OSEAD:
How to save money using optical power meters
Case study and tool experience report

Barbara Fila and Wojciech Wideł

Univ Rennes, INSA Rennes, CNRS, IRISA, Rennes, France
{barbara.kordy, wojciech.widel}@irisa.fr

Abstract. Tampering with their power meter might be tempting to many people. Appropriately configured home-placed meter will record lower than the actual electricity consumption, resulting in substantial savings for the household. Organizations such as national departments of energy have thus been interested in analyzing the feasibility of illegal activities of this type. Indeed, since nearly every apartment is equipped with a power meter, the negative financial impact of tampering implemented at a large scale might be disastrous for electricity providers.
In this work, we report on a detailed analysis of the power meter tampering scenario using attack–defense trees. We take various quantitative aspects into account, in order to identify optimal strategies for customers trying to lower their electricity bills, and for electricity providers aiming at securing their infrastructures from thefts. This case study allowed us to validate some advanced methods for quantitative analysis of attack–defense trees as well as evaluate the `OSEAD` tool that we have developed to support and automate the underlying computations.

1 Introduction

Electricity theft is a widespread proceder [16,22] that generates huge financial losses yearly across the world [12,17,24,33], with more than the third of the losses affecting the BRIC countries (Brazil, Russia, India and China) [24]. One of the ways in which electricity is being stolen, is by tampering with power meter in a way that results in the household's or facility's power consumption being under-reported. Modern smart meters make identifying crude power meter tampering attempts easier, but remain vulnerable to (not necessarily sophisticated) hacking attacks [27].

This study is concerned with the issue of tampering with power meters. We consider a malicious user whose aim is to reconfigure their power meter, in order to lower the recorded electricity consumption of their household. We extend the *attack tree*-based model of possible behaviour of such a user, analyzed by the U.S. Department of Energy in [28], to take possible countermeasures into account. We report the results of analysis of the obtained *attack–defense tree* (ADTree) model using some of the recent developments in the field.

Our objectives. The objective of the work presented in this report is threefold. First, we thoroughly analyze a tampering with a power meter scenario which is of high importance for the industrial community. The goal is to identify optimal attacks and optimal defender's strategies, while taking various optimality criteria into account. Second, we test in practice some recently proposed techniques for quantitative analysis of ADTrees: the one from [21], focusing on the correctness of the quantitative analysis of trees with repeated actions, the one from [10], addressing the problem of multi-parameter evaluation on such trees, and the one from [20], which relies on linear integer programming (ILP) and aims at selecting optimal sets of countermeasures. Third, we develop and validate a tool, called `OSEAD` – *Optimal Strategies Extractor for Attack–Defense trees* – supporting an automated handling of the techniques from [20,21], and [10].

Related studies. ADTrees have already been used in the past to perform practical studies of security scenarios. The security of ATM machines was analyzed in [11]. The main difference between [11] and the current study is that the former focuses on the modeling aspects only, i.e., it does not involve any quantitative analysis. In [4], an RFID-based management system has been analyzed. This work resulted in a list of guidelines describing how to carry out a case study involving the ADTree modeling and its quantitative analysis. These guidelines were respected in our power meter study. However, [4] concentrates on analysis wrt single parameter, and it uses the classical bottom-up approach, sketched in [32] and formalized in [25] and [18], which is not well-suited for trees with repeated basic actions. In the current work, we employ novel, recently developed techniques for quantitative analysis of ADTrees with repeated basic actions, and use and validate a tool that we developed to support an automated handling of these techniques. Unlike in [4] and [11], the power meter ADTree is based on an attack tree developed by industry, so the people performing the analysis are not the sole authors of the analyzed model.

Paper structure. The ADTree for tampering with the power meter is described in Section 2. The parameters of interest for our study, their values, and the techniques we employ to identify optimal attacks and optimal defender's strategies are presented in Section 3. The results of the actual analysis are discussed in Section 4, and Section 5 is devoted to the `OSEAD` tool. We conclude in Section 6.

2 Tampering with a power meter scenario

We start with presenting the scope of this study, which includes the profiles of the actors involved, a specification of the analyzed system, and the modeling technique that we use, in Section 2.1. Then, in Section 2.2, we explain the scenario analyzed in this study.

2.1 The set-up

We consider a fifth year student of an engineering school, whom we will name *Antoine*, who is renting an apartment where he needs to pay for the electricity

consumption. Antoine would like to lower his electricity bill and he decided to achieve this by reconfiguring the power meter in his apartment. In this study, Antoine plays a role of an *attacker* and his opponent, i.e., a *defender*, is the electricity provider. The meter under study is equipped with an optical port that allows a user to connect to the meter using an optical probe. Illustrations of a meter (Fig. 8) and an optical probe (Fig. 9) can be found in Appendix A

To describe possible ways in which Antoine may reprogram the meter, we use attack–defense trees [18]. An attack–defense tree (ADTree) is a graphical security model representing how an attacker may attack the analyzed system and how the defender may counter these attacks. The main building blocks of an ADTree are the *labels* of its nodes, two types of *refinement* – `OR` and `AND` – and the *countermeasures*. The labels describe the goals that the attacker or the defender want to achieve. The refinements allow to decompose these goals into simpler subgoals (refined nodes) and basic actions (leaves). To achieve the goal of an `OR` node, the corresponding agent needs to achieve the goal of at least one of its child nodes. The goal of an `AND` node is achieved if the goals of all of its children are achieved. A countermeasure attached to a node of one agent represents how the node's goal may be overcome by the other agent. This means that a node of the attacker can be countered by a node of the defender, which in turn can be counterattacked by a node of the attacker, and so on. In addition, a countermeasure can also be decomposed using `OR` and `AND` refinements.

ADTrees extend the well-known model of attack trees [32] widely used in industry [8,9,28]. The starting point of our analysis was the scenario and the attack tree described in Section 2.3 of [28]. We complemented this tree with additional attacks, and added possible countermeasures that we identified based on [7,26], and [34]. The resulting ADTree contains 68 nodes, 5 repeated basic actions of the attacker and 3 repeated basic actions of the defender. Following the convention established in [6] and [21], we assume that nodes having the same label represent exactly the same instance of an action. They thus model that executing some action may contribute to several, different attacks or defenses.

2.2 The scenario

In order to reconfigure his power meter via optical port, Antoine has to have physical access to the power meter and reconfigure it using appropriate software tools. Since the power meter is located in the apartment where Antoine lives, we assume that accessing the power meter is a basic action, i.e., the corresponding node is not refined. In order to reconfigure the power meter with the help of software, we have identified the following three sub-scenarios that Antoine can follow, taking into account his knowledge and capabilities:

The *do it yourself* approach – Antoine reconfigures the meter himself by using unauthorized software tools (Fig. 2, 10, 11, 12, 3),
The *social engineering* approach – Antoine social engineers a technician employed by the electricity provider to reconfigure the power meter for him using authorized software tools (Fig. 4),

The *get employed* approach – Antoine gets employed by the electricity provider as a field technician to gain access to the authorized tools and to be able to reconfigure the meter himself (Fig. 5).

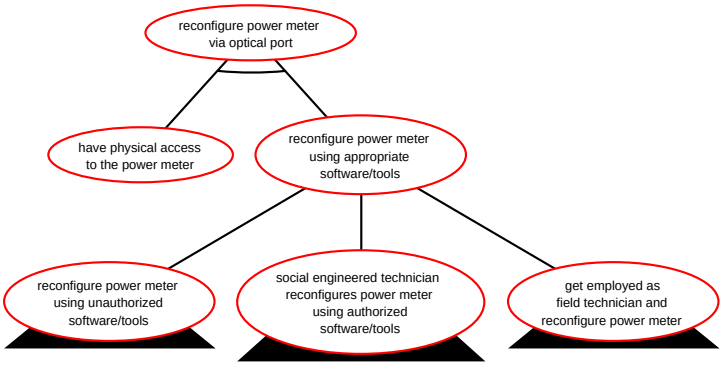

Fig. 1: How to reconfigure the power meter – a high level view

This high-level view on the analyzed scenario is presented by the tree from Fig. 1, where the black triangles illustrate subtrees presented in further figures. In this work, we use the standard graphical notation for ADTrees: red and green nodes represent attacker's and defender's goals, respectively, and an arc is used to denote the `AND` refinement. In the rest of this section, we detail the three approaches considered by Antoine.

