Mitigating Security Risks through Attack Strategies Exploration

B. L. Mediouni[1], A. Nouri[1], M. Bozga[1], A. Legay[2], and S. 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 sizes and 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 defense configurations to counter sophisticated attack strategies minimizing resource usage while ensuring a high probability of success. For this, we combine Statistical Model Checking techniques with Genetic Algorithms. Experiments performed on real-life case studies show substantial improvements compared to existing techniques.

1 Introduction

Modern organizations strongly rely on information and communication technologies in their daily activities. This reliance raises serious questions about the security threats that may be occasioned because of their inherent vulnerabilities and the way to mitigate the risks accompanying them. The damages that a cyberattack exploiting such vulnerabilities might cause, e.g. [7], highlight the urgent need for organizations to integrate risk assessment activities as part of their main processes. Risk assessment consists of analyzing and evaluating systems vulnerabilities in order to design reliable security policies.

Cyberattacks usually combine various techniques that exploit different vulnerabilities to circumvent deployed defense configurations. Such combinations are generally referred to as *Attack Strategies*. Reasoning at this level turns out to be more suitable than trying to fix individual vulnerabilities, especially since these are difficult to detect. *Offensive security* aims at identifying reliable defense configurations for a system by exploring attacks exploiting its vulnerabilities.

All is about resources. Both attack and defense actions require resources in order to be achieved. For instance attack actions require equipment and take time to be set up. Accordingly, they have some probability of success, i.e. actions that require a limited amount of resources generally have lower probability of success and conversely. Similarly, defense actions are subject to budgetary considerations (equipment, tools, training, etc.) and do generally provide overlapping protection

** Institute of Engineering Univ. Grenoble Alpes

mechanisms, hence they are not required to be deployed simultaneously. Therefore, it is primordial for organizations to be able to quantitatively analyze and evaluate potential defense actions in order to design configurations that prevent cyberattacks while involving a sufficient set of defense mechanisms.

Diverse attacker profiles can be observed in practice with regard to resources utilization. Some would settle for attack actions requiring limited resources, accepting a low probability of success, while others would privilege actions with high probability of success and allocate resources for that. These profiles are generally the product of various human factors such as experience, budget and motivations. A sophisticated attack strategy would try to optimize these criteria, namely, to find trade-offs requiring an affordable (within a given budget) amount of resources with an acceptable probability of success.

In this work, we propose a risk assessment approach that allows to synthesize defense configurations making sophisticated attacks harder to achieve. Concretely, we consider resources (e.g., the cost) required by an attack to be the hardness criterion. The rational is that since a sophisticated attack tries to optimize the cost with respect to the probability of success, defense actions that increase this cost are expected to prevent those attack strategies from being achieved with high probability. Relevant defense configurations are hence those involving a sufficient set of defenses with the highest impact on the attack cost.

As opposed to [5] that relies on reinforcement learning, our approach combines Statistical Model Checking (SMC) [6, 13] with Genetic Algorithms (GA) [10] to synthesize sophisticated attack strategies, which serve as a basis for exploring relevant defense configurations. The proposed approach considers Attack-Defense Tree [8] as a representation of the organization's security breaches, the potential attacks that could exploit them and the deployed defense configuration. Furthermore, the approach takes into account an attacker model that simulates arbitrary attack actions targetting the systems.

The remainder of the paper is organized as follows. We first discuss related work in Section 2. In Section 3, we formally introduce the considered models for risk assessment. The proposed techniques for attack strategies exploration and for the synthesis of an impactful defense configuration are respectively presented in Sections 4 and 5. In Section 6, we evaluate the proposed methods on four case studies. Finally, Section 7 concludes the paper and discusses future directions.

2 Related Work

Attack Trees (AT) [9] are widely used in security to model system vulnerabilities and the different combinations of threats to address a malicious goal. Attack-Defense Trees (ADT) [8] 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. More recently, Attack-Countermeasure Trees (ACT) [11] 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 [4] authors focus on the probabilistic analysis of ATs, through the computation of the probability, cost, risk and impact of an attack. A similar analysis is performed on ADTs in [12], called Threat Risk Analysis (TRA), and applied to the security assessment of cloud systems. In addition to the aforementioned probabilistic analysis, Roy et al. [11] make use of the structural and Birnbaum importance measure to prioritize attack events and countermeasures in ACTs.

