arises when the investigator believes that an event was caused by an attack, but was not. Clearly, it is necessary to estimate both types of errors.

Because a logical evidence graph is constructed using attack evidence chosen by the forensic investigator, there is always the possibility of false positive errors. Therefore, the cumulative false positive rate of the constructed attack paths must be computed. False negative errors are not computed in this work.

Algorithm 1. : Computing $P(H|E)$ for the entire graph.

Input: A LEG = (N_r, N_f, N_c, E, L, G) with multiple attack paths and $P(N_{i,r})$ $(i = 1..n)$, $P(N_{1,f}|N_{1,r})$, $P(N_{1,f})$, $P(N_{i-1,c}, N_{i,f}|N_{i,r})$, $P(N_{i-1,c}, Ni, f)$ $(i = 2..n)$ obtained from expert knowledge about each attack path. $N_{1,f}$, $N_{i-1,c}$ and $N_{i,f}$ $(i \geq 2)$ correspond to evidence E. $N_{i,r}$ $(i \geq 1)$ corresponds to H.

Output: The joint posterior probability of the hypothesis of every attack path $P(H|E)$=$P(H_1, H_2 \cdots H_n|E_1, E_2, E_3 \cdots E_n)$ given all the evidence represented by fact nodes $N_{i,f}$ and $N_{i,c}$ $(i = 1..n)$. ($P(H|E)$ is written as P in the algorithm.

1: $Q_g \leftarrow \emptyset$ ▷ set Q_g to empty
2: **for** each node n ∈ LEG **do**
3: color[n] ← WHITE ▷ mark every node in the graph as white
4: **end for**
5: ENQUEUE($Q_g, N_{1,f}$) ▷ push all fact nodes from the first attack step to queue Q_g
6: j ← 0 ▷ use j to identify the attack path being computed
7: **while** $Q_g \neq \emptyset$ **do** ▷ when queue Q_g is not empty
8: n ← DEQUEUE(Q_g) ▷ remove fact node n
9: $N_{1,r}$ ← child[n] ▷ find a rule node as the child node of n
10: **if** ($color[N_{1,r}] \equiv$ WHITE) **then** ▷ if the rule node is not traversed (white)
11: j ← j+1 ▷ must be a new attack path
12: P[j] ← PATH($N_{1,r}$) ▷ compute joint posterior probability of the path
13: color[$N_{1,r}$] ← BLACK ▷ mark the rule node as black
14: **end if**
15: **end while**

16: **PATH**($N_{1,r}$) { ▷ compute the posterior probability of an attack path
17: $N_{1,c}$ ← child[$N_{1,r}$] ▷ consequence fact node of the first attack step
18: E ← parents[$N_{1,r}$] ▷ E is the evidence for the first attack step
19: P[$N_{1,c}$] ← $\frac{P(N_{1,r})P(E|N_{1,r})}{P(E)}$ ▷ probability of the first attack step
20: color[E] ← BLACK ▷ mark all traversed evidence as black
21: P ← P[$N_{1,c}$] ▷ use P to do the recursive computation
22: **for** $i \leftarrow 2$ to n **do** ▷ from the second attack step to the last attack step
23: $N_{i,r}$ ← child[$N_{i-1,c}$] ▷ rule node as H of the i^{th} attack step
24: E ← parents[$N_{i,r}$] ▷ evidence for the i^{th} attack step
25: $N_{i,c}$ ← child[$N_{i,r}$] ▷ consequence fact node of the i^{th} attack step
26: P[$N_{i,c}$] ← $P(N_{i,r}|E) \leftarrow \frac{P(N_{i,r})P(E|N_{i,r})}{P(E)}$ ▷ posterior probability of the i^{th} attack step
27: color[E] ← BLACK ▷ mark all traversed evidence as black
28: $P \leftarrow P \times P(N_{i,c})$ ▷ joint posterior possibility of attack steps $(1..i)$
29: **end for**
30: **Return** P ▷ return the posterior attack possibility of the attack path

