Determining the Probability of Smart Grid Attacks by Combining Attack Tree and Attack Graph Analysis*

Kristian Beckers[1], Maritta Heisel[1], Leanid Krautsevich[2],
Fabio Martinelli[2], Rene Meis[1], and Artsiom Yautsiukhin[2]

[1] paluno – The Ruhr Institute for Software Technology – University of Duisburg-Essen
{firstname.lastname}@paluno.uni-due.de
[2] Istituto di Informatica e Telematica – Consiglio Nazionale delle Ricerche
Via G. Moruzzi 1, Pisa 56124, Italy
{firstname.lastname}@iit.cnr.it

Abstract. Smart grid is an intelligent energy distribution system consisting of multiple information and communication technologies (ICT). One of the challenges for such complex and heterogeneous system as smart grid is to unite security analysis on a high level of abstraction and concrete behavioral attack patterns that exploit low-level vulnerabilities. We provide a structured method that combines the Si* language, which can express attacker motivations as a goal hierarchy, and vulnerability specific attack graphs, which shows every step available for an attacker. We derive system specific information from the low-level representation of the system for a high-level probabilistic analysis.

Keywords: smart grid, threat analysis, attack graphs, attack trees, Si* model.

1 Introduction

The smart grid is an electricity grid (or network) that intelligently manages the behavior and actions of its participants. The benefit of this network is envisioned to be a more economic, sustainable, and secure supply of energy. The part of the smart grid called smart metering system meters the consumption or production of energy and forward the data to external entities. This data is further used for billing and steering the energy production. In this paper, we focus on the analysis of security issues in the smart grid to help developers and administrators to protect their systems against possible attackers.

Running Example In our running example, we focus on attacks on an energy supplier called *Huntsville Consortium*. The Huntsville Consortium requires information about the prosumers energy consumption for billing purposes. The Huntsville Consortium acquires this information via an ICT system in the smart grid (see Sect. 4 for details). The energy supplier is interested in a protection against a certain group of attackers, e.g., network attackers, which are considered the most relevant for the system.

* This research was partially supported by the EU FP7 Network of Excellence on Engineering Secure Future Internet Software Services and Systems (NESSoS, no 256980) and SESAMO, no 295354 projects.

There are high-level and low-level views on attacks execution. The analysis of attacks often starts with determination of high-level attack scenarios [4, 7], e.g., determination of goals of attackers. There is the Si* framework for high-level describing and analyzing the system from the viewpoint of actors. The Si* framework provides the ability to model the actors, their relations and goals, as well as the technology needed to fulfill the goals. We use the Si* framework [11] for modeling and decomposing attackers goals. The result of decomposing goals is a usual attack tree [12, 18].

In addition to attack tree modeling notation, the Si* framework provides the means to model a high-level abstraction of a system, which is related to goals in the attack tree. Moreover, the Si* framework has been created to analyze secure socio-technical systems by making their capabilities explicit. In particular, the framework allows to express delegation and transfer of capabilities and the expected behavior of actors (in our example the energy supplier delegates tasks and capabilities to subcontractors). Thus, Si* provides the means to describe the context of the attacker goals in addition to usual description and decomposition of goals. There are several examples of application of Si* to the analysis of security in general [1, 10, 13] and smart grid in particular [2].

At the low level, an attacker realizes his/her attack plan by exploiting vulnerabilities existing in a system. High-level attack trees usually do not considered these vulnerabilities. Attack graphs are a low-level description of a system that organizes vulnerabilities in such a way that denotes the propagation of gained privileges during an attack [6, 14].

Attack graphs are constructed by security personnel, while a real attacker does not have this complete picture of the existing vulnerabilities. In contrast, the attacker has only a rough plan (attack tree) and focuses only on the relevant parts of the system for her plan. Therefore, our contribution proposes to limit the analysis of the attack graph to the plan of the attacker represented in the attack tree. To achieve this, we propose to map Si* attack trees to attack graphs and explicitly considers how the attacker may satisfy his/her high-level goals by exploiting low-level vulnerabilities.

To sum up, we aim at an approach that connects two views on attacks execution in the smart grid. We provide a mapping between a high-level view on attacks expressed in the Si* language as an attack tree composed of sub-goals and a low-level view expressed as an attack graph consisting of vulnerabilities. This mapping helps us to aggregate the information available at a low-level and to analyze attacker plans. We use this aggregated information for a quantitative analysis, which allows the evaluation of probabilities of attackers goals to be achieved analyzing vulnerabilities that should be exploited while pursuing the goal. We suppose that such analysis allows more efficient investment distribution to achieve a more secure smart grid system.

The paper is organized as follows. Section 2 reviews the related work. Section 3 describes our structured method and Sect. 4 exemplifies the application of our method to our running example. Section 5 presents conclusions and the future work.

2 Related Work

Several methods were proposed for modeling an attacker during system engineering in the past. The aim of such methods was to understand the socio-technical view of the system and find the ways how an attacker may affect the work of the system.

Liu et al. [10] presented an i*-based framework for agent-oriented software engineering and requirements analysis. The framework is mainly focused on insider attackers rather than on hackers threatening a system from the outside. Mouratidis et al. [13] proposed a similar approach using scenarios to analyse a reaction of information systems on potential security attacks during the development process. The authors consider a simple single attacker and model the attacker as an i* actor who is trying to achieve her goal by completing sub-goals. Lamsweerde [19] considered attack scenarios during the development of services on the application layer. He considered obstacles as anti-goals for the system and, thus, as goals for the attacker. Asnar and Massacci [1] added risks related to the goals of a system to the Si* framework. The work has no explicit focus on the source of the risk (e.g., on attackers).

In this work we focus on goals of the attacker and their decomposition. In fact, this goal decomposition is similar to the attack tree technique, introduced by B. Schneier [18]. Since then, the attack trees have been acknowledged as a useful technique for analysis of attacker behaviour [4, 12]. The usage of Si* from the attacker's point of view is similar to the attack tree technique, but allows modelling additional aspects of possible attacks (e.g., attacker motivation).