Authors of [5] propose a reinforcement learning 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 also characterized by time duration as intervals. It identifies the sequence of attack actions and associated duration towards satisfying a specified time budget. However, it is not always the case that an attacker can control the duration of an attack action, eg. the time necessary for a brute-force attack. Instead, we consider time as a characteristic of an attack action, i.e., not 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 success. This strategy schedules attack actions and 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 an attacker and a 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, denoted env, the success probabilities of attack actions in Σ_A.

3.1 Attacker, Defender and Attack-Defense tree

Attacker. The attacker model represents all possible attack combinations, given the alphabet of attack actions 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 $\mathcal{A}$ performs a sequence of attack actions by choosing each time among the enabled ones. At a given state, an attack action a may succeed, leading to a new state where a is no more enabled [3] and where all other actions remain unchanged. In case a fails, the state of the attacker does not change. The success or failure of a selected attack action is not controlled by the attacker, but is determined by the environment env. We formally define the behavior of an attacker as follows.

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

- $S = L \times V_{\Sigma_A}$, *where $v \in V_{\Sigma_A}$ is a state vector that contains the status of all the attack actions in Σ_A (succeeded or not), i.e., $V_{\Sigma_A} = \{v : \Sigma_A \longrightarrow \{0,1\}\}$,*
- $s_0 = (l_0, v_0)$ *is the initial state, where $v_0 = [0, \ldots, 0]$ 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_i') respecting the following rules:*
  1. *Success:* $\frac{(l_i,\ a,\ l_i') \in T,\ v_i(a) = 0,\ v_i'(a) = 1,\ \forall a' \neq a\ v_i'(a') = v_i(a')}{((l_i,\ v_i),\ a,\ (l_i',\ v_i'))}$
  2. *Failure:* $\frac{(l_i,\ a,\ l_i') \in T,\ v_i(a) = 0}{((l_i,\ v_i),\ a,\ (l_i,\ v_i))}$

We use the notation $status(a, s)$ to denote the status of the attack action a at state $s = (l, v)$, i.e., $status(a, s) = status(a, (l, v)) = v(a)$.

Note that the attacker semantics above is non-deterministic, that is, the choice of an attack action at each state is performed non-deterministically. 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[4]. 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

$$P(s_i, a) = \begin{cases} 0 & \text{if } status(a, s_i) = 1 \\ \frac{\mathcal{S}(a)}{\sum\limits_{a' \in \Sigma_A} \mathcal{S}(a') \times (1 - status(a', s_i))} & \text{otherwise} \end{cases}$$

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.*

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

[3] This reflects a realistic behavior expressing the monotony of an attack.

[4] In this work, we restrict to static strategies, i.e., the same in any state. Considering dynamic strategies is a future work.

Attack-Defense Tree. It represents some knowledge about the system under 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 mechanisms available 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 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 considers the attacker and the defender models. This evaluation is performed as part of the risk analysis procedure based on a Risk Assessment Model introduced below.

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*.

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

- $\mathcal{A}|_{\mathcal{S}}$ *is an attacker following 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 the constraints c_{max} and t_{max} – on the system (abstracted by the environment env) against a fixed defense configuration (modeled by $\mathcal{D}$). 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 $true$ (respectively $false$) whenever $enabled(ap) = 1$ (respectively $enabled(ap) = 0$).
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 atomic proposition 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 following scenarios. 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 with 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 $cost(\omega) > c_{max}$ or $duration(\omega) > t_{max}$,
- 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 solutions for optimization and search problems. An important requirement to achieve an exploration is to be able to quantify solutions in order to establish an order over them. In this work, we rely on SMC to fulfill this goal as explained hereafter.

