Quantitative Attack Tree Analysis via Priced Timed Automata

Rajesh Kumar(✉), Enno Ruijters, and Mariëlle Stoelinga

Formal Methods and Tools, University of Twente, Enschede, The Netherlands
r.Kumar@utwente.nl

Abstract. The success of a security attack crucially depends on the resources available to an attacker: time, budget, skill level, and risk appetite. Insight in these dependencies and the most vulnerable system parts is key to providing effective counter measures.

This paper considers attack trees, one of the most prominent security formalisms for threat analysis. We provide an effective way to compute the resources needed for a successful attack, as well as the associated attack paths. These paths provide the optimal ways, from the perspective of the attacker, to attack the system, and provide a ranking of the most vulnerable system parts.

By exploiting the priced timed automaton model checker Uppaal CORA, we realize important advantages over earlier attack tree analysis methods: we can handle more complex gates, temporal dependencies between attack steps, shared subtrees, and realistic, multi-parametric cost structures. Furthermore, due to its compositionality, our approach is flexible and easy to extend.

We illustrate our approach with several standard case studies from the literature, showing that our method agrees with existing analyses of these cases, and can incorporate additional data, leading to more informative results.

1 Introduction

Security attacks are a primary concern for business and government organizations, as they are a threat to vital infrastructure, such as internet banking, power grids, health care and transportation systems. The challenge for security engineers is to protect such systems, by providing countermeasures for the most damaging and most likely attacks. Thus, defense against cyberattacks is an optimization problem: given the available budget, what are the most effective countermeasures.

Often security decisions are made informally, e.g. by brainstorming. More structured approaches are based on spreadsheets and technical standards, like FMEA [11], the AS/NZS 4360 standard [5], and Factor Analysis of Information Risks (FAIR) [6]. Model-based approaches are gaining popularity, such as the UML-based extension CORAS [1], the ADVISE method [21], and security extension of the SAE standardized language AADL [17].

S. Sankaranarayanan and E. Vicario (Eds.): FORMATS 2015, LNCS 9268, pp. 156–171, 2015.
DOI: 10.1007/978-3-319-22975-1_11

One of the most prominent model-based security formalisms is attack trees (ATs), see Figure 1. Much of its popularity comes from its hierarchical, intuitive representation of multi-step attack scenarios. A wide range of qualitative and quantitative analysis methods for attack trees are available, see [20] for an overview. The most well-known follow a bottom-up approach and propagate values from the leaves to the top of the tree. Although they are very efficient and flexible, most current approaches cannot handle temporal and causal dependencies, or shared subtrees. Also, existing approaches do not support realistic cost structures. Finally, little attention is given to the important issue of attack path generation and ranking: which steps are taken in the most dangerous attacks?

This paper provides a multi-objective optimization framework for attack trees. We augment AT leaves with a rich cost structure that consists of various components, such as time, skill, and resources. These components can be dependent, e.g. time can depend on skill level. Our framework supports the computation of a wide range of security metrics: (1) *Attack values.* Given an attack tree, and a quantity of interest, we can compute its value: What is the minimal time, resources, or skill level needed to complete a successful attack? What is the maximal damage an attacker can do? (2) *Attack paths.* Apart from the value of an attack, it is very useful to know the attack path leading to the optimal attack. Note that the path is in fact a subtree, since optimal attacks often carry out several steps need in parallel. (3) *Ranking.* Apart from computing the optimal attack values and path, we can also determine the top-10 of worst attacks, which is very important to determine appropriate counter measures. (4) *Pareto-optimal curves* that show trade-offs when multiple objectives conflict. For instance, what is the minimal time needed to complete a successful attack within a given budget? What is the maximal damage that can be incurred in one year?

Technically, our framework is realized via Uppaal CORA: we translate each attack tree gate and leaf into a priced timed automata (PTA). Together these form a network of PTAs representing the entire attack tree. This modular approach yields a flexible framework that can easily be extended with future needs, such as countermeasures. We express our security queries in weighted CTL, and use the model checker Uppaal CORA [8] to obtain the cost optimal traces that correspond to optimal attack paths in the AT.

We illustrate our approach with several well-known examples in attack tree analysis, namely the forestalling release of software [18], obtaining administrator privileges [19] and a password protected file [25]. We provide the results from two perspectives: For the attacker, we consider cost and time, and for the attackee, the incurred damage. Also, we have considered several attacker profiles into consideration. Our analysis shows that the vulnerable paths in the system are strongly linked to the skills and risk appetite of attacker. Hence, any security risk analysis should be multifaceted, taking the potential attackers into consideration.

Related Work. Attack trees; as popularized by Schneier [27], were introduced by Weiss as threat logic trees [30] and by Amoroso as threat trees [3]. Amid several variants studied in literature, they can broadly be classified as

static [19,23,26] or *dynamic* [15,24] based on evolution of time. Classically, an attack tree takes a single parameter such as time or cost [23,27]. Buldas [14] et al. introduces a multiparameter attack tree consisting of interdependent parameters. In [22], Lenin et al., while making a clear distinction between threat and vulnerability landscape, improve the parallel model [18] by integrating attacker profiles. A comprehensive overview of attack trees can be found in [20]. Some other approaches to model the system description and attacker behaviour are via attack graphs [28] or adversary based security analysis [13,16].

2 Graphical Security Modeling

2.1 Attack Trees

Attack trees (ATs) are an important formalism to model and analyze the security of complex systems. An attack tree consists of a *root*, representing the attacker's goal. The root is further refined into subgoals via *gates*, until the subgoals cannot be refined further and the *basic attack steps (BASs)* are reached, constituting the leaves of the attack tree. When subtrees can be shared, ATs can be directed acyclic graphs, rather than trees.

Gates. Classical attack trees model the propagation of success through AND- and OR-gates: an AND-gate is a conjunctive composition of child nodes, indicating that all children need to be successfully attacked for an attacker to successfully execute the subgoal at hand. Similarly, the OR-gate is a disjunctive union of child nodes, where an attacker has to execute at least one child node successfully.

