Quantitative Evaluation of Attack Defense Trees using Stochastic Timed Automata

Peter Gjøl Jensen, Kim Larsen, Axel Legay, Danny Poulsen

To cite this version:

Peter Gjøl Jensen, Kim Larsen, Axel Legay, Danny Poulsen. Quantitative Evaluation of Attack Defense Trees using Stochastic Timed Automata. GraMSec 2017 - The Fourth International Workshop on Graphical Models for Security, Aug 2017, Santa Barbara United States. hal-01640091

HAL Id: hal-01640091

https://hal.inria.fr/hal-01640091

Submitted on 20 Nov 2017

Quantitative Evaluation of Attack Defense Trees using Stochastic Timed Automata

Peter Gjøl Jensen[1], Kim Guldstrand Larsen[1], Axel Legay[2], and Danny Bøgsted Poulsen[3]

[1] Department of Computer Science, Aalborg University
[2] INRIA - Rennes
[3] Christian Albrechts Univertät, Kiel

Abstract. Security analysis is without doubt one of the most important issues in a society relying heavily on computer infrastructure. Unfortunately security analysis is also very difficult due to the complexity of systems. This is bad enough when dealing with ones own computer systems - but nowadays organisations rely on third-party services - *cloud services* - along with their own antiquated legacy systems. Combined this makes it overwhelming difficult to obtain an overview of possible attack scenarios. Luckily, some formalisms such as attack trees exists that can help security analysts. However, temporal behaviour of the attacker is rarely considered by these formalisms.
In this paper we build upon previous work on attack-defence trees to build a proper temporal semantics. We consider the attack-defence tree a reachability objective for an attacker and thereby separates the attacker logic from the attack-defence tree. We give a temporal stochastic semantics for arbitrary attackers (adhering to certain requirements to make the attacker "sane") and we allow annotating attacker actions with time-dependent costs. Furthermore, we define what we call a cost-preserving attacker profile and we define a parameterised attacker profile. The defined semantics is implemented via a translation to UPPAAL SMC. Using UPPAAL SMC we answers various questions such as the expected cost of an attack, we find the probability of a successful attack and we even show how an attacker can find a optimal parameter setting using ANOVA and Tukeys test.

1 Introduction

Society is in increasing fashion relying on computer systems to support it in every-day life and users are as such depending on computer systems to store their personal data to automate tasks: companies may store information about customers e.g. addresses for billing purposes and credit card information for automatic payment. The more information stored in computer systems, the higher the need for protecting these data. History has however shown that making systems inpenetrable is extremely difficult and we have witnessed several security breaches. The IT-infrastructure of today is complex and gaining a high-level overview of how attacks may occur is extremely difficult for all but the simplest infrastructures - a problem worsened by the fact that organisations nowadays heavily rely on *Cloud Services.* A tool that has been developed for battling this complexity is *Attack-trees* [11, 13]. Attack-trees is a tree-based formalism in which an overall attack on an organisation may be refined into sub-attacks - or different ways of achieving an attack. This formalism provides a security specialist a quick way of describing the possible ways that he sees an attack can happen. Attack trees was originally introduced by Schneier [15] and since given a formal semantics by Mauw and Oostdijk [13] - an overview of attack-trees is given by Kordy et al. [12]. Basic quantitative analysis is possible on the attack-tree formalism (with some annotations regarding costs and probabilities) in a bottom-up fashion (i.e. costs for leafs nodes are propagated up the attack-tree to calculate the cost of individual sub-attacks) [4]. In addition also various extensions have been considered in order to push the analysis of attacks. Given an overview of possible attacks; it is intuitively straightforward to connect the attack nodes with possible defence measures, allow defence measures to be split into sub-tasks, and allow defences to be countered by other attack options which gives us the *Attack-Defence-trees* (AD-tree) formalism [2, 11]. These trees allows analysis of the interplay between the attacker and defender and by doing so gives better understanding of possible attack-defence scenarios [4, 11]. Attack-defence trees describes a game between two players but most analyses are still based on a bottom-up-propagations [2] approach. Two recent works by Hermanns et al. [10] and Gadyatskaya et al. [9] develops a temporal and stochastic semantics for attack-defence scenarios. Hermanns et al. [10] develops their own formalism *Attack-Defence-Diagrams* (ADD) highly inspired by attack-defence trees. A problem with ADDs, in our view, is that while giving a stochastic semantics they also compromise the simplicity of attack-defence trees by requiring the user to consider things like *reset*-gates. The work by Gadyatskaya et al. [9] on the other hand take their outset on the attack-defence trees and build the semantics directly on top of these - thus a security specialist may do their modelling exactly as usual and let a tool translate this to a stochastic semantics. Both Hermanns et al. [10] and Gadyatskaya et al. [9] translated their respective formalism to a timed automata [1] with stochastic interpretations. Another recent work concerned with temporal ordering of events is that of Aslanyan et al. [3] which adds a sequential sequence operator to the syntax of attack-defence trees. This logically split the tree into sub-attack-defence-trees which are processed in sequence. They then use this split into sub-trees to develop a game semantics playing out as follows: first the defender select defence measures for the first tree, then the attacker selects attacks and a result for the first tree is found (using a probabilistic outcome of the attackers selected attacks). The result of the first tree is then propagated onwards to the second subtree- and so it continues till the last sub-tree. Unlike Hermanns et al. [10] and Gadyatskaya et al. [9] the attacker is not allowed to retry failed attacks.

In this paper we expand upon the work of Gadyatskaya et al. [9] by firstly allowing the cost of an attack to be a function of time instead of a constant cost per atomic attack attempt. Secondly, we introduce the concept of parameterised attackers and show how we can find attacker-parameters that minimise cost by applying Analysis Of VAriance (ANOVA). The entire modelling framework is translated into a UPPAAL SMC model automatically by a python script. All the user needs to provide is a textual description of the AD-tree, the description of the cost per atomic attack and the duration interval for each atomic attack.

In Section 2 we introduce the attack-defence tree formalism and define our stochastic semantics with costs. We also present attacker and defender questions and describes how these may be solved. In Section 3 we instantiate our framework with a parameterised attacker. In Section 4 we discuss a translation of Section 3 into UPPAAL SMC timed automata and show how to use UPPAAL SMC to answer the questions of Section 2.

2 Attack-Defence Trees

In this section we follow the course of first presenting the traditional "static" attack-defence-trees, followed by an extension to a temporal semantics following the work of Gadyatskaya et al. [9]. Further, this temporal semantics is given a stochastic semantics by defining success probabilities to attacker actions and by associating a time distribution to atomic attacks. As a final part of this section, and a main contribution of the paper, is parameterising the attacker.

At the simplest a static attack-defence tree (AD-tree) is a propositional formula with propositions split into attacker propositions ($\mathtt{A}_a$) and defender propositions ($\mathtt{A}_d$).

Definition 1 (AD-tree). *An AD-tree over the attacker actions $\mathtt{A}_a$ and defender actions $\mathtt{A}_d$ is generated by the syntax $t ::= p \mid t \wedge t \mid t \vee t \mid \sim t$ where $p \in \mathtt{A}_a \cup \mathtt{A}_d$. We denote by $\mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$ all AD-trees over $\mathtt{A}_a$ and $\mathtt{A}_d$.*

For an AD-tree $\psi \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$ and selected attacker action $A \subseteq \mathtt{A}_a$ and selected defence measures $D \subseteq \mathtt{A}_d$, we inductively define $[\![\psi]\!]A, D$ as follows

- $[\![p]\!]A, D = \mathtt{tt}$ if $p \in A \cup D$, $\mathtt{ff}$ otherwise
- $[\![\psi_1 \wedge \psi_2]\!]A, D = ([\![\psi_1]\!]A, D) \wedge ([\![\psi_2]\!]A, D)$
- $[\![\psi_1 \vee \psi_2]\!]A, D = ([\![\psi_1]\!]A, D) \vee ([\![\psi_2]\!]A, D)$
- $[\![\sim \psi]\!]A, D = \neg([\![\psi]\!]A, D)$

We depict an AD-tree by its parse tree as shown in Figure 1. The particular AD-tree in Figure 1 models how an attacker may substitute an RFID-tag in a shop [2, 9]. One way involves threatening the employees; another involves bribing a subject - which also requires identifying the subject before-hand.

