A Stochastic Framework for Quantitative Analysis of Attack-Defense Trees

Ravi Jhawar, Karim Lounis(✉), and Sjouke Mauw

CSC/SnT, University of Luxembourg, Luxembourg, Luxembourg
{ravi.jhawar,karim.lounis,sjouke.mauw}@uni.lu

Abstract. Cyber attacks are becoming increasingly complex, practically sophisticated and organized. Losses due to such attacks are important, varying from the loss of money to business reputation spoilage. Therefore, there is a great need for potential victims of cyber attacks to deploy security solutions that allow the identification and/or prediction of potential cyber attacks, and deploy defenses to face them. In this paper, we propose a framework that incorporates Attack-Defense trees (ADTrees) and Continuous Time Markov Chains (CTMCs) to systematically represent attacks, defenses, and their interaction. This solution allows to perform quantitative security assessment, with an aim to predict and/or identify attacks and find the best and appropriate defenses to reduce the impact of attacks.

Keywords: Attack-Defense Trees · Markov chains · Security modeling · Quantitative analysis

1 Introduction

Cyber attacks are becoming more and more technically sophisticated, and well organized. Losses due to such attacks are important, varying from the loss of money to business reputation spoilage. On the other side of the coin, in order to fend and stop this destructive cyber attacks wave, research efforts on cyber attacks and security have considerably risen, trying to come with the best solutions that allow security engineers to predict cyber attacks, estimate their likelihood, and find the most feasible defenses to prevent or reduce the negative impact of these cyber attacks. As a consequence of these research efforts, a great number of graphical models have been proposed in the last two decades (e.g., attack trees [14], attack graphs [7], attack-countermeasure trees [17], and attack-defense trees [10]) and have been widely used for cyber security modeling and

The research leading to the results presented in this work received funding from the European Commission's Seventh Framework Programme (FP7/2007–2013) under grant agreement number 318003 (TREsPASS) and Fonds National de la Recherche Luxembourg under the grant C13/IS/5809105 (ADT2P).

G. Barthe et al. (Eds.): STM 2016, LNCS 9871, pp. 138–153, 2016.
DOI: 10.1007/978-3-319-46598-2_10

assessment. In spite of their similarities, these models differ on how to model attacks and defenses, and how to integrate aspects like time, and dependencies between actions within the model. The perfect model should be easy to use, and practically implementable. It should provide a user-friendly and comprehensive representation of real-life security scenarios, and should integrate aspects like time and dependencies as well as security assessment functions. These requirements are well defined by the ADTrees model [10], which extends attack trees [14] with refinable defenses, and allows the representation of sequential dependencies between actions. It also supports security assessments of attributes such as the likelihood, the cost, and/or the efficiency of attacks/defenses.

ADTrees are defined as a graphical methodology used to represent security scenarios by systematically representing the different actions that an attacker may undertake to realize a security goal, and the different actions that a defender may apply to stop the attacker's actions from being realized. It comes with a strong formal framework for reasoning about security scenarios through different types of semantics (propositional, multiset, De Morgan lattice, and equational), and has proven to be simple, easy to use, and yet powerful in its modeling capability. It has been validated in a large industrial case study [4].

To perform quantitative security assessment for evaluating attributes like cost, efficiency, time, and probabilities, ADTrees apply a bottom-up procedure [14]. Unfortunately this procedure can *only* be used for attribute evaluation under the assumption that *all considered actions (attacks/defenses) are independent.* This is a very strong assumption which is unrealistic in practice, since actions are usually dependent, or in the simplest case sequentially dependent. To overcome this limitation, we propose a new approach for security assessment of ADTrees involving dependencies between actions. This approach relies on Continuous Time Markov Chains. Being a powerful model, provided with a useful quantitative analysis approach, CTMCs tend to be the perfect candidate to assess ADTrees involving dependencies. Inspired by authors of [1–3,12,15], we model atomic attacks/defenses using an exponential distribution. In fact, exponential distribution seems to be a suitable distribution to model a great number of attacks/defenses like brute force attacks, adaptive defense mechanisms (e.g., moving target defenses), or countermeasures with delayed impacts like policies execution. In this paper, we propose a framework that combines the graphical and formal methodology of ADTrees with CTMCs, and allows performing a system's security assessment. The framework takes as an input an ADTree representing a security scenario, and transforms it into a CTMC. This CTMC is then used to perform the security assessment through the evaluation of security attributes such as likelihood, the mean time required by an attack scenario, and other attributes that security engineers may define. To achieve this, we define a new semantics for ADTrees in terms of CTMCs. These semantics express how to translate attacks/defenses into individual CTMCs and how to combine these individual CTMCs into one final CTMC representing the entire ADTree.

Related work. Over the last two decades, a number of graphical security models (e.g., attack trees [14], attack graphs [7], and attack-countermeasure trees [17]) have been proposed in the literature and have been widely used for cyber security modeling and assessment. Moreover, due to the development of cyber attacks in terms of techniques, dependencies, and organization, these models have been enriched and elevated in order to correctly model and assess sophisticated cyber attacks. For instance, attack trees have been enriched and augmented with adaptable countermeasures to become ADTrees [10]. Performing quantitative analysis on these models usually goes through applying an analytic approach such as Markov chains [1–3,12,15], Petri-Nets [5,16], or Bayesian networks [11].

The choice of the analytical approach depends mainly on the model itself, the aspects (time, dependencies) that it considers, and the user preference toward the approach. We find that the Markov chains approach has mostly been chosen for assessing these models. For instance, they have been applied in [2,3,15] on attack trees, and in [1,12] on attack graphs to perform quantitative analysis, and have shown their easy and useful applicability. Inspired by the previous works, we have chosen to apply the Markov chains approach on ADTrees for the following reasons: Models used in the previously cited works like attack trees [2,3,5,15,16], and attack graphs [1,12] do not define the modeling of defenses in their specification.

