Attack Trees with Time Constraints

Aliyu Tanko Ali, Damas Gruska

Comenius University, Mlynska Dolina 842 48, Bratislava, Slovak Republic

Abstract

We propose how attack trees formalism can be extended with time constraints. An attack tree is a basic description of how an attacker can compromise an asset, we refine this basic description by adding time constraints which can prevent an attacker from reaching the root node, if the attack actions performed cannot be completed within the defined time constraint. Adding time to attack trees causes an infinite number of possible states, to overcome this problem, we translate the tree into (an extended version of) timed automata and later use UPPAAL verification tool to analyse.

Keywords

Attack trees, timed attack trees, cyber-physical systems, security, threat modelling, timed automata, reachability

1. Introduction

Attack tree's security. The revolution that brought the introduction of assets such as IoTs, CPS, and industrial control systems etc., in the last decades also brought many security challenges. At its inception, attack trees are used to model how an asset (mostly static) may be compromised and allow a security engineer to plan on how to address the potential security threats. For example, in its early days, attack trees were used to model how to gain access (open) to secure documents, how a PGP encrypted file or password can be cracked, potential ways to infect a system files with a virus, how unauthorized users can obtain admin privileges, and how to gain remote access to a system [1, 2, 3] etc. In recent days, attack trees are used for security and risk assessment of assets such as SCADA systems [4], IoT systems [5], CPS [6], and medical equipment [7] etc. It is important to note that while attack trees remain a powerful graphical security tool that can be used to identify how an asset may be compromised, the shift in the dynamics of the assets i.e. from modelling and analysing single static systems to dealing with complex and dynamic (sometimes run concurrently) raised some questions in the effectiveness of using attack trees to analyse certain assets.

The settings of (traditional) attack trees are to depict (static) varying ways an asset may be compromised. However, most assets nowadays have erratic behaviour and can interact with other (sub)systems. This makes their vulnerabilities change according to the threat environments. Therefore, to capture such dynamism using attack trees, the estimated annotations (i.e. nodes) need to be updated regularly. Attempts have been made to extend attack trees through the introduction of attack-defence trees [8], attack protection trees [9], sequential and parallel attack

29th International Workshop on Concurrency, Specification and Programming (CS&P'21)

aliyu.ali@fmph.uniba.sk (A. T. Ali); gruska@fmph.uniba.sk (D. Gruska)

CEUR Workshop Proceedings (CEUR-WS.org)

trees [10], and attack countermeasure trees [11] etc. These proposed extensions introduced how potential threats can be refuted. For example, if an attack tree models potential ways an asset may be attacked (i.e. access to a secure sever room), the security engineer having good knowledge of the assets surroundings, will design a security defence (hidden or open) to defend the asset. These concepts are proven effective to assets that can be modelled with finite number of nodes (states). The asset is modelled with an attack tree, and all possible attack paths are blocked with defence or attack countermeasures. However, for a large and complex asset that has infinitely many states, this concept is ineffective. Another important point is that; most assets nowadays are safety-critical. They do require a timely response to threats. This means apart from identifying possible ways an asset may be compromised, the model has to also provide a means to slow down or prevent the attack. In previous papers [9, 12], we investigated the concept of attack trees for stand-alone assets. In [9] we proposed the use of protection nodes as an alternative to defence nodes. However, we realised that such a method is less effective to CPS assets even with a small number of states. The challenge here is, CPS assets interact with a wide range of objects (i.e. routing signs, cameras, buildings) and respond accordingly. Each of these objects has its characteristics, and an attacker can manipulate these objects in different ways (e.g. blur, blocked, add) to compromise the asset (e.g. DoS, message falsification attacks). As such, a single pre-defined countermeasures or defence nodes is not enough to stop potential attacks. In [12] we explored informally an idea to introduce time constraints in the set of parent nodes in addition to the associated gates refinement. In this idea, we assume that even if the threat environment of an asset changed, new vulnerabilities that emerged are connected to an existing parent nodes. Although these vulnerabilities might not be refuted with the already existing defence or protection nodes for other vulnerabilities, the time constraints defined on the parent node will still be apply to the new vulnerabilities that connect to the parent node.

In this paper we push forward our approach, we formally defined attack trees with time constraints, we provided a case study to highlight situations where such model is applicable. Also, we translated the concept into a (weighted) timed automata, and use a formal verification tool UPPAAL to analyse. Let q be a parent node that is associated with a gate refinement, assume we want to prevent an attacker from achieving the parent node. We introduce a set of constant time intervals τ that is represented by a constant pair $< b, f >$, with b marking the start of an attack and f marking the end of the attack on a parent node. $b, f \in \mathbb{Q}$, and $b \leq f$. We associate the set of attack actions Act with a set of attack time T. An attack time defines the attack execution time for each action. To reach to the parent node, the attack time $t \in T$ for each action on a given child node under attack has to be less than or equal to the interval. In other words, if the attack time for child node(s) is more than the interval, the attacker cannot reach to the parent node. We formalized this idea and translated the parent node reachability to state reachability of timed automata.