The syntax given in Definition 1 is overly liberal in what AD-trees it allows. In particular, it allows defining trees in which an actor may do an action that helps the opponent - for instance $\mathtt{StealCreditCard} \wedge \neg \mathtt{LooseCreditCard}$ would express that the attacker achieve an attack if he steals a credit cards and does not loose it again. We consider that the attacker only does things actively and thus he would have chosen to loose the credit card i.e. he decides to do something

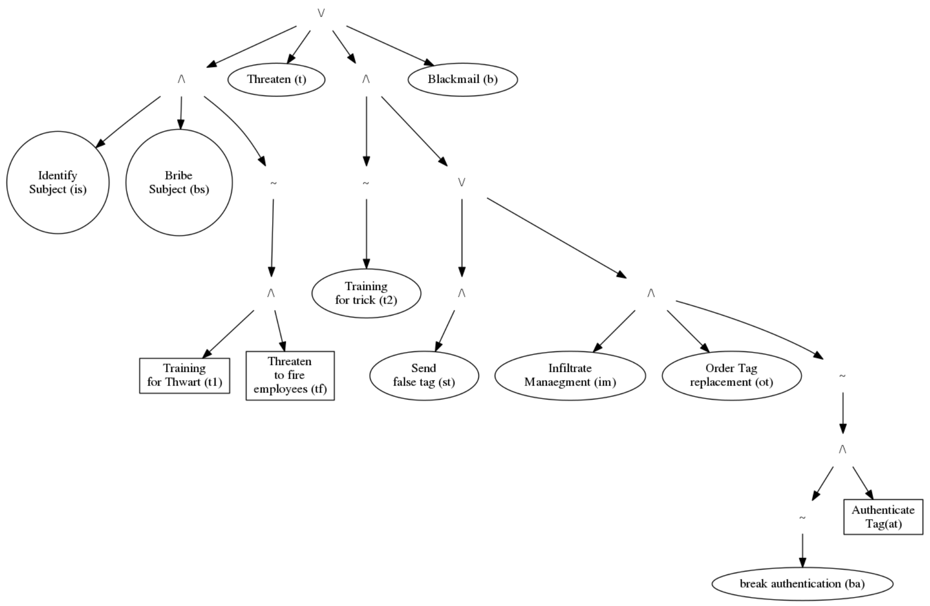


Fig. 1: An Attack-Defence Tree. Circles denote attacker actions, squares defender actions.

harmful to his goal. To avoid this problem we follow along [2] and impose a type-system that disallows such trees. The type system has two types a and d, denoting attacker and defender respectively. The type system is given in Figure 2.

For the remainder of this paper we only consider well-formed AD-trees with respect to the type-system in Figure 2. Since we are interested in analysing possible attacks, we restrict our attention to AD-trees, ψ, for which $\mathtt{A}_d, \mathtt{A}_a \vdash \psi : a$. From now on let $\mathcal{L}^w(\mathtt{A}_a, \mathtt{A}_d) = \{\psi \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d) \mid \mathtt{A}_a, \mathtt{A}_d \vdash \psi : a\}$.

A key question for AD-trees is whether an attacker for any set of defence measures can select a set of atomic attacks such that an attack will be successful. In a similar fashion, the defender is interested in finding defence measures that guarantees an attack can never occur.

$$\frac{}{\mathtt{A}_d, \mathtt{A}_a \vdash p : a}, p \in \mathtt{A}_a \qquad \frac{}{\mathtt{A}_d, \mathtt{A}_a \vdash p : d}, p \in \mathtt{A}_d \qquad \frac{\mathtt{A}_d, \mathtt{A}_a \vdash \psi_1 : r \qquad \mathtt{A}_d, \mathtt{A}_a \vdash \psi_2 : r}{\mathtt{A}_d, \mathtt{A}_a \vdash \psi_1 \wedge \psi_2 : r}$$

$$\frac{\mathtt{A}_d, \mathtt{A}_a \vdash \psi_1 : r \qquad \mathtt{A}_d, \mathtt{A}_a \vdash \psi_2 : r}{\mathtt{A}_d, \mathtt{A}_a \vdash \psi_1 \vee \psi_2 : r} \qquad \frac{\mathtt{A}_d, \mathtt{A}_a \vdash \psi_1 : r}{\mathtt{A}_d, \mathtt{A}_a \vdash \sim \psi_1 : r^{-1}}, r^{-1} = \begin{cases} a & \text{if } r = d \\ d & \text{if } r = a \end{cases}$$

Fig. 2: Type system to make attack-defence trees well-formed

Attacker Question 1 *Given an AD tree $\psi \in \mathcal{L}^w(\mathtt{A}_a, \mathtt{A}_d)$ and a $D \subseteq \mathtt{A}_d$ does there exist an $A \subseteq \mathtt{A}_a$ such that $[\![\psi]\!]A, D = \mathtt{tt}$.*

Defender Question 1 *Given an AD tree $\psi \in \mathcal{L}^w(\mathtt{A}_a, \mathtt{A}_d)$ does there exist a $D \subseteq \mathtt{A}_d$ such that for any $A \subseteq \mathtt{A}_a$, $[\![\psi]\!]A, D = \mathtt{ff}$.*

2.1 Temporal Semantics

The temporal semantics we consider consists of four components: the tree, the attacker, the defender and an environmental model. The tree component defines a finite graph with vertices encoding the current atomic attacks that are true and the currently selected defense measures. The attacker is then merely a mapping from the tree-vertices to actions he may choose to perform. The actual outcome of performing an action is simultaneously ($\mathtt{tt}$ or $\mathtt{ff}$) selected by the environment component.

Definition 2 (Tree-Graph). *Let $\psi \in \mathcal{L}^w(\mathtt{A}_a, \mathtt{A}_d)$ be an AD-tree, then the tree-graph over ψ, denoted $\mathtt{Graph}(\psi)$, is a tuple $(\mathcal{V}^t, \mathcal{V}^t_\dagger, {v^t}^0, \mathtt{E}, \mathtt{E}_d)$ where 1) $\mathcal{V}^t = 2^{\mathtt{A}_a} \times 2^{\mathtt{A}_d}$ is a set of vertices, 2) $\mathcal{V}^t_\dagger \subseteq \{(A,D) \in \mathcal{V}^t \mid [\![A, D]\!]\psi = \mathtt{tt}\}$ is a set of final states 3) ${v^t}^0 = (\emptyset, \emptyset)$ is the initial vertex, 4) $\mathtt{E} \subseteq \mathcal{V}^t \times \mathtt{A}_a \times \mathcal{V}^t$ is a set of edges where $((A,D), a, (A \cup \{a\}, D)) \in \mathtt{E}$ if $a \notin A$ and $[\![\psi]\!]A, D = \mathtt{ff}$, 5) $\mathtt{E}_d = \{(v^0, D, S) \mid D \subseteq \mathtt{A}_d \wedge S = (\emptyset, D)\}$ is the defence select edges.*

An attacker looks into the state of the tree-graph and based on this information selects a set of actions that are feasible for him to perform. Likewise, a defender maps the initial vertex of the graph to possible subsets of defenses to perform. In the following, assume there is a special symbol $\dagger \notin \mathtt{A}_a$ used by an attacker to signal he will not do any action

Definition 3. *Let $\psi \in \mathcal{L}^w(\mathtt{A}_a, \mathtt{A}_d)$ be an AD-tree and let $\mathtt{Graph}(\psi) = (\mathcal{V}^t, \mathcal{V}^t_\dagger, {v^t}^0, \mathtt{E}, \mathtt{E}_d)$ be its associated tree graph. An attacker is a function $\mathtt{Att} : \mathcal{V}^t \to 2^{\mathtt{A}_a} \cup \{\{\dagger\}\}$ with the restriction that if $a \in \mathtt{Att}((A,D))$ then $a \notin A$ and either $(A,D) \notin \mathcal{V}^t_\dagger$ or $a = \dagger$. We call $\mathtt{Att}$ deterministic if for all $v^t \in \mathcal{V}^t$, $|\mathtt{Att}(v)| = 1$ - otherwise it is non-deterministic.*

The requirement on the attacker function is expressing that the attacker can only choose to do an action if that action has not previously succeeded and the tree is not already true.

Definition 4. *Let $\psi \in \mathcal{L}^w(\mathtt{A}_a, \mathtt{A}_d)^w$ be an AD-tree and let $\mathtt{Graph}(\psi) = (\mathcal{V}^t, \mathcal{V}^t_\dagger, {v^t}^0, \mathtt{E}, \mathtt{E}_d)$ be its associated tree graph. A defender is a function $\mathtt{Def} : \{{v^t}^0\} \to 2^{2^{\mathtt{A}_d}}$. We call $\mathtt{Def}$ determinstic if $|\mathtt{Def}({v^t}^0)| = 1$- otherwise it is non-deterministic.*

The temporal semantics we consider is based on the idea that a defender selects his defence measures once, followed by the attacker selecting atomic attacks in a step-wise fashion.

Remark 1. The choice of letting the defender select defence measures only once is contrasting the work of Hermanns et al. [10] that considers a semantics where defender and attacker can do actions interchangeably. We believe it is a more realistic scenario that a defender choose actions to prevent attacks and not actively work against an attacker.