An attack graph is a technique for security modelling and analysis of a system which specifies states, related to the privileges the attacker may have, and transitions between them. There are two types of attack graphs in the literature. The first type denotes every state as a set of all privileges the attacker possesses at a certain stage of an attack [6]. The graph in this case is acyclic if we assume that an attacker cannot loose her privileges. The advantage of this model is that such type of analysis as Markov Decision Process (MDP) may be applied to it [8, 9, 17]. The main drawback of this model is the state-explosion problem. The second type was proposed by Noel and Jajodia [14] who represented nodes as disjoint sets of privileges. An attacker possesses several privileges at some stage of an attack, "owns" several nodes, i.e., the privileges she has is a union of the privileges assigned to these nodes. A transition requires "owning" certain nodes and leads to a new privilege. This model is free from the state explosion problem, but requires handling cycles and cannot be used for analysis with MDP.

Several authors considered transformation of attack trees into models which allows analysis of attacker steps as a sequence. Qin and Lee [16] proposed a conversion of attack trees into causal networks representing an order of execution of steps contained in the tree. Dalton et al. [5] and Piètre-Cambacédès and Boisou [15] proposed a transformation of attack trees into Petri Nets. The purpose of the transformation to Petri Nets is to use simulation to determine the likelihood of particular sequences of steps of the attacker, meaning paths in the attack tree. In contrast, we are proposing a mapping between an existing attack tree (expressed as a Si* model) and an existing attack graph. We rely on the precise information and existing probabilities in the attack graph and map these to the attack tree rather than determining probabilities via simulation.

3 A Method for Combining Attack Trees and Attack Graphs

Our approach is attacker centric, which means that we analyse a system against a certain type of an attacker (e.g., a network attacker). The attacker achieves high-level goals

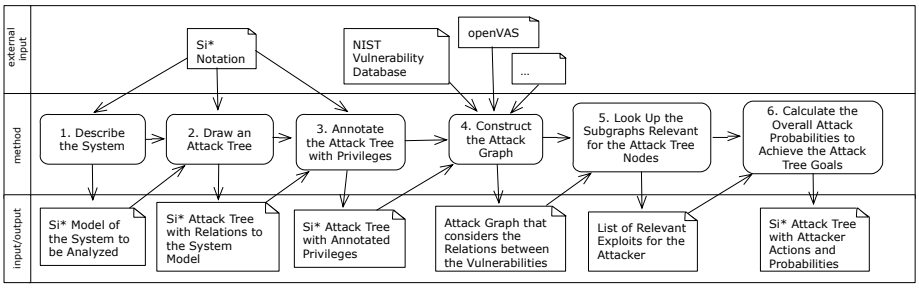

Fig. 1. An overview of our method for combining attack trees and attack graphs

executing low-level actions. For example, the attacker exploits vulnerabilities in an operating system and database software installed on a server in order to get an access to billing data. High-level goals of the attacker are usually described as an attack tree while low-level actions as an attack graph. We aim at finding sequences of low-level actions that an attacker needs to execute in order to achieve her high-level goals. We formalize attack trees and attack graphs and then present a formal approach for mapping goals to sequences of actions. This sequence is exploited latter for a probabilistic analysis of the attacker's plan. The analysis could be further exploited in the development of a defensive strategy against the attacker of the selected type.

Figure 1 shows our security analysis method that is explained further in this section. In the following we describe the steps of our approach in details.

Step 1. Describe the System Describe the system in a high level of abstraction using the Si* notation. All stakeholders and electronic systems in the scope of the analysis shall be in this system description. Ensure that all relations between elements of the system are a part of the Si* diagram.

Step 2. Draw an Attack Tree Select a type of attacker for the analysis from the types: network attacker, software attacker, social engineering attacker, or physical attacker (c.f., [2, 3]). Elicit the main motivations of the attacker using Si* soft goals and a corresponding main goal of the attacker. Divide the main goal into subgoals until all leaf goals of the tree concern a part of the system described in Step 1. Draw a relation from each leaf goal to a part of the system. The attack tree is the overall plan of the attacker.

The Si* language provides three types of operators for goal decomposition: AND, OR, and MEANS-END. The AND operator means that all subgoals must be satisfied to fulfil the target goal. The OR operator means, that at least one subgoal must be satisfied. The MEANS-END operator points out that the subgoals are required but they are not enough to satisfy the target goal. We assume that goals are decomposed into subgoals in a way that it is enough to achieve subgoals in order to fulfil target goal. Thus, we do not consider the MEANS-END operator. A goal tree of an attacker formed with only AND and OR operators is a well-known attack tree [18].

In our paper we use a formalization of attack trees described in [12]. The goal tree of an attacker is $T = (N, \mapsto, n_0)$ where N is a set of goals, $n_0 \in N$ is a top goal, and

$\mapsto$ is a finite acyclic relation: $\mapsto \subseteq N \times \mathbb{M}(N)$, where $\mathbb{M}(N)$ is a multiset[3] of goals N. Note, that if $n \mapsto \mathbb{M}(N')$, then $n \notin N'$, because T is a tree.

Step 3. Annotate the Attack Tree with Privileges Firstly, annotate each leaf goal of the tree with the initial (start) privileges that are required to start achieving the goals. Secondly, determine the resulting (end) privileges that the attacker gains after achieving each leaf goal.

In this work by a "privilege" we mean access rights, possessed resources, information, etc., required for or gained after an execution of an attack. Let P be a set of all possible privileges in a system and $\mathbb{P}(P)$ be a powerset of this set.

We assume there are two functions for getting such sets:

$$tprivsbgn : N \to \mathbb{P}(P) \tag{1}$$

$$tprivsend : N \to \mathbb{P}(P) \tag{2}$$

The function $tprivsbgn$ returns the set of privileges that the attacker needs to begin achieving a certain goal $n \in N$. The function $tprivsend$ returns the resulting set of privileges that the attacker possesses after achieving the goal n.

Step 4. Construct the Attack Graph Conduct vulnerability scanning and derive available actions Act of the attacker using different scanning tools (e.g., openVAS[4]). This is a usual procedure before construction of attack graphs although the procedure may be a tedious and time consuming [6, 14]. Afterwards, construct an attack graph based on the available actions.

In this work, we follow the model of attack graphs as it is presented in [6]. In order to build an attack graph a set of possible actions Act is defined using different scanning tools (e.g., openVAS[5]).

The result of a scanning is a set of vulnerabilities. Every action of an attacker $a \in Act$ is a single exploit of a vulnerability where Act is a set of possible actions. Let also $Act \subseteq \mathbb{P}(P) \times \mathbb{P}(P)$ be a relation such that:

$$(P^b, P^e) \in Act \,.\, P^b, P^e \in \mathbb{P}(P), P^b \subset P^e \tag{3}$$

Where P^b is a minimal set of privileges required to perform the action and P^e is the resulting set of privileges. In a system, the execution of an action may require some initial privileges. For example, an attacker needs to get root privileges on a server in order to run an exploit against a database installed on the server. Please note that we make the usual for attack graphs assumption that privileges once gained remain until the end of an attack [6].