It has been widely recognized that temporal order is crucial in security. Therefore, the sequential versions of the AND and OR gates, named SAND and SOR, have been proposed [4,25]. Both represent attacks executed from left to right: Starting with the the leftmost child, the attacker will only start executing the next subgoal (i.e., the subsequent child node) after all previous subgoals have been executed successfully. The SAND gate is successful if all steps are executed successfully; the SOR-gate is successful if any of its children is executed successfully.

Example 1. The attack tree in Figure 1, combined with the values in Table 1, models the forestalling of the release of some software, adopted from [14]. Here, a competitor steals a piece of software code and then builds it into his own product, as modeled by the top-level SAND gate. The OR gate at node *Steal code* shows that the code can be stolen in three different ways: via *Bribing*, a *Network Attack* or *Physical robbery*. Bribing is modeled as a two-step sequential process of first successfully bribing a programmer and then obtaining the code, represented by a SAND-gate. Similarly, one can employ a robber who has networking knowledge. This can lead to two different attack paths modeled through a shared node. One in which the hired person finds a bug and exploits it to obtain the code via a network attack, and another path in which he is physically involved in a robbery after being hired to steal the code. This dependency is again modeled through a SAND gate.

Basic Attack Steps. Basic attack steps (BASs) represent individual atomic steps within a composite attack, and appear as leaves of the AT.

We consider a fixed set of attribute variables $\mathsf{Attr} = \{T, a_1, \ldots a_n\}$. Here, T is a special attribute, namely the time since the BAS was started. Other attributes can be skill level, monetary costs, damage, difficulty, etc. We denote by $Val = (\mathbb{R}^{\infty}_{\geq 0})^{n+1}$ the set of complete valuations of the attributes, and by $Val_{\setminus t} = (\mathbb{R}^{\infty}_{\geq 0})^{n}$ the set of valuations excluding time. For simplicity, we assume that attribute variables take values in $\mathbb{R}^{\infty}_{\geq 0}$; handling other domains is technically no more complex, but syntactically more cumbersome.

Each BAS is equipped with two preconditions $\mathit{Enable} : Val_{\setminus t} \to \{0, 1\}$ and $\mathit{CanSucceed} : Val \to \{0, 1\}$ that indicate when the BAS is enabled, and when it can succeed. Each BAS also has an effect *Eff* that updates the attribute values when the BAS is successfully executed. Preconditions are Boolean combinations over linear equations over Attr. In this way, an attack step that requires (at least) medium skill level is equipped with the enabling precondition $\mathit{Skill} \geq \mathit{med}$ (where *med* is a suitable constant); and an attack step that takes between 90 and 100 time units for medium-skilled attackers gets a success precondition $(\mathit{Skill} = \mathit{med}) \to (90 \leq T \leq 100)$.

The effect $\mathit{Eff} : \mathsf{Attr} \times Val_{\setminus t} \to \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a function that updates the values for the attributes when this BAS is started. For example, costs are incurred by the attacker, and damage is incurred by the attacked entity. These effects are typically time dependent: the longer an attack takes, the higher the costs and damage. We assume that time dependence is linear, i.e., is incurred with a fixed rate v_i per time unit. Thus, the effect function is given by $\mathit{Eff}(a_i, (p_1, \ldots, p_n))(t) = f_i + v_i \cdot t$, where $f_i = f_i(p_1, \ldots p_n)$ and $v_i = v_i(p_1, \ldots p_n)$ are parameters that depend on the attribute values. The effects are summed to the existing value of the variable, to obtain the cumulative effect.

Attacker Profiles. An attacker profile is an assignment $R : \{a_1, \ldots a_n\} \to \mathbb{R}_{\geq 0}$ of the non-time attribute variables to concrete values. Thus, we obtain an initial valuation of the attributes as $(0, R(a_1), \ldots, R(a_n))$.

Example. Consider an burglary that takes between 5 and 10 minutes to execute for a medium-skilled attacker, between 1 and 2 minutes for a highly

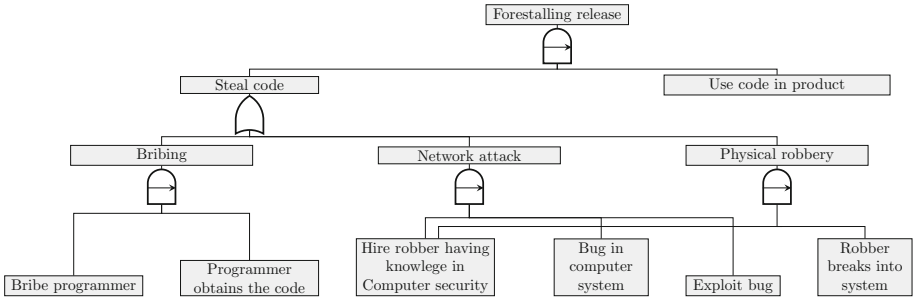


Fig. 1. Attack tree modeling the forestalling of software.

skilled attacker, and cannot be performed by a low-skilled attacker. All attackers steal 1000 dollars worth of goods if they are successful, but the medium-skilled attacker also inflicts 500 dollars of property damage in the process. This BAS can be described using the attribute set $\mathsf{Attr} = \{T, Skill, Damage\}$. It has enabling precondition $Enable(s, d) = s \geq med$ and success condition $CanSucceed(t, s, d) = (s = med \wedge 5 \leq t \leq 10) \vee (s = hi \wedge 1 \leq t \leq 2)$. The effect is $Eff(Damage, (s, d))(t) = 1000$ *if* $s = hi, 1500$ *otherwise* and $Eff(a, V)(t) = 0$ for all $a \neq Damage$ (here *med* and *hi* are appropriate constants).

Based on the explanation above, attack trees can be defined as follows.