Although defense specification is not present in those models, some authors [1–3,5,12,15,16] tried to incorporate defenses to model security scenarios. Unfortunately, they have assumed that the defenses can totally, with no delay, mitigate a given attack. In other words, an attack node is simply deleted from the attack model when it is counter-defended. This assumption is too strong since it is not always the case for a defense to immediately stop an attack once the defense is set up. In fact, there exist defenses whose impact comes after a certain delay like a password changing policy. The ADTree model overcomes the limitation of modeling defense by nature, as it allows to model and represent defenses of different types independent of their impact delay. Secondly, compared to attack-countermeasure trees model, an ADTree model allows the refinement of defenses, which is more realistic. Kordy el al. [11] adopted ADTrees model and used Bayesian networks approach to assess the likelihood of security scenarios. This approach requires for each instant of time the construction of a conditional probability table for each action because of the stochastic dependency between actions. This requirement can be time consuming, error prone, and not practical when a large ADTree is evaluated. Thanks to CTMCs, we can represent the same information (conditional probability tables) using a temporal probability function known as CDF (Cumulative Distribution Function). This function is associated to each action, and provides for each given instant of time t the probability of occurrence of the action with respect to its dependencies.

Contributions. To summarize, the contributions of this paper are threefold.

- We define a new semantics of ADTree model in terms of CTMCs. The semantics express the way attacks/defense action must be represented as a CTMC, and how different CTMCs can be composed to represent the entire attack-defense tree.

- We use the composed CTMC that represents the attack-defense scenario to perform quantitative analysis. Given that attack trees are formally a sub-class of attack-defense trees, our analysis technique also applies to attack trees.
- We demonstrate the applicability of our solution using a simple but realistic example study.

Organization. Section 2 presents the basics of the attack-defense tree model and CTMC model. Section 3 defines our semantics for attack-defense trees in terms of CTMCs. Section 4 discusses the analytical approach of CTMCs to perform quantitative security assessment. Section 5 performs quantitative analysis on an example study. Finally, Sect. 6 provides conclusions and perspectives.

2 ADTrees and CTMCs

2.1 ADTrees

ADTrees are a graphical methodology used to represent security scenarios. They can be seen as a two-player game. The first player is qualified by proponent 'p', and the second by opponent 'o' [9]. Depending on the root of the attack-defense tree, if the root is an attack, then the proponent is the attacker, else it is the defender. Graphically, each performed action or accomplished sub-goal is represented by a node depicted by a red circle (○) if it refers to an attack action/subgoal, or by a green square (□) if it refers to a defense action/subgoal. Any node of either type in an ADTree can be refined (either disjunctively, conjunctively, or sequentially conjunctive), or countered by another node of the opposite type. Nodes that cannot be refined any further are qualified by basic actions. When a node is disjunctively refined, its accomplishment requires at least one of its refinement nodes to be accomplished. A conjunctively refined node requires all its refinement nodes to be realized without any prefixed order. The sequential conjunctive refinement is similar to the latter but requires a predefined accomplishment order for its refinement nodes. We depict a conjunctive refinement of a node by an arc over all edges connecting the node and its refinement nodes, and the sequentially conjunctive refinement with a directed arc. When a node is countered with another node of opposite type, they are linked together using a dashed line.

Figure 1 illustrates an ADTree for a simple networked system where the attacker wants to compromise a server host by executing malicious scripts. To achieve his goal, the attacker must first perform reconnaissance in order to gain knowledge about the network's assets (e.g., topology, protocols, addresses, open ports) using some tools like Nmap. On the other side, to prevent the attacker from gaining knowledge on the network, the defender can apply one of the two adaptive defenses. The first defense regularly changes IP addresses of network hosts and the second defense 'Mutable network' dynamically shuffles IP addresses, routing tables and topology of the network. In the second step, the attacker looks for any vulnerabilities using Nesus for instance. Then, using Metasploit for example, he exploits the discovered vulnerability, and executes a

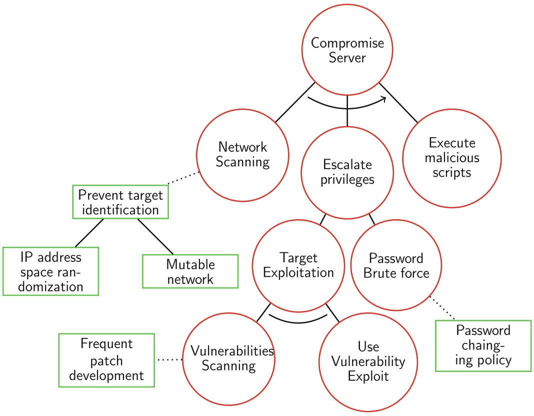
Compromise Server
Network Scanning
Escalate privileges
Execute malicious scripts
Prevent target identification
IP address space randomization
Mutable network
Target Exploitation
Password Brute force
Frequent patch development
Vulnerabilities Scanning
Use Vulnerability Exploit
Password chaing- ing policy

Fig. 1. An example ADTree representing the security scenario

specific designed payload to gain high privileges on the target host. To fend this, the defender frequently performs penetration tests to discover vulnerabilities and develop appropriate patches. As an alternative to target exploitation through vulnerabilities, the attack can brute force the root's password to gain the privilege. The defender implements in this case a policy to periodically change the passwords. This will delay the attacker from succeeding his goal. Finally, if the attacker manage to escalate the privileges, he can execute malicious command and cause harm to the server.

Given that this multi-step attack must be performed in a particular order, we use a sequential conjunction refinement. To realize the 'Escalate Privilege' attack, the attacker must either successfully brute force the root password or (disjunction refinement) exploit a discovered vulnerability and (conjunction refinement) running its dedicated Exploit and payload program.

2.2 Formal Definition of ADTrees

Formally, ADTrees are defined by means of an abstract syntax called ADTerms [10]. The ADTerms in this paper are typed over the signature $\Sigma = (\mathbb{S}, \mathbb{F})$, where:

- $\mathbb{S} = \{p, o\}$ is the set of types (proponent p and opponent o)
- $\mathbb{F} = \{(\vee_k^p)_{k\in\mathbb{N}}, (\wedge_k^p)_{k\in\mathbb{N}}, (\overrightarrow{\wedge}_k^p)_{k\in\mathbb{N}}, (\vee_k^o)_{k\in\mathbb{N}}, (\wedge_k^o)_{k\in\mathbb{N}}, (\overrightarrow{\wedge}_k^o)_{k\in\mathbb{N}}, c^p, c^o\} \cup \mathbb{B}^p \cup \mathbb{B}^o$ is a set of function symbols.