The *do it yourself* approach

To reconfigure the power meter by himself, Antoine needs to obtain unauthorized software and tools, use optical probe to establish connection with the meter via its optical port, and finally reconfigure the meter using unauthorized software. He can find and download unauthorized software from the Internet. As for the optical probe, he can buy it or make it himself. The corresponding tree is given in Fig. 2.

Establishing connection to the meter via its optical port might be secured by password authentication. Also, independently of whether a password-based protection is implemented or not, an authentication could be required before the power consumption configuration can be modified. These two possible countermeasures are depicted with the green nodes in Fig. 2.

If the connection to the power meter was protected by a password, Antoine could still reach his goal if he was able to authenticate using the correct credentials. To do so, he would need to obtain the credentials and enter them to the power meter while authenticating, as visualized in Fig. 10, placed in Appendix B due to space restrictions. The power meter credentials could be obtained by

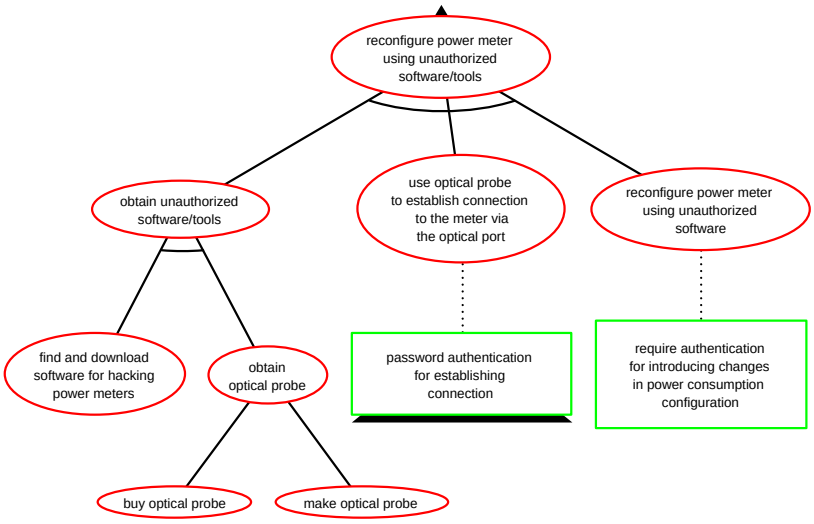

Fig. 2: The *do it yourself* approach

– exploiting the hardware components of the power meter (Fig. 11),
– performing a brute force attack (Fig. 12), or
– social engineering a technician working for the power provider (Fig. 3).

Extracting credentials from the power meter hardware components, illustrated in Fig. 11, can be achieved in two ways: either by extracting them from a data dump or by spying on communication between the hardware components. To extract the credentials from the data dump, the dump needs to be made, the location where encrypted credentials are placed in the dump needs to be identified, and finally the credentials need to be extracted from the encrypted dump. To extract the credentials from the communication between the hardware components, the communication needs to be monitored and the credentials need to be intercepted. In this study, we assume that during the communication between the hardware components, the data are sent unencrypted.

A brute force attack is illustrated in Fig. 12. It makes use of software for hacking power meters (in our scenario, this is exactly the same software as the one used by the attacker to reconfigure power meter). An off-line brute force attack using tools like `Ophcrack` [29], `John the Ripper` [30], or `hashcat` [15], can be prevented if a strong password is used. To make an on-line cracking impossible, the number of possible invalid authentication attempts could be limited.

Finally, credentials could also be obtained by social engineering a technician, as depicted in Fig. 3. To do so, a suitable technician would need to be selected and social engineered. A social engineering attack would require to assemble background information on employees of the energy provider and select one who would fall into the social engineering attack to reveal the credentials. Antoine could obtain the background knowledge on employees by searching on the In-

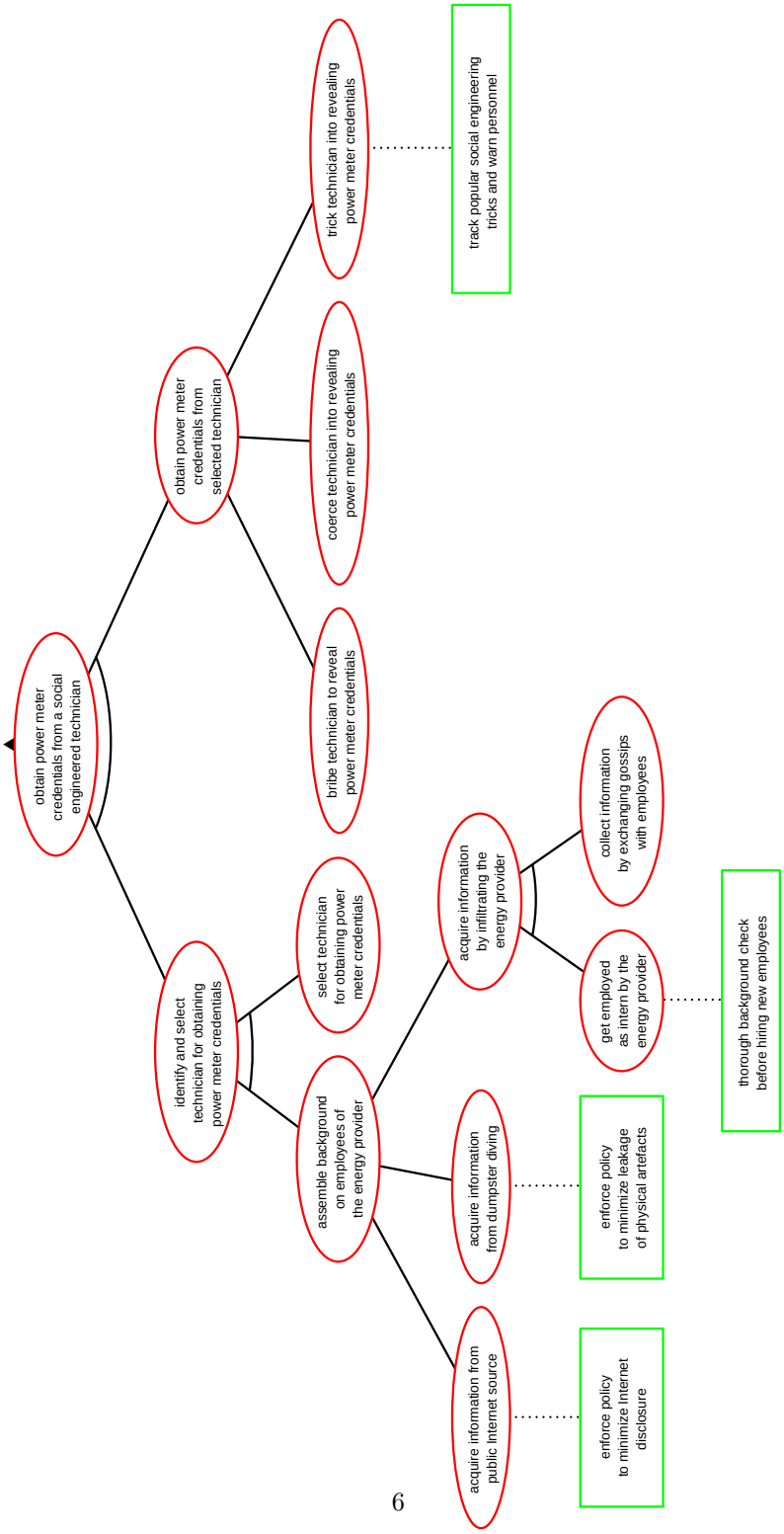

Fig. 3: Obtain credentials by social engineering a technician

ternet, diving into dumpster and looking for relevant documents and physical artefacts, or by infiltrating the energy provider. To infiltrate the energy provider, Antoine could get hired as an intern student and then collect information by exchanging gossips with the company employees. The following policies could be enforced by the company to prevent access to the background information about its employees:

- a policy to minimize the Internet disclosure,
- a policy to minimize the leakage of physical documents and artefacts,
- a policy of performing thorough background check before hiring new employees.

Once the right social engineering target is selected, the attack itself consists in bribing, coercing or tricking the technician so that they reveal the power meter credentials. The tricking attack could be prevented by a security training during which the personnel is made aware of popular social engineering tricks.