The individual false positive estimate for an attack step is expressed as $P(E|\neg H)$, where $\neg H$ is the alternative hypothesis, usually written as "not H," and the value of $P(E|\neg H)$ can be obtained from expert

knowledge. To demonstrate the computation of the cumulative false positive rate of an entire attack path, let $N_{i,f}$, $N_{i,r}$ and $N_{i,c}$ correspond to the fact, rule and consequence fact nodes, respectively, of the i^{th} attack step. Then, the cumulative false positive rate of the entire attack path is computed as follows:

$$\begin{aligned} P(E|\neg H) &= P(E_1, E_2, \cdots, E_n|\neg(H_1, H_2, \cdots, H_n)) \\ &= \bigcup_{i=1}^{n} P(E_i|\neg N_{i,r}) \\ &= 1 - (\cdots(1 - (1 - P(E_2|\neg N_{2,r}) \cdot (1 - P(E_1|\neg N_{1,r}))))) \\ &\quad \cdot (1 - P(E_n|\neg N_{n,r})) \end{aligned} \tag{7}$$

Note that all the evidence supporting an attack step is independent of the evidence supporting the other attack steps.

As mentioned above, E_1 in Equation (7) is $N_{1,f}$ and E_i includes $N_{i-1,c}$ and $N_{i,f}$ $(i = 2..n)$. The symbol $\cup$ denotes the noisy-OR operator [7]. For a serial connection, if any of the attack steps is a false positive, then the entire attack path is considered to be a false positive. Algorithm 2 formalizes the computation of $P(E|\neg H)$ for the entire evidence graph.

Lines 1 through 15 in Algorithm 2 are the same as in Algorithm 1 (i.e., they find a new attack path). Lines 16 through 29 use Equation (7) to recursively compute the cumulative false positive rate of an entire attack path.

5. Case Study

This case study demonstrates how probabilistic attack scenarios can be reconstructed using Bayesian analysis [13].

5.1 Experimental Network

Figure 4 shows the experimental network [6] used to generate a logical evidence graph from post-attack evidence. In the network, the external Firewall 1 controls Internet access to a network containing a Portal Web Server and Product Web Server. The internal Firewall 2 controls access to a SQL Database Server that can be accessed from the web servers and workstations. The Administrator Workstation has administrative privileges to the Portal Web Server that supports a forum for users to chat with the administrator. In the experiment, the Portal and Product Web Servers and the Database Server were configured to log all accesses and queries as events and Snort was used as the intrusion detection

Algorithm 2. : Computing $P(E|\neg H)$ for the entire graph.

Input: A LEG = (N_r, N_f, N_c, E, L, G) and $P(N_{1,f}|N_{1,r})$ as $P(E_1|H_1)$, $P(N_{i-1,c}, N_{i,f}|N_{i,r})$ as $P(E_i|H_i)$ $(i = 2..n)$ for every attack path.