4.1 Overview

We consider as input a risk assessment model $\mathcal{M}$ 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 our approach (IEGA), an individual denoted $\mathcal{I} = \langle \mathcal{S}, cost, p \rangle$ is an attack strategy $\mathcal{S}$ annotated with an expected $cost$ and a probability p of success of

an attack when applying it. The *cost* and the probability p of success for an individual are computed using SMC. More precisely, the probability estimation algorithm (PESTIM) [6] is used 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 and its confidence are respectively controlled by the parameters δ and α.

It is worth mentioning that SMC is not only used in a passive way. In some cases, it can lead to update the strategy $\mathcal{S}$ when one or more primordial attack actions were assigned a zero-probability of selection, 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.

IEGA starts by randomly generating N initial strategies (individuals) to constitute the initial population P_0, evolving over M generations, as depicted in Fig. 1. 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 improving it by exploring its neighbor solutions,
4. **Mutation**: an individual has a p_m probability to be mutated, i.e., altering the selection probability of a randomly chosen attack action.

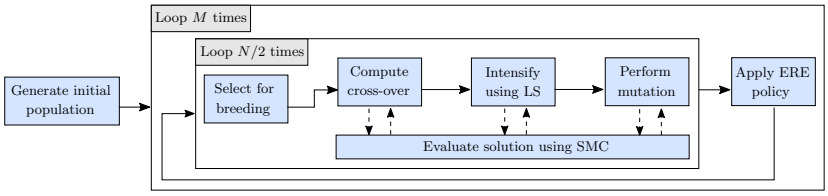


Fig. 1: Workflow of IEGA with a population of N individuals over M generations

The last phase of the outer loop (ERE) in Fig. 1 identifies among parent individuals in population P_i and their $N/2$ children, the ones to keep in the next generation $i + 1$. We use Extreme Ranking Elitism (ERE) [10] as a replacement policy, which 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 prevent early convergence.

In the next section, we further detail the cross-over, LS and ERE operations. Selection for breeding and mutation are both performed by random sampling in this work, and will therefore not be further detailed.

4.2 IEGA Operations Description

Cross-over operation. It consists of building a child $\mathcal{I} = \langle \mathcal{S}, cost, p \rangle$ 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., inherits the first half of its genes from $\mathcal{I}_1$ and the second half from $\mathcal{I}_2$ as follows:

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

Cross-over is followed by a normalization operation to ensure that the obtained 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 disabling a single attack action a_i, as follows:

- if $\mathcal{S}[i] = 1$ or $\mathcal{S}[i] = 0$ then the i^{th} neighbor individual $\mathcal{I}_i$ does not exist. In the first case, it is because a_i is the only enabled action and disabling it makes $\mathcal{S}$ an invalid mass function. In the second case, a_i is already disabled.
- otherwise, individual $\mathcal{I}_i$ is identified by a strategy $\mathcal{S}_i$ such that:

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

The normalization in the second case is again to ensure well-formedness of the synthesized strategy (probability mass function). It is worth mentioning that an individual has at most $|\Sigma_A|$ neighbors. Fig. 2 illustrates the computation of the neighbors of an individual with a scheduler $\mathcal{S} = [0.3, 0.5, 0.2]$, over 3 attack actions. For example, the first neighbor is obtained by disabling the first attack action in $\mathcal{S}_1$ and then normalizing it.

LS improves the current solution by repeatedly moving to better solutions 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, used for weighting and scaling the two parameters.

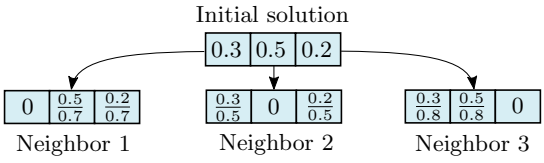


Fig. 2: Illustration of a neighborhood construction

ERE replacement policy A genetic algorithm maintains a population of size N over 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_j is known as dominated by another solution I_k if the latter is better for every criterion, in our case, $cost_j \geq cost_k \wedge p_j \leq p_k$ excluding the case where they are all equal. Considering this definition, the ranking consists of 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.
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. Therefore, 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 his attack. More precisely, we aim at identifying the defense actions that have the largest impact on the attack cost.

We propose a heuristic, denoted Impact-Optimal Defense (IO-Def), that evaluates the impact of the defenses on the attack cost. A naive approach to security would be to enable all available 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 impact on the attack cost. This is particularly important if the organization's defense budget is limited.