Definition 1 (AT elements). *We define the set of AT gate types as* $\mathsf{Gates} = \{\mathsf{AND}, \mathsf{SAND}, \mathsf{OR}, \mathsf{SOR}\}$*, the set of BAS information as* BAI*, where each element of* BAI *is a triple* $(Enable, CanSucceed, Eff)$ *of the functions described above. We denote* $\mathsf{Elements} = \mathsf{Gates} \cup \mathsf{BAI}$.

Definition 2 (Attack tree). *An attack tree* A *is a tuple* $(V, \mathsf{Child}, \mathsf{Top}, \mathsf{Attr}, R, L)$*, where*

- V *is a finite set of nodes.*
- $\mathsf{Child} : V \to V^*$ *maps each node to its child nodes.*
- $\mathsf{Top} \in V$ *is the unique top level element, representing the goal of the attacker.*
- Attr *is the set of attributes.*
- R *is the attacker profile.*
- $\mathsf{L} : V \to \mathsf{Elements}$ *is a labelling function that assigns an AT element to each node in* V*.*

ATs must be well-formed. We define the set of *edges* of A by $E = \{(v, w) \in V^2 \mid \exists i \,.\, w = (\mathsf{Child}(v))_i\}$ and $\mathsf{Leaves} = \{v \in V \mid \mathsf{Child}(v) = \epsilon\}$. We require for each AT that (a) the graph (V, E) is a directed acyclic graph with a unique root $\mathsf{Top} \in V$ from which all other nodes are reachable; (b) the labelling function assigns to each leaf in the tree a value in BAI and to each non-leaf an element in Gates, i.e., $\mathsf{L}(v) \in \mathsf{BAI}$ iff $v \in \mathsf{Leaves}$.

Table 1. Values used for annotating leaves of Figure 1

	Attacker		Values		
BAS	Profile	Skill	Time (in days)	Cost (in US $)	Cost to company (in US $)
Bribe a programmer	Generic attacker	Low	15-20	1500 + 50t	500.000
	Generic attacker	Med	10-20	1000 + 150t	500.000
	Generic attacker	High	0-10	500	500.000
	Software Engineer	Any	0-5	5000 + 100t	500.000
Programmer obtains the code	Generic attacker	Any	5-15	1000 + 100t	1.000.000
	Software Engineer	Any	0-5	2000 + 50t	1.000.000
Hire robber with knowledge of computer security	Any	Any	5-15	4000 + 50t	0
Bug in Computer system	Any	Low	15-20	1000 + 50t	0
	Any	Med	5-10	1000 + 50t	0
	Any	High	0-5	1000 + 50t	0
Person exploits the bug	Any	Any	0-5	1000 + 50t	1.000.000
Person breaks into the system	Any	Any	0-5	2000 + 100t	400.000
Code is completed into product	Any	Any	5-15	2000 + 50t	100.000

2.2 Metrics on ATs

Our framework can be used to determine several important security metrics. (1) *(Constrained) attack values.* For any of the attributes a_i, we can compute the minimum value along the tree. These values can be affected by constraints on other attributes. For instance, we can compute the minimum time needed to complete an attack within a maximum budget and skill level. (2) *Pareto optimal curves.* For any pair of attributes, we can compute the minimum value needed of one attribute given a value of the other. By varying the bound of one attribute, we can generate curves indicating the relation between these minima. For instance, there is typically a trade-off between spending more time or more money; a Pareto curve shows for every budget how much time is needed for the attack. (3) *Attack paths.* When computing the minimal value of an attribute that can complete an attack, we generate a concrete attack path showing the steps an attacker can take to perform the attack incurring as little of the attribute as possible. For instance, considering Figure 1, to reach the goal in the minimum time, we can obtain the attack trace which consists of *Hire a robberer*, *Robberer breaks into system* and *Use code in product*. (4) *Ranking.* In addition to the single minimum of an attribute and a corresponding attack path, we can enumerate further attacks in increasing value of the attribute. We can, for example, list the ten cheapest attacks on a given system, or all attack paths that meet a given time constraint. For example, in Figure 1, with the attributes in Table 1, the optimal cost is 6000 units and the second best cost is 8500 units.

3 Priced Timed Automata

The Priced Timed Automata Model. Priced timed automata (PTA) [8] extend timed automata [2], by adding costs to locations and actions. In the following definition, we denote by $\Phi(X)$ the set of all possible boolean predicates over a set X of clocks.

Definition 3. *A priced timed automaton P is a tuple $\langle L, l_0, X, Act, E, I, C\rangle$ where:*

- *L is a finite set of* locations*,*
- *$l_0 \in L$ is the initial state,*
- *X is a set of clock variables,*
- *Act is a set of actions, also called signals or labels,*
- *$E \subseteq L \times \Phi(X) \times Act \times 2^X \times L$ gives the set of transitions. Here an edge $\langle l, \phi, a, \lambda, l'\rangle$ represents a transition from state l to state l' taking an action a. This transition can only be taken when the clock constraint ϕ over X is true, and the set $\lambda \subseteq X$ gives the set of clocks to be reset with this transition,*
- *$I : L \to \Phi(X)$ assigns invariants to locations,*
- *$C : L \cup E \to \mathbb{N}^n_{\geq 0}$ assigns cost rates to locations and costs to edges.*

Definition 4. *A trace of a PTA $P = \langle L, l_0, X, Act, E, I, C\rangle$ is a sequence of states and transitions $\rho = l_0 \xrightarrow[\lambda_0]{a_0 \xrightarrow{t_0}_{c_0}} l_1 \xrightarrow[\lambda_1]{a_1 \xrightarrow{t_1}_{c_1}} l_2 \ldots$ where:*

- *For every i, there is some transition $T_i = (l_i, \phi_i, a_i, \lambda_i, l_{i+1}) \in E$.*
- *For every i, $c_i = C(T_i) + t_i \cdot C(l_i)$ is the cost incurred in the transition.*
- *There is an initial clock valuation $X_0 = 0$.*
- *After every transition, there is a new clock valuation $X_{i+1} = (X_i + t_i)[\lambda_i = 0]$ obtained by increasing every clock variable in X_i by t_i and resetting all clocks in λ_i to 0.*
- *Every clock valuation $X_i + t$ for $t < t_i$ satisfies the invariant $I(l_i)$.*
- *The clock valuation $X_i + t_i$ satisfies ϕ_i for every i.*

