Optimizing System Architecture Cost and Security Countermeasures

Sahar Berro, Ludovic Apvrille, and Guillaume Duc

LTCI, Télécom Paris, Institut polytechnique de Paris, France
firstname.lastname@telecom-paris.fr

Abstract. The design of an embedded system is built on a trade-off between its performance and its cost. Nowadays, the security threats that target most of the embedded systems introduce a new factor in this trade-off: the security level of the system. So system architects must consider, during the design, the different attacks that target the system and the possible countermeasures, and their costs. In this article, we present a methodology to help designers explore different countermeasures and evaluate their impact on the cost of the architecture and the probability of success of an adversary. This methodology is based on extended and formalized Attack-Defense Trees that allow to assess the impact of countermeasures on system components and attacks. We use propagation rules to characterize a main attack from its different steps, and we formalize the trade-off between security and cost by an optimization problem between attack probability and total architecture cost.

Keywords: Attack-defense tree · security of embedded system · countermeasures.

1 Introduction

System-level embedded system design — e.g. with model based approaches — is a common practice that unfortunately frequently ignores cyber-security aspects. Thus, usual system-level approaches rather target safety and performance aspects [24, 11]. Security can be important (i) by itself because of e.g. privacy concerns and (ii) because it could impact safety and performance [1].

Model-based approaches usually rely on security requirements diagrams and on attack (defense) trees to capture security aspects [22, 19, 3]. Attack trees structure attacks in a way that intends to help designers select the relevant countermeasures that can prevent the root attacks of trees. Yet, countermeasures are strongly linked to system architectural aspects for two reasons. First, a countermeasure may be implemented only in a given architecture. For instance, a powerful crypto accelerator can surely not be implemented in a very low power device. A similar example can be found in the EVITA architecture where Hardware Security Modules are added to Electronic Control Units: since some of them are expected to be of very low cost, different versions of the HSM (light,

medium, full) have been defined. Second, a countermeasure depends on the already existing components of an architecture. For instance, if an attack consists in exploiting a known CVE for a given Operating System O1, replacing O1 by another operating system O2 depends on the fact that O2 exists for the processors of the target architecture. Finally, identifying the right countermeasures is an optimization process that should take into account both the interest in resisting to attacks and the cost it implies on the architecture.

SysML-Sec [4, 2] has already been proposed to handle in the same framework security requirements, attack trees and architecture design (in particular platform cost). In particular, previous contributions have shown the relations between attacks and architectural elements. This paper enhances previous contributions with the definitions on how an optimal architecture could be found according to attacks likeliness and countermeasure cost. For this, the paper introduces a new approach based on Attack-Defense Trees (ADT) enhanced with formalized links to architectural elements.

The rest of the paper is organized as follows. Section 2 details the previous works related to our work. Section 3 presents the context of our work. Section 4 describes our main contributions. Section 5 explains some choices we made. Section 6 presents the application of our methodology to a case study. Section 7 discusses future directions for our work. Section 8 concludes our work.

2 State of the art

The basic formalism of Attack Trees (AT) was first introduced by Schneier in [22].

Mauw and Oostdjik in [20] proposed and alternative formalism for the one presented by Schneier. Their contribution consists in associating a set of *mincuts* to the root attack.

Contributions of Jürgenson et al in [13] improved the semantics of Mauw and Oostdjik in [20] and introduced an exact and consistent set of computational rules to determine attackers' expected outcomes. In particular, their contribution takes into account parameters in leaves: attack cost, probability of success, expected penalty on the attacker if the attack was unsuccessful and the expected penalty on the system in case the attack was successful. They also used a global parameter "gains" to evaluate the benefit of the attacker in case it could achieve the root attack (and not only elementary attacks).

Audinot et al in [7] showed the complexity results for three notions of the soundness of an attack tree: admissibility, consistency and completeness. They also show how the tree operators influence complexity results.

Other proposals suggest to make a combine use of attack trees and fault trees. Steiner and Liggesmeyer [23] have extended the qualitative and quantitative safety analysis to take in consideration the influence of security problems on the safety of a system. They introduced "SECFT" (that stands for Security Event Component Fault Tree), a component fault tree that contains both safety and security events. They conduct their security analysis using likelihood of

occurrence for security events (attacks) and the probability of occurrence for faults.

However, none of these contributions takes into account defense nor protection mechanisms. Bistarelly et al [8] introduce the notion of defense tree DT. They proposed to model attackers and defenders using game theory in order to find the set of countermeasures that has the most effective cost. To evaluate the return on attack ROA, they used the expected gain of a successful attack, its cost and the augmented cost caused by the use of a countermeasure (revised cost). To evaluate the return on security investment ROI, they used the annual financial loss caused by a threat, the annual number of occurrence of a threat, the cost of the countermeasure and the impact of threats on these countermeasures. Kordy et al in [16] gave a formalism where attack trees used in [14], [9], [21] and defense trees introduced by Bistarelly are covered in a single framework: attack-defense tree ADT. They developped in [15] ADTool that supports quantiative and qualitative analysis of ADTs and their instances.

Edge et al [9] proposed to use protection trees along with attack trees to determine the protections needed for computer networks in homeland security. They also defined a set of rules to evaluate metrics associated with the leaves of both trees. For attack trees, they used (only) the probability of success and the cost of attacks in order to evaluate the impacts on the risk of a system.