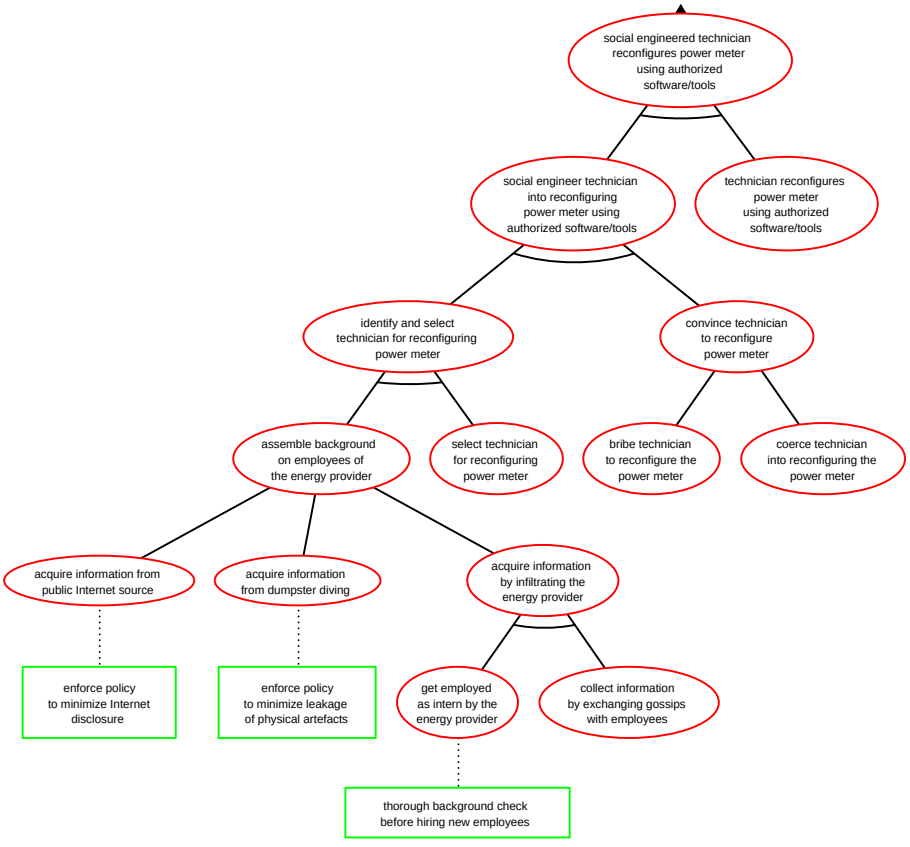

Fig. 4: The *social engineering* approach

The *Social engineering* approach

Instead of attacking by himself, Antoine can social engineer a technician, so that they reconfigure the power meter for him, as modeled in Fig. 4. To perform the social engineering, a suitable technician who would reconfigure the power meter needs to be identified and Antoine needs to convince them to reconfigure the meter. Identification of the suitable social engineering target is performed in exactly the same way as in the *do it yourself* approach by assembling relevant background knowledge on employees. Once identified, the technician who will reconfigure the power meter is selected. To persuade the technician to reconfigure the power meter, Antoine can bribe or coerce them.

The *get employed* approach

Antoine can also get hired by the power provider company to be officially able to reconfigure power meters. To do so, he needs to get employed as a field technician and then reconfigure his power meter using authorized software provided by the company to its technicians. Performing thorough background check on future employees would mitigate this attack, as it was the case in the two previous approaches. The get employed attack is illustrated in Fig. 5.

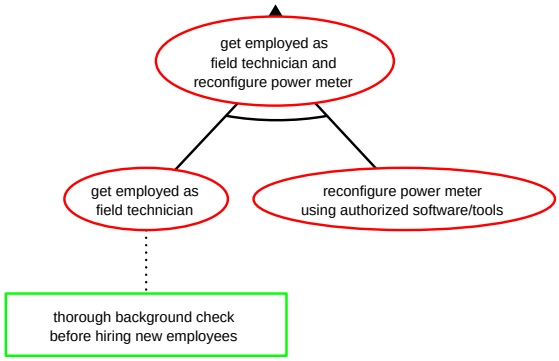

Fig. 5: The *get employed* approach

3 Quantitative analysis of the tampering scenario

The first objective of this case study is to analyze the scenario described in Section 2. This includes enumeration of all possible attacks, identification of those that are optimal from the point of view of the attacker, as well as pinpointing the countermeasures that offer the best protection to the analyzed system. A plethora of algorithms exist to perform quantitative analysis of ADTrees, e.g., [3,13,18,19,20,21]. However, some of them may produce incorrect results when applied to ADTrees where an action may contribute to several attacks. In this study, we thus focus on the methods developed in [10,21] which have been especially developed to deal with ADTrees containing such repeated basic actions, and an adaptation of the methods of [20] applicable to such trees.

In what follows, we will use the notion of an *attack* which we define as *a minimal set of basic actions of the attacker the execution of which achieves the*

goal of the root node of the ADTree, under some fixed behavior of the defender. By *defender's strategy*, we understand a *set of countermeasures* that the defender can implement to secure the system.

In Section 3.1, we explain the problems that are of interest for our study. Then, in Section 3.2, we provide clear definitions of the parameters that we have used. Finally, in Section 3.3, we present the input values that we have used and we explain how they have been estimated.

3.1 The problems of interest

The three types of optimization problems that we tackle in this study are:

- selection of attacks optimal wrt one parameter,
- selection of attacks optimal wrt several parameters,
- selection of the defender's strategy optimal from the point of view of their resources and objective.

The parameters (often also called *attributes* in the context of ADTrees [19]) used in this work are *cost*, *time*, *probability*, and *special skills*. Before we present their meaning in Section 3.2, we provide generic, i.e., parameter independent, explanation of the problems that we address.

First, we are interested in selecting the *best* attacks, taking only one parameter into account. Here 'best' means optimal wrt the parameter considered. An optimal attack may thus be a one that minimizes *time* or *cost*, or maximizes *probability* of success. The selection of optimal attacks is performed according to the method proposed in [21], which ensures that the computations on ADTrees with repeated labels yield correct results, contrary to the classical bottom-up approach, used for instance in [19]. This method also allows to rank possible attacks, and return k most optimal ones, where $k \in \mathbb{N}$ can be specified by the user.

Second, it is well known that considering one parameter at a time may not be sufficient to decide which attack is the best (or the worst, if the scenario is analyzed from the defender's point of view). A possible approach to address this problem is to identify *Pareto optimal attacks*, i.e., attacks that are undominated by others, when several parameters are regarded simultaneously. In our study, we employ the framework developed in [10] to select the Pareto optimal attacks extracted from an ADTree with repeated labels. The solution proposed in [10] relies on so called *Pareto attribute domains* that allow for combining a number of paramaters into a single one, and therefore make it possible to perform a multi-objective analysis using the same tools as in the case of a single parameter.

Finally, the solutions to the two problems presented above depend on which countermeasures have been implemented to protect the system. Instead of considering all possible cases, it is thus wise to determine a strategy which from the point of view of the defender is optimal. In an ideal world, the best would be to implement all possible countermeasures. However, in real-life, this is not possible, because the budget of the defender is limited. Also, it does not make sense to implement countermeasures which do not add more security wrt the

ones already in place. To select an optimal strategy of the defender, we use the method relying on integer linear programing proposed in [20], adapted to ADTrees with repeated labels. It allows for selecting one of the two criteria – *maximizing the number of countered attacks* (coverage problem) or *maximizing the necessary investment of the attacker* (investment problem) – and selecting the defender's strategy which optimizes the chosen criterion and is compatible with the defender's budget.

3.2 The parameters used

We now describe the parameters that we have used in our study to address the problems presented in Section 3.1. We give an intuitive meaning to each parameter, provide the set of values that they can attain, and explain how the parameter's value of an attack is computed from the values corresponding to the attack's components.

Cost $(\min, +)$. The first parameter of interest is the *monetary investment* necessary to implement an attack (or a defender's strategy). To express it, we use non-negative real numbers representing the necessary investment in euro. The actions that are too expensive to be executed are assigned the value of $+\infty$. When optimizing the cost parameter, we are interested in attacks (or set of defenses) yielding minimal monetary investment for the attacker (resp. defender). Thus, to compute the cost of an attack (resp. set of defenses), we *sum* the values of the basic actions composing it. To select the best attack (resp. defender's strategy), we then choose the one with *minimal* value.