Output: The cumulative false positive rate of each attack path $P(E|\neg H) = P(E_1, E_2, \cdots, E_n|\neg(H_1, H_2 \cdots H_n$. $P(E|\neg H)$ is written as P_f in the algorithm.

```
1: Q_g ← Ø                                              ▷ set Q_g to empty
2: for each node n ∈ LEG do
3:     color[n] ← WHITE                                 ▷ mark every node in the graph as white
4: end for
5: ENQUEUE(Q_g, N_{1,f})   ▷ push all fact nodes from the first attack step to queue Q_g
6: j ← 0                                  ▷ use j to identify the attack path being computed
7: while Q_g ≠ Ø do                                     ▷ when queue Q_g is not empty
8:     n ← DEQUEUE(Q_g )                                ▷ remove fact node n
9:     N_{1,r} ← child[n]                               ▷ find a rule node as the child node of n
10:    if (color[N_{1,r}] ≡ WHITE) then   ▷ if the rule node is not traversed (white)
11:        j ← j+1                                      ▷ must be a new attack path
12:        P_r[j] ← PATH(N_{1,r})   ▷ compute the cumulative false positive rate of the path
13:        color[N_{1,r}] ← BLACK                       ▷ mark the rule node as black
14:    end if
15: end while

16: PATH(N_{1,r}) {                  ▷ compute the false positive rate of an attack path
17: N_{1,c} ← child[N_{1,r}]                ▷ consequence fact node of the first attack step
18: E ← parents[N_{1,r}]                    ▷ E is the evidence for the first attack step
19: P[N_{1,c}] ← P(E|¬N_{1,r})              ▷ false positive rate of the first attack step
20: color[E] ← BLACK                        ▷ mark all traversed evidence as black
21: P_f ← P[N_{1,c}]                        ▷ use P_f to do the recursive computation
22: for i ← 2 to n do          ▷ from the second attack step to the last attack step
23:     N_{i,r} ← child[N_{i-1,c}]          ▷ rule node as H of the i^th attack step
24:     N_{i,c} ← child[N_{i,r}]            ▷ consequence fact node of the i^th attack step
25:     E ← parents[N_{i,r}]                ▷ evidence for the i^th attack step
26:     P_f ← 1 − (1 − P_f) × (1 − P(E|¬N_{i,r}))   ▷ cumulative false positive rate
27:     color[E] ← BLACK                    ▷ mark all traversed evidence as black
28: end for
29: Return P_f          ▷ return the cumulative false positive rate of the attack path
```

system. The evidence in the case study constituted the logged events and intrusion alerts.

By exploiting vulnerabilities in a Windows workstation and a web server with access to the Database Server, the attacker was able to successfully launch two attacks on the Database Server and a cross-site scripting (XSS) attack on the Administrator Workstation. The attacks involve: (i) using a compromised workstation to access the Database

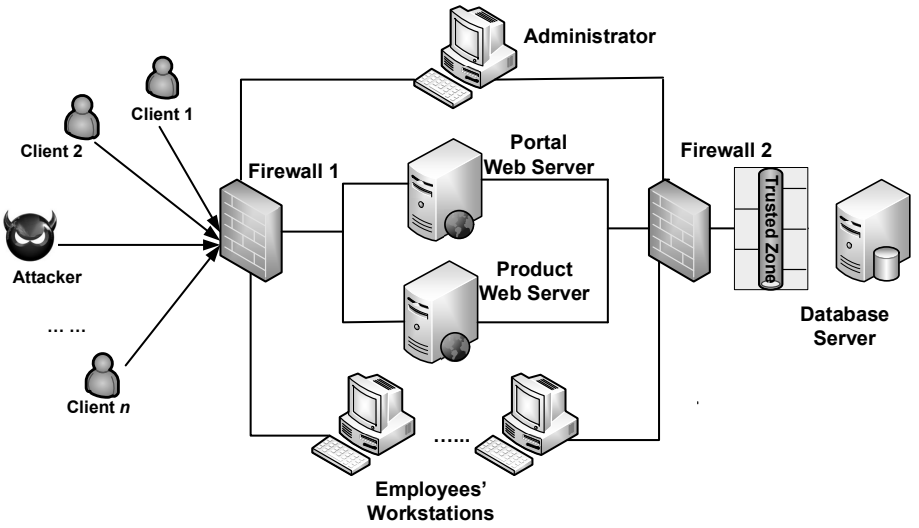

Figure 4. Experimental network.

Table 2. Evidence comprising logged events and alerts.

Timestamp	Source IP	Destination IP	Content	Vulnerability
08\13-12:26:10	129.174.124.122 Attacker	129.174.124.184 Workstation1	SHELLCODE x86 inc ebx NOOP	CVE-2009-1918
08\13-12:27:37	129.174.124.122 Attacker	129.174.124.185 Workstation2	SHELLCODE x86 inc ebx NOOP	CVE-2009-1918
08\13-14:37:27	129.174.124.122 Attacker	129.174.124.53 Product Web Server	SQL Injection Attempt	CWE89
08\13-16:19:56	129.174.124.122 Attacker	129.174.124.137 Administrator	Cross Site Scripting	XSS
08\13-14:37:29	129.174.124.53 Product Web Server	129.174.124.35 Database Server	name='Alice' AND password="alice' OR '1'='1'	CWE89
...	...	...	...	...

