Mitigating Security Risks through Attack Strategies Exploration

Braham Lotfi Mediouni, Ayoub Nouri, Marius Bozga, Axel Legay, Saddek Bensalem

Mitigating Security Risks through Attack Strategies Exploration

Braham Lotfi Mediouni[1], Ayoub Nouri[1], Marius Bozga[1], Axel Legay[2], and Saddek Bensalem[1]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP**, VERIMAG, 38000 Grenoble, France
[2] INRIA, Rennes, France

Abstract. Security assessment of organization's information systems is becoming increasingly complex due to their growing size and the underlying architectures (e.g., cloud). Analyzing potential attacks is a pragmatic approach that provides insightful information to achieve this purpose. In this work, we propose to synthesize effective defense configurations for sophisticated attack strategies, which are obtained by minimizing resource usage while ensuring a high probability of success. Obtained results on real-life case studies show substantial improvement compared to existing techniques.

1 Introduction

Modern organizations strongly rely on information and communication technologies in their daily activities. This reliance put forward serious questions about the inherent risks brought by these technologies and how to manage them. Risk management is the activity consisting of identifying, analyzing, evaluating, treating and monitoring risks that an organization is subject to.

Risk assessment consists on the analysis and the evaluation of the organization's vulnerabilities with respect to the deployed security policy. Attack Trees (AT) [**?**] have emerged as a promising formalism to structure risk assessment and security analysis. It is a compact representation of the organization's probable threats and its vulnerabilities to these threats. They allow to cover deliberate and accidental threats that may affect organization security, ranging from hacking and viruses, theft or equipment deterioration, to human errors. This formalism has been recently extended to support both the organization's security breaches and their countermeasures. This extension, called Attack-Defense Tree (ADT) [**?**], brings a more realistic view of possible attack scenarios with respect to existing defenses and hence leads to more viable diagnoses.

In this paper, we propose a risk assessment method based on the ADT formalism to find impactful defenses that prevent cost-effective attack strategies. We follow an offensive approach, that is, evaluating defenses by performing attacks on the system under study. In our heuristic (IO-Def), we focus on finding the adequate defenses against an optimized attack strategy characterized in terms

** Institute of Engineering Univ. Grenoble Alpes

of attack cost and success probability. These characteristics are computed using Statistical Model Checking (SMC) techniques with respect to a near-optimal cost-effective attack strategy. This strategy is explored by a hybrid variant of a genetic algorithm and local search (IEGA), as opposed to [**?**] that relies on reinforcement learning. Genetic algorithms are evolutionary algorithms that have shown effectiveness in exploring large solution spaces to select high-quality answers for optimization and search problems. Moreover, several extensions allow to perform multi-objective explorations [**?**,**?**,**?**,**?**]. In this work, the role of IEGA is to learn a strategy of attack that minimizes the attacker cost while maximizing its probability to succeed, given a deployed defense configuration.

The idea of the paper is as follows:

We start from the a set of known vulnerabilities in the system (subject of study). Identifying vulnerabilities is challenging and is an active field of research. This work assumes that vulnerabilities identifications has been already performed. It is also to note that this should a continuous activity since the system changes over time (updates, etc.)

Identified vulnerabilities give an idea about attack actions that potentially exploit them (this gives concretely the set of attack actions to consider). Given these attacks, defense actions are foreseen to deal with them (this gives the set of defense actions to consider). However, defense actions provide overlapping protections and all of them are always pertinent to deploy in the same time. Moreover, companies have generally limited budget for security. It is thus primordial to rely on rigorous quantitative techniques that helps exploring and identifying the most pertinent defenses in such a setting.

Besides, attack actions can be combined in different ways (by a sophisticated attacker) to circumvent existing defense actions. This is what we call attack strategy. It thus important to think of defense configuration/strategy and not in term of individual defense actions. The second requirement is thus to explore defense strategies with respect to different attack profiles, *offensive security*.

We intend by attack profiles, that is attack strategies fulfilled with respect to some logic (not the goal as the goal is always to get control, etc.). In reality attackers are also subject to physical constraints, e.g. limited resources, time, some knowledge about the probability of success of individual attack actions, etc. It is also to remark that attack actions requiring short time and limited resources have lower probability of success and conversely. Accordingly, different profiles could be identified. The underlying logic (profile) of attacker differ with respect to different human parameter (experience, etc.). It could for instance that the attacker aim at using attack actions with limited resources or the opposite. However, we believe that a sophisticated attacker (I would say the majority) would try to optimize these criterion, that is aiming at finding compromises that end-up with attack strategies requiring an affordable amount resources and an acceptable probability of success.

In this paper, we propose a parametric approach (in the sense that it may consider different attacker profiles) the allows to synthesize pertinent defense strategies to make sophisticated attacks (with respect to a specific profile) harder

to achieve. In this work, we consider the cost required for an attack as the hardness criterion. In other words, the exploration of the defense strategies aims at identifying the most pertinent defense actions (conversely the one not pertinent) that maximizes the cost of an attack.

This paper is organized as follows. Related work is presented in Section 2. In Section 3, we provide an overview of the attacker, defender, and attack-defense tree models, in addition to their interaction in the risk assessment model. Details about the attack strategy exploration technique (IEGA) is given in Section 4. Section 5 develops the approach we introduce for the discovery of an impactful defense configuration. In Section 6, we evaluate the proposed methods on four case studies. Finally, Section 7 concludes and presents future directions.

2 Related Work

Attack Trees (AT) [**?**] are widely used in security to model system vulnerabilities and the different combinations of threats to address a malicious goal. Attack-Defense Trees (ADT) [**?**] extend ATs with defense measures, also known as countermeasures, to include the organizations defenses and bring into consideration the impact of attacks on these organizations. These defense actions try to prevent an attacker from reaching its final goal, and they can appear at any level of the tree. More recently, Attack-Countermeasure Trees (ACT) [**?**] were introduced to model defense mechanisms that are dynamically triggered upon attack detection.

Different types of analysis are proposed on these variants of trees. In [**?**] authors focus on the probabilistic analysis of ATs, through the computation of the probability, cost, risk and impact of an attacker's goal. A similar analysis is performed on ADTs in [**?**], called Threat Risk Analysis (TRA), applied to the security assessment of cloud systems. In addition to the aforementioned probabilistic analysis, Roy et al. [**?**] make use of the structural and Birnbaum importance measure to prioritize attack events and countermeasures in ACTs.

Authors in [**?**] propose a reinforcement-based method on ADTs to find a near-optimal attack strategy. In this work, an attacker with a complex probabilistic and timed behavior is considered which makes it more difficult to perform a static analysis. The authors propose to address the security analysis problem from the attacker's viewpoint by synthesizing the stochastic and timed strategy that minimizes the attack cost using UPPAAL STRATEGO tool. The strategy indicates the attack action to perform in each state in order to realize a successful attack with a minimal cost.