Time $(\min, \max)$. Since Antoine would like to lower his electricity bill as soon as possible, the *time that an attack would take* is an important parameter to consider. The following scale is used to express time values:

- *Instantaneous* (0): can be performed by the actor in less than a minute.
- *Quick* (10): can be performed by the actor in less than an hour, but not less than a minute.
- *Slow* (10^2): can be performed by the actor in less than a week, but not less than an hour.
- *Very slow* (10^3): can be performed by the actor in less than six months, but not less than a week.
- *Extremely slow* (10^4): can be performed by the actor within a human lifetime, but not less than six months.
- *Impossible* ($+\infty$): not doable within a human lifetime.

Since this scale is discrete, it is reasonable to assume that the time necessary to perform an attack is the *maximum* value over the time values of its composing actions. As in the case of cost, we are interested in *minimizing* the time necessary to attack the system, thus we select the attack which requires *minimal* time.

Success probability $(\max, \times)$. Attacks that are very cheap or very fast are useless if their probability of succeeding is negligible. Here, we are thus interested in what is the probability that, if executed, an attack will be successful. The probability of an action is a value from the interval $[0, 1]$, and the probability of an attack is computed by multiplying the probability values of its components. When optimizing, we select the attack with the *maximal* probability of success, because this is the attack that a reasonable attacker would prefer.

The remaining three parameters assess the level of special skills – cybersecurity, technical, and social – that is necessary to be able to perform an action successfully. In all tree cases, the skill level necessary to perform an attack is defined as the *maximum* amongst the skill levels necessary to perform its components. By optimal, we mean an attack requiring *minimal* skill level.

Cybersecurity skills level $(\min, \max)$. Some of the actions considered in our scenario may require specific *expertise regarding cybersecurity*. We distinguish five levels of such expertise:

- *None* (0): no cybersecurity-related skills required.
- *Basic* (1): requires basic cybersecurity knowledge and skills.
- *Advanced* (2): requires employing advanced cybersecurity-related skills, e.g., executing a man in the middle attack on a protocol.
- *Expert* (3): requires employing cybersecurity-related skills available to few experts, e.g., return-oriented programming or fault attack on AES.
- *Impossible* $(+\infty)$: beyond the known capability of today's human beings.

Technical skills level $(\min, \max)$. Similarly to cybersecurity skills, some actions may require some *technical expertise*. Here again, we distinguish five levels:

- *None* (0): no technical skills required.
- *Basic* (1): requires basic technical skills, e.g., finding information online.
- *Advanced* (2): requires advanced technical skills, available for graduates of technical vocational schools.
- *Expert* (3): requires technical skills available to experienced engineers.
- *Impossible* $(+\infty)$: beyond the known capability of today's human beings.

Social skills level $(\min, \max)$. Finally, since some attacks in our scenario rely strongly on social engineering, we are also interested in *social skills* necessary to perform the considered actions. The five levels of social skills are defined as follows:

- *None* (0): does not involve social interactions.
- *Basic* (1): requires basic social interactions, e.g., obtaining information via a conversation.
- *Advanced* (2): requires convincing or tricking someone into doing something they would not do otherwise.
- *Expert* (3): requires convincing or tricking someone into doing something punishable by law.
- *Impossible* $(+\infty)$: beyond the known capability of today's human beings.

3.3 Estimation of input values

We solve the optimization problems presented in Section 3.1 using frameworks developed in [21,20] and [10]. The underlying algorithms take as input an ADTree as well as parameter values assigned to its basic actions. Estimation of these values is not an easy task. It requires a thorough understanding of

- the parameters employed,
- the meaning of the basic actions present in the tree,
- the attacker's and defender's profiles and knowledge.

Despite a substantial effort made by both academic and industrial security modeling community, there still does not exist a clean and sound methodology allowing to determine reliable input values. In practice, this estimation is very subjective, and may rely on historical data, statistics, information gathered from surveys or open sources such as Internet, or the modeler's expertise.

Table 1: Cost of basic actions of the defender

Basic action	***Cost***
$\mathtt{d}_1$ = enforce policy of using strong passwords	11600
$\mathtt{d}_2$ = enforce policy to minimize Internet disclosure	9600
$\mathtt{d}_3$ = enforce policy to minimize leakage of physical artefacts	9600
$\mathtt{d}_4$ = limit the number of possible invalid authentication attempts	11600
$\mathtt{d}_5$ = password authentication for establishing connection	13600
$\mathtt{d}_6$ = require authentication for introducing changes in power consumption configuration	13600
$\mathtt{d}_7$ = thorough background check before hiring new employees	320
$\mathtt{d}_8$ = track popular social engineering attacks and warn personnel	1500

The values of basic actions of the attacker that we have used in this study are given in Table 7, available in Appendix C. They represent a consensus reached as a result of the following procedure. Seven independent participants, whose profiles correspond to the expertise of Antoine, were involved in the values' estimation. The participants were given a document describing the scenario and the ADTree from Section 2. They had access to the Internet and relevant materials, including [7,28,34]. Each participant estimated the values for all six parameters at every basic action present in the tree. Unsurprisingly, some of the values were not consistent amongst different participants. A semi-automatic procedure has thus been used to extract a single value for each parameter at every basic action:

- for the parameters different than *probability*: if all (but one) amongst the seven values were the same, this value was retained,
- for the *probability* parameter, a simple average over seven values was computed,
- for the cases that do not fall into any of the above items, the retained value is the result of a discussion between the two authors of this paper,

– finally, in the case of strong disagreement, the author of the analyzed ADTree who, amongst the seven participants, knows the best the optical meter technology, had the decisive power.

The estimation of values took one hour to each participant, on average. The consensus discussions lasted for 3 hours, in total.

4 Optimal strategies for the attacker and the defender

We now present the results of the power meter tampering scenario analysis. We begin, in Section 4.1, by determining sets of countermeasures that the defender can implement under specified budget and that are optimal wrt a given criterion (coverage or attacker's investment). For some of these sets, we then perform a *what-if* analysis: if a given strategy of the defender is implemented, what are the attacks optimal wrt one (Section 4.2) or many (Section 4.3) parameters? Our objective is to verify whether an attacker having a profile of Antoine would be able to launch a successful attack on its power meter. Due to space restrictions, this section summarizes some main observations drawn from our analysis. The files presenting the raw data and all the obtained results are available at `https://people.irisa.fr/Wojciech.Widel/studies/meter_study.zip`.

Performing this analysis is a laborious task, because the underlying algorithms are complex [10,20,21] and our tree is too big to be analyzed manually. To obtain the results presented in this section, we have used the `OSEAD` tool that supports the techniques proposed in [10,20], and [21]. We postpone the technical description of `OSEAD` to Section 5.

4.1 Selection of optimal sets of countermeasures

The choice of an optimal strategy for the defender depends on the budget that they have at their disposal, and on the optimization problem of interest. In our study, we consider a small, local electricity provider, and we thus analyze three possible values for the defender's budget: 20000, 30000, and 40000 euros. Table 2 presents optimal strategies for a defender interested in maximizing the number of prevented attacks (coverage problem) and another one focused on maximizing the necessary investment of the attacker necessary to achieve his objective (investment problem). They have been obtained using the `OSEAD` tool.

Requiring authentication for introducing changes in power consumption configuration ($\mathtt{d}_6$) and *performing thorough background check before hiring new employees* ($\mathtt{d}_7$) is an optimal strategy for a defender interested in covering a maximal number of possible attacks and having the budget of 20000 euros. We denote this strategy by D_1. Unfortunately, the budget of 20000 is not sufficient to obtain any optimal strategy for the defender in the case of the attacker's investment problem. Indeed, for every possible strategy of the defender, that can be implemented with 20000 euros, there still exists a possible attack requiring no monetary investment from the attacker, i.e., having cost of 0 euros.

The other two strategies that we consider are D_2 which corresponds to D_1 extended with the action of *enforcing policy to minimize Internet disclosure* ($\mathtt{d}_2$), and D_3 consisting of *enforcing policy to minimize Internet disclosure* ($\mathtt{d}_2$), *enforcing policy to minimize leakage of physical artefacts* ($\mathtt{d}_3$), *performing thorough background check before hiring new employees* ($\mathtt{d}_7$), and *tracking popular social engineering attacks and warning personnel* ($\mathtt{d}_8$). These strategies are optimal for a defender having 30000 euros, and interested in the coverage problem (D_2) and the attacker's investment problem (D_3), respectively.