Ji et al [12] analyzed the performance of ADTs by assigning additional parameters to each attack and countermeasure nodes: cost, impact on the system and success probability for attacks, and cost for countermeasures. These parameters are then used for evaluating the revised impact on the system and the revised cost of the attack after the countermeasure is deployed. Then, they analyze the performance of the ADT by considering the ROI and the ROA.

Fraile et al [10] presented a case study where they modeled and analyzed threats on ATMs using ADTs. They used occurrence probability for each attack node to derive the likelihood for the overall root ATM attacks. The probability of occurrence of a refined attack node was calculated by taking in consideration whether or not this node is counteracted by a defense mechanism.

Kordy and Widel [18] defined a set of rules to conduct a quantitative analysis on an ADT with repeated labels. As metrics, they used the minimal cost for executing an attack, the maximal damage caused by an attack, the minimal skill level required to execute an attack, the minimal number of experts needed to mount an attack and the satisfiability for the defender.

Finally, some contributions rely on attack trees to conduct only a security analysis and thus determine the impact of attacks on a given system. Some others rely on fault and attack trees in order to evaluate how attacks may cause faults and impact system safety. Some other proposals propose to rely on ADTs to represent how attacks are countered and thus they can analyze the interactions between the gain of a successful attack and the security investment cost. A state of the art on attack and defense modeling approaches for security is presented in [17] and a survey on graphical security models that gives an overview of their developpements, complexity analysis and application has been provided in [5].

However none of these approaches have considered system architecture while evaluating attacks, countermeasures and their costs. The rest of the paper explains how we can efficiently iterate between attacks, countermeasures and system architectures in order to setup a "good" solution.

3 Context

Designing a system architecture is an incremental process. Each time a new component (it could be a new hardware or software component, or simply a change of configuration of an existing component), new attacks may be possible on the system, and attacks that have been previously captured in attack trees may not be possible anymore. To counter all possible attacks, security engineers employ a set of countermeasures. These countermeasures could be implemented by modifying existing software, by introducing new software components, by adding hardware components, or by changing the configuration of hardware components, taking us back to identify new attacks and so on. Obviously, the use of countermeasures is expected to decrease the impact of at least one attack, yet it could introduce new attacks and increase the cost of the platform.

The design of a system usually involves trade-offs between safety, security, performance and cost. The approach proposed in this paper aims at helping system designers to perform trade-offs between security level and platform cost, based on enhanced and formalized ADTs.

4 Contributions

4.1 Main definitions: adversary, attack, countermeasure

Since attacks are related to the architectural components, our contribution first consists in introducing a new formalism for ADTs, where not only attacks and countermeasures are modeled, but also the architecture components. In other terms, components where attacks may occur and countermeasures components are represented in the ADT in order to support our optimization based on attacks - countermeasures - architecture.

In order to conduct a security analysis using an ADT, first metrics are associated to the child nodes of the trees. Then, a set of propagation rules are defined. Finally, based on a bottom-up approach, metrics are evaluated from the leaf nodes to the root attack using these rules.

Attackers and attacks

Definition 1 *A **Malicious attacker** M is a 2-uple (R_M, E_M) where:*

1. *R_M represent the set of resources of the adversary, i.e. hardware equipment, financial and time resources...*

2. *E_M represents the expertise of the adversary. E_M can be expressed as a value $E_M \in [0..1]$ (for instance) or as a label $E_M \in \{beginner, intermediate, expert\}$ (as long as the set of labels is ordered).*

The hardware equipment includes computation capabilities, electronic equipment such as soldering stations, boards, microscopes, lasers... Time represents the manpower and the time frame that the attacker has. For simplicity reasons, R could also be simply represented as a sum of all resources, thus simply representing the financial capabilities of an attacker.

Definition 2 *An **attack** A is a 3-uple (l, R_A, E_A) where:*

1. *l is a labeling function.*
2. *R_A represents the minimal set of resources that are necessary to perform this attack.*
3. *E_A represents the minimal expertise necessary for an adversary to perform this attack.*

Countermeasure The objective of a countermeasure is to make an attack more difficult, i.e. to increase its cost or the necessary expertise.

Definition 3 *A **countermeasure** CM is a 4-uple $(l, C_{CM|SA}, I = \{(A, R, E)\}, N = \{A\})$ where:*

1. *l is a labeling function.*
2. *$C_{CM|SA}$ represents the cost of this countermeasure in a given architecture.*
3. *I is a set of 3-uple (A, R, E) that represents for a given attack A how it impacts its resources R and its expertise E. By definition, a countermeasure must increase either R or E of A. R and E may depend on the cost of the countermeasure.*
4. *N is a set of attacks that must be performed in addition to the existing attacks to circumvent the countermeasure. The parameters of these attacks may depend on the cost of the countermeasure.*

Countermeasures either impact existing attacks by making them more difficult to realize (e.g. a masking countermeasure against a side-channel attack makes this attack more difficult by requiring a second-order attack) or introduce new attacks that must be performed in addition to existing ones to achieve the same result (e.g. a bus probing attack can be prevented by using encryption mechanisms; but this countermeasure can be circumvent by retrieving the key used to encrypt data transiting on the bus; at the end, the adversary must perform both attacks to retrieve the data in cleartext).

The cost of a given countermeasure may vary (for instance a countermeasure may have different levels of security but at different costs). Thus, the impact of a countermeasure may vary depending on the cost.

4.2 Success of attacks

The success of an attack with regards to an attacker depends on the resources and expertise of an attacker. It can be defined as follows:

Definition 4 *The function* ***success*** $(attack, attacker) \rightarrow true/false$ *is a function that returns true if the attacker has the necessary resources and expertise to perform the attack, false otherwise:* $success(A, M) = R_M \geq R_A \wedge E_M \geq E_A$.