Definition 5. *Let $\psi \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)^w$ be an AD-tree, $\mathtt{Graph}(\psi) = (\mathcal{V}^t, \mathcal{V}^t_\dagger, {v^t}^0, \mathtt{E}, \mathtt{E}_d)$ be its associated tree-graph, $\mathtt{Def}$ be a defender and let $\mathtt{Att}$ be an attacker then the transition system over ψ with attacker $\mathtt{Att}$ and defender $\mathtt{Def}$, denoted $\mathtt{LTS}(\psi|\mathtt{Att}|\mathtt{Def})$, is a tuple $(\mathcal{V}, \mathcal{V}_\dagger, v^0, \rightarrow, \rightarrow_\neg, \rightarrow_\dagger, \dashrightarrow)$ where 1) $\mathcal{V} = \mathcal{V}^t$ is a set of states, 2) $\mathcal{V}_\dagger \subseteq \mathcal{V}^t_\dagger \cup \{v \in \mathcal{V} \mid \mathtt{Att}(\mathcal{V}) = \dagger\}$ is a set of dead-end states, 3) $v^0 = {v^t}^0$ is the initial state, 4) $\rightarrow \subseteq \mathcal{V} \times \mathtt{A}_a \times \mathcal{V}$ is a set of transitions where $(v, a, v') \in \rightarrow$ if $a \in \mathtt{Att}(v)$ and $(v, a, v') \in \mathtt{E}$, 5) $\rightarrow_\neg \subseteq \mathcal{V} \times \mathtt{A}_a \times \mathcal{V}$ is a set of transitions where $(v, a, v) \in \rightarrow_\neg$ only if $(v, a, v') \in \rightarrow$, 6) $\rightarrow_\dagger = \{(v, \dagger, v) | v \in \mathcal{V} \wedge \dagger \in \mathtt{Att}(v)\}$ is the do-no-attack transition relation, and 7) $\dashrightarrow \subseteq \{(v^0, D, v) \mid D \in \mathtt{Def}(v^0) \wedge (v^0, D, v) \in \mathtt{E}_d\}$ is the defence select transitions.*

As per tradition we write $v \xrightarrow{a} v'$ in lieu of $(v, a, v') \in \rightarrow$, $v \xrightarrow{\neg a} v'$ instead of $(v, a, v') \in \rightarrow_\neg$ and similarly for $\dashrightarrow$ and $\rightarrow_\dagger$. We write $v \rightarrow^* v'$ if there is a sequence of transitions emanating for v and ending in v' and denote by $\mathtt{Reach}(v) = \{v' | v \rightarrow^* v'\}$.

A run over $\mathtt{LTS}(\psi|\mathtt{Att}|\mathtt{Def})$ is a sequence $(v^0, D)(v_1, \alpha_1)(v_2, \alpha_2) \dots (v_{n-1}, \alpha_{n-1})(v_n, \dagger)v_n$ where $v^0 \overset{D}{\dashrightarrow} v_1$ and for all $1 \leq i < n$, $\alpha_i \in \{a, \neg a \mid a \in \mathtt{A}_a\}$, $v_i \xrightarrow{\alpha_i} v_{i+1}$ and finally $v_n \xrightarrow{\dagger} v_n$. We denote by $\Omega(\psi|\mathtt{Att}|\mathtt{Def})$ all runs over ψ.

Obviously, as an attacker, we would be interested in for any D to find a run $(v^0, D)(v_1, \alpha_1)(v_2, \alpha_2) \dots (v_n, \dagger)v_n$ where $v_n \in \mathcal{V}_\dagger$ and $[\![\psi]\!]v_n = \mathtt{tt}$ whereas a defender wishes to find a D' such that all runs ends in a state $v'_\dagger$ where $[\![\psi]\!]v'_\dagger = \mathtt{ff}$. In reality this is just expressing Attacker Question 1 and Defender Question 1 within our temporal semantics.

Attacker Question 2 *Given an AD tree $\psi \in \mathcal{L}^w(\mathtt{A}_a, \mathtt{A}_d)$, an attacker $\mathtt{Att}$, non-deterministic defender $\mathtt{Def}$ is it the case for all D that there exists a run $(v^0, D)(v_1, \alpha_1)(v_2, \alpha_2) \dots (v_n, \dagger)v_n \in \Omega(\psi|\mathtt{Att}|\mathtt{Def})$ such that $[\![\psi]\!]v_n = \mathtt{tt}$.*

Defender Question 2 *Given an AD tree $\psi \in \mathcal{L}^w(\mathtt{A}_a, \mathtt{A}_d)$, attacker $\mathtt{Att}$, non-deterministic defender $\mathtt{Def}$ does there exists a $D \subseteq \mathtt{A}_d$ such that for any run $(v^0, D)(v_1, \alpha_1)(v_2, \alpha_2) \dots (v_n, \dagger)v_n \in \Omega(\psi|\mathtt{Att}|\mathtt{Def})$, $[\![\psi]\!]v_n = \mathtt{ff}$.*

Technique 1 *The verification technique that may be used to answer Attacker Question 2 and Defender Question 2 is model checking. Consider we are given an AD-tree $\psi \subseteq \mathcal{L}^w(\mathtt{A}_a, \mathtt{A}_d)$, an attacker $\mathtt{Att}$ and non-deterministic defender $\mathtt{Def}$ and wants to answer Attacker Question 2. Let $\mathtt{LTS}(\psi|\mathtt{Att}|\mathtt{Def}) = (\mathcal{V}, \mathcal{V}_\dagger, v^0, \rightarrow, \rightarrow_\neg, \rightarrow_\dagger, \dashrightarrow)$ and let $\mathcal{V}_D = \{v \in \mathcal{V} \mid v^0 \overset{D}{\dashrightarrow} v\}$ be the set of states reached by the defender doing an action. Then the straightforward approach for answering Attacker Question 2 is to do a reachability search from all of the states in $\mathcal{V}_D$ for a state $v \in \mathcal{V}_\dagger$ where $[\![v]\!]\psi = \mathtt{tt}$. That is, if for all vertices $v \in \mathcal{V}_D$ the set*

$R_v = \{v' \in \texttt{Reach}(v) \mid [\![v']\!]\psi = \texttt{tt}\}$ is non-empty then Attacker Question 2 is true. On the other hand, if for some v, R_v is empty we have found an answer for Defender Question 2.

The possibility of an attack is intereseting in its own right; but disregards how long time an attack may take. In the real world, an attacker may only be interested in attacks that are executable within a given time horizon. Likewise, the defender could possibly be happy as long he can guarantee a successful attack can only occur after a specific time. To incorporate timing information, consider that each atomic attack is assigned a duration interval by a function $\Delta : \texttt{A}_a \to \mathcal{B}(\mathbb{R})$ - where $\mathcal{B}(\mathbb{R})$ denotes all possible intervals over $\mathbb{R}$. A timed attacker is thus a tuple $\texttt{Att}^\tau = (\texttt{Att}, \Delta)$ where $\texttt{Att}$ is an attacker and Δ is defined as above.

With a timed attacker $(\texttt{Att}, \Delta)$ and defender $\texttt{Def}$ define a timed run as a sequence $(v^0, D)(v_1, d_1, \alpha_1)(v_2, d_2, \alpha_2) \ldots (v_n, \dagger)v_n$ where $(v^0, D)(v_1, \alpha_1)(v_2, \alpha_2) \ldots (v_n, \dagger)v_n$ is a run and for all $1 \leq i < n$, $d_i \in \Delta(c(\alpha_i))$ where $c(a) = c(\neg a) = a$. In the rest of this paper we let $\Omega^\tau(\psi|\texttt{Att}^\tau|\texttt{Def})$ be all timed runs over timed attacker $\texttt{Att}^\tau$ and defender $\texttt{Def}$.