The unranked functions $(\vee_k^s)_{k\in\mathbb{N}}, (\wedge_k^s)_{k\in\mathbb{N}}$ and $(\overrightarrow{\wedge}_k^s)_{k\in\mathbb{N}}$, where $s \in \mathbb{S}$, represent the disjunctive ($\vee$), conjunctive ($\wedge$), and sequential conjunction ($\overrightarrow{\wedge}$) refinement operators for the proponent and the opponent, respectively. The binary functions c^s connect an action of a given type $s \in \mathbb{S}$ with an action of the opposite type $\overline{s} \in \mathbb{S}$. If we model the proponent as the attacker, then the set $\mathbb{B}^p$ (and respectively $\mathbb{B}^o$)

consists of atomic attacks (and atomic countermeasures). Conventionally $\overline{p} = o$ and $\overline{o} = p$.

Definition 1. *ADTrees are closed terms over the signature* $\Sigma = (\mathbb{S}, \mathbb{F})$*, generated by the following BNF-grammar, where* $b^s \in \mathbb{B}^p \cup \mathbb{B}^o$*, and* $s \in \mathbb{S}$ *is the type of players.*

$$t :\equiv b^s \mid \vee^s(t, \ldots, t) \mid \wedge^s(t, \ldots, t) \mid \overrightarrow{\wedge}(t, ..., t) \mid c^s(t, t) \tag{1}$$

Example 1. If we label the basic events of the ADTree in Fig. 1 by $b_0^p, b_1^p, b_2^p, b_3^p, b_4^p, b_0^o, b_1^o, b_2^o$, and b_3^o, respectively for Network scanning, Vulnerability scanning, Use vulnerability Exploit, Password brute forcing, Execute dangerous commands, IP randomization, Mutable network, patches development, and finally password policy, then the resulting ADTerm of the ADTree is:

$$t = \overrightarrow{\wedge}^p \left(c^o \left(b_0^p, \vee^o \left(b_0^o, b_1^o \right) \right), \vee^p \left(\wedge^p \left(c^o(b_1^p, b_2^o), b_2^p \right), c^o(b_3^p, b_0^3) \right), b_4^p \right) \tag{2}$$

2.3 Continuous Time Markov Chains

Markov chains [13] are stochastic processes used to model system behavior where probabilistic events are considered. They are called Markovian since the predictions are made based only on the current state of the system, and not on any previous state. A Markov process that transits from one state to another via an exponential rate is called Continuous Time Markov Chain or CTMC.

Definition 2. *A continuous time Markov chain is a tuple* (S, G, π)*, where:*

- S *is a finite disjoint set of states,*
- $G\colon S \times S \to \mathbb{R}$ *is the infinitesimal generator matrix which gives the rate of transition between two states* $s \in S$ *and* $s' \in S$*,*
- $\pi\colon S \to [0, 1]$ *is the initial probability distribution on* S*.*

The proposed semantics for ADTree requires to differentiate between the initial state, intermediate states, and the final states. Therefore, we slightly modify Definition 2 and adapt it to our needs. The new explicit notation (Definition 3) will help us to easily formalize and define our semantics. Moreover, the initial distribution π is usually devoted to the initial state of the system. Therefore, we omit the variable π from the definition, since we arbitrary devote the entire initial distribution to the initial state.

Definition 3. *An enumerated continuous time Markov chain* M *is a tuple* (S, S_0, S_*, G)*, where:*

- S *is a finite disjoint set of states,*
- $S_0 \subset S$ *is a finite set of initial states,*
- $S_* \subset S$ *is a finite set of final states,*

- *$G\colon S \times S \to \mathbb{R}$ is the infinitesimal generator matrix which gives the rate of transition between two states $s \in S$ and $s' \in S$.*

We note that there exists a set of intermediate states that we denote by $S_{mid} \subset S$, where $S = S_0 \cup S_{mid} \cup S_*$, and $S_0 \cap S_{mid} \cap S_* = \emptyset$.

The infinitesimal generator matrix G defines the exponential rates $g_{s,s'}$ of the transitions that go from one state $s \in S$ to an other state $s' \in S$. The element $g_{s,s}$ of the infinitesimal generator matrix G are chosen such that each row of the matrix sums to zero. Therefore, the generator matrix G is built as follows:

$$G = \begin{cases} -\sum_{s \neq s'} g_{s,s'} & \textit{if} \quad s = s', \\ g_{s,s'} & \textit{otherwise}. \end{cases} \tag{3}$$

Here, each $g_{s,s'} \geq 0$ represents the exponential rate of transition from state $s \in S$ to state $s' \in S$. The inverse $1/g_{s,s'}$ represents the average time needed to transit from $s \in S$ to $s' \in S$ and $|1/g_{s,s}|$ is the average amount of time (sojourn time) spent in state $s \in S$. Furthermore, for a given state $s \in S$, if $\forall s' \in S,\ s' \neq s$, $G(s,s') = 0$, then state $s \in S$ is called an absorbing state and M a continuous time absorbing Markov chain.

3 Markov Chain Semantics for Attack-Defense Trees

We now define the semantics for ADTrees in terms of CTMC. In particular, we first define the semantics for basic events $b^s \in \mathbb{B}^p \cup \mathbb{B}^o$, followed by the semantics for the three refinement operators $(\vee_k^s)_{k \in \mathbb{N}}, (\wedge_k^s)_{k \in \mathbb{N}}$ and $(\overrightarrow{\wedge}_k^s)_{k \in \mathbb{N}}$, where $s \in \mathbb{S}$, and finally the semantics for counteractions c^s (see Sect. 2.2). We then use the semantics of these ADTree components to compose a single CTMC that represents the semantics of the complete ADTree.

Semantics for basic events. Consider a set $\mathbb{M}$ of all possible CTMCs. We can then define a function $\Psi\colon \mathbb{B} \to \mathbb{M}$ that associates, for each basic event $b^s \in \mathbb{B}^p \cup \mathbb{B}^o$, a CTMC defined as $(\{s_0, s_*\}, \{s_0\}, \{s_*\}, G^{b^s})$, where s_0 and s_* are the initial and final states, respectively. The element G^{b^s} represents the infinitesimal generator to the CTMC. It is computed using Eq. 3, and hence given by Eq. 4:

$$G^{b^s} = \begin{bmatrix} -\lambda_{b^s} & \lambda_{b^s} \\ 0 & 0 \end{bmatrix} \tag{4}$$

Figures 2-a and b, illustrate the CTMC corresponding to basic events $b^p \in \mathbb{B}^p$ and $b^o \in \mathbb{B}^o$, respectively. The rates λ_{b^p} and μ_{b^o} represent the exponential rates of an atomic attack and an atomic countermeasure, respectively.