Server (CVE-2009-1918); (ii) exploiting a vulnerability in the web application (CWE89) in the Product Web Server to attack the Database Server; and (iii) exploiting the XSS vulnerability in the chat forum hosted by the Portal Web Server to steal the Administrator's session ID, enabling the attacker to send phishing emails to the clients and trick them to update their confidential information.

Table 3. Post-attack evidence.

Timestamp	Attacked Computer	Attack Event	Post Attack Status
08\-14:37:29	129.174.124.35 Database Server	Information Retrieved Maliciously	Malicious Access
...	...	...	...

```
Observed Attack Events
1. attackGoal(execCode(workStation1, _)).
2. attackGoal(execCode(dbServer, user)).
3. attackGoal(execCode(clients, user)).

Network Topology
4. attackerLocated(internet).
5. hacl(internet, webServer, tcp, 80).
6. hacl(internet, workStation1, tcp, _).
7. hacl(webServer, dbServer, tcp, 3660).
8. hacl(internet, admin, _, _).
9. hacl(admin, clients, _, _).
10. hacl(workStation1, dbServer, _, _).

Computer Configuration
11. hasAccount(employee, workStation1, user).
12. networkServiceInfo(webServer, httpd, tcp, 80, user).
13. networkServiceInfo(dbServer, httpd, tcp, 3660, user).
14. networkServiceInfo(workStation1, httpd, tcp, 4040, user).

Information from Table 2 (Software Vulnerability)
15. vulExists(webServer, 'CWE89', httpd).
16. vulProperty('CWE89', remoteExploit, privEscalation).
17. vulExists(dbServer, 'CWE89', httpd).
18. vulProperty('CWE89', remoteExploit, privEscalation).
19. vulExists(workStation1, 'CVE-2009-1918', httpd).
20. vulProperty('CVE-2009-1918', remoteExploit, privEscalation).
21. timeOrder(webServer, dbServer, 14.3727, 14.3729).
```

Figure 5. Input file for generating the logical evidence graph.

The logging system and intrusion detection system captured evidence of network attack activities. Table 2 presents the processed data. Table 3 presents the post-attack evidence obtained using forensic tools.

5.2 Constructing the Graph

To employ the Prolog-based rules for evidence graph construction, the evidence and system state were codified as instantiations of the rule

predicates as shown in Figure 5. In Figure 5, Lines 1 through 3 model evidence related to the post-attack status (Table 3), Lines 4 through 10 model the network topology (system setup), Lines 11 through 14 model system configurations and Lines 15 through 21 model vulnerabilities obtained from the captured evidence (Table 2).

The input file with rules representing generic attack techniques was submitted to the reasoning system along with two databases, including an anti-forensic database [6] and MITRE's CVE [8], to remove irrelevant evidence and obtain explanations for any missing evidence.

The results are: (i) according to the CVE database, Workstation 2, which is a Linux machine using Firefox as the web browser, rendered an attack using CVE-2009-1918 unsuccessful because the exploit only succeeds on Windows Internet Explorer; (ii) a new attack path expressing that the attacker launched phishing attacks at the clients using the Administrator's stolen session ID was found; and (iii) an attack path between the compromised Workstation 1 and the Database Server was found.

The network forensic analysis tool created the logical evidence graph shown in Figure 6. The nodes in Figure 6 are described in Tables 4 and 5. The third column of each table lists the logical operators used to distinguish fact nodes, rule nodes and consequence fact nodes. A fact node is marked as LEAF, a rule node is marked as OR and a consequence fact node is marked as AND.