Attacker Question 3 *Given an AD tree $\psi \in \mathcal{L}^w(\texttt{A}_a, \texttt{A}_d)$, a time horizon τ, timed attacker $\texttt{Att}^\tau$, non-deterministic defender $\texttt{Def}$ does there for all D exist a timed run $(v^0, D)(v_1, d_1, \alpha_1)(v_2, d_2, \alpha_2) \ldots (v_n, \dagger)v_n \in \Omega^\tau(\psi|\texttt{Att}^\tau|\texttt{Def})$ such that $[\![\psi]\!]v_n =$ and $\sum_{i=1}^{n-1} d_i \leq \tau$.*

Defender Question 3 *Given an AD tree $\psi \in \mathcal{L}^w(\texttt{A}_a, \texttt{A}_d)$, timed attacker $\texttt{Att}^\tau$, a time horizon τ and non-deterministic defender $\texttt{Def}$ does there exists a D such that for all timed runs $(v^0, D)(v_1, d_1, \alpha_1)(v_2, d_2, \alpha_2) \ldots (v_n, \dagger)v_n \in \Omega^\tau(\psi|\texttt{Att}|\texttt{Def})$ either $[\![\psi]\!]v_n = \texttt{ff}$ or $\sum_{i=1}^{n-1} d_i > \tau$.*

Technique 2 *For answering Attacker Question 3 for $\psi \in \mathcal{L}^w(\texttt{A}_a, \texttt{A}_d)$ with timed attacker $\texttt{Att}^\tau = (\texttt{Att}, \Delta)$ and non-deterministic defender $\texttt{Def}$ we will consider a symbolic transition system with states of the form (v, I) where $v \in \mathcal{V}$ and I is an interval of $\mathbb{R}_{\geq 0}$ and initial state $(v^0, [0, 0])$. From a symbolic state (v, I) we can do a symbolic transition $(v, I) \overset{a}{\rightsquigarrow} (v', I + I')$ (and $(v, I) \overset{\neg a}{\rightsquigarrow} (v, I + I')$ if $a \neq \dagger$) and $I' = \Delta(a)$ ($I' = [0, 0]$ if $a = \dagger$). Similarly to the traditional semantics we define $\texttt{Reach}((v, I))$ to be the set of reachable symbolic states from $\texttt{Reach}((v, I))$. Answering Attacker Question 3 with a time bound τ is now a matter of generating the sets $R_v\{(v', [a, b]) \in \texttt{Reach}((v^0, I)|a \leq \tau \wedge [\![v']\!]\psi = \texttt{tt}\}$ where $v \in \{v' \mid v^0 \overset{D}{\dashrightarrow} v'\}$ and verifying they are all non-empty. Conversely if one of them is empty, we have found a solution to Defender Question 3.*

2.2 Stochastic Semantics

Our end goal is to have a fully stochastic model. For the defender part of the transition system we let the choice of defence measures be selected according to a probability mass function $\gamma_{\texttt{Def}} : 2^{\texttt{A}_d} \to [0, 1]$. A stochastic defender is thus a tuple $\texttt{Def}^{\mathcal{S}} = (\texttt{Def}, \gamma_{\texttt{Def}})$ where $\texttt{Def}$ is a defender and $\gamma_{\texttt{Def}}(D) \neq 0 \implies D \in \texttt{Def}(v^{t^0})$.

A stochastic attacker for ψ is a tuple $\mathtt{Att}^{\mathcal{S}} = (\mathtt{Att}^{\tau}, \gamma_{\mathtt{Att}}, \delta)$ where $\mathtt{Att}^{\tau} = (\mathtt{Att}, \Delta)$ is a timed attacker, $\gamma_{\mathtt{Att}} : \mathcal{V} \to \mathtt{A}_a \cup \{\dagger\} \to \mathbb{R}$ assigns a probability mass function to each state for selecting the action to perform and $\delta : \mathtt{A}_a \to \mathbb{R} \to \mathbb{R}$ assigns a probability density to the possible execution times of each action. A few requirements must be stated here

1. if $\gamma_{\mathtt{Att}}(v^t)(a) \neq 0$ then $a \in \mathtt{Att}(v^t)$ and
2. if $\delta(a)(r) \neq 0$ then $r \in \Delta(a)$.

The first requirement is simply expressing that the stochastic attacker may only select actions defined by the timed attacker while the second requirement expresses that only execution times inside the interval defined by Δ should be given a density. The final component we need before giving the stochastic semantics is a an environment $\mathtt{Env}$ that assigns success probabilities to the execution of individual atomic actions: formally for each action we let $\mathtt{Env}_a : \{a, \neg a\} \to]0, 1[$ be a probability mass function assigns a non-zero probability of succeeding an atomic attack.

Forming the core of a probability measure over runs of $\mathtt{LTS}(\psi|\mathtt{Att}^{\tau}|\mathtt{Def})$ with a stochastic attacker $\mathtt{Att}^{\mathcal{S}} = (\mathtt{Att}^{\tau}, \gamma_{\mathtt{Att}}, \delta)$ and stochastic defender $\mathtt{Def}^{\mathcal{S}} = (\mathtt{Def}, \gamma_{\mathtt{Def}})$, we define a measure over a structure $\pi = (v_0, I_0, \alpha_0)(v_0, I_0, \alpha_0) \dots (v_n, \dagger)v_n$ where for all i, I_i is an interval and $\alpha_i \in \{a, \neg a \mid a \in \mathtt{A}_a\}$ from v inductively as follows:

$$G_v^{\mathtt{Att}^{\mathcal{S}}|\mathtt{Def}^{\mathcal{S}}|\mathtt{Env}}(\pi) = (v_0 = v) \cdot \gamma_{\mathtt{Att}}(v)(c(\alpha_0)) \cdot \left(\int_{I_0} (\delta(c(\alpha_0))(\tau))\, \mathrm{d}\tau \right) \cdot \mathtt{Env}_{c(\alpha)}(\alpha_0) \cdot G_{v'}^{\mathtt{Att}^{\mathcal{S}}|\mathtt{Def}^{\mathcal{S}}|\mathtt{Env}}(\pi^1),$$

where $v \xrightarrow{\alpha_0} v'$, $c(a) = c(\neg a) = a$ and base case $G^{\mathtt{Att}^{\mathcal{S}}|\mathtt{Def}^{\mathcal{S}}|\mathtt{Env}}(v_n v_n) = \gamma_{\mathtt{Att}}(v_n)(\dagger)$. As we will notice, the structure π does not include defence measures and thus we are lacking the probabilistic behaviour of the defender. For taking this into account, we instead consider structures like $\Pi = (v^0, D)\pi$ to which we can easily assign a probability measure as follows $F_{v^0}^{\mathtt{Att}^{\mathcal{S}}|\mathtt{Def}^{\mathcal{S}}|\mathtt{Env}}(\Pi) = \gamma_{\mathtt{Def}}(D) \cdot G_v^{\mathtt{Att}^{\mathcal{S}}|\mathtt{Def}^{\mathcal{S}}|\mathtt{Env}}(\pi)$ where $v^0 \overset{D}{\dashrightarrow} v$. We will usually omit the v^0 subscript and thus whenever we write $F^{\mathtt{Att}^{\mathcal{S}}|\mathtt{Def}^{\mathcal{S}}|\mathtt{Env}}$ we really mean $F_{v^0}^{\mathtt{Att}^{\mathcal{S}}|\mathtt{Def}^{\mathcal{S}}|\mathtt{Env}}$.

With a measure over timed over runs, we are now ready to define the probability of a succesful timed attack: let $\omega = (v^0, D)(v_1, d_1, \alpha_1)(v_2, d_2, \alpha_2) \dots (v_n, \dagger)v_n$ be a timed run, then we define the indicator function for ψ and time bound τ as

$$1_{\psi,\tau}(\omega) = \begin{cases} 1 & \text{if } [\![\psi]\!] v_n \text{ and } \sum_{i=1}^{n} -1 d_i \leq \tau \\ 0 & \text{otherwise} \end{cases}.$$

Integrating $1_{\psi,\tau}$ over all runs yields the probability of a successful attak.

$$\mathbb{P}^{\mathtt{Att}^{\mathcal{S}}}(\psi, \tau) = \int_{\omega \in \Omega^{\tau}(\psi)} 1_{\psi,\tau} \mathrm{d}F^{\mathtt{Att}^{\mathcal{S}}|\mathtt{Def}^{\mathcal{S}}}$$

Attacker Question 4 *Given an AD-tree, a stochastic attacker, a stochastic defender and time bound τ; what is the probability of a succesful attack.*

Technique 3 *The technique we shall later apply for answering Attacker Question 4 is statistical model checking. This technique relies on having a simulator for the stochastic semantics that can generate timed runs up to a time bound τ. Each run is now either a succesful attack or a failed attack i.e. a Bernoulli experiment. Generating several runs we can, using classic statistical methods, estimate the probability with some confidence.*

2.3 Adding Cost

In the running example, we note that some of the attacks naturally will result in some cost for the attacker. The most obvious one being the "bribe" option. Therefore researchers started augmenting their modelling languages with costs. [2, 9, 10] . In this paper we are also considering a cost-based semantics but instead of a fixed cost per atomic attack a, we define a cost rate C_a and define the cost of a timed run $\omega = (v^0, D)(v_1, d_1, \alpha_1)(v_2, d_2, \alpha_2) \dots (v_{n,\dagger})v_n$ as $\mathtt{C}(\omega) = \sum_{i=1}^{n-1} \frac{1}{C_{c(\alpha_i)}} \cdot d_i$. Given a time bound τ we define, in the style of Gadyatskaya et al. [9], the cost of a timed bounded run to be the accumulated cost up to reaching the time bound or the accumulated cost before reaching a succesful state. Formally, if $\omega = (v^0, D)(v_1, d_1, \alpha_1)(v_2, d_2, \alpha_2) \dots (v_n, \dagger)v_n$ and $i = \max\{i \mid [\![\psi]\!]v_n = \mathtt{ff} \wedge \sum_{j=1}^{i-1} d_j \leq \tau\}$ then we we define $\mathtt{C}^\tau(\omega) = \sum_{j=1}^{i-1} \frac{1}{C_{c(\alpha_j)}} \cdot d_j$. For a stochastic attacker, $\mathtt{Att}^\mathcal{S}$, we can now define his expected cost of an attempted attack against a stochastic defender $\mathtt{Def}^\mathcal{S}$ within a time bound τ as

$$\mathbb{E}^{\mathtt{Att}^\mathcal{S}|\mathtt{Def}^\mathcal{S}|\mathtt{Env}}(\psi, \tau) = \int_{\omega \in \Omega^\tau(\psi)} \mathtt{C}^\tau(\omega)\, \mathrm{d}F^{\mathtt{Att}^\mathcal{S}|\mathtt{Def}^\mathcal{S}|\mathtt{Env}}$$