Related work. Currently, threat modelling methods commonly used in industry mainly include graphical models, such as attack trees [1], and attack graph [13]. Among them, attack tree is a systematic attack scenario modelling method proposed by Schneier [1] and formally defined by Mauw and Oostdijk [2]. Since attack tree is a static model (and considered a semiformal model), several analysis frameworks have been proposed to establish its analysis

method based on formal methods. These analysis frameworks have been developed based on timed automata [14], petri nets [15], and stochastic games [16] etc. The authors of [17] developed a stochastic framework for quantitative analysis of ADTrees. The framework adopts ADTree methodology to represent attack scenarios in a simple graphical representation and performs quantitative assessment using CTMC analytical approach. The authors of [18] introduced a multi-objective optimization framework for attack trees whereby the leaves of the attack trees were enriched with various parameters such as cost, time, skills and resources. The framework supports the computation of a wide range of security metrics such as attack values, attack paths, and ranking. They translated each attack tree gate and leaf into a priced timed automata and analyse the framework via UPPAAL CORE. In another effort, the work in [19] developed a modelling framework for expressing the temporal behaviour of an attacker as a boolean formula, and use a model checking tools to perform fully automated analyses of the modelled system by performing both qualitative (boolean) and quantitative (probabilistic) analysis. The authors demonstrated an example using UPPAAL tool, a network of timed automata that shows the model of a thief who wants to enter a house while the resident is not at home. The work in [20] defined the semantics for arbitrary attackers in an attack-defence tree using schematic timed automata (STA), and implemented the model by translating it into UPPAAL SMC. The authors modelled the encoding of an AD-Tree with one automaton modelling the defender, one automaton modelling the attacker and a separate one modelling the environment of an outcome, and coordinated their behaviour through synchronising channels. Other related works are [21, 22, 11, 8].

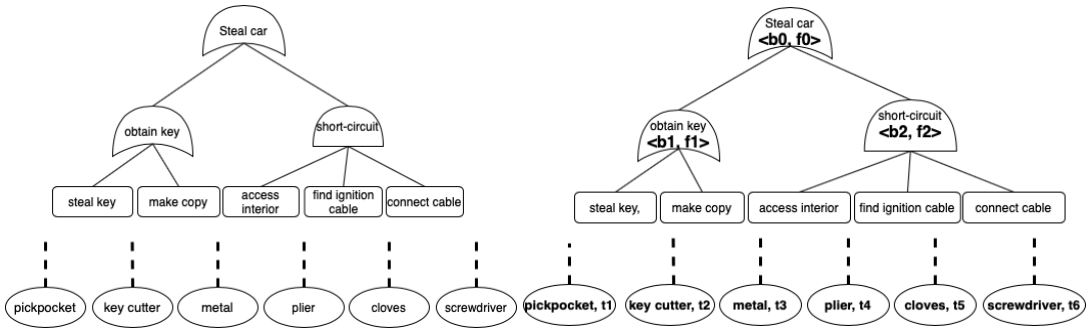
Unlike the works mentioned here, our work focus on traditional attack trees without considering defence nodes/actions. In our view, as mentioned earlier, defence actions are effective only to pre-identified vulnerabilities. As such cannot be applied to a (new) set of attack surface that emerged as a result of a change in the vulnerabilities landscape of the asset; modelled in an attack tree. Therefore, we shift our focus on preventing attacks by introducing a set of time constraints at the parent nodes; in addition to the gate refinement of the tree. We introduce a set of time constraints τ that is represented by a constant pair $< b, f >$, with b marking the start of an attack and f marking the end of the attack on a parent node. $b, f \in \mathbb{Q}$, and $b \leq f$. We associate the set of attack actions Act with a set of attack time T. An attacker can achieve the parent node if and only if the attack action(s) together with the attack time can be executed within the defined constant time intervals. To model the time constraints, we translate the tree into a parallel composition of weighted timed automata (WTA). The sets of nodes of the attack trees are translated to a set of locations in the weighted timed automata. For each leaf node in the attack tree, we have an automaton that represents a linear path from the leaf to the root node. Altogether their areas many WTAs as the leaf nodes in the attack tree. Each location that represents a leaf has a clock that is activated when there is an attack in the corresponding leaf in the tree. A transition is enabled only if a simple attack is successful in the attack tree.

The paper is organized as follows. In Section 2 we describe attack trees formalism. In Section 3 we present the motivation for enhanced restrictions on attack trees model, and why time constraint is a good option. In Section 4 we present time constraint as a form of security in attack trees, and Section 5 timed automata and the translation of attack trees with time constraint into (weighted) timed automata. Section 6 contains discussions and plans for future work.