Considering a set (or population) of malicious attackers $\mathcal{M}$, it is now possible to define the probability of success of an attack.

Definition 5 *The* ***probability of success of an attack*** *A by a set of malicious attackers* $\mathcal{M}$ *is defined as:*
$\mathcal{P}_{A/\mathcal{M}} = \frac{|\{M \in \mathcal{M} | success(A,M)=true\}|}{|\mathcal{M}|}$

We can note that the success of a given attack with a given attacker and so the probability of success of a given attack depend on the countermeasures implemented on the system because they impact the resources or the level of expertise needed to carry off the attack or add additional attack steps that must be performed.

4.3 Attack Tree

An Attack Tree (AT) is a conceptual multi-level diagram used to describe the security of a system. It contains a root attack, intermediate and leaf attacks as well as some nodes of operators (usually AND and OR). This multi-layered approach allows to represent the different possible attack scenarios to reach the root one.

An Attack-Defense Tree (ADT) adds defense mechanisms to the set of attacks.

In this paper, we consider the following definition for our ADTs:

Definition 6 *An Attack-Defense Tree for a given system architecture SA* ***ADT-SA*** *is a 5-uple*
$(OP, A_{root}, ATK, D, COMP)$ *where:*

1. *OP is the set of the different operators.*
2. *A_{root} is the root attack i.e. the main goal of the adversary.*
3. *ATK is the set of attack i.e. intermediate and leaf attacks.*
4. *D is the set of defense mechanisms i.e. the available countermeasures used to counter attacks.*
5. *COMP is the set of all the hardware and software components of the system architecture.*

Note that ATK, D and $COMP$ are not independent since a countermeasure may be implemented only if specific components are part of the system architecture. Similarly, a component or a set of components may introduce new attacks.

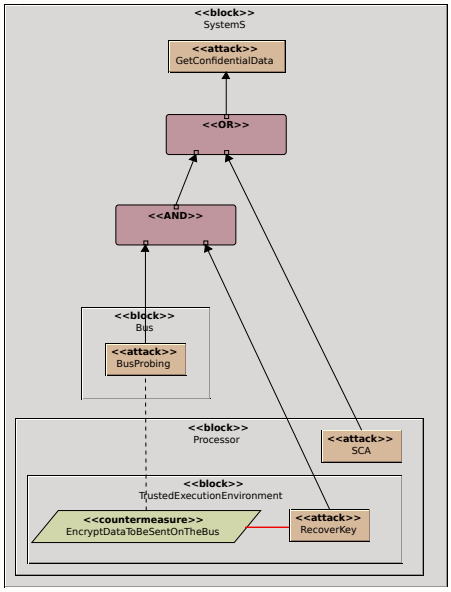

Fig. 1. Example of an ADT-SA

4.4 Example

To illustrate the concept of ADT-SA, we consider a toy embedded system with a processor, an external RAM (not embedded inside the processor), a bus between the processor and the RAM. An operating system and an application that performs cryptographic operations (such as encryption) using a secret key run on this system.

Figure 1 shows one ADT-SA of the considered system. For simplicity reasons, we did not represent the whole system architecture. By the way, an ADT-SA does not require to model all system architecture but only the part related to the ADT-SA under study.

In the considered ADT-SA, the main objective of the attacker (attack *getConfidentialData*) is to retrieve confidential data (in our case, the secret key manipulated by the application). To perform this attack, the attacker can either (due to the OR operator) perform a bus probing attack (attack *ProbingTheBus* that targets the component *Bus*) or a side-channel attack against the *Processor* during the manipulation of the secret key.

A countermeasure, *EncryptDataOnTheBus* (which consists in using a Trusted Execution Environment to encrypt all the data that are sent outside of the processor, for instance to the memory), can be used to prevent the bus probing attack. With this countermeasure, the adversary can probe the bus but it will only retrieve encrypted data that are useless unless it recovers the key used to

encrypt the bus. So, this countermeasure adds an additional attack step (*RecoverKey*) that must be performed by the adversary, in addition to retrieving the encrypted data by bus probing, to achieve its goal.

4.5 Analysis

Describing attack scenarios in a graphical way is not the only objective of attack trees. They can indeed be used to perform a quantitative analysis with respect to given metrics called *attributes*. Attributes are metrics assigned to the basic actions in the tree (leaf nodes) and are used in a bottom-up evaluation. The bottom-up procedure consists in propagating attributes from leaf nodes to the root of the tree by applying appropriate operations to the intermediate operators connecting different intermediate attack steps. In this paper we will call these operations *propagation rules*.

Metrics and Propagation rules Our approach considers the following metrics:

- Attacks: minimal resources R and E necessary to perform the attack.
- Countermeasure: cost $C_{CM|SA}$ to implement the countermeasure in the corresponding system architecture, the increase of the minimal resources ΔR and of the minimal expertise ΔE required induced by the addition of this countermeasure in the corresponding system architecture (these increases can be directly caused by the countermeasure or indirectly by adding some attack steps needed to circumvent the countermeasure).
- System architecture: cost C_{arch}.

We now show how these metrics can be propagated from leaf nodes to higher nodes in the ADT-SA when using conjunction (denoted by *AND*) and disjunction (denoted by *OR*) operators.

