A Probabilistic Framework for Security Scenarios with Dependent Actions

Barbara Kordy, Marc Pouly, Patrick Schweitzer

▶ **To cite this version:**

Barbara Kordy, Marc Pouly, Patrick Schweitzer. A Probabilistic Framework for Security Scenarios with Dependent Actions. Integrated Formal Methods, Sep 2014, Bertinoro, Italy. pp.256 - 271, 10.1007/978-3-319-10181-1_16 . hal-01093276

HAL Id: hal-01093276
https://hal.science/hal-01093276

Submitted on 10 Dec 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Probabilistic Framework for Security Scenarios with Dependent Actions*

Barbara Kordy[1,2], Marc Pouly[3], Patrick Schweitzer[1]

[1]University of Luxembourg, SnT, Luxembourg
[2]INSA/IRISA, Rennes, France
barbara.kordy,patrick.schweitzer@uni.lu
[3]Lucerne University for Applied Sciences and Arts, Switzerland
marc.pouly@hslu.ch

Abstract. This work addresses the growing need of performing meaningful probabilistic analysis of security. We propose a framework that integrates the graphical security modeling technique of attack–defense trees with probabilistic information expressed in terms of Bayesian networks. This allows us to perform probabilistic evaluation of attack–defense scenarios involving dependent actions. To improve the efficiency of our computations, we make use of inference algorithms from Bayesian networks and encoding techniques from constraint reasoning. We discuss the algebraic theory underlying our framework and point out several generalizations which are possible thanks to the use of semiring theory.

1 Introduction

Attack–defense trees [12] extend the well-known model of attack trees [26], by considering not only actions of an attacker, but also possible countermeasures of a defender. Since the augmented formalism models interactions between an attacker and a defender explicitly and is able to capture evolutionary aspects of attack–defense scenarios, it allows for a more accurate security assessment process compared to attack trees. In [16], we have proven that the analysis of attack–defense trees is computationally not more expensive than the analysis of attack trees. Furthermore, the usefulness of attack–defense trees for the analysis of real-world security problems has been validated in a large industrial case study [2]. These results show that attack–defense trees have the potential to become an efficient and practical security modeling and risk assessment tool.

Quantifying probabilistic aspects of attacks is one of the most important issues in security evaluation. Decisions concerning which defensive mechanisms should be implemented are based on the success probability of potential attacks. Furthermore, estimation of probability is necessary in order to evaluate risk related measures. Hence, a fully fledged methodology for security analysis needs to contain a mature framework for probabilistic computations. Unfortunately, the standard bottom-up approach for quantitative analysis of attack

* The original publication is available at: http://www.springer.com/

tree-based formalisms [18,13] can *only* be used for computing probabilities under the assumption that *all considered actions are independent*. This is a very strong assumption which is unrealistic for real-life situations.

In this paper, we develop a *complete framework for probability computations on attack–defense trees*. Our approach combines the security methodology of attack–defense trees with the probabilistic framework of Bayesian networks. This allows us to overcome the mentioned limitation of the bottom-up approach and *perform probabilistic computations in the presence of dependent actions*. Since attack trees are formally a subclass of attack–defense trees, our framework *applies directly for the analysis of the former model*. Thus, the paper also contributes to the development of full-fledged analysis technique for attack trees which are widely accepted and commonly used by industry [15].

We give a brief overview of the attack–defense tree methodology in Section 2. After recalling basic concepts for Bayesian networks, we present our framework for dependent probability computations on attack–defense trees, in Section 3. Sections 4 and 5 are concerned with methods for improving the efficiency of the framework. We describe related work in Section 6 and conclude in Section 7.

2 Modeling of Security Scenarios

This section provides background knowledge about attack–defense trees, which is necessary to understand the framework developed in this paper. For a more detailed description of the formalism, we refer to [13] and [16].

2.1 Attack–Defense Trees

Attack–defense trees (ADTrees) allow to illustrate and quantify security scenarios that involve two opposing players: an attacker and a defender. The root of an ADTree represents the main goal of one of the players. When the root is an attack node, the tree represents how to attack a considered system. Conversely, when the root is a defense node, the tree is concerned with defending the system. In ADTrees, both types of nodes, attacks and defenses, can be conjunctively or disjunctively refined. A goal represented by a conjunctively refined node is reached when *all the subgoals* depicted by its child nodes are reached. A goal represented by a disjunctively refined node is reached when *at least one of the subgoals* depicted by its child nodes is reached. The refinement operation is applied until *basic actions* are obtained. Actions are considered to be basic if they can be easily understood and quantified. Basic actions are represented by the nodes which do not have any children of the same type. Each node of an ADTree can also have one child of the opposite type. Children of the opposite type represent countermeasures. These countermeasures can be refined and countered again. In ADTrees, attack nodes are modeled by circles, defense nodes by rectangles. A conjunctive refinement is depicted with an arc. Countermeasures are connected to the actions they counteract by a dotted line.

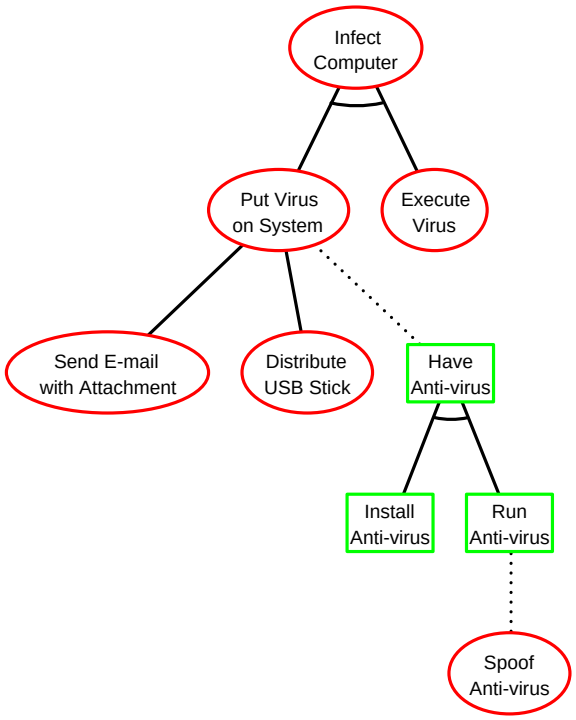

Fig. 1. ADTree for infecting a computer.