The heuristic implicitly builds an exploration tree where the root is the defense configuration with all the actions enabled, i.e., $D_1^1 = \Sigma_D$. The defense

D_j^i at the i^{th} level of the tree is obtained by disabling the defense action j that was enabled in its parent node. For example, the third child of D_1^1 is $D_3^2 = D_1^1 \setminus \{d_3\}$. Each defense configuration D_j^i is characterized by the cost C_j^i and the success probability P_j^i of the attack strategy 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 it for further exploration in the case its impact is lower than a given threshold 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. Algorithm 1 presents the IO-Def heuristic, that identifies the subset $\mathcal{D}$ of defense actions Σ_D such that the individual impact of each enabled defense is above ϵ.

Data: a set of defense actions Σ_D, a threshold ϵ
Result: the optimal subset $\mathcal{D}$ of enabled defenses
$\mathcal{D} = \Sigma_D$;
Boolean *improved* = true;
Integer i = 1;
while *improved* **do**
  i++;
  improved = false;
  Compute the minimal attack cost C_*^{i-1} againt $\mathcal{D}$ using $IEGA$;
  foreach $d_j \in \mathcal{D}$ **do**
    Compute the minimal attack cost C_j^i against $\mathcal{D} \setminus \{d_j\}$ 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

Fig. 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 ($i = 1$). Then we disable one defense action at a time, resulting in 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 resumed 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. However, in this case, $g_1^3 \geq \epsilon$, which leads to the end of the exploration. Therefore, $\mathcal{D} = \{a, c\}$ is considered to be the most impactful defense configuration.

In the worst case, Algorithm 1 executes the while loop $n + 1$ times where $n = |\Sigma_D|$. Each iteration computes $m + 1$ attack strategies using IEGA, where

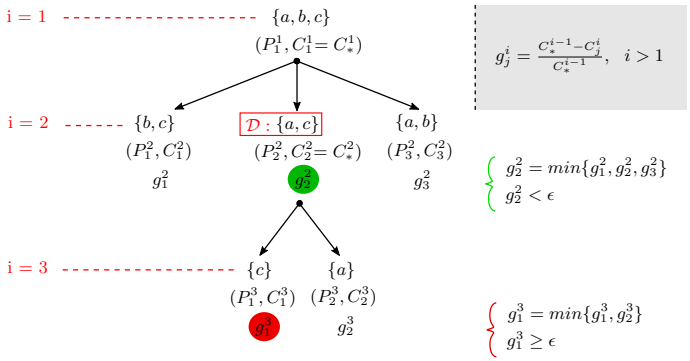


Fig. 3: Illustration of the IO-Def heuristic

$m = |\mathcal{D}|$ (initially $m = n$), and evaluates the impact of defenses. Remark that each iteration decreases m by one (the deactivated defense action). Hence, IEGA is executed, at worst, $\frac{(n+1)(n+2)}{2}$ times. This happens when all the available defenses fail to prevent the identified strategy. Therefore, they are all disabled.

6 Experiments

In this section, we present the experiments performed using IEGA and IO-Def heuristics. We considered case studies addressing security issues at the level of organizations (ORGA, MI), gateway protocols (BGP), and sensor network infrastructures (SCADA). Comparison with the state-of-the-art technique STRATEGO [5] shows that our approach performs better in most of the cases.

6.1 Overview and Experimental Setting

In our experiments, we considered four case studies briefly discussed below[5]:

1. ORGA. In this study, eight cyber and social attack actions can be combined to infiltrate an organization. To prevent such actions, the organization considers different defense actions, namely, train employees for thwart ($t1$) and for tricks ($t2$), threaten to fire them (tf) and authenticate tags (at).
2. Resetting a BGP session. In this case study, an attacker can execute six attack actions to reset a BGP session. The system is protected by three defense actions, i.e. check TCP sequence number by MD5 authentication (au), check trace-route by using randomized sequence numbers (rn), and secure routers with firewall alert (sr).
3. Supervisory Control And Data Acquisition system (SCADA). On these systems, the attacker tries to access some of the thirteen system components and

[5] Further details are provided in Appendix A

provoke hardware failures in order to disturb the system. The system considers four defense mechanisms: switch the Human-Machine Interface (sw) or restart one of the three system agents ($rst1$, $rst2$, $rst3$).