2. Attack trees formalism

In this section, we describe and define attack trees. An attack tree is a graphical way of describing varying ways an asset may be compromised by a malicious user (an attacker). The structure of attack trees we use in this paper is based on the existing model introduced by Schneier [1] but here we introduce a set of attacker's actions Act (explain later) over the tree. $(\mathcal{Q}, \mathcal{E})$ is a tree, where $\mathcal{Q} = \{q_0\} \cup \mathcal{Q}_S \cup \mathcal{Q}_L$. $\{q_0\}$ is the root node of the tree, it represents attackers ultimate goal, $\mathcal{Q}_S$ is a set of internal nodes which we will refer to as sub-goal (also called parent nodes), they represent the decomposition of the root node into smaller units that are easier to solve, and $\mathcal{Q}_L$ is a set of leaf nodes (also called child nodes). The leaf nodes represent atomic nodes or end nodes (vulnerability), indicating an attack step. These sets of nodes are connected by a set of edges $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$. Each node in the attack tree (except for leaf nodes) has a gate refinement. A gate indicates (the fashion) how a node can be achieved (compromised). The interpretation of this fashion is on a node at a level above the current node. For this work, we make use of the AND and OR gates refinement. Informally, for a (parent) node with AND gate, it is said to have a set of (child) nodes, that are linked to the parent node by a set of edges and all these (child) nodes must be achieved first before the parent node is reached. For a (parent) node with OR gate, a single node from the set of child nodes when achieved is enough for the parent node to be reached. Formally we associate nodes with gates by the mapping $\mathcal{G} : \{q_0\} \cup \mathcal{Q}_S \to \{AND, OR\}$.

Definition 1. *An attack tree is a tuple $\mathcal{T} = (\mathcal{Q}, q_0, \mathcal{Q}_S, \mathcal{Q}_L, \mathcal{E}, \mathcal{G})$ where, $\mathcal{Q}$ is a finite set of nodes, q_0 is the root node of the tree, $\mathcal{Q}_S, \mathcal{Q}_L \subseteq \mathcal{Q}$ are two subsets such that $\mathcal{Q}_S \cup \mathcal{Q}_L = \mathcal{Q}$ and $\mathcal{Q}_S \cap \mathcal{Q}_L = \emptyset$, $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$ is a set of edges, connecting the nodes, and $\mathcal{G} : \{q_0\} \cup \mathcal{Q}_S \to \{AND, OR\}$ is a mapping that associate some nodes to a gate.*



Steal a car attack tree (A) An attack tree with time constraint (B)

Figure 1: Simple attack trees of how to steal a car that could be extended with time restriction

Note, to compromise nodes in the tree, an attacker needs to perform a set of attack actions. These set of (possible) actions are denoted by Act, and we defined an attack as a mapping $\mathcal{A} : \mathcal{Q}_L \to Act \cup \{Nil\}$ such that $A(l) = a$ means that an attacker can compromises leaf node $l \in \mathcal{Q}_L$ by executing an action $a \in Act$, $\mathcal{A}(l) = Nil$ when no action is performed. We say that attack $\mathcal{A}$ is simple if $\mathcal{A}(l) \neq Nil$ only for one leaf i.e. only one leaf is attacked. Practically, these set of attack actions are aided by the use of tools or/and techniques to carry out the attack.

Therefore, in this work, we will name a tool that can aid an attacker in executing the attack when referring to attack process in the working examples.

Example 1. *Shown in Figure.1 (A) is a simple attack tree that depicts possible ways to steal a car. The car can be stolen by achieving the sub-goal obtain key or short-circuit. To obtain the key, the sub-goal is of OR refinement and therefore, an attacker must either steal the key or make a copy. While to achieve short-circuit, the sub-goal is of AND refinement and the attacker must get access interior, find ignition cable, and connect the cable. Since the nodes must be achieved sequentially, in this case the AND gate is said to be extended to $SAND$ (sequential AND). Linked with dotted lines (below the tree) are a set of aided tools or/and techniques, and an attack is performed when a tool is mapped to a node i.e. $\mathcal{A}$(make copy) = key cutter.*

To model the attack propagation (i.e. from leaf nodes, through sub-goals to root), we define a state of attack tree, denoted by s i.e. state after an attacker has performed some actions from Act. A state of attack tree is defined as a set of nodes, and for each successful execution of attack actions (depending on the gate refinement of the target state), an attacker progresses to another state. An initial state is denoted by s_0. Let s be a state of an attack tree, and let $\mathcal{A}$ be an attack. The transition $\delta : (s, \mathcal{A}) \rightarrow s^{'}$ defines a change in state from s, when an attack $\mathcal{A}$ is performed, to state $s^{'}$. We define this for simple attacks but it can be extended for arbitrary ones.

Definition 2. *Let $\mathcal{A}$ be a simple attack such that $\mathcal{A}(l_i) = b$ and let s be an attack state. Then the next state s' after attack $\mathcal{A}$ is defined as $s^{'} = r(s \cup l_i)$ where operation r on the set of nodes is defined as follows: $t^{'} = r(t)$ iff t' is the smallest set with the following properties*

- $t \subseteq t'$
- *if q is a sub-goal that is associated with AND gate and all its child nodes are in $r(t)$ then $q \in r(t)$*
- *if q is a sub-goal that is associated with OR gate and at least one of its child node is contained in $r(t)$ then $q \in r(t)$,*

A state is called *final* if it contains the root node. The transitive closure of δ defines reachability and will consider only states reachable from s_0 by some attacks. Henceforth, we will use the notation $s \xrightarrow{\mathcal{A}} s'$ instead of $\delta : (s, \mathcal{A}) \rightarrow s^{'}$. In the following lemma, we can see that an attack can be decomposed into simple attacks, however, the order of executing the attacks is important in succeeding.