1. **Minimal resources and expertise**
   Let us suppose that we have a root attack A_{root} (*AttackRoot*) with the following characteristics $(l_{root}, R_{root}, E_{root})$ and p other refined attacks such as $\forall i \in \{1, ..., p\}$, A_i (*Attacki*) is characterized by (l_i, R_i, E_i). Let us suppose as well that a malicious attacker M with (R_M, E_M) wants to perform A_{root} in the system.
   Figure 2 represents the *AND* operator where p attack steps are required to be performed in order to achieve the root attack. Figure 3 illustrates the *OR* where at least one attack step among p elementary attacks need to be achieved in order to realize the attack root.
   - **AND operator** The attacker M has to perform all the p attacks in order to realize A_{root}. Thus, its resources R_M must be at least equal to the sum of the resources needed to perform each attacks (we suppose here that all the attacks are totally independent, this supposition is discussed later). In other words, to succeed and realize its goal, its resources must be

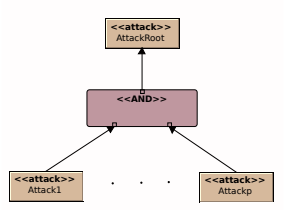

Fig. 2. AND operator

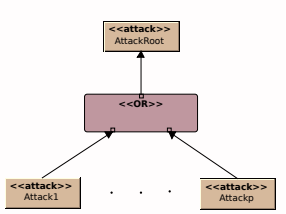

Fig. 3. OR operator

greater or equal to $\sum_{i=1}^{p} R_i$. Regarding the expertise, to mount A_{root}, the level of expertise of the attacker must be greater or equal to the maximum level of expertise required by the different attack steps: $\max_{i=1}^{p} E_i$.
Therefore, $R_{root} = \sum_{i=1}^{p} R_i$ and $E_{root} = \max_{i=1}^{p} E_i$.

- **OR operator** To achieve A_{root}, the attacker M, characterized by (R_M, E_M), has to perform at least one of the p attacks.
If $\forall i \in \{1, ..., p\}$, $R_i > R_M$ or $E_i > E_M$, the attacker M will not be able to achieve A_{root}. Otherwise, among the attacks it has a sufficient level of expertise to perform ($Q = \{i | E_i \leq E_M\}$), it will choose the one which requires the minimum resources (A_k such as $\forall i \in Q, R_k \leq R_i$). In this case, $R_{root} = R_k$ and $E_{root} = E_k$.
We note that in this situation, the propagation rule depends on the attacker.

2. **Countermeasures and the increase of the minimal resources and level of expertise**
A countermeasure makes an attack more difficult to be mounted due to the fact that it introduces either new attack steps (which indirectly increase the resources and expertise needed to perform the attack thanks to the propagation rules defined previously) or directly increase the resources and expertise of the original attack.
Let us consider Figure 4. For the sake of simplicity, we will use an arbitrary attack-defense tree with 2 leaf attacks connected with an *OR* to illustrate the evaluation of the impact of countermeasures. Note that the *OR* operator that connects A_1 and A_2 could be replaced by an *AND* operator, provided that we then use the corresponding propagation rules.
In this illustration, to achieve A_{root}, an attacker needs to perform either A_1 or A_2. The red segment between a countermeasure CM and an attack A, means that A is introduced by CM. The arc between CM_3 **and** CM_4 means that they are both needed to counter A_2 (they could have been represented by only one countermeasure that includes both). A_1 is prevented by CM_1 **or** CM_2. We will study how these two combinations of countermeasures (OR and AND) impact the attack.
In this paper, we will suppose that all the countermeasures added to the system are independent.

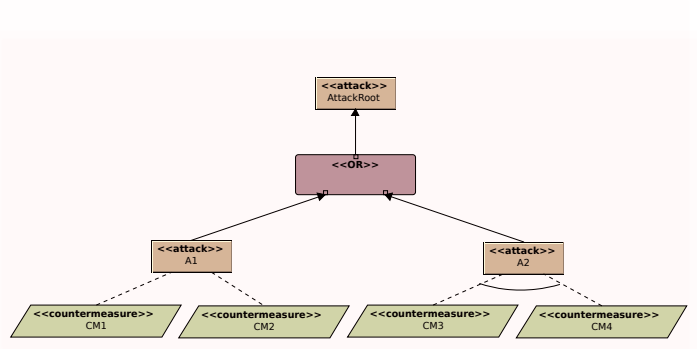

Fig. 4. Example of the impact of countermeasures

- **OR operator** Let us suppose that the attacker chooses to perform A_1 to achieve A_{root} and that the countermeasure CM_1 (respectively CM_2) adds an additional attack step A_{CM_1} (respectively A_{CM_2}). Figure 5 shows the developed tree with these two new attack steps (we did not represent A_2 countermeasures on the graph to make the tree less complicated).
  Since A_1 is prevented by CM_1 or by CM_2, to reach its goal, the attacker M has to bypass both countermeasures by performing the two new attacks A_{CM_1} and A_{CM_2}. Thus, to perform A_{root}, M must perform A_1, A_{CM_1} and A_{CM_2}, instead of just A_1.
  Therefore, according to the propagation rules defined above, $R_{root} = R_1 + R_{CM_1} + R_{CM_2}$ and $E_{root} = \max(E_1, E_{CM_1}, E_{CM_2})$.
- **AND operator** Let us now suppose that the attacker M chooses to perform A_2 to achieve A_{root} and that the countermeasure CM_3 (respectively CM_4) adds an additional attack step A_{CM_3} (respectively A_{CM_4}). Figure 6 shows the developed tree with these two new attack steps (we did not represent A_1 countermeasures on the graph to make the tree less complicated).
  A_2 is prevented by CM_3 and CM_4 (both of them are required). Therefore, the attacker M has to bypass at least one of these countermeasures to reach its goal and achieve A_{root}. To do so, it must perform either A_{CM_3} or A_{CM_4}. As described before when the adversary can choose, among the attacks it has the level of expertise required, it will choose the one which requires the less amount of resource.