4. A Malicious Insider attack (MI). In this case study, an insider tries to attack an organization system from inside by exploiting seventeen identified vulnerabilities. The system sets up protections by deploying an anti-virus (dva) and a mechanism to track the number of tries on passwords (tpt).

Experimental approach. For each of the case studies, we performed two kind of experiments. In the first, we manually tried all the possible combinations of available defense actions, synthesized sophisticated attack strategies for them and evaluated their induced costs and probabilities of success. We proceeded as follows: each time, we fixed a defense configuration and applied IEGA in order to synthesize a near-optimal attack strategy. Since IEGA relies on SMC, which is an estimation technique, to synthesize strategies, we performed 25 runs of IEGA each time and measured the expected values and standard deviations of the cost and the probability of success (reported in Table 1). Furthermore, for this first experiment, we compared the results obtained by IEGA with the ones of STRATEGO on the ORGA case study. As stated earlier, our technique synthesizes better attack strategies in terms of cost as reported in Table 2.

The second kind of experiments aims at identifying the most impactful defense configurations against a near-optimal attack strategy obtained in the first experiments. To do so, we rely on the IO-Def heuristic that automatically explores the defense configurations as explained in Section 5. The results of this set of experiments are reported in Fig. 5.

For all the experiments, we considered the same budgetary constraints $c_{max} = 50000$ and $t_{max} = 300$ and we set the threshold $\epsilon = 0.05$ for the experiments with IO-Def. We also investigated the performance (exploration time) of the proposed heuristics (IEGA and IO-Def). We observed that IEGA shows a linear growth with respect to the size of Σ_A while IO-Def grows polynomially in the size of Σ_D.

6.2 Results and Discussion

Manual Exploration of Defenses. We first report in Table 1 the results of IEGA on the BGP, SCADA and MI case studies. In this table, the first column corresponds to the deployed defense configuration, the second and third columns report respectively the average cost $\bar{x}_{cost}$ over 25 runs of IEGA and standard deviation σ_{cost}, the last column shows the average execution time of IEGA. We omit reporting the average probability of success (resp. standard deviation) as it is always 1 (resp. 0) [6]. Note that for each study, we also investigated the setting where no defense action is deployed which allows to see the impact of different defense actions on the attack cost when enabled.

[6] Except for the first three cases in BGP where the probability of success is 0.

For BGP, we observed that the first three defense configurations lead inevitably to exceed the maximum allowed cost c_{max}. That is, no attack strategy can be synthesized within this budget, whereas in the case of the remaining defense configurations, strategies requiring lower cost can be synthesized. Moreover, one can see that the cost growth is minor when using *rn* or *au* compared to the case when no defense is used. For SCADA, we notice that the computation of the near-optimal strategy results almost in the same cost for all defense configuration. This can be explained by the existence of a low cost strategy that can always be applied, regardless of the implemented defenses. Furthermore, we observed that the cost induced by using any combination of defense actions does not significantly improve compared to the defenseless case. For MI, we obtained different costs depending on the defenses used. We noticed that defense action *dva* insignificantly increases the attack cost as opposed to *tpt*. The results for the ORGA case study are reported in Table 2 for the sake of comparison with STRATEGO. Except the last two columns, the table presents the same information as Table 1. For this study, we observed that varying the enabled defenses significantly affects the minimal attack cost and that the defense action *at* does not have a great impact on the cost. We actually observed that the attack strategies blocked by this defense action can be also blocked by *t*2.