Lemma 1. *Given an attack tree T. Let $\mathcal{A}_1, \mathcal{A}_2$ be two simple attacks and s is a state. Then $s \xrightarrow{\mathcal{A}_1\mathcal{A}_2} s' \neq s \xrightarrow{\mathcal{A}_2\mathcal{A}_1} s'$, i.e. attacks are asymmetry.*

Proof 1. *By example. State s is composed of all the nodes needed to reach s', while s' contains all the nodes attacked by $\mathcal{A}_1$ and $\mathcal{A}_2$. As we can see from example 1, it is impossible to connect the cable before accessing the interior and/or before finding the ignition cable. As such ordering of attacks has influences on multiple simple attacks.*

Example 2. *Let $s, s^{'}$ be a state and its successor respectively. Suppose from Fig.1 (A) to access the interior, an attacker needs a plier. Then a transition $s \xrightarrow{\mathcal{A}} s'$ iff $\mathcal{A}$(access interior) = plier.*

3. Motivation for enhanced restrictions

In this section, we will present motivations for a new security concept that will be formally introduced in the next section. We start by presenting why it is important to have *enhanced restrictions* in attack trees that will serve as a form of security in addition to identifying varying ways an asset (modelled with attack trees) may be compromised. But first, we start by explaining enhanced restriction and why it is needed.

An attack tree is a basic description of how an attacker may compromise an asset, without the description of how to prevent or repel the attack. This can play well into potential attackers; by analysing an asset using attack trees to identify the set of vulnerabilities. The gates describe an order (restriction) that guides how a set of parent nodes can be achieved. For example, the AND gate refinement required an attacker to execute attack on all the child nodes (link to a parent node) before the attack succeeds. A more restrictive version of this was proposed in [10], where the attacker is required to achieve the (child) nodes in sequential order. Failing to achieve "all" the child nodes or " accordingly", will result in the attack process failing. Mechanisms like this can be added (hidden) to components that will help prevent an attack. However, assets such as CPS (i.e. interaction with objects from the physical environment which can be observed by the attacker), this can be uncovered easily. The following scenario motivates the need for extending attack trees (gates refinement) with time constraints.

Attack Scenario: Assume that an auto company developed an app tool that connects its customers with the service centre for

- *technical analysis*: whereby, some sensors in the vehicle can send data back to the manufacturer for intelligent and autonomous vehicle studies,
- *threat analysis*: whereby, safety or/and varying ways a vehicle can be in danger is identified and cautions message sent to the user.

Combined this, a security threat analysis of the vehicle can be carried out based on the threat environment (i.e. vehicle moving or parked), at each instance; an attack tree identifies potential threats and display either in the vehicles' onboard TFT LCD screen display or/and the user mobile phone as a notification. For each identified possible threat/fault, the app indicates the originating location (leaf node), other components in the vehicle that can be affected (sub-goals) and the resultant effect/damage (attack goal). Potential adversaries to the vehicle are classified as follows:

- *an insider*: someone with close relation to the auto company i.e. rough employee that directly/indirectly misuses his/her privileges,
- *generic attacker*: someone with the intention to exploit vulnerabilities, that can result in putting the vehicle to harm,
- *component failure*: part of vehicle components with rust/ware-out that can lead to damages.

For this paper, we will model working examples from a single threat environment i.e. the vehicle is at a parked position, and in our future work, we will consider working with a dynamic

threat environment. Now, consider the vehicle user who is notified (by the app) of potential danger. Even though the attack tree can (correctly) show the originating point and target, the user cannot prevent or slow down the attack process.

Example 3. *Let us revisit example 1. let us assume a case whereby an attacker has already made some progress with the attack (i.e. access interior and locate ignition cable) and (s)he is interrupted. By the settings of traditional attack trees model, the attacker is not restricted from returning later in time to complete the attack, or with the previous knowledge, restart the attack process.*

It is important that, apart from identifying possible ways the attacker can achieve the target, a security measure is defined that will constrain the attacker from unlimited attempts.

4. Time constraints and security

In this section, we extend the set of gates refinement with a set of time constraints. The time constraints is an addition to the already existing gate refinement that is associated with each parent node. We also associate each attack action with an attack time, indicating the time needed for an attacker to complete executing the action.