In the sequel, we use superscript b to indicate the initial privileges when e specifies the ending privileges. We also use two special functions: fst and snd, which return the first and the second element of a Cartesian product, i.e., $fst\ a = P^b$ and $snd\ a = P^e$ for $a = (P^b, P^e)$. Finally, for the sake of brevity we define the privileges gained during step $a_i = (P_i^b, P_i^e) \in Act$ as: $\Delta_i = P_i^e \setminus P_i^b = snd\ a_i \setminus fst\ a_i$.

[3] In fact, Si* does not allow using the same subgoals in different parts of the tree, but we still keep multiset of nodes (instead of a powerset) for compliance with [12].

[4] http://www.openvas.org/

[5] http://www.openvas.org/

Let $seq\ Act$ be a set of all possible sequences of elements from Act and $s \in seq\ Act$ be a sequence of actions of an attacker:

$$s = a_1 \dots a_n \,.\, \forall\, i \in \{1, \dots, n\},\ a_i \in Act, \tag{4}$$
$$\forall\, j \in \{2, \dots, n\},\ fst\ a_j \subseteq \bigcup_{k=1}^{j-1} snd\ a_k,\ snd\ a_j \not\subseteq \bigcup_{k=1}^{j-1} snd\ a_k$$

An i-th element of a sequence s is an action $a_i = s[i]$, $i \in \{1, ..., n\}$, and $n = \#s$ is the length of the sequence.

Definition 1. *Let P be a set of all possible privileges and Act be a set of all possible attacker actions found in the system. Then, the attack graph $\mathcal{G} \subseteq \mathbb{P}(\mathbb{P}(P) \times Act \times \mathbb{P}(P))$ associated to P and Act is defined as follows:*

$$G := \{(P^b, a, P^e) \in \mathbb{P}(P) \times Act \times \mathbb{P}(P) \mid \tag{5}$$
$$1)\ fst\ a \subseteq P^b,\ 2)\ P^e = P^b \cup snd\ a,\ 3)\ snd\ a \setminus fst\ a \not\subseteq P^b\}$$

In words, the attack graph is defined as a set of edges, which relate to actions and allow an attacker to move from one set of privileges to a wider set. A vertex in the attack graph is a set of privileges. The attack graph defined in Definition 1 is a direct acyclic graph (DAG).

A path π in an attack graph is a sequence of edges. We may say that the π is also an attack graph with ordered edges. We define sequential numbers for these edges $\pi[i]$ for all $i \in \{1, \dots, n\}$ where $n = \#\pi$. We assume that there is a function $Paths(P^b, P^e)$ that returns all paths from P^b to P^e.

In our analysis we are interested in sequences of attacker's actions that are required to satisfy some goal of the attacker. We would like to derive these sequences from an attack graph as a subgraph. However, there is the following issue with the graph in Definition 1. One action in the real world can refer to multiple edges in the graph. Thus, the multiple distinct paths can refer to the same sequence of actions in the real world.

To address this issue, we, first, define a relation between a sequence of actions and a path.

Definition 2. *Let Act be a set of actions, $s \in seq\ Act$ be some sequence of these actions, G be an attack graph, $\Pi(G)$ be a set of paths π in G. Then, we define the relation $\rightrightarrows \subseteq seq\ Act \times \Pi(G)$:*

$$\rightrightarrows := \{(s, \pi) \mid s \in seq\ Act,\ \pi \in Act,\ \#s = \#\pi, \tag{6}$$
$$\forall\, i \in \{1, \dots, \#\pi\},\ \pi[i] = (P^b, s[i], P^e), P^b, P^e \in \mathbb{P}(P)\}$$

Second, we would like to show formally that it is enough to consider paths starting from a single vertex with required amount of initial privileges, because paths from other vertices containing the same amount of privileges will correspond to the same sequences.

Theorem 1. *Let s be some sequence of actions and G be an attack graph. Then:*

$$\forall\, \tilde{P}^b \in \mathbb{P}(P)\,.\,\forall\, i \in \{1,\ldots,\#s\}, \tag{7}$$
$$1)\, \mathit{fst}\ a_i \setminus \bigcup_{j=1}^{i-1} \Delta_j \subseteq \tilde{P}^b,\ 2)\, \Delta_i \setminus \bigcup_{j=1}^{i-1} \Delta_j \not\subseteq \tilde{P}^b \Rightarrow$$
$$\exists\, \hat{\pi} \in \mathit{Paths}(\tilde{P}^b, \tilde{P}^e)\,.\, s \rightrightarrows \hat{\pi}$$

Where $\tilde{P}^e = \tilde{P}^b \cup \bigcup_{i=1}^{\#s} \Delta_i$.

*Thus, an attacker can execute the sequence where for each next step she either has the privileges at the beginning or she gains the privileges during earlier steps of the sequence (*condition 1*). Moreover, in any step of the sequence the attacker must gain something she did not have at the beginning and did not gain during earlier steps (*condition 2*) .*

Proof. We prove by induction that there is a path from some $\tilde{P}^b \in \mathbb{P}(P)$ which satisfies the conditions of Theorem 1. To do that, we need to show, that for every action a_k, $k \in \{1, ..., n\}$, $n = \#s$ there is always an edge related to this action. This edge starts with $\tilde{P}^b_k$ to which there is a path from $\tilde{P}^b$ related to the sequence of actions from s predeceasing a_k. We use Definition 1 to prove the existence of such edge.

First, we always can reach $\tilde{P}_0 = \tilde{P}^b$ with an empty s.

Second, lets assume, that there is a path $\hat{\pi}_{k-1}$ from $\tilde{P}^b$ to $\tilde{P}^b_k$, which relates to a sequence s_{k-1} formed with $k-1$ first actions of s (i.e., $s_{k-1} \rightrightarrows \hat{\pi}_{k-1}$). Using Equation 5 we prove that there is always an edge $(\tilde{P}^b_k, a_k, \tilde{P}^b_{k+1}) \in G$.