In the previous approach, attack actions are described with an additional time interval determining their duration range. The learned strategy identifies, in addition to the attack sequences, the time durations to respect a maximum time bound of reaching the attacker's goal. However it is not always the case that an attacker can control the duration time of an attack action, eg. the execution time of a brute-force attack on a password. Instead, we consider time as a characteristic of an attack action, i.e., cannot be controlled as it depends

on the system, environment, etc. We consider the maximum time bound as a global success condition of an attack, and we propose IEGA, a hybrid Genetic Algorithm to find the stochastic strategy minimizing the attack cost while maximizing the probability of reaching the attacker's goal. This strategy acts similarly to a scheduler on attack actions that tells the attacker which action to perform when a choice is required.

3 Background

In this section, we formally introduce definitions and notations used in the remainder of the paper. We first introduce the models for attacker and defender. Then, we recall the definition of an attack-defense tree, and finally, we describe the model used for risk assessment.

For the following definitions, we consider Σ_A to be a set of attack actions, Σ_D is a set of defense actions, and $\Sigma = \Sigma_A \cup \Sigma_D$ the set of all actions. Furthermore, we consider that each attack action $a \in \Sigma_A$ is associated with 1) a time interval $[l_a, u_a]$ that represents lower and upper time bounds allowed to perform a, 2) a cost $c_a \in \mathbb{R}$ which models needed resources to perform a and 3) a probability of success p_a that represents the likelihood for a to succeed when performed. We call *environment* the probabilities of success of all the attack actions in Σ_A and we denote it env.

3.1 Attacker, Defender and Attack-Defense tree

Attacker. The attacker model represents all possible attack combinations given the alphabet of attack action Σ_A. It is syntactically defined as follows:

Definition 1 (Attacker). *An attacker $\mathcal{A}$ is a tuple $\langle L, l_0, T\rangle$ where :*

- *$L = \{l_0, \ldots\}$ is a set of locations, where l_0 is the initial location,*
- *$T \subseteq L \times \Sigma_a \times L$ is a set of labeled transitions of the form (l_i, a, l_j).*

Intuitively, an attacker $\mathcal{A}$ starts at the initial state and performs a sequence of attack actions by choosing each time among the available ones in Σ_A. At a given state, an attack action a may succeed, which leads to a new state where a is no more available and where all other actions remain unchanged. In the case where a fails, the state of the attacker does not change. Remember that the success or failure of a selected attack action is not controlled by the attacker, but is determined by the environment introduced in the beginning of this section. We formally define the behavior of an attacker as follows. Let $status : \Sigma_A \times S \to \{0, 1\}$ be a predicate that indicates, at a given state s, whether an attack action has previously succeeded.

Definition 2 (Attacker semantics). *An attacker $\mathcal{A} = \langle L, l_0, T\rangle$ is labeled transition system $\langle S, s_0, R\rangle$, where*

- *$S = L \times V_{\Sigma_A}$, where V_{Σ_A} is a vector that contains the status (according to the predicate status) of all the attack actions in Σ_A,*

- $s_0 = (l_0 \times V^0_{\Sigma_A})$ *is the initial state, where* $V^0_{\Sigma_A} = [0, \ldots, 0]$ *is the initial status of all the attack actions in* Σ_A*,*
- $R \subseteq S \times \Sigma_A \times S$ *is a set of transitions of the form* (s_i, a, s_j) *built out of* T*:*
  1. *Success:* $\frac{s_i=(l_i,\ V^i_{\Sigma_A})\ \wedge\ status(a,s_i)=0}{s_j=(l_j,\ V^j_{\Sigma_A}),\ l_j \neq l_i,\ status(a,s_j)=1,\ \forall a_k \neq a\ \ status(a_k,s_j)=status(a_k,s_i)}$
  2. *Failure:* $\frac{s_i=(l_i,\ V^i_{\Sigma_A}),\ status(a,s_i)=0}{s_j=s_i}$

Note that the attacker semantics above is non-deterministic, that is the choice of an attack action at each state is performed non-deterministically. We introduce the notion of attack *strategy* to cope with this non-determinism. An attack strategy $\mathcal{S} : \Sigma_A \longrightarrow [0, 1]$ is a mass probability function that associates each attack action with a probability of being selected by the attacker[3]. We denote by $\mathcal{A}|_{\mathcal{S}}$ the attacker $\mathcal{A}$ that applies the strategy $\mathcal{S}$. Thus, the probability $P : S \times \Sigma_A \longrightarrow [0, 1]$ to select an attack action a at any state s_i is defined as

$$P(s_i, a) = \begin{cases} 0 & \text{if } status(a, s_i) = 1 \\ \frac{\mathcal{S}(a)}{\sum_{j=1} \mathcal{S}(a_j) \times (1 - status(a_j, si))} & \text{otherwise} \end{cases}$$

Defender. A defender models the deployed set of defense actions. In this work, it represents a static defense configuration, where a defense action $d \in \Sigma_D$ is either enabled or not in all the states of the system. It is defined as follows:

Definition 3 (Defender). *A defender* $\mathcal{D} \subseteq \Sigma_D$ *is the subset of enabled defense actions in* Σ_D*.*

Similarly to the attacker, we define a predicate $enabled : \Sigma_D \rightarrow \{0, 1\}$ that tells if a defense action is currently enabled. Formally, $enabled(d) = 1$ when $d \in \mathcal{D}$, and 0 otherwise.

Attack-Defense Tree. It represents some knowledge about the system subject to analysis. For instance, it includes the attack combinations (with respect to the analyzed system vulnerabilities) that may lead to the success of an attack, along defense mechanism deployed (in use) in the system. In this work, we define it as a Boolean combination of attack and defense actions as follows:

Definition 4 (Attack-Defense Tree). *An attack-defense tree* $\mathcal{T}$ *is a defined by the following inductive grammar:*

$$\phi, \phi_1, \phi_2 ::= true \mid ap \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \neg\phi \mid (\phi),\ \ where\ ap \in \Sigma$$

The evaluation of the attack-defense tree requires an attacker model $\mathcal{A}$ and a defender model $\mathcal{D}$. This is performed as part of the risk analysis activity that relies on a Risk Assessment Model introduced below.

[3] Note that the strategy is static, i.e., the same in any state, in this work. Considering dynamic strategies is a future work.

3.2 Risk Assessment Model

We now explain how the previous models, namely Attacker, Defender and Attack-Defense Tree are used together to build a complete view for analysis, called *Risk Assessment Model* and defined as follows.