Given a set of attack actions Act, we introduce a set of attack time T. T is the time needed to complete an action that can result in compromising a node, denoted simply by $(a, t) \in Act \times T$ (see Figure.1 (B)). By doing so, we are extending the attack definition (defined in section 2) by $\mathcal{A} : \mathcal{Q}_L \rightarrow (Act \times T) \cup \{Nil\}$. The set of gates refinements mapping is also extended with a set of time constraints τ that is represented by a constant pair $\langle b, f \rangle$, with b marking the start of an attack and f marking the (expected) end of attack on the parent node such that $b, f \in \mathbb{Q}$, and $b \leq f$. This allows us to redefine the gates refinement mapping as $\mathcal{G} : \{q_0\} \cup \mathcal{Q}_S \rightarrow \{OR, AND\} \times \tau$. For the sake of simplicity, we denote this as $\langle b, f \rangle$. From the graphical representation shown in Figure.1 (B), one can easily derive these time extensions whereby $\langle b_0, f_0 \rangle$, $\langle b_1, f_1 \rangle$, and $\langle b_2, f_2 \rangle$ is added to the parent nodes (root node and sub-goals), meaning they can only be reached if an attacker can execute the attack within the interval respectively. Also, below with the dotted lines, attack time t_i is added to each action, where $i \in \{1 \dots 6\}$. Regardless of the gates refinement i.e. OR, AND, an attack can only succeed if the action(s) execution time does not exceed the time intervals at the gates of the (parent) node.

Example 4. *Let us consider attack tree shown in Figure.1(B), the parent nodes are extended with time intervals represented as pairs $\langle b_0, f_0 \rangle$ for the root node, $\langle b_1, f_1 \rangle$ for the OR sub-goal, and $\langle b_2, f_2 \rangle$ for the AND sub-goal. The attack actions (represented by tools) are also extended with attack time. Each time indicates the time needed to complete the attack. From the initial attack attempt, an attacker has until elapsed of t_i to complete the attack, otherwise the whole attack process is consider failed. If t_i is larger than the constant time interval for the connected sub-goal, the attack cannot succeed.*

Definition 3. ***Attack trees with time constraint.*** *Let $(\mathcal{Q}, \mathcal{E})$ be a tree, an attack tree with time constraint is a tuple $\mathcal{T}_r = (\mathcal{Q}, q_0, \mathcal{Q}_S, \mathcal{Q}_L, \mathcal{E}, \tau, \mathcal{G}, T, \mathcal{A})$, where, $\mathcal{Q}$ is a finite set of nodes, q_0 is the root node of the tree that represents the attack target, $\mathcal{Q}_S, \mathcal{Q}_L \subseteq \mathcal{Q}$ are two subsets such that $\mathcal{Q}_S \cup \mathcal{Q}_L = \mathcal{Q}$ and $\mathcal{Q}_S \cap \mathcal{Q}_L = \emptyset$, $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$ is a set of edges that connect the nodes, τ is*

a set of time constraints, $\mathcal{G} : \{q_0\} \cup \mathcal{Q}_S \rightarrow \{AND, OR\} \times \tau$ *is the mapping that associate some nodes to a gate and time constraint,* T *is a set of attack time, and* $\mathcal{A} : \mathcal{Q}_L \rightarrow (Act \times T) \cup \{Nil\}$.

Definition 4. *Given an attack tree with time constraint* $\mathcal{T}_r$*, an attack over the tree that can reach to the root node is defined as (assuming* $a \in Act, l \in \mathcal{Q}_L$ *and* $t \in T$*)*

- *If the tree has* AND *gates,* $\forall \mathcal{A}_1 \ldots \mathcal{A}_n : \forall t_i \ldots t_n$ *are less than or equal to the constant time interval on the parent node,*
- *If the tree has* OR *gates,* $\exists \mathcal{A}_i : t_i$ *is less than or equal to the constant time interval on the parent node.*

As an example, let us consider a sub-goal *short-circuit* from Figure. 1 (B), an attacker can succeed in achieving the sub-goal if and only if the summation of attack time $t_3 + t_4 + t_5$ does not exceed the time interval defined at $\langle b, f \rangle$.

From Figure.1 (B), $\mathcal{A}(\mathcal{Q}_L) = Nil$ if the attacker cannot complete the attack within the time constraint at the gates. The following lemma guarantees that under some conditions a parent node remains unreachable to a potential attacker regardless of the gate refinement associated with the parent node.

Lemma 2. *Given a sub-goal* S*, a set of time constraint defined at the gate refinement* $\langle b, f \rangle$*, and a set of attack (actions)* $X \subseteq Act$ *required to compromise* S*. The sub-goal is unreachable to an attacker if the attack execution time* $\forall i \in X$ *exceeds the attack time defined at the gate refinement.*

Proof 2. *Since each action has an attack time, we need to show that this attack time for all the child nodes of* S *exceeds the time constraint at the gate. Now if this action cannot be executed within the attack time, the sub-goal cannot be achieved.*

4.1. Enhanced restrictions and security

There are some relations between time constraint on the gates refinement and improved security for different kinds of systems; in general. For example, the use of idle time outs for web session is a common practice in the development of high-risk applications such as online banking platforms. Other instances where time constraint is used to improve the security of asset can be found on e-locks (eg. car central locking system) that can automatically locked itself after a specified passage of time [23]. The implementation of time restriction such as action time-out in assets such as ATMs [15], and SMART-doors [24] are further good examples.