Example 1. Consider a scenario in which an attacker wants to infect a computer with a virus. In order to do this, the attacker needs to ensure that the virus file is accessible from the targeted computer and that it is executed. There are two possibilities to make the file accessible: an attacker can send the virus in an e-mail attachment or distribute an infected USB stick to the computer user. The computer user, on his part, can protect himself against a virus with an anti-virus program. For the anti-virus to be effective, it needs to be installed and it needs to be running. A resourceful attacker, in turn, could attack the anti-virus by using a fake version of an anti-virus, that disables the real anti-virus from running and only pretends that it is running. Fig. 1 depicts the described attack–defense scenario using an ADTree. In this tree, the basic actions are:

For the attacker
SE – "Send E-mail with Attachment"
DU – "Distribute USB Stick"
SA – "Spoof Anti-virus"
EV – "Execute Virus"

For the defender
IA – "Install Anti-virus"
RA – "Run Anti-virus"

The attack–defense scenario described above is used as the running example in this paper. Its main role is to illustrate how the introduced methodology

works. We purposely keep the example simple (and incomplete) in order not to overwhelm the reader with too complex models.

Remark 1. Since the root of the ADTree in Fig. 1 represents an attack goal, the paper is concerned with the probability of attacking a system. In the case of an ADTree having a defensive root node, we would talk about the probability of defending a system.

2.2 The Propositional Semantics for ADTrees

In order to provide a broad spectrum of analysis methods, several formal semantics for ADTees have been defined in [13]. In this paper, we employ the *propositional semantics* which makes use of Boolean functions.

By r, we denote a countable set of propositional variables. A *configuration* with finite domain $u \subseteq r$ is a function $\mathbf{x}\colon u \to \{0,1\}$ that associates a value $\mathbf{x}(X) \in \{0,1\}$ with every variable $X \in u$. Thus, a configuration $\mathbf{x} \in \{0,1\}^u$ represents an assignment of Boolean values to the variables in u.

Definition 1. *A Boolean function f with domain u is a function $f\colon \{0,1\}^u \to \{0,1\}$ that assigns a value $f(\mathbf{x}) \in \{0,1\}$ to each configuration $\mathbf{x} \in \{0,1\}^u$.*

Given a configuration $\mathbf{x}$ with domain $u \subseteq r$, we denote by $\mathbf{x}^{\downarrow w}$ the projection of $\mathbf{x}$ to a subset $w \subseteq u$. Let f and g be two Boolean functions with domains u and w, respectively. The *disjunction* $(f \vee g)$ and the *conjunction* $(f \wedge g)$ of f and g are Boolean functions with domain $u \cup w$, defined for every $\mathbf{x} \in \{0,1\}^{u \cup w}$ by:

$$(f \vee g)(\mathbf{x}) = \max\{f(\mathbf{x}^{\downarrow u}), g(\mathbf{x}^{\downarrow w})\}, \qquad (f \wedge g)(\mathbf{x}) = f(\mathbf{x}^{\downarrow u}) \times g(\mathbf{x}^{\downarrow w}).$$

The *negation* of f (denoted by $\neg f$) is a Boolean function with domain u, defined for every $\mathbf{x} \in \{0,1\}^u$ by: $(\neg f)(\mathbf{x}) = 1 - f(\mathbf{x})$.

Now, we explain how the propositional semantics associates ADTrees with Boolean functions. Let $\mathbb{B}$ denote the set of all basic actions. First, for every $B \in \mathbb{B}$, a propositional variable $X_B \in r$ is constructed. We assume that for $B, B' \in \mathbb{B}$, $B \neq B' \implies X_B \neq X_{B'}$. Next, a Boolean function f_t is associated with every ADTree t, as follows.

- If $t = B \in \mathbb{B}$, then $f_B\colon \{0,1\}^{\{X_B\}} \to \{0,1\}$ is defined as $f_B(X_B) = X_B$. In other words, the Boolean function associated with B is an identity function. Thus, we often abuse notation and use X_B instead of f_B.
- If t is disjunctively refined into $t_1, \ldots t_k$, then[1] $f_t = \bigvee_{i=1}^{k} f_{t_i}$,
- If t is conjunctively refined into $t_1, \ldots t_k$, then $f_t = \bigwedge_{i=1}^{k} f_{t_i}$,

[1] Here, $\bigwedge$ and $\bigvee$ stand for extensions of conjunction and disjunction of two Boolean functions to any finite number of Boolean functions. They are well-defined by associativity of $\times$ and max.

– If t is countered, then $f_t = f_{t_1} \wedge \neg f_{t_2}$, where t_1 corresponds to the refining subtree and t_2 represents the countering subtree.

Example 2. Applying the introduced recursive construction results in the following Boolean function for the ADTree t from Figure 1:

$$f_t = \Big((X_{\mathrm{SE}} \vee X_{\mathrm{DU}}) \wedge \neg(X_{\mathrm{IA}} \wedge (X_{\mathrm{RA}} \wedge \neg X_{\mathrm{SA}}))\Big) \wedge X_{\mathrm{EV}}. \tag{1}$$

Given an ADTree t, we denote by var_t the domain of the Boolean function f_t. In other words, var_t is the set of propositional variables corresponding to the basic actions involved in t. A configuration $\mathbf{x} \in \{0,1\}^{\mathrm{var}_t}$ represents which actions succeed (the corresponding variables are set to 1) and which do not (the corresponding variables are set to 0). Following our terminology convention from Remark 1, if $f_t(\mathbf{x}) = 1$, then we say that $\mathbf{x}$ is an *attack with respect to* t.

3 Probabilistic Evaluation of ADTrees

The most often used computational procedure for quantitative assessment of ADTrees relies on a bottom-up procedure [26,18,13,14]. In this approach, values are assigned to the basic actions and the bottom-up algorithm is used to determine the values of the remaining nodes as a function of the values of their children. The computation stops when the value for the root node has been found. Since the value of a node only depends on the *values* of its children, and *not on their meaning*, the bottom-up procedure cannot take dependencies between actions into account. Thus, this technique implicitly assumes that all actions of an ADTree are independent. In the case of the probability parameter, such an assumption is unrealistic. For instance, the probability that the defender runs an anti-virus program depends on whether the anti-virus is installed or not.

In order to compute the probability of attacking a system, while taking dependencies between involved actions into account, we propose a framework which combines attack–defense trees with Bayesian networks.

3.1 Bayesian Network Associated with an ADTree

A *Bayesian network* [20] is a graphical representation of a *joint probability distribution* over a finite set of variables with finite domains. The network itself is a directed, acyclic graph that reflects the conditional interdependencies between the variables associated with the nodes of the network. A directed edge from the node associated with variable X_1 to the node associated with variable X_2 means that X_2 stochastically depends on X_1. Each node contains a *conditional probability table* that quantifies the influence between the variables. The joint probability distribution p of a Bayesian network over $\{X_1, \ldots, X_n\}$ is given by

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i | \operatorname{par}(X_i)), \tag{2}$$

where, $\mathrm{par}(X_i)$ denotes the set of nodes that have an outgoing edge that points into X_i. If the set $\mathrm{par}(X_i)$ is empty, the conditional probability becomes an ordinary probability distribution. Nodes (or subgraphs) of the Bayesian network that are unconnected represent stochastically independent (sets of) variables.