Figure 6 has three attack paths:

- The attacker used an XSS attack to steal the Administrator's session ID and obtain administrator privileges to send phishing emails to clients (Nodes: $11 \rightarrow 9 \rightarrow 8 \rightarrow 7 \rightarrow 6 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$) (Left).
- The attacker used a buffer overflow vulnerability (CVE-2009-1918) to compromise a workstation and then obtain access to the Database Server (Nodes: $34 \rightarrow 33 \rightarrow 32 \rightarrow 31 \rightarrow 30 \rightarrow 28 \rightarrow 18 \rightarrow 17 \rightarrow 16$) (Middle).
- The attacker used a web application that does not sanitize user input (CWE89) to launch a SQL injection attack at the Database Server (Nodes: $11 \rightarrow 24 \rightarrow 23 \rightarrow 22 \rightarrow 21 \rightarrow 19 \rightarrow 18 \rightarrow 17 \rightarrow 16$) (Right).

5.3 Computations

This section uses Algorithms 1 and 2 to compute $P(H|E_1, E_2, \cdots, E_n)$ and $P(E_1, E_2, \cdots, E_n|\neg H)$ for the attack paths in Figure 6 (H corresponds to $H_1 \cap H_2 \cdots \cap H_n$).

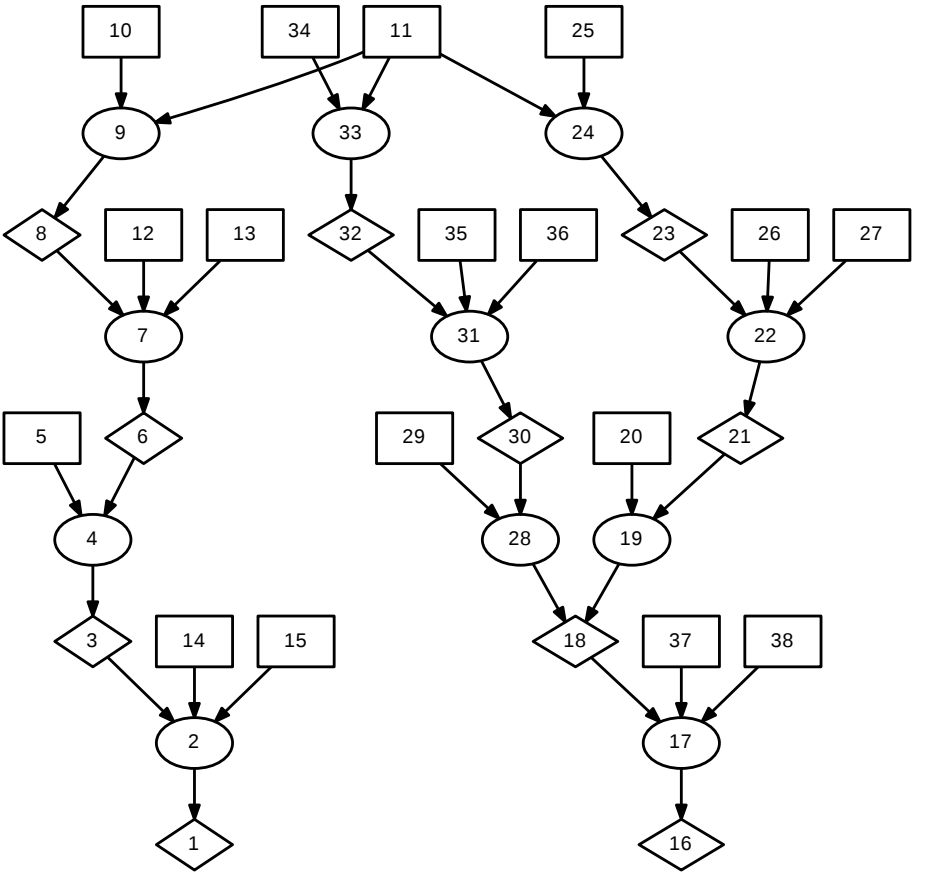

Figure 6. Constructed logical evidence graph.