Parallel Composition. The parallel composition operator $\|$ allows one to construct a large PTA from several smaller ones. The component PTAs synchronize their transitions via joint signals. For example, a basic attack step can send a 'success' signal to its parent gate, based on which this gate may itself succeed or wait for signals from its other children. Our models use broadcast signals, which can be either output, denoted with an exclamation mark (e.g. 'succ!'), or input, denoted with a question mark (e.g. 'succ?'). We require PTAs to be input-enabled, that is, all input actions of a PTA are enabled at any location, at any time. Thus, if some PTA performs a transition labeled with an output action $a!$, then all receiving PTAs synchronize by taking an $a?$-labeled transition. The formal definition can be found in the Uppaal CORA documentation [29], or in [9].

Queries. We express our security questions in an extension of the *Weighted CTL* logic [12], over a PTA whose locations l are decorated with a set of atomic propositions $Prop(l) \subseteq AP$. We slightly extend the syntax given by [10]: Rather than providing a single constraint $v \sim c$ asking that value v meets bound c, we need a vector of constraints $x_i \sim c_i$, asking that all values v_i meets their bounds c_i.

Definition 5. *The syntax of the WCTL logic is given by the following grammar:*

$$WCTL \ni \phi, \psi ::= p \mid \neg\phi \mid \phi \wedge \psi \mid \exists(\phi \; U_{\sim \boldsymbol{c}} \; \psi) \mid \forall(\phi \; U_{\sim \boldsymbol{c}} \; \psi)$$

Where $p \in AP$, $\boldsymbol{c} \in \mathbb{R}^n_{\geq 0}$, and $\sim \in \{<, \leq, =, \geq, >\}^n$.

The semantics of the boolean operators follows the usual conventions. The existential until operator $\exists(\phi U_{\sim \boldsymbol{c}} \psi)$ is true if there exists a trace of the PTA in which some state s_f satisfies ψ, all states before s_f satisfy ϕ, and the total costs incurred before reaching s_f satisfy the relation $\sim \boldsymbol{c}$. Note that time is considered a cost in this notation. The universal until operator is similar, except that the conditions must hold for every trace. As usual, $\exists \Diamond_{\sim \boldsymbol{c}} \phi$ is shorthand for $\exists(\text{true } U_{\sim c} \phi)$.

Uppaal CORA. Uppaal CORA is an extension of Uppaal with an additional variable *Cost* used for optimal scheduling and cost optimal reachability analysis. With the inbuilt *Best trace option*; it can be used to it find an optimal trace [7]. The rate of change of cost is specified as *Cost*′. Here, the optimal path refers to the trace with the lowest accumulated costs.

4 Analyzing Attack Trees via Price Timed Automata

To analyze an attack tree, we provide a compositional semantics in terms of priced timed automata. That is, we translate each AT element into a PTA and obtain the PTA for the entire AT by putting together all element PTAs via the parallel composition operator $\|$. Then, we analyze ATs by formulating the security measures as queries in the logic mWCTL, which is a slightly extended version of standard weighted computational tree logic that allows us to perform multi-criterion optimization.

In this way, we obtain a versatile and flexible framework for AT analysis. Indeed, if one wants to add a new AT element to the framework, one can simply provide the AT translation, while leaving the rest of the framework unchanged.

4.1 From Attack Trees to Price Timed Automata

Basic Attack Steps. The PTA for a basic attack step v is shown in Figure 2. This PTA models the attacker's choice of whether and when to execute basic attack step, and tracks the time and costs used to do so.

Formally, we convert a BAS S with BAI $(\mathit{Enable}, \mathit{CanSucceed}, \mathit{Eff})$, given attacker profile R, into a PTA $P(S) = \langle L, l_0, X, \mathit{Act}, E, I, C\rangle$ with elements:

- $L = \{I, A, B, F, D\}$
- $l_0 = I$
- $X = \{x\}$
- $\mathit{Act} = \{\mathit{Act}_S?, \mathit{succ}_S!, \mathit{fail}_S!, \tau\}$
- $E = \{\langle I, \top, \mathit{Act}_S?, \emptyset, A\rangle, \langle A, \mathit{Enable}(R), \tau, \{x\}, B\rangle,$
  $\langle B, \top, \mathit{fail}_S!, \emptyset, F\rangle, \langle B, \mathit{CanSucceed}(R), \mathit{succ}_S!, \emptyset, D\rangle\}$
- $I(l) = \top$

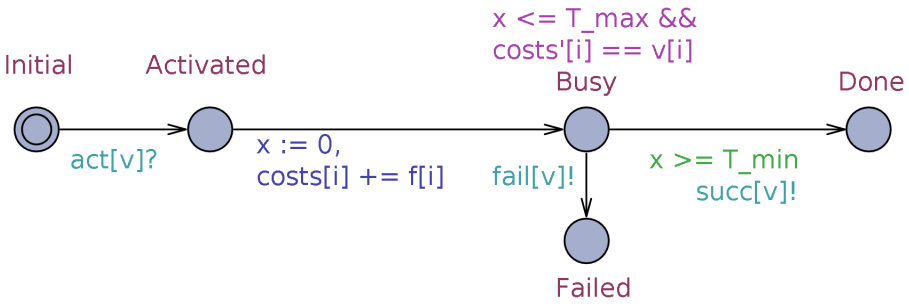


Fig. 2. PTA for a basic attack step. Here v is a unique identifier for the BAS, x is a clock to track the duration of BAS[v], T_min and T_max are the minimum and maximum times, *costs* is an array keeping track of all accumulated costs, and *costs′* is an array for variable costs.