Finally, a defender having 40000 euros is able to fully secure the analyzed system, by implementing countermeasures $\mathtt{d}_2, \mathtt{d}_3, \mathtt{d}_6$, and $\mathtt{d}_7$. Due to space restrictions, we refer the reader to Table 1 for their meaning.

Table 2: Optimal strategies of the defender

	Coverage problem		**Investment problem**	
Defender's budget	Optimal strategy	Prevented/ /preventable	Optimal strategy	Necessary attacker's investment
20000	$D_1 = \{\mathtt{d}_6, \mathtt{d}_7\}$	29/33	$\emptyset$	0
30000	$D_2 = \{\mathtt{d}_2, \mathtt{d}_6, \mathtt{d}_7\}$	31/33	$D_3 = \{\mathtt{d}_2, \mathtt{d}_3, \mathtt{d}_7, \mathtt{d}_8\}$	14
40000	$\{\mathtt{d}_2, \mathtt{d}_3, \mathtt{d}_6, \mathtt{d}_7\}$	33/33	$\{\mathtt{d}_2, \mathtt{d}_3, \mathtt{d}_6, \mathtt{d}_7\}$	$+\infty$

For the rest of our study, we retain the strategies D_1, D_2, and D_3 and look for optimal attacks in the case when one of these strategies is implemented by the defender.

4.2 Attacks optimizing single parameter

For determining attacks optimal wrt to one parameter, `OSEAD` first extracts the attacks from the model (attacks in the sense of *set semantics* defined in [21]), and then computes the values of the parameter corresponding to each of them. In total, there are 33 attacks in the studied scenario, and their list is available at `https://people.irisa.fr/Wojciech.Widel/studies/meter_attacks.txt`. Whether an attack is successful or not depends on the countermeasures that are implemented by the defender. The attacks of interest for us are those that are not countered by at least one of the three defender's strategies D_1, D_2 or D_3. There are twelve such attacks, and they are presented in Table 3.

By analyzing the data gathered in Table 3, we notice that if the defender decides to implement one of the strategies D_1 or D_2, Antoine will be able to succeed only by executing some of the attacks from the *social engineering* approach. If the strategy D_3 is implemented, then the only possible attacks are those from the *do it yourself* approach.

Once the values corresponding to the attacks are obtained, `OSEAD` returns the optimal ones. We list them in Table 4. This table can be used to check whether an attacker of interest would be able to launch a successful attack. We recall that Antoine is a fifth year student of an engineering school. We assume that he has

Table 3: Some of the attacks available to Antoine

Attacking approach: ***do it yourself*** **(Y);** ***social engineering*** **(S);** ***get employed*** **(E)**	S	S	S	S	Y	Y	Y	Y	Y	Y	Y	Y
Basic action	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}
acquire information from dumpster diving		✓		✓								
acquire information from public Internet source	✓		✓									
bribe technician to reconfigure the power meter			✓	✓								
bribe technician to reveal power meter credentials												
buy optical probe									✓	✓	✓	✓
coerce technician into reconfiguring the power meter	✓	✓										
coerce technician into revealing power meter credentials												
collect information by exchanging gossips with employees												
enter power meter credentials					✓		✓	✓		✓	✓	✓
extract credentials							✓	✓				
find and download software for hacking power meters					✓	✓	✓	✓	✓	✓	✓	✓
get employed as field technician												
get employed as intern by the energy provider												
have physical access to the power meter	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
intercept credentials								✓		✓		
locate encrypted credentials in the dump							✓			✓		
make optical probe					✓	✓	✓	✓				
make the data dump from hardware component							✓			✓		
monitor communication between hardware components								✓			✓	
perform brute force attack					✓							✓
provide power meter credentials												
reconfigure power meter using authorized software/tools												
reconfigure power meter using unauthorized software					✓	✓	✓	✓	✓	✓	✓	✓
select technician for obtaining power meter credentials												
select technician for reconfiguring power meter	✓	✓	✓	✓								
technician reconfigures power meter using authorized software/tools	✓	✓	✓	✓								
trick technician into revealing power meter credentials												
use optical probe to establish connection to the meter via the optical port					✓	✓	✓	✓	✓	✓	✓	✓
Defender's strategy under which the attack is successful	D_1	D_1, D_2	D_1	D_1, D_2	D_3	D_3	D_3	D_3	D_3	D_3	D_3	D_3

advanced technical skills, but he has only basic knowledge of cybersecurity. As most of students, he is not rich, but he can manage his time availability freely.

On the one hand, since the cost aspect is of the highest priority for Antoine, he would analyze the attacks optimal wrt to this parameter first, see the *Cost* column in Table 4. The preference is given to attack A_2 which consists of *having physical access to the power meter*, *acquiring information from dumpster diving*, *selecting technician for reconfiguring power meter*, *coercing technician into reconfiguring power meter* and the *technician reconfiguring power meter using authorized software/tools*. While this attack is optimal from the point of view of *cost* and all the three *skills levels* under strategies D_1 and D_2, it would require from Antoine to force someone to perform an action punishable by law. Also, A_2 is not prevented by the strategy D_3. Indeed, implementation of D_3 counters all the attacks from the *social engineering* approach.

On the other hand, D_3 does not secure the meter from any attack in the *do it yourself* approach. An interesting attack within this approach is A_6, consisting of *having physical access to the power meter*, *making optical probe*, *finding and downloading software for hacking power meters*, *using optical probe to establish connection to the meter via the optical port*, and *reconfiguring power meter using unauthorized software*. Note that A_6 corresponds to the profile of Antoine, from the point of view of his resources and skills. Its only drawback is that its probability of success is quite low – only 0.26, as can be seen in Table 5.

Thanks to Table 4, we can also study the impact of the implemented countermeasures on the attacks available to the attacker. Upgrading the system's protection from D_1 do D_2 (by *enforcing policy to minimize Internet disclosure*) at the cost of 9600 euros (see Table 1) is not worthwhile if the defender considers cheap attacks to be the most tempting for the attacker – the attack A_2 achieves the root goal under both strategies D_1 and D_2. However, if the defender aims at making the attacker less likely to succeed, then this investment is beneficial, as it lowers the attacker's success probability from 0.41 (for attack A_3 which would not work under D_2) to 0.10 (for A_4 that still works when D_2 is implemented).

Table 4: Attacks optimal wrt a single parameter and their values

	Attacks optimal wrt					
Defender's strategy	***Cost***	***Time***	***Prob***	***Cyber***	***Tech***	***Social***
D_1	A_1, A_2	A_1, A_3	A_3	A_1, A_2, A_3, A_4	A_2, A_4	A_1, A_2, A_3, A_4
Optimal value	0	100	0.41	0	0	3
D_2	A_2	A_2, A_4	A_4	A_2, A_4	A_2, A_4	A_2, A_4
Optimal value	0	1000	0.10	0	0	3
D_3	A_5, A_6, A_7, A_8	$A_5 - A_{12}$	A_9	A_5, A_6, A_9, A_{12}	A_5, A_6, A_8, A_9, A_{11}, A_{12}	$A_5 - A_{12}$
Optimal value	14	100	0.64	1	2	0

4.3 Attacks optimizing several parameters

Unfortunately, for every attack listed in Table 4, i.e., optimal wrt to some given parameter, there always exists another one that is better from the point of view of another parameter. To overcome this problem, we are now looking for Pareto optimal attacks, i.e., attacks that are not dominated by another one, while taking all six parameters into account simultaneously.

The Pareto optimal attacks are presented in Table 5, along with the values corresponding to their execution. Observe that under strategies D_1 or D_2, all of the attacks available to Antoine are Pareto optimal, including the attack A_2 discussed in the previous section. If the strategy D_3 is implemented by the defender, there exist eight possible attacks that achieve the root goal, but only two of them are Pareto optimal, namely A_6 and A_9. It is crucial to notice that A_9 is a very interesting attack. It is almost the same as A_6, except that it involves *buying optical probe* instead of *making it*. Attack A_9 is optimal wrt to all parameters, except *cost*. However, when checking its *cost* value, one realizes that the investment necessary to perform it (71.2 euros) would probably be acceptable for Antoine. The greatest advantage of A_9 is that its success probability (0.64) is significantly higher than that of A_6 (0.26).