3. **System architecture** System architecture is a description of the design of this system. It represents a plan of the interrelations between its existing or future components and subsystems. Initially this representation is general. As we go deeper into details, it can be refined to more concrete description. To prevent attacks, security experts use countermeasures. Hence, the cost of

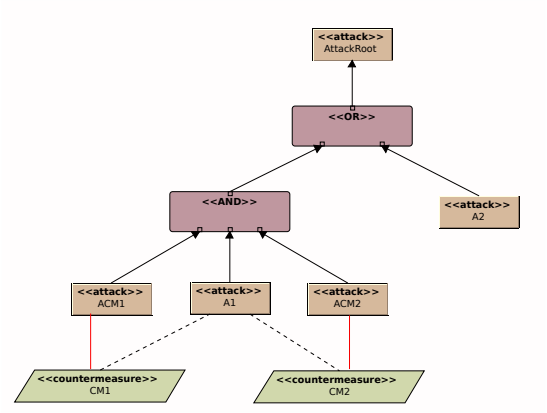

Fig. 5. Example of an OR operator applied on countermeasures

the whole system architecture C_{arch} is equal to the sum of the cost of all its components $C_{Total_{Comp}}$ and the cost of all its countermeasures $C_{Total_{CM|SA}}$:

$$C_{arch} = C_{Total_{Comp}} + C_{Total_{CM|SA}} \tag{1}$$

However, using countermeasures may have some secondary effects: it may introduce new attacks and increase the cost of the platform. Moreover, system engineer usually fix a budget that should be respected while constructing the system. Furthermore, designing a system require adjustments between performance, security/safety and cost. Thus, we need to find the set of countermeasures such that (i) equation 1 respects the pre-defined budget and (ii) leads to the minimal probability of success for a given attack. We will describe this problematic with the following optimization problem.

4.6 The optimization problem

Definition 7 *Let* $\boldsymbol{S}$ *be the set of the possible countermeasures used in the system and let* $\mathcal{P}(\mathbf{S})$ *be the powerset of* S*.*

Definition 8 *We define the function* $C_{arch} : \mathcal{P}(S) \rightarrow \mathbb{R}^+$ *as* $C_{arch}(x) = C_{Total_{Comp_x}} + C_{Total_x|SA}$*, where:*

- $C_{Total_{Comp_x}}$ *represents the cost of all the components of the system. That cost depends on the countermeasures that are implemented (x), since a countermeasure may need specific hardware components, e.g a powerful processor, an hardware cryptographic accelerator, etc.*
- $C_{Total_x|SA}$ *represents the cost of all the countermeasures used in system architecture SA.*

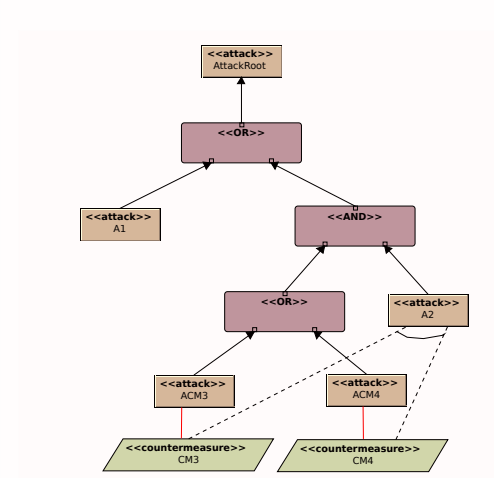

Fig. 6. Example of an AND operator applied on countermeasures

This function represents the total cost of the system depending on the implemented countermeasures (possibly none).

Definition 9 *We define the function* $P : \mathcal{P}(S) \rightarrow [0,1]$ *as* $P(x) = \mathcal{P}_{A_{root|x}/\mathcal{M}}$ *i.e. the probability of success of the root attack given a population of malicious attackers* $\mathcal{M}$ *and the countermeasures* x *implemented on the system.*

The problem is to find the set of countermeasures x that minimizes $P(x)$ with the constraint $C_{arch}(x) \leq B$, where B is the pre-defined budget fixed by the system engineers. This formalizes the trade-off that the system architect has to perform between the cost of the architecture and the probability of success of an attack. A solution of this optimization problem can be found by manually or automatically exploring all the possible sets of countermeasures or by more advanced optimization algorithms.

If the system is targeted by several root attacks, the definition of P can be refined by including the probability of success of the different root attacks weighted by their impact (for instance in terms of financial losses).

5 Discussions

5.1 Independent attack steps

As explained before, some attacks require several steps to be achieve. In other words, to perform a root or an intermediate attack A, an adversary may need to mount other q elementary attacks $A_1, ..., A_q$. For now, we have supposed that these elementary attacks are independent which means for instance that

the resources needed to perform all the elementary attacks are the sum of the resources needed by each elementary attack.

However, two elementary attacks A_1 and A_2 can both require the same expensive piece of equipment (for instance an oscilloscope). In this case, the resources needed to perform the two attacks is not the sum (we do not require two oscilloscopes, one is sufficient). However, when two attacks are not independent, we have chosen to represent them as three independent attacks that all have to be performed to succeed, one (A_c) that contains the shared resources (for instance the oscilloscope) and the two other (A'_1 and A'_2) that embeds the resources specific for A_1 and A_2.