Defense	$\bar{x}_{cost}$	σ_{cost}	Runtime (s)
BGP			
au rn sr	50000	0.00	2.65
au sr	50000	0.00	2.54
rn sr	50000	0.00	2.71
au rn	284.31	2.83	3.95
au	285.00	2.38	4.02
sr	428.95	3.60	4.99
rn	284.45	1.97	3.93
none	283.96	1.94	4.09
SCADA			
*sw rst*1 *rst*2 *rst*3	327.71	3.85	40.74
*sw rst*1 *rst*2	328.68	3.61	39.49
*sw rst*1 *rst*3	328.69	3.00	41.63
*sw rst*2 *rst*3	329.20	3.20	42.63
*rst*1 *rst*2 *rst*3	328.57	2.87	42.67
*sw rst*1	328.09	3.63	39.46
*sw rst*2	328.48	3.07	38.32
*sw rst*3	328.29	3.29	39.90
*rst*1 *rst*2	327.87	2.91	41.68
*rst*1 *rst*3	328.52	4.47	39.43
*rst*2 *rst*3	327.78	3.68	39.20
sw	329.03	4.16	38.64
*rst*1	327.96	3.43	39.29
*rst*2	326.60	4.38	40.26
*rst*3	326.95	3.32	42.30
none	330.21	3.11	41.35
MI			
dva tpt	328.83	3.53	49.62
dva	163.04	3.66	48.60
tpt	331.08	3.42	47.84
none	159.85	2.69	49.26

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

Detailed results regarding the runtime performance of IEGA are reported in the Table 1 and summarized in Fig. 4. The latter shows a linear evolution of the runtime

when increasing the size of Σ_A, i.e., the number of available attack actions. The measures in Fig. 4 correspond respectively to the average runtime on BGP (6 actions, 3.6s), ORGA (8 actions, 9.9s), SCADA (13 actions, 40.8s) and MI (17 actions, 48.8s). We also observed that IEGA shows a certain stability of the synthesized attack strategy over different runs as testified by the small standard deviation observed in the different experiments.

		IEGA			STRATEGO	Improvement
		$\bar{x}_{cost}$	σ_{cost}	Runtime (s)	$\bar{x}'_{cost}$	(%)
Defenses	$t1\ t2\ tf\ at$	968.08	5.30	9.6	1038.33	7
	$t2\ tf\ at$	237.97	1.39	10.2	410.52	42
	$t1\ t2\ at$	238.37	1.55	10.6	309.35	23
	$at\ t2$	237.92	1.27	10.1	359.48	34
	$t1\ tf\ t2$	967.05	7.90	9.8	1000.90	3
	$tf\ t2$	238.18	1.58	10.2	288.53	17
	$t1\ t2$	238.20	1.29	10.2	295.70	19
	$t2$	238.21	1.59	10.6	298.67	20
	$t1\ tf\ at$	96.19	1.14	9.4	112.17	14
	$tf\ at$	96.04	1.08	9.7	103.37	7
	$t1\ at$	96.35	0.98	9.5	133.60	28
	at	96.15	0.98	9.4	110.00	13
	$t1\ tf$	96.08	1.29	9.8	121.07	21
	tf	96.27	1.14	9.8	105.97	9
	$t1$	95.99	0.67	9.4	109.33	12
	none	96.48	0.91	10.2	110.57	13

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

Finally, we compared the results obtained by IEGA with STRATEGO [5] on the ORGA case study. Comparison results are shown in the last two columns of Table 2 which respectively present the average cost obtained using STRATEGO and the percentage of improvement provided by our approach. This improvement is measured as $\frac{\bar{x}'_{cost}-\bar{x}_{cost}}{\bar{x}'_{cost}}$ where $\bar{x}'_{cost}$ (respectively $\bar{x}_{cost}$) is the minimal cost returned by STRATEGO (respectively IEGA). The obtained results show that our method is able to find attack strategies with lower attack costs than STRATEGO within the specified cost budget. In this case study, the improvement induced by our approach –in term of cost reduction– compared to STRATEGO ranged from 3% to 42% depending on the deployed defense configuration.