Table 5: Pareto optimal attacks and their values for: *cost* (c), *time* (t), *prob* (pb), *cyber skills* (cs), *tech. skills* (ts), and *social skills* (ss)

Defender's strategy	**Pareto optimal attacks**	**Values** (c, t, pb, cs, ts, ss)
D_1	A_1	(0, 100, 0.24, 0, 1, 3)
	A_2	(0, 1000, 0.06, 0, 0, 3)
	A_3	(500, 100, 0.41, 0, 1, 3)
	A_4	(500, 1000, 0.10, 0, 0, 3)
D_2	A_2	(0, 1000, 0.06, 0, 0, 3)
	A_4	(500, 1000, 0.10, 0, 0, 3)
D_3	A_6	(14.0, 100, 0.26, 1, 2, 0)
	A_9	(71.2, 100, 0.64, 1, 2, 0)

The importance of the multi-parameter analysis is further illustrated by two facts. First, securing the system in a way that maximizes the necessary investment of the attacker, by implementing D_3, not only leaves the system vulnerable to more attacks than it is the case for the coverage problem (eight attacks versus two or four, see last row of Table 3), but also allows the attacker to execute attack A_9, which has a high probability of succeeding. Second, when the defender implements strategy D_3, the attack A_6 is among the cheapest ones, and the attack A_9 is the optimal one wrt the probability. When we analyze the scenario taking only one of these parameters into consideration, we overlook one of these two attacks. But both of them are Pareto optimal, and as such, both can be considered equally appealing for the attacker.

5 The OSEAD tool

We now present the OSEAD tool – *Optimal Strategies Extractor for Attack–Defense trees* – that we used for performing analysis of Section 4. We describe its features in Section 5.1, its performance in Section 5.2, and give some implementation details in Section 5.3.

5.1 OSEAD from the user's perspective

OSEAD allows its users to benefit from the theoretical developments of [10,20,21] in a simple and intuitive way. Users operate the tool in a step-by-step manner, via a graphical interface illustrated in Fig. 6. The first step is to provide a file storing structure of ADTree of interest, which is an XML file produced by ADTool [14], well-known software for creating ADTrees. Furthermore, should the user want to analyze an attack tree created with the help of ATCalc [1] or ATE [2], the output files of these tools can be easily transformed into an ADTool-like XML file with the help of ATTop [23].

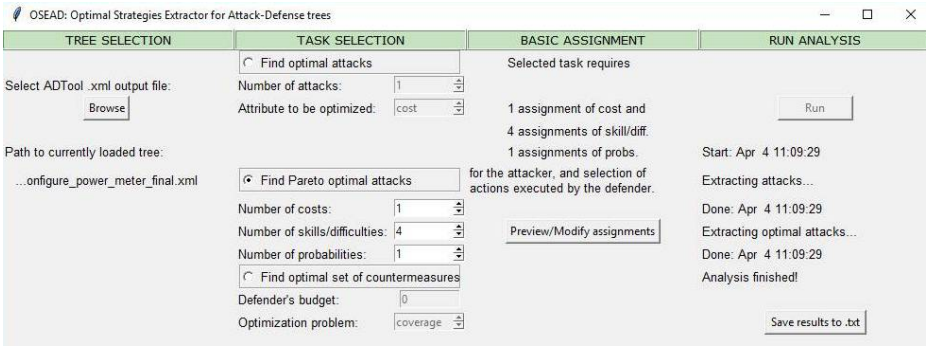

Fig. 6: OSEAD's main user interface

Once the tree is provided, users select the problem of interest, which is extraction of attacks that optimize a single parameter (tab *Find optimal attacks* in 6), attacks that are Pareto optimal (tab *Find Pareto optimal attacks*), or an optimal strategy of the defender (tab *Find optimal set of countermeasures*). The last step preceding the actual analysis is the assignment of values of parameters of interest to the basic actions in the tree. The values can be entered manually, imported from an XML file produced by ADTool, or loaded from a TXT file produced by OSEAD, as visualized in Fig. 13 in Appendix C. With all the inputs provided, OSEAD solves the optimization problem specified by the user. The results obtained can be exported to a TXT file.

5.2 OSEAD's performance

To solve the optimization problems, OSEAD first extracts attacks, as well as defender's strategies in the case of optimal set of countermeasures selection, from

ADTree. Theoretically, this means that either a *set semantics* [6] or a *defense semantics* [20] of the tree is computed. The worst case size of both of these semantics is exponential in the number of basic actions of the tree. Another possible bottleneck in the process of determining an optimal strategy for the defender is solving the integer linear programming problem (ILP). `OSEAD` employs free ILP solver `lp_solve` [5] to achieve this task.

Table 6: `OSEAD`'s runtime for determining Pareto optimal attacks

Name of file storing tree structure	Name of file storing basic assignment	Number of attacks	Number of Pareto optimal attacks	Runtime in seconds
tree03	*tree03_1_cost*	640	2	1
tree10	*tree10_1_cost*	14336	3	438
tree12	*tree12_5_costs*	2436	63	11
tree29	*tree29_5_costs*	640	304	1
tree30	*tree30_5_costs*	704	184	1
tree32	*tree32_5_costs*	832	378	1

In practice, `OSEAD` performs well. Each of the problems considered in Section 4 was solved in time not exceeding one second, on a Windows machine running Intel Core i7-5600U CPU at 2.60 GHz dual core with 16 GB of RAM. We have also tested `OSEAD`'s performance on trees having structure significantly more complex than the one studied in the previous sections, i.e., on trees encoding hundreds and thousands of attacks. Using trees available at `https://github.com/wwidel/pareto-tests/tree/master/trees` and basic assignments given at `https://github.com/wwidel/pareto-tests/tree/master/assignments`, we have measured the time `OSEAD` needs to determine Pareto optimal attacks. An excerpt from the tests' results is presented in Table 6.

5.3 Implementation details

`OSEAD`'s computation engine and its user interface have been implemented in Python. Its architecture is depicted in Fig. 7. The implementation model consists of the *Tree Model* (storing the tree structure), the *Attribute Domain* (defining operations to be used when determining optimal attacks, e.g., $(\min, +)$ in the case of cost), the *ILP Problem* (derived from the *Tree Model*, using *defense semantics*, and storing the matrix of the selected optimization problem) and the *Basic Assignment* (storing values of parameters assigned to the basic actions).

`OSEAD` is open source and it runs on all main platforms. The version for Windows can be downloaded from `https://people.irisa.fr/Wojciech.Widel/suftware/osead.zip`. Using `OSEAD` on other platforms requires installing the *adtrees* Python package [31].

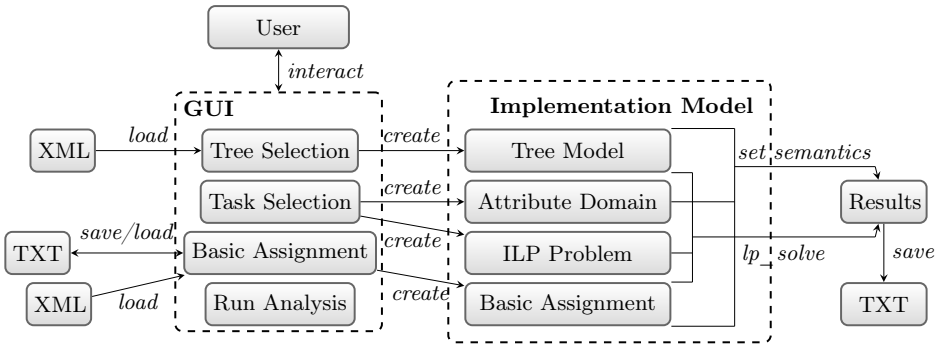

Fig. 7: An overview of the OSEAD architecture

6 Conclusion

In this paper, we analyzed a real-life scenario of tampering with a power meter, using an ADTree. In addition to the actual evaluation of the power meters' security, this study allowed us to validate the quantitative analysis methods that we have recently developed for ADTrees with repeated labels [10,20,21]. To facilitate and automate their usage, we have implemented the OSEAD tool described in Section 5.