5.2 *SEQUENCE* vs *AND*

Sometimes, when several steps are required to perform an attack, these steps may need to be performed in a specific order. For this case, it could be interesting to define a *SEQUENCE* operator, in addition to the *AND* operator. However, the propagation rules of these two operators will be the same in our case.

5.3 A 2-uple attacker vs a 3-uple one

Instead of describing an attacker by its resources R_M and its level of expertise E_M, we could have decomposed its resources into two parts: its financial resources F_M (which condition the hardware equipment it has access to) and the time window T_M it has to perform its attack. However, T_M and R_M are not always independent, for instance, an attacker may use money to buy extra equipment and save time.

Thus, we chose to represent time and financial resources in one parameter R (that includes the potential trade-off between financial resources and time) and define the attacker as a 2-uple (R, E). For the same reasons, an attack is defined as a 3-uple (l, R, E) and not a 4-uple (l, T, F, E).

6 Case study

Let us consider the same system S that we have already described in Figure 1 but with a more detailed ADT-SA.

Figure 7 represents an ADT-SA for an embedded system with a processor, an external RAM (not embedded inside the processor), a bus between the processor and the RAM. A small operating system and an application that performs cryptographic operations (such as encryption), using a secret key, runs on this system.

The attacker M wants to retrieve the confidential data manipulated by the system (in our case, the secret key manipulated by the application). The root attack A_{root} is *GetConfidentialData*. To perform this attack, the attacker M can either perform A_{bus} a bus probing attack that targets the component *Bus*, A_{SCA} a side-channel attack against the *Processor* during the manipulation of

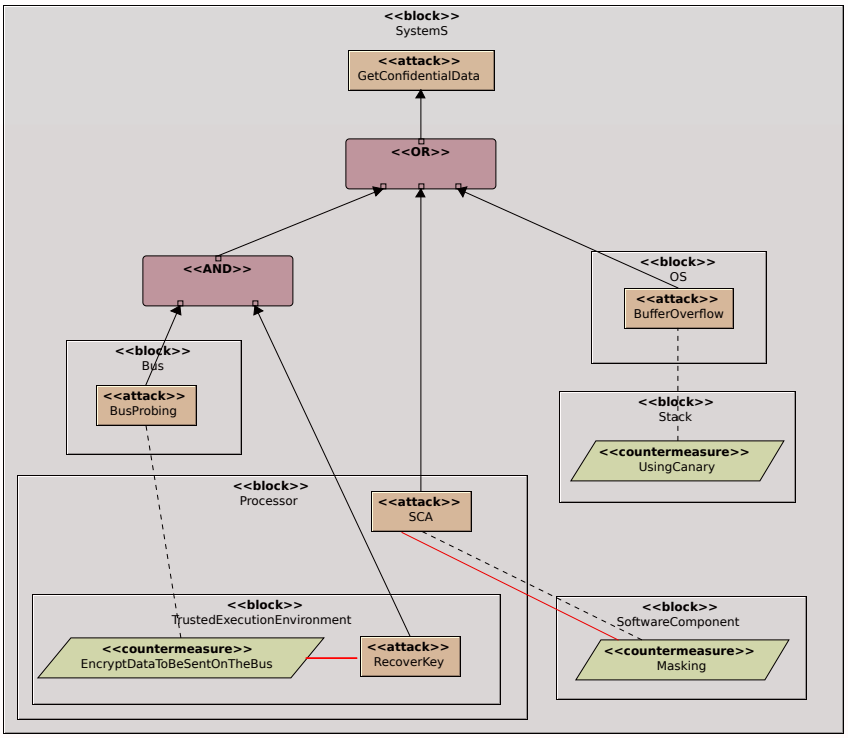

Fig. 7. Example of a more detailed ADT-SA for the system S

the secret key or A_{OS} a buffer overflow on the *Operating system* in order to take control of it and read confidential data directly from the memory.

We suppose that the required level of expertise to success an attack is represented by three values: *beginner*, *intermediate*, *expert*. We also arbitrarily fix the parameters of the three attacks:

- $R_{A_{bus}} = 40, E_{A_{bus}} = intermediate$
- $R_{A_{SCA}} = 50, E_{A_{SCA}} = intermediate$
- $R_{A_{OS}} = 90, E_{A_{OS}} = beginner$

Three countermeasures are described on the ADT-SA:

- *EncryptDataToBeSentOnTheBus* (CM_{enc}) which consists in using a *Trusted Execution Enviroment TEE* that automatically encrypts data that are sent on the bus to the memory. The cost of data encryption on the bus is 1000 and the cost of TEE component is 2000. Thus, the cost of this countermeasure is $C_{enc} = 1000+2000 = 3000$. CM_{enc} adds a new attack step $N_{CM_{enc}} = \{A_{key}\}$. This new attack step (*RecoverKey* in the ADT-SA) consists in recovering the

key used to encrypt the bus in order to decrypt data intercepted on the bus. For this new step: $R_{A_{key}} = 70, E_{A_{key}} = expert$.
- *Masking* (CM_{msk}) which protects against first order side-channel attacks. The cost of the countermeasure is $C_{msk} = 5000$ and it increases the difficulty of the attack A_{SCA} by requiring a second-order attack instead of a first-order one, so $I_{CM_{msk}} = \{(A_{SCA}, 70, expert)\}$.
- *UsingCanary* (CM_{OS}) which consists in using canary values to prevent buffer overflows on the stack from modifying the return address of functions. The cost is $C_{cnry} = 200$ and it increases the difficulty of A_{OS} so $I_{CM_{cnry}} = \{(A_{OS}, 80, expert)\}$.