Attacker Question 5 *Given an AD-tree ψ, a stochastic attacker $\mathtt{Att}^\mathcal{S}$, a stochastic defender $\mathtt{Def}^\mathcal{S}$ and a time bound τ, what is the exepcted cost of an attack? i.e. calculate $\mathbb{E}^{\mathtt{Att}^\mathcal{S}|\mathtt{Def}^\mathcal{S}}(\psi, \tau)$.*

Technique 4 *As for Attacker Question 4 we answer Attacker Question 5 by applying statistical model checking. The approach is intuitively simple: generate a set of sample runs $\omega_1, \dots \omega_n$, for all i calculate $c_{\omega_i} = \mathbb{E}^{\mathtt{Att}^\mathcal{S}|\mathtt{Def}^\mathcal{S}|\mathtt{Env}}(\psi, \tau)$. The c_{ω_i}s are random variables from the same underlying distribution and thus we can estimate their expected value.*

For the remainder of the paper we require on-the-fly information of the cost of runs; which means that we need to annotate our states with cost-information. Let $\mathtt{LTS}(\psi|\mathtt{Att}^\tau|\mathtt{Def}) = (\mathcal{V}, \mathcal{V}_\dagger, v^0, \rightarrow, \rightarrow_\neg, \rightarrow_\dagger, \dashrightarrow)$ then a cost-annotated state is a tuple $c(v, r)$ where $v \in \mathcal{V}$ and $r \in \mathbb{R}$. We let $\mathcal{C}$ denote all cost-annotated states; and we say a cost-annotated-run is a sequence

$$\omega = ((v^0, c_0), D)((v_1, c_1), d_1, \alpha_1)((v_2, c_2), d_2, \alpha_2) \dots (v_n, c_n)(v_n, c_n)$$

where $c_o = c_1 = 0$,$(v^0, D)(v_1, d_1, \alpha_1)(v_2, d_2, \alpha_2)\dots(v_n, \dagger)v_n$ is a timed run and for all $2 \leq i \leq n$, $c_i = c_{i-1} + (\frac{1}{C_{c(\alpha_{i-1})}} \cdot d_{i-1})$. Lastly, for the remaining part of this paper we will have $\gamma_{\mathtt{Att}}$ map from cost-annotated states thus $\gamma_{\mathtt{Att}} : \mathcal{C} \to \mathtt{A}_a \to \mathbb{R}$. In addition we let $\mathcal{S}(\psi)$ be the set of all possible stochastic attackers for the AD-tree ψ.

2.4 Parameterised Attacker

The framework so far defines the interaction between a specific stochastic attacker and a specific stochastic defender. Howevxer, attackers may be defined in terms of parameters that define their stochastic behaviour. Formally, a parameterised attacker is a tuple $(\mathcal{P}, \mathcal{D}, B)$ where $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$ is a list of parameters each with finite domain D_i given by $\mathcal{D} : \mathcal{P} \to 2^{\cup_{i=1}^{n} D_i}$ and $B : \times_{i=1}^{n} D_i \to \mathcal{S}(\psi)$ gives the stochastic attacker corresponding to a given parameter setting.

Attacker Question 6 *Let ψ be an attack defence-tree, $\mathtt{Def}^{\mathcal{S}}$ be a stochastic defender, $\mathtt{Env}$ be and environment, and let t $(\mathcal{P}, \mathcal{D}, B)$ be a parameterised attacker where $\mathcal{P} = \{p_1, \dots, pn\}$ and for all i, $\mathcal{D}(p_i) = D_i$.*

Find $Q = \arg\min_{q \in \times_{i=0}^{n} D_i} \mathbb{E}^{B(q)|\mathtt{Def}^{\mathcal{S}}|\mathtt{Env}}(\psi)$.

Technique 5 *For answering our final question of optimising parameters we will apply the statistical method* ANalysis Of VAriance *(ANOVA).*

ANOVA Analysis of variance is a collection of statistical tests developed to compare whether one or more discrete factors *has a significant effect on a continuous* response *variable. In the following we consider the one-factor design where we only have one factor and wishes to determine if this factor affects the response [14]. Consider we have a single factor with k levels, then for each level i we associate a random variable $\mathcal{X}_i$ giving the response values. Then an anova analysis tests the hypothesis that $\mathbb{E}(\mathcal{X}_0) = \mathbb{E}(\mathcal{X}_1) \cdots = \mathbb{E}(\mathcal{X}_n)$ i.e. whether the mean reponse is the same for all the levels. For each level i we obtain samples $X_i = \{x_0^i, x_1^i, \dots x_m^i\}$ and denote by $\bar{X}_i = \sum_{k=0}^{m} \frac{1}{m} x_k^i$ the mean of these and by $\bar{X} = \frac{1}{n*m} \sum_{l=0}^{n} \sum_{k=0}^{m} x_k^l$ the average of all samples.*
ANOVA now calculates what is called the F-statistic

$$F = \frac{(\frac{1}{n-1}) \sum_{i=0}^{n} (\bar{X}_i - \bar{X})^2}{\frac{1}{m-n} (\sum_{i=0}^{n} \sum_{k=0}^{m} (x_k^i - \bar{X})^2)}$$

which if the hypothesis is true should follow a F-distribution with $n-1$ and $m-n$ degrees of freedom. Now, as usual with hypothesis, we can calculate a value p which characterises how likely it is to obtain a value of F if the hypothesis is true. If p is less than some predetermined α then we reject the hypothesis. ANOVA is only capable of determining whether there is a difference between the levels, but is inadqueate for finding which level is the one being different. For finding the different one we need to apply a post-hoc *test that will perform a comparison between all the pairs. One such test is Tukeys-Test which will compare all pair-wise means and return whether they are different - with some significance.*

Optimising with ANOVA The approach we take for finding the optimal parameter for minimising the expected cost of an attack is to iteratively generating samples for each configuration until a significant difference of α is found using ANOVA. Afterwards we apply Tukeys test - and for each pair of configurations that are significantly different from each other we remove the one with the smallest mean. In Algorithm 1 `Simulate`*(ψ,c,τ) simulates the AD-tree ψ with the configuration c and returns the cost over that run,* `Anova`*(simulations) runs the anova analysis and returns the $p-value$,* `FilterTukey`*(simulations) runs Tukeys test and filter out the configuration with smallest cost for each significantly differents pairs of configurations.*

A small caveat with Algorithm 1 is that it may spin into an infinite loop if ANOVA determines there is significant difference while Tukeys-Test find no different pairs. In this case no filtering occurs and thus the algorithm will continue generating samples (and as ANOVA already determined there is a difference, α will never get larger than $1-\alpha$ to terminate. In practice we overcome this problem by limiting the number of times the algorithm can determine there is a difference without removing a configuration.

```
Data: Set of configurations C, a time bound $\tau$ and an AD-tree $\psi$, samples per iteration $x$
foreach $c \in$ C do
    $simulations[c] = \emptyset$;
end
while $\neg conf$ do
    foreach $c \in$ C do
        foreach $i \in \{0, 1, \ldots, x\}$ do
            $simulations[c] = simulations[c] \cup$ Simulate($\psi$,c,$\tau$)
        end
    end
    $a$ =Anova($simulations$);
    if $a > 1 - \alpha$ then
        $conf$ = tt;
    end
    else if $a < \alpha$ then
        $C$ =FilterTukey($simulations$)
    end
end
return C
```

Algorithm 1: Parameter Optimisation Algorithm.

3 Instantiating the framework

The preceding section has defined a general modelling framework for attackers and defenders. In this section we define one way of defining a parameterised attacker. The first parameter we will consider adding is a cost budget which is the maximal amount of resources an attacker can use during an attack. Given a cost budget variable, B, we define a cost-preserving stochastic attacker as one that assign higher probabilities to atomic attacks that preserves most of the cost budget.