Definition 5 (Risk Assessment Model). *A risk assessment model $\mathcal{M}$ is a composition of:*

- $\mathcal{A}|_{\mathcal{S}}$ *is an attacker provided with a strategy* $\mathcal{S}$*,*
- $env : \Sigma_A \longrightarrow [0,1]$ *is the environment,*
- $\mathcal{D}$ *is a defender,*
- $\mathcal{T}$ *is an attack-defense tree,*
- $c_{max}, t_{max} \in \mathbb{R}$ *are the maximal attacker cost and time resources.*

It allows to simulate attacks (represented by an Attacker $\mathcal{A}|_{\mathcal{S}}$) – under constraints c_{max} and t_{max} – on the system (abstracted by the environment *env*) against a fixed defense configuration (modeled by the Defender $\mathcal{D}$). The status of an attack is given by the current status of the Attack-Defense Tree $\mathcal{T}$. The evaluation of the status of an attack using the attack-defense tree $\mathcal{T}$ is twofold:

1. the defense configuration $\mathcal{D}$ is used to evaluate the defense part of the tree, (i.e., ap of $\mathcal{T}$ such that $ap \in \Sigma_D$). This phase is done statically since the defense is fixed in our case. For each $ap \in \mathcal{T}$, where ap is a defense action, ap is evaluated to $true$ (resp. $false$) whenever $enabled(ap) = 1$ (resp. $enabled(ap) = 0$).
2. second, the attacker $\mathcal{A}|_S$ is used dynamically to sequentially generate attack actions a_i that may succeed or fail according to the environment vector *env*. Whenever an attack a_i succeeds, the corresponding action in $\mathcal{T}$ is evaluated to $true$. Attack actions in $\mathcal{T}$ are either evaluated to $true$ or not yet.

An execution trace ω of the risk assessment model $\mathcal{M}$ (denoted *attack trace*) is a sequence of timed attack actions (a_i, τ_i), where $\tau_i \in [l_{a_i}, u_{a_i}]$ is the duration of action a_i. We call $\Omega_{\mathcal{M}}$ the set of all attack traces generated by $\mathcal{M}$. Remark that the attacker model is constrained by c_{max} and t_{max} which define a budget of available resources and time to perform a sequence of attack actions. Hence, an attack trace is finite and ends in one of the scenarios below. Let us first introduce the *attack cost* and the *attack duration* as follows. Given a trace $\omega \in \Omega_{\mathcal{M}}$ of length n, the attack cost is $cost(\omega) = \Sigma_{i=1}^{n} c_{a_i}$, where c_{a_i} is the cost associated to action a_i. Similarly, the attack duration is $duration(\omega) = \Sigma_{i=1}^{n} \tau_i$. Thus, an attack trace ends when:

- the attack-defense tree $\mathcal{T}$ is evaluated to $true$ or $false$,
- the attacker has exhausted his resources or time budget, i.e., when $cost(\omega) > c_{max} \vee duration(\omega) > t_{max}$,
- the attacker cannot select more attack actions based on the strategy $\mathcal{S}$.

It is worth mentioning that the attack-defense tree $\mathcal{T}$ is evaluated to $false$ only when the defense configuration $\mathcal{D}$ prevents all the tree branches from simplifying to $true$. In contrast, the tree evaluates to $true$ when the attacker's goal is fulfilled. The third situation happens when the attacker cannot choose an action according to the strategy $\mathcal{S}$ that could have simplified the attack-defense tree.

Given a trace ω, we interpret it as a successful attack whenever the attack-defense tree is simplified to $true$ in addition to having $cost(\omega)$ and $duration(\omega)$ below the c_{max} and t_{max} respectively, and as a failed attack otherwise.

4 Synthesizing Cost-effective Attack Strategies

In this section, we present our approach to explore attack strategies. As explained earlier, our goal is to identify the most cost-effective strategy under which an attack is most likely to succeed. Our proposal is based on a hybrid variant of GA and Local Search (LS), called Intensified Elitist Genetic Algorithm (IEGA) that allows to identify a near-optimal attack strategy.

A Genetic Algorithm (GA) is an evolutionary algorithm inspired from natural selection and genetics. It provides an efficient way to explore large solution spaces to select high-quality answers for optimization and search problems. For that, solutions (individuals of the genetic population) need to be comparable in a quantitative basis.

4.1 Approach Overview

The approach considers, as input, a Risk Assessment Model $\mathcal{M}$ that is composed of an attacker model $\mathcal{A}$, an environment env, a defender model $\mathcal{D}$, an attack-defense tree $\mathcal{T}$ and the constraints t_{max} and c_{max}.

In the proposed approach, an individual is denoted $\mathcal{I} = \langle \mathcal{S}, cost, p \rangle$, where $\mathcal{S}$ is an attack strategy together with its expected $cost$ and the probability p of an attack being successful under $\mathcal{S}$. Besides, we use SMC as a mean to evaluate individuals. That is, given a strategy $\mathcal{S}$, SMC estimates the $cost$ and the probability p of an attack being successful. More precisely, we rely on the probability estimation algorithm (PESTIM) [**?**] to check the risk assessment model against the property $\phi = \Diamond_{t<t_{max}}^{c<c_{max}} \mathcal{T}$. Recall that the precision of PESTIM is controlled by the confidence level (δ, α).

The SMC can lead to update the strategy $\mathcal{S}$ when one or more primordial attack actions were assigned a zero-probability in the strategy resulting in a zero probability of success ($p(\phi) = 0$). In this case, $\mathcal{S}$ is updated by assigning residual probabilities to actions with a null probability to occur.

As depicted in Fig. 1, $IEGA$ starts by randomly generating N initial strategies (individuals) to constitute the initial population P_0, evolving over M generations. For each generation, $N/2$ new children strategies are generated as follows:

1. **Selection for breeding**: we randomly choose two parent individuals in the current population as candidates for the cross-over operation,

2. **Cross-over operation**: a child individual is built by performing a single-point cross-over,
3. **Intensification with LS**: the resulting individual is intensified using LS, i.e., a heuristic aiming at iteratively improving it by exploring its neighbor solutions,
4. **Mutation**: after intensification, an individual has a $p_{mutation}$-probability to be mutated, i.e., altering the selection probability of a randomly chosen attack action.

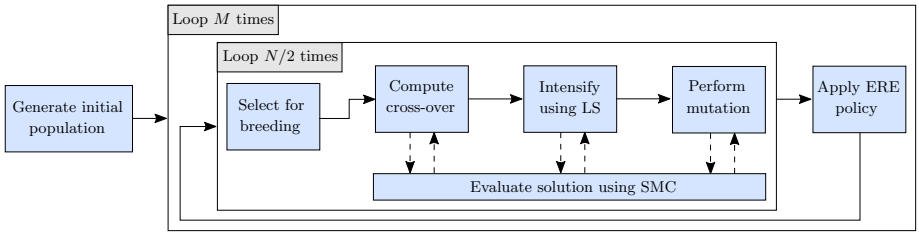


Fig. 1: Workflow of IEGA with a population of N individuals over M generations

The last phase of the outer loop in Fig. 1 identifies among parents individuals in population P_i and their $N/2$ children, the ones kept in the next generation i+1. More precisely, we use Extreme Ranking Elitism (ERE) [**?**] as a replacement policy. This technique aims at selecting the best individuals while keeping some diversity in the population. Concretely, in addition to the best solutions, bad ones are kept to provide diversity and prevent early convergence.

In the next section, we further detail the cross-over, the Local Search (LS) and the Extreme Ranking Ellitism (ERE) operations. Selection for breeding and mutation are both based on random sampling in this case. Hence, they will not be further discussed.