Our goal is to create a Bayesian network depicting stochastic dependencies between the actions involved in a security scenario given as an ADTree. In the ADTree methodology, refined nodes do not contain any additional information, other than how their children are connected (conjunctively or disjunctively). This means that refined nodes *do not* depict any additional actions. This is why, when constructing a Bayesian network for an ADTree, we take only basic actions into account.

A Bayesian network associated with an ADTree t, denoted by BN_t, is a Bayesian network over the set of propositional variables var_t, such that there exists a directed edge from X_A to X_B if and only if action B stochastically depends on action A. Bayesian network BN_t complements ADTree t with additional information which is not contained in t. The structure of the Bayesian network BN_t is usually constructed manually. This process can however be supported by numerous existing approaches for constructing Bayesian networks [8].

Example 3. A Bayesian network BN_t associated with our running ADTree t is shown in Fig. 2. The joint probability distribution for BN_t is

$$p(X_{\mathrm{EV}}, X_{\mathrm{SE}}, X_{\mathrm{DU}}, X_{\mathrm{SA}}, X_{\mathrm{RA}}, X_{\mathrm{IA}}) = p(X_{\mathrm{RA}}|X_{\mathrm{IA}}) \times p(X_{\mathrm{IA}}) \times \\ p(X_{\mathrm{EV}}|X_{\mathrm{SE}}, X_{\mathrm{DU}}) \times p(X_{\mathrm{SE}}|X_{\mathrm{SA}}) \times p(X_{\mathrm{DU}}|X_{\mathrm{SA}}) \times p(X_{\mathrm{SA}}). \tag{3}$$

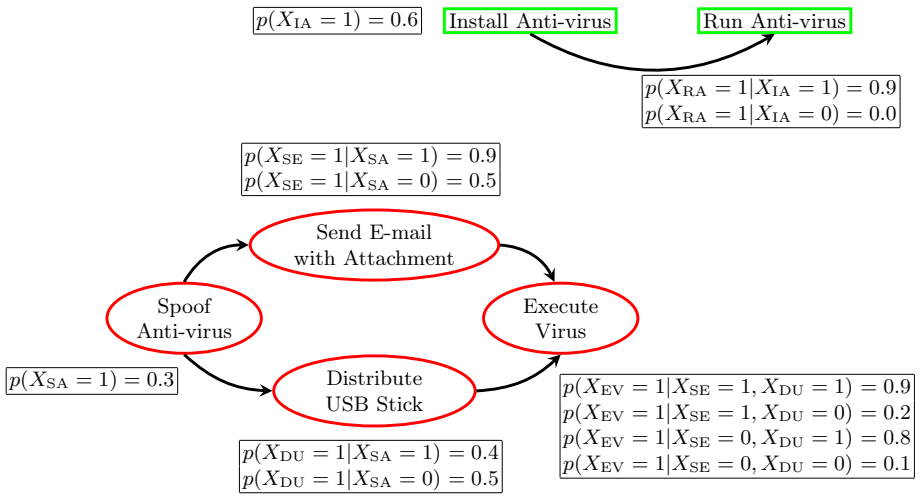

Fig. 2. Bayesian network BN_t associated with ADTree t from Figure 1.

The conditional probability tables used in Figure 2 have been constructed on an intuitive basis. The accuracy of the input values, as well as the actual methods for their estimation are a research topic in itself and are outside the scope of this submission. In the rest of the paper, we assume that the conditional probability tables have been constructed and are available.

3.2 Probabilistic Computations in the Presence of Dependencies

We now present our framework for probability computations on an ADTree t, taking the dependencies between the involved actions into account. Our computation makes use of the Boolean function f_t and the Bayesian network BN_t.

Given configuration $\mathbf{x} \in \{0,1\}^{\mathrm{var}_t}$, we define

$$\psi_t(\mathbf{x}) = f_t(\mathbf{x}) \times p(\mathbf{x}), \tag{4}$$

where p is the joint probability distribution of BN_t. If $\mathbf{x}$ is an attack with respect to t, then $f_t(\mathbf{x}) = 1$ and $\psi_t(\mathbf{x})$ returns the probability value for $\mathbf{x}$ from the Bayesian network, representing the success probability of attack $\mathbf{x}$. If $\mathbf{x}$ is not an attack with respect to t, then $f_t(\mathbf{x}) = 0$ and thus $\psi_t(\mathbf{x}) = 0$.

Example 4. Consider the situation where the attacker installs a virus file on the system by sending an e-mail with attachment ($X_{\mathrm{SE}} = 1$ and $X_{\mathrm{DU}} = 0$), executes the virus file ($X_{\mathrm{EV}} = 1$), but does not use a fake anti-virus program ($X_{\mathrm{SA}} = 0$). The defender, in turn, installs a real anti-virus ($X_{\mathrm{IA}} = 1$) which however is not running ($X_{\mathrm{RA}} = 0$). The corresponding configuration

$$\mathbf{x} = (X_{\mathrm{EV}} = 1, X_{\mathrm{SE}} = 1, X_{\mathrm{DU}} = 0, X_{\mathrm{SA}} = 0, X_{\mathrm{RA}} = 0, X_{\mathrm{IA}} = 1)$$

is an attack, because

$$f_t(\mathbf{x}) \overset{(1)}{=} \Big(\big((X_{\mathrm{SE}} \vee X_{\mathrm{DU}}) \wedge \neg(X_{\mathrm{IA}} \wedge (X_{\mathrm{RA}} \wedge \neg X_{\mathrm{SA}}))\big) \wedge X_{\mathrm{EV}}\Big)(\mathbf{x}) = 1.$$

By instantiating formula (3) with values from Fig. 2, we obtain that this attack will be successfully executed with the probability

$$\begin{aligned}\psi_t(\mathbf{x}) = f_t(\mathbf{x}) \times p(\mathbf{x}) &= p(X_{\mathrm{EV}} = 1 | X_{\mathrm{SE}} = 1, X_{\mathrm{DU}} = 0) \times p(X_{\mathrm{SE}} = 1 | X_{\mathrm{SA}} = 0) \\ &\times p(X_{\mathrm{DU}} = 0 | X_{\mathrm{SA}} = 0) \times p(X_{\mathrm{SA}} = 0) \times p(X_{\mathrm{RA}} = 0 | X_{\mathrm{IA}} = 1) \times p(X_{\mathrm{IA}} = 1) \\ &= 0.2 \times 0.5 \times (1 - 0.5) \times (1 - 0.3) \times (1 - 0.9) \times 0.6 = 0.0021.\end{aligned}$$