Semantics for conjunctive refinements. We define an unranked function $\wedge_{k \in \mathbb{N}}\colon \mathbb{M}^k \to \mathbb{M}$ which takes k Markov chains and composes them in a way that all k Markov chains should be executed in an irrelevant execution order. Therefore, the composed Markov chain is $(\prod_{i=1}^k S^{b_i}, \prod_{i=1}^k S_0^{b_i}, \prod_{i=1}^k S_*^{b_i}, G^{\wedge_{k \in \mathbb{N}}^s})$. The set $S^{\wedge_{k \in \mathbb{N}}^s}$ contains all possible combinations of the states of the k involved

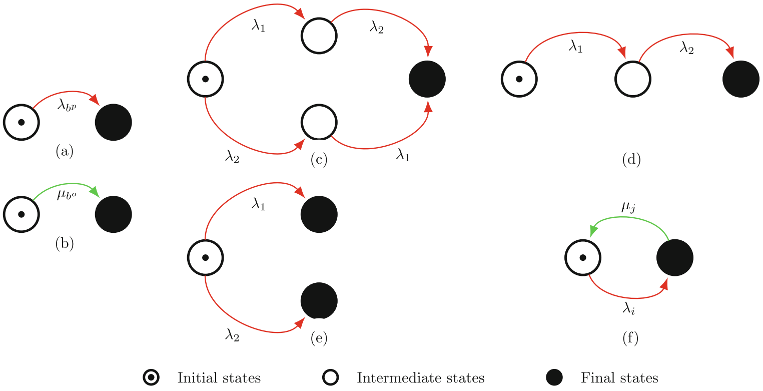

Fig. 2. CTMC-Semantics of basic events, refinements, and countermeasures.

Markov chains. The initial state is equal to $(s_0^{b_1}, s_0^{b_2}, ..., s_0^{b_k})$, and similarly, the final state is $(s_*^{b_1}, s_*^{b_2}, ..., s_*^{b_k})$. The remaining states refer to intermediate (transitive) states.

Figure 2-c illustrates the CTMC obtained by applying the function $\wedge_2^s$ on CTMC M_1^s and M_2^s, where M_1^s and M_2^s are two CTMCs corresponding to two basic events $b_1^s, b_2^s \in \mathbb{B}$. The generator $G^{\wedge^s_{k\in\mathbb{N}}}$ is obtained by the following equation:

$$G^{\overrightarrow{\wedge}^s{}_{k\in\mathbb{N}}}(s_i, s_j) = \begin{cases} -\sum_{i\neq j} G^{\wedge^s_{k\in\mathbb{N}}}(s_i, s_j) & \mathit{if} \quad i = j, \\ 0 & \mathit{if} \quad i \neq j \quad \wedge \quad |s_i \mathit{\Delta} s_j| > 2, \\ 0 & \mathit{if} \quad s_i \in \mathit{\Omega}, \\ G^{idf}(S^{idf} \cap s_i, S^{idf} \cap s_j) & \mathit{otherwise}. \end{cases} \tag{5}$$

Where $\mathit{\Delta}$ is the symmetric difference between two sets, $|S|$ is the cardinality of a given set S, $\mathit{\Omega}$ is the set of absorbing states, and $idf = \vartheta(s_i \cap s_j)$ is a function which returns the identifier of the Markov chains from where the input sets belong to. For example, $\vartheta(s_0^1, s_5^2, s_8^2)$ returns $\{1, 2\}$. In summary, this formulation consists in identifying which transition (t_i^{idf}) is linking state s_i to state s_j. Note that this formulation is valid for sequential conjunction and disjunction refinement as well.

Semantics for sequential conjunctive refinements. We define the sequential conjunction refinements using the function $\overrightarrow{\wedge}_{k\in\mathbb{N}}\colon \mathbb{M}^k \rightarrow \mathbb{M}$ which takes k CTMCs as input and composes them sequentially. The final state of the n^{th} CTMC is merged with the initial state of the $n+1^{th}$ CTMC. Figure 2-d illustrates how two CTMCs are composed by $\overrightarrow{\wedge}_2^s$. The result of $\overrightarrow{\wedge}_{k\in\mathbb{N}}$ composition is a CTMC $(S^{\overrightarrow{\wedge}^s{}_{k\in\mathbb{N}}}, S_0^{\overrightarrow{\wedge}^s{}_{k\in\mathbb{N}}}, S_*^{\overrightarrow{\wedge}^s{}_{k\in\mathbb{N}}}, G^{\overrightarrow{\wedge}^s{}^s_{k\in\mathbb{N}}})$ where:

- $S^{\overrightarrow{\wedge}^s{}_{k\in\mathbb{N}}} = S_0^{\overrightarrow{\wedge}^s{}_{k\in\mathbb{N}}} \bigcup S_*^{\overrightarrow{\wedge}^s{}_{k\in\mathbb{N}}} \bigcup S_{mid}^{\overrightarrow{\wedge}^s{}_{k\in\mathbb{N}}}$
- $S_0^{\overrightarrow{\wedge}^s{}_{k\in\mathbb{N}}} = \prod_{i=1}^k S_0^{b_i}$

- $S_*^{\overrightarrow{\wedge^s}_{k\in\mathbb{N}}} = \prod_{i=1}^{k} S_*^{b_i}$
- $S_{mid}^{\overrightarrow{\wedge^s}_{k\in\mathbb{N}}} = \bigcup_{i=1}^{k-1} \quad S_*^{b_i} \times S_0^{b_{i+1}} \quad \bigcup \quad S_{mid}^{b_i} \times S_0^{b_{i+1}} \quad \bigcup \quad S_*^{b_i} \times S_{mid}^{b_{i+1}}$

The set $S^{\overrightarrow{\wedge^s}_{k\in\mathbb{N}}}$ is composed of the initial state $S_0^{\overrightarrow{\wedge^s}_{k\in\mathbb{N}}}=\{(s_0^{b_1}, s_0^{b_2}, ..., s_0^{b_k})\}$, the final state $\{(s_*^{b_1}, s_*^{b_2}, ..., s_*^{b_k})\}$, and the intermediate states of S_{mid}, which is composed of intermediate states of the k involved Markov chains plus the linking states $\bigcup_{i=1}^{k-1} S_*^{b_i} \times S_0^{b_{i+1}}$(chains each CTMC with the next CTMC).