In the following we let $\mathtt{Att}^{nd}$ be the fully non-deterministic attacker - i.e. if $\psi \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$ then $\mathtt{Att}((A, D) = \mathtt{A}_a \setminus A$ if $[\![D, a]\!]\psi = \mathtt{ff}$ otherwise $\dagger$.

Attack	Cost Rate	Probability
is	4	0.80
bs	5	0.70
t	5	0.70
b	5	0.70
st	2.5	0.50
ba	4.25	0.60
im	3.5	0.50
ot	0	0.60

Table 1: Parameters of the running example: the Cost column is the cost-rate of attack; while probability is success probability of the attack.

Definition 6. *Let $\psi \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$, let B be a cost budget and let $\mathcal{C}$ be the cost-annotated states; then a cost-preserving stochastic attacker is a stochastic attacker $\mathtt{Att}^{\mathcal{S}}(B) = (\mathtt{Att}^{\tau}, \gamma^{C}_{\mathtt{Att}}, \delta)$ with $\mathtt{Att}^{\tau} = (\mathtt{Att}^{nd}, \Delta)$ where*

- *$\delta(a)$ is a uniform distribution over $\Delta(a)$,*
- *$\gamma^{C}_{\mathtt{Att}}((v, c))(a) = \frac{W(a)}{\sum_{b \in \mathtt{A}_a} W(b)}$ where*
  - *$W(a) = \frac{B - (c + C_a \cdot d_h)}{B}$ with $\Delta(a) = [d_l, d_h]$ if $B - (c + C_a \cdot d_h) > 0$ and $a \in \mathtt{Att}(v)$ - otherwise 0.*

  if $a \in \mathtt{A}_a$ and $W(a) \neq 0$,
- *$\gamma^{C}_{\mathtt{Att}}((v, c))(\dagger) = 1 - \sum_{a \in \mathtt{A}_a} \gamma^{C}_{\mathtt{Att}}((v, c))(a)$ and*
- *$\delta((v, c))(r) = \frac{1}{d_h - d_l}$ if $r \in [d_l, d_h]$ and $\Delta(a) = [d_l, d_h]$.*

It is not unreasonable to think that some attackers may have higher tendency to take some attacks - even though they are more costly. Based on this thought we adapt the cost-preserving attacker to one with multiple parameters - one for each atomic attack. We call these parameters for likelyness-parameters.

Definition 7 (Parameterised-Attacker). *Let $\psi \in \mathcal{L}(\mathtt{A}_a, \mathtt{A}_d)$ where $\mathtt{A}_a = \{a_1, \dots, a_n\}$ be an AD-tree, let B be a cost budget, let $\mathtt{Att}^{\mathcal{S}}(B) = (\mathtt{Att}^{\tau}, \gamma^{C}_{\mathtt{Att}}, \delta)$ be the cost-preserving stochastic attacker, let $\mathcal{C}$ be the cost-annotated states and let $D \subseteq \mathbb{R}_{\geq 0}$ be some finite domain. Then we define our parameterised attacker $\mathtt{Att}^{\mathcal{SP}}(B) = (\mathcal{P}, \mathcal{D}, B)$ where $\mathcal{P} = \{p_a | a \in \mathtt{A}_a\}$, $\mathcal{D}(p_a) = D$ for all $a \in \mathtt{A}_a$, and $B(p_{a_1}, \dots, p_{a_n}) = (\mathtt{Att}^{\tau}, \gamma^{P}_{\mathtt{Att}}, \delta)$ where*

- *$\gamma^{P}_{\mathtt{Att}}((v, c))(a) = \frac{W(a)}{\sum_{b \in \mathtt{A}_a} W(b)}$ where $W(d) = p_d \cdot \gamma^{C}_{\mathtt{Att}}(d)$ for $a \in \mathtt{A}_a$ and*
- *$\gamma^{P}_{\mathtt{Att}}((v, c))(\dagger) = 1 - \sum_{b \in \mathtt{A}_a} \gamma^{P}_{\mathtt{Att}}((v, c))(b)$*

Example 1. As an example of how the parameters may affect expected cost of an attacker we consider four different configurations for our running example; namely 1) all likelyness parameters are 1 , 2) all likelyness parameters are 1500 , 3) all likelyness parameters are 1 except for *threaten* which is 1500 and 4) all likelyness parameters are 1 except for *threaten* and *Bribe* which are 1500.

Estimating the expected cost for these different configurations with a time limit of 1000 yield expected cost 229 ± 35 for Item 1, 208 ± 33 for Item 2, 120 ± 20 for Item 3 and finally for Item 4 we get 102 ± 20. All of the estimates were computed using 30 simulations.

4 Experiments

The framework developed has been translated into a UPPAAL SMC [8] timed automata model based on a textual description of the AD-tree, the cost parameters and their timing intervals. Also the textual description includes the probabilities of an atomic attack to be successful. In the following we will use these scripts in conjunction with UPPAAL to answer the questions raised throughout the running text.

4.1 Encoding

Although this encoding is specific for the instantiation of Section 3 we note that it shows the applicability of UPPAAL SMC to encode the general framework from Section 2. The encoding of an AD-tree within UPPAAL SMC follows along the lines of [9] with one automaton modelling the defender, one automaton modelling the attacker, and one automaton modelling the environmental choice of an outcome for the execution of an atomic attack. To coordinate their behaviour the automata synchronises over channels for indicating what action the attacker wants to perform - and boolean variables to maintain the discrete state of the AD-tree (i.e. what atomic attacks and defenses are true at any given time). In the following paragraph double concentric circles denotes the initial state of an automaton.

Defender In Figure 3 we present the template of the Defender Timed Automaton for an AD-tree with the defender actions $\mathtt{A}_d = \{p1d, p2d, p3d\}$. The automaton inititate by selecting a subset of $\mathtt{A}_d$ and setting the corresponding boolean variables (e.g. b_p1d) to true. Afterwards, the defender do nothing. Since there are no weights on the edges from the initial locations, the choice of an edge is a uniform choice - corresponding to the uniform choice in our stochastic semantics previously given.

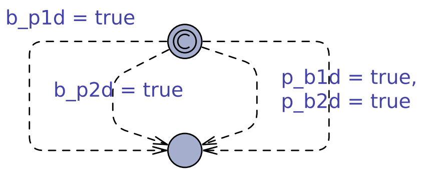


Fig. 3: The Defender Automaton

Attacker Let $\mathtt{Att}^{\mathcal{S}p}(B) = (\mathcal{P}, \mathcal{D}, B)$ with $\mathcal{P} = \{p_{a\_1}, p_{a\_2}\}$ be a parameterised attacker with attacker actions $\mathtt{A}_a = \{a1, a2\}$; then we show in Figure 4 how to encode the attacker given by $B(T1,T2)$. Here the AD-tree has $\Delta(ai) = [L\_ai, H\_ai]$ and a cost rate of C_ai/H_ai for executing a_i. The automaton keeps tracks of the currently used resources in the clock $usedRes$. Initially the automaton can go to a dead-end if one of two conditions are true: either (1) the tree is already true ($\{t\}$) or (2) for all atomic

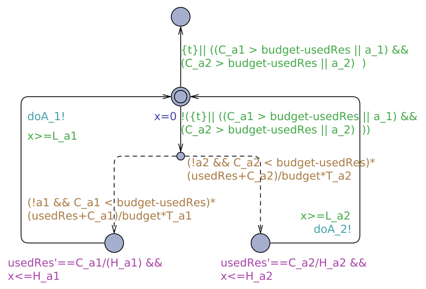

{t}|| ((C_a1 > budget-usedRes || a_1) &&
(C_a2 > budget-usedRes || a_2))
doA_1!
x>=L_a1
x=0
!({t}|| ((C_a1 > budget-usedRes || a_1) &&
(C_a2 > budget-usedRes || a_2)))
(!a2 && C_a2 < budget-usedRes)*
(usedRes+C_a2)/budget*T_a2
(!a1 && C_a1 < budget-usedRes)*
(usedRes+C_a1)/budget*T_a1
x>=L_a2
doA_2!
usedRes'==C_a1/(H_a1) &&
x<=H_a1
usedRes'==C_a2/H_a2 &&
x<=H_a2

Fig. 4: The Attacker Automaton

actions it is either the case that they cannot be performed without risking exceeding the budget (i.e. $(C\_ai > budget - usedRes)$) or they are already true. In case this edge is not enabled the automaton will instead go to a probabilistic choice to choose what action to perform according to the weights previously described in Section 3. After selecting an attack a_i, the attacker goes to a location where a delay between $d \in [L\_ai, H\_ai]$ is selected and the clock $usedRes$ is increased by $d \cdot (C\_a1/H\_a1)$. After this, the automaton synchronises on $doAi!$ to tell the environment that he attempted to perform ai.