– $C(e) =$
$$\begin{cases} \bigoplus_{i=1}^{n} \mathit{Eff}(a_i, R)(0) & \text{if } e = \langle A, \mathit{Enable}(R), \tau, \{x\}, B\rangle \\ \bigoplus_{i=1}^{n} (\mathit{Eff}(a_i, R)(1) - \mathit{Eff}(a_i, R)(0)) & \text{if } e = B \\ 0 & \text{otherwise} \end{cases}$$

Here we slightly abuse the notation so that $f(R)$ denotes the result of applying f to the valuation obtained from R of all the attributes, and $\bigoplus$ denotes the combination of elements into a vector.

Initially, the BAS waits for an activation signal. As it is received from the parent, the attacker may begin executing the step by incurring the fixed costs. The execution of a BAS is bounded by the minimum and maximum time to complete the attack. While the step is being performed, the variable costs are incurred. The attack may fail at any time, stop incurring further costs and send a failure signal to its parents. Otherwise, it succeeds between the minimum and maximum time constraint for the step and transmits a success signal.

Gates. To model attacker preferences and behavior as illustrated in example 1, we distinguish between sequential and parallel gates. The automata for the parallel AND gate is shown in Figure 3 while the automata for sequential AND gate is shown in Figure 4.

The gates depicted here have only two children. We can construct PTAs having more than two children, however this is cumbersome and requires many more states. Hence, we express AT gates with multiple children by simply chaining two-input gates: For example, an AND-gate with inputs A, B, and C can also be expressed as $A \wedge (B \wedge C)$.

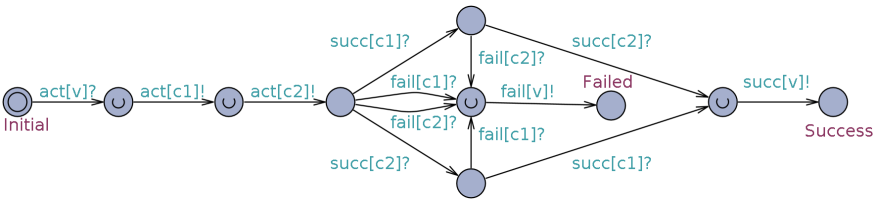


Fig. 3. PTA for parallel AND gate of node v, when child$(v) = c_1c_2$.

Note that the semantics of OR and AND gates are identical except that the behaviours of success and failure are inverted.

The PTAs for these gates begin by waiting for their activation signal, and activating their children. After this, they wait for one of their children to send a signal. For an AND gate, receiving a failure signal always leads the gate to emit its own failure signal, since it is no longer possible for both children to succeed. Conversely, when an OR gate receives one success signal, it always emits its own success. When both children of an AND or OR gate have succeeded or resp. failed, the gate also succeeds or fails.

The sequential gates operate similarly, but they enforce an ordering on their children. First the leftmost child is activated, and the gate waits for a signal from this child. In case of an SAND gate, success of the first child leads to an activation of the second child, and the success of this child cause the success of the gate. Failure of either child leads to failure of the gate, possibly before even activating the second child. The behavior of SOR is similar to sequential AND with success and failure signals swapped.

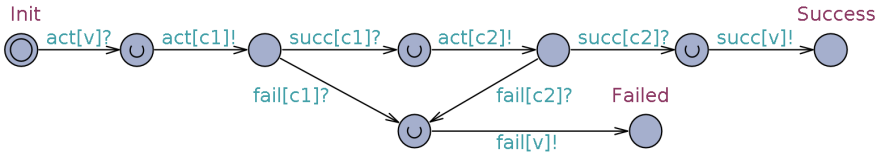


Fig. 4. PTA for sequential AND gate of node v, when child$(v) = c_1c_2$.

Combining the Nodes. For an attack tree A, the PTA associated with A is obtained as the parallel composition of the PTAs for all the nodes, and an additional PTA A_{Top}. If we denote by $P(v, A)$ the PTA corresponding to node v of attack tree A, the total PTA consists of $P_A = P(v_1, A)||P(v_2, A)||\ldots||P(v_n, A)||A_{\mathsf{Top}}$.

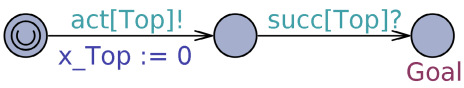


Fig. 5. Automaton for the attack goal Top

The top-level gate Top is associated with a second PTA A_{Top}, shown in Figure 5 that initializes the attack by generating an activation signal for Top. Moreover, it has a clock x_{Top} that tracks the global time, and observes successful completion of an attack via its input *succ*[Top]?. Thus, the location 'Goal' indicates that an attacker has reached the goal.

4.2 Quantitative Analysis of Attack Trees

Given the PTA for an attack tree, we can compute the security metrics as enumerated in Section 2.2 as follows:

(Unconstrained) Attack Values and Attack Paths: The Uppaal CORA program has a built-in method to find an optimum if only one cost needs to be tracked. Here, we obtain the *Optimal accumulated costs* through the 'Best first' function built-in Uppaal CORA.

(Constrained) Attack Values and Attack Paths: Optimal attack values can be obtained by repeatedly querying for the existence of traces reaching the

attack goal with increasingly tight constraints. When the tightest possible bound has been obtained, this corresponds to an optimum. Since a positive result for the query also produces a trace that satisfies it, this procedure also yields an optimal attack path.

For example, to obtain the minimum time to succeed in the AT in Figure 1 given a cost limit of 10000 (assuming this is the only cost variable), we first query $\exists \Diamond_{x_{\mathsf{Top}} \geq 0, C \leq 10000}(P_{\mathsf{Top}}.Goal)$ to obtain some successful attack and its corresponding time, e.g. Suppose this yields an attack that takes 10 days to complete, then we perform a new query $\exists \Diamond_{x_{\mathsf{Top}} < 10, C \leq 10000}(P_{\mathsf{Top}}.Goal)$ to try to find a faster attack. If no such attack exists, we know that the minimal time to complete an attack given the budget is 10 days, and we have obtained an attack path that succeeds in this time and budget.