4.2 Operations Description

Cross-over operation. A cross-over consists on building a child individual $\mathcal{I} = \langle \mathcal{S}, cost, p \rangle$ by combining two randomly selected parents $\mathcal{I}_1 = \langle \mathcal{S}_1, cost_1, p_1 \rangle$ and $\mathcal{I}_2 = \langle \mathcal{S}_2, cost_2, p_2 \rangle$. $\mathcal{I}$ is obtained by performing a single-point cross-over, i.e., $\mathcal{I}$ inherits the first half of its genes from $\mathcal{I}_1$ and the second half from $\mathcal{I}_2$ as follows:

$$\mathcal{S}[i] = \begin{cases} \mathcal{S}_1[i], & i \leq |\Sigma_A|/2 \\ \mathcal{S}_2[i], & \text{otherwise} \end{cases}$$

This operation is followed by a normalization to guarantee that the strategy $\mathcal{S}$ is a valid mass function, i.e., $\Sigma_i(\mathcal{S}[i]) = 1$.

Intensification with LS. The individuals resulting from the cross-over are intensified, i.e. improved, using a local search (LS) over a set of neighbor solutions.

Individuals are said to be neighbors when their respective strategies are slightly different. More formally, given an individual $\mathcal{I} = \langle \mathcal{S}, cost, p \rangle$, the set of neighbor solutions $V(\mathcal{I}) = \{\mathcal{I}_i = \langle \mathcal{S}_i, cost_i, p_i \rangle\}$ to individual $\mathcal{I}$ is identified by reducing the set of enabled actions by one, as follows:

- if $\mathcal{S}[i] = 1$ or $\mathcal{S}[i] = 0$ then the i^{th} neighbor individual $\mathcal{I}_i$ does not exist,
- otherwise, individual $\mathcal{I}_i$ is identified by a strategy $\mathcal{S}_i$ such that:

$$\mathcal{S}_i[j] = \begin{cases} 0, & j = i \\ \frac{\mathcal{S}[j]}{\Sigma_k(\mathcal{S}[k]) - \mathcal{S}[i]}, & \text{otherwise} \end{cases} \quad (1)$$

The normalization in the second case is again to ensure well-formedness of the synthesized strategy (probability distribution). It is worth mentioning that an individual has at most $|\Sigma_A|$ neighbors.

LS improves the current solution by repeatedly jumping to better ones residing in its neighborhood, until no improvement is possible. A neighbor solution I_i is said to improve the current one I if it has a better fitness value. The latter is computed using the fitness function $Score$ which is a weighted sum of the $cost$ and the probability of success p. Formally, the fitness function is defined as $Score(cost, p) = a \times p - (1 - a) \times cost$, where $a \in [0, 1]$ represents a linearization factor [4].

ERE replacement policy A genetic algorithm maintains a population of size N through M generations. The replacement operation rules the survival of individuals through generations. Extreme Ranking Elitist replacement is a balanced solution to provide elitism while avoiding early convergence.

Given a population P_i of N parents and their $N/2$ children, an Extreme Ranking Elitist replacement policy identifies the N candidate individuals for the next generation's population P_{i+1}. This policy is parametrized by p_{ere}, that represents the proportion of the population to be selected by elitism. More precisely, the replacement is performed as follows:

1. We consider an intermediate population P'_i of size $\frac{3N}{2}$ composed of the N parents and their $N/2$ children. Individuals in this population are ranked based on the Pareto dominance principle, and sorted in an ascending order. In the Pareto dominance principle, a solution I_1 is known as dominated by another solution I_2 if the latter is better for every criterion, in our case, $cost_1 \geq cost_2 \wedge p_1 \leq p_2$ excluding the case where they are all equal. Considering this definition, the ranking consists on assigning rank 1 to non-dominated solutions of the population. Iteratively, we temporarily remove the non-dominated ones and identify the new non-dominated solutions that we assign the next rank, until all the solutions are ranked. Fig. 2 is an example of Pareto ranking on a population of 10 individuals.

[4] This is used for weighting and scaling the two inputs.

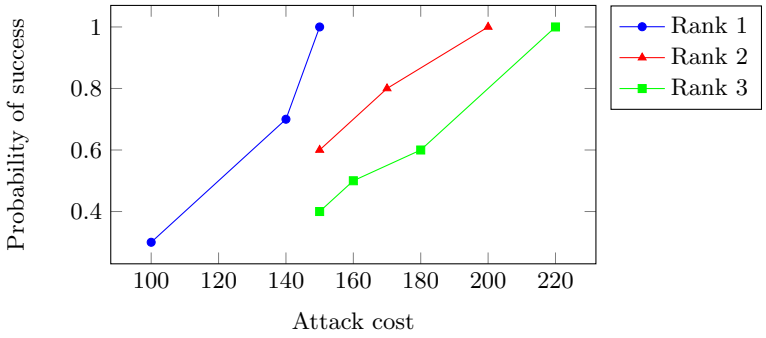


Fig. 2: Illustration of Pareto dominance ranking

2. To select the N individuals to be part of generation $(i+1)$, we compute the number of best (elite) individuals $N_b = N \times p_{ere}$, and the number of worst individuals $N_w = N \times (1 - p_{ere})$ kept for diversification. Population P_{i+1} is computed as:

$$P_{i+1} = \bigcup_{j=1}^{N_b} \{P'_i(j)\} \cup \bigcup_{k=\frac{3N}{2}-N_w+1}^{\frac{3N}{2}} \{P'_i(k)\}$$

where $P'_i(j)$ is the j^{th} individual in population P'_i. That is, we select the N_b first (best) individuals and the N_w last (worst) solutions in P'_i.

5 Identifying Impactful Defenses

In this section, we explore defense configurations that make the system harder to attack, in the sense that the best attacker – obtained with IEGA – needs more resources to achieve an attack. More precisely, we aim at identifying the defense actions that have the largest impact on the attack cost.

To fulfill this goal, we propose the Impact-Optimal Defense (IO-Def) heuristic that evaluates the impact of the defenses on the attack cost. A naive approach would be to enable all the defense actions. However, some of them may not significantly increase the attack cost. A more pragmatic approach is to look for a good balance between defenses and their provided impact on the attack cost. This is particularly important if the organization's defense budget is limited.

The heuristics implicitly builds an exploration tree where the root is the defense configuration $D_1^1 = \Sigma_D$. The defense D_j^i at the i^{th} level of the tree is obtained by disabling the defense action j that was available in its parent node. For example, the third child of D_1^1 is $D_3^2 = D_1^1 \setminus \{d_3\}$, where $d_3 \in D_1^1$. Each defense configuration D_j^i is characterized by the cost C_j^i (and the success probability P_j^i) of the optimized attacker against it, obtained with IEGA. The tree is explored in a breadth-first order. For each level $i > 1$, we identify the defense configuration with the minimal impact on the attack cost, and select