Semantics for disjunction refinement. We define a disjunctive refinement using an unranked function $\vee_{k\in\mathbb{N}}\colon \mathbb{M}^k \to \mathbb{M}$, which takes k CTMCs as input and composes them in a way that each CTMC can evolve independently to the other CTMCs. Therefore, there will be k final states (one from each k involved CTMC). The result of composing k CTMCs by means of a disjunction refinement is $(S^{\vee^s_{k\in\mathbb{N}}}, S_0^{\vee^s_{k\in\mathbb{N}}}, S_*^{\vee^s_{k\in\mathbb{N}}}, G^{\vee^s_{k\in\mathbb{N}}})$, where:

- $S^{\vee^s_{k\in\mathbb{N}}} = S_0^{\vee^s_{k\in\mathbb{N}}} \cup S_*^{\vee^s_{k\in\mathbb{N}}} \cup S_{mid}^{\vee^s_{k\in\mathbb{N}}}$,
- $S_0^{\vee^s_{k\in\mathbb{N}}} = \prod_{i=1}^{k} S_0^{b_i}$,
- $S_*^{\vee^s_{k\in\mathbb{N}}} = \bigcup\limits_{i=1}^{n} S_*^{b_i} \times \prod_{j\neq i} S_0^{b_j}$,
- $S_{mid}^{\vee^s_{k\in\mathbb{N}}} = \bigcup\limits_{i=1}^{n} S_{mid}^{b_i} \times \prod_{j\neq i} S_0^{b_j}$.

The set $S^{\vee^s_{k\in\mathbb{N}}}$ is composed of the initial state $(s_0^{b_1}, s_0^{b_2}, ..., s_0^{b_k})$, the intermediate states of S_{mid}, and the final states $\{(s_*^{b_1}, s_0^{b_2}, ..., s_0^{b_k}), (s_0^{b_1}, s_*^{b_2}, ..., s_0^{b_k}), ..., (s_0^{b_1}, s_0^{b_2}, ..., s_*^{b_k})\}$. The set S_{mid} is composed of intermediate states of the k involved CTMCs. Figure 2-e illustrates how two CTMCs are disjunctively composed.

Semantics for countermeasures. We represent counter-measuring with an unranked function $c^s(b^p, b^o)$, where $s \in \mathbb{S}$, $b^p \in \mathbb{B}^p$ and $b^o \in \mathbb{B}^o$. If we consider the proponent to be the attacker and the opponent to be the defender, this function will link the atomic attack $b^p \in \mathbb{B}^p$ with an atomic defense $b^o \in \mathbb{B}^o$. Note that besides taking as inputs atomic attacks/defenses, the function c^s can also take as inputs conjunctively, disjunctively or sequential conjunctive refined inputs.

The CTMC-semantics for a countermeasure is characterized using a new unranked function $c^s\colon \mathbb{M}\times\mathbb{M} \to \mathbb{M}$. This new function takes two CTMCs M^s and $M^{\overline{s}}$ as inputs, one representing the proponent action, and the second representing the opponent action. It links them such that they counter each other. In other words, the final state of the proponent action will be the initial state of the opponent action and vice-versa. Therefore, if the proponent starts his next step before the opponent action is executed, the proponent would skip successfully the countermeasure set by the opponent. However, if the countermeasures is successfully executed (before the proponent manages to move to next step), the proponent is brought to the initial state where he has to re-perform his action. For example, in Fig. 1, if the password changing policy is executed the attacker has to re-perform the brute force attack again.

The constructed CTMC for counter-measuring is defined as follows:

- $S^{c(M^s,M^{\overline{s}})} = S_0^{c(M^s,M^{\overline{s}})} \cup S_{mid}^{c(M^s,M^{\overline{s}})} \cup S_*^{c(M^s,M^{\overline{s}})}$,
- $S_0^{c(M^s,M^{\overline{s}})} = S_0^{M^s} \times (s_1^{M^{\overline{s}}}, s_2^{M^{\overline{s}}}, ..., s_{|S_*^{M^{\overline{s}}}|}^{M^{\overline{s}}})$ where $s_i^{M^{\overline{s}}} \in S_*^{M^{\overline{s}}}$ and $i \in \{1, ..., |S_*^{M^{\overline{s}}}|\}$,
- $S_*^{c(M^s,M^{\overline{s}})} = S_*^s \times S_0^{M^{\overline{s}}}$,
- $S_{mid}^{c(M^s,M^{\overline{s}})} = S_{mid}^{M^s} \times S_0^{M^{\overline{s}}} \quad \cup \quad S_{mid}^{M^{\overline{s}}}$.

Similarly to the other semantics, the set $S^{c(M^s,\overline{s})}$ is composed of the initial state, which contains the initial state of player (proponent/opponent) s and a tuple of all final states of player (opponent/proponent) $\overline{s}$. The final state consists of the final states of the player (proponent/opponent) and the initial state of player (opponent/proponent) $\overline{s}$. It also contains intermediate states from each player's chain. Figure 2-f shows how counter-measuring of an atomic attack with an exponential rate λ is performed against an atomic countermeasure with an exponential rate μ.

We have formulated the generator matrix $G^{c(M^s,M^{\overline{s}})}$ as follows:

$$G^{c(M^s,M^{\overline{s}})}(s_i, s_j) = \begin{cases} -\sum_{i \neq j} Q^{c(M^s,M^{\overline{s}})}(s_i, s_j) & \textit{if} \quad i = j, \\ \sum G^{M^s}(s_{i'}, s_{j'}) + \sum G^{M^{\overline{s}}}(s_{i''}, s_{j''}) & \textit{otherwise.} \end{cases} \tag{6}$$

Where $(s_{i'}, s_{j'}), (s_{i''}, s_{j''}) \in s_i \times s_j$ and $\vartheta(s_{i'}) = \vartheta(s_{j'})$ and $\vartheta(s_{i''}) = \vartheta(s_{j''})$.