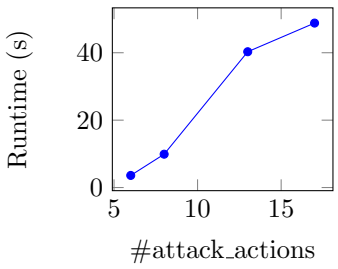


Fig. 4: IEGA runtime variation

Automatic Exploration of Defenses. We report in Fig. 5 exploration results using IO-Def for the different case studies. For each of them, we present the identified most impactful defense configuration $\mathcal{D}$ in a separate table showing respectively, the defense actions, their status (on/off), their impact on the attack cost (in percentage) in the context of $\mathcal{D}$ and the IO-Def exploration time.

Defense Actions	$t1$	$t2$	tf	at
Status	On	On	On	Off
Impact on cost	+75%	+90%	+75%	-
Exploration time	1min 25s			

(a) Results on ORGA

Defense Actions	au	rn	sr
Status	Off	On	On
Impact on cost	-	+99%	+99%
Exploration time	23s		

(b) Results on BGP

Defense Actions	sw	$rst1$	$rst2$	$rst3$
Status	Off	Off	Off	Off
Impact on cost	-	-	-	-
Exploration time	9min 11s			

(c) Results on SCADA

Defense Actions	dva	tpt
Status	Off	On
Impact on cost	-	+50%
Exploration time	4min 7s	

(d) Results on MI

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

We recall that identifying a defense action to be impactful or not, is done by comparing its impact to the threshold $\epsilon = 0.05$. We observed that 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 leads 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.

Regarding the exploration time of IO-Def, the main observation is that it does not only depend on the size of Σ_D but also on the nature of the system to explore and the IEGA runtime (i.e., the size of Σ_A). In spite of the fact that ORGA and SCADA have the same number of defense actions, they are explored in significantly different amounts of time (respectively 1min 25s and 9min 11s). This is due to the inefficient available defense actions in the case of SCADA, leading to the worst case exploration time of IO-Def where all the defenses have to be disabled. Moreover, even though MI has the smallest number of defense actions to explore, it is not the fastest. This is explained by the time required for a single run of the IEGA algorithm (48.8s in average) in comparison to the cases of ORGA and BGP (respectively 3.6s and 9.9s in average).

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 the IO-Def heuristic can be adapted for risk assessment from the defense perspective. This can be easily done by extending it to consider a maximal defense budget, which allows to make a more realistic analysis. Other criteria, such as the return on investment (ROI) [11], 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.

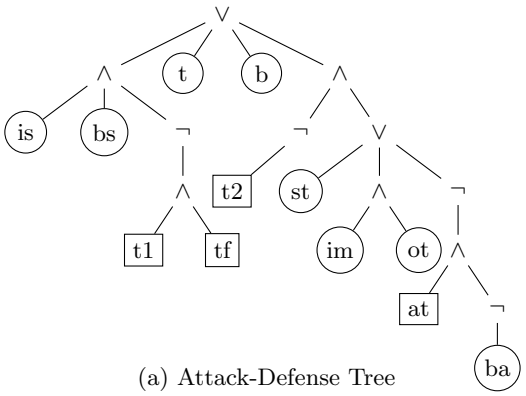
References

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 *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. 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.
5. 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 *Conference on Formal Modeling and Analysis of Timed Systems*, pages 35–50. Springer, 2016.
6. T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate Probabilistic Model Checking. In *VMCAI'04*, pages 73–84, 2004.
7. S. ICS. Analysis of the cyber attack on the ukrainian power grid, march 2016. Accessed on 25 April 2018.
8. 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.
9. S. Mauw and M. Oostdijk. Foundations of attack trees. In *International Conference on Information Security and Cryptology*, pages 186–198. Springer, 2005.
10. 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.
11. 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.
12. 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 *ICCM*, pages 106–111, 2012.
13. H. L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, Carnegie Mellon, 2005.