We took great care so that our model and analysis are as unbiased and impartial as possible. The tree was created by crossing several industrial and academic sources, and the input values estimation was performed by independent participants with various cultural background, from Estonia, France, Poland, and Russia.

As discussed in Section 4.3, we were able to confirm the intuitive conjecture about the practical importance of the multi-parameter analysis. We note that, despite the fact that the algorithms implemented in OSEAD are highly complex, the tool performs extremely well when applied to trees encoding hundreds of attacks, and reasonably well in the case of trees with up to several thousands of attacks.

This study corroborates practical usefulness of ADTrees in security and risk analysis. However, solutions for some pragmatic issues still need to be found. The bottleneck of our study was the attribution of parameter values to basic actions. While for some parameters, e.g., *cost*, finding an accurate estimate is easy (nowadays, it suffices to search on the Internet), for some other, e.g., *success probability*, this task is much more difficult, if not impossible. More research and practical investigation is definitely necessary before a reliable methodology for the estimation of values for basic actions can be proposed. In parallel, one could investigate how sensible the quantitative analysis methods proposed in [10,20,21] are wrt input values, i.e., how do errors propagate through the computations, if the input values used are not exact. In the light of its very promising efficiency, it seems that OSEAD will greatly facilitate performing this task, and this is the issue on which we will concentrate in the nearest future.

Acknowledgement.

We would like to thank the students who contributed to this study by performing (far from being trivial) task of estimating parameter values for basic actions: Jean-Loup Hatchikian-Houdot (INSA Rennes, France), Pille Pullonen (Cybernetica AS, Estonia), Artur Riazanov (Saint Petersburg Department of V.A. Steklov Institute of Mathematics of the Russian Academy of Sciences, Russia), Peter Smirnov (Saint Petersburg State University, Russia), and Aivo Toots (Cybernetica AS, Estonia).

References

1. Arnold, F., Belinfante, A., van der Berg, F., Guck, D., Stoelinga, M.: DFTCalc: A Tool for Efficient Fault Tree Analysis. In: SAFECOMP. LNCS, vol. 8153, pp. 293–301. Springer (2013)
2. Aslanyan, Z.: TREsPASS toolbox: Attack Tree Evaluator (2016), `https://vimeo.com/145070436`, presentation of a tool developed for the EU project TREsPASS
3. Aslanyan, Z., Nielson, F.: Pareto Efficient Solutions of Attack–Defense Trees. In: POST'15. LNCS, vol. 9036, pp. 95–114. Springer (2015)
4. Bagnato, A., Kordy, B., Meland, P.H., Schweitzer, P.: Attribute Decoration of Attack–Defense Trees. IJSSE 3(2), 1–35 (2012)
5. Berkelaar, M., Eikland, K., Notebaert, P.: lp_solve: Open source (Mixed-Integer) Linear Programming system (2005), `http://lpsolve.sourceforge.net/5.5/`, version 5.5.2.5, dated September 24, 2016
6. Bossuat, A., Kordy, B.: Evil Twins: Handling Repetitions in Attack–Defense Trees – A Survival Guide. In: GraMSec 2017. LNCS, vol. 10744, pp. 17–37. Springer (2018)
7. Carpenter, M.: Advanced Metering Infrastructure Attack Methodology (2009), `http://docshare.tips/ami-attack-methodology_5849023fb6d87fd2bb8b4806.html`, accessed: 2019-02-20
8. Dürrwang, J., Braun, J., Rumez, M., Kriesten, R., Pretschner, A.: Enhancement of automotive penetration testing with threat analyses results. SAE Int. J. Cybersecurity 1, 91–112 (11 2018), `https://doi.org/10.4271/11-01-02-0005`
9. EAC Advisory Board and Standards Board: Election Operations Assessment – Threat Trees and Matrices and Threat Instance Risk Analyzer (TIRA) (2009), `https://www.eac.gov/assets/1/28/Election_Operations_Assessment_Threat_Trees_and_Matrices_and_Threat_Instance_Risk_Analyzer_(TIRA).pdf`
10. Fila, B., Wideł, W.: Efficient attack–defense tree analysis using Pareto attribute domains (2019), under submission
11. Fraile, M., Ford, M., Gadyatskaya, O., Kumar, R., Stoelinga, M., Trujillo-Rasua, R.: Using Attack–Defense Trees to Analyze Threats and Countermeasures in an ATM: A Case Study. In: PoEM. LNBIP, vol. 267, pp. 326–334. Springer (2016)
12. Frederic Byumvuhore: FEATURED: REG steps up crackdown on electricity theft (2019), `https://www.newtimes.co.rw/news/featured-reg-steps-crackdown-electricity-theft`, accessed on: 2019-04-05
13. Gadyatskaya, O., Hansen, R.R., Larsen, K.G., Legay, A., Olesen, M.C., Poulsen, D.B.: Modelling Attack–defense Trees Using Timed Automata. In: FORMATS. LNCS, vol. 9884, pp. 35–50. Springer (2016)

14. Gadyatskaya, O., Jhawar, R., Kordy, P., Lounis, K., Mauw, S., Trujillo-Rasua, R.: Attack Trees for Practical Security Assessment: Ranking of Attack Scenarios with ADTool 2.0. In: QEST. LNCS, vol. 9826, pp. 159–162. Springer (2016)
15. hashcat: https://hashcat.net/hashcat/ (2016), accessed on: 2019-03-27
16. Kelly-Detwiler, P.: Electricity Theft: A Bigger Issue Than You Think (2013), https://www.forbes.com/sites/peterdetwiler/2013/04/23/electricity-theft-a-bigger-issue-than-you-think/#5475872972ef, accessed: 2019-02-20
17. Kiana Wilburg: GPL lost US$450M in 19 years to electricity theft, poor networks (2018), https://www.kaieteurnewsonline.com/2018/12/10/gpl-lost-us450m-in-19-years-to-electricity-theft-poor-networks/, accessed on: 2019-04-05
18. Kordy, B., Mauw, S., Radomirovic, S., Schweitzer, P.: Attack–defense trees. Journal of Logic and Computation 24(1), 55 – 87 (2014)
19. Kordy, B., Mauw, S., Schweitzer, P.: Quantitative Questions on Attack–Defense Trees. In: ICISC. LNCS, vol. 7839, pp. 49–64. Springer (2012)
20. Kordy, B., Wideł, W.: How well can I secure my system? In: iFM'17. LNCS, vol. 10510, pp. 332–347. Springer (2017)
21. Kordy, B., Wideł, W.: On quantitative analysis of attack–defense trees with repeated labels. In: POST. LNCS, vol. 10804, pp. 325–346. Springer (2018)
22. Krebs, B.: FBI: Smart Meter Hacks Likely to Spread (2012), https://krebsonsecurity.com/2012/04/fbi-smart-meter-hacks-likely-to-spread/, accessed: 2019-02-20
23. Kumar, R., Schivo, S., Ruijters, E., Yildiz, B.M., Huistra, D., Brandt, J., Rensink, A., Stoelinga, M.: Effective Analysis of Attack Trees: a Model-Driven Approach. In: Russo, A., Andy Schürr, A. (eds.) FASE. LNCS, vol. 10802, pp. 56–73. Springer (2018)
24. LLC, N.G.: World Loses $89.3 Billion to Electricity Theft Annually, $58.7 Billion in Emerging Markets (2014), https://www.prnewswire.com/news-releases/world-loses-893-billion-to-electricity-theft-annually-587-billion-in-emerging-markets-300006515.html, accessed: 2019-02-20
25. Mauw, S., Oostdijk, M.: Foundations of Attack Trees. In: ICISC 2005. LNCS, vol. 3935, pp. 186–198. Springer (2005)
26. McCullough, J.: Deterrent and detection of smart grid meter tampering and theft of electricity, water, or gas (2010), https://www.elstersolutions.com/assets/downloads/WP42-1010A.pdf, accessed: 2019-02-20
27. Ms. Smith: FBI Warns Smart Meter Hacking May Cost Utility Companies $400 Million A Year (2012), https://www.csoonline.com/article/2222111/fbi-warns-smart-meter-hacking-may-cost-utility-companies--400-million-a-year.html, accessed on: 2019-04-05
28. National Electric Sector Cybersecurity Organization Resource (NESCOR): Analysis of selected electric sector high risk failure scenarios, version 2.0 (2015), http://smartgrid.epri.com/doc/NESCOR%20Detailed%20Failure%20Scenarios%20v2.pdf
29. Ophcrack: http://ophcrack.sourceforge.net/ (2016), accessed on: 2017-03-17
30. Ophcrack: https://www.openwall.com/john/ (2016), accessed on: 2019-03-27
31. adtrees Python package: https://github.com/wwidel/adtrees (2019), accessed on: 2019-04-05
32. Schneier, B.: Attack Trees: Modeling Security Threats. Dr. Dobb's Journal of Software Tools 24(12), 21–29 (1999)
33. T&D World: India To Spend $21.6 Billion On Smart Grid Infrastructure By 2025 (2015), https://www.tdworld.com/smart-grid/india-spend-216-billion-smart-grid-infrastructure-2025, accessed on: 2019-04-05

34. Weber, D.C.: Optiguard: A Smart Meter Assessment Toolkit (2012), https://media.blackhat.com/bh-us-12/Briefings/Weber/BH_US_12_Weber_Eye_of_the_Meter_WP.pdf, accessed: 2019-02-20

A Optical power meter

Figure 8 shows a power meter with an optical port.