Ranking: To find different attacks ranked according to their cost, we repeat the procedure above, each time excluding the attack paths we have already found. For example, if the attack consisting of BASs 1 and 3 is the fastest possible attack, the second-fastest is found using the query $\exists \Diamond_{x_{\mathsf{Top}} \leq T}(P_{\mathsf{Top}}.Goal \wedge \neg(P(v_1, A).Success \wedge P(v_3, A).Success))$ and finding the smallest value for T to obtain the second-fastest attack. This process can be repeated until the desired number of optimal attacks has been found.

Pareto Optimal Curves: Pareto optimal curves can be obtained by finding the optimal attacks subject to an increasing constraint. For example, to find the curve of minimal time vs. cost, we begin by finding the minimal time to attack, and computing the lowest-cost attack that meets this time bound. Then, we compute the minimal time to attack with a smaller budget, and again find the lowest-cost attack that meets the new time bound. This process is repeated until no attacks exist that meet the latest budget.

To illustrate, consider again the attack tree in Figure 1. The minimal time to complete an attack is 5 days, and the lowest-cost attack that meets this time limits costs $9250. The fastest attack that costs less than $9250 takes 10 days, and the lowest cost attack that can be performed within 10 days costs $8500. There is no attack that costs less than $8500. Thus we obtain the pareto curve shown in Figure 7.

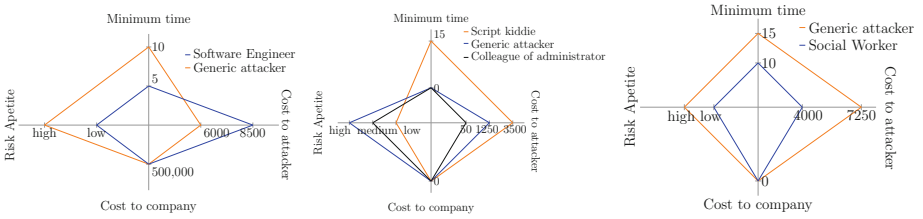
5 Case Studies

We demonstrate our approach through three well-known case studies taken from the literature. For each case, we analyze optimal attack values such as time and cost for the attacker, and the minimal damage borne by the company by taking different attacker profiles. Here, we consider the attacker's resources (time and budget), skills, motivation, access to infrastructure, risk appetite, and preferences as attributes of a rational attacker.

As is often the case in security analysis, it is difficult to obtain precise data, and our guesses are not intended to reflect reality, but rather to illustrate the analysis method.

Table 2. Analysis results for the cracking password protected file.

Profile	Criterion	Attack value	Attack Path		
			BAS	Time	Cost
Generic attacker	Minimum cost	7250	Dictionary	0 - 15	7250
	2nd best min. cost	7250	Brute force	0 - 15	7250
	Minimum time	15	Bruteforce	0 - 15	7250
	Min. cost to company	0	Guessing	0 - 15	7250
Social Worker	Minimum cost	4000	Generic reconnaissance	0 - 0	50
			Phone trap Execution	0 - 15	3500
	2nd best min. cost	4500	Generic reconnaissance (*fails*)	0 - 0	500
			Physical reconnaissance	0 - 0	500
			Key logger local installation	0 - 5	1750
			Password intercept	5 - 10	1750
	Minimum time	10	Generic reconnaissance (*fails*)	0 - 0	500
			Physical reconnaissance	0 - 0	500
			Key logger local installation	0 - 5	1750
			Password intercept	5 - 10	1750
	Min. cost to company	0	Dictionary	0 - 15	7250



(a) Forestalling release of software

(b) Obtaining administrator privileges

(c) Password protected file

Fig. 6. Analysis of attacker attributes in all three case studies.

The attack steps are decorated with the time required to successfully execute the step, as well as fixed and variable costs incurred by the attacker. These values are specified for the different attacked roles and skills levels. A cost for the attacked company is also included, but this is independent of the attacker profile.

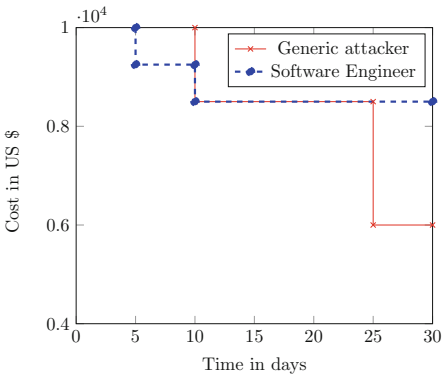


Fig. 7. Pareto curve of attack tree in Figure 1

Our analysis can provides insightful information about vulnerable paths and values relevant to risk managers. An input table is provided as Table 1 for Case study 1 to illustrate our methodology and we use similar scale to perform other case studies provided in the paper. The exact values for other case studies will be provided in a detailed report.

Forestalling release of software. As elaborated in Example 1, the AT in Figure 1 models the forestalling of software from [18]. We consider two attacker profiles:

A generic attacker and a software engineer. The generic attacker is profit-motivated and has a high risk appetite, but is not particularly skilled in this type of attack. The software engineer has better access, skills, equipment, but low risk appetite. The role of the attacker and the skill level are explicitly included in the attacker profile, while the other attributes are reflected in the values of time and cost to perform the step.

Formally, the profile of the generic attacker is defined by $R_{GA}(Role) =$ '*Generic Attacker*' and $R_{GA}(Skill) =$ '*Low*'. Similarly, the profile for the software engineer is $R_{SE}(Role) =$ '*Software Engineer*' and $R_{SE}(Skill) =$ '*High*'.

Table 1 shows the input parameters. The analysis results are presented as a Pareto curve in Figure 7, where the generic attacker requires 10 days incurring a minimum cost of $9250 while a software engineer incurs a cost of $8500, but can complete the attack in 5 days.