Overall, this formulation consists in summing the rates of all possible transitions that go from state s_i to state s_j. Since every transition t_i^{idf} belongs to only one CTMC M_{idf}, the execution of t_i^{idf} will only affect states of M_{idf}. Therefore, there will generally be only one transition (one rate), unless it regards a disjunction of countermeasures, where the rates of the involved countermeasures are summed.

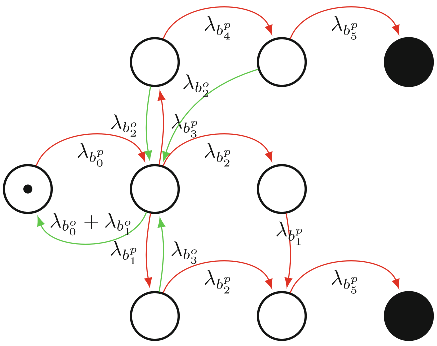

Fig. 3. CTMC obtained by composing individual CTMCs for the ADTree in Fig. 1

Example 2. We use the ADTerm given by Eq. 2 (Sect. 2.2) and apply the new semantics described above to obtain the entire CTMC (see Fig. 3) representing the whole ADTree of Fig. 1. This is merely achieved by first building CTMCs of basic events, then composing them according to the involved refinements operators that links the basic events.

4 Quantitative Security Assessment

In this section, we show how to perform quantitative analysis using Markov chains [18]. We show how to compute and extract matrices that are necessary for the analysis. To achieve this, we consider an enumerated continuous time Markov chain $M = (S, S_0, S_*, G)$, from which we can extract and compute the following structures:

The first structure is called the instantaneous probability matrix $P(t)$. It gives the instantaneous probability to transit from a state s_i to state s_j. In an other words, for each state $s_i \in S$ in a CTMC M, is associated a cumulative distribution function $0 \leq F_X(t) \leq 1$ (where X is a random variable and t is the time) that describes the probability of being in state $s_i \in S$, in time interval $[0{-}t)$, starting from state $s_j \in S$. The instantaneous probability matrix is computed using the infinitesimal generator matrix G as $P(t) = e^{G.t}$.

Application. *We exploit this matrix to draw the cumulative distribution function (CDF) of each final state (representing the final goal) starting from the initial state. Therefore, we can compute at any time the probability of success for each possible attack scenario leading to a final state (final goal).*

The second structure is the mean probability transition matrix P, where each element $P_{i,j}^{i \neq j}$ is equal to $|g_{ij}/g_{ii}|$ for $i \neq j$, and gives the mean probability to transit from state s_i to state s_j. The elements $P_{i,i}$ however, are null. In an absorbing CTMC, this matrix particularly takes a canonical form defined as:

$$P = \begin{bmatrix} Q & R \\ 0 & Id \end{bmatrix} \tag{7}$$

Here Id is the identity matrix, and 0 is the null matrix. We exploit the submatrix Q to compute the fundamental matrix N, using: $N = (1 - Q)^{-1}$, where each element $n_{i,j}$ in the fundamental matrix N gives the expected number of steps the process visited a state s_j starting from state s_i. The sum of the i^{th} row in matrix N represents the expected number of steps performed to reach any of the absorbing state starting from state s_i.

Application. *Knowing for each scenario, the set of visited states, we can use the fundamental matrix N, to compute the amount of steps performed in each scenario and hence determine the most/less probable scenario, or exert a ranking for the different possible scenarios.*

The third component is the absorbency frequency matrix $B = N \times R$, where each element b_{ij} of B gives the probability of getting absorbed by each absorbing states.

Application. *This matrix will serve to identify the most probable goal if we have many, or to confirm again the most probable scenario and with which steady state probability.*

Finally, using the fundamental Matrix N and the sojourn times $|1/g_{ii}|$ of each state, we can compute the mean time required to reach the final goal starting from the initial state.

Application. *In the context of security modeling, we compute the MTTSF (Mean Time To Security Failure) also known as MTTB (Mean Time To Breach) or MTTA (Mean Time To Attack). We can also compute the mean time for each scenario. The MTTSF is given by:*

$$MTTSF = \sum_{i \in X_t} n_{0,i} \times \frac{1}{|g_{ii}|} \tag{8}$$

Where $n_{0,i}$ is the expected number of steps performed to go from the initial state s_0 to state s_i.

5 Security Assessment of the Networked System

We report on the analysis conducted to evaluate the security of the scenario discussed in Sect. 2.1. In particular, we consider the ADTree in Fig. 1 and its CTMC representation in Fig. 3 to perform security assessment. To achieve this, we go through three cases: one (case 1), where we don't consider existence of countermeasures (attack trees), the second (case 2) we add countermeasure 'prevent target identification', and the last case (case 3), in addition to the previously added countermeasure we add two more countermeasures respectively 'Frequent patches' and 'Passwords policy'. The results of our analysis can be used by security engineers to choose appropriate defenses in order to harden the security of