Next, assume we are not interested in calculating the probability of successfully executing a specific set of basic actions, but more generally in the success probability of attacking a system according to the scenario represented with ADTree t. This corresponds to the sum of the probabilities of all possible attacks with respect to t. We thus have

$$P(t) = \sum_{\mathbf{x} \in \{0,1\}^{\mathrm{var}_t}} \psi_t(\mathbf{x}) \overset{(4)}{=} \sum_{\mathbf{x} \in \{0,1\}^{\mathrm{var}_t}} f_t(\mathbf{x}) \times p(\mathbf{x}). \tag{5}$$

We refer to the value $P(t)$ as *the probability related to ADTree t*. Finally, the success probability of the most probable attack with respect to t is computed as

$$P_{\max}(t) = \max_{\mathbf{x} \in \{0,1\}^{\mathrm{var}_t}} \psi_t(\mathbf{x}) \stackrel{(4)}{=} \max_{\mathbf{x} \in \{0,1\}^{\mathrm{var}_t}} f_t(\mathbf{x}) \times p(\mathbf{x}). \tag{6}$$

4 ADTrees as Constraint Systems

We know that the number of possible configurations is exponential with respect to the number of basic actions. Thus, for large systems, the brute force computation of $P(t)$ and $P_{\max}(t)$, as suggested by formulæ (5) and (6), is no longer possible. We now present methods allowing us to represent $P(t)$ and $P_{\max}(t)$ in a factorized form, in order to increase the efficiency of their computations.

4.1 Indicator Functions for ADTrees

We employ an encoding technique from constraint reasoning and construct a factorized *indicator function* ϕ_t for the Boolean function f_t. Indicator ϕ_t maps to 1 if and only if its arguments represent a valid assignment with respect to f_t. The construction of the global indicator ϕ_t relies on *local indicators* that make use of *inner variables* and are defined as follows.

1. If $f_t = \bigvee_{i=1}^{k} f_{t_i}$, then the propositional variables $Y, Y_1, \ldots, Y_k$ are associated with $f_t, f_{t_1}, \ldots, f_{t_k}$, respectively, and the local indicator function for f_t is defined as: $\phi(Y, Y_1, \ldots, Y_k) = 1$ if $Y = \max\{Y_1, \ldots, Y_k\}$ and 0 otherwise.
2. If $f_t = \bigwedge_{i=1}^{k} f_{t_i}$, then the propositional variables $Y, Y_1, \ldots, Y_k$ are associated with $f_t, f_{t_1}, \ldots, f_{t_k}$, respectively, and the local indicator function for f_t is defined as: $\phi(Y, Y_1, \ldots, Y_k) = 1$ if $Y = Y_1 \times \ldots \times Y_k$ and 0 otherwise.
3. If $f_t = f_{t_1} \wedge \neg f_{t_2}$, then the propositional variables Y, Y_1 and Y_2 are associated with f_t, f_{t_1} and f_{t_2}, respectively, and the local indicator function for f_t is defined as: $\phi(Y, Y_1, Y_2) = 1$ if $Y = Y_1 \times (1 - Y_2)$ and 0 otherwise.

Example 5. A step-wise construction of the local indicators for the Boolean function given in Example 2 proceeds as follows:

$$f_t = \Big(\underbrace{(X_{\mathrm{SE}} \vee X_{\mathrm{DU}})}_{Y_1} \wedge \neg \underbrace{(X_{\mathrm{IA}} \wedge \underbrace{(X_{\mathrm{RA}} \wedge \neg X_{\mathrm{SA}})}_{Y_2})}_{Y_3} \Big)}_{Y_4} \wedge X_{\mathrm{EV}}}_{Y_t}$$

In this case, the inner variables are Y_1, Y_2, Y_3, Y_4, Y_t and the local indicators are

$$\begin{aligned}
&\phi_1(Y_1, X_{\mathrm{SE}}, X_{\mathrm{DU}}) = 1 \quad \text{exactly if} \quad Y_1 = \max(X_{\mathrm{SE}}, X_{\mathrm{DU}}),\\
&\phi_2(Y_2, X_{\mathrm{RA}}, X_{\mathrm{SA}}) = 1 \quad \text{exactly if} \quad Y_2 = X_{\mathrm{RA}} \times (1 - X_{\mathrm{SA}}),\\
&\phi_3(Y_3, X_{\mathrm{IA}}, Y_2) = 1 \quad \text{exactly if} \quad Y_3 = X_{\mathrm{IA}} \times Y_2,\\
&\phi_4(Y_4, Y_1, Y_3) = 1 \quad \text{exactly if} \quad Y_4 = Y_1 \times (1 - Y_3),\\
&\phi_5(Y_t, Y_4, X_{\mathrm{EV}}) = 1 \quad \text{exactly if} \quad Y_t = Y_4 \times X_{\mathrm{EV}}.
\end{aligned}$$

Let t be an ADTree. Having constructed all local indicators, we can build the *global indicator function* ϕ_t. The domain of ϕ_t contains all variables used by the local indicators, i.e., the inner variables and the variables corresponding to basic actions of t. An assignment over all variables is valid if and only if each local assignment is valid. Hence, we may compute the global indicator function for f_t by multiplying all its local indicators. For the function from Example 5, we get:

$$\begin{aligned}
&\phi_t(Y_1, Y_2, Y_3, Y_4, Y_t, X_{\mathrm{SE}}, X_{\mathrm{DU}}, X_{\mathrm{RA}}, X_{\mathrm{SA}}, X_{\mathrm{IA}}, X_{\mathrm{EV}}) = \phi_1(Y_1, X_{\mathrm{SE}}, X_{\mathrm{DU}}) \times\\
&\phi_2(Y_2, X_{\mathrm{RA}}, X_{\mathrm{SA}}) \times \phi_3(Y_3, X_{\mathrm{IA}}, Y_2) \times \phi_4(Y_4, Y_1, Y_3) \times \phi_5(Y_t, Y_4, X_{\mathrm{EV}}). \qquad (7)
\end{aligned}$$

In this paper, we use the following notation: given the global indicator function ϕ_t for t, we denote by Y_t the inner variable corresponding to the entire tree t. The set of all inner variables of ϕ_t is denoted by d_t.

Consider an indicator function $\phi(Y, Y_1, \ldots, Y_k)$. Let $\mathbf{z}$ be an assignment of values to the variables $Y_1, \ldots, Y_k$. There is, by definition, exactly one value $y \in \{0, 1\}$ for Y, such that $\phi(y, \mathbf{z}) = 1$. Since the global indicator function is obtained by multiplication, we may directly conclude the following theorem.