Environment An example of an environment automaton is shown in Figure 5. Initially it awaits $doA?$ synchronisation from the attacker after which it instantly selects a result. A succesful execution happends with probability p while an unsuccesful execution occurs with probability $1 - p$. If succesful the boolean variable b_A is set to true.

In practice there is one of these loops for each attacker proposition of the AD-tree.

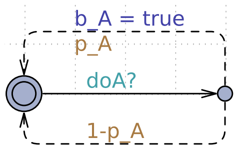


Fig. 5: The Environment Automaton

4.2 Estimating Probability of Attack

Consider that we use the Cost-preserving attacker profile, and are given a time bound τ and consider the defender select defence measures according to a uniform distribution; then we can answer Attacker Question 4 and Attacker Question 5 by using the statistical model checking capabilities of UPPAAL SMC. For Attacker Question 4 we simply ask the question $\mathtt{Pr}[<=\tau]\{\mathtt{t}\}$ which will return a confidence interval for performing a successful attack.

For Attacker Question 5 we use the query $\mathtt{E}[<=\tau; \mathtt{nbRun}](\mathtt{max} : \mathtt{usedRes} * (1 - \{\mathtt{t}\}))$, where nbRun is the number of runs used by UPPAAL SMC to estimate the cost and $usedRes$ is a UPPAAL variable counting the cost. The $(1 - \{t\})$ is a technicality to make sure $usedRes$ stop increasing after the attack has been successful.

In Table 2 we show the expected cost and probability of a successful attack. Estimating the cost was done with 500 runs in all cases.

4.3 Parameter Optimisation

We have applied Algorithm 1 (with UPPAAL SMC as a simulating backend) to our running example with the likelyness parameters obtaining values in the range $\{1, 150\}$ and with a time bound of 1000. The x in the algorithm was set to 10. The algorithm determined that 64 of the 256 configurations yield higher expected cost. The remaining ones are indistinguishable. In Figure 6 we show boxplots for the various configurations ordered such that configurations deemed optimal by ANOVA are at the left while configurations yielding higher expected cost are to the right. Visually we notice that

Bound	Cost	Probability
10	34.78	0.08
50	143.28	0.52
100	221.64	0.84
200	192.697	0.95
400	232.75	0.95

Table 2: Probability Estimates and expected cost for the Cost-preserving attacker for various time bounds.

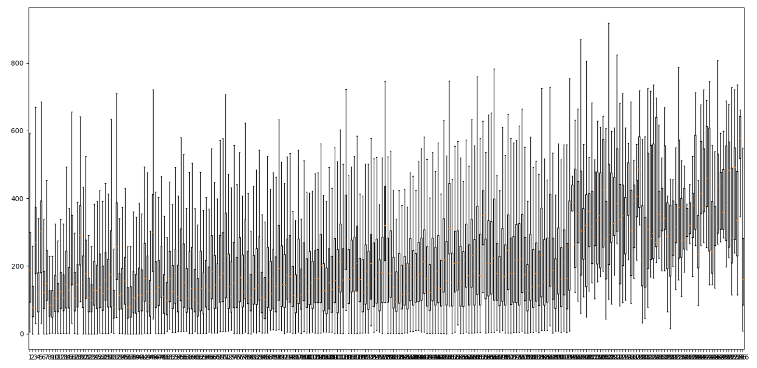


Fig. 6: Sampled data from the ANOVA analysis.

there is a seperation between these groups two groups.

5 Conclusion

In this paper we have developed a temporal and stochastic semantics for attack-defense trees. The stochastic semantics distinguishes itself from similar work of [9] by allowing time-dependent cost functions. The attack-defense tree is translated into UPPAAL SMC stochastic timed automata, and it is shown how this tranlsation may answer several of the questions posed throughout the running: among others is estimating the expected cost of an attacker. This paper also includes a parameterised description of attackers which leads to the interesting question of finding optimal paramter settings. We develop a method based upon the statistical test anova and observes that our algorithm found 64 out of 256 configurations to yield higher cost than the remining ones.

In the future we intend to continue our work in expanding the expressive power of our framework; and we wish to handle temporal dependencies between attacker actions: currently, the attacker is capable of choosing to bribe a subject before actually having identified the subject. Another interesting work is to allow defenders to do counter measures while the attacker is attacking and create a more game-like feeling of the semantics.

Bibliography

[1] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In Mike Paterson, editor, *ICALP*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990. ISBN 3-540-52826-1.

[2] Zaruhi Aslanyan and Flemming Nielson. Pareto Efficient Solutions of Attack-Defence Trees. In *Principles of Security and Trust*, volume 9036, page 95, 2015. doi:10.1007/978-3-662-46666-7_6.

[3] Zaruhi Aslanyan, Flemming Nielson, and David Parker. Quantitative verification and synthesis of attack-defence scenarios. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 105–119. IEEE Computer Society, 2016. doi:10.1109/CSF.2016.15.

[4] Alessandra Bagnato, Barbara Kordy, Per Håkon Meland, and Patrick Schweitzer. Attribute decoration of attack–defense trees. *International Journal of Secure Software Engineering (IJSSE)*, 3(2), 2012.

[5] Peter E. Bulychev, Alexandre David, Kim Guldstrand Larsen, Marius Mikucionis, Danny Bøgsted Poulsen, Axel Legay, and Zheng Wang. UPPAAL-SMC: statistical model checking for priced timed automata. In Herbert Wiklicky and Mieke Massink, editors, *Proceedings 10th Workshop on Quantitative Aspects of Programming Languages and Systems, QAPL 2012, Tallinn, Estonia, 31 March and 1 April 2012.*, volume 85 of *EPTCS*, pages 1–16, 2012. doi:10.4204/EPTCS.85.1.

[6] Alexandre David, Kim G. Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wasowski. Timed I/O automata: a complete specification theory for real-time systems. In Karl Henrik Johansson and Wang Yi, editors, *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2010, Stockholm, Sweden, April 12-15, 2010*, pages 91–100. ACM, 2010. ISBN 978-1-60558-955-8. doi:10.1145/1755952.1755967.

[7] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikučionis, Danny Bøgsted Poulsen, Jonas van Vliet, and Zheng Wang. Statistical model checking for networks of priced timed automata. In *FORMATS*, volume 6919 of *LNCS*, pages 80–96, 2011.

[8] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikucionis, and Danny Bøgsted Poulsen. Uppaal SMC tutorial. *STTT*, 17(4):397–415, 2015. doi:10.1007/s10009-014-0361-y.

[9] Olga Gadyatskaya, René Rydhof Hansen, Kim Guldstrand Larsen, Axel Legay, Mads Chr. Olesen, and Danny Bøgsted Poulsen. Modelling attack-defense trees using timed automata. In Martin Fränzle and Nicolas Markey, editors, *Formal Modeling and Analysis of Timed Systems - 14th International Conference, FORMATS 2016, Quebec, QC, Canada, August 24-26, 2016, Proceedings*, volume 9884 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2016. ISBN 978-3-319-44877-0. doi:10.1007/978-3-319-44878-7_3.

[10] Holger Hermanns, Julia Krämer, Jan Krčál, and Mariëlle Stoelinga. The value of attack-defence diagrams. In *Principles of Security and Trust*, pages 163–185. Springer, 2016.

[11] Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Patrick Schweitzer. Attack–Defense Trees. *Journal of Logic and Computation*, 24(1):55–87, 2014.

[12] Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer. DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. *Computer Science Review*, 13–14(0):1–38, 2014. doi:10.1016/j.cosrev.2014.07.001.

[13] Sjouke Mauw and Martijn Oostdijk. Foundations of attack trees. In *Proceedings of the International Conference on Information Security and Cryptology (ICISC 2005)*, volume 3935 of *Lecture Notes in Computer Science*, pages 186–198. Springer, 2005.

[14] Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, 2006. ISBN 0470088109.

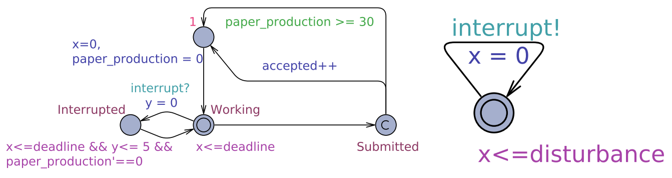
[15] Bruce Schneier. Attack trees: Modeling security threats. *Dr. Dobb's Journal*, December 1999.

A Stochastic Timed Automata

The modelling formalism we will use for encoding the stochastic semantics of AD-trees is Stochastic Timed Automata [7] - a stochastic extension of standard timed automata [1]. This formalism is effective for encoding the semantics of AD-trees. As an example of Timed Automata consider the two automata in Figure 7a and Figure 7b modelling a researher and his environment. The researcher continuously attempts to submit a paper with a fixed *deadline* - his clock x is counting the days since he started working on the paper and thus this clock must be less than deadline i.e. $x <= deadline$ (this is a location invariant(. While he is Working on his paper someone may *interrupt?* him whereafter he will be interrupted for $y <= 5$ days. While he is interrupted he cannot work on his paper thus the paper production is stalled ($paper\_production' == 0$). Sometime before the deadline, the paper will be Submitted and depending on how finished the paper is it can either be accepted (and we increment a counter *accepted*) or we do nothing and enters a waiting state. The paper must have had 30 days of work to have a possiblity of being accepted ($paper\_production >= 30$). The environment is much simpler, in the sense that it chooses some time between 0 and disturbance to interrupt the researcher.