The amount of privileges an attacker gains after executing some action a_i is Δ_i. Therefore, the set of privileges $\tilde{P}^b_k$ is equal to the privileges $\tilde{P}^b$ plus all privileges gained after executing $k-1$ steps, i.e., $\tilde{P}^b_k = \tilde{P}^b \cup \bigcup_{i=1}^{k-1} \Delta_i$. Now according to the first condition in the theorem, we have:

$$\tilde{P}^b_k = \tilde{P}^b \cup \bigcup_{i=1}^{k-1} \Delta_i \supseteq (\bigcup_{i=1}^{\#s} \mathit{fst}\ a_i \setminus \bigcup_{j=1}^{i-1} \Delta_j) \cup \bigcup_{i=1}^{k-1} \Delta_i \supseteq \tag{8}$$
$$(\mathit{fst}\ a_k \setminus \bigcup_{j=1}^{k-1} \Delta_j) \cup \bigcup_{i=1}^{k-1} \Delta_i = \mathit{fst}\ a_k$$

Thus, $\mathit{fst}\ a_k \subseteq \tilde{P}^b_k$, which is the first conditions in Equation 5.

The second condition of Equation 5 is satisfied because both $\tilde{P}^b_k$, $\mathit{snd}\ a_k \in \mathbb{P}(P)$, therefore there is always a set of privileges $\tilde{P}^b_k \cup \mathit{snd}\ a_k \in \mathbb{P}(P)$.

According to the second condition of the theorem we see that $\tilde{P}^b_k = \tilde{P}^b \cup \bigcup_{i=1}^{k-1} \Delta_i \not\supseteq \Delta_k$, thus, $\mathit{snd}\ a_i \setminus \mathit{fst}\ a_i \not\subseteq P^b_k$. Therefore, the third condition of Equation 5 is satisfied and the edge $(\tilde{P}^b_k, a_k, \tilde{P}^b_{k+1})$ exists in the graph G.

To conclude the step we define an auxiliary function $front(P')$ which gives a minimal set of vertices where a set of privileges P' is achieved:

Definition 3. *Let $P' \subseteq P$ be some set of privileges and G be an attack graph. Then, the function $front : \mathbb{P}(P) \to \mathbb{P}(\mathbb{P}(P))$ is defined as follows:*

$$front(P') := \{P^f \in \mathbb{P}(P) \mid \exists (P^b, a, P^f) \in G,\ P' \not\subseteq P^b,\ P' \subseteq P^f\} \tag{9}$$

According to Theorem 1, the same sequences, that require initial privileges P', start from every vertex in $front(P')$.

Step 5. Look Up the Subgraphs Relevant for the Attack Tree Nodes Query the graph from the set of the initial privileges of a leaf goal to the resulting privileges. Derive the further subgraphs for each node to refine the high level attacker goals to concrete attack actions.

Before we start mapping goals from a goal tree with parts of an attack graph we must make some assumptions. First, we assume that an attack graph and a goal tree were created separately but for the same system. Therefore, they must correspond to each other. We assume that the attacker behaves only according to the plan and use only the privileges specified in the goal tree. Another important aspect of goal tree is that its leaves should be independent from each other, i.e., the way of execution of one goal should not affect another one apart of the sequence of actions. Usually, quantitative analysis of attack trees makes the same assumption (see for example [18]).

In particular, these assumptions mean the following.

- We ignore the paths which require more privileges than specified by the starting privileges assigned to a goal (assumption of correspondence).
- The actions which do not lead the attacker to his goal are possible but they should not affect our analysis (assumption of "useless" actions).
- The attacker heads towards the states where *only* the target privileges are satisfied (plus some irrelevant privileges). In other words, no privileges to be obtained on the later steps should be gained at early ones (assumption of independence).

The later assumption, which follows from the independence of goals, shows that we have to consider every goal in the context of the whole tree.

Let us have a complete attack graph G for a system and a goal tree T of an attacker. And let us aim at finding a subgraph corresponding to a leaf goal n from the tree. In fact, we are interested only in a part of the complete attack graph. The complete graph created in Definition 1 is usually huge and most parts are not considered in the plan of the attacker (the tree) and, thus, are not relevant for us. Therefore, we purge the graph as most attack graph techniques do (e.g., [6]). We consider only the part of the graph which can be potentially exploited by the attacker. Thus, we consider an attacker with initial set of privileges P^i and the target set of privileges P^t. Therefore, we are interested only in the part of G which is built by all paths from $front(P^i)$ to all nodes from $front(P^t)$. This selection does not affect the following discussion, but reduces the computational power required to implement our method.

In order to find the subgraph G_n which realizes the leaf goal n from the tree we need to know the initial privileges P^b of an attacker and final privileges P^e corresponding to

the goal. According to Equations 1 and 2:

$$P^b = tprivsbgn(n) \tag{10}$$
$$P^e = tprivsend(n) \tag{11}$$

Next we look for the set of vertices (sets of privileges) $\boldsymbol{P}^{fb}$ in the graph from which the attacker may start achieving the goal n and vertices $\boldsymbol{P}^{fe}$ where this goal is achieved.

First, we select all possible sequences of actions which lead an attacker from starting privileges to end privileges and which require *only* P^b as starting set of privileges.

$$S^b := \{s \in seq\ Act \mid \exists\, \pi \in Paths(\tilde{P}^b, \tilde{P}^e),\ \tilde{P}^b \in front(P^b), \tag{12}$$
$$\tilde{P}^e \in front(P^e),\ s \rightrightarrows \pi,\ \forall\, i \in \{1, \ldots, \#s\},\ fst\ a_i \subseteq P^b \cup \bigcup_{j=1}^{i-1} \Delta_j\}$$

We need also a set of sequences of actions from end privileges to target privileges, since our tree must be independent from the perspective of future steps.

$$S^e := \{s : seq\ Act \mid \exists\, \pi \in Paths(\tilde{P}^e, \tilde{P}^t),\ \tilde{P}^e \in front(P^e), \tag{13}$$
$$\tilde{P}^t \in front(P^t),\ s \rightrightarrows \pi,\ \forall\, i \in \{1, \ldots, \#s\},\ fst\ a_i \subseteq P^e \cup \bigcup_{j=1}^{i-1} \Delta_j\}$$

Second, we are able to filter fronts for P^b and P^e to remove the sets of privileges, which already have the privileges scheduled to be obtained on the later steps (assumption of independence).