Hence, when using CM_{enc}, the minimal resources required to perform A_{bus} are $R_{A_{bus}} + R_{A_{key}} = 40 + 70 = 110$ and the minimal expertise required for A_{bus} is $\max(E_{A_{bus}}, E_{A_{key}}) = \max(intermediate, expert) = expert$ (the adversary has to perform A_{bus} and $_{key}$). CM_{msk} and CM_{cnry} do not introduce new attack steps, but they increase the minimal resources and expertise need to achieve A_{SCA} and A_{OS}. Table 1 shows the new minimal resources and expertise required to perform A_{SCA} and A_{OS}:

Attack	Minimal resources	Minimal expertise
A_{SCA}	$50 + 70 = 120$	$\max(intermediate, expert) = expert$
A_{OS}	$90 + 80 = 170$	$\max(beginner, expert) = expert$

Table 1. Minimal resources and expertise required with the countermeasures

If we suppose that the 3 countermeasures CM_{enc}, CM_{msk} and CM_{cnry} are implemented, we can use the propagation rules of the *OR* operator to find that in order to achieve A_{root}, an attacker should be *expert* and its resources should be greater or equal to 110.

Let us consider the population $\mathcal{M}$ of 5 attackers presented in table 2.

Attacker	Resources	Expertise
M_1	130	*expert*
M_2	180	*intermediate*
M_3	60	*expert*
M_4	40	*expert*
M_5	70	*beginner*

Table 2. Resources and expertise of each attacker in the population

We supposed that the system architect has fixed a budget of 14000 to build it. We supposed also that $C_{Total_{Comp}} = 6000$ (cost of the basic components of

the system without the security equipements: OS, processor, stacks, bus, RAM). Thus, there are $14000 - 6000 = 8000$ for security investment i.e. use a TEE and implement CM_{enc}, CM_{msk} and CM_{cnry}. However, $C_{Total_{CM}|SA} = 5000 + 3000 + 200 = 8200$. Therefore, it could not deploy all of the countermeasures and it has to choose the set of countermeasures x that minimizes $P(x)$ for $\mathcal{M}$ and respect the constraint $C_{arch}(x) \leq 14000$.

If it chooses to implement CM_{enc} only, then the minimal resources and expertise needed to achieve A_{SCA} and A_{OS} will not increase. Thus, attackers with resources ≥ 50 and expertise $\geq$ *intermediate* will perform A_{SCA} and those whose resources ≥ 90 and expertise $\geq$ *beginner* will perform A_{OS} (OR operator). Hence, only attackers $M1$, $M2$ and $M3$ will successfully perform A_{root}, thus $P(\{CM_{enc}\}) = 3/5 = 0.6$ and $C_{arch}(x) = 6000 + 3000 \leq 14000$.

We can perform the same analysis for each set of countermeasures. Results are showed in table 3 below.

Countermeasures	Successful attackers	$C_{Total_{CM}\|SA}$	C_{arch}	Prob. of success
$\{CM_{enc}\}$	$\{M_1, M_2, M_3\}$	3000	9000	$3/5 = 0.6$
$\{CM_{msk}\}$	$\{M_1, M_2, M_3, M_4\}$	5000	11000	$4/5 = 0.8$
$\{CM_{cnry}\}$	$\{M_1, M_2, M_3, M_4\}$	200	6200	$4/5 = 0.8$
$\{CM_{enc}, CM_{msk}\}$	$\{M_1, M_2\}$	8000	14000	$2/5 = 0.4$
$\{CM_{enc}, CM_{cnry}\}$	$\{M_1, M_2, M_3\}$	3200	9200	$3/5 = 0.6$
$\{CM_{msk}, CM_{cnry}\}$	$\{M_1, M_2, M_3, M_4\}$	5200	11200	$4/5 = 0.8$
$\{CM_{enc}, CM_{msk}, CM_{cnry}\}$	$\{M_1\}$	8200	14200	$1/5 = 0.2$

Table 3. Costs and probability of success for each set of countermeasures

Hence, the set of countermeasure that solves our minimization problem is $x = \{CM_{enc}, CM_{msk}\}$ since $C_{arch}(x) = 14000$ and $P(x)$ is the minimal among the other values.

This example shows, on a small system, how our methodology can be used to find a good trade-off between the cost of an architecture and its security.

7 Conclusion

In this paper, we have presented a mechanism to help a system architect choose the right trade-off between the level of security of a system (that can be increased by adding countermeasures) and the total cost of the system. We improve attack-defense trees to describe the different attacks scenarios and how the countermeasures affect both the attack and the required hardware or software components. From the characterization of the elementary attack steps (in terms of resources and level of expertise required to successfully perform the attack), we use propagation rules to characterize root attacks. Given a population of malicious adversary, this leads to compute a probability of success of a root attack

depending on the countermeasures that are implemented. From this point, the trade-off between security and cost can be expressed as an optimization problem (minimizing the probability of success of the adversary) with a constraint on the cost of the system.

We are currently working in two directions.

7.1 Automation

First, we are implementing our methodology in TTool [3] in order to help the system architect detail and refine attack-defense trees and make the trade-off between the cost of the architecture and the probability of success of the attacks. This is realized by automatically exploring the set of available countermeasures and computing the probability of success of attacks and architecture cost in the different configurations.