Here, we see that both attack values and the choice of attack path heavily depend on the attacker profile. In contrast to the generic attacker whose cost optimal attack trace is to bribe a programmer, a better skilled software engineer exploits a bug in the computer system to steal the code. The minimum time required to accomplish the attack also heavily depends on which attack steps are executed and when. While a generic attacker takes 10 days to successfully execute the attack by physical robbery, a software engineer with insider benefits takes only 5 days to accomplish his goal. Also, there is an attack trace i.e *Hire a robber, Robber breaks into system, Code is completed into product* which results in an *optimum Cost to company* as $500,000 irrespective of the considered attacker profiles.

Cracking a password protected file. The attack tree depicted in Figure 8 models an attack on a password protected file. It is taken from [25] and modified to add SAND and SOR gates. The goal of the attack is to obtain a password. This, can be done by either performing a *brute force attack* or taking a multistep approach

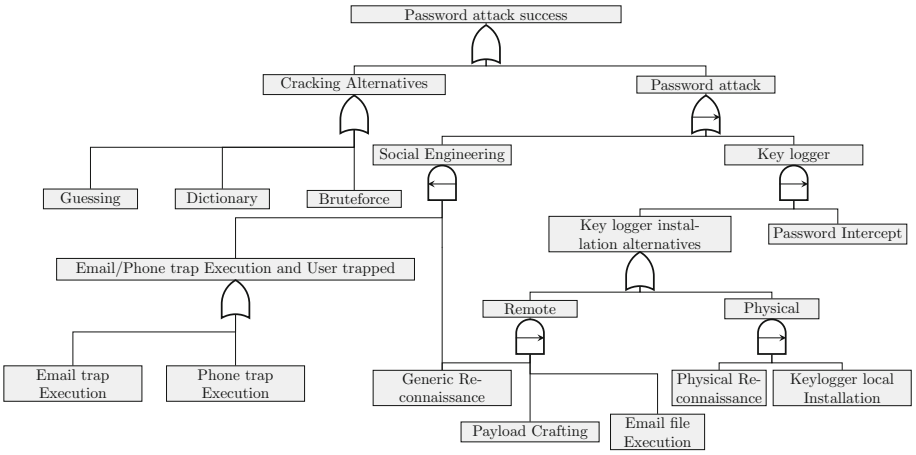


Fig. 8. Dynamic Attack Tree modelling the attack on password protected file.

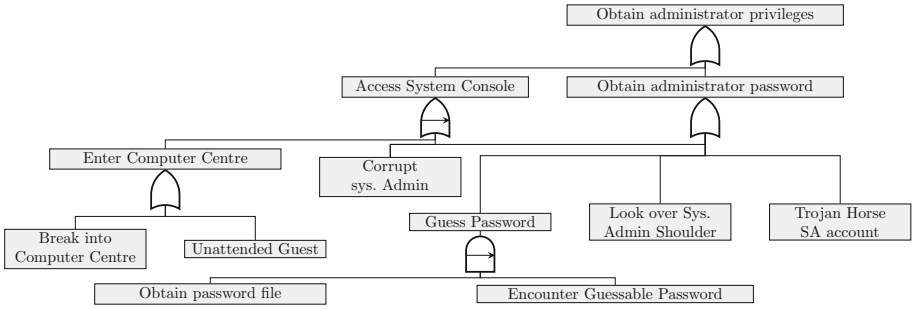


Fig. 9. To obtain administrator privileges.

of trying a *password attack*. To model different attacker behavior, we take two attackers profiles into account.

A generic attacker who is a profit-motivated, skilled professional willing to bear penalties; and a social worker, a popular public figure who is also profit-motivated, but has a low risk appetite. Table 2 tabulates the optimum attack values and traces which illustrate that adversarial behavior greatly depends on his possessed attributes and his perceived goal. While a social worker; being good in social engineering can crack the password in minimum 10 days through *Physical reconnaissance*, he prefers *Generic reconnaissance* for achieving his goal, incurring the minimum cost of $4000 US. In contrast, a generic attacker prefers more technical approaches like *Bruteforce* in achieving his goal in a minimum time of 15 days and *Dictionary attack* by incurring the minimum cost of $7250.

Obtaining administrator privileges. The goal of the attack tree in Figure 9 is to obtain administrator privileges and has been adopted from [19]. We consider three different attacker profiles for our analysis.

A generic attacker who is a professional hacker, with high risk appetite and malicious intentions to disrupt the availability of the system; a script kiddie fearful by conscience trying to hack just for fun and who has low risk taking ability; and an insider: a colleague of a system administrator with better access to the computer center. The insider, knowing system details expects a huge profit from the attack and is this willing to bear risk to a greater extent. The results in Figure 6(b) show that the colleague of the system administrator, knowing the vulnerabilities of the system, can reach the goal with minimal investments. Having less resources, a juvenile attacker's optimal cost and time are both higher than professional generic attacker and the malicious insider. Note that the fastest attack may not be the cheapest one due to several attack steps being performed concurrently under different constraints of time and base costs.

Figure 6 provides a succinct representation of these different attack scenarios. We put the attacker objectives as vertices (i.e Minimum cost, Minimal time, Risk appetite, Cost to company) and the connecting lines are the attacker profiles (discussed is the case descriptions). The figure shows a trade-off among different attack values for the considered attacker profiles which an enterprise risk manager can use to effectively plan countermeasures.

6 Conclusion

We have presented a framework of security risk analysis by reducing a multi-parameter attack tree into priced timed automata. By slightly deviating from the strict formalism of attack trees by allowing shared subtrees, we preserve the intuitive representation of attack scenarios while also providing insightful qualitative and quantitative information in terms of optimal attack paths and values. Furthermore, our analysis takes temporal dependencies into account by defining the semantics of SAND and SOR gates.

As future work, we plan to analyze case studies incorporating realistic values, and to extend our framework by including success probabilities of basic attack steps. We see clear parallels between our approach and stochastic games and in the future we would like to integrate the best of both.