Theorem 1. *Consider an ADTree t with basic actions $B_1, \ldots, B_n$ and its global indicator function ϕ_t. Given a specific assignment $\mathbf{x}$ of values to the variables $X_{B_1}, \ldots, X_{B_n}$ corresponding to basic actions, there is exactly one assignment $\mathbf{y}$ to the inner variables from d_t, such that $\phi_t(\mathbf{y}, \mathbf{x}) = 1$.*

An immediate consequence of Theorem 1 is that, for a specific assignment $\mathbf{x} \in \{0, 1\}^{\mathrm{var}_t}$ of values to the variables associated with basic actions, we have

$$\max_{\mathbf{y} \in \{0,1\}^{d_t}} \phi_t(\mathbf{y}, \mathbf{x}) = \sum_{\mathbf{y} \in \{0,1\}^{d_t}} \phi_t(\mathbf{y}, \mathbf{x}) = 1. \qquad (8)$$

When performing probabilistic computations as specified by formulæ (4), (5) and (6), we are only interested in those combinations of basic actions that correspond to attacks. Thus, when reasoning in terms of global indicator functions, we need to restrict our considerations to those configurations where variable Y_t equals 1. This can be achieved by *conditioning* ϕ_t on $Y_t = 1$, which means that we invalidate all configurations with $Y_t = 0$. We therefore define a *filter* F_t for the ADTree t that satisfies $F_t(Y_t) = 1$ if and only if $Y_t = 1$. In other words, $F_t \colon \{0, 1\}^{\{Y_t\}} \to \{0, 1\}$ is the identity function for variable Y_t. Multiplying filter F_t and global indicator ϕ_t results in a function, denoted by $\phi_{t|Y_t=1}$, which maps

to 1 if and only if the assignment of values to the variables is valid with respect to f_t *and* it represents an attack according to t. We thus have,

$$\forall \mathbf{z} \in \{0,1\}^{\mathrm{var}_t \cup d_t} \colon \phi_{t|Y_t=1}(\mathbf{z}) = F_t(\mathbf{z}^{\downarrow\{Y_t\}}) \times \phi_t(\mathbf{z}). \tag{9}$$

Let t be an ADTree, ϕ_t be its global indicator function and F_t be the filter for t. Assume furthermore that we are given a specific configuration $\mathbf{x} \in \{0,1\}^{\mathrm{var}_t}$. Configuration $\mathbf{x}$ is an attack with respect to t if and only if, there exists $\mathbf{y} \in \{0,1\}^{d_t}$, such that $\phi_{t|Y_t=1}(\mathbf{y},\mathbf{x})$ maps to 1. Using formula (8), we obtain

$$\max_{\mathbf{y}\in\{0,1\}^{d_t}} \phi_{t|Y_t=1}(\mathbf{y},\mathbf{x}) = \sum_{\mathbf{y}\in\{0,1\}^{d_t}} \phi_{t|Y_t=1}(\mathbf{y},\mathbf{x}) = f_t(\mathbf{x}) = \begin{cases} 1 \text{ if } \mathbf{x} \text{ is an attack} \\ 0 \text{ otherwise.} \end{cases} \tag{10}$$

4.2 Indicators for Probability Computation

Making use of the property described by (10), the procedure for the probabilistic computations developed in Section 3.2 can be redefined as follows. Let t be an ADTree and $\mathbf{x} \in \{0,1\}^{\mathrm{var}_t}$ be an assignment of Boolean values to the variables corresponding to the basic actions of t. If $\mathbf{x}$ is an attack with respect to t, then its probability is computed as

$$\psi_t(\mathbf{x}) \overset{(4)}{=} f_t(\mathbf{x}) \times p(\mathbf{x}) \overset{(10),\ \text{distrib.}}{=} \sum_{\mathbf{y}\in\{0,1\}^{d_t}} \Big(\phi_{t|Y_t=1}(\mathbf{y},\mathbf{x}) \times p(\mathbf{x})\Big) \tag{11}$$

$$\overset{(10),\ \text{distrib.}}{=} \max_{\mathbf{y}\in\{0,1\}^{d_t}} \Big(\phi_{t|Y_t=1}(\mathbf{y},\mathbf{x}) \times p(\mathbf{x})\Big). \tag{12}$$

The probability related to ADTree t is expressed as

$$P(t) \overset{(5)}{=} \sum_{\mathbf{x}\in\{0,1\}^{\mathrm{var}_t}} \psi_t(\mathbf{x}) \overset{(11)}{=} \sum_{\mathbf{x}\in\{0,1\}^{\mathrm{var}_t}} \sum_{\mathbf{y}\in\{0,1\}^{d_t}} \Big(\phi_{t|Y_t=1}(\mathbf{y},\mathbf{x}) \times p(\mathbf{x})\Big)$$

$$= \sum_{\mathbf{z}\in\{0,1\}^{\mathrm{var}_t \cup d_t}} \Big(\phi_{t|Y_t=1}(\mathbf{z}) \times p(\mathbf{z}^{\downarrow \mathrm{var}_t})\Big). \tag{13}$$

Similarly, the probability of the most probable attack with respect to t is

$$P_{\max}(t) \overset{(6)}{=} \max_{\mathbf{x}\in\{0,1\}^{\mathrm{var}_t}} \psi_t(\mathbf{x}) \overset{(12)}{=} \max_{\mathbf{z}\in\{0,1\}^{\mathrm{var}_t \cup d_t}} \Big(\phi_{t|Y_t=1}(\mathbf{z}) \times p(\mathbf{z}^{\downarrow \mathrm{var}_t})\Big). \tag{14}$$

Example 6. Let $u = \{Y_1, Y_2, Y_3, Y_4, Y_t, X_{\mathrm{SE}}, X_{\mathrm{DU}}, X_{\mathrm{RA}}, X_{\mathrm{SA}}, X_{\mathrm{IA}}, X_{\mathrm{EV}}\}$. The factorized form for the probability related the ADTree from Figure 1 is

$$P(t) \overset{(13),(9),(7),(3)}{=} \tag{15}$$

$$\sum_{\mathbf{z}\in\{0,1\}^u} \Big(F_t(\mathbf{z}^{\downarrow\{Y_t\}}) \times \phi_1(\mathbf{z}^{\downarrow\{Y_1,X_{\mathrm{SE}},X_{\mathrm{DU}}\}}) \times \phi_2(\mathbf{z}^{\downarrow\{Y_2,X_{\mathrm{RA}},X_{\mathrm{SA}}\}}) \times \phi_3(\mathbf{z}^{\downarrow\{Y_3,X_{\mathrm{IA}},Y_2\}})$$

$$\times \phi_4(\mathbf{z}^{\downarrow\{Y_4,Y_1,Y_3\}}) \times \phi_5(\mathbf{z}^{\downarrow\{Y_t,Y_4,X_{\mathrm{EV}}\}}) \times p(\mathbf{z}^{\downarrow\{X_{\mathrm{EV}},X_{\mathrm{SE}},X_{\mathrm{DU}}\}}) \times p(\mathbf{z}^{\downarrow\{X_{\mathrm{SE}},X_{\mathrm{SA}}\}})$$

$$\times p(\mathbf{z}^{\downarrow\{X_{\mathrm{DU}},X_{\mathrm{SA}}\}}) \times p(\mathbf{z}^{\downarrow\{X_{\mathrm{SA}}\}}) \times p(\mathbf{z}^{\downarrow\{X_{\mathrm{RA}},X_{\mathrm{IA}}\}}) \times p(\mathbf{z}^{\downarrow\{X_{\mathrm{IA}}\}}) \Big).$$