Using Algorithm 1 to Compute $P(H|E1, E2..En)$. Algorithm 1 requires $P(N_{1,r})$, $P(N_{1,f})$, $P(N_{1,f}|N_{1,r})$, $P(N_{i,r})$, $P(N_{i-1,c}, N_{i,f}|N_{i,r})$, $P(N_{i-1,c}, N_{i,f})$ $(i = 2..n)$. All these probabilities are derived from expert knowledge. To minimize subjectivity, the average value of the probability based on the judgments of multiple experts should be computed [5]. Because the case study is intended to demonstrate the computations, for simplicity, all $P(H_i) = P(\neg H_i) = 50\%$, $P(E_i) = k \in [0, 1]$ (k obviously would differ for different evidence in a real scenario). Also, the $P(E_i|H_i)$ values were assigned based on the judgment of the authors of this chapter; the probability values of $P(E_i|H_i)$ are listed in Table 6.

Table 4. Descriptions of the nodes in Figure 6.

Node	Notation	Relation
1	execCode(clients, user)	OR
2	THROUGH 3 (remote exploit of a server program)	AND
3	netAccess(clients, tcp, _)	OR
4	THROUGH 7 (multi-hop access)	AND
5	hacl(admin,clients, tcp, _)	LEAF
6	execCode(admin, apache)	OR
7	THROUGH 3 (remote exploit of a server program)	AND
8	netAccess(admin, tcp, 80)	OR
9	THROUGH 8 (direct network access)	AND
10	hacl(internet, admin, tcp, 80)	LEAF
11	attackerLocated(internet)	LEAF
12	networkServiceInfo(admin, httpd, tcp, 80, apache)	LEAF
13	vulExists(admin,'XSS', httpd, remoteExploit, privEscalation)	LEAF
14	networkServiceInfo(clients, httpd, tcp, _, user)	LEAF
15	vulExists(clients, 'Phishing', httpd, remoteExploit, privEscalation)	LEAF
16	execCode(dbServer, user)	OR
17	THROUGH 3 (remote exploit of a server program)	AND
18	netAccess(dbServer, tcp, 3660)	OR
19	THROUGH 7 (multi-hop access)	AND
20	hacl(webServer, dbServer, tcp, 3660)	LEAF
21	execCode(webServer, user)	OR
22	THROUGH 3 (remote exploit of a server program)	AND
23	netAccess(webServer, tcp, 80)	OR

Thus, $P(H_i|E_i)$ for each attack step without considering the other attack steps is given by:

$$\begin{aligned} \frac{P(H_i)P(E_i|H_i)}{P(E_i)} &= \frac{0.5 \cdot P(E_i|H_i)}{k} \\ &= \frac{P(E_i|H_i)}{2k} \\ &= c \cdot P(E_i|H_i) \end{aligned} \tag{8}$$

Table 5. Descriptions of the nodes in Figure 6 (continued).

Node	Notation	Relation
24	THROUGH 8 (direct network access)	AND
25	hacl(internet, webServer, tcp, 80)	LEAF
26	networkServiceInfo(webServer, httpd, tcp, 80, user)	LEAF
27	vulExists(webServer, 'CWE89', httpd, remoteExploit, privEscalation)	LEAF
28	THROUGH 7 (multi-hop access)	AND
29	hacl(workStation1, dbServer, tcp, 3660)	LEAF
30	execCode(workStation1, user)	OR
31	THROUGH 3 (remote exploit of a server program)	AND
32	netAccess(workStation1, tcp, 4040)	OR
33	THROUGH 8 (direct network access)	AND
34	hacl(internet, workStation1, tcp, 4040)	LEAF
35	networkServiceInfo(workStation1, httpd, tcp, 4040, user)	LEAF
36	vulExists(workStation1, 'CVE-2009-1918', httpd, remoteExploit, privEscalation)	LEAF

where $c = \frac{1}{2k}$. Algorithm 1 is used to compute $P(H|E_1, E_2, \cdots E_n)$ as shown in the last column of Table 6.

Note that Node 17 has two joint posterior probabilities, which are from the middle path and right path, respectively. Note also that the middle attack path has a lower probability than the right attack path. This is because the attacker destroyed the evidence obtained from the middle path that involved using a compromised workstation to gain access to the database. Additionally, the $P(E|H)$ value is lower. Therefore, the corresponding hypothesized attack path has a much lower probability $P(H|E_1, E_2, \cdots, E_n)$. In reality, it is unlikely that the same attacker would attempt a different attack path to attack the same target if the previous attack had already succeeded. A possible scenario is that the first attack path was not anticipated, so the attacker attempted to launch the attack via the second attack path. The joint posterior probability $P(H|E_1, E_2, \cdots, E_n)$ could help an investigator select the most pertinent attack path.