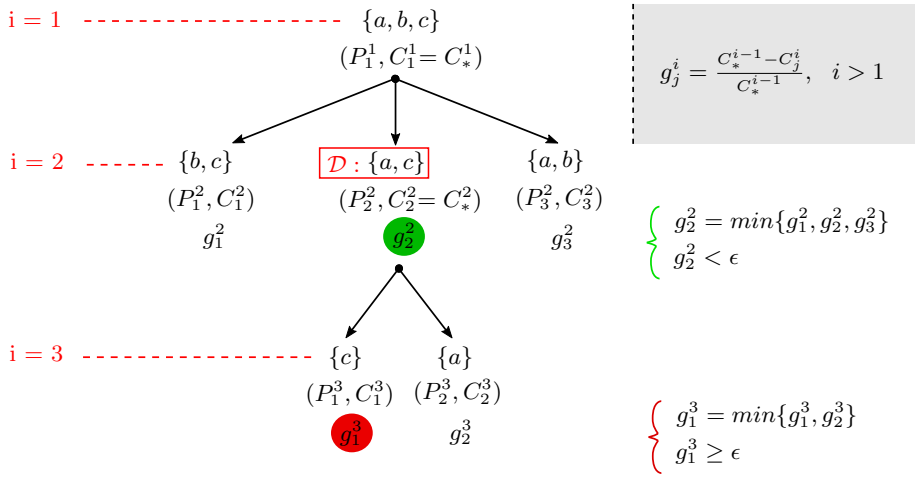


Fig. 3: Illustration of IO-Def method

it for further exploration in the case its impact is lower than ϵ. The impact g_j^i is a measure that scores a defense D_j^i by computing the relative decrease in the attack cost due to the deactivation of the j^{th} defense. It is defined as $(C_*^{i-1} - C_j^i)$ / C_*^{i-1}, where C_*^{i-1} is the attack cost of the selected parent node. The exploration ends whenever all the impacts of level $i+1$ are greater than or equal ϵ, or no more defenses are available, i.e., $D_1^{i+1} = \emptyset$. Finally, the most impactful defense configuration $\mathcal{D}$ is the one in which no defense can be disabled.

Figure 3 illustrates the exploration of the best defense configuration given three defense actions $\Sigma_D = \{a, b, c\}$, using IO-Def. In this example, the three defense actions are initially enabled, represented in the root node (level $i = 1$). Then we disable one defense action at a time, resulting to three new defense configurations $\{a, b\}$, $\{a, c\}$ and $\{b, c\}$, that constitute level $i = 2$. Their impacts are then computed and compared to identify the smallest value, in this case g_2^2. Since $g_2^2 < \epsilon$, $\{a, c\}$ is selected as the new best defense and the exploration is continued from it. Again, we disable defenses one by one to generate defense configurations of level $i = 3$, and g_1^3 is identified as the smallest impact value. This impact value $g_1^3 \geq \epsilon$ leads to the end of the exploration and $\mathcal{D} = \{a, c\}$ is recognized as the most impactful defense configuration.

Algorithm 1 presents the IO-Def algorithm, that identifies the subset $\mathcal{D}$ of the defense actions Σ_D such that the individual impact of each enabled defense is above a given threshold ϵ. In the worst case, the while loop is executed $n + 1$ times where $n = |\Sigma_D|$. An iteration computes $m + 1$ cost-effective attack strategies using IEGA, where $m = |\mathcal{D}|$ (initially n), and evaluates the impact of defenses. Remark that each iteration decreases m by one (the deactivated defense action). Hence, IEGA is executed, at worst case, $\frac{(n+1)(n+2)}{2}$ times. This is the case when all the available defenses fail to prevent the cost-minimal attack scenario. Therefore they are all disabled.

Data: a set of defense actions Σ_D, a threshold ϵ
Result: the optimal subset $\mathcal{D}$ of enabled defense actions
$\mathcal{D} = \Sigma_D$;
Boolean *improved* = true;
Integer i = 1;
while *improved* **do**
  i++;
  improved = false;
  Compute the minimal attack cost C_*^{i-1} for the defenses $\mathcal{D}$ using $IEGA$;
  foreach $d_j \in \mathcal{D}$ **do**
    Compute the minimal attack cost C_j^i for the defenses $\mathcal{D} \setminus \{d_j\}$ using $IEGA$;
    Compute the impact $g_j^i = \frac{C_*^{i-1} - C_j^i}{C_*^{i-1}}$;
  end
  Find the defense $d_{min} \in \mathcal{D}$ having the lowest impact g_{min}^i ;
  if $g_{min}^i < \epsilon$ **then**
    $\mathcal{D} = \mathcal{D} \setminus \{d_{min}\}$;
    improved = true;
  else
    return $\mathcal{D}$;
  end
end

Algorithm 1: Impact-Optimal Defense heuristic for defense exploration

6 Experimental results

In this section we report the results of the experiments performed using the attack strategy exploration approach (IEGA) and the defense exploration heuristic (IO-Def). The considered case studies address security issues at the level of organizations (ORGA, MI), a gateway protocol (BGP), and a sensor network infrastructure (SCADA). We also present a comparison with the state-of-the-art technique STRATEGO [**?**]. Obtained results show significant improvement.

6.1 Experimental settings

We perform experiments using IEGA and IO-Def on each of the four case studies. The considered setup in each case is as follows:

- **IEGA.** For each case study, we fix a defense configuration and we apply IEGA in order to identify a near-optimal attack strategy. This is done for all the possible defense configurations. For each configuration, we perform 25 runs of IEGA and measure the expected cost and the success probability of an attack. The obtained results are summarized as the average $\bar{x}$ and the standard deviation σ over the 25 runs. For all the case studies, the cost and time constraints are respectively set to $c_{max} = 50000$ and $t_{max} = 300$.
- **IO-Def.** The IO-Def heuristic is used on the four case studies with a threshold $\epsilon = 0.05$.

6.2 Results for the attack strategy exploration

Case study	Defenses	$\bar{x}_{cost}$	σ_{cost}	$\bar{x}_p$	σ_p	Runtime(s)
BGP	au rn sr	50000	0.00	0.00	0.00	2.65
	au sr	50000	0.00	0.00	0.00	2.54
	rn sr	50000	0.00	0.00	0.00	2.71
	au rn	284.31	2.83	1.00	0.00	3.95
	au	285.00	2.38	1.00	0.00	4.02
	sr	428.95	3.60	1.00	0.00	4.99
	rn	284.45	1.97	1.00	0.00	3.93
	none	283.96	1.94	1.00	0.00	4.09
SCADA	sw rst1 rst2 rst3	327.71	3.85	1.00	0.00	40.74
	sw rst1 rst2	328.68	3.61	1.00	0.00	39.49
	sw rst1 rst3	328.69	3.00	1.00	0.00	41.63
	sw rst2 rst3	329.20	3.20	1.00	0.00	42.63
	rst1 rst2 rst3	328.57	2.87	1.00	0.00	42.67
	sw rst1	328.09	3.63	1.00	0.00	39.46
	sw rst2	328.48	3.07	1.00	0.00	38.32
	sw rst3	328.29	3.29	1.00	0.00	39.90
	rst1 rst2	327.87	2.91	1.00	0.00	41.68
	rst1 rst3	328.52	4.47	1.00	0.00	39.43
	rst2 rst3	327.78	3.68	1.00	0.00	39.20
	sw	329.03	4.16	1.00	0.00	38.64
	rst1	327.96	3.43	1.00	0.00	39.29
	rst2	326.60	4.38	1.00	0.00	40.26
	rst3	326.95	3.32	1.00	0.00	42.30
	none	330.21	3.11	1.00	0.00	41.35
MI	dva tpt	328.83	3.53	1.00	0.00	49.62
	dva	163.04	3.66	1.00	0.00	48.60
	tpt	331.08	3.42	1.00	0.00	47.84
	none	159.85	2.69	1.00	0.00	49.26

Table 1: IEGA results with various defense configurations on BGP, SCADA and MI case studies

We report, in Table 1, the results of IEGA on BGP, SCADA and MI case studies. In this table, columns correspond to, respectively, the name of the case study, the deployed defense, the average $\bar{x}_{cost}$ and the standard deviation σ_{cost} of the attack cost, the average $\bar{x}_p$ and the standard deviation σ_p of the probability of success, and the execution time of IEGA.