5 Efficiency Considerations

The factorization of the global indicator function, in terms of local indicators which are bounded in size, introduces additional structure that can be exploited for so-called *local computation* [22]. In this section, we show how the *fusion* algorithm allows us to improve the efficiency of evaluating formulas (13) and (14).

5.1 Semiring Valuations

An algebraic structure $\langle A, \oplus, \odot \rangle$ with binary operations $\oplus$ and $\odot$ is called *commutative semiring* if both operations are associative, commutative, and if $\odot$ distributes over $\oplus$, i.e., if for $a, b, c \in A$, we have $a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$ and $(a \oplus b) \odot c = (a \odot c) \oplus (b \odot c)$. Typical examples of commutative semirings include the Boolean semiring $\langle \{0,1\}, \max, \times \rangle$, the tropical semiring $\langle \mathbb{N}, \min, + \rangle$, the product t-norm semiring $\langle [0,1], \max, \times \rangle$ and the arithmetic semiring $\langle \mathbb{R}, +, \times \rangle$.

Let $u \subseteq r$ be a finite set of propositional variables and $\langle A, \oplus, \odot \rangle$ be a commutative semiring. A *semiring valuation* over $\langle A, \oplus, \odot \rangle$ is a function $\phi : \{0,1\}^u \to A$ associating a value from A with each configuration from $\{0,1\}^u$. We denote by $\mathrm{dom}(\phi) = u$ the domain of valuation ϕ. The *combination* of two valuations ϕ and ψ over a semiring $\langle A, \oplus, \odot \rangle$ is defined, for all $\mathbf{x} \in \{0,1\}^{\mathrm{dom}(\phi) \cup \mathrm{dom}(\psi)}$, as:

$$(\phi \otimes \psi)(\mathbf{x}) = \phi(\mathbf{x}^{\downarrow \mathrm{dom}(\phi)}) \odot \psi(\mathbf{x}^{\downarrow \mathrm{dom}(\psi)}).$$

The *elimination* of variable $X \in \mathrm{dom}(\phi)$ is defined, for all $\mathbf{x} \in \{0,1\}^{\mathrm{dom}(\phi) \setminus \{X\}}$, as:

$$\phi^{-X}(\mathbf{x}) = \phi(\mathbf{x}, 0) \oplus \phi(\mathbf{x}, 1).$$

Due to associativity of semiring addition $\oplus$, we can eliminate variables in any order. For $\{X_1, \ldots, X_m\} \subseteq \mathrm{dom}(\phi)$, we may therefore write

$$\phi^{-\{X_1,\ldots,X_m\}} = \Big(\ldots \big((\phi^{-X_1})^{-X_2} \big) \ldots \Big)^{-X_m}.$$

Indicator functions are Boolean semiring valuations over $\langle \{0,1\}, \max, \times \rangle$. Arithmetic semiring valuations over $\langle \mathbb{R}, +, \times \rangle$ capture conditional probability tables from Bayesian networks, and product t-norm semiring valuations over $\langle [0,1], \max, \times \rangle$ compute maximum attack probabilities, as in formula (14).

It has been shown in [10] that semiring valuations over arbitrary commutative semirings always satisfy the axioms of a valuation algebra [9,22]. The computational interest in valuation algebras is stated by the *inference problem*. Given a set of (semiring) valuations $\{\phi_1, \ldots, \phi_n\}$, called *knowledgebase*, with domains $u_i = \mathrm{dom}(\phi_i)$, for $i = 1, \ldots, n$, and a set of variables $\{X_1, \ldots, X_m\} \subseteq u_1 \cup \ldots \cup u_n$, the inference problem consists of computing

$$\phi^{-\{X_1,\ldots,X_m\}} = (\phi_1 \otimes \ldots \otimes \phi_n)^{-\{X_1,\ldots,X_m\}}. \tag{16}$$

Example 7. Let $u = \{Y_1, Y_2, Y_3, Y_4, Y_t, X_{\mathrm{SE}}, X_{\mathrm{DU}}, X_{\mathrm{RA}}, X_{\mathrm{SA}}, X_{\mathrm{IA}}, X_{\mathrm{EV}}\}$. Computing the probability in Example 6 amounts to solving the inference problem

$$\Big(F_t(\mathbf{z}^{\downarrow\{Y_t\}}) \times \phi_1(\mathbf{z}^{\downarrow\{Y_1, X_{\mathrm{SE}}, X_{\mathrm{DU}}\}}) \times \ldots \times p(\mathbf{z}^{\downarrow\{X_{\mathrm{IA}}\}})\Big)^{-u} = \sum_{\mathbf{z}\in\{0,1\}^u} \Big(F_t(\mathbf{z}^{\downarrow\{Y_t\}}) \times \phi_1(\mathbf{z}^{\downarrow\{Y_1, X_{\mathrm{SE}}, X_{\mathrm{DU}}\}}) \times \ldots \times p(\mathbf{z}^{\downarrow\{X_{\mathrm{IA}}\}})\Big).$$

Here, the knowledgebase consists of all local indicator functions, filter F_t and all conditional probability tables, which instantiate arithmetic semiring valuations. Likewise, computing maximum attack probability, expressed by formula (14), amounts to solving a similar inference problem over the product t-norm semiring.

5.2 Fusion

A direct evaluation of formulas (13), (14), and more generally of (16), is in most cases not possible, due to the exponential complexity of combination of semiring valuations. However, because the computational tasks are stated with respect to a factorization of the global indicator function and the joint probability distribution, we may exploit the additional structure inside the factorization and perform calculations locally on the domain of the factors. *Fusion* [27] (or *bucket-elimination* [5]) is one of the local computation algorithms that can be applied to factorizations of arbitrary valuation algebras. Thus, we may use it for processing inference problems obtained from ADTrees.