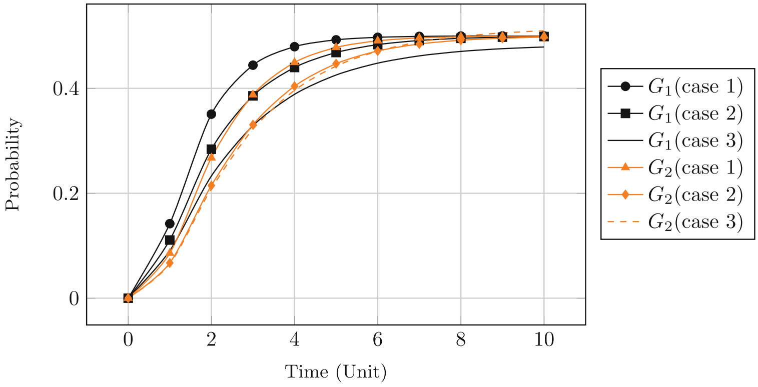

Fig. 4. Cumulative Distribution Function of scenarios group

the system. For the purpose of performing quantitative analysis, we have arbitrarily affected rational values for the different exponential rate of each basic event. Therefore, by denoting atomic attack as $b_0^p \dots b_5^p$ and countermeasures as b_0^o and b_3^o. We assign exponential rates λ as follows: $\lambda_{b_0^p} = 1$ for $b_0^p =$ Network scanning, $\lambda_{b_1^p} = 2$ for $b_1^p =$ Vulnerability scanning, $\lambda_{b_2^p} = 1$ for $b_2^p =$ Use vulnerability exploit, $\lambda_{b_5^p} = 5$ for $b_5^p =$ Execute malicious scripts. The atomic attack b_3^p=Password brute force attack use an Erlang distribution $Erl(2, 3)$, which corresponds to a sequence of two exponentials of rate $\lambda_{b_3^p} = 3$. Therefore, we model this attack with two atomic attacks b_3^p and b_4^p of the same rate equal to 3. The countermeasures $b_0^o =$ IP address randomization, $b_1^o=$ Mutable network, $b_2^o=$ Updates, and $b_3^o =$ Password policy are modeled using exponential distributions $\mu_{b_0^o} = 1$, $\mu_{b_1^o} = 2$, $\mu_{b_2^o} = 1$, and $\mu_{b_3^o} = 1$, respectively. A point that we should highlight, is that the fact of having sub-goals disjunctively refined, and at least one of the refinements is conjunctively refined, induces the replication of the final goal in the constructed Markov chain (see the two black states in Fig. 3). Therefore, there will be more than one state referencing to the same final goal but reached through different scenarios. As shown in Fig. 3, the first final state is reached through $[b_0^p; b_3^p; b_4^p; b_5^p]$. However, the second final state is reached through two different scenarios $[b_0^p; b_2^p; b_1^p; b_5^p]$ or $[b_0^p; b_1^p; b_2^p; b_5^p]$. Therefore, we can define for each final goal (final state) f_i a group of scenarios G_i composed of the same atomic attacks, but conducted in different order.

Probabilistic security attributes. We compute the probability of reaching the final goal over time expressed in terms of CDF. We also try to determine the probability of each group of scenarios, and draw the evolution of their probability of success over time. We compute the most probable scenario and perform a ranking for the possible scenarios. To achieve this, we first use the instantaneous probability matrix to draw the CDF of each final goal f_i. In our example of study, we have determined two groups of scenarios G_1 and G_2.

From Fig. 4, in the two first cases, we see that the group of scenarios G_1 is instantaneously more probable than the group of scenarios G_2. This means that the scenarios of group G_1 are more probable than scenarios of $G2$. However, they both converge to the same steady state probability of 50 %. We explain this from the fact that in case 2, the countermeasure 'prevent target identification' is applied to an atomic attack which is common to both groups G_1 and G_2. In other words, defense contributes in reducing the total probability of the goal over time (the sum of all groups CDFs) as we can see in Fig. 5. Nonetheless, in case 3, we have put more countermeasures in a way to reduce the probability of success for G_1, and we can see that the instantaneous probability of reaching the final goal through G_1 has slightly reduced for $t \geq 3$ time units to a point where G_2 becomes more probable.

In Fig. 5, we can see the impact of the countermeasures on the probability of succeeding the final goal. For instance, for a working time of [0-5] time units, the attacker has a probability to succeed of 97 % in the first case, 91 % in the second case, and finally 86 % in the last case, which is more secure. Note that

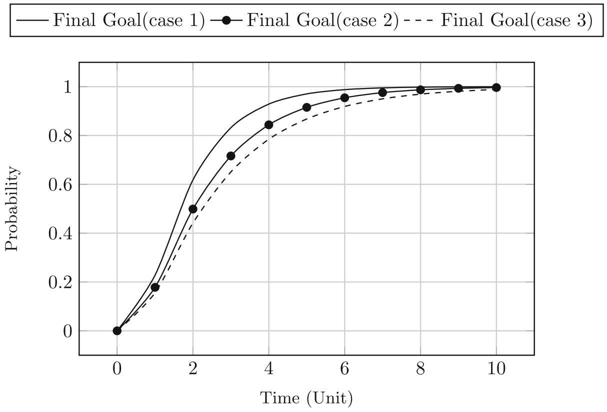

Fig. 5. Cumulative Distribution Function of the final goal

the impact is slightly small since the rates that we have affected are too close to the attacks rates.

Then we use the fundamental matrix N to compute the expected number of steps realized in each scenario of each group, and therefore determine the most probable scenario MPS (we can rank scenarios). The results are depicted in Fig. 6, where we can see that the most probable scenario has the largest amount of steps. In this case, scenario $[b_0^p; b_3^p; b_4^p; b_5^p]$ is the most probable one. Furthermore, the number of steps is increasing each time countermeasures are added. We explain that from the fact that the attacker performs more steps since the execution of a countermeasure forces the attacker to restart from his initial state. Therefore, in each scenario, the number of expected steps is increased as long as countermeasures are added. Moreover, we can rank the three scenarios as follows: $[b_0^p; b_3^p; b_4^p; b_5^p]$, then $[b_0^p; b_1^p; b_2^p; b_5^p]$, and finally scenario $[b_0^p; b_2^p; b_1^p; b_5^p]$. Finally, we use the absorbency frequency matrix B to compute the percentage in which the attacker succeed in reaching his final goal through a particular scenario in the steady state, that is to say, when he is given enough time. This will allow us to testify which group of scenarios is the most probable. The results are shown in Fig. 7 (left side).

The steady state probability is 50 % for the first two cases, where no countermeasures are applied, then when a common countermeasures is applied. The third case shows that it is more probable to perform attacks through G_2 than trough G_1, since this last one contains more countermeasures.