7.2 Attack-Defense Trees and Fault Trees

The next step is to take care of the safety aspect. A countermeasure increases the security of a system but it may degrade its safety by introducing new components that can fail. In addition, classical safety improvement techniques may degrade the security of the system by creating new attack scenarios. So there is a link between attack-defense trees and fault trees. We are working to formalize this link in order to allow the system architect to not only make a trade-off between security and cost but also take into account the safety.

8 Acknowledgments

This work is supported by the research chair *Connected Cars and Cyber Security* (C3S) [6] founded by Nokia, Renault, Thales, Valeo, Wavestone, Fondation Mines-Télécom and Télécom Paris.

References

1. A deep flaw in your car lets hackers shut down safety features, https://www.wired.com/story/car-hack-shut-down-safety-features/
2. Sysml-sec, http://sysml-sec.telecom-paristech.fr/
3. TTool, https://ttool.telecom-paristech.fr/
4. OMG Systems Modeling Language (OMG SysML^{TM}), V1.0. Tech. rep., Object Management Group (Sep 2007), http://www.omg.org/spec/SysML/1.0/PDF
5. A survey on the usability and practical applications of graphical security models. Comput. Sci. Rev. **26**(C), 1–16 (Nov 2017). https://doi.org/10.1016/j.cosrev.2017.09.001, https://doi.org/10.1016/j.cosrev.2017.09.001
6. Research chair Connected Cars and Cyber Security (C3S). https://www.telecom-paristech.fr/c3s (2019)

7. Audinot, M., Pinchinat, S.: On the soundness of attack trees. vol. 9987, pp. 25–38 (06 2016). https://doi.org/10.1007/978-3-319-46263-9_2
8. Bistarelli, S., Dall'Aglio, M., Peretti, P.: Strategic games on defense trees. In: Formal Aspects in Security and Trust (2006)
9. Edge, K., Dalton, G., Raines, R., Mills, R.: Using attack and protection trees to analyze threats and defenses to homeland security. pp. 1 – 7 (11 2006). https://doi.org/10.1109/MILCOM.2006.302512
10. 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. vol. 267 (11 2016). https://doi.org/10.1007/978-3-319-48393-1_24
11. Garro, A., Tundis, A.: A model-based method for system reliability analysis. vol. 44 (03 2012)
12. Ji, X., Yu, H., Fan, G., Fu, W.: Attack-defense trees based cyber security analysis for cpss. In: 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). pp. 693–698 (May 2016). https://doi.org/10.1109/SNPD.2016.7515980
13. Jürgenson, A., Willemson, J.: Computing exact outcomes of multi-parameter attack trees. In: OTM Conferences (2008)
14. Jürgenson, A., Willemson, J.: Serial model for attack tree computations. In: Proceedings of the 12th International Conference on Information Security and Cryptology. pp. 118–128. ICISC'09, Springer-Verlag, Berlin, Heidelberg (2010), http://dl.acm.org/citation.cfm?id=1883749.1883762
15. Kordy, B., Kordy, P., Mauw, S., Schweitzer, P.: Adtool: Security analysis with attack–defense trees. In: Joshi, K., Siegle, M., Stoelinga, M., D'Argenio, P.R. (eds.) Quantitative Evaluation of Systems. pp. 173–176. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
16. Kordy, B., Mauw, S., Radomirovic, S., Schweitzer, P.: Foundations of attack-defense trees. In: Formal Aspects in Security and Trust (2010)
17. Kordy, B., Pièvre-cambacédès, L., Schweitzer, P.: Dag-based attack and defense modeling: Don't miss the forest for the attack trees. CoRR p. 2013
18. Kordy, B., Wideł, W.: On Quantitative Analysis of Attack–Defense Trees with Repeated Labels, pp. 325–346 (04 2018). https://doi.org/10.1007/978-3-319-89722-6_14
19. van Lamsweerde, A.: Elaborating security requirements by construction of intentional anti-models. In: Proceedings of the 26th International Conference on Software Engineering. pp. 148–157. ICSE '04, IEEE Computer Society, Washington, DC, USA (2004), http://dl.acm.org/citation.cfm?id=998675.999421
20. Mauw, S., Oostdijk, M.: Foundations of attack trees. In: Proceedings of the 8th International Conference on Information Security and Cryptology. pp. 186–198. ICISC'05, Springer-Verlag, Berlin, Heidelberg (2006). https://doi.org/10.1007/11734727_17, http://dx.doi.org/10.1007/11734727_17
21. Saini, V., Duan, Q., Paruchuri, V.: Threat modeling using attack trees. J. Comput. Sci. Coll. **23**(4), 124–131 (Apr 2008), http://dl.acm.org/citation.cfm?id=1352079.1352100
22. Schneier, B.: Secrets & Lies: Digital Security in a Networked World. John Wiley & Sons, Inc., New York, NY, USA, 1st edn. (2000)
23. Steiner, M., Liggesmeyer, P.: Qualitative and quantitative analysis of cfts taking security causes into account. In: Koornneef, F., van Gulijk, C. (eds.) Computer Safety, Reliability, and Security. pp. 109–120. Springer International Publishing, Cham (2015)

24. Zhou, S., Sun, Q., Jiao, J.: A safety modeling method based on sysml. In: 2014 10th International Conference on Reliability, Maintainability and Safety (ICRMS). pp. 1180–1185 (Aug 2014). https://doi.org/10.1109/ICRMS.2014.7107390