The elimination of a single variable $X \in \mathrm{dom}(\phi) = \mathrm{dom}(\phi_1) \cup \ldots \cup \mathrm{dom}(\phi_n)$ from a set $\{\phi_1, \ldots, \phi_n\}$ of valuations can be performed as follows:

$$\mathrm{Fus}_X(\{\phi_1, \ldots, \phi_n\}) = \{\psi^{-X}\} \cup \{\phi_i \colon X \notin \mathrm{dom}(\phi_i)\}, \tag{17}$$

where $\psi = \bigotimes_{i\colon X\in\mathrm{dom}(\phi_i)} \phi_i$. This means that we only need to eliminate X from the factors that have X in the domain. As described in [9], the fusion algorithm then follows by repeated application of this operation:

$$(\phi_1 \otimes \ldots \otimes \phi_n)^{-\{X_1,\ldots,X_m\}} = \bigotimes \mathrm{Fus}_{X_m}(\ldots(\mathrm{Fus}_{X_1}(\{\phi_1, \ldots, \phi_n\}))).$$

In every step $i = 1, \ldots, m$ of the fusion algorithm, the combination in (17) creates an intermediate factor ψ_i with domain $\mathrm{dom}(\psi_i)$. Then, variable X_i is eliminated only from ψ_i in (17). We define $\lambda(i) = \mathrm{dom}(\psi_i) \setminus \{X_i\}$ called *label* and observe that $\lambda(m) = (\mathrm{dom}(\phi_1) \cup \ldots \cup \mathrm{dom}(\phi_n)) \setminus \{X_1, \ldots, X_m\}$. The domains of all intermediate results of the fusion algorithm are therefore bounded by the size of the largest label plus one. The smaller the labels are, the more efficient fusion is. We further remark that the labels depend on the chosen elimination sequence for variables $X_1, \ldots, X_m$. Finding a sequence that leads to the smallest label is NP-complete [1], however, there are good heuristics that achieve reasonable execution time [6]. In summary, the complexity of computing (16) is not necessarily exponential in the number of variables involved in the problem, but only in the size of the largest label, also called *tree width* [25], that occurs during fusion.

We have applied fusion to the inference problem from Example 7. The results show that, when fusion is used, time and space complexity of the computation of $P(t)$ for our running ADTree are bounded by 2^5. To compare, a naive, direct computation, as in (15), is bounded by 2^{11}. We have also automated the computation of $P(t)$ with the help of the open-source tool Nenok [23] which provides an extensive library of generically implemented local computation algorithms. When applying fusion, Nenok outputs the value of $P(t)$ after 0.031 sec in contrast to 3.422 sec that the application requires to compute the same value in a naive way, i.e., by using expression (15) directly.

6 Related Work

ADTrees are only one of more than 30 graphical formalisms for security assessment, which are based on attack trees. A recent survey, by Kordy et al. [15] presents a complete state of the art in the field of DAG-based approaches for modeling of attacks and defenses. It summarizes existing formalisms, compares their features and proposes their taxonomy. The reader is referred to this survey for an overview of existing methods for quantitative, and in particular probabilistic, analysis of security. In the remainder of this section, we compare our framework with the most prominent, existing models that combine AND-OR graphs with Bayesian networks.

Qin and Lee are one of the pioneers in applying Bayesian networks for security analysis [24]. They propose a conversion of regular attack trees into Bayesian networks, in order to make use of probabilistic inference techniques to evaluate the likelihood of attack goals and predict potential upcoming attacks. Edges representing disjunctive refinements in the tree are also present in the corresponding Bayesian network, because they represent cause-consequence relations between components. Contrary to our interpretation, a conjunction in attack trees is assumed to have an explicit or implicit order in which the actions have to be executed. This allows to convert conjunctions into a directed path in the Bayesian network, starting from the first child, according to the given order, and ending with the parent node. The construction from [24] implies that the Bayesian network and the attack tree contain the same set of nodes. Furthermore the Bayesian network models cause-consequence relationships that correspond to the child-parent connections in the underlying attack tree. In our case, the Bayesian network depicts *additional* dependencies that represent how different basic actions are influenced by each other.

In [7], Frigault and Wang advance a model, called *Bayesian attack graphs*. They construct a Bayesian network starting from an attack graph which depicts how multiple vulnerabilities may be combined in an attack. The resulting directed acyclic graph contains all nodes of the original attack graph. Employing the CVSS mechanism [19], the nodes are then associated with the conditional probability tables. In [21], Poolsappasit et al. revisit the framework of Bayesian attack graphs to be able to deal with asset identification, system vulnerability and connectivity analysis, as well as mitigation strategies. In addition to the con-

ditional probability tables that represent the probability with which an attack takes place, they consider edge probabilities expressing how likely a present attack succeeds. Furthermore, the authors of [21] augment Bayesian attack graphs with additional nodes and values representing defenses. This extended structure allows them to solve the multiobjective optimization problem of how to select optimal defenses. Even though this model is similar to ours, it does not cover interleaved attacks and defenses.

Yet another approach that makes use of Bayesian networks for security analysis was described by Sommestad et al. [28]. It transforms defense trees [3] (an extension of attack trees with defenses attached to leaf nodes) into extended influence diagrams [17] (an extension of Bayesian networks with conjunctive and disjunctive nodes as well as countermeasures). The relationships between the nodes are encoded in conditional probability tables assigned to each node. The authors state that with this setup, Bayesian inference can be used to derive values, however they do not provide detailed computation algorithms. Our paper specifies how the necessary computational steps could be performed.

Contrary to our design, none of the above approaches separate the logical structure (conjunctions and disjunctions) from the probabilistic structure. One advantage of our approach is that we are not transforming one model into another, but we are using them modularly. Merging the two methodologies is only implicitly done during fusion. Unlike our model, all related approaches assume a one-to-one correspondence between the nodes in the original graph and the Bayesian network. Since in our framework, the Bayesian network concerns only basic actions, its size is much smaller compared to the size of Bayesian networks used by the approaches described in this section.

7 Conclusion and Future Work

This paper proposes to combine the ADTree methodology with Bayesian networks in order to *evaluate probabilistic measures on attack–defense scenarios involving dependent actions*. The introduced approach improves upon the standard, bottom-up, computational routine for attack tree-based formalisms, which assumes that all actions involved in the model are independent. By lifting the independency assumption, we provide a pragmatic methodology for accurate probabilistic assessment of security scenarios modeled using attack trees or ADTrees.

In our framework, the Bayesian network *does not replace* the information represented by the structure of an ADTree, but complements it with *additional probabilistic dependencies* between attack steps, which cannot be depicted using AND-OR relations. Keeping the two models separated allows us to take advantage of the strengths of both formalisms. The propositional encoding of ADTrees is used to identify configurations which represent attacks. Bayesian networks together with the fusion algorithm and techniques based on semiring valuation algebras provide ways to improve the efficiency of probabilistic computations.