Formally, a Timed Automaton (TA) is a finite automaton equipped with a set of real-valued counters ($\mathcal{X}$), called clocks, that measures the time-progression. Furthermore, a timed automaton may synchronise with other timed automata over a finite set of channels (Σ). For a set of channels Σ we let $\Sigma_o = \{a! \mid a \in \Sigma\}$ be the set of output actions over Σ while we let $\Sigma_i = \{a? \mid a \in \Sigma\}$ be the set of input actions. For a set of clocks $\mathcal{X}$ we call an element $c \bowtie n$ where $c \in \mathcal{X}$ and $n \in \mathbb{N}$ and $\bowtie \in \{\leq, <\}$ ($\bowtie \in \{\geq, >\}$) an upper (lower) bound over $\mathcal{X}$. Let $\mathcal{B}^{\leq}(\mathcal{X})$ ($\mathcal{B}^{\geq}(\mathcal{X})$) be the set of all upper (lower) bounds over $\mathcal{X}$.

Definition 8 (Timed Automaton). *A timed automaton (TA) is a 6-tuple $\mathcal{A} = (L, \ell_0, \mathcal{X}, \Sigma, \rightarrow, I, \mathcal{R})$, where 1) L is a finite set of control locations, 2) $\ell_0 \in L$ is the initial location, 3) $\mathcal{X}$ is a finite set of clocks, 4) Σ is a finite set of channels, 5) $\rightarrow \subseteq L \times \mathcal{B}^{\geq}(\mathcal{X}) \times (\Sigma_o \cup \Sigma_i) \times 2^{\mathcal{X}} \times L$ is a set of edges. We write $\ell \xrightarrow{g,a,R} \ell'$ for an edge where ℓ is the source and ℓ' the target location, $g \in \mathcal{B}^{\geq}(\mathcal{X})$ is a guard, $a \in \Sigma_o \cup \Sigma_i$ is a label, and $R \subseteq \mathcal{X}$ is the set of clocks to reset, 6) $I \colon L \rightarrow \mathcal{B}^{\leq}(\mathcal{X})$*



(a) A Researcher (b) An Environment

Fig. 7: A researcher and his environment.

is an invariant function, mapping locations to a set of invariant constraints and 7) $\mathcal{R} : L \to \mathcal{X} \to \mathbb{R}$ *assign rates to the individual clocks in each location.*

We call a mapping $v : \mathcal{X} \to \mathbb{R}$ for a valuation over $\mathcal{X}$ and denote all valuations over $\mathcal{X}$ by $\mathcal{V}(\mathcal{X})$. Let $v \in \mathcal{V}(\mathcal{X})$ be a valuation, $R : \mathcal{X} \to \mathbb{R}$ give clocks rates, $d \in \mathbb{R}$ be a real-valued number and let $X \subseteq \mathcal{X}$; then we let $(v + d \cdot R)$ be the valuation v' where $v'(x) = v(x) + d \cdot R(x)$ and we let $v[X]$ be the valuation v'' for which $v''(x) = 0$ if $x \in X$ and agrees with v otherwise. If $g = c \bowtie n$ is a clock bound over $\mathcal{X}$ and $v \in \mathcal{V}(\mathcal{X})$ then we v satisfies g ($v \vDash g$) iff $v(c) \bowtie n$. This is straightforwardly generalised to a set of clock bounds.

The state of timed automaton $\mathcal{A} = (L, \ell_0, \mathcal{X}, \Sigma, \to, I, \mathcal{R})$ is a tuple (ℓ, v) where $\ell \in L$ and $v \in \mathcal{V}(\mathcal{X})$. From a state (ℓ, v) the timed automaton may 1) do a timed transition $(\ell, v) \xrightarrow{d} (\ell, v')$ if $v' = (v + d \cdot \mathcal{R}(\ell))$ and $v' \vDash I(\ell)$ or 2) do a discrete transitions $(\ell, v) \xrightarrow{a} (\ell', v')$ if there exists $\ell \xrightarrow{g,a,r} \ell'$ such that $v \vDash g$, $v' = v[r]$ and $v' \vDash I(\ell')$.

Following the compositional framework of David et al. [6] we require that a timed automaton for any state s is 1) *input-enabled* i.e. for any $a! \in \Sigma_o$ there exists some s' such that $s \xrightarrow{a!} s'$ and 2) *action-deterministic* i.e. if $s \xrightarrow{a} s'$ and $s \xrightarrow{a} s''$ then $s' = s''$.

Network Of Timed Automata Let $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ be timed automata where $A_i = (L_i, \ell_0^i, \mathcal{X}_i, \Sigma, \to_i, I_i, \mathcal{R}_i)$. Also let $\mathcal{S}(\mathcal{A}_i) = L_i \times \mathcal{V}(\mathcal{X}_i)$; then we define the states of the network $\mathcal{A}_1 \| \mathcal{A}_2 \| \ldots \| \mathcal{A}_n$ to be tuples $(s_1, s_2, \ldots, s_n)$ where $s_i \in \mathcal{S}(\mathcal{A}_i)$. A network may transit from $(s_1, s_2, \ldots, s_n)$ by

- a timed transition $(s_1, s_2, \ldots s_n) \xrightarrow{d} (s'_1, \ldots, s'_n)$ if for all i, $s_i \xrightarrow{d} s'_i$ or
- a discrete transition $(s_1, s_2, \ldots, s_n) \xrightarrow[i]{a!} (s'_1, \ldots, s'_n)$ if $s_i \xrightarrow{a} s'_i$ and for all $j \neq i$, $s_j \xrightarrow{a?} s'_j$.

A.1 Stochastic Semantics

The stochastic semantics of Stochastic Timed Automata was originally laid out by David et al. [7]. The semantics is given as repeated races among the networks constituent components in which all of the components choose a delay - according to some delay distribution - and the winner of the race being the one with the smallest selected delay. Afterwards the winning component selects an output-action to which the remaining components must follow. Having performed the action a new race commences.

The actual delay distribution is selected in the following way: if the possible delays are bounded, the distribution is a uniform distribution between between the minimal delay before some action is possible and the maximal delay where a delay is still possible.

Formally, we assume there for any state (s) of any TA $\mathcal{A}$ exists a delay-density $\delta_s^{\mathcal{A}} : \mathbb{R} \to \mathbb{R}$ and a probability mass function $\gamma_s^{\mathcal{A}} : \Sigma_o \to \mathbb{R}$. Naturally we will require these functions to be "sane" in the sense that they do not assign probability mass (density) to impossible actions (delays) i.e. $\gamma_s^{\mathcal{A}}(a!) \neq 0$ ($\delta_s^{\mathcal{A}}(a!) \neq 0$) implies $s \xrightarrow{a!} s'$ ($s \xrightarrow{d} s'$).

Let $\omega = a_1!a_2! \ldots a_n!$ be a finite sequence of output-actions: then we define the probability of a network $\mathcal{A}_1 \| \ldots \mathcal{A}_m$ generating the sequence from state $s = (s_1, \ldots S_n)$ recursively as:

$$F_s(\omega) = \sum_{i=0}^{m} \left(\int_{t>0} \delta_{s_i}^{A_i}(t) \cdot \prod_{j \neq i} \left(\int_{\tau > t} \delta_{s_j}^{A_j}(t) d\tau \right) \cdot \gamma_{s_i^t}^{A_i}(a_1! \cdot F_{s'}(\omega^1) dt \right),$$

where $s_i \xrightarrow{d} s_i^d$, $s \xrightarrow{d} \xrightarrow[i]{a_1!} s'$, $\omega^1 = a_2! \ldots a_n!$ and base case $F_s(\epsilon) = 1$.

Example 2. Returning to our previous timed automata in Figure 7a and Figure 7b and instantiate them with *deadline* = 60 and *disturbance* = 10, we can with UPPAAL SMC [5, 8] ask questions such as "What is the probability that the researcher within 500 days has 4 papers accepted?". In UPPAAL SMC this is phrased like `Pr`$[<= 500](<>$ `Researcher.accepted` $>= 4)$, and UPPAAL SMC repsonds by giving a 95%−confidence interval of $[0.30, 0.39]$. Alternatively we can estimate the number of finished papers within 500 days by asking the query `E`$[<= 500; 5000]($`max` : `Researcher.accepted`$)$ which returns 2.0 ± 0.034.