In addition to other extensions of attack trees with security mechanism such as defence nodes or attack countermeasures, time constraint can be used to model and analyse different kinds of attack scenarios for different kinds of assets.

5. Timed automata

As in [12], we now consider the use of a weighted timed automata to analyse our model. A weighted timed automata (WTA) is an extension of timed automata [14] with cost/price

information on both locations and edges that can be used to solve several interesting problems. There exists a formal verification tool UPPAAL [25] that accepts WTA as its modelling language. Before we explain further, we first recall the definition of a timed automata. (henceforth, we refer to a location of timed automata by l).

Definition 5. *A timed automaton is a tuple $TA = (\mathcal{L}, \mathcal{L}_0, \mathsf{E}, \mathcal{C}, \mathcal{I}, T, f)$, where $\mathcal{L}$ is a finite set of locations, $\mathcal{L}_0 \subseteq \mathcal{L}$ is a set of initial locations, E is a finite set of synchronization actions (events), $\mathcal{C}$ is a finite set of clocks, $\mathcal{I} : \mathcal{L} \rightarrow \Phi(\mathcal{C})$ is an invariant, assigning to every location $l \in \mathcal{L}$ a clock constraint, $T \subseteq \mathcal{L} \times \mathsf{E} \times \Phi(\mathcal{C}) \times 2^{\mathcal{C}} \times \mathcal{L}$ is a set of transition relations such that $\langle l, a, \phi, c, l^{'} \rangle$ represents an edge from location l to location $l^{'}$ on symbol a. ϕ is a clock constraint, $c \subseteq \mathcal{C}$ is a set of clocks to be reset, and f is a final location.*

The semantics of a timed automaton TA is defined by associating a (labelled) transition system TA_{LTS} (defined in section 1) with it. A state in TA_{LTS} consists of a pair (l, v) whereby l is a location of TA and v indicates that for a clock c, v satisfies the label constraint $\mathcal{I}(l)$.

5.1. Weighted time automata and attack trees translation

In this subsection, we introduce the basics of a WTA and give the translation of attack trees with time constraint into a WTA.

A Weighted timed automata, otherwise known as price timed automata (PTA), is an extended version of timed automata (TA) with weight/cost information added on both locations and edges. To arrive at a location, the weight value defined at that location has to be satisfied, also, to enabled a transition via an edge, the weight value at that edge has to be satisfied. At the final/desired location, a global weight/cost which is the accumulated weight/cost along the run is calculated. With this accumulative weight/cost value, it is easy to calculate the distance or cost of travelling from point A to point B in different case studies. Formally, a weighted timed automata is defined as follows [25].

Definition 6. *The tuple $WTA = (\mathcal{L}, \mathcal{L}_0, \mathsf{E}, \mathcal{C}, \mathcal{I}, \mathcal{W}, \mathcal{WP}, Tf)$ is a weighted timed automaton, where $\mathcal{L}$ is a finite set of locations, $\mathcal{L}_0 \subseteq \mathcal{L}$ is a set of initial locations, E is a finite set of synchronization actions (events), $\mathcal{C}$ is a finite set of clocks, $\mathcal{I} : \mathcal{L} \rightarrow \Phi(\mathcal{C})$ is an invariant, assigning to every location $l \in \mathcal{L}$ a clock constraint, $\mathcal{W} : \mathcal{L} \cup \mathsf{E} \rightarrow \mathbb{N}^{n}_{\geq 0}$ is a function that assigns weight value to location and edge, $w : \mathcal{WP} \rightarrow \mathbb{Q}$ is a set of weight parameters that updates the weight value $\alpha : \mathcal{W} \rightarrow (\mathcal{W} \cup \mathcal{WP})$, T is a set of transitions such that $\langle l, \phi, a, c, \alpha, l^{'} \rangle$, where $l, l^{'} \in \mathcal{L}$ are the source and target locations, ϕ is a clock constraint, $a \in \mathsf{E}$, $c \subseteq \mathcal{C}$ is a set of clocks to be reset, and α is a parametric weight update, and finally f is a final location.*

The trace of a weighted timed automata is a sequence of states with the transitions across the states given as $\pi = l_0 \xrightarrow[c_0]{a_0}{}^{\alpha_0} l_1 \xrightarrow[c_1]{a_1}{}^{\alpha_1} l_2 ...$ such that

- there is always an initial location l_0 with an initial clock valuation $c_0 = 0$,
- for every $i \in \{1, .., k\}$, there is some transition $(l_i, \phi, a_i, c_i, \alpha_i, l_{i+1}) \in T$,
- a transition is enables only if; for every clock valuation v, there exists a constraint ϕ such that v satisfies ϕ,

- for every successful transition, a new clock valuation c_{i+1} is obtained by increasing every clock variable in c_i by a transition i and resetting all previous clocks 0.