To support modeling and quantitative analysis of security using the ADTree methodology, a free software tool, called ADTool [11], has recently been devel-

oped. We are currently extending the functionality of ADTool by interfacing it with Nenok, so that it can handle the framework introduced in this paper. Since Nenok implements generic algorithms for efficient processing of semiring-based computations, the extended tool will support efficient, automated probabilistic analysis of real-world, possibly large-scale, attack–defense scenarios.

Employing fusion implies that time and space complexity are bounded by a structural parameter of the problem rather than by the total number of variables involved. It thus cannot be predicted in general how well fusion can cope with large problems involving many variables. It all depends on whether a small tree width (or good elimination sequence) can be found by some heuristic. Prediction of the tree width is possible for specific families of graphs [4]. It is one of our future research directions to investigate whether combination of an ADTree with a Bayesian network, both produced by human security experts, would satisfy the definition of one such family.

The algorithmic technique based on semiring valuations that we have used in this paper also works in a broader context. From an algebraic perspective, the combination of indicator functions and probabilities is possible because the Boolean semiring for indicator functions is a sub-algebra of both semirings used for expressing probabilities, i.e., the arithmetic and product t-norm semiring. Consequently, we may directly apply the same construction to other semiring valuations under the additional condition that the corresponding semiring takes the Boolean semiring as sub-algebra. The large family of t-norm semirings [22] are important candidates used in possibility and fuzzy set theory [29].

Acknowledgments. We would like to thank Sjouke Mauw, Pieter Hartel, Jan-Willem Bullée, Lorena Montoya Morales, and the anonymous reviewers for their valuable comments that helped us to improve the paper. The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement number 318003 (TREsPASS) and from the Fonds National de la Recherche Luxembourg under grants PHD-09-167 and C13/IS/5809105.

References

1. Arnborg, S., Corneil, D., Proskurowski, A.: Complexity of Finding Embeddings in a k-Tree. SIAM J. of Algebraic and Discrete Methods 8, 277–284 (1987)
2. Bagnato, A., Kordy, B., Meland, P.H., Schweitzer, P.: Attribute Decoration of Attack–Defense Trees. IJSSE 3(2), 1–35 (2012)
3. Bistarelli, S., Fioravanti, F., Peretti, P.: Defense Trees for Economic Evaluation of Security Investments. In: ARES. pp. 416–423. IEEE Computer Society (2006)
4. Bodlaender, H.L.: A Partial K-arboretum of Graphs with Bounded Treewidth. Theoretical Computer Science 209(1-2), 1–45 (1998)
5. Dechter, R.: Bucket Elimination: A Unifying Framework for Reasoning. Artif. Intell. 113, 41–85 (1999)
6. Dechter, R.: Constraint Processing. Morgan Kaufmann (2003)
7. Frigault, M., Wang, L.: Measuring Network Security Using Bayesian Network-Based Attack Graphs. In: COMPSAC. pp. 698–703 (2008)

8. van Harmelen, F., van Harmelen, F., Lifschitz, V., Porter, B.: Handbook of Knowledge Representation. Elsevier Science, San Diego, USA (2007)
9. Kohlas, J.: Information Algebras: Generic Structures for Inference. Springer (2003)
10. Kohlas, J., Wilson, N.: Semiring induced Valuation Algebras: Exact and Approximate Local Computation algorithms. Artif. Intell. 172(11), 1360–1399 (2008)
11. Kordy, B., Kordy, P., Mauw, S., Schweitzer, P.: ADTool: Security Analysis with Attack–Defense Trees. In: Joshi, K.R., Siegle, M., Stoelinga, M., D'Argenio, P.R. (eds.) QEST. LNCS, vol. 8054, pp. 173–176. Springer (2013)
12. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Foundations of Attack–Defense Trees. In: Degano, P., Etalle, S., Guttman, J.D. (eds.) FAST. LNCS, vol. 6561, pp. 80–95. Springer (2010)
13. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Attack–Defense Trees. Journal of Logic and Computation 24(1), 55–87 (2014)
14. Kordy, B., Mauw, S., Schweitzer, P.: Quantitative Questions on Attack–Defense Trees. In: ICISC. LNCS, vol. 7839, pp. 49–64. Springer (2013)
15. Kordy, B., Pièrre-Cambacédès, L., Schweitzer, P.: DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. CoRR 1303.7397 (2013), available at `http://arxiv.org/abs/1303.7397` (under submission)
16. Kordy, B., Pouly, M., Schweitzer, P.: Computational Aspects of Attack–Defense Trees. In: SIIS. LNCS, vol. 7053, pp. 103–116. Springer (2011)
17. Lagerström, R., Johnson, P., Närman, P.: Extended Influence Diagram Generation. In: Jardim-Gonçalves, R., Müller, J.P., Mertins, K., Zelm, M. (eds.) IESA. pp. 599–602. Springer (2007)
18. Mauw, S., Oostdijk, M.: Foundations of Attack Trees. In: Won, D., Kim, S. (eds.) ICISC. LNCS, vol. 3935, pp. 186–198. Springer (2005)
19. Mell, P., Scarfone, K., Romanosky, S.: A Complete Guide to the Common Vulnerability Scoring System Version 2.0. `http://www.first.org/cvss/cvss-guide.html` (2007)
20. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988)
21. Poolsappasit, N., Dewri, R., Ray, I.: Dynamic Security Risk Management Using Bayesian Attack Graphs. IEEE Trans. Dep. Sec. Comp. 9(1), 61–74 (2012)
22. Pouly, M., Kohlas, J.: Generic Inference - A Unifying Theory for Automated Reasoning. John Wiley & Sons, Inc. (2011)
23. Pouly, M.: NENOK - A Software Architecture for Generic Inference. Int. J. on Artif. Intel. Tools 19, 65–99 (2010)
24. Qin, X., Lee, W.: Attack plan recognition and prediction using causal networks. In: ACSAC. pp. 370–379 (2004)
25. Robertson, N., Seymour, P.: Graph Minors I: Excluding a Forest. J. Comb. Theory, Ser. B 35(1), 39–61 (1983)
26. Schneier, B.: Attack Trees. Dr. Dobb's Journal of Software Tools 24(12), 21–29 (1999)
27. Shenoy, P.: Valuation-Based Systems: A Framework for Managing Uncertainty in Expert Systems. In: Zadeh, L., Kacprzyk, J. (eds.) Fuzzy Logic for the Management of Uncertainty, pp. 83–104. John Wiley & Sons, Inc. (1992)
28. Sommestad, T., Ekstedt, M., Nordström, L.: Modeling security of power communication systems using defense graphs and influence diagrams. IEEE Trans. Pow Del. 24(4), 1801–1808 (2009)
29. Zadeh, L.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems 1, 3–28 (1978)