Acknowledgement. This work has been supported by the EU FP7 project TREsPASS (318003) and by the STW-ProRail partnership program ExploRail under the project ArRangeer (12238).

References

1. Aagedal, J., Braber, F., Dimitrakos, T., Gran, B.A., Raptis, D., Stølen, K.: Model-based risk assessment to improve enterprise security. In: Proc. 6th Int. Enterprise Distributed Object Computing Conf. (EDOC 2002), p. 51 (2002)
2. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science **126**(2), 183–235 (1994)
3. Amoroso, E.: Fundamentals of computer security technology. Prentice-Hall Inc., Upper Saddle River (1994)
4. Arnold, F., Hermanns, H., Pulungan, R., Stoelinga, M.: Time-dependent analysis of attacks. In: Abadi, M., Kremer, S. (eds.) POST 2014 (ETAPS 2014). LNCS, vol. 8414, pp. 285–305. Springer, Heidelberg (2014)
5. Risk Management. Australian/New Zealand Standard, AS/NZS 4360:2004 14443 (2004)
6. Technical standard to Risk Taxonomy, The Open Group, C081 (2009)
7. Behrmann, G., Larsen, K.G., Rasmussen, J.I.: Optimal scheduling using priced timed automata. SIGMETRICS Performance Evaluation Review **32**(4) (2005)
8. Behrmann, G., Larsen, K.G., Rasmussen, J.I.: Priced timed automata: algorithms and applications. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2004. LNCS, vol. 3657, pp. 162–182. Springer, Heidelberg (2005)
9. Bengtsson, J.E., Yi, W.: Timed automata: semantics, algorithms and tools. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) Lectures on Concurrency and Petri Nets. LNCS, vol. 3098, pp. 87–124. Springer, Heidelberg (2004)
10. Bouyer, P.: Weighted timed automata: Model-checking and games. Electronic Notes in Theoretical Computer Science **158**, 3–17 (2006)
11. Bowles, J.B., Hanczaryk, W.: Threat effects analysis: Applying FMEA to model computer system threats. In: 2008 Annual Reliability and Maintainability Symp., pp. 463–468. IEEE, January 2008
12. Brihaye, T., Bruyère, V., Raskin, J.-F.: Model-checking for weighted timed automata. In: Lakhnech, Y., Yovine, S. (eds.) FORMATS 2004 and FTRTFT 2004. LNCS, vol. 3253, pp. 277–292. Springer, Heidelberg (2004)

13. Buckshaw, D.L.: Use of Decision Support Techniques for Information System Risk Management. John Wiley Sons, Ltd. (2014)
14. Buldas, A., Laud, P., Priisalu, J., Saarepera, M., Willemson, J.: Rational choice of security measures via multi-parameter attack trees. In: López, J. (ed.) CRITIS 2006. LNCS, vol. 4347, pp. 235–248. Springer, Heidelberg (2006)
15. Dacier, M., Deswarte, Y.: Privilege graph: an extension to the typed access matrix model. In: Proc. Third European Symp. on Research in Computer Security (ESORICS), Brighton, UK, November 7–9. pp. 319–334 (1994)
16. Ford, M.D., Keefe, K., LeMay, E., Sanders, W.H., Muehrcke, C.: Implementing the ADVISE security modeling formalism in Möbius. In: Proc. 43rd Int. Conf. on Dependable Systems and Networks (DSN), pp. 1–8 (2013)
17. Hansson, J., Wrage, L., Feiler, P.H., Morley, J., Lewis, B.A., Hugues, J.: Architectural modeling to verify security and nonfunctional behavior. IEEE Security & Privacy **8**(1), 43–49 (2010)
18. Jürgenson, A., Willemson, J.: Processing multi-parameter attacktrees with estimated parameter values. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 308–319. Springer, Heidelberg (2007)
19. Jürgenson, A., Willemson, J.: Computing exact outcomes of multi-parameter attack trees. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part II. LNCS, vol. 5332, pp. 1036–1051. Springer, Heidelberg (2008)
20. Kordy, B., Pièrre-Cambacédès, L., Schweitzer, P.: DAG-based attack and defense modeling: Don't miss the forest for the attack trees. Computer Science Review **13–14**, 1–38 (2014)
21. LeMay, E., Ford, M.D., Keefe, K., Sanders, W.H.: Model-based security metrics using adversary view security evaluation (ADVISE). In: 2011 Eigth Int. Conf. on Quantitative Eval. of Systems (QEST). IEEE (2011)
22. Lenin, A., Willemson, J., Sari, D.P.: Attacker profiling in quantitative security assessment based on attack trees. In: Bernsmed, K., Fischer-Hübner, S. (eds.) NordSec 2014. LNCS, vol. 8788, pp. 199–212. Springer, Heidelberg (2014)
23. Mauw, S., Oostdijk, M.: Foundations of attack trees. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 186–198. Springer, Heidelberg (2006)
24. McQueen, M., Boyer, W., Flynn, M., Beitel, G.: Quantitative cyber risk reduction estimation methodology for a small scada control system. In: Proc. 39th Annual Hawaii Int. Conf. on System Sciences (HICSS), vol. 9, p. 226, January 2006
25. Pièrre-Cambacédès, L., Bouissou, M.: Beyond attack trees: Dynamic security modeling with boolean logic driven markov processes (BDMP). In: Dependable Computing Conf. (EDCC), pp. 199–208 (2010)
26. Ray, I., Poolsapassit, N.: Using attack trees to identify malicious attacks from authorized insiders. In: di Vimercati, S.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 231–246. Springer, Heidelberg (2005)
27. Schneier, B.: Attack trees: modeling security threats. In: Dr. Dobb's journal, December 1999
28. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.: Automated generation and analysis of attack graphs. In: Security and Privacy, Proc. 2002 IEEE Symp., pp. 273–284 (2002)
29. Uppaal CORA. http://people.cs.aau.dk/adavid/cora/index.html
30. Weiss, J.: A system security engineering process. In: Proc. 14th National Computer Security Conference, vol. 249, October 1991