For BGP, we can see that the best attack strategy against the three first

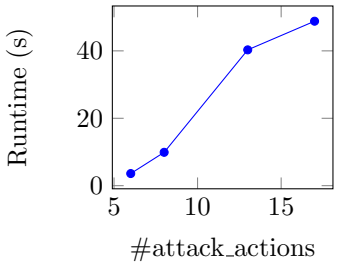


Fig. 4: IEGA runtime variation

defense configurations lead to a rejected cost, equal to c_{max} and a success probability of zero. This shows that these defense configurations cover all the possible attack scenarios. In the other hand, strategies with lower attack cost can be found in the case of other defense configurations. Considering SCADA, we notice that the computation of the minimal cost results to the same solution for every defense configuration. This can be explained by the existence of a low cost strategy that can always be taken, independently of the implemented defense. Regarding MI, the minimal cost varies depending on the defenses. However, defense action *dva* only brings a slight change in the attack cost.

In addition to the previous results, we also analyze the runtime performance of IEGA. We can see that it increases linearly with respect to the size of Σ_A, i.e., the number of available attack actions, as illustrated in Fig. 4. On the latter, we can see that the runtime on BGP (6 actions) is in average $3.6s$ and moves up to $9.9s$ on ORGA (8 actions), and keeps growing linearly on SCADA (resp. MI) to average $40.8s$ (resp. $48.8s$).

In table 2, we compare our results to STRATEGO [**?**] on the ORGA case study. This comparison is quantified using a cost improvement measure as follows:

$$Improvement = \frac{\bar{x}'_{cost} - \bar{x}_{cost}}{\bar{x}'_{cost}}$$

where $\bar{x}'_{cost}$ (respectively $\bar{x}_{cost}$) is the minimal cost returned by STRATEGO (respectively IEGA). We report this improvement measure in the last column of table 2 (the other columns are similar to Table 1).

The obtained results (summarized in the improvement column in Table 2) show that our method is able to find attack strategies with lower attack costs while respecting the time and cost constraints, in an average runtime of $9.9s$. Also, the best solution returned by IEGA does not drastically change from an execution to another, as explained by the small standard deviations.

We can see that varying the enabled defenses significantly affects the minimal attack cost. However, only disabling the defense *at* does not degrade the system security. This can be explained by the fact that the attack scenarios blocked by defense action *at* are already blocked by $t2$ (see Appendix A).

6.3 IO-Def heuristic

Tables in Fig. 5 summarize the most impactful defense configurations we identified for the four considered case studies. Rows in these tables represent, respectively, the possible defense actions, their status (on/off) and impact on the attack cost and finally IO-Def exploration time.

		IEGA					STRATEGO	Improvement
		$\bar{x}_{cost}$	σ_{cost}	$\bar{x}_p$	σ_p	Runtime(s)	$\bar{x}'_{cost}$	
Defenses	t1 t2 tf at	968.08	5.30	1.00	0.00	9.6	1038.33	0.07
	t2 tf at	237.97	1.39	1.00	0.00	10.2	410.52	0.42
	t1 t2 at	238.37	1.55	1.00	0.00	10.6	309.35	0.23
	at t2	237.92	1.27	1.00	0.00	10.1	359.48	0.34
	t1 tf t2	967.05	7.90	1.00	0.00	9.8	1000.90	0.03
	tf t2	238.18	1.58	1.00	0.00	10.2	288.53	0.17
	t1 t2	238.20	1.29	1.00	0.00	10.2	295.70	0.19
	t2	238.21	1.59	1.00	0.00	10.6	298.67	0.20
	t1 tf at	96.19	1.14	1.00	0.00	9.4	112.17	0.14
	tf at	96.04	1.08	1.00	0.00	9.7	103.37	0.07
	t1 at	96.35	0.98	1.00	0.00	9.5	133.60	0.28
	at	96.15	0.98	1.00	0.00	9.4	110.00	0.13
	t1 tf	96.08	1.29	1.00	0.00	9.8	121.07	0.21
	tf	96.27	1.14	1.00	0.00	9.8	105.97	0.09
	t1	95.99	0.67	1.00	0.00	9.4	109.33	0.12
	none	96.48	0.91	1.00	0.00	10.2	110.57	0.13

Table 2: IEGA results with various defense configurations on ORGA benchmark

We recall that the decision of identifying a defense action to be impactful or not, is done by comparing its impact to the threshold $\epsilon = 0.05$. The best defense configuration for ORGA (Table 5a) is $\mathcal{D} = \{t1, t2, tf\}$. In this setting, the role played by *at* was found to be negligible, while the highest impact (+90%) is brought by $t2$. The exploration results for BGP (Table 5b) show that the deployment of both *rn* and *sr* defenses is mandatory. Both of them have an impact of +99%, i.e. , disabling any of them lead to a heavy decrease of the attack cost. In contrast, in the case of SCADA (Table 5c), none of the defenses has a significant impact on the attack cost. Basically, this means that the available defenses are useless against the synthesized cost-effective attack strategy. Table 5d shows the best defense obtained in the MI case study. In this defense configuration, only *tpt* plays a significant role in increasing the attack cost, with a +50% impact.

In addition to the previous results, we report in the tables the exploration time. The main observation is that the exploration time does not only depend on the size of Σ_D. In fact, despite the fact that ORGA and SCADA have the same number of defense actions, they are explored in significantly different amounts of time (1min 25s and 9min 11s, respectively). This is due to the inefficient available defense actions in the case of SCADA, leading to the worst case execution time of IO-Def where all the defenses have to be disabled. Moreover, although MI has the smallest number of defense actions to explore, it is nevertheless not the fastest exploration. This is explained by the time required for a single run of the IEGA algorithm (in average 48.8s) in comparison to the cases of ORGA and BGP (resp. average 3.6s and 9.9s).