$$\boldsymbol{P}^{fb} := \{P^{fb} \in \mathbb{P}(P) \mid P^{fb} \in front(P^b), \forall\, s \in S^b, \tag{14}$$
$$\forall\, i \in \{1, \ldots, \#s\},\ \Delta_i \not\subseteq P^{fb}\}$$
$$\boldsymbol{P}^{fe} := \{P^{fe} \in \mathbb{P}(P) \mid P^{fe} \in front(P^e), \forall\, s \in S^e, \tag{15}$$
$$\forall\, i \in \{1, \ldots, \#s\},\ \Delta_i \not\subseteq P^{fe}\}$$

The subgraph G_n corresponding to the goal n consists of paths from vertices in $\boldsymbol{P}^{fb}$ to vertices in $\boldsymbol{P}^{fe}$. However we may consider only one arbitrary vertex $P^{fb} \in \boldsymbol{P}^{fb}$ and look for paths to all $P^{fe} \in \boldsymbol{P}^{fe}$ since all sequences of actions corresponding to paths from other vertices in $\boldsymbol{P}^{fb}$ are *the same due to Theorem 1*:

$$G_n = \bigcup_{\forall\, P^{fe} \in \boldsymbol{P}^{fe}} \{\pi \in \Pi(G) \mid \pi \in Paths(P^{fb}, P^{fe}), \tag{16}$$
$$\nexists (P', a, P'') \in \pi,\ P', P'' \in \boldsymbol{P}_f^b,\ P', P'' \in \boldsymbol{P}^{fe},$$
$$\exists\, s \in S^b,\ s \rightrightarrows \pi\}$$

Equation 16 gives us a subgraph formed with an aggregated set of paths from P_f^b to $\boldsymbol{P}_f^e$ (first line), which do not contain any other nodes from these sets, apart of P_f^b

and P_f^e (second line), and which do not use additional privileges that are gained on the further steps (third line). The attacker will follow one of the paths specified in G_n in order to achieve the goal n.

We would like to make a small remark. Some goals in a goal tree state *what* the attacker should achieve (e.g., "get login and password") when other goals also define *how* to achieve the goal (e.g., "eavesdrop login and password" or "bribe an employee to get login and password"). The goals of the second type restrict the ways for getting to the desired state. In order to address this restriction someone can define a function which assigns a set of essential actions for every goal in a goal tree:

$$esseqs : N \to \mathbb{P}(seq\ Act) \tag{17}$$

This function should be further used to filter the set of sequences S^b in order to obtain only sequences relevant to the way of achieving the goal.

Step 6. Calculate the Overall Attack Probabilities to Achieve the Attack Tree Goals Analyze obtained subgraphs and calculate the probabilities of achieving corresponding subgoals. Annotate the attack tree with the probabilities. Derive the overall probability for the attacker to achieve the main goal.

To find the probability of successful achievement of a goal n (denoted as $\mathbf{pr}[n]$) we need to find the probability to get from an initial vertex P_f^b of G_n to one of the target vertices from $\boldsymbol{P}_f^e$ (see Equation 15). We assume that an attacker may fail to exploit a vulnerability (e.g., when it is already patched or it is too hard) and, thus, for every action we have a probability of successful exploitation, denoted as $\mathbf{pr}[a]$.

We consider a greedy attacker who would like to execute the attack with the highest probability of success. Thus, the attacker tries the most probable path to achieve the goal. Since the attacker may fail to make an action and thus, to complete the most probable path, the attacker will try an alternative path, and so on until she reaches her goal (or fail all paths). We assume, that we have an adapted algorithm of Edsger Dijkstra for a search of the shortest path P_f^b to $\boldsymbol{P}_f^e$, called here as **shortestPath**.

Although, one action cannot be used in one attack several times, the same action may be used in different attacks. Thus, if an attacker successfully performs an action during execution of one attack but then switches to another attack, then there is no need to perform the same step again. We assign the probability of all edges referring to the successfully performed action to one. Once failed an action cannot be used in other attacks and we change its probability to zero. Algorithm 1 shows a recursive function **computeProbability** which consists of two parts. The first part contains a search for a shortest path (line 2), computation of the probability of successful exploitation of the path (line 4), computation of the probability to reach a certain amount of privileges (line 5), and changing of the probability associated with an action to 1 (as successfully performed).

The second part follows the path backwards assuming that an action failed to succeed. In this case the attacker should find an alternative path. Thus, she is going back to the last set of privileges, where alternatives are possible, called a *fork* (step 8). Note, that only the nearest fork should be considered, since it contains all privileges of the vertexes down in the path, and thus all possible alternatives for the considered failure are rooted in this vertex. Since the attacker considers alternatives only in case of a failure of the shortest path, then we compute the probability of such failure (line 10). Then

```
   Input  : P^s – starting vulnerability
   Output: pr[n] – probability of successful exploitation
   Global : G – associated graph
            P^e – set of target privileges
   Data: pr[P] – probability to reach a certain vertex
   pr[a] – probability to perform action a successfully
   pr^fail – probability to fail at a certain step
   P^f – a fork, i.e., a vulnerability with several outgoing edges
   G^p – a shortest path
1  function computeProbability(P^s)
2      pr[n] = 1;
3      Find G^p = shortestPath(P^s, P^e);
4      for ∀(P', a, P'') ∈ G^p forward do
5          pr[n] = pr[n] · pr[a];
6          pr[P''] = pr[n];
7          pr[a] = 1;
8      for ∀(P', a, P'') ∈ G^p backward do
9          Find nearest fork at P^f or exit;
10         Restore pr[a], where a ⇉ x̃;
11         pr^fail = pr[P'] · (1 − pr[a]);
12         pr[a] = 0;
13         pr[n] = pr[n] + pr^fail ·computeProbability(P^f);
14         Restore pr[a];
15     return pr[n];
```

Algorithm 1: Computation of probabilities for a goal

we put to 0 the probability of the action which is considered as failed. After that, we add the probability to re-consider the course of action to the overall probability (line 12). Finally, we restore the initial value of the probability of the considered action (since in the next round of the cycle we will assume another action to fail).

When the probabilities for leaf goals are computed and assigned, we can find the set of leaf goals, which lead an attacker to target goal with highest probability of success. The countermeasures should be applied against the vulnerabilities from the subgraphs corresponding to these leaf goals.