Now, in other to analyse attack trees with time constraint using UPPAAL, we translate the tree into a parallel composition of weighted timed automata. The sets of nodes of the attack trees are translated to a set of locations in the weighted timed automata. For each leaf node in the attack tree, we have a WTA that represents a linear path from the leaf to the root node. Altogether there are as many WTAs as the leaf nodes in the attack tree. Each location that represents a leaf which has a clock that is activated when there is an attack in the corresponding leaf in the tree. An attack on a node in the tree represents an enabled transition in the WTA. Initially, clocks become active when events synchronized, and end with either a success or fail synchronization action.

More general, an attack tree is translated into a parallel composition of weighted timed automata $WTA_1 \dots WTA_n$ that represents linear path from the leaf to the root node. Given an attack tree with time constraint $\mathcal{T}_r$, the semantics of successfully reaching the root node that satisfies the weighted timed automata WTA can be given as $[\![\mathcal{T}_r]\!] \subseteq$ WTA if

- $[\![q_0]\!] =$ WTA, a final location $f \in$ WTA is always satisfied,
- $[\![\mathcal{Q}]\!]^{OR} = \{\text{WTA}_1 \dots \text{WTA}_n\}$such that at least WTA_i reached the final location f_i,
- $[\![\mathcal{Q}]\!]^{AND} = \{\text{WTA}_1 \dots \text{WTA}_n\}$such that for all A_i, the final location f_i reached.

Lemma 3. *Let $\mathcal{T}_r$ be an attack tree with time constraint and let WTA be the equivalent product of weighted timed automata. Suppose S is a (target) node, and location l is the (equivalent) translation in WTA. The number of the active clock(s) in WTA to reach l, is the same as the number of (attack) actions carried out by an attacker before reaching S.*

Proof 3. *We know that a clock is reset for each enabled transition in WTA, therefore, since the locations of WTA corresponds to the nodes in the attack trees, each enabled clock indicates an active attack on a node. As such, the number of attack executed will correspond to the number of clock(s) reset for events in the WTA.*

Theorem 1. *Let WTA be a product of weighted timed automata for the translation of $\mathcal{T}_r$. For a transition $(l_i, \phi, a_i, c_i, \alpha_i, l_{i+1})$, the clock c_i is inactive for a corresponding node $q \in \mathcal{Q}$ in the tree, if there is no active attack process on that node.*

Proof 4. *Directly from lemma 3.*

We can model an attack $\mathcal{A}$ as a special weighted timed automata, and denote it as $\mathcal{A}WTA$, this weighted timed automata shows the attackers' action and time when they are performed on the attack tree. We use this in the following theorem to check the attack target (root node) reachability for both the attack tree and the WTA.

Theorem 2. *Given an attack tree with time constraint $\mathcal{T}_r$ and a time automata WTA, an attacker A can only reach the root node of the tree $\{q_0\}$ iff corresponding WTA plus $\mathcal{A}WTA$ running with the attack path in the tree can reach the final location.*

Proof 5. *The main idea. We know that a root node can only be reached if an attacker can succeed in completing the attack process. Therefore we have to show that a WTA plus $\mathcal{A}WTA$ can also reach the final location.*

5.2. UPPAAL model

Uppaal accepts the synchronization of events using channels (input and output). As such, we modelled a set of events $a?, b?, c?, d?$ (receive) at the initial location to serve as a set of leave nodes. We use two clocks t and x, with t being a clock associated with each event while x

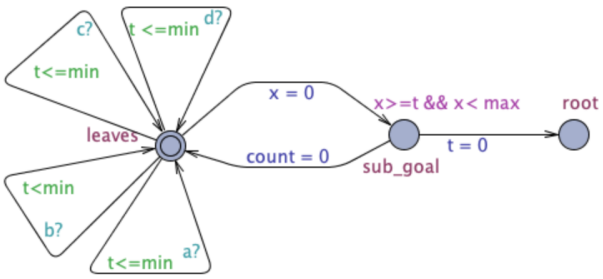


Figure 2: A simple UPPAAL model of nodes and time constraints

associated with a target location. Here, x is a clock to track t. We also define two invariant min and max, to check whether both t and x are satisfied before the transition to the target location is enabled.

The UPPAAL template shown in Figure.2 is a simple model of time constraint in achieving nodes in an attack trees. The *leaves* locations begin by waiting for the activation of signal by one of the events $a?, b?, c?, d?$ An event become activate only when the clock is less than (predefined) min. If the *sub_goal* location is of OR gate, the activation of a single event is enough for the transition to be enabled to the target location (*sub_goal*), otherwise all the events (i.e. AND gate) needs to be activated, and the *sub_goal* location can only be reached if the clock x is less than (predefined) max.

6. Discussion and future work

In this paper, we have presented attack trees with a time constraint to serve as a secured version of attack trees. We discussed how the gates refinement can be extended with time parameters and also propose how the attack trees with time constraint can be modelled using a formal verification tool UPPAAL.

As further work, we plan to study how attack trees can be used to analyse a CPS. A CPS is a special kind of asset that can interact with objects from the physical environment as well as other cyber-systems. This allows its operations to run parallel and concurrent, making it easy for a potential attacker to observe (from the physical components), and perform some dangerous attacks such as message falsification attack, DoS, message spoofing etc. by simply

adding, blocking or blurring the objects from the physical environment. This kind of threats cannot be captured using attack trees alone. We plan to investigate *opacity*, a security property formalizing the information leakage of a system to an external observer, namely intruder and study how it can be used with attack trees.