Time based attributes. Finally, we evaluate the mean time to breach the system in terms of MTTSF. Therefore, we make use of Eq. (5) and compute the MTTSF for the three cases. The results are illustrated in Fig. 7 (right side). We can see that the attacker is each time delayed as long as we add countermeasures. Indeed, the countermeasure 'prevent target identification' has delayed the attacker to spend 25.64 % more time units than usual (initial case). In the third

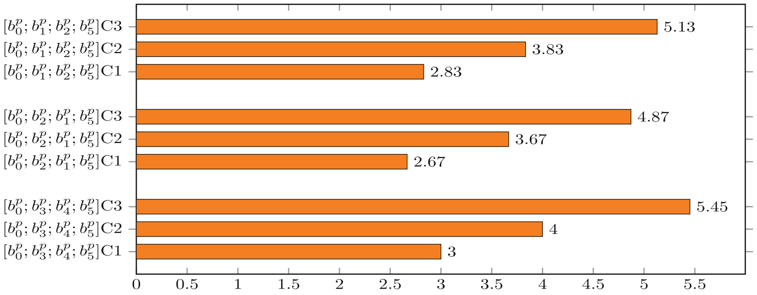

Fig. 6. Expected number of steps for each scenario and for each case

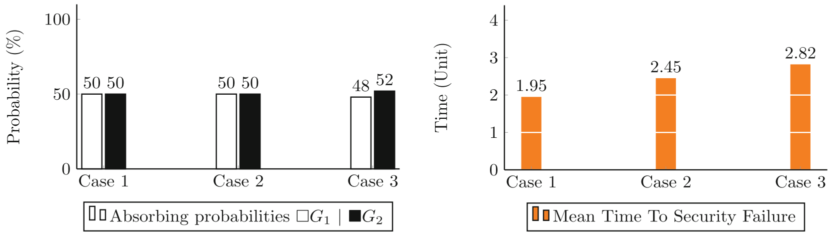

Fig. 7. Absorbing probabilities (left), Mean Time To Security Failure (right)

case, the attacker has to spend 44.62 % more time units compared to the initial non-secure case.

6 Conclusions

We presented a stochastic framework to perform quantitative analysis of ADTrees. We started by defining a new semantics for ADTrees in terms of CTMCs, then showed how to construct a final CTMC representing the entire ADTree. We then applied the analytical approach of CTMC to perform quantitative analysis. We finally demonstrated the usefulness of our solution by means of a simple but realistic example study.

As part of our future work, we will extend our framework to model and quantitatively assess complex security scenarios like social attacks. We will also extend our framework in order to embed it within the ADTool [6,8], which is a free software tool for security modeling and quantitative analysis using ADTrees.

References

1. Abraham, S., Nair, S.: Predictive cyber-security analytics framework: a non-homogenous markov model for security quantification. arXiv preprint arXiv:1501.01901 (2015)

2. Arnold, F., Guck, D., Kumar, R., Stoelinga, M.: Sequential and parallel attack tree modelling. In: Koornneef, F., Gulijk, C. (eds.) SAFECOMP 2015. LNCS, vol. 9338, pp. 291–299. Springer, Heidelberg (2015). doi:10.1007/978-3-319-24249-1_25
3. Arnold, F., Hermanns, H., Pulungan, R., Stoelinga, M.: Time-dependent analysis of attacks. In: Abadi, M., Kremer, S. (eds.) POST 2014 (ETAPS 2014). LNCS, vol. 8414, pp. 285–305. Springer, Heidelberg (2014)
4. Bagnato, A., Kordy, B., Meland, P.H., Schweitzer, P.: Attribute decoration of attack-defense trees. Int. J. Secur. Softw. Eng. **3**(2), 1–35 (2012)
5. Dalton II, G.C., Mills, R.F., Colombi, J.M., Raines, R.A.: Analyzing attack trees using generalized stochastic Petri nets. In: IEEE Information Assurance Workshop, pp. 116–123. IEEE (2006)
6. Gadyatskaya, O., Jhawar, R., Kordy, P., Lounis, K., Mauw, S., Trujillo-Rasua, R.: Attack trees for practical security assessment: ranking of attack scenarios with ADTool 2.0. In: Agha, G., Houdt, B. (eds.) QEST 2016. LNCS, vol. 9826, pp. 159–162. Springer, Heidelberg (2016). doi:10.1007/978-3-319-43425-4_10
7. Hughes, T., Sheyner, O.: Attack scenario graphs for computer network threat analysis and prediction. Complexity **9**(2), 15–18 (2003)
8. Kordy, B., Kordy, P., Mauw, S., Schweitzer, P.: ADTool: security analysis with attack-defense trees (extended version). arXiv preprint arXiv:1305.6829 (2013)
9. Kordy, B., Mauw, S., Melissen, M., Schweitzer, P.: Attack–defense trees and two-player binary zero-sum extensive form games are equivalent. In: Alpcan, T., Buttyán, L., Baras, J.S. (eds.) GameSec 2010. LNCS, vol. 6442, pp. 245–256. Springer, Heidelberg (2010)
10. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Foundations of attack–defense trees. In: Degano, P., Etalle, S., Guttman, J. (eds.) FAST 2010. LNCS, vol. 6561, pp. 80–95. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19751-2_6
11. Kordy, B., Pouly, M., Schweitzer, P.: A probabilistic framework for security scenarios with dependent actions. In: Albert, E., Sekerinski, E. (eds.) IFM 2014. LNCS, vol. 8739, pp. 256–271. Springer, Heidelberg (2014)
12. Madan, B.B., Gogeva-Popstojanova, K, Vaidyanathan, K., Trivedi, K.S.: Modeling and quantification of security attributes of software systems. In: International Conference on Dependable Systems and Networks, pp. 505–514. IEEE (2002)
13. Markov, A.: Extension of the limit theorems of probability theory to a sum of variables connected in a chain. In: Howard, R. (ed.) Dynamic Probabilistic Systems (Volume I: Markov Models), pp. 552–577. Wiley, New York (1971)
14. Mauw, S., Oostdijk, M.: Foundations of attack trees. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 186–198. Springer, Heidelberg (2006). doi:10.1007/11734727_17
15. Piètre-Cambacédès, L., Bouissou, M.: Beyond attack trees: dynamic security modeling with Boolean logic Driven Markov Processes (BDMP). In: European Dependable Computing Conference, pp. 199–208. IEEE (2010)
16. Pudar, S., Manimaran, G., Liu, C.-C.: PENET: a practical method and tool for integrated modeling of security attacks and countermeasures. Comput. Secur. **28**(8), 754–771 (2009)
17. Roy, A., Kim, D.S., Trivedi, K.S.: Attack countermeasure trees (ACT): towards unifying the constructs of attack and defense trees. Secur. Commun. Netw. **5**(8), 929–943 (2012)
18. Stewart, W.J.: Introduction to the Numerical Solutions of Markov Chains. Princeton University Press, Princeton (1994)