4 Application of our Method to a Smart Grid Scenario

We illustrate the application of our method to a smart grid example in the following. **Step 1. Describe the System.** Figure 2 (upper part) presents the part of an Si* diagram of our smart grid scenario, which we would like to use as an illustrative example. The actor *Huntsville Consortium* plays the role of the *Energy Supplier* (circles in the Fig. 2). The *Huntsville Consortium* has the main goal *Sell Energy* for which the subgoals *Collect Prosumer Data* and *Calculate Bill* need to be fulfilled (rounded rectangles in Figure 2). Various resources (represented as rectangles) are required for achieving these goals, *Prosumer Information* is required for *Collect Prosumer Data* goal and *Aggregated Billing Data* is required for the goal *Calculate Bill*. Another goal of the consortium is to *Provide Grid Services*, which also can be broken down into the three subgoals: *Manage ESS*, *Manage EMS*, and *Manage Smart Meters*.

The energy supplier server (ESS) collects metering data from the smart meters, as well as stores and aggregates this data. The *Manage ESS* goal requires obviously an

ESS. The ESS uses a specific *ESS Network Gateway* to communicate with the smart meters and authorized external entities, e.g., the *Billing Operator* uses the *Billing Operator's Laptop* to get billing data, which are stored in the *ESS Database (DB)*. The *Billing Operator* requires this data to fulfill the goal *Calculate Bill*, which is also delegate from the *Huntsville Consortium*.

The home energy management system (EMS) controls the smart appliances in the smart home. It is a computer system that visualizes the prosumers energy consumption and support the selection of offers for buying and selling energy. The EMS relies upon a *Network Gateway* to communicate with authorized external parties. The goal *Manage EMS* is delegated to the *Meter Point Operator*, who conducts the maintenance for the EMS. The *Meter Point Operator* has also been delegated the goal *Manage Smart Meters*, meaning he/she conducts the maintenance for Smart Meters. The maintenance can be conducted partially from a remote location, which makes this goal also reliant on the *Network Gateway*. We focus on attacks on the energy supplier, i.e., *Huntsville Consortium*. Note that we include the entire scope (large oval with all the actor's goals inside) of the actor *Huntsville Consortium*. In particular, goals that are delegated to other actors, as well. For the sake of brevity, we show only the threats involving the goal *Manage ESS*.

Step 2. Draw an Attack Tree. We show exemplary our threat analysis for a network attacker (the discussion on other types of attackers and their intentions could be found in [2, 3]). The network attacker is of interest for this scenario, because the smart home scenario relies on a secure ICT network. The grid is not usable anymore when data cannot be transmitted over it. Figure 2 (bottom part) describes the possible plan of a network attacker as an attack tree in Si*. The network attacker (see Fig. 2) is motivated by *Financial Gain*, *Self Interest*, and *Curiosity*. Driven by these motivations the attacker forms the goal to *Change Metering Data*. This goal represents a threat that can harm *Aggregated Billing Data*, *Prosumer Information* and others. There are different ways to achieve this goal. For example, the attacker may *Change Metering Data during Transmission* or *Get Local Access to ESS*, *Gain Access to Database Locally* and directly change data in the ESS database.

The attack tree is designed with the scope of the analysis in mind, meaning the scope of the Huntsville Consortium (Fig. 2 top part). This is reflected by the relations from each leaf goal to a resource in the scope. For example, the leaf goal *Gain Access to Database Locally* has a relation to the resource *ESS Database*. This resource is exploited by the attacker to satisfy the leaf goal. It represents an entry point of the attacker into the scope.

Hence, our Si* diagram contains: the scope of the analysis, the motivations of the attacker, the subsequent goals, and resulting entry points of the attacker from leaf goals to the scope. In the future, we will use all these information to check of our threat analysis for completeness with validation conditions. For example, we can check if all leaf goals exploit at least one resource in the scope. In addition, we could check if there are resources in the scope that do not represent entry points.

Remember that this diagram (Fig. 2) represents the high-level plan of the attacker. Later we enrich the information in the plan with low-level details existing in the system related attack graph (Fig. 3).

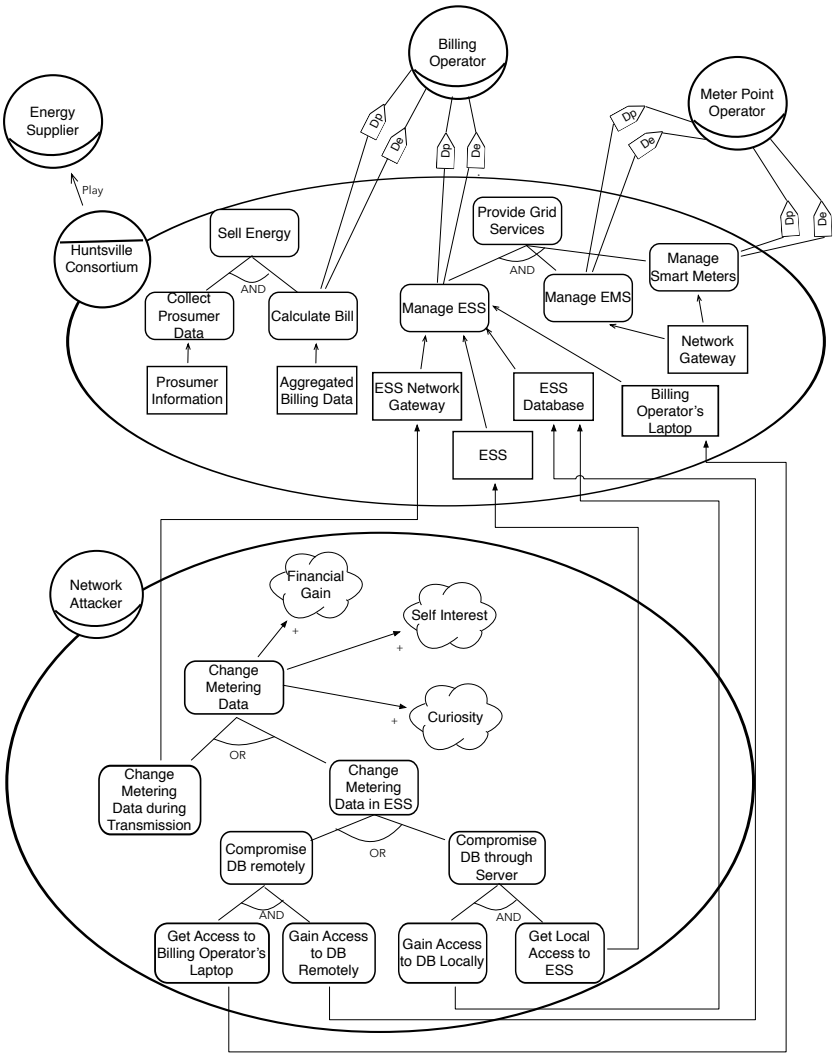

Fig. 2. Si* model for a network attacker in our smart grid example

Step 3. Annotate the Attack Tree with Privileges. We show the privileges relevant for our running example in Tab. 1. The table lists the initial and resulting privileges for each leaf goal in the tree. The initial privileges are the ones required to start a sequence of actions to achieve the goal. The resulting privileges contain the gained privileges by the sequence of actions in addition to the initial privileges.