Acknowledgement

This work was supported by the Slovak Research and Development Agency under the Contract no. APVV-19-0220 (ORBIS) and by the Slovak VEGA agency under Contract no. 1/0778/18 (KATO).

References

[1] B. Schneier, Attack trees, Dr. Dobb's journal 24 (1999) 21–29.
[2] S. Mauw, M. Oostdijk, Foundations of attack trees, in: International Conference on Information Security and Cryptology, Springer, 2005, pp. 186–198.
[3] B. Kordy, L. Piètre-Cambacédès, P. Schweitzer, Dag-based attack and defense modeling: Don't miss the forest for the attack trees, Computer science review 13 (2014) 1–38.
[4] C.-W. Ten, C.-C. Liu, G. Manimaran, Vulnerability assessment of cybersecurity for scada systems, IEEE Transactions on Power Systems 23 (2008) 1836–1846.
[5] D. Beaulaton, N. B. Said, I. Cristescu, S. Sadou, Security analysis of iot systems using attack trees, in: International Workshop on Graphical Models for Security, Springer, 2019, pp. 68–94.
[6] F. Xie, T. Lu, X. Guo, J. Liu, Y. Peng, Y. Gao, Security analysis on cyber-physical system using attack tree, in: 2013 Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IEEE, 2013, pp. 429–432.
[7] M. A. Siddiqi, R. M. Seepers, M. Hamad, V. Prevelakis, C. Strydis, Attack-tree-based threat modeling of medical implants., in: PROOFS@ CHES, 2018, pp. 32–49.
[8] A. Roy, D. S. Kim, K. S. Trivedi, Attack countermeasure trees (act): towards unifying the constructs of attack and defense trees, Security and Communication Networks 5 (2012) 929–943.
[9] A. T. Ali, D. P. Gruska, Attack protection tree., in: CS&P, 2019.
[10] F. Arnold, D. Guck, R. Kumar, M. Stoelinga, Sequential and parallel attack tree modelling, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2014, pp. 291–299.
[11] X. Ji, H. Yu, G. Fan, W. Fu, Attack-defense trees based cyber security analysis for cpss, in: 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), IEEE, 2016, pp. 693–698.
[12] A. T. Ali, Simplified timed attack trees, in: International Conference on Research Challenges in Information Science, Springer, 2021, pp. 653–660.
[13] X. Ou, A. Singhal, Attack graph techniques, in: Quantitative Security Risk Assessment of Enterprise Networks, Springer, 2012, pp. 5–8.

[14] R. Alur, Timed automata, in: International Conference on Computer Aided Verification, Springer, 1999, pp. 8–22.
[15] B. Berthomieu, M. Diaz, Modeling and verification of time dependent systems using time petri nets, IEEE transactions on software engineering 17 (1991) 259.
[16] L. S. Shapley, Stochastic games, Proceedings of the national academy of sciences 39 (1953) 1095–1100.
[17] K. Lounis, S. Ouchani, Modeling attack-defense trees' countermeasures using continuous time markov chains, in: International Conference on Software Engineering and Formal Methods, Springer, 2020, pp. 30–42.
[18] R. Kumar, E. Ruijters, M. Stoelinga, Quantitative attack tree analysis via priced timed automata, in: International Conference on Formal Modeling and Analysis of Timed Systems, Springer, 2015, pp. 156–171.
[19] O. Gadyatskaya, R. R. Hansen, K. G. Larsen, A. Legay, M. C. Olesen, D. B. Poulsen, Modelling attack-defense trees using timed automata, in: International Conference on Formal Modeling and Analysis of Timed Systems, Springer, 2016, pp. 35–50.
[20] R. R. Hansen, P. G. Jensen, K. G. Larsen, A. Legay, D. B. Poulsen, Quantitative evaluation of attack defense trees using stochastic timed automata, in: International Workshop on Graphical Models for Security, Springer, 2017, pp. 75–90.
[21] I. A. Tøndel, M. G. Jaatun, M. B. Line, Threat modeling of ami, in: Critical Information Infrastructures Security, Springer, 2013, pp. 264–275.
[22] A. E. M. AL-Dahasi, B. N. A. Saqib, Attack tree model for potential attacks against the scada system, in: 2019 27th Telecommunications Forum (TELFOR), IEEE, 2019, pp. 1–4.
[23] D. Ray, The time structure of self-enforcing agreements, Econometrica 70 (2002) 547–582.
[24] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling tcp throughput: A simple model and its empirical validation, in: Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication, 1998, pp. 303–314.
[25] P. Bulychev, A. David, K. G. Larsen, M. Mikučionis, D. B. Poulsen, A. Legay, Z. Wang, Uppaal-smc: Statistical model checking for priced timed automata, arXiv preprint arXiv:1207.1272 (2012).