Table 6. Computation of $P(H|E_1, \cdots, E_n)$ for the attack paths.

Attack Step	**Attack Step 1**				**Attack Step 2**			
	H1	$\mathbf{P(E_1|H_1)}$	$\mathbf{P(H_1|E_1)}$	$\mathbf{P(H|E_1)}$	$\mathbf{H_2}$	$\mathbf{P(E_2|H_2)}$	$\mathbf{P(H_2|E_2)}$	$\mathbf{P(H|E_1, E_2)}$
Left	Node 9	0.90	0.90c	0.90c	Node 7	0.80	0.80c	$0.720c^2$
Middle	Node 33	0.99	0.99c	0.99c	Node 31	0.87	0.87c	$0.861c^2$
Right	Node 24	0.99	0.99c	0.99c	Node 22	0.85	0.85c	$0.842c^2$
Attack Step	**Attack Step 3**				**Attack Step 4**			
	$\mathbf{H_3}$	$\mathbf{P(E_3|H_3)}$	$\mathbf{P(H_3|E_3)}$	$\mathbf{P(H|E_1, E_2, E_3)}$	$\mathbf{H_4}$	$\mathbf{P(E_4|H_4)}$	$\mathbf{P(H_4|E_4)}$	$\mathbf{P(H|E_1, E_2, E_3, E_4)}$
Left	Node 4	0.90	0.90c	$0.648c^3$	Node 2	0.75	0.75c	$0.486c^4$
Middle	Node 28	0.87	0.87c	$0.750c^3$	Node 17	0.75	0.75c	$0.563c^4$
Right	Node 19	0.97	0.97c	$0.817c^3$	Node 17	0.95	0.95c	$0.776c^4$

Using Algorithm 2 to Compute $P(E_1, E_2, \cdots, E_n|\neg H)$. Algorithm 2 requires $P(N_{1,f}|N_{1,r})$ corresponding to $P(E_1|\neg H_1)$ and $P(N_{i-1,c}, N_{i,f}|N_{i,r})$ corresponding to $P(E_i|\neg H_i)$ $(i = 2..n)$ to recursively compute $P(E_1, E_2, \cdots, E_n|\neg H)$. As an example, $P(E_i|\neg H_i)$ was assigned to each attack step in the three attack paths and $P(E_1, E_2, \cdots, E_n|\neg H)$ was computed (Table 7). The results show that the right attack path has the smallest cumulative false positive estimate.

Values computed for $P(H|E_1, E_2, \cdots, E_n)$ and $P(E_1, E_2, \cdots, E_n|\neg H)$ show the beliefs in the three constructed attack paths given the collected evidence. The right attack path (Nodes: $11 \rightarrow 24 \rightarrow 23 \rightarrow 22 \rightarrow 21 \rightarrow 19 \rightarrow 18 \rightarrow 17 \rightarrow 16$) is the most convincing attack path because it has the largest $P(H|E)$ value and smallest $P(E|\neg H)$ value. The left attack path is not convincing because its joint posterior probability is less than $0.5c^4$. The middle path is not so convincing because it has a higher cumulative false positive rate, suggesting that the attack path should be re-evaluated to determine if it corresponds to a real attack scenario.

6. Conclusions

The principal contribution of this research is a method that automates the construction of a logical evidence graph using rules and mapping the graph to a Bayesian network so that the joint posterior probabilities and false positive rates corresponding to the constructed attack paths can be computed automatically. The case study demonstrates how the method can guide forensic investigators to identify the most likely attack scenarios that fit the available evidence. Also, the case study shows that the method and the companion tool can reduce the time and effort involved in network forensic investigations. However, the method cannot deal with zero-day attacks; future research will attempt to extend the underlying model to address this deficiency.