Step 4. Construct the Attack Graph. The Billing Operator has an infrastructure in order to achieve the goal *Manage ESS*. We assume that the infrastructure consists of a laptop and a workstation that are used by an administrator in order to manage the ESS.

Table 1. Available privileges considered in our running example

Goal name	Initial privileges	Resulting privileges
Change data during the transmission	∅	{ruggedcom}
Get access to billing operator's laptop	∅	{laptop}
Gain access to the DB remotely	{laptop}	{laptop,mysql}
Get local access to ESS	∅	{server}
Gain access to the DB locally	{server}	{server,mysql}

Table 2. Actions available for the attacker in our example

CVE code	Software	Initial privileges	Resulting privileges
CVE-2011-3108	Chrome browser	∅	$\{laptop\}$
CVE-2009-4781	TUKEVA Pass.Man.	$\{laptop\}$	$\{laptop, database\}$
CVE-2012-2369	Pidgin	∅	$\{workstation\}$
CVE-2011-4862	FreeBSD	$\{workstation\}$	$\{workstation, server\}$
CVE-2012-0173	Windows	$\{workstation\}$	$\{workstation, laptop\}$
CVE-2011-4913	Linux	$\{laptop\}$	$\{workstation, laptop\}$
CVE-2013-6926	Rugged OS	∅	$\{ruggedcom\}$
CVE-2012-0114	MySQL	$\{server\}$	$\{server, database\}$

We assume that the laptop has Windows OS installed and the workstation has Linux OS installed. The laptop and the workstation are connected by VPN. There is the ESS itself with a database server installed. The server runs FreeBSD OS and the database server is Oracle MySQL. The server and the workstation are connected by LAN. Clients access the server via network that is based on the RuggedCom equipment with RuggedCom OS installed.

The vulnerabilities related to the infrastructure are listed in Tab. 2. The table contains CVE vulnerability codes, software that contains a vulnerability, and sets of initial and resulting privileges for each vulnerability. The privilege values mean the root access to the specified network node. These vulnerabilities are obtained from the NIST Vulnerability Database[6].

An attack graph related to the infrastructure is presented in Fig. 3 and the list of privileges for each vertex is presented in Tab. 3. The edges of the graph are labeled by CVE vulnerability codes from Tab. 2. In this graph we show only one edge related to the action *CVE-2013-6926* for simplicity's sake. We use the names of vertices from Tab. 3 such as v_1 and v_3 instead of the full set of privileges, i.e. such that $\{workstation\}$ and $\{workstation, server, database\}$.

Step 5. Look Up the Subgraphs relevant for the Attack Tree Nodes. The main privilege to achieve is to have possibility to modify metering data, which could be seen as a subset of vertices where root access to RuggedCom or database (e.g., to reach v_3, v_5, v_8, v_9 or v_{10}). There are three alternative ways to achieve the main goal for the attacker (see corresponding subgoals on Fig. 2). The first one is to fulfill the subgoal *Change Metering Data during Transmission* (n_{tran}) compromising RuggedCom equipment. The goal in this case specifies the way, how the goal should be achieved, i.e., "during transmission", and $esseqs(n_{tran}) = \{\langle$CVE-2013-6926$\rangle\}$. The mapping is straightforward: one goal to a subgraph $G_{n_{tran}} = \{(v_0, \text{CVE-2013-6926}, v_8)\}$.

[6] NIST Vulnerability Database: http://nvd.nist.gov/

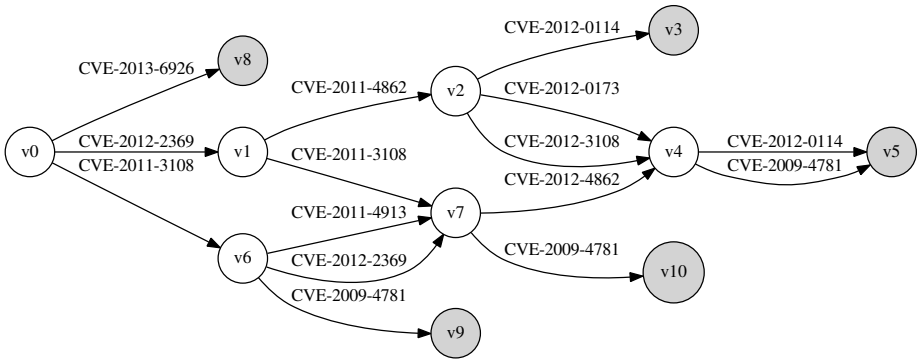

Fig. 3. A part of the attack graph for our smart grid example

Other ways require either getting root access to the server or to a workstation and then getting root access to the database. Consider *Get Local Access to ESS* goal (n_{ESS}). In our example, $tprivsbgn(n_{ESS}) = \varnothing$ and $tprivsend(n_{ESS}) = \{freebsd\}$ and:

$$\begin{aligned} esseqs(n_{ESS}) = \{&\langle \text{CVE-2012-2369}, \text{CVE-2011-4862}\rangle, \\ &\langle \text{CVE-2012-2369}, \text{CVE-2011-3108}, \text{CVE-2011-4862}\rangle, \\ &\langle \text{CVE-2011-3108}, \text{CVE-2011-4913}, \text{CVE-2012-4862}\rangle, \\ &\langle \text{CVE-2011-3108}, \text{CVE-2011-2369}, \text{CVE-2012-4862}\rangle\} \end{aligned} \tag{18}$$

Thus, the attacker should move from v_0 to v_2 or v_4 and, by Equation 16, $G_{n_{ESS}}$ includes vertices $v_0, v_1, v_2, v_4, v_6, v_7$ and the edges between them, apart of CVE-2012-0173 and CVE-2012-3108 which are eliminated by the second line of Equation 16. We see that the next subgoal *Gain Access to DB locally* (n_{DBl}) can be achieved by the same action CVE-2012-0114 from v_2 and v_4, when vulnerability CVE-2009-4781 is not considered in this case, since the attacker decided to get *Remote Access to ESS* using the privileges from the earlier steps. I.e.:

$$G_{n_{DBl}} = \{(v_2, \text{CVE-2012-0114}, v_3), (v_4, \text{CVE-2012-0114}, v_5)\} \tag{19}$$