Fig. 8: A power meter with an optical port (source: https://nl.wikipedia.org/wiki/IEC_62056)

Figure 9 illustrates how to connect to a power meter using an optical probe.

Fig. 9: Optical probe connected to the power meter (source: https://www.aliexpress.com/item/China-Manufacturer-DHL-free-Shipping-electricity-optical-meter-reading/32455842504.html?spm=2114.10010108.100009.1.6810cc24soIZC4&gps-id=pcDetailLeftTopSell&scm=1007.13482.95643.0&scm_id=1007.13482.95643.0&scm-url=1007.13482.95643.0&pvid=65873a85-f01b-4876-970d-b58b38041880)

B ADTree

The XML file, compatible with `ADTool` and `OSEAD`, containing the entire ADTree for tampering with the power meter is available at `https://people.irisa.fr/Wojciech.Widel/studies/meter_study.zip`.

The subtree for how to overcome a password-based authentication is given in Fig. 10

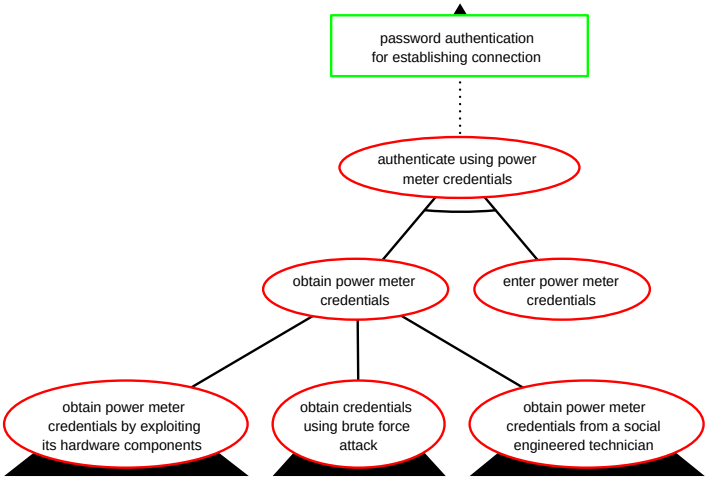

Fig. 10: Overcoming the password-based authentication

ADTree for obtaining credentials from hardware components is given in Fig. 11.

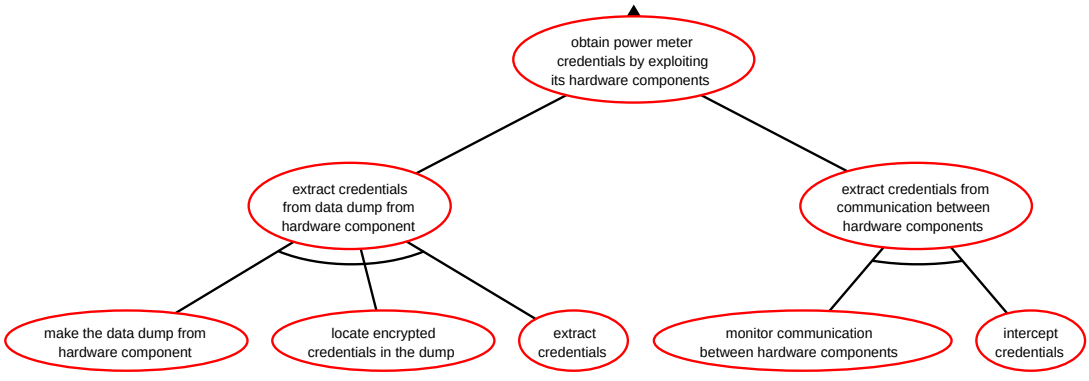

Fig. 11: Obtain credentials from hardware components

The tree in Fig. 12 shows how to obtain credentials using a brute force attack.

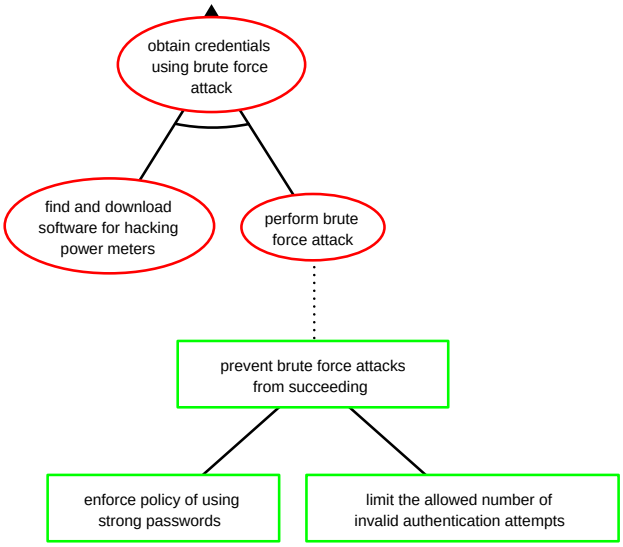

Fig. 12: Obtain credentials by brute force attack

C Input values for basic actions of the attacker

Table 7 gathers input values of the six parameters used in our study, and Fig. 13 gives an overview of the input management in `OSEAD`. The TXT files with our input values, usable by `OSEAD`, are available at `https://people.irisa.fr/Wojciech.Widel/studies/meter_study.zip`.

Table 7: Parameter values for basic actions of the attacker

Basic action	***Cost***	***Time***	***Prob***	***Cyber***	***Tech***	***Social***
acquire information from dumpster diving	0	1000	0.2	0	0	0
acquire information from public Internet source	0	100	0.79	0	1	0
bribe technician to reconfigure the power meter	500	10	0.52	0	0	3
bribe technician to reveal power meter credentials	300	10	0.5	0	0	2
buy optical probe	71.2	100	1	0	1	0
coerce technician into reconfiguring the power meter	0	100	0.3	0	0	3
coerce technician into revealing power meter credentials	0	100	0.33	0	0	3
collect information by exchanging gossips with employees	0	1000	0.46	0	0	1
enter power meter credentials	0	0	0.99	0	0	0
extract credentials	0	10	0.56	0	1	0
make the data dump from hardware component	0	100	0.73	1	3	0
find and download software for hacking power meters	0	10	0.9	1	1	0
get employed as field technician	0	1000	0.48	0	2	1
get employed as intern by the energy provider	0	1000	0.52	0	1	1
have physical access to the power meter	0	0	1	0	0	0
intercept credentials	0	0	0.62	2	1	0
locate encrypted credentials in the dump	0	100	0.6	2	2	0
make optical probe	14	100	0.41	0	2	0
monitor communication between hardware components	0	100	0.5	1	2	0
perform brute force attack	0	100	0.65	1	2	0
provide power meter credentials	0	0	1	0	0	0
reconfigure power meter using authorized software/tools	0	10	0.94	0	1	0
reconfigure power meter using unauthorized software	0	10	0.75	0	2	0
select technician for obtaining power meter credentials	0	100	1	0	0	0
select technician for reconfiguring power meter	0	100	1	0	0	0
technician reconfigures power meter using authorized software/tools	0	10	1	0	0	0
trick technician into revealing power meter credentials	0	100	0.24	0	0	2
use optical probe to establish connection to the meter via the optical port	0	10	0.95	0	1	0

Fig. 13: Input management in OSEAD