Defense Actions	t1	t2	tf	at
Defense status	On	On	On	Off
Cost gain	+75%	+90%	+75%	-
Exploration time	1min 25s			

(a) Results on ORGA

Defense Actions	au	rn	sr
Defense status	Off	On	On
Cost gain	-	+99%	+99%
Exploration time	23s		

(b) Results on BGP

Defense Actions	sw	rst1	rst2	rst3
Defense status	Off	Off	Off	Off
Cost gain	-	-	-	-
Exploration time	9min 11s			

(c) Results on SCADA

Defense Actions	dva	tpt
Defense status	Off	On
Cost gain	-	+50%
Exploration time	4min 7s	

(d) Results on MI

Fig. 5: Results obtained with IO-Def on different case studies

7 Conclusion

In this paper we presented a method for identifying impactful defense actions with respect to sophisticated attack strategies. Our proposal relies on two new heuristics. The first is a bi-objective method to synthesize a cost-effective attacker strategy given a risk assessment model. The second heuristic allows to find the defense configuration with the biggest impact on the attack cost.

It is worth mentioning that IO-Def can be adapted for risk assessment from the defense perspective. This can be easily done by extending the heuristic to consider a maximal defense budget, which allows to make a more realistic analysis. Other criteria, such as the return on investment (ROI) [**?**], can be also used to evaluate defense actions. Another investigation would be to synthesize attack strategies for more detailed models, where vulnerabilities and nominal behavior are explicitly described.

References

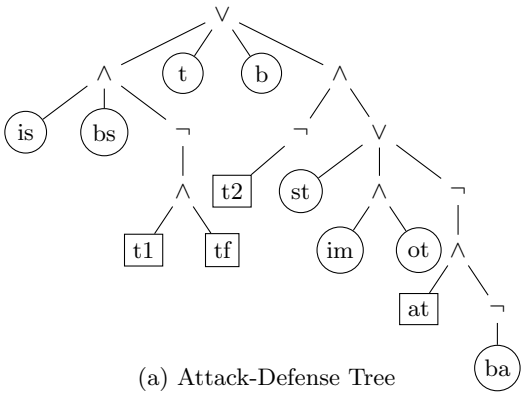
1. G. H. Baker and A. Berg. Supervisory control and data acquisition (scada) systems. *The Critical Infrastructure Protection Report*, 1(6):5–6, 2002.
2. J. W. Butts, R. F. Mills, and R. O. Baldwin. Developing an insider threat model using functional decomposition. In *International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security*, pages 412–417. Springer, 2005.
3. S. Convery, D. Cook, and M. Franz. An attack tree for the border gateway protocol. *Cisco Internet Draft*, 2002.
4. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *International Conference on Parallel Problem Solving From Nature*, pages 849–858. Springer, 2000.

5. K. S. Edge, G. C. Dalton, R. A. Raines, and R. F. Mills. Using attack and protection trees to analyze threats and defenses to homeland security. In *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pages 1–7. IEEE, 2006.
6. O. Gadyatskaya, R. R. Hansen, K. G. Larsen, A. Legay, M. C. Olesen, and D. B. Poulsen. Modelling attack-defense trees using timed automata. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 35–50. Springer, 2016.
7. T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate Probabilistic Model Checking. In *International Conference on Verification, Model Checking, and Abstract Interpretation, VMCAI'04*, pages 73–84, January 2004.
8. H. Ishibuchi and T. Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. volume 28, pages 392–403. IEEE, 1998.
9. B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer. Foundations of attack–defense trees. In *International Workshop on Formal Aspects in Security and Trust*, pages 80–95. Springer, 2010.
10. S. Mauw and M. Oostdijk. Foundations of attack trees. In *International Conference on Information Security and Cryptology*, pages 186–198. Springer, 2005.
11. B. L. Mediouni, S. Niar, R. Benmansour, K. Benatchba, and M. Koudil. A bi-objective heuristic for heterogeneous mpsoc design space exploration. In *Design & Test Symposium (IDT), 2015 10th International*, pages 90–95. IEEE, 2015.
12. A. Roy, D. S. Kim, and K. S. Trivedi. Attack countermeasure trees (act): towards unifying the constructs of attack and defense trees. *Security and Communication Networks*, 5(8):929–943, 2012.
13. P. Wang, W.-H. Lin, P.-T. Kuo, H.-T. Lin, and T. C. Wang. Threat risk analysis for cloud security based on attack-defense trees. In *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*, volume 1, pages 106–111. IEEE, 2012.
14. E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. volume 103. Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK), 2001.

A Benchmarks Description

In the following case study descriptions, attack actions are characterized by their lower (LB) and upper (UB) time bounds, the required resources (Cost) and their probability to succeed (Env). In the ADTs, attack actions are represented by ellipses and defense actions by rectangles.

A.1 An organization system attack (ORGA) [?]



(a) Attack-Defense Tree

Action	LB	UB	Cost	Env
Identify Subject (is)	0	20	80	0.8
Bribe Subject (bs)	0	20	100	0.7
Threaten (t)	0	20	700	0.7
Blackmail (b)	0	20	700	0.7
Send false Tag (st)	0	20	50	0.5
Break Authentication (ba)	0	20	85	0.6
Infiltrate Management (im)	0	20	70	0.5
Order Tag replacement (ot)	0	20	0	0.6

(b) Attack actions characteristics

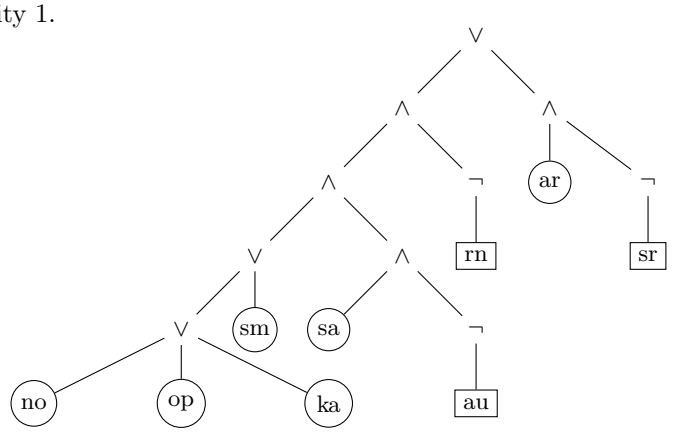
Defense action	Label
t1	Training for thwart
tf	Threaten to Fire employees
t2	Training for trick
at	Authenticate Tag

(c) Defense actions labels

Fig. 6: ORGA case study description

A.2 Resetting a BGP session (BGP) [?]

We constructed this case study based on [?], in which detection and mitigation events are attached with success probabilities (resp. P_D and P_M). We transpose these probabilities to the attack actions in a straightforward manner: the probability of an attack action to succeed is computed as the probability that all the implemented countermeasures set to block it, fail. For example, the attack action sa can be blocked by both defense actions au and rn. So, the probability of sa to succeed equals $Env(sa) = (1 - P_{D1} \times P_{M1}) \times (1 - P_{D2} \times P_{M2})$, where P_{D1}, P_{D2}, P_{M1} and P_{M2} are given in [?]. Note that, in our case, a pair of detection-mitigation events is combined is a single defense action. For example, P_{D1} and P_{M1} are merged into a defense au, and, P_{D2} and P_{M2} into the defense action rn. Also, the defense mechanisms are fixed before starting an analysis and have a probability 1.