A Case Studies 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) [5]



(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

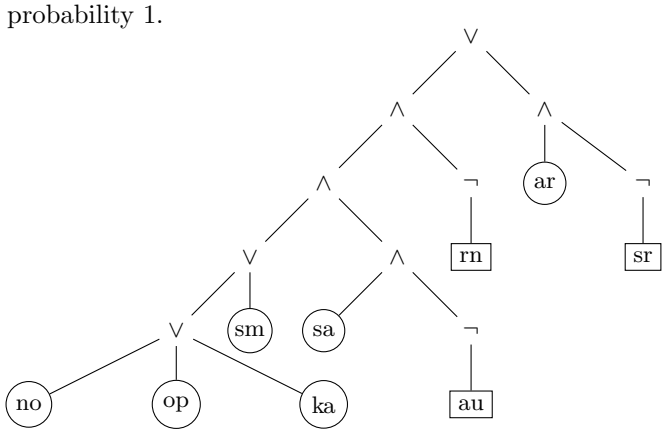
Defense action	Label
t1	Training for thwart
tf	Threaten to Fire employees
t2	Training for trick
at	Authenticate Tag

(c) Defense actions labels

Fig. 6: ORGA case study description

A.2 Resetting a BGP session (BGP) [3]

We constructed this case study based on [11], 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 [11]. 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

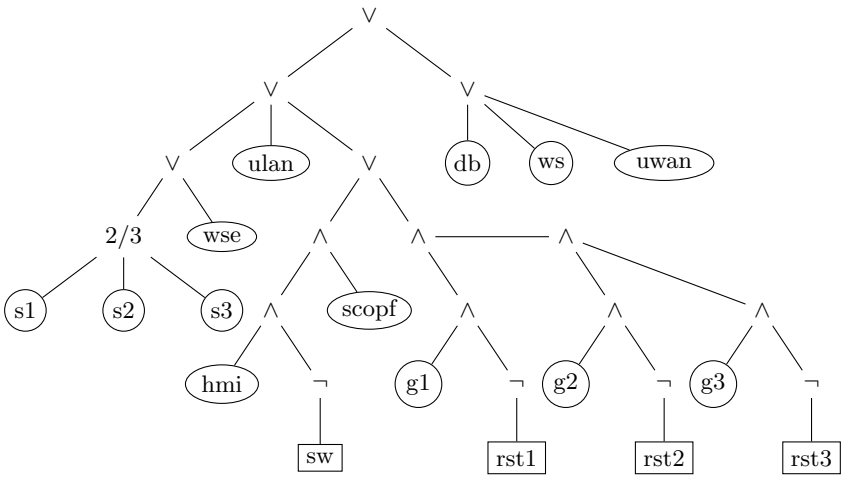
(b) Attack actions characteristics

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)[1]



(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

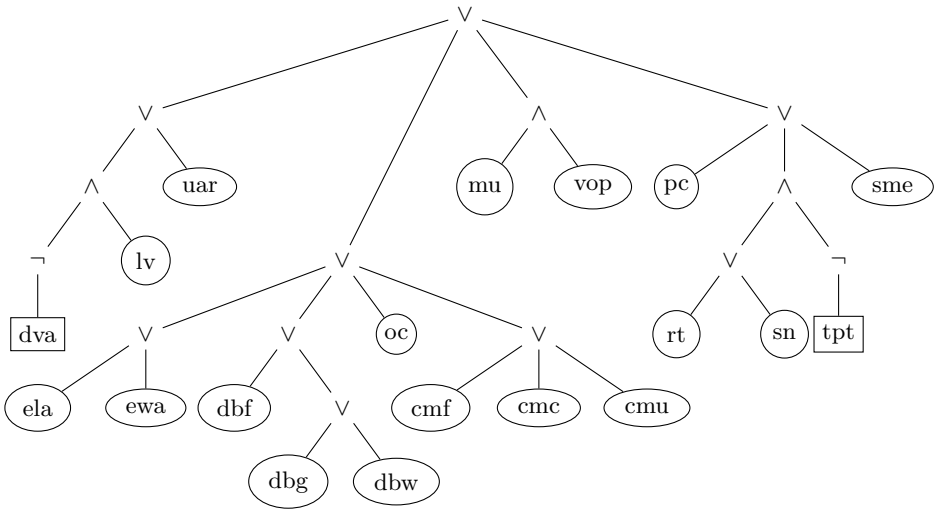
Fig. 8: SCADA system description

Similarly to BGP, SCADA is inspired from [11]. 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 [11].

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) [2]

In what follows, we describe a Malicious Insider attack (MI). It is presented in [11] 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