References

[1] Argus Cyber Security Lab, MulVAL: A Logic-Based Enterprise Network Security Analyzer, Department of Computer Science and Engineering, University of South Florida, Tampa, Florida (`www.arguslab.org/mulval.html`), 2016.

Table 7. Computation of $P(E_1, E_2, \cdots, E_n|\neg H)$ for the attack paths.

Attack Step	**Attack Step 1**			**Attack Step 2**		
	$\mathbf{H_1}$	$\mathbf{P(E_1|\neg H_1)}$	$\mathbf{P(E_1|\neg H_1)}$	$\mathbf{H_2}$	$\mathbf{P(E_2|\neg H_2)}$	$\mathbf{P(E_1, E_2|\neg H)}$
Left	Node 9	0.002	0.002	Node 7	0.001	0.0030
Middle	Node 33	0.002	0.002	Node 31	0.003	0.0050
Right	Node 24	0.002	0.002	Node 22	0.001	0.0030
Attack Step	**Attack Step 3**			**Attack Step 4**		
	$\mathbf{H_3}$	$\mathbf{P(E_3|\neg H_3)}$	$\mathbf{P(E_1, E_2, E_3|\neg H)}$	$\mathbf{H}$	$\mathbf{P(E_4|\neg H_4)}$	$\mathbf{P(E_1, E_2, E_3, E_4|\neg H)}$
Left	Node 4	0.004	0.007	Node 2	0.030	0.0368
Middle	Node 28	0.003	0.008	Node 17	0.040	0.0477
Right	Node 19	0.002	0.005	Node 17	0.007	0.0120

[2] B. Carrier, A Hypothesis-Based Approach to Digital Forensic Investigations, Ph.D. Thesis, Department of Computer Science, CERIAS Tech Report 2006-06, Center for Education and Research in Information Assurance and Security, Purdue University, West Lafayette, Indiana, 2006.

[3] A. Darwiche, *Modeling and Reasoning with Bayesian Networks*, Cambridge University Press, Cambridge, United Kingdom, 2009.

[4] N. Fenton, M. Neil and D. Lagnado, A general structure for legal arguments about evidence using Bayesian networks, *Cognitive Science*, vol. 37(1), pp. 61–102, 2013.

[5] M. Kwan, K. Chow, F. Law and P. Lai, Reasoning about evidence using Bayesian networks, in *Advances in Digital Forensics IV*, I. Ray and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 275–289, 2008.

[6] C. Liu, A. Singhal and D. Wijesekara, A logic-based network forensic model for evidence analysis, in *Advances in Digital Forensics XI*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 129–145, 2015.

[7] Y. Liu and H. Man, Network vulnerability assessment using Bayesian networks, *Proceedings of SPIE*, vol. 5812, pp. 61–71, 2005.

[8] MITRE, Common Vulnerabilities and Exposures, Bedford, Massachusetts (`cve.mitre.org`), 2016.

[9] B. Olshausen, Bayesian Probability Theory, Redwood Center for Theoretical Neuroscience, Helen Wills Neuroscience Institute, University of California at Berkeley, Berkeley, California, 2004.

[10] X. Ou, W. Boyer and M. McQueen, A scalable approach to attack graph generation, *Proceedings of the Thirteenth ACM Conference on Computer and Communications Security*, pp. 336–345, 2006.

[11] J. Pearl, Fusion, propagation and structuring in belief networks, *Artificial Intelligence*, vol. 29(3), pp. 241–288, 1986.

[12] F. Taroni, A. Biedermann, P. Garbolino and C. Aitken, A general approach to Bayesian networks for the interpretation of evidence, *Forensic Science International*, vol. 139(1), pp. 5–16, 2004.

[13] F. Taroni, S. Bozza, A. Biedermann, G. Garbolino and C. Aitken, *Data Analysis in Forensic Science: A Bayesian Decision Perspective*, John Wiley and Sons, Chichester, United Kingdom, 2010.

[14] C. Vlek, H. Prakken, S. Renooij and B. Verheij, Modeling crime scenarios in a Bayesian network, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law*, pp. 150–159, 2013.