Table 3. Privileges corresponding to the nodes of the attack graph

Node	Set of privileges
v_0	$\varnothing$
v_1	$\{workstation\}$
v_2	$\{workstation, server\}$
v_3	$\{workstation, server, database\}$
v_4	$\{workstation, server, laptop\}$
v_5	$\{workstation, server, laptop, database\}$
v_6	$\{laptop\}$
v_7	$\{laptop, workstation\}$
v_8	$\{ruggedcom\}$
v_9	$\{laptop, database\}$
v_{10}	$\{laptop, workstation, database\}$

which are equivalent from the point of view of applied actions. Thus, their probability is the same. Similar reasoning could be applied to the third alternative, which leads from v_0 to v_4, v_6 or v_7 and then CVE-2009-4781 is the last action for *Gain Access to DB remotely*.

Step 6. Calculate the Overall Attack Probabilities to Achieve the Attack Tree Goals. The shortest path in $G_{n_{ESS}}$ is $\langle$CVE-2012-2369, CVE-2011-4862$\rangle$. If the probabilities of the actions are $\mathbf{pr}_1 = 0.6$ and $\mathbf{pr}_2 = 0.9$, then the overall probability to execute only this path is 0.54. Note that, in case CVE-2011-4862 is patched, there is no way for the attacker to achieve his goal (since the same vulnerability must be used to follow from v_7 to v_4). In case of CVE-2012-2369 patched, there is a path to achieve the goal and the attacker should add its probability (let it be 0.36) to the overall computation. Thus, the attacker first tries CVE-2012-2369, fails, and then she follows the only available path. The probability to *Get Access to ESS* is $(1 - 0.6) * 0.36 + 0.54 = 0.684$.

5 Conclusions

We contributed a structured method for threat analysis that concerns the mapping of the plan of the attacker (represented as an attack tree) to concrete vulnerabilities of a system (documented in an attack graph). We showed that it is possible to extract a part of a complex graph, which relates to a specific goal in the attack tree. We found that the complexity of the analysis of attack graphs can be significantly reduced because for a specific attacker we can consider only a part of the whole attack graph. We proposed an algorithm that computes the overall probability of success of an attacker on the basis of the mapping. Finally, we illustrated our method on a smart grid example.

In the future, we will consider in detail how the method can be used to identify the most appropriate countermeasures. Furthermore, we plan to identify further attacker motivations and provide a more extensive application of our approach in collaboration with industrial partners. Finally, we are planing to refine our method for creating an ISO 27001 compliant Information Security Management System.

References

1. Y. Asnar and F. Massacci. A method for security governance, risk, and compliance (grc): A goal-process approach. In *FOSAD Tutorial Lectures*, pages 152–184. Springer, 2011.
2. K. Beckers. Goal-based establishment of an information security management system compliant to iso 27001. In *SOFSEM*, LNCS 8327, pages 102–113. Springer, 2014.
3. K. Beckers, I. Côté, D. Hatebur, S. Faßbender, and M. Heisel. Common Criteria CompliAnt Software Development (CC-CASD). In *Proceedings of 28th SAC*, pages 937–943. ACM, 2013.
4. S. Bistarelli, F. Fioravanti, and P. Peretti. Defense trees for economic evaluation of security investments. In *Proceedings of the 1st ARES*, pages 416–423, 2006. IEEE.
5. G. C. Dalton II, J. M. Colombi, R. F. Mills, and R. A. Raines. Analyzing attack trees using generalized stochastic petri nets. In *Proceedings of the IAS*, pages 116–123. IEEE, 2006.
6. S. Jha, O. Sheyner, and J. Wing. Two formal analyses of attack graphs. In *Proceedings of the 2002 IEEE CSF*, page 49, 2002. IEEE.

7. J. Jürjens. Using UMLsec and goal trees for secure systems development. In *Proceedings of the 2002 SAC*, pages 1026–1030. ACM Press, 2002.
8. L. Krautsevich, F. Martinelli, and A. Yautsiukhin. Towards modelling adaptive attacker's behaviour. In *Proceedings of 5th FPS*, pages 357–364. Springer, 2012.
9. E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke. Model-based security metrics using adversary view security evaluation (advise). In *Proceedings of the 8th QEST*, pages 191–200. IEEE, 2011.
10. L. Liu, E. Yu, and J. Mylopoulos. Security and privacy requirements analysis within a social setting. In *Proceedings of the 11th RE*, pages 151–161. IEEE, 2003.
11. F. Massacci, J. Mylopoulos, and N. Zannone. Security requirements engineering: The si* modeling language and the secure tropos methodology. In Z. Ras and L.-S. Tsay, editors, *Advances in Intelligent Information Systems*, volume 265 of *Studies in Computational Intelligence*, pages 147–174. Springer, 2010.
12. S. Mauw and M. Oostdijk. Foundations of attack trees. In *Proceedings of the 8th ICISC*, Lecture Notes in Computer Science. Springer-Verlag, 2005.
13. H. Mouratidis, P. Giorgini, and G. Manson. Using security attack scenarios to analyse security during information systems design. In *Proceedings of ICEIS*, pages 10–17, 2004.
14. S. Noel and S. Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the VizSEC/DMSEC*, 2004.
15. L. Piètre-Cambacédès and M. Bouissou. Beyond attack trees: Dynamic security modeling with boolean logic driven markov processes (bdmp). In *Proceedings of the EDCC*, pages 199–208. IEEE, 2010.
16. X. Qin and W. Lee. Attack plan recognition and prediction using causal networks. In *Proceedings of the 20th ACSAC*, pages 370–379. IEEE, 2004.
17. C. Sarraute, G. Richarte, and J. L. Obes. An algorithm to find optimal attack paths in nondeterministic scenarios. In *Proceedings of the 4th AISec*, pages 71–80. ACM, 2011.
18. B. Schneier. Attack trees: Modelling security threats. *Dr. Dobb's journal*, December 1999.
19. A. van Lamsweerde. Elaborating security requirements by construction of intentional anti-models. In *Proceedings of the 26th ICSE*, pages 148–157. IEEE, 2004.