(a) Attack-Defense Tree

Action	LB	UB	Cost	Env
Send RST message to TCP stack (sm)	0	20	50	0.7
Send BGP message: notify (no)	0	20	60	0.7
Send BGP message: open (op)	0	20	70	0.7
Send BGP message: keep alive (ka)	0	20	100	0.7
TCP sequence number attack (sa)	0	20	150	0.42
Alter config. via router (ar)	0	20	190	0.65

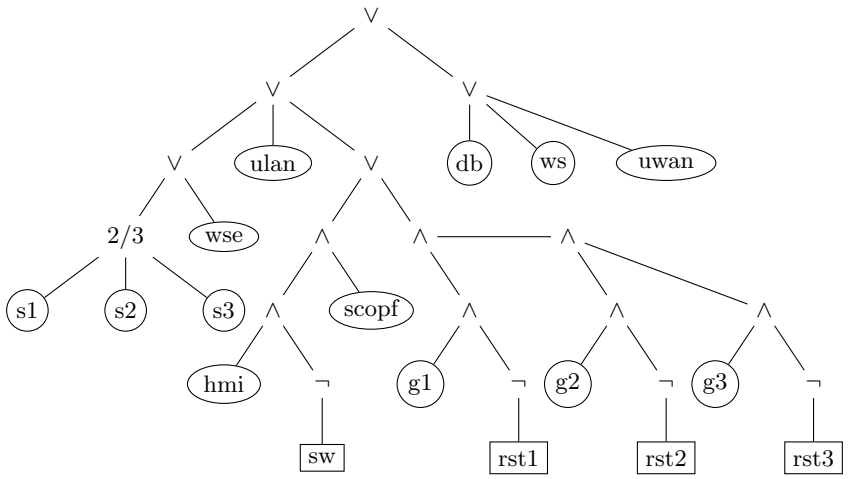
(b) Attack actions characteristics

Defense action	Label
au	Check TCP sequence number by MD5 authentication
rn	Check Trace-route by using randomized sequence numbers
sr	Secure routers with firewall alert

(c) Defense actions labels

Fig. 7: Resetting a BGP session description

A.3 Supervisory Control And Data Acquisition system (SCADA)[?]



(a) Attack-Defense Tree

Action	LB	UB	Cost	Env
Sensor one (s1)	0	20	100	0.1
Sensor two (s2)	0	20	110	0.1
Sensor three (s3)	0	20	90	0.1
Wrong estimation (wse)	0	20	250	0.25
Unavailable network LAN (ulan)	0	20	275	0.3
Control server one (hmi)	0	20	100	0.15
Control server two (scopf)	0	20	120	0.15
Controlling agent one (g1)	0	20	100	0.09
Controlling agent two (g2)	0	20	30	0.15
Controlling agent three (g3)	0	20	40	0.08
Database (db)	0	20	170	0.5
Unavailable network (uwan)	0	20	160	0.35
Workstation (ws)	0	20	150	0.4

(b) Attack actions characteristics

Defense action	Label
sw	Switch
rst1	Restart agent one if an attack is detected on it
rst2	Restart agent two if an attack is detected on it
rst3	Restart agent three if an attack is detected on it

(c) Defense actions labels

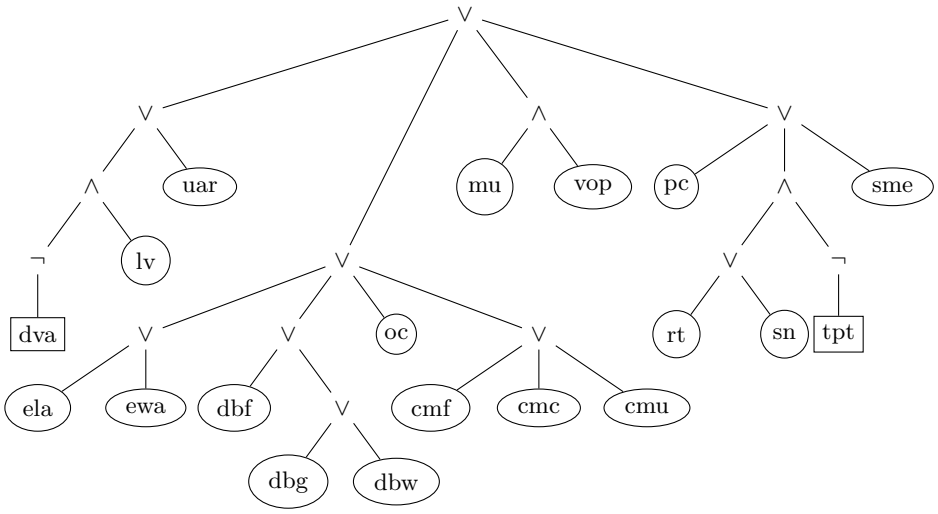
Fig. 8: SCADA system description

Similarly to BGP, SCADA is inspired from [**?**]. This case study represents an example of how attack trees are used to answer the failure assessment problem where attack actions represent the possible hardware/software failures. Since we are interested to identify what an attacker can do to reach a malicious goal on a system, we then interpret these attack actions as an attacker trying to trigger a hardware/software failure. So, in addition to the transposition from probabilities of successful defenses to probabilities of successful attack actions, Env also scales with the probability of failures. For example, the probability of $g1$ to succeed, provided it is guarded by a defense $rst1$, is computed as: $Env(g1) = P_{g1} \times (1 - P_D \times P_M)$, where the probabilities of a failure of the controlling agent one P_{g1}, the detection of its failure P_D and its restarting P_M are given in [**?**].

In figure 8a, the operator "2/3" is a shortcut designating the case where at least two events s_i and s_j occur, with $i \neq j$. It is equivalent to the boolean expression $\phi = (s1 \wedge s2) \vee (s1 \wedge s3) \vee (s2 \wedge s3)$.

A.4 A Malicious Insider attack (MI) [**?**]

In what follows, we describe a Malicious Insider attack (MI). It is presented in [**?**] and is adapted to our context in a similar way to BGP.



(a) Attack-Defense Tree

Action	LB	UB	Cost	Env
Unauthorized alternation of registry (uar)	0	20	50	0.08
Launch virus (lv)	0	20	60	0.07
Email local account (ela)	0	20	70	0.15
Email web-based account (ewa)	0	20	100	0.2
Drop-box: FTP to file server (dbf)	0	20	150	0.1
Drop-box: post to new group (dbg)	0	20	190	0.4
Drop-box: post to website (dbw)	0	20	100	0.1
Online chat (oc)	0	20	110	0.1
Copy to media: Floppy disk (cmf)	0	20	90	0.1
Copy to media: CD-ROM (cmc)	0	20	250	0.25
Copy to media: USB drive (cmu)	0	20	275	0.3
Misuse (mu)	0	20	100	0.2
Violation of organization policy (vop)	0	20	120	0.15
Poor configuration (pc)	0	20	100	0.15
Sniff Network (sn)	0	20	30	0.18
Root Telnet (rt)	0	20	40	0.12
Sendmail exploit (sme)	0	20	170	0.5

(b) Attack actions characteristics

Defense action	Label
dva	Detect viruses with anti-virus
tpt	Track number of tries at password

(c) Defense actions labels

Fig. 9: A Malicious